GNU LINUX MAGAZINE / FRANCE

France Métro : 5,95€ - DOM 6,45€ - BEL : 6,50€ - LUX : 6,50€ - PORT. CONT. : 6,50€ - CH : 12FS - CAN : 11$ - MAR : 65DH

L 19275 - 82 - F: 5,95 €

▶AVRIL ▶2006 ▶NUMÉRO 82

eCos, une autre solution libre pour les systèmes embarqués

eCos est un RTOS que l'on retrouve en concurrence avec Linux ou uClinux sur bien des architectures. Découvrez les spécificités de ce système et faites vos premiers pas sur PC en écrivant vos premiers codes pour eCos.

▶HACK/CODES

Développez pour Dreamcast

Avec un minimum d'efforts et d'investissement, apprenez à programmer pour une console 128 bits

Administration et développement sur systèmes UNIX

DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT - BACKUP

A partir du 7 avril,
459 millions de personnes
pourront acheter
votre nom de domaine en .eu.

Premier arrivé,
premier servi !

N'attendez pas ! Pour 12 € HT/AN (soit 14,35 € TTC/AN),
protégez votre nom de domaine européen.

AMEN RCS PARIS: B 421 527 797. IN WEB WE TRUST : Nous croyons au web * Voir conditions sur le site www.amen.fr. Conditions Générales de Vente sur www.amen.fr. Prix au 01/01/2006, modifiables sans préavis. Photo : Getty : Ryan McVay

Pack Serveur Privé Virtuozzo

Les avantages d'un serveur dédié au prix d'un mutualisé

- Accès ROOT ou administrateur
- 1 adresse IP
- Interface d'administration Plesk 7.5
- 400 Mo extensibles à 2 Go
- Multi-sites, Multi-domaines
- Comptes POP/IMAP illimités
- 10 bases de données MySQL
- Nombreuses applications pré-installées (Python, Webalyser, MySQL, Apache, Frontpage, Phpmyadmin...)
- Trafic illimité

10 € HT/MOIS (à partir de 10 € HT/MOIS soit 11,96 € TTC/MOIS pour un engagement annuel)

Pack Serveur Dédié Baby

Tous les atouts pour vous séduire

- Redhat/Fedora/Windows Server 2003
- Interface d'administration Webmin ou TSE
- AMD Sempron 2200 ou Duron 1600 (ext. à Athlon XP 3000)
- IDE 80 Go (ext. à 2 x 160 Go RAID1)
- RAM 256 Mo (ext. à 2 Go)
- 1 adresse IP fixe (ext. à 4)
- Trafic 1 To (ext. à illimité)

39 € HT/MOIS (à partir de 39 HT/MOIS soit 46,64 € TTC/MOIS pour un engagement semestriel)

Avec plus de **200 000 noms de domaine gérés**, AMEN est l'un des principaux fournisseurs européens de services Internet. AMEN est Registrar Officiel auprès de l'EURid (European Registry of Internet Domain Names).

amen
IN WEB WE TRUST

Compatibles & 0892 55 66 77 (0,34 € TTC/mn depuis la France 9H - 19H) www.amen.fr

EDITO

LÉGENDE :

	> LIENS <
aA	> REMARQUES <
	> ATTENTION <
	> NOTES <
	> ASTUCE <
	> ANNEXE <

« Le petit con de bois ! »

Veuillez avoir l'obligeance de m'excuser... cela m'a échappé. Il faut bien avouer que le fait de regarder LCI au petit déjeuner a parfois de quoi vous mettre de mauvaise humeur pour la journée. En particulier lorsque votre télévision diffuse le commentaire d'un artiste n'hésitant pas à comparer l'éventuelle licence globale à un droit de libre service forfaitaire dans toutes les boulangeries d'une ville. J'ai failli m'étouffer avec mon petit pain... justement.

Si même certains artistes n'arrivent pas à faire la différence entre licences, droits d'auteur et biens matériels, comment voulez-vous que le débat avance ? Pire encore, alors même que notre ministre de la Culture fait l'actualité du Web en semblant jouer à pile ou face la présence de l'article premier de DADVSI v2, on masque le plus important des problèmes : les clauses concernant les systèmes de protection. Licence globale ou non, comment légitimer le fait de taxer indistinctement les internautes pour du téléchargement s'ils ne peuvent pas lire et transformer les œuvres en raison d'une protection (MTP) qu'il est illégal de contourner ou même de simplement étudier (« décrytper » selon les textes) ?

La future réalité se présente un peu comme une publicité pour certaines cartes bancaires :

- Un CD tout neuf de mon artiste que j'aime : 17 Euro
- La licence globale sur la connexion ADSL : entre 10 et 80 Euro (option)
- La taxe sur mon CD-R de 700 Mo : 0,352 Euro
- La compréhension du système de protection et son contournement pour user de mon droit à la copie privée (que je paie avec les CD-R) : 3 750 Euro
- La modification/production d'un logiciel libre dans ce but : 300 000 Euro
- L'utilisation de ma version du logiciel libre pour ripper/encoder le CD et copier les MP3/OGG sur le CDR : 750 Euro
- La diffusion des sources modifiées du logiciel libre : 30 000 Euro

Comprendre les droits d'auteurs selon notre ministre de la Culture et contribuer aux logiciels libres, ça n'a pas de prix... heu... en fait si... mais c'est très cher.

Plus sérieusement, Monsieur le Ministre, s'il vous plaît, évaluez le dossier dans son ensemble et une fois pour toute ! Faire intervenir des experts pour étudier en détail toutes les implications sera rentable à long terme. Produire une loi brouillonne mettant en péril une partie sans cesse plus importante de l'industrie informatique n'est pas intelligent, ni pour la France, ni pour le peuple, ni pour l'industrie, ni pour la culture...

Mon calme revenant peu à peu, je vous donne rendez-vous le 29 avril pour le prochain numéro, en espérant que les choses auront un peu évolué dans le bon sens d'ici là.

Denis Bodor

PS : Aux dernières nouvelles on dit la licence globale «enterrée» car rejetée par l'Assemblée nationale. Pour ou contre ? Peu importe, le problème concerne les MTP. Ceci dit, si notre ministre est aussi doué en nécromancie qu'en prestidigitation, l'article 1er pourrait bien ressortir de sa tombe...

GNU Linux Magazine
est édité par Diamond Editions
B.P. 20142
67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail :
lecteurs@gnulinuxmag.com
Service commercial :
abo@gnulinuxmag.com
Sites : www.gnulinuxmag.com
www.ed-diamond.com

Directeur de publication :
Arnaud Metzler

Rédaction :
Rédacteur en chef :
Denis Bodor

Conception graphique :
Franck Toussaint

Responsable publicité :
Véronique Wilhelm
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Relecture :
Dominique Grosse

Impression :
IPBS

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12

Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias :
Tél. : 05 61 72 76 24

IMPRIMÉ en France
Dépôt légal :
A parution /N° ISSN : 1291-78 34
Commission Paritaire :
09 08 K78 976
Périodicité : Mensuel
Prix de vente :
5,95 €

Toute l'actu du magazine sur :
www.gnulinuxmag.com

Deb'News

Denis Bodor

EN DEUX MOTS **Comme chaque mois, voici les nouvelles de la communauté Debian, des développements et des actions en cours. Nous mettrons également ici en avant certains aspects sociaux ou techniques du projet ou tout simplement des points intéressants en rapport avec le projet.**

visual-regexp

Incontournables, indispensables, puissantes, efficaces... les expressions rationnelles (ou régulières) ont bien des qualificatifs, une fois assimilée leur syntaxe. Malheureusement, lorsqu'on fait connaissance avec ces « petites choses », d'autres qualificatifs s'imposent parfois : difficiles, obscures, cryptiques... Comme tout le monde le sait, les expressions rationnelles sont bien plus faciles à écrire qu'à lire. Pour faciliter la tâche aux utilisateurs touchant pour la première fois à ce domaine, nous avons `visual-regexp`. Cette application Tcl/Tk de Laurent Riesterer est une interface graphique permettant la composition et surtout la lecture aisée des expressions rationnelles. Elle permet également de tester l'expression sur différentes chaînes de caractères. La lecture et la compréhension des expressions sont simplifiées par la colorisation des chaînes données en exemple. `visual-regexp` est à la fois un outil pour comprendre rapidement une expression, mais également un moyen d'apprentissage efficace.

Il n'existe qu'une seule entrée non résolue dans le BTS pour ce paquet et il s'agit d'un souhait (Wishlist : nouvelle version amont).

softbeep

Bip ! Quelle tristesse ! Nos machines disposent de toutes sortes de moyens graphiques et sonores pour communiquer avec nous mais, dans la plupart des cas, lorsqu'il s'agit d'attirer notre attention, c'est un simple son qui nous prévient, émis par le haut-parleur interne de la machine (notez au passage qu'avec un noyau 2.6, il est nécessaire de charger le module `pcspkr` pour que celui-ci fonctionne). `softbeep` est un petit utilitaire bien sympathique utilisant `LD_PRELOAD` et fonctionnant ainsi entièrement dans l'espace utilisateur. Son objectif est d'intercepter différents appels dont `XBell()`, `beep()` et `gdk_bell()` ainsi que le caractère BEL. On installe l'interception en précisant au chargeur de bibliothèque que l'on souhaite charger `libsoftbeep.so` via la variable d'environnement `LD_PRELOAD`. Une autre variable d'environnement, `SB_RUN`, permet de spécifier la commande à lancer en guise de bip. Le script `sb-beep`, contenant un simple appel à `esdplay`, est utilisé par défaut. Notez que `softbeep` ne fonctionne pas avec les programmes SUID/SGID (`xterm` est SGID tout comme `aterm`). Il s'agit d'une limitation de `LD_PRELOAD` parfaitement compréhensible.

Quelques entrées existent dans le BTS pour ce paquet. Jetez-y un œil avant tout rapport de bug.

Encore des histoires de noyau

Dans GLMF 76, j'avais abordé la compilation du noyau ainsi que des modules selon « *the Debian's way* ». Il restait cependant un point spécifique qui n'avait pas fait l'objet d'explications : la modification du noyau. Bon nombre de fonctionnalités intéressantes ne sont pas directement présentes dans les sources officielles du noyau Linux.

Les versions de Linux empaquetées par Debian sont souvent très proches des noyaux officiels, comprennent toutefois quelques modifications mais pas ses fonctionnalités intéressantes. Il suffit, pour s'en convaincre, de récupérer les sources d'une image du noyau :

```
% apt-get source linux-image-2.6.15-1-686
```

On notera au passage que les sources APT de `linux-image-2.6.15-1-686` sont les mêmes que celles de `linux-source-2.6.15`. Ceci est parfaitement normal puisque dans la logique Debian, l'archive `linux-2.6_2.6.15.orig.tar.gz` à laquelle on applique le patch `linux-2.6_2.6.15-7.diff.gz` nous donne le contenu du paquet `linux-source-2.6.15`.

Celui-là même qui permettra, via des commandes comme `make-kpkg`, de produire un paquet binaire contenant l'image du noyau, les modules, etc. Notez que `linux-2.6_2.6.15.orig.tar.gz` n'est pas une archive des sources officielles Linux renommée, comme on pourrait le penser.

Pour ajouter à la confusion, je tiens à signaler l'existence des paquets `linux-patch-debian*`. Il s'agit d'une collection de patches permettant, à partir de l'archive d'origine des sources du noyau d'obtenir des sources identiques à celles présentes dans le paquet `linux-source-*`.

On notera d'ailleurs que tout ceci semble embrouiller un peu tout le monde puisque le descriptif du paquet `linux-patch-debian-2.6.15` parle, par exemple, d'une archive `linux-source-2.6.15_2.6.15.orig.tar.gz` et non de `linux-2.6_2.6.15.orig.tar.gz`.

Bref, après vous avoir inquiété, il est temps pour moi de vous rassurer un peu. Dans la quasi-totalité des manipulations concernant le noyau dans Debian, vous n'utiliserez que les paquets `linux-image*` et `linux-source*`. Inutile d'utiliser `apt-get source`, le système est relativement bien conçu et simple. Il suffit de récupérer les bons paquets et d'utiliser `apt-get install` et `dpkg -i`.

Ce qui nous concerne aujourd'hui va au-delà de la construction d'un noyau ou de la compilation de modules empaquetés.

Un certain nombre de développeurs préfèrent travailler sur une modification du noyau. Soit parce qu'ils n'ont pas le choix et qu'il s'agit d'une modification du code existant qu'il n'est pas possible d'envisager sous la forme d'un module, soit parce qu'il s'agit, tout simplement d'un choix personnel (plus rare étant donné la modularité du noyau).

Ces modifications d'un code existant prennent la forme de patches. C'est alors le seul moyen permettant d'ajouter les fonctionnalités souhaitées à un noyau.

Debian sait parfaitement gérer le type de patches sur le noyau et met à disposition de l'utilisateur un mécanisme simple permettant de modifier le source et de générer un nouveau paquet pour le noyau. Ces patches prennent la forme de paquets aisément reconnaissables par leur nom :

```
apt-cache search "^kernel-patch"
[...]
kernel-patch-relayfs - High-Speed Data Relay Filesystem
kernel-patch-2.6-gfs - Global File System - kernel patch
kernel-patch-adamantix - Kernel patches introduced in Adamantix
kernel-patch-nfs-swap - patch to linux to enable swapping over nfs
kernel-patch-quota - Netfilter QUOTA support patch
kernel-patch-time - Netfilter time match patch
kernel-patch-ttl - TTL matching and setting
kernel-patch-uml - User-mode Linux (kernel patch)
kernel-patch-xen - patch to linux for the XEN sub-arch
```

Certains de ces paquets sont spécifiques à une série du noyau (2.4, 2.6 voire 2.2). D'autres peuvent être utilisés avec toutes les versions (ou presque). Notez au passage que certains d'entre eux, comme `kernel-patch-nfs-swap` ou `kernel-patch-badram` sont, pour le moins, surprenants.

Dans tous les cas, un `apt-cache show` s'impose pour avoir toutes les informations nécessaires sur les fonctionnalités qu'apporte le patch, les outils complémentaires qu'il est recommandé d'installer et la version du noyau concernée.

Nous allons naïvement nous pencher sur un patch des plus importants : `kernel-patch-debianlogo` qui permet de remplacer Tux par un *swirl* Debian.

Après installation du paquet, on trouve un répertoire `/usr/src/kernel-patches` contenant les éléments permettant l'application du patch. Il suffit alors de se placer dans le répertoire `/usr/src/linux-source-2.6.15` et de lancer la compilation de l'image du noyau et des modules puis, la construction du paquet avec :

```
% make-kpkg --append-to-version=-glmf-k7 \
--initrd \
--revision=1.0.custom \
-added-patches=debianlogo \
kernel_image modules_image
```

L'option `--append-to-version` permet à la fois de déterminer le nom du paquet mais également le répertoire d'installation des modules dans `/lib/modules`. On remarquera l'option `-added-patches` renseignant le processus de compilation sur le patch à utiliser. Après de longues minutes d'attente, on trouve un fichier `.deb` dans `/usr/src` qu'il ne reste plus qu'à installer avec `dpkg -i`.

```
% dpkg -i linux-image-2.6.15-glmf-k7_1.0.custom_i386.deb
[...]
Dépaquetage de linux-image-2.6.15-glmf-k7
Done.
Paramétrage de linux-image-2.6.15-glmf-k7 (1.0.custom) ...
Running depmod.
Finding valid ramdisk creators.
Using mkinitramfs to build the ramdisk.
```

autossh

On ne présente plus SSH, l'une des rares méthodes sécurisées (sinon la seule) permettant d'obtenir un *shell* distant, de lancer des commandes sur un hôte ou de créer des tunnels. Ce dernier point n'est pas des moindres, car cette simple fonctionnalité permet bien des manipulations et la sécurisation d'échange se faisant normalement « en clair » sans modifications du client ou du serveur. Le problème des tunnels SSH arrive avec les déconnexions intempestives. Contrairement à des solutions VPN, `ssh` ne gère pas directement les coupures/reconnexions (ce n'est pas là son objectif principal). Heureusement pour nous, il existe `autossh`. `autossh` reprend le concept de `rstunnel` implémenté en C. Il s'agit de lancer une instance `ssh` à la place de l'utilisateur et de la surveiller. La surveillance du lien se fait via une boucle/tunnel sur un port déterminé par l'utilisateur (7 par défaut). Ainsi, toute coupure provoque une reconnexion automatique. Bien entendu, si l'authentification se fait par mot de passe, celui-ci sera à nouveau demandé à l'utilisateur. Dans le cas de tunnels, on préfèrera donc une authentification par clef non protégée par un mot de passe (avec les risques que cela comporte).

Deux entrées d'importance relative sont présentes dans le BTS pour ce paquet. Le premier problème concerne l'échec de la commande si le port de surveillance est déjà utilisé et le second concerne également un échec silencieux s'il existe déjà un lien de ce type vers la même destination.

xmove

Vous connaissez `screen`, cet utilitaire permettant de détacher un terminal d'une console ou d'un `xterm` pour ensuite le rattacher par ailleurs (depuis une session SSH par exemple) ? Et bien, `screen` a, presque, un équivalent pour X. `xmove` se lance tout simplement dans une session X existante. Dès lors et sous réserve d'avoir contrôlé l'authentification (`xhost` et `xauth`), il peut être considéré comme un display X (habituellement `:1` si la session X a créé `:0`). Si on lance ensuite une application X sur `:1`, celle-ci apparaîtra sur `:0`. Mais il suffit d'avoir accès à un autre display (même sur une autre machine) pour déplacer l'application (ou plutôt le contenu de sa ou ses fenêtres) vers celui-ci. Exemple classique pour des tests, il sera très facile de « catapulter » une application sur un serveur Xnest en `:2` en utilisant : `xmovectrl -move localhost:2 8` où `8` est le numéro de notre application, obtenu par un `xmovectrl -list`. Ainsi, avec un bon débit (ou de la patience), vous pouvez par exemple, à l'instar des utilisateurs du duo `screen`/`irssi`, ne jamais vous déconnecter de votre serveur IRC préféré.

Quelques entrées existent dans le BTS pour ce paquet. Jetez-y un œil avant tout rapport de bug.

colormake

C'est petit, c'est tout bête, mais cela améliore un peu la vie au quotidien. `colormake` ne fait pas grand-chose, il s'agit simplement d'un *wrapper* pour la commande `make` permettant de coloriser les différents messages et types de lignes affichées (avertissement, erreur, etc.). L'intérêt est bien plus que cosmétique puisque ce script Perl appelé via un script shell vous permettra de travailler plus rapidement. C'est tout simple, mais lorsqu'on n'en connaît pas l'existence...

Quelques entrées existent dans le BTS pour ce paquet. Jetez-y un œil avant tout rapport de bug

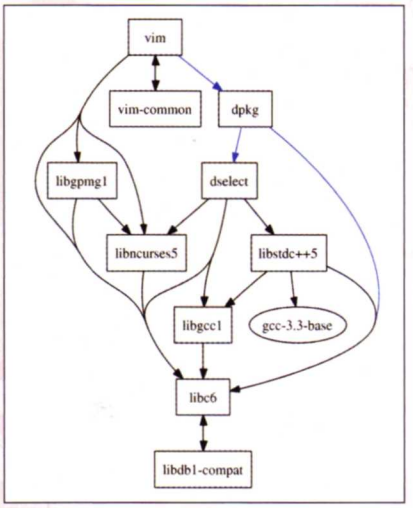
vde

vde est l'acronyme de *Virtual Distributed Ethernet*. L'objectif est de construire un réseau Ethernet virtuel en connectant des interfaces à des *switches* VDE en utilisant des liens VDE (équivalent des câbles Ethernet). Il est ainsi possible de construire entièrement un réseau Ethernet reposant sur le support TUNTAP à des fins de tests par exemple. vde vous permettra également de créer un réseau complet en utilisant plusieurs sessions Qemu sur une même machine. Tout comme avec un véritable réseau Ethernet, il est possible de connecter deux switchs VDE et ainsi de tester des infrastructures plus complexes. vde permet également de créer des tunnels et des VPN dans un but plus sécuritaire. Enfin, l'architecture très similaire à Ethernet reprenant le concept de switches et de câbles permet de rendre le réseau virtuel modulaire. On peut, en effet, changer un lien VDE exactement de la même manière qu'on change un câble Ethernet, sans avoir à revoir la configuration de l'ensemble.

Une seule entrée est présente dans le BTS, et ce, afin de préciser que vde2 est disponible en amont.

apt-rdepends

Les dépendances entre les paquets, un cauchemar pour certaines distributions, un terrain de jeux pour Debian et APT... Pour comprendre l'étendue du problème, mais également utiliser intelligemment les informations sur les dépendances, il existe une solution permettant une représentation graphique. Cette solution s'appelle apt-rdepends. Il s'agit d'une commande (et d'un paquet) permettant de lister récursivement les dépendances entre paquets. L'un des gros avantages, pour nous humains, est la possibilité de produire une sortie au format dot utilisable par la commande du même nom (paquet graphviz). Ainsi, la commande `apt-rdepends -d vim | dot -Tps > vim.ps` vous permettra de produire un graphique au format PostScript représentant les dépendances de Vim telles que, l'image ci-dessous.



Notez que c'est une très mauvaise idée que de tenter un `apt-rdepends -d -r libc6`. Lister et organiser les dépendances entre les paquets nécessitant les bibliothèques C GNU est un travail un peu trop important pour la plupart des machines...

Quelques entrées mineures existent dans le BTS pour ce paquet.

ATTENTION

Assurez-vous que tous les paquets en rapport avec la construction du nouveau noyau sont à jour et en particulier kernel-package et initramfs-tools. Dans le cas contraire, vous risquez de vous retrouver avec un paquet qui ne génèrera pas d'image initramfs pour votre noyau à l'installation. Idem si vous oubliez l'option --initrd.

De 2.6.12 à 2.6.15

La mise à jour du noyau de la version 2.6.12 à 2.6.15, en plus de nécessiter un malheureux redémarrage du système, impose quelques changements de taille. Tout d'abord, cette mise à jour marque le passage d'image initrd à initramfs (voir article RIM-Linux dans GLMF 80). Rien de bien grave ici. Il s'agit d'une simple évolution vers un système d'image système en mémoire bien plus souple que le classique RAMdisk populaire jusqu'alors. Un autre changement majeur concerne l'utilisation de udev. Il s'agit du remplaçant de devfs, une solution développée pour les noyaux 2.4.x qui, malheureusement, posait bien plus de problèmes qu'elle ne fournissait de solutions. udev permet la création « à la volée» des entrées dans /dev qui, dès lors, est un système de fichiers tmpfs monté au démarrage. Par opposition à l'ancien devfs, la totalité du code d'udev se trouve dans l'espace utilisateur. C'est udevd qui, placé en mémoire, reçoit les évènements permettant l'ajout d'entrées dans /dev et la mise en place des permissions adéquates.

Pour l'anecdote, le changement est particulièrement important si, comme moi, vous utilisez un code accédant, par exemple, à /dev/parport0. Le chargement manuel du module ppdev est alors nécessaire pour créer l'entrée. Dans le cas contraire, votre code ne pourra accéder au port parallèle depuis l'espace utilisateur et il ne fonctionnera pas. Le problème a également été rapporté concernant les outils SANE pour les scanners et tous les utilitaires reposant sur la libieee1284. Enfin, le support ALSA pour les périphériques audio est presque maintenant incontournable. Il n'existe aujourd'hui, bien entendu, aucune raison de préférer OSS si ce n'est un certain laxisme dans les mises à jour. L'utilisation du démon udevd chargera les pilotes ALSA de la carte son détectée. Il vous faudra donc vous pencher sur la configuration du système ALSA ou plus simplement utiliser alsaconf.

Denis Bodor,

lefinnois@lefinnois.net, db@ed-diamond.com

Interview de Allison Randal

Entretien et traduction réalisés par Philippe Bruhat.

EN DEUX MOTS **Cette interview a été réalisée pendant OSCON Europe, le 18 octobre 2005, au NH Grand Hotel Krasnapolsky à Amsterdam. Allison Randal était la présidente de la Perl Foundation jusqu'à la veille de l'interview (Bill Odom ayant été élu à ce poste quelques jours auparavant). Elle travaille pour O'Reilly Media et fait partie de l'équipe de conception de Perl 6.**

Le logo du défunt Perl Institute

GLMF : Qu'est-ce que la Fondation Perl (Perl Foundation) ? Quel est son rôle ?

Allison : La Fondation Perl soutient le développement de Perl. Elle ne dirige pas le développement de Perl.

Le développement de Perl a sa propre structure de direction. TPF (*The Perl Foundation*) fait des choses comme le soutien de conférences Perl, la publicité autour de Perl, le soutien de groupes d'utilisateurs et l'implication dans la communauté.

Les *Perl Mongers* et *Perl Monks* font partie de TPF.

La Fondation Perl se charge aussi de maintenir les services de perl.org : listes de diffusion, serveurs web, dépôts de contrôle de version, etc.

Ainsi, quand nous avons besoin de fonds pour acheter du matériel neuf ou des choses de ce genre, cela passe également par la Fondation Perl.

GLMF : Et l'Institut Perl (Perl Institute) était différent ?

Allison : L'Institut Perl était une sorte de première tentative. Si on regarde en arrière, il y a eu un petit scandale financier, aussi l'Institut Perl a-t-il été complètement fermé. Nous avons fait trop confiance à quelqu'un qui ne méritait pas vraiment notre confiance.

Un peu plus tard, de façon complètement indépendante *Yet Another Society* (encore une autre association) a été lancée pour soutenir les conférences YAPC (*Yet Another Perl Conference*).

Au fil du temps, *Yet Another Society* a pris comme autre nom *The Perl Foundation*, car les gens ne comprenaient pas vraiment d'après son nom ce que *Yet Another Society* était supposée faire.

GLMF : Je me souviens vaguement qu'à une époque YAS est devenue la Fondation Perl...

Allison : Oui, c'était un changement de nom graduel...

GLMF : Et maintenant tout le monde a oublié.

Allison : Légalement, c'est toujours YAS dans les documents officiels. Mais elle porte aussi légalement le nom commercial de *The Perl Foundation*.

GLMF : Et où va-t-elle ? Vers... Perl 6 ?

Allison : Pas complètement. En fait, son rôle est de soutenir le développement de Perl dans tous les domaines, donc Perl 6 en fait partie.

Mais une plus grande part de cela est pour le moment Perl 5, en réalité. Faire connaître Perl aux particuliers et aux entreprises, et s'assurer que Perl 5 reçoit les ajouts dont il a besoin et que la maintenance continue de fonctionner correctement. Ce sont tous des buts importants.

GLMF : Et qu'en est-il du logo ? Personnellement, je l'ai vu comme une façon de devenir plus indépendant des contraintes légales liées à O'Reilly et du fait que partout où vous utilisez le chameau (NDLR : qui est un dromadaire), vous devez indiquer le fait que celui-ci est une marque déposée de O'Reilly. Y a-t-il autre chose ?

Allison : Eh bien, c'est un effet secondaire appréciable. Le point principal est qu'à cause du lien avec la marque déposée, il y a quelques endroits où nous ne pouvons *vraiment* pas du tout utiliser le chameau.

Par exemple, si nous publiions une collection de documentation Perl ou quelque chose comme ça, nous ne pourrions vraiment pas utiliser le chameau dans ce contexte. Parce que c'est le contexte pour lequel O'Reilly le protège.

Aussi, nous avons commencé simplement pour nous donner une autre possibilité dans quelques cas.

Mais comme nous avançons, il semble avoir pris une vie propre. On l'a vu apparaître sur des T-shirts YAPC et des choses comme ça.

GLMF : C'est un logo sympa, parce que l'oignon fait partie des présentations de Larry Wall depuis un moment (NDLR : Les fameux State of the Onion), donc il fait déjà partie de la culture Perl.

Allison : Oui. Nous avons passé beaucoup de temps à chercher le bon logo. Nous avons essayé beaucoup de choses différentes. Au tout début, nous pensions « Pas un oignon, ça ne fait pas un très bon logo ».

Nous avons cherché autre chose, mais nous revenions toujours à l'oignon, à cause de ce lien avec l'histoire de Perl. Puis nous avons trouvé un artiste qui a fait un oignon que nous aimions, alors nous sommes partis dessus.

GLMF : Qui est l'auteur de l'oignon ?

Allison : Il s'appelle Devin Shane Muldoon (http://devinshane.com/). C'est un artiste de la région de San Francisco. Il a beaucoup de talent.

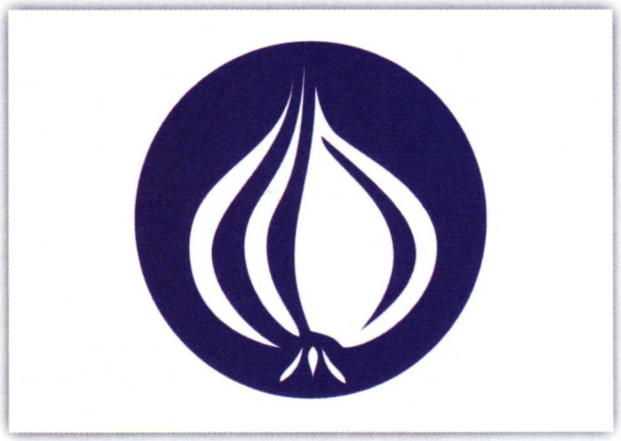
GLMF : Tu es très impliquée dans le développement de Perl 6, tu fais partie de l'équipe de développement... Est-ce que tu peux nous donner des dates ?

Allison : Pour la livraison ? Non, c'est exprès. Cela fait un moment maintenant que nous n'avons intentionnellement pas de date. Pour commencer, c'est très décourageant de glisser sans cesse sur les dates. (rires)

Mais aussi, Perl 5 fonctionne bien maintenant et il y a beaucoup de sociétés qui en dépendent ; il est donc important d'avoir tous les éléments en place avant d'annoncer Perl 6.

Il est très important d'avoir une stratégie de migration propre, afin qu'il n'y ait pas de sociétés qui l'essayent et qui disent ensuite : « Oh tant pis, il n'est pas prêt, il n'est pas stable. Laissez tomber, nous ne l'utiliserons jamais. ».

Il vaut mieux simplement continuer et les laisser utiliser Perl 5 aussi longtemps qu'elles en sont satisfaites, puis leur présenter un Perl 6 stable, vers lequel elles peuvent migrer et utiliser tout leur code Perl 5 dessus.



Le logo de la Perl Foundation

GLMF : J'ai discuté précédemment de Pugs avec Damian Conway (cf GNU/Linux Magazine 81, mars 2006). Et j'ai eu l'impression que, grâce à Pugs, Perl 6 sera en réalité Perl 6.1. Parce que la plupart des problèmes que les concepteurs n'auraient peut-être pas pu prévoir auront été vus, parce que les gens vont pouvoir bricoler avec Perl 6 avant qu'il soit effectivement sorti.

Allison : Oui. Pugs a vraiment été très bénéfique pour Perl 6 dans son ensemble. D'une part parce qu'il a permis de jouer avec la sémantique de Perl 6 et de travailler à l'implémenter, mais aussi parce qu'il y a des gens qui commencent à porter des modules dessus.

Avoir un CPAN qui marche pour Perl 6 quand on le livrera est très important pour tout le processus de migration. C'est vraiment crucial.

GLMF : Est-ce que le CPAN de Perl 6 sera différent du CPAN de Perl 5 ?

Allison : Il sera très similaire. Tu verras des modules qui seront portés à l'identique : un portage direct avec la même interface, le même nom et tout.

Tu verras aussi d'autres modules qui ne seront pas portés du tout, car on n'en aura plus besoin. Je pense que beaucoup des modules `Class::` disparaîtront en cours de route.

Et il y aura aussi de nouveaux modules, de nouvelles idées qui apparaîtront à cause de Perl 6.

GLMF : Tous les filtres de source vont disparaître.

Allison : Oui. Ou ils pourraient bien arriver avec le même nom, mais au lieu d'être des filtres de source, ils seront implémentés comme des modifications de la grammaire.

GLMF : Et est-ce qu'on aura un site web différent, ou bien est-ce que les modules Perl 6 seront mélangés aux modules Perl 5 ?

Allison : Nous avons beaucoup parlé de ça, en particulier avec Jarkko (NDT : le *Master Librarian* du CPAN), et l'idée générale est de conserver cpan.org, mais peut-être d'ajouter quelque chose au bout, d'utiliser cpan.org/perl6 ou quelque chose comme ça

pour les modules Perl 6. Et rendre le CPAN.pm livré avec Perl 6 assez intelligent pour savoir où aller. Le CPAN.pm de Perl 5 n'est pas assez malin pour différencier plusieurs dépôts (si on veut les appeler comme ça), mais le CPAN.pm de Perl 6 sera assez malin pour cela. Et comme Perl 6 devra être capable de faire tourner des modules Perl 5 et des modules Perl 6, il faudra être capable d'utiliser l'un ou l'autre et de différencier intelligemment les deux.

GLMF : Hormis la transition, c'est important de supporter les modules Perl 5 ?

Allison : Ce n'est vraiment important que pour la transition. Mais si tu te rappelles du fait que, à certains endroits, les gens utilisent encore Perl 5.005_03 (NDT : La version de mars 1999), alors tu vois que la transition pourrait durer assez longtemps. Il pourrait donc être important d'être capable de faire marcher des modules Perl 5 pendant 5 ou 6 ans [après la publication de Perl 6].

GLMF : En plus de faire partie de l'équipe de conception de Perl 6, d'être à la tête de la Perl Foundation, du moins jusqu'à hier...

Allison : Jusqu'à hier, oui ! (rires)

GLMF : ... tu travailles également pour O'Reilly. Tu es chargée de l'édition de plusieurs livres.

Allison : Principalement des livres de Perl, mais également des livres sur PHP, Ruby et quelques livres sur la sécurité.

GLMF : En quoi consiste ton travail ?

Allison : Chez O'Reilly, le travail d'un rédacteur (*editor*) couvre l'ensemble d'un projet du début à la fin. Ça commence donc par la recherche d'une idée, la recherche des auteurs, la sélection des auteurs, ensuite il faut travailler sur les manuscrits qui arrivent, superviser les relecteurs techniques de la communauté. Et enfin, réviser le livre qui entre en production, les pré-impressions et les épreuves, pour vérifier que le résultat est vraiment bon avant qu'il ne soit livré.

GLMF : Qu'est-ce que tu attends de cet OSCON, en tant qu'employée d'O'Reilly, mais aussi en tant que membre de la communauté Perl ?

Allison : En tant qu'employée d'O'Reilly, c'est une bonne occasion de rassembler des personnes qui *font* des choses, de les faire se parler. Beaucoup de nouvelles idées évoluent

Allison Randal et Autrijus Tang
Photo James Duncan Davidson/O'Reilly Media

à partir de conférences comme celle-ci. Pas nécessairement parce que quelque chose d'énorme a été inventé ici, mais plutôt parce que tu vas avoir deux personnes qui travaillent toutes deux sur des choses différentes, et d'un seul coup elles vont se rendre compte comment ces deux parties sont bien adaptées l'une à l'autre, et tu vas obtenir quelque chose de complètement nouveau.

GLMF : Et tu penses que ça se passe plutôt dans les salles de conférences, dans les couloirs et les halls d'exhibition, ou pendant les BOF sessions ?

Allison : Les trois. Mais de manière différente à chaque fois. Les sessions de conférence représentent une opportunité pour une personne d'entendre des détails au sujet de l'idée d'une autre personne. Et parfois cela donne des idées à la personne qui écoute et elle peut se dire : « Oh, voilà l'autre moitié de ce qui me manquait » ou « Je n'avais jamais pensé à essayer ça ». Les discussions de couloirs ou dans les halls d'exhibition sont extrêmement enrichissantes du point de vue des interactions entre les modes de pensée des participants.

GLMF : Elles ont parfois pour origine les sessions elles-mêmes.

Allison : Oui, exactement. Et parfois il y a comme un effet de rétroaction. Au fur et à mesure de la conférence, tu verras des personnes dont la présentation aura été effectivement modifiée par les conversations qu'elles ont eu dans les couloirs la veille. Les BOF sont une espèce de croisement entre les deux. Elles sont à la fois interactives comme les discussions de couloir, et elles représentent aussi une possibilité d'obtenir plus de détails, comme lors d'une présentation.

GLMF : Et en tant que membre de la communauté Perl ?

Allison : Pour commencer, les conférences sont à peu près ma seule chance de voir mes amis ! (rires)

GLMF : Il en faudrait plus !

Allison : Oui ! Plus ! Je vais passer tout mon temps à voyager ! Ceci dit, c'est la même chose, de nouvelles idées naissent. C'est aussi une occasion pour la communauté de se mettre au courant de ce qui se passe. Nous avons des sources

d'informations, use.perl.org et perlmonks.org et ce genre de choses, mais il semble que les conférences soient le moment où les gros échanges d'idées se font dans la communauté.

GLMF : En tant que membre de la communauté Perl moi-même, j'avais un peu peur, bien que le but général soit que les gens se rencontrent et échangent des idées, que la bande des Perleurs reste dans son coin. Il y aurait des échanges d'idées, mais à l'intérieur de la communauté Perl. Peut-être que c'est de la fierté (hubris) : très peu d'échanges avec les autres groupes de gens. Je ne sais pas si c'est le cas.
Je me demande si la plupart des membres de la communauté Perl étaient en fait aux conférences Perl, même s'ils les avaient déjà vues ailleurs.

Allison : Oui, c'est difficile. Ce n'est pas unique à notre communauté, car beaucoup de communautés tendent à se retrouver en groupes de gens qui se ressemblent. Mais quand même, il est à peu près inévitable d'avoir des échanges d'idées dans les couloirs et les exhibitions.

GLMF : Les geeks sont un peu timides, aussi. Ils ne vont pas aller voir quelqu'un et lui dire « Bonjour, je ne te connais pas, qu'est-ce que tu fais ? ». Bien sûr, il y a les BOF sessions, qui sont une bonne manière de faire ça, car il n'y a pas de présentateur officiel et que les gens sont supposés parler entre eux. Mais sinon, je n'ai aucune idée de la façon dont ça se passe.

Allison : Peut-être que nous avons besoin de plus de choses comme le jeu *Werewolf* pour mélanger les communautés. (NDT : Voir le compte-rendu d'OSCON Europe dans GLMF 78) « OK tout le monde, il nous faut dix villageois ! Une personne Perl, une personne Ruby, allez, asseyez-vous, et jouez à ce jeu ! » (rires)

ERRATUM

Dans GLMF 81, page 21 en haut à droite, il fallait lire «Et nous aurons besoin de langages pour...» au lieu de «Et nous aurons besoin de langages pourris...»

Cette erreur est entièrement de ma faute et elle a été scrupuleusement copiée lors de la mise en page du magazine.

Voici l'explication. Je suis un utilisateur de Vim, ce qui a beaucoup d'avantages et un inconvénient : quand j'oublie que je suis déjà en mode insertion, des « i » surnuméraires viennent s'ajouter à ma prose. (Je connais un docteur en physique qui m'a avoué que sa thèse est truffée de « i » surnuméraires pour la même raison.)

C'est ainsi que ma question est devenue dans le fichier source de l'entretien :

«Et nous aurons besoin de langages pouri...»

Hélas, mon collègue Mongueur qui a relu ce passage est un utilisateur d'Emacs. Il n'a donc pas reconnu la coquille due à Vim, et a corrigé l'orthographe, altérant ainsi le sens de ma question. N'ayant pas relu attentivement sa relecture, je suis le seul fautif et présente cette anecdote (culturelle) accompagnée de mes excuses aux lecteurs de Linux Magazine et à Damian Conway, pour avoir involontairement déformé ses propos.

→ Utilisation de RT, impact organisationnel, extensions de RT

Nicolas Chuche, Laurent Gautrot et Jérôme Fenal

EN DEUX MOTS **Après avoir vu la mise en œuvre technique de RT, voyons maintenant l'utilisation de l'outil, mais du point de vue fonctionnel, et par là même, organisationnel.**

Utilisation

Comme tout outil, RT [1] nécessite donc une procédure d'utilisation à laquelle les utilisateurs devront adhérer. Cette procédure relève de l'organisation interne mais a une forte incidence sur le bon usage de l'outil.

Cette partie n'est donc pas tant technique qu'organisationnelle. Mais elle n'en est pas moins importante comme nous allons le voir dans deux exemples de mise en œuvre qui, s'ils se ressemblent, ressemblent surtout aux organisations dans lesquelles RT s'insère.

Faire passer les demandes par RT

La première condition à l'utilisation de RT est... son utilisation (j'adore enfoncer des portes ouvertes). Ce qui signifie que les équipes intervenantes devront *a minima* créer les tickets au fur et à mesure des coups de fils des demandeurs. Mais le plus simple reste de leur demander (soit directement, soit par le biais de leur hiérarchie), de passer par RT en envoyant un mail à la bonne adresse.

Le bon argument pour vendre RT aux demandeurs potentiels est le fait qu'en passant par le mail, ils sont sûrs que leur demande sera prise en compte à partir du moment où ils réceptionnent l'accusé de réception de la demande.

Une fois que le message est passé, à savoir utiliser RT par le mail, l'autre point est la diffusion des adresses. Une page web est intéressante, mais dans une entité importante, cela suppose que cette même page soit connue de tous. L'avantage de la page web est que l'on pourra aussi y détailler, dans le cas d'un RT multi-entrée, les cas d'utilisation d'une file ou d'une autre.

Mais la solution reste de se faire renseigner dans l'annuaire de la messagerie de l'entité. Ce qui peut poser problème, car entrées d'annuaire et adresses sur la susdite messagerie sont souvent liées. Et l'avantage d'avoir obtenu une délégation de sous-domaines de messagerie peut s'effacer devant la nécessité de s'intégrer à l'annuaire. Pas de solution miracle, mais un éventuel problème d'œuf et de poule auquel songer à la mise en place d'un serveur RT.

De l'intérêt de bien utiliser RT

Rappelons le but recherché : voir ce qu'il y a à faire, éviter le travail en double, historisation et capitalisation, vue de la charge de travail courante, voire statistiques à plus long terme.

Il est donc important que les intervenants s'attachent autant que faire se peut à suivre ce qui suit.

Exemple d'utilisation : un environnement centralisé

Contexte et paramétrage associé

RT peut être paramétré de nombreuses façons : depuis une file unique jusqu'à de nombreuses files réparties par équipes et par typologies d'évènements traités.

Le plus difficile est sans doute de trouver un juste milieu. Compte tenu des contraintes techniques inhérentes, la séparation minimum à faire est technique : une file par comportement (*workflow*, droit...) à mettre en place est le minimum.

Mélanger dans une seule file sera sans doute aussi ingérable au final que d'avoir trop de files différentes entre lesquelles on devra jongler. Il faudra donc trouver le bon compromis.

Un exemple de paramétrage peut être :

- **Réception des demandes extérieures**

Une file unique pour la réception des demandes d'utilisateurs. Les demandes liées à un incident ou à un problème sont déplacées dans la file « incidents et problèmes ».

Configuration spécifique :

- *Scrip* prévenant le demandeur que sa demande est bien prise en compte à l'arrivée d'un nouveau mail ;
- Scrip envoyant un mail pour indiquer qu'un ticket a été clos ;
- Un champ personnalisé indiquant le type d'intervention (mise à jour majeur, mineure, intervention imprévue, ...)
- Un champ personnalisé indiquant l'application/le projet à facturer

- **Incidents et problèmes**

Une file pour la gestion et le suivi des incidents/problèmes.

Avoir une file à part permet de bien séparer les demandes de travaux des demandes liées à un incident. On pourra également y attribuer des mots-clefs spécifiques.

Configuration spécifique :

- Scrip : les mêmes que ci-dessus ;
- Un champ personnalisé décrivant d'où semble venir

l'incident (OS, SGBD, Réseau, Logiciel, Application interne) ;

- Un champ personnalisé listant les applications touchées.

Les adresses mails déclarées pour cette file sont les mêmes que celles de la première file. Cela garantit que personne ne peut faire de demande initiale directement dans cette file, les demandes arriveront forcément dans la file de réception.

Cette file sera une candidate idéale pour l'utilisation de RTFM, une extension décrite plus loin dans l'article.

- **Les alertes automatiques**

Une file dédiée aux alertes mails automatiques (*cron*, outil de supervision...). Cette file permettra de voir tous les mails et la notion de résolution sera synonyme d'acquittement.

Configuration spécifique : cette file ne doit envoyer aucun mail.

- **Gestion de tâches**

Une file pour la gestion des tâches internes à l'équipe. Cette file permet de gérer au quotidien les projets internes de l'équipe. Elle relève de la même logique que tous les *todos* du marché.

Flux de travaux (*workflow*)

Une cinématique possible dans le cadre d'un RT simple est :

- À la réception d'un ticket, l'opérateur en charge de la surveillance de RT attribue le ticket à la personne compétente (cela peut être lui-même) et, le cas échéant, le déplace dans la file qui lui semble adéquate (incident, SGBD, etc.).
- Au moment de traiter la demande, la personne compétente ouvre le ticket ;
- Toutes les recherches, diagnostics, manipulations faites sont ajoutées en tant que commentaire afin de permettre un suivi de l'intervention ;
- Toute la correspondance avec l'utilisateur est faite avec la fonction « répondre » et, quand le responsable du ticket attend une réponse, il passe le ticket en mode bloqué. Le ticket repassera automatiquement en mode ouvert sur réception d'un nouveau mail ;
- Une fois la demande réalisée, le responsable répond au demandeur en lui demandant de valider l'intervention. Il passe à nouveau le ticket en mode bloqué ;
- Si le demandeur valide l'intervention, le responsable résout le ticket.
- Régulièrement, les tickets bloqués depuis plus d'un certain temps (un mois paraît un bon délai) sont recherchés et résolus d'office avec un petit message aux demandeurs : « sans nouvelle de votre part depuis 1 mois, nous fermons votre ticket ».

L'état bloqué peut aussi être utilisé uniquement pour les tickets dont le temps de latence (depuis le dernier échange ou commentaire) s'allonge. De même, il est tout aussi concevable de fermer un ticket par défaut, sachant que le mécanisme de base de RT est de rouvrir le ticket sur réception d'une réponse sur un ticket résolu. Cela s'accompagnera aussi d'un message signifiant la possibilité de réouverture du ticket si la résolution n'est pas satisfaisante.

Exemple d'utilisation : des équipes connexes

Contexte

Nous allons voir une organisation plus simple, s'appliquant par exemple à des équipes techniques pouvant avoir à faire entre elles en plus de clients extérieurs (exploitation, voire utilisateurs).

C'est aussi une organisation qui permet de démarrer rapidement sans trop se poser de question, de façon à appréhender l'outil. Au fur et à mesure de la prise en main de l'outil, et de l'adhésion des personnes l'utilisant, une réorganisation vers un guichet unique pourra ensuite se faire.

Ce qu'on veut donc mettre en place permet de déployer RT pour des équipes de façon relativement autonomes entre elles, au moins au départ. Cette façon de faire permet de déployer RT là où aucun outil n'est mis en œuvre.

Nous avons donc plusieurs équipes, de plusieurs disciplines, travaillant éventuellement ensemble : OS, SGBD, intégration applicative, supervision, etc.

Les tickets sont ouverts par les membres et les responsables des équipes, et uniquement pour des tâches de type projet. Cela suppose une certaine autonomie dans l'utilisation de RT, ainsi que vous avez pu le voir au travers du grand nombre de droits donnés.

Configuration

La configuration de chaque file est relativement indépendante, a son adresse mail, et chaque file a son groupe associé, comme présenté dans l'article précédent. La chose à retenir au départ, c'est « une file = un groupe » (deux si vous voulez séparer les *spammés* des personnes exemptes de notifications sur les tickets, un *über-chef* d'équipe, par exemple).

Flux de travaux (workflow)

Les tickets arrivent donc, et sont pris au fur et à mesure par les membres de l'équipe. Il peut éventuellement y avoir des assignations entre membres, de façon à limiter autant que faire se peut les tickets non assignés.

Dès que le travail commence sur un ticket, celui-ci est ouvert par l'intervenant.

À chaque étape dans la réalisation de la demande, des commentaires sont écrits dans

le ticket avec les temps de travail afférents, des correspondances ont lieu entre demandeur et intervenant. Le but est d'avoir le maximum de renseignements sur la demande afin de pouvoir la documenter.

La réponse finale prévient obligatoirement le demandeur de la nature de la résolution de sa demande et de sa clôture. Si besoin était, le comportement par défaut de RT est de rouvrir la demande sur réception d'un courrier, ce qui permet de reprendre le ticket.

S'il est besoin de transférer un ticket entre équipes, comme par exemple des allers-retours entre stockage et système, il est nécessaire de mettre en œuvre deux mécanismes : donner le droit aux intervenants de ces files de voir les files associées, afin de pouvoir transférer le ticket, et mettre en place un scrip d'alerte sur le changement de file d'un ticket, afin que les intervenants de la nouvelle file soient prévenus de l'arrivée du ticket.

Un problème par rapport à un guichet unique est l'absence de responsabilité bien définie sur les tickets non directement traitables. Nécessitant des requalifications, il convient aussi de définir le traitement qui leur sera donné : poubellisation ou non. Mais cela reste un problème d'organisation pure.

D'autre part, cela nécessite que les demandeurs connaissent exactement les attributions des différentes équipes, et leur adressent des demandes de travaux précises, correspondant à leurs compétences. Les demandes génériques, adressées à plus d'une file s'exposeront à des pertes d'information. En effet, RT considérera qu'un seul mail à plusieurs adresses constitue une erreur, et le ticket sera créé dans une seule file.

De l'importance de l'encadrement

De par l'absence d'un premier niveau qui relancera les demandes oubliées, il conviendra d'avoir équipe par équipe un responsable de l'avancement des tickets, de façon à ne pas laisser s'accumuler les tickets non encore ouverts, ou ouverts, résolus, mais non notifiés comme tels.

Les évolutions

Il se peut qu'au cours de la vie de votre RT vous vouliez changer la façon dont sont faites certaines choses. Un exemple en est la file pour l'équipe gérant le stockage mutualisé de l'entité, dont les demandes arrivant de l'extérieur ne sont pas pertinentes, ni suffisamment qualifiées. Elles devraient idéalement rester confinées aux seules équipes système et stockage. Mais on ne le sait pas forcément au début...

Après quelques allers-retours sur chaque demande, et plutôt que de demander aux clients de faire leurs demandes d'espace disque directement à l'équipe stockage, on se rend compte qu'il revient plutôt aux différentes équipes système de prendre en compte ces types de demandes, de composer avec les réserves déjà attribuées aux serveurs, et seulement lorsque celles-ci ne suffisent pas, de demander l'attribution de LUN supplémentaires.

Cela passe donc par à la fois un changement organisationnel (qui enregistre la demande, et donc qui est en mesure de facturer), un changement dans le workflow (qui fait la demande au stockage), et seulement après par un changement de configuration de l'outil. Cela impose donc d'interdire la création de ticket dans la file de l'équipe Stockage par quiconque n'appartenant pas aux équipes Unix, Mainframe ou Windows. Quand vous en arriverez à ce point, allez jeter un coup d'œil à `RTx::RightsMatrix` dont on parle plus bas, ça vous aidera ;-).

Exploitation de RT

Comme toute application, il est nécessaire de mettre en place un cadre d'exploitation de cette application. Ce n'est pas parce que cette application n'est utilisée (apparemment) que par les gens de la production qu'elle ne doit pas rentrer dans le cycle de vie normal d'une application au sein de votre système d'information.

Nous avons déjà vu l'administration applicative, ce qui suppose que vous avez une file **Support** (la file par défaut **General** rebaptisée ?) avec des gens derrière. En effet, il y aura toujours à prendre en compte les départs et les arrivées de personnels dans les services intervenant sur les files, voire des créations de files (comme par exemple une file par application du SI), bref, il faut gérer RT.

Mais après cette administration applicative, il ne faudra pas non plus oublier l'administration du socle technique. Ce que nous vous proposons de voir rapidement.

Sauvegarde, restauration, migration de base

La sauvegarde/restauration de RT tient en trois points : RT et sa configuration (`RT_SiteConfig.pm`), les personnalisations de l'interface (arborescence `/opt/rt3/local`), et le contenu de la base de données.

En ce qui concerne la base de données, ne l'ayant utilisé qu'avec MySQL, pas de gros souci à se faire. L'utilisation régulière de `mysqldump` fera très bien l'affaire.

Faites cependant attention si vous avez à passer d'un serveur MySQL 4.0 à 4.1 ou supérieur : certaines colonnes de certaines tables, comme la table Links, ont des longueurs importantes (240 caractères). Or comme MySQL 4.1 gère maintenant les encodages, et que souvent l'encodage par défaut de nos distributions est UTF-8, la tentation est grande de créer la base en UTF-8. Erreur. Non seulement la taille maximale d'un enregistrement en InnoDB (le type de stockage demandé par RT) est limitée à 1024 octets (et avec la taille maximale

d'un caractère UTF-8 qui peut aller jusqu'à 4 octets, je vous laisse faire le calcul), mais en plus il y a des effets de bord à la sauvegarde/restauration.

Donc si vous devez créer la base de RT à la main, **créez-la avec l'encodage ISO-8859-1**, voire sans spécifier d'encodage (ce qui revient à l'avoir créé en ISO-8859-1). Un simple `create database rt3` suffit donc.

Vous vous exposez à deux problèmes :

- Perdre de l'information à la restauration, comme le contenu des pièces jointes qui sera altéré de façon définitive, ou comme la perte des accents ;
- Perdre le contenu des pièces jointes dès leur insertion soit par messagerie, soit par l'interface. Là, vous êtes sûr d'avoir vos utilisateurs sur le dos...

Vous êtes prévenus.

Mise à jour de RT

Au sein de la version 3

Passons rapidement sur les migrations entre versions de RT. Entre versions mineures au sein de la version 3 de RT, les migrations sont relativement simples. Il suffit effectivement de mettre à jour les modules Perl de CPAN le nécessitant (comme `DBIx::SearchBuilder`, publié par Jesse et qui semble avoir RT comme seule dépendance), de mettre à jour RT lui-même (par `make upgrade`, et en vérifiant les surcharges locales), et enfin de mettre à jour le schéma de la base de données via les fichiers SQL correspondants dans `rt-x.y.z/etc/upgrade/version-seuil/`.

Le seul inconvénient est que cette mise à jour n'est pas automatisée.

Encore que... des procédures de mise à jour sont fournies pour vous assister. Consultez les fichiers `UPGRADING` et `README` de la distribution sur la marche à suivre.

Pour corriger, si besoin est, la base de données de manière incrémentale en répétant pour chaque version inférieure à la version, vous pouvez utiliser quelque chose comme :

```
sbin/rt-setup-database \ --action schema \ --datadir
etc/upgrade/<version>
/rt/rt-3.4.2/sbin/rt-setup-database \ --action acl \ --datadir
etc/upgrade/<version>
/rt/rt-3.4.2/sbin/rt-setup-database \ --action insert \ --datadir
etc/upgrade/<version>
```

pour chaque version trouvée dans `etc/upgrade` des sources. Si vous êtes très joueur, vous pouvez même automatiser la mise à jour à l'aide d'une boucle `for` en shell :

```
cd etc/upgrade/ for VERSION in * ; do for ACTION in schema acl insert ;
do sbin/rt-setup-database \ --action $ACTION \ --datadir $VERSION done
done
```

Mais seulement si vous êtes joueur... `;oP`

De la version 2 à la version 3

A contrario, si vous avez une version 2 de RT déjà en production, les choses seront un peu plus compliquées. Il vous faudra déjà un nouveau serveur pour RT, avec un perl plus récent que celui de la version 2 (qui devait être un 5.6 ou 5.6.1). Idem pour tous les modules CPAN, pour Apache, `mod_perl` ou FastCGI, donc changement de serveur quasi obligatoire...

Ensuite, les modifications internes de RT étant trop importantes pour permettre une mise à jour toute simple, il vous faudra télécharger sur le site de Best Practical l'utilitaire rt2-to-rt3 [2]. Un utilitaire rt-2.0-to-dumpfile à faire tourner sur l'ancien serveur extrait tous les tickets de la base de RT2, à raison d'un fichier par ticket, plus les informations annexes (utilisateurs, files, etc.).

L'extraction peut prendre un certain temps, une demi-heure sur un (vieux) Pentium III 666 MHz pour environ 6000 tickets.

Une fois l'arborescence de fichiers contenant les tickets transférée sur le nouveau serveur (vive rsync !), l'utilitaire dumpfile-to-rt-3.0 permet de les réintégrer dans le nouveau serveur RT3. À noter que cela prend quasiment plus de temps que l'extraction, même si la machine est censée être deux fois plus rapide... Patience, donc !

Il vous restera ensuite à réintégrer les modifications de l'interface, modifications qui risquent d'être lourdes tant cette interface web a changé de la version 2 à la version 3 (eh oui, les couleurs sont plus jolies en version 3 qu'en version deux `:-)` Et ça devrait être encore mieux en version 3.6, qui ne devrait pas tarder à passer en version *beta* au moment où ces lignes sont écrites.

Automatismes et améliorations

Dans le cadre des tâches d'administration, un certain nombre de celles-ci sont des tâches relativement simples qui peuvent (doivent) être automatisées.

Escalade des priorités

Il est facile d'oublier les tickets peu importants au moment où ils arrivent. Sans mécanisme pour les remonter à notre attention ou à celle de l'équipe qui suit une file, ils seront perdus dans les profondeurs d'un classement où n'apparaîtront que des perdants.

Heureusement, il est possible d'automatiser l'élévation des priorités. Pour cela, il vous faudra vous assurer que vous avez bien défini sur vos files les limites basse et haute des priorités des tickets, ainsi que le temps alloué à la résolution d'un ticket.

Plusieurs exemples de script automatisant cette escalade des priorités existent sur le

wiki de Best Practical, mais la solution de Tim Bishop de l'université du Kent [3] est la plus sympathique.

Une amélioration que vous aimerez peut-être y faire est non pas de spécifier la liste des files sur lesquelles lancer l'escalade, mais plutôt de spécifier celles sur lesquelles **ne pas** la lancer. Afin d'éviter d'alourdir l'article, cela vous est laissé en exercice... :-)

Requêtes CLI

Pour ceux qui en ont le besoin (scripts ou liaison très bas débit), il est possible d'utiliser (mais franchement moins facilement) RT au travers de la ligne de commande, grâce à la commande `rt` fournie et installée dans la distribution. Cette possibilité est citée pour assurer la complétude de l'article. Nous vous laisserons expérimenter si cela vous tente. Mais franchement, l'interface graphique est tout de même plus simple à utiliser.

Liens RSS dans les requêtes

RT est capable de fournir des alimentations RSS. Le seul inconvénient est qu'il manque juste dans les pages le petit bout de code qui va bien pour ajouter dans votre Firefox préféré les liens dynamiques au travers du petit bouton orange à glisser dans vos favoris.

Qu'à cela ne tienne, nous allons le rajouter, puisqu'il s'agit d'une seule ligne de code (HTML) à rajouter au bon endroit.

Mais il faut d'abord trouver le morceau de page où le rajouter. La page en question est `Results.html`. La ligne qui nous intéresse est la suivante :

```
<a href="<%$RT::WebPath%>/Search/Results.rdf<%$ShortQueryString%>"><&|/l&>RSS</&></a>
```

Rien de bien sorcier, juste la syntaxe de Mason avec ses `%`. Nous allons la réutiliser quasiment telle quelle avec la balise `HTML <LINK>`. Mais avant de la rajouter, n'oubliez pas de copier `Results.html` dans la partie locale de l'installation de RT :

```
cd    /opt/rt3/share/html/Search; mkdir /opt/rt3/local/html/Search
cp Results.html /opt/rt3/local/html/Search
cd /opt/rt3/local/html/Search; vi Results.html
```

Ajoutez ensuite, où vous voulez dans la page (et c'est tant mieux, parce que ce fichier étant composé avec Mason, nous n'avons pas facilement accès à la balise `<HEAD>` du HTML) ce qui suit :

```
<link rel="alternate" title="RT search"
      href="<%$RT::WebPath%>/Search/Results.rdf<%$ShortQueryString%>"
      type="application/rss+xml">
```

Redémarrez RT, rechargez la page, et vous verrez apparaître le petit bouton d'inscription à un flux RSS. Cela vous permettra par exemple de voir l'évolution de la charge de travail de vos collaborateurs avec la file des tickets qu'ils ont pris.

RT Multi-instances : hébergement

Dans le contexte de RT, il est possible d'avoir des groupes d'utilisateurs et des files différentes. Il est aussi possible de personnaliser pratiquement tout, de l'interface aux actions automatisées, en écrivant des scrips ou en développant des modules supplémentaires.

Les questions qui peuvent se poser sont :

- Comment personnaliser une ou plusieurs files et pas les autres ?
- Que faire si la base de données grossit de manière incontrôlable et que seules quelques files posent problème ?
- Comment mettre à jour un RT en le testant au préalable en place ?
- Comment personnaliser un RT pour certaines files seulement ?
- Comment déléguer intégralement l'administration d'un RT ?

À certaines de ces questions, une solution possible est d'utiliser les listes de contrôle d'accès (*Access Control Lists* ou ACL) du système, mais l'administration peut devenir lourde et complexe – comme avec toutes les mises en œuvre d'ACL, cette problématique n'est pas propre à RT. On peut néanmoins utiliser les groupes et les droits globaux pour factoriser les droits dans une certaine mesure.

Cependant, la personnalisation des RT n'est pas possible avec les ACL. En tout cas, en version 3.4.2.

Pour l'administration déléguée, il est possible de créer plusieurs comptes « superutilisateur » qui ont un accès complet, et d'en conserver un pour d'éventuels dépannages.

On peut aussi installer plusieurs RT et démarrer plusieurs serveurs web qui écoutent sur des ports TCP différents, mais cette solution est très coûteuse, et au-delà de 65000 sites, ça va commencer à être difficile. Oui, il faudra déjà une belle machine pour faire fonctionner un truc pareil, objection retenue.

Une solution élégante et efficace consiste à mettre en place un seul moteur RT et des instances indépendantes avec leurs bases de données propres et leurs personnalisations.

On peut même pousser plus en avant ce système pour tester des versions différentes de moteurs avant un passage en production, et le tout de manière transparente aux utilisateurs.

L'utilisation de `mod_perl` n'est alors plus possible, parce que les espaces de noms sont partagés entre plusieurs instances. On utilise alors FastCGI qui permet une séparation des espaces de noms et le passage de variables d'environnement.

La modification des RT en multi-instances doit s'accompagner d'une politique de suivi de versions et de sécurité. Il est important de se fixer des règles simples dès le départ, et en particulier des règles de *nommage* (emplacement des fichiers, des bases de données, etc.).

Règles de nommage

Les règles suivantes sont appliquées sur la plate-forme d'hébergement mutualisée pour Request Tracker. Par la suite, vous constaterez l'usage massif des variables (`${MA_VARIABLE}`).

- Les moteurs RT vont dans `/opt/rt3` ;
- Les instances RT (les sites) dans `/opt/rt3-sites` ;
- Les bases de données RT ont pour nom `rtdb-` « Nom de l'instance » ;
- Le domaine SMTP pour les messages est `rt.example.com` ;
- Les adresses web seront de la forme `<a href="http://exemple.fr/NOM_INSTANCE">http://exemple.fr/NOM_INSTANCE` ;
- `/opt/rt3` est monté sur système de fichiers indépendant et contient les configurations, les bases de données, les moteurs et les exports.

Installation d'un moteur RT

Vous devez avoir installé `mod_fastcgi` et l'avoir chargé. Vous devez par conséquent avoir au moins deux lignes ressemblant à celles-ci dans votre configuration :

```
LoadModule fastcgi-module /usr/lib/apache/1.3/mod_fastcgi.so
AddHandler
 fastcgi-script fcgi
```

L'installation d'un moteur RT consiste d'abord à récupérer les sources, et les décompresser :

```
REP_MOTEURS=/opt/rt3 mkdir ${REP_MOTEURS} tar -C ${REP_MOTEURS} -xvzf
rt-3.4.2.tar.gz
```

Il faut ensuite les modifier à l'aide du patch `RT-3.4.2-multi-instances.diff`.

Ce patch modifie les sources de RT (`webmux.pl`, `RT.pm.in`, `rt-setup-database.in`) pour supporter les variables d'environnement qui permettront de découpler les moteurs RT et les instances en spécifiant l'emplacement des fichiers de configuration. Il ajoute aussi la protection des noms de bases de données MySQL contenant un caractère `-` (tiret).

Le patch initial pour le support des variables d'environnement est disponible en ligne sur le wiki de **bestpractical.com** [4].

```
cd rt-3.4.2 && \ patch -p1 < ../RT-3.4.2-multi-instances.diff
./configure --prefix=${REP_MOTEURS}/rt-3.4.2
```

Il suffit ensuite de lancer `make` et `make install` pour installer RT dans le répertoire de notre choix (`$REP_MOTEURS/rt-3.4.2`).

```
make sudo make install
```

Installation d'un site RT

Cette opération consiste à créer une structure vide de répertoires et d'y placer les personnalisations. Dans notre cas, la seule personnalisation que nous utiliserons est la configuration de base et éventuellement un logo.

Nous reviendrons plus tard sur la personnalisation pour ajouter des modules supplémentaires de statistiques.

Les opérations à réaliser sont :

- La création de l'arborescence vide avec les bons propriétaires (Apache doit pouvoir écrire dans un répertoire) ;

```
REP_INSTANCES=/opt/rt3-sites/ INSTANCE=mon-rt sudo mkdir -p \
${REP_INSTANCES}/${INSTANCE}/etc \
${REP_INSTANCES}/${INSTANCE}/var/mason_data/{obj,cache} \
${REP_INSTANCES}/${INSTANCE}/var/mason_data/obj/standard

sudo chown -R www-data \ ${REP_INSTANCES}/${INSTANCE}/var/mason_data
```

- L'édition de la configuration de l'instance dans le fichier `${REP_INSTANCES}/${INSTANCE}/etc/RT_SiteConfig.pm`. Le compte `rtuser` est utilisé pour se connecter au serveur de bases de données MySQL ;
- La création d'une base vierge avec affectation des droits au compte `mysql` défini dans (nécessite le mot de passe de l'administrateur du serveur MySQL) ;

```
cd ${REP_MOTEURS}/rt-3.4.2 && sudo sbin/rt-setup-database \
 ${REP_INSTANCES}/${INSTANCE} \ --action=init \ --dba root \
 --prompt-for-dba-password
```

REMARQUE

Si l'on dispose d'une base de données RT vierge, il est possible d'utiliser les commandes d'administration (`mysqladmin`, `mysqldump`, etc.) de MySQL pour dupliquer cette base, en substitution à l'opération précédente. Cf. Alternative à `rt-setup-database`.

REMARQUE

Re-remarque : Vous avez certainement noté l'apparition du premier paramètre sur la ligne de commande `${REP_INSTANCES}/${INSTANCE}`. Ce paramètre facultatif est ajouté par le patch `RT-3.4.2-multi-instances.diff`. Il permet de spécifier l'emplacement de la configuration, et en particulier la racine à partir de laquelle on trouvera le `RT_SiteConfig.pm` qui contient les paramètres de connexion à la base de données.

- La création d'un lien du *handler* FastCGI. Comme il n'est pas possible de démarrer plusieurs serveurs FastCGI à partir du même descripteur de serveur FastCGI, il est impératif de le copier ou de le lier (avec un lien symbolique ou direct. Bref, faites comme voulez !) ;

```
cd ${REP_MOTEURS}/rt-3.4.2/bin && sudo ln -s mason_handler.fcgi \
mason_handler_${INSTANCE}.fcgi
```

▶ D'aucuns préfèrent le lien symbolique, plus visible lors d'un listage de répertoire. D'autres préfèrent le lien direct, mais dans ce cas, utilisez si besoin l'option `-i` de la commande `ls`.

Configuration de l'application web

Voici la configuration que l'on pourrait trouver dans `/etc/apache/conf.d/rt.conf` :

```
AddHandler fastcgi-script fcgi ErrorLog /var/log/apache/rt-error_log
CustomLog /var/log/apache/rt-access_log common Include
/opt/rt3-sites/conf/apache
```

Chaque instance RT a alors sa configuration dans un fichier séparé dans le répertoire `/opt/rt3-sites/conf/apache` :

```
FastCgiServer /opt/rt3/rt-3.4.2/bin/mason_handler_mon-rt.fcgi
-initial-env RT_INSTANCE_PATH=/opt/rt3-sites/mon-rt ScriptAlias /mon-rt
/opt/rt3/rt-3.4.2/bin/mason_handler_mon-rt.fcgi <Directory
"/opt/rt3-sites/mon-rt/" > Options FollowSymLinks ExecCGI AllowOverride
None </Directory>
```

ATTENTION

Il est impératif de sortir la clause FastCgiServer de tout VirtualHost. Pour ceux qui voudraient tout de même l'inclure, sachez que votre serveur web refusera de démarrer ;o).

Si vous souhaitez utiliser malgré tout des serveurs virtuels, votre configuration ressemblera plutôt à :

```
NameVirtualHost * FastCgiServer
/opt/rt3/rt-3.4.2/bin/mason_handler_mon-rt.fcgi -initial-env
RT_INSTANCE_PATH=/opt/rt3-sites/mon-rt <VirtualHost *> ServerName
mon-rt.exemple.fr ScriptAlias /mon-rt
/opt/rt3/rt-3.4.2/bin/mason_handler_mon-rt.fcgi <Directory
"/opt/rt3-sites/mon-rt/" > Options FollowSymLinks ExecCGI
AllowOverride
None </Directory> </VirtualHost>

FastCgiServer /opt/rt3/rt-3.4.2/bin/mason_handler_ton-rt.fcgi
-initial-env RT_INSTANCE_PATH=/opt/rt3-sites/ton-rt <VirtualHost *>
ServerName ton-rt.exemple.fr ScriptAlias /mon-rt
/opt/rt3/rt-3.4.2/bin/mason_handler_ton-rt.fcgi <Directory
"/opt/rt3-sites/ton-rt/" > Options FollowSymLinks ExecCGI
AllowOverride
None </Directory> </VirtualHost>
```

Dans cet exemple, nous avons deux RT distincts, `mon-rt` et `ton-rt` accessibles respectivement aux adresses http://mon-rt.exemple.fr/mon-rt et http://ton-rt.exemple.fr/ton-rt.

Pour valider les modifications, utilisez :

```
sudo apachectl configtest && sudo apachectl graceful
```

Alternative à rt-setup-database

La méthode standard pour créer une base de données avec un schéma initial est d'utiliser la commande `rt-setup-database` de la distribution RT.

Si l'on dispose d'une base de données RT fonctionnelle, mais vierge, il est possible d'utiliser les commandes d'administration de MySQL (`mysqladmin`, `mysqldump`, etc.) pour dupliquer cette base.

Par exemple :

```
mysqladmin create rtdb-${INSTANCE} mysqldump rtdb-VIERGE | \
mysql rtdb-${INSTANCE}
```

Si l'on procède ainsi, il faut explicitement donner au compte `rtuser` le droit de manipuler la base :

```
echo "GRANT SELECT,INSERT,UPDATE,DELETE,INDEX ON
`rtdb-${INSTANCE}` TO
rtuser@localhost ;" | mysql
```

Pour le reste, c'est par le biais de l'interface web que les réglages sont affinés.

Mise à jour du schéma de la base

Si l'on a une base de données en version 3.0 et que l'on veut basculer dans une version multi-instance d'une version ultérieure, il est alors nécessaire de la mettre à jour à l'aide des scripts fournis dans les sources de RT. Comme le script `rt-setup-database` est patché pour le passage de l'emplacement du moteur en paramètres, on peut utiliser l'astuce décrite précédemment, en passant simplement le chemin de l'instance en premier paramètre. C'est grâce à ce chemin que l'on trouvera la bonne configuration de la base de données (dans `etc/RT_SiteConfig.pm`).

```
cd etc/upgrade/ for VERSION in * ; do for ACTION in schema acl insert ;
do sbin/rt-setup-database \ ${REP_INSTANCES}/${INSTANCE} \ --action
$ACTION \ --datadir $VERSION done done
```

Mais là, vous ne serez plus joueur. Vous serez directement propulsé « *hardcore-gamer* ».

Les extensions de RT

RTFM

Que nenni ! Il ne s'agit pas de la locution anglo-saxonne très fleurie « *Read That F*ck*ng Manual* », ce n'est que l'acronyme pour « RT FAQ *Manager* », ou en totalement déroulé : « *Request Tracker Frequently Asked Questions Manager* ».

En d'autres termes, il s'agit d'un outil qui se greffe sur RT (à partir de RT 3.0) et lui adjoint des capacités de base de connaissances.

Ceci implique que vous devrez avoir une installation fonctionnelle de RT pour pouvoir utiliser RTFM, même si vous n'avez configuré que l'application web, et pas la passerelle vers la messagerie électronique.

RTFM utilise la base de données de RT en ajoutant quelques objets au schéma existant, et en ajoutant quelques pages et quelques nouvelles possibilités qui deviennent accessibles dans les pages d'éditions des tickets (commentaires, réponses, etc.).

Si vous n'avez pas encore mis en place un SPIP (comme présenté dans un article précédent), RTFM peut vous permettre de démarrer une base de connaissances à peu de frais.

Et si vous avez déjà mis en place un SPIP, c'est un excellent complément pour indexer des tickets exemplaires quand on manque de temps ou bien ne justifiant pas un article complet.

Installation de RTFM

Il va être difficile de faire plus simple. Vous devrez récupérer l'archive depuis le site de BestPractical, puis la décompresser et l'installer après avoir édité le Makefile pour faire pointer les chemins sur ceux de votre installation de RT (Ha ! Quand je vous disais que RT est un pré-requis !).

Notez que si vous avez utilisé le chemin par défaut, à savoir `/opt/rt3`, vous n'aurez rien à modifier.

Accrochez-vous :

```
cd src wget
http://download.bestpractical.com/pub/rt/release/RTFM-2.0.4.tar.gz tar
xvzf RTFM-2.0.4.tar.gz cd RTFM-2.0.4
# Ci-dessous vous précisez les chemins de RT
vi Makefile sudo make install
```

Ayé ! C'est installé. Il vous reste juste à redémarrer votre serveur web.

```
sudo /etc/init.d/apache restart
```

Si vous utilisez une Debian. Notez que dans ce cas-là, vous pourriez aussi tirer parti du système APT en utilisant :

```
apt-get install rtfm
```

Si vous utilisez une Mandriva ou une RedHat, alors pour le redémarrage de votre serveur web, ça sera plutôt quelque chose comme :

```
sudo service httpd restart
```

En revanche, pour l'installation, ni les sources officielles, ni les contributions ne fournissent de paquets pour RTFM.

Configuration par l'interface web

Les classes de RTFM sont l'équivalent des files dans RT. Il en faut au moins une pour commencer à écrire des articles.

Seul un compte privilégié peut en créer par le Menu *Configuration >Classes >Nouvelle « classe » > (Saisir un nom de classe et éventuellement une description) >Valider.*

Il faut ensuite accorder les droits aux utilisateurs et/ou aux groupes d'écrire des articles dans cette classe, par le menu *Configuration >Classes >(Sélectionner la classe) >Droits de groupe*. Les droits `AjouterArticle`, `ModifierArticle`, `VoirClasse` et `AfficherArticle` vous permettront d'éditer des articles dans la classe sélectionnée précédemment.

Comme pour les tickets dans RT, il existe des champs personnalisés dans RTFM. Les droits associés sont gérés de la même façon.

Exemples d'utilisation

En supposant que vous ayez le droit de créer des articles, vous pouvez extraire un article directement à partir de l'historique du ticket à l'aide du lien *Extraire Article* rendu accessible à côté de *Commenter* dans RT. De cette façon, vous pourrez choisir pour chaque transaction du ticket celle que vous voudrez conserver.

Une autre façon de procéder est d'entrer dans le menu *RTFM* et de naviguer jusqu'à *Créer un article*. Vous pourrez alors choisir des tickets et intégrer des transactions.

La modification d'un article répond à la logique générale de RT. Il vous suffit de choisir l'article pour pouvoir changer ses attributs (titre, commentaires, tickets liés, etc.).

Si vous avez en outre eu la bonne idée de créer des champs personnalisés textuels, alors vous pourrez enrichir vos articles d'une prose qui ne manquera pas de ravir vos lecteurs.

RTIR

RTIR [5] est l'extension de RT pour la gestion d'une équipe de réponse à des incidents de sécurité (des CERT, en anglais). Ce sur-ensemble logiciel permet donc de gérer les incidents, depuis l'alerte sur un incident, le recensement et le suivi des contre-mesures mises en œuvre jusqu'à la chronologie des investigations.

Nous ne l'avons pas mis en œuvre, le besoin étant relativement restreint et spécifique, mais sachez qu'il existe.

Asset Tracker

Asset Tracker [6] est un projet prometteur qui permet de gérer des biens (*assets*), comme les serveurs, équipements réseau, etc. Il permet de gérer une main courante sur chacun des équipements, ainsi que d'associer ces équipements aux tickets de RT.

Nous ne pourrons en dire plus, sinon que l'installation avec les versions 3.4.3 et 3.4.5 de RT (AT version 1.2.1 et *-current*) se sont révélées impossibles.

Notons juste que cette association entre les tickets (qui peuvent gérer incidents, changements, problème ou demandes de service) permet d'utiliser RT dans un cadre ITIL, même sans CMDB automatisée.

Les espaces de noms RT::*, RTx::* et RT::Extension::*

L'espace de noms `RTx::` est celui qui a été utilisé au départ pour y nommer les petites extensions à RT qui sont diffusées entre

autres sur le CPAN. `RT::Extension::*` est l'espace utilisé pour certaines extensions par Jesse Vincent, l'auteur de RT.

Enfin, d'autres extensions ou applications supplémentaires sont publiées dans l'espace de nommage `RT::*`.

Parmi les principales extensions sont `RTx::RightsMatrix` de Todd Chapmann (par ailleurs auteur de *Asset Tracker*) et `RTx::Shredder` de Ruslan Zakirov, gros contributeur à RT.

Jesse Vincent a lui publié plusieurs extensions, dont `RT::Extension::MergeUsers`, non testé, mais qui doit être pratique en cas de migration de messagerie sur votre infrastructure (Jérôme parle en connaissance de cause).

Le répertoire CPAN de Jesse [7] est rempli de choses intéressantes. CPAN Suggest [7] vous aidera à trouver les autres. Mais voyons les deux modules les plus intéressants :

RTx::RightsMatrix

`RTx::RightMatrix` s'installe très facilement avec CPAN ou CPANPLUS pour peu que vous passiez la suite de tests (vous devrez pouvoir lire `RT_Config.pm`).

Une fois installé, pointez votre navigateur sur la page **http://localhost/rt/Admin/Tools/RightsMatrix/** qui devrait vous afficher une page simple avec trois entrées de menu sur la page principale : droits sur file, droits sur champs personnalisés, et droits sur groupes.

Choisissez ensuite si vous voulez travailler sur les files, les tickets, ou les droits annexes comme les droits sur les groupes.

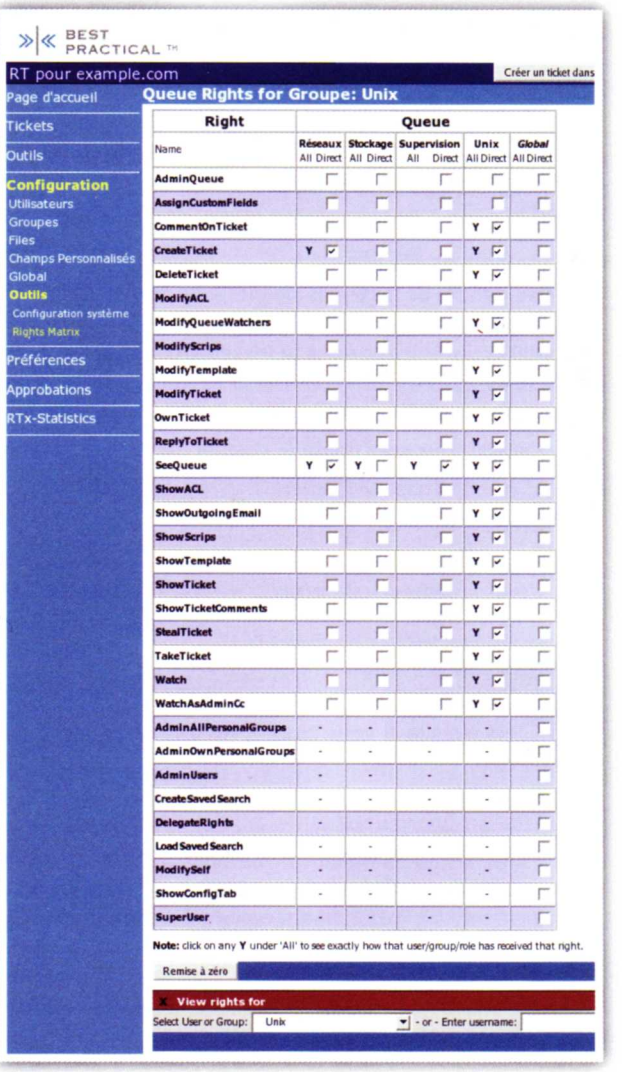
Cet écran qui peut prendre de la place si vous commencez à avoir beaucoup de files ou de groupes vous permet en un coup d'œil de voir qui a quels droits, et surtout quand ce sont des droits dérivés (globaux ou issus d'un rôle, par exemple), d'où ils sont issus.

En cochant une petite case, vous pouvez même passer en mode modification, ce qui vous permettra de régler tous les problèmes de droits que vous pouvez avoir.

Cet écran est très pratique pour, par exemple, retirer le droit global de création des tickets pour ne l'attribuer qu'aux files qui le nécessitent, comme celles qui trient les demandes extérieures, présentées au début de cet article.

RTx::Shredder

Cité ici pour saluer son existence, ce module permet de contrevenir au premier principe de base de RT : l'histoire est l'histoire, et



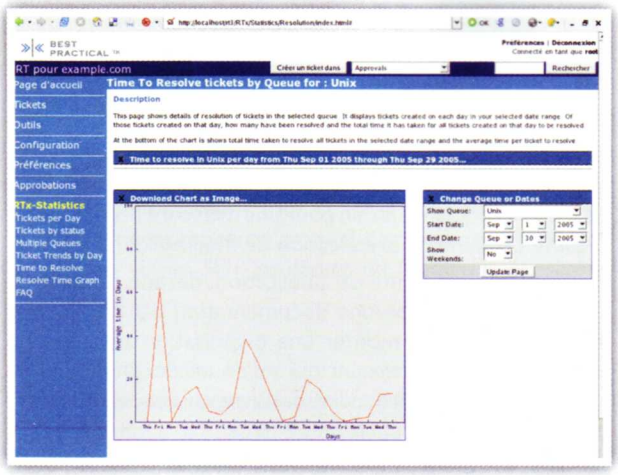
Exemple de feuille de mise à jour des droits sur files avec RTx::RightsMatrix.

on n'y touche pas. Il s'installe comme simple module CPAN où il est publié. N'oubliez pas de faire des sauvegardes avant de jouer avec...

RTx::Statistics

`RTx::Statistics` est un module Perl qui va s'installer dans l'arborescence de RT comme les précédents. Il a cependant un inconvénient : il n'est pas disponible sur CPAN. Visitez cependant le wiki [8] et vous trouverez l'endroit où le télécharger. Il s'installe manuellement comme tout bon module CPAN (`perl Makefile.PL ; make ; make test ; make install`). Les tests ne passeront pas si l'utilisateur d'exécution n'a pas les droits de lecture sur la configuration de RT (`RT_Config.pm` et `RT_SiteConfig.pm`). Deux dépendances sont à installer : les modules `GD` et `GD::Graph` qui eux-mêmes nécessiteront des paquetages de votre distribution, comme `gd-devel`.

Une fois installés, redémarrez RT, et vous aurez de quoi commencer à analyser la façon dont les tickets arrivent, sont traités, le tout avec de jolis graphiques. De quoi préparer vos réunions hebdomadaires en toute sérénité.



Exemple de graphique (temps moyen de résolution des tickets)

Conclusion

Nous espérons que ce tour d'horizon que nous pensons relativement complet de RT vous aura intéressé. L'activité de la part de Français sur la liste `rt-users` nous laisse à penser que cette série aura intéressé au moins une personne `:-)`

Cela dit, et nous allons encore une fois nous répéter, un outil sans les processus organisationnels n'aidera en rien une entité : cela ne fera qu'alourdir les tâches des différentes personnes, demandeurs ou intervenants. Nous l'avons évoqué, mais ne pouvons aller plus loin, *Linux Magazine* n'étant pas « Gestion d'équipe Magazine ».

Réfléchissez aussi à savoir si vous devez tout faire passer par RT. Certains d'entre nous connaissent des entités où le suivi du cœur de l'activité se fait par au moins trois outils (plus ou moins interconnectés), libres (bugzilla, Issue Tracker) ou propriétaires, avec RT sur d'autres sujets (comme le *helpdesk* interne ou le marketing).

Si vous voulez aller plus loin avec RT, n'oubliez pas le livre [9] qui, s'il ne s'étend pas toujours sur les sujets qui fâchent (les *approvals*, par exemple), sera complété par le wiki [10] que vous pourrez modifier vous-même, pour le plaisir de tous. Enfin, si RT est largement extensible comme montré sur ce dernier article, sachez que Jesse Vincent travaille sur un nouveau logiciel, Hiveminder [11], et qu'il en a profité pour abandonner Mason, et créer son propre cadriciel web : Jifty [12].

Hiveminder semble avoir les fonctionnalités de RT, mais avec seulement 10% du code. Attendons de voir ce que cela donnera lorsqu'une version beta sortira. Nul doute qu'il faudra revoir notre jugement au sujet de RT.

Nicolas Chuche,

nchuche@barna.be,

Nicolas Chuche est ingénieur système au ministère de l'Équipement et utilisateur de systèmes GNU/Linux et Unix depuis une dizaine d'années.

Laurent Gautrot,

l.gautrot@free.fr,

Laurent est administrateur de systèmes GNU/Linux et UNIX au Ministère de l'Équipement. Utilisateur et prosélyte de Logiciels libres depuis une dizaine d'années.

Jérôme Fenal,

jfenal@free.fr,

Jérôme Fenal est utilisateur de GNU/Linux depuis 1994, de divers Unix ou Unix-like depuis un peu plus longtemps.

Merci aux Mongueurs de toute la francophonie qui ont assuré la relecture de cet article.

LIENS & LIVRES

- [1] Request Tracker, http://bestpractical.com/rt/
- [2] `rt2-to-rt3`, script de migration, http://download.bestpractical.com/pub/rt/devel/rt2-to-rt3.tar.gz
- [3] Scripts pour RT3 de Tim Bishop, http://www.cs.kent.ac.uk/people/staff/tdb/rt3/
- [4] RT multi-instances, Wiki RT, http://wiki.bestpractical.com/index.cgi?MultipleInstances
- [5] RT-Incident Response, http://bestpractical.com/rtir/
- [6] Asset Tracker, http://atwiki.chaka.net/
- [7] CPAN Suggest, http://cpantools.com/
- [8] La page de `RTx::Statistics` sur le wiki de RT, http://wiki.bestpractical.com/index.gi?RT3StatisticsPackage
- [9] Jesse Vincent, Robert Spier, Dave Rolsky, Darren Chamberlain & Richard Foley., *RT Essentials*, O'Reilly, http://rtbook.bestpractical.com/ et sur http://www.lmet.fr/
- [10] Le wiki de RT, http://wiki.bestpractical.com/
- [11] Hiveminder, le successeur de RT, http://www.hiveminder.com/
- [12] Jifty, cadriciel utilisé par Jesse Vincent et Best Practical, http://search.cpan.org/dist/Jifty/

→ Votre base de connaissance avec Mediawiki

Yves Mettier

EN DEUX MOTS **Le logiciel faisant tourner Wikipédia, Mediawiki, peut être utilisé chez vous, comme base de connaissance. Nous allons le configurer comme tel dans cet article.**

Introduction

Comme chacun sait, Mediawiki est le logiciel qui tourne derrière la célèbre encyclopédie collaborative Wikipédia. Il est également utilisé dans d'autres sites web qui, souvent, ne sont rien d'autres que des bases de connaissances. En ce sens, vous pouvez l'utiliser aussi bien comme bloc-notes en ligne que comme serveur de documentation pour votre service en entreprise. Nous allons donc mener une petite réflexion sur ce qu'est la documentation.

La documentation d'entreprise apparaît sous au moins cinq formes différentes :

- La documentation produite par une équipe pour elle-même : c'est la documentation interne, que l'on trouve souvent dans les cahiers de chacun, et qu'il est dommage de ne pas mettre à disposition des autres ;
- La documentation produite par l'équipe pour les autres équipes ou pour l'extérieur : ce sont les normes, consignes...
- La documentation issue de l'extérieur, sous forme diverse (courriers électroniques, fichiers PDF, doc, images...), à destination de l'équipe ;
- Les rapports automatiques à destination de l'équipe ;
- Les rapports automatiques à destination de l'extérieur.

Un outil adapté à chacune de ces formes n'existe aujourd'hui pas à notres connaissance. La différence entre les types de documents et leur destination est trop grande pour qu'un seul outil effectue ces tâches correctement, aussi nous préférons la philosophie Unix : à chaque besoin son outil.

Nous distinguerons trois types de besoins. Le premier est le bloc-note, dans lequel vous pouvez placer toute documentation à destination de l'équipe, qu'elle vienne d'elle-même ou de l'extérieur. Les documents doivent pouvoir y être intégrés soit tels quels, soit reformatés en fonction de l'outil de documentation pour une meilleure homogénéité. C'est le domaine de prédilection de Mediawiki.

Le deuxième est l'outil de publication, destiné à fournir à l'extérieur de l'équipe une documentation dans un format unique. Vous devrez monter une organisation autour de la publication, en déterminant qui valide un document, qui le publie, quelle est la procédure à suivre en cas de révision.

Vous chercherez ensuite des outils adaptés à cette organisation. Lorsque le format de la documentation est libre, vous aurez intérêt à trouver un outil tel que SPIP, qui propose de nombreuses fonctionnalités autour de cette organisation.

Si vous relisez l'article de Jérôme Fenal (*Linux Mag* numéro 79) qui propose de donner les droits administrateur de SPIP à tous ceux qui ont un compte, vous verrez que l'organisation qu'il choisit est basée sur des comités de validation, de publication et de révision composés chacun de tous les membres de l'équipe.

Ce choix signifie une bonne organisation dans l'équipe pour éviter toute erreur dans une publication. Vous pouvez au contraire restreindre les droits et laisser à l'outil une partie de la gestion de l'organisation de l'équipe. Lorsque le format est imposé par la Direction, vous n'avez d'autre choix que de prendre l'outil qui va avec ou éventuellement construire vos propres outils.

Un cas classique est l'obligation d'utiliser un format .doc de Microsoft, avec lequel vous vous empresserez de tirer parti des propriétés du document, que vous pouvez relire avec une moulinette comprenant le format OLE. Si la Direction n'a rien dit, essayez de sponsoriser un outil tel que SPIP. Si la Direction a parlé, tentez encore de la convaincre...

Le troisième besoin n'est pas vraiment un outil, mais plutôt une norme visant à homogénéiser au mieux la forme des données générées automatiquement. En général, un en-tête, un pied de page et un logo font l'affaire.

Mediawiki est un *wiki* qui se distingue d'un moteur de site de publication (tel que SPIP) par plusieurs différences :

- Ce que vous écrivez est, dès validation de votre part, immédiatement en ligne. Un site de publication propose un système de relecture et c'est l'autorité compétente qui valide votre article pour publication ;
- Un document avec son propre format (PDF, tableau, audio, vidéo...) peut être directement intégré dans Mediawiki. Un site de publication ne peut fonctionner ainsi car sa vocation est la publication d'article : un tel document serait un complément, pas l'objet de l'article. Il faut donc créer l'article et lui joindre le document comme pièce attachée ;
- La page d'accueil contient ce que vous désirez, contrairement à un site de publications qui, par principe, indiquera les derniers articles en ligne.

Il est évidemment possible de détourner Mediawiki de son rôle de base de connaissance et d'en faire un site de publication. Il existe par exemple une page spéciale indiquant les dernières pages créées.

Il est tout aussi possible de détourner un outil de publication pour en faire une base de connaissances. Néanmoins, chacun ayant un public différent, vous veillerez à ne pas mélanger les genres et pouvez sans problème diffuser votre documentation aux autres via un SPIP et dédier un Mediawiki à celle de votre équipe.

Pré-requis

Trêve de discussion, entrons dans le vif du sujet. Votre serveur web Apache, votre base de données MySQL et votre moteur PHP ronronnent-ils déjà à la nouvelle d'accueillir votre tout nouveau Mediawiki ? S'ils ne faisaient pas déjà partie de la famille, faites le nécessaire pour les installer. N'importe quelle version d'Apache devrait convenir, du moment que vous pouvez y installer une version stable de PHP.

Que ce soit pour PHP ou MySQL, une coïncidence fait que les numéros de version majeure qui conviennent à Mediawiki sont 4 et 5. Privilégiez ce que propose votre distribution pour vous simplifier l'installation et la maintenance.

Si vous n'utilisez pas Linux (personne n'est parfait), vous pouvez être amené à installer l'ensemble en recompilant à partir des sources. Dans ce cas, toujours dans un souci de maintenance, n'hésitez pas à choisir les dernières versions, Mediawiki n'est pas un enfant difficile.

Configuration des outils

Vous pouvez installer Mediawiki directement dans un répertoire de l'arborescence connue d'Apache. Néanmoins, pour isoler ce logiciel des documents, nous préférons l'installer ailleurs et créer un *alias* :

```
Alias /wiki "/opt/mediawiki/mediawiki-1.5.6"
<Directory "/opt/mediawiki/mediawiki-1.5.6">
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

PHP ne nécessite pas de configuration particulière. Quant à MySQL, vous pouvez déléguer à Mediawiki la création de la base et de l'utilisateur moyennant le mot de passe administrateur de MySQL que vous devrez lui communiquer. Nous préférons effectuer le travail nous-même :

```
$ mysql -u root -p
Enter password: *********
[...]
mysql> create database wikidb;
mysql> grant all on wikidb.* to
          'wikiuser'@'localhost'
          identified by 'MOTDEPASSEENCLAIR';
```

Si le serveur Apache tourne sur la même machine que le serveur MySQL, vous pouvez mettre *localhost*. Il s'agit en réalité d'être cohérent sur le nom de la machine, car MySQL est pointilleux lors de l'authentification et n'accepte pas d'utilisateur ne venant pas de la machine attendue. De plus, nous choisissons localhost qui utilise la boucle locale et qui marche même quand le réseau n'est pas configuré (ce qui est peut-être le cas sur votre portable en DHCP quand vous voyagez), sans parler de l'optimisation légendaire et imperceptible au niveau de la vitesse d'exécution des requêtes !

Le nom de la machine, le nom de l'utilisateur et son mot de passe, que vous indiquez pour vous connecter à MySQL, ne sont pas à retenir. Nous les utiliserons une fois plus loin lors de la configuration de Mediawiki. Vous pourrez les oublier ensuite : ils ne servent pas à l'utilisation du wiki.

Installation de Mediawiki

Si vous n'avez pas relancé Apache pour prendre en compte l'alias, faites-le. Décompactez l'archive de Mediawiki (`mediawiki-1.5.6.tar.gz` lors de l'écriture de cet article) dans `/opt/mediawiki` et dans la mesure du possible, faites appartenir l'arborescence `/opt/mediawiki` à l'utilisateur du serveur Apache. Lors de la configuration du wiki, celui-ci devra avoir un accès en écriture au répertoire `/opt/mediawiki/mediawiki-1.5.6/config`. Cela ne pose de problème que si l'utilisateur d'Apache n'a pas les permissions en écriture sur ce répertoire. Dans ce cas, mettez les permissions à 777 sur ce répertoire le temps de la configuration, ce qui ne devrait pas poser trop de problème si vous allez vite. Personne ne sait en effet que vous installez cet outil.

Allez ensuite visiter cette page : **http://localhost/wiki/**. Après l'un ou l'autre clic, vous devrez donner quelques informations relativement triviales, dont le nom de la base, l'utilisateur et son mot de passe...

Vous devrez aussi choisir un nom d'utilisateur, dit « *sysop* », qui sera administrateur de l'outil. Choisissez un nom adapté, par exemple `sysop`, `admin` ou `wikiadmin`. Retenez bien le mot de passe, sinon...

Mediawiki devrait, après configuration, vous indiquer la marche à suivre : déplacez le fichier `config/LocalSettings.php` dans `/opt/mediawiki/mediawiki-1.5.6`.

Changez les permissions du répertoire `config` à 700. Votre wiki est maintenant prêt à l'emploi. Mais n'allez pas tout de suite jouer avec...

Configuration post-installation : sécuriser l'outil

Outre une protection façon `.htaccess` d'Apache pour protéger votre wiki, vous pouvez (devez ?) interdire aux utilisateurs non identifiés d'éditer les pages. Cela oblige en quelque sorte à chaque utilisateur de signer ses modifications, et au responsable du wiki de déterminer facilement l'auteur d'une faute d'orthographe (ou pire). Dans le même ordre d'idées, vous souhaiterez probablement interdire aux visiteurs de se créer un compte à la demande.

Ces deux protections s'effectuent en éditant le fichier `LocalSettings.php` auquel vous pouvez ajouter ceci à la fin :

```
// Seul l'admin peut créer des comptes
$wgGroupPermissions['*']['createaccount'] = false;
```

et

```
// L'édition en anonyme est interdite
$wgGroupPermissions['*']['edit'] = false;
```

Vous disposez aussi d'un paramètre pour interdire la lecture aux utilisateurs non identifiés, mais mieux vaut l'autoriser par ce biais. En effet, cela vous évitera d'avoir à vous connecter pour voir des pages si vous n'avez pas à en éditer.

De plus, dans la version 1.5.6 au moins, un bug fait que si vous interdisez la lecture des pages aux utilisateurs non autorisés, vous leur interdisez également l'accès à la page de connexion, ce qui rend impossible toute authentification.

Espérons que ce bug sera corrigé rapidement. Et de toute façon, il vaut mieux protéger le site avec une authentification par Apache, par exemple avec un fichier `.htaccess` bien placé.

```
$wgGroupPermissions['*']['read']          = true;
```

Vous pouvez maintenant vous amuser à créer des utilisateurs. Pour cela, vous devez vous servir de votre compte `sysop` qui est maintenant devenu le seul habilité à créer des comptes.

Cliquez en haut à droite ou allez sur **http://localhost/wiki/index.php/Special:Userlogin** pour vous identifier. Vous devez obtenir une page comme celle de la figure 1. Retournez à cette même URL directement ou en cliquant sur les *pages spéciales* puis sur *créer un compte ou se connecter*.

Avec le compte `sysop`, la page est alors similaire à la figure 2. C'est ici que vous créez les comptes utilisateur.

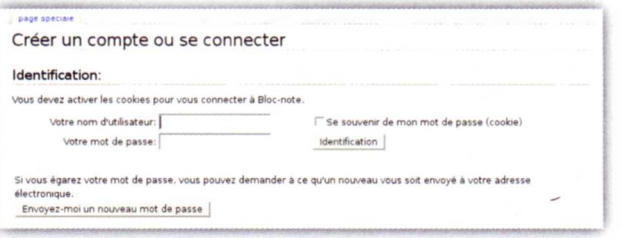


Figure 1

Figure 2

Changez le logo

Le logo par défaut est la première chose qui se voit sur le site. Il risque de vous énerver, vous faire perdre patience, et avant que vous n'ayez l'idée d'aller vous pendre à cause de lui, mieux vaut le changer. Une façon consiste à écraser le logo en place (`skins/monobook/wiki.png`).

Mais mieux vaut indiquer dans `LocalSettings.php` l'endroit où se trouve le vôtre. La variable à modifier est `$wgLogo` dans laquelle vous y mettez la bonne URL. Pour votre information, l'image `wiki.png` a pour dimensions 135 x 135.

Autorisez le chargement

Indiquez un nom de fichier de la façon suivante : `[[Image:fichier.pdf]]` ou `[[Media:fichier.pdf]]`. Cette syntaxe permet de créer un lien directement vers un fichier, qui n'est pas forcément `image` ou `média` contrairement à ce que le tag indique. Le premier (`image`) vous renvoie vers une page décrivant le fichier, avec un lien pour le télécharger. L'autre amène à un téléchargement direct. A ce stade, la question que vous vous posez est : comment mettre le fichier en question à disposition ? Cliquez sur le lien et, si le fichier n'existe pas encore dans les tablettes de Mediawiki, il vous proposera une interface pour le lui charger.

Par défaut, l'envoi de fichiers est désactivé. Voici comment nous allons procéder. Tout d'abord, nous allons créer un répertoire distinct de l'installation de mediawiki, par exemple `/opt/mediawiki/data`. Pensez à le déclarer dans la configuration d'Apache et à relancer ce dernier :

```
Alias /wiki "/opt/mediawiki/wikidata"
<Directory "/opt/mediawiki/data">
    AllowOverride AuthConfig
    Order allow,deny
```

```
    Allow from all
</Directory>
```

Nous allons ainsi pouvoir accéder à nos données via l'URL http://localhost/wikidata. Éditez maintenant le fichier de configuration LocalSettings.php pour y ajouter ceci :

```
$wgEnableUploads = true;
$wgUploadPath = "/wikidata";
$wgUploadDirectory = "/opt/mediawiki/data";
```

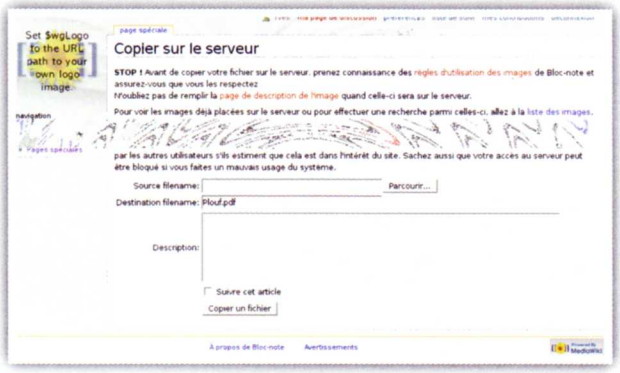


Figure 3

D'autres paramètres permettent de limiter les chargements, selon le type de fichier ou la taille :

```
$wgFileExtensions = array( 'png', 'gif',
          'jpg', 'jpeg', 'doc', 'xls',
          'ppt', 'sxw', 'pdf', 'gz' );
$wgCheckFileExtensions = true;
$wgStrictFileExtensions = true;

$wgUploadSizeWarning = 150000;
```

Essayez à nouveau de charger un fichier : cela devrait être possible maintenant.

Modifiez la boîte de navigation de gauche

Prenez l'identité de l'administrateur de Mediawki (votre compte sysop) car ceci ressemble plus à un *cheat code* qu'à autre chose, et est donc réservé au *master*. Visitez maintenant l'adresse suivante : http://localhost/wiki/index.php/MediaWiki:Sidebar.

La syntaxe est quasi évidente. Vous noterez néanmoins qu'à gauche de la barre verticale se trouve l'URL et à droite son intitulé. A vous de jouer !

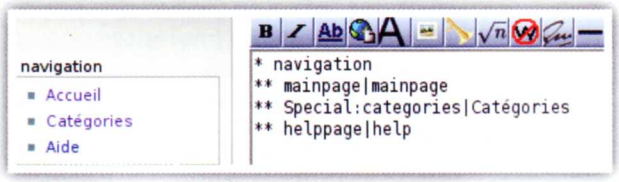


Figure 4

Réalisation d'une extension

L'extension FileSystemListing

Mediawiki peut facilement être étendu à l'aide d'extensions. Vous pouvez définir de nouveaux tags afin d'afficher un contenu dynamique. Un bon exemple est l'affichage d'une arborescence de fichiers.

Vous pouvez aller sur http://meta.wikimedia.org/wiki/FileSystemListing et y copier/coller le code de l'extension FileSystemListing dans un fichier que vous appellerez par exemple FileSystemListing.php et que vous placerez dans le répertoire /opt/mediawiki/mediawiki-1.5.6/extensions.

Pour activer l'extension, ajoutez include ("extensions/FileSystemListing.php"); dans votre fichier LocalSettings.php. Il ne vous reste plus qu'à écrire une page contenant ceci :

```
<dirlist dir="/home/votrecompte"></dirlist>
```

Remarquez que l'auteur de cette extension a aussi prévu l'attribut fileprefix qui sert typiquement dans des cas comme celui-ci :

```
<dirlist dir="/var/www/htdocs/fichiers" fileprefix="http://localhost/
fichiers"></dirlist>
```

Notre extension ShowDir

Passons à la réalisation de notre propre extension qui, contrairement à FileSystemListing, ne sera pas récursive à travers les sous-répertoires, mais montrera également la taille et la date de dernière modification de chaque fichier. Elle définira le tag dirinfo et aura pour attributs dir et fileprefix comme FileSystemListing.

Pour commencer, vous pouvez créer une extension pour dire bonjour au monde :

```
 1: <?php
 2:
 3: $wgExtensionFunctions[] = "wfDirInfoExtension";
 4:
 5: function wfDirInfoExtension() {
 6:    global $wgParser;
 7:    $wgParser->setHook("dirinfo","renderDirInfo");
 8: }
 9:
10: function renderDirInfo( $input, $argv ) {
11:    $output = "Bonjour tout le monde !";
12:    $output .= "<br /><b>$input</b>";
13:    $output .= "<br />dir=".$argv["dir"];
14:    $output .= "<br/>fileprefix=".$argv["fileprefix"];
return $output;
15: }
16: ?>
```

Déchiffrons cette extension. La ligne 3 ajoute notre extension à celles connues de Mediawiki. Nous avons donc appelé la notre `wfDirInfoExtension`. Cela nous oblige à créer une fonction avec ce nom, ce qui doit permettre à Mediawiki d'initialiser certaines choses, dont ce qui nous intéresse le plus, notre (nos) nouveau(x) tag(s). C'est chose faite ligne 7 où nous indiquons à l'analyseur (via la variable globale `$wgParser`) que notre tag s'appelle `dirinfo` et que pour l'utiliser, il peut appeler notre fonction `renderDirInfo()`.

Dans cette fonction, nous entrons dans le vif du sujet. Elle prend en premier argument le texte se trouvant entre le tag ouvrant et le tag fermant, et en second les attributs sous forme de table de hachage.

Les lignes 12 à 14 illustrent la façon de s'en servir. Le résultat de cette fonction est du HTML. Vous n'avez pas à générer une sortie avec la syntaxe wiki.

N'oubliez pas d'ajouter un `include ("extensions/DirInfo.php");` à `LocalSettings.php` puis mettez `<dirinfo dir="/var/www/htdocs" fileprefix="/ ">Géniâââl</dirinfo>` dans une page du wiki, et regardez !

La suite est simple, aussi simple que du PHP. Voici une nouvelle version de notre fonction `renderDirInfo()` :

```
function renderDirInfo( $input, $argv ) {
# Récupération des variables
$dir=$argv["dir"];
$base=$argv["fileprefix"];

# Création du tableau
$output = "<table border='1'>\n";
$output .= "<tr>";
$output .= "<th>Name</th><th>Last modification</th><th>Size</th>";
$output .= "</tr>\n";

# Lecture du répertoire
$fd = opendir($dir);
while (($f = readdir($fd)) !== false)
{
          if ($f == "." || $f == "..")
                    continue;
# Pour chaque fichier, une nouvelle ligne du tableau...
          $output .= "<tr>";
          $s = stat("$dir/$f");
          if (is_dir("$dir/$f")) {
            $output .= "<td>$f/</td>";
            $output .= "<td>".
              strftime("%d/%m/%Y %H:%M:%S",$s[9]).
              "</td>";
            $output .= "<td> </td>";
          } else {
            $output .= "<td><a href='$base/$f'>".
              "$f</a></td>";
            $output .= "<td>".
              strftime("%d/%m/%Y %H:%M:%S", $s[9]).
              "</td>";
            $output .= "<td>$s[7]</td>";
          }
          $output .= "</tr>\n";
}
# On ferme !
closedir($fd);
$output .= "</table>\n";
return $output;
}
```

Sauvegardez vos données

La sauvegarde des données d'un Mediawiki n'est pas aussi simple que vous pouvez le penser. En elle-même, rien de plus facile : *dump* de la base, sauvegarde des fichiers et c'est gagné. Perdu : vous allez suer pour restaurer une telle sauvegarde.

La première étape consiste effectivement à sauvegarder la base avec `mysqldump`. La restauration peut s'effectuer avec une injection du dump dans la base comme cela s'effectue généralement.

Les problèmes surviennent avec les accents et autres cédilles. Vous n'avez aucune connaissance a priori du jeu de caractères employé, ni à la sauvegarde ni à la restauration. Pour en avoir la maîtrise, vous devez impérativement vous servir de l'option `--default-character-set=latin1`. Si vous avez appelé votre base `wikidb`, voici la sauvegarde :

```
$ mysqldump -u wikiuser -p \
  --default-character-set=latin1 \
  -B wikidb > wikidb.dump
$ bzip2 wikidb.dump
```

Nous avons comparé la compression d'un tel dump avec gzip et bzip2. Sur un seul échantillon – cela montre l'objectivité du test – bzip2 semble meilleur avec un taux de compression de 27%.

Nous ne ferons pas l'affront au lecteur de lui décrire les conditions d'un tel test. La restauration d'un tel dump s'effectue ainsi :

```
$ mysql -u root -p
Enter password: *********
[...]
mysql> create database wikidb;
mysql> grant all on wikidb.* to
  'wikiuser'@'localhost'
  identified by 'MOTDEPASSEENCLAIR';
mysql> exit
$ mysql --default-character-set=latin1 \
  -h localhost -D wikidb \
  -u wikiuser -p < wikidb.dump
Enter password: *********
```

La sauvegarde ne se limite pas à celle de la base. Vous devez en effet effectuer une copie de la configuration de Mediawiki ainsi que, éventuellement, de tout fichier (thèmes, extensions...) que vous auriez pu ajouter ou modifier.

Et vous ne devez pas oublier non plus les fichiers que les utilisateurs auraient pu charger. Pour nous, la façon la plus simple est de disposer d'une copie de `/opt/mediawiki/mediawiki-1.5.6` et de `/opt/mediawiki/data`. Le script de sauvegarde peut ressembler à celui-ci :

```
#! /bin/sh
# Preparatifs
DATE=`date +'%Y%m%d'`
cd /opt/mediawiki
mkdir -p backup
# Sauvegarde des fichiers
tar cvzf backup/mediawiki-1.5.6.${DATE}.tar.gz \
  mediawiki-1.5.6
tar cvzf backup/mediawiki-data.${DATE}.tar.gz \
  data
# Sauvegarde de la base
mysqldump -u wikiuser -p MOTDEPASSEENCLAIR \
  --default-character-set=latin1 \
  wikidb > wikidb.dump
bzip2 wikidb.dump
```

Ce script contient le mot de passe en clair de l'utilisateur de la base. Aussi, vous ferez le nécessaire pour le protéger en lecture contre les indiscrets.

Nous avons vu la restauration de la base. La restauration des fichiers s'effectue en remettant les archives des fichiers à leur place comme cela était auparavant. Vous devrez néanmoins vérifier que le fichier `LocalSettings.php` correspond bien à la réalité.

Déplacez votre Mediawiki

L'opération s'effectue comme une sauvegarde suivie d'une restauration sur une machine différente. Vous n'oublierez pas de configurer le serveur web pour qu'il connaisse l'existence des répertoires de Mediawiki. Vous penserez également à vérifier le propriétaire et les permissions de chaque fichier.

Cette opération peut présenter un intérêt. En effet, outre le fait de tester la sauvegarde et la restauration de votre Mediawiki, cela vous permet de disposer d'une copie en état de fonctionnement de votre base de connaissance. Ceux qui disposent d'un ordinateur portable comprendront rapidement l'intérêt que cela peut présenter : avoir accès à ses données sans être connecté y compris à celles hébergées par MySQL. Le coût est faible puisque vous n'avez pas à investir dans des mécanismes de synchronisation des données entre deux bases de données. Le prix à payer est par conséquent élevé : vous ne devrez jamais travailler que sur une seule base à la fois, et sauvegarder puis restaurer la base la plus récente sur l'autre. Mais nous n'allons pas entrer dans un débat sur le travail collaboratif sur un même ensemble de données dans cet article...

Conclusion

Vous voilà avec un Mediawiki qui, s'il n'est pas encore configuré aux petits oignons, ne demande qu'à l'être. Si vous n'êtes pas le seul utilisateur, pensez à éditer la page de l'aide au plus vite pour y indiquer l'URL de la documentation utilisateur.

En effet, il n'est pas trivial qu'un lien hypertexte s'effectue entre [crochets] simples et un lien wiki entre [[crochets]] doubles. N'oubliez pas non plus les mentions légales, comme quoi chacun est responsable de ce qu'il écrit ou, sur un intranet, comme quoi il ne faut pas s'en servir à d'autres fins que le rôle qui lui est attribué (documentation, base de connaissance...).

Mediawiki, comme de nombreux outils web, peut être complété d'autres outils. Vous pouvez ainsi paramétrer un Mnogosearch pour qu'il serve de moteur de recherche complémentaire à celui intégré. Ceci n'est pas simple à réaliser car il existe de trop nombreuses combinaisons dans les URL qui permettent d'accéder à l'information qu'un Mediawiki peut contenir. Si vous l'utilisez, n'oubliez pas que le répertoire des données peut, par contre, être facilement indexé.

Au niveau de la sécurité, vous pouvez penser au pare-feu applicatif `mod_security` qui filtre les URL envoyées au serveur web. Il permet entre autres d'éviter des attaques par injection de code SQL. Enfin, vous pourrez changer le thème par défaut et proposer le vôtre, respectant la charte graphique de votre site. Le thème par défaut est le Monobook dont vous trouvez les fichiers dans le répertoire `skins` (`MonoBook.php` et `monobook/`).

Recopiez-les en changeant le nom, et éditez les à souhait. Votre site n'en sera que plus beau, plus agréable à utiliser.

RÉFÉRENCES

- Mediawiki : http://www.mediawiki.org
- Documentation de l'éditeur : http://meta.wikimedia.org/wiki/Aide:%C3%89diteur
- Documentation administrateur : http://meta.wikimedia.org/wiki/Aide:Administrateur
- Écrire une extension : http://meta.wikimedia.org/wiki/Write_your_own_MediaWiki_extension
- FAQ de Mediawiki : http://meta.wikimedia.org/wiki/MediaWiki_FAQ
- Extensions existantes : http://meta.wikimedia.org/wiki/Category:Mediawiki_Extensions
- Extension FileSystemListing : http://meta.wikimedia.org/wiki/FileSystemListing
- Chargement de fichiers : http://www.mediawiki.org/wiki/Configuring_file_uploads et http://meta.wikimedia.org/wiki/Uploading_files
- Écrire son propre thème : http://meta.wikimedia.org/wiki/User_styles
- Spip : http://www.spip.net
- Mnogosearch : http://www.mnogosearch.org/
- Mod_security : http://www.modsecurity.org/

Yves Mettier,

ymettier@libertysurf.fr,

Consultant Unilog/LogicaCMG et auteur de C en action paru chez O'Reilly

→ Principes et utilisation de SSH

Alexandre Courbot

EN DEUX MOTS **Le protocole SSH est surtout connu comme étant un moyen sécurisé d'obtenir un shell sur une machine distante. Cette vision n'est pas fausse, mais elle est en revanche terriblement limitée. Dans cet article, nous allons couvrir l'utilisation basique de SSH ainsi que quelques-unes de ses utilisations moins connues, comme le tunneling ou le montage de systèmes de fichiers distants. Nous verrons ainsi que SSH est bien plus qu'un simple remplaçant de telnet !**

Historique

SSH (pour *Secure SHell*) est un protocole réseau sécurisé développé à l'université de Helsinki. Son but premier était de fournir un remplaçant robuste aux vieillissants `telnet`, `rlogin` et `rsh`. `telnet` est un vieux protocole (1969 !) qui permet d'établir une connexion TCP sur un hôte pour y envoyer et recevoir des données.

Typiquement, il était utilisé pour se connecter à une machine distante et utiliser son propre ordinateur comme un terminal de cette machine.

Les caractères tapés au clavier étaient envoyés au serveur, et ce dernier renvoyait les caractères à afficher à l'écran de l'utilisateur. `rlogin` et `rsh` utilisent deux protocoles proches de celui de `telnet`. Dans la même veine, `rcp` permettait de copier des fichiers d'une machine à une autre.

Ces programmes étaient certes très pratiques, mais transmettaient leurs données sur le réseau en clair.

Ce n'était pas un problème à une époque où les réseaux étaient fermés, mais dès lors qu'ils sont devenus accessibles au grand public (notamment dans les universités), un problème s'est vite posé : n'importe quelle personne ayant accès aux câbles pouvait écouter ce qui passait sur le réseau et récupérer ainsi fichiers, *logins* et mots de passe !

Quand on sait qu'il y a encore quelques années, les machines d'un réseau Ethernet étaient souvent reliées entre elles par un *hub* (équipement renvoyant les données émises par l'un des postes branchés dessus à tous les autres), on comprend la réalité de la menace. Et ne parlons même pas d'internet.

SSH offre un service relativement similaire à `telnet` et ses petits frères, mais l'encapsule dans une connexion sécurisée basée sur une cryptographie à clé publique. Il existe deux versions du protocole SSH.

La version 1 est considérée comme obsolète et trop vulnérable, et ne devrait donc plus être utilisée. La version 2, la seule que nous considérerons ici, fait encore ses preuves de nos jours. La plupart des clients SSH supportent les deux protocoles.

Les exemples de cet article sont prévus pour Openssh, l'implémentation libre du client SSH que l'on devrait retrouver sur toutes les distributions Linux.

Utilisation de base

Voici la configuration réseau que nous allons assumer pour les exemples de cet article : `lancelot` est la machine client sur laquelle nous travaillons. Nous voulons nous connecter à `percival`, qui dispose donc d'un serveur SSH (Figure 1).

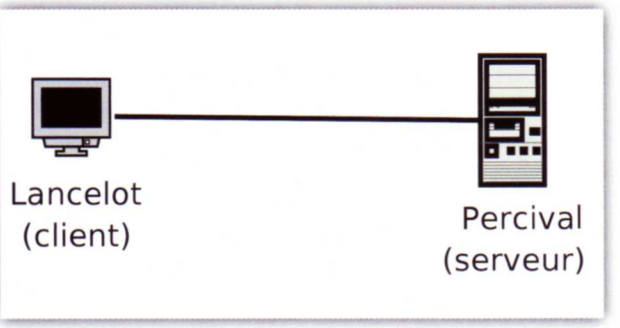


Fig. 1 : Configuration réseau de lancelot et percival.

Utiliser SSH comme telnet

L'exemple le plus simple d'utilisation de SSH est de se connecter directement à un serveur afin d'obtenir un shell :

```
moi@lancelot:~$ ssh percival
Password:
moi@percival:~$
```

Ici, nous avons simplement ouvert une session pour l'utilisateur `moi` (qui lance la connexion) sur `percival` en donnant son mot de passe sur cette machine. Un autre utilisateur peut être précisé en utilisant la syntaxe suivante :

```
moi@lancelot:~$ ssh lui@percival
Password:
lui@percival:~$
```

Openssh supporte de nombreuses options. Voici quelques-unes des plus fréquentes :

- `-X` permet de rediriger l'affichage des programmes Xwindow vers le serveur X du client. En d'autres termes, si vous lancez un programme X tel que `xeyes` sur le serveur, celui-ci s'exécutera sur ce dernier mais s'affichera sur l'écran du client.
- `-C` permet de compresser les données transférées. Cette option est souvent utilisée conjointement à `-X` pour soulager les lignes à bas débit.
- `-v`, `-vv` et `-vvv` seront utiles si le serveur SSH vous refuse l'accès alors que vous pensez qu'il ne devrait pas. Ces options rendent votre client plus bavard sur ce qu'il fait.

Copie de fichiers distants

SSH offre également un remplaçant à `rcp`. `scp` permet de copier un ou plusieurs fichiers en utilisant une syntaxe similaire à celle de `cp` :

```
$ scp monfichier.txt lui@percival:~/
```

Ici, `monfichier.txt` sera copié dans le répertoire personnel de l'utilisateur `lui` de la machine `percival`, en s'identifiant comme ce dernier. Le caractère `:` permet de séparer le nom d'hôte du chemin vers le fichier distant.

`scp` supporte aussi beaucoup d'options. On notera les suivantes :

- `-p` préserve les attributs des fichiers copiés (date de modification, droits, etc.).
- `-r` copie récursivement les répertoires donnés.
- `-C`, comme son homologue de SSH, compresse les données durant le transfert.

À un plus haut niveau, existe également le protocole SFTP. SFTP n'est **pas** une version de FTP sur SSH, mais au contraire un nouveau protocole. Toutefois, les commandes supportées par le client sont très semblables à celle d'un client FTP :

```
$ sftp lui@percival
Connecting to percival...
Password:
sftp> ls
Desktop
Bin
sftp> put monfichier.txt
Uploading monfichier.txt to /home/lui/monfichier.txt
monfichier.txt                                100%   12KB
12.1KB/s   00:00
sftp> quit
```

`scp` et `sftp` disposent de nombreux clients graphiques, comme `lftp` ou `krusader`. Nous verrons également à la fin de l'article que ce protocole est bien intégré aux environnements de bureau modernes.

Ces quelques exemples d'utilisation constituent les bases de SSH. À première vue, SSH n'est qu'un équivalent sécurisé à `telnet` ou `rcp`... En fait, il se comporte juste comme tel.

Utilisation avancée

Nous savons maintenant utiliser SSH comme on utiliserait `telnet` ou `rcp`. C'est fort bien, mais ce serait se limiter grandement que d'en rester là : SSH peut grandement faciliter la vie de celui qui comprend comment il fonctionne.

Clés SSH et méthodes d'authentification

Bien que l'utilisation de SSH du point de vue de l'utilisateur semble aussi simple que celle de `telnet`, une importante négociation se produit entre client et serveur quand vous vous connectez.

Celle-ci a deux buts : s'assurer de l'identité de la personne qui se connecte, et établir une connexion chiffrée entre client et serveur. Observons ce qui se passe en détail...

La cryptographie de SSH est dite « à clé publique ». S'il y a clé publique, il y a forcément... clé privée. Un couple clé publique/clé privée a la propriété suivante : tout ce qui est chiffré par la clé publique ne peut être déchiffré que par la clé privée, et vice-versa.

Il est également impossible de dériver l'une à partir de l'autre. Comme son nom l'indique, la clé publique est destinée à être diffusée, tandis que la clé privée ne doit être connue que de son propriétaire.

Ce genre de cryptographie a de nombreuses autres applications, comme la signature ou le chiffrage des emails. En fait, l'authentification de SSH ne fonctionne pas différemment de ces applications.

Pour pouvoir se connecter à un serveur SSH, il faut donc un tel couple de clés. Celles-ci sont personnelles à l'utilisateur. Il existe également une paire de clés pour l'hôte, qui se trouvent généralement dans `/etc/ssh`.

L'intérêt de disposer de sa propre clé réside dans le fait qu'il est alors possible de s'affranchir de toute authentification manuelle, comme nous allons le voir.

Les clés d'un utilisateur se trouvent dans `~/.ssh/`, et sont contenues dans les fichiers `id_algo[.pub]`, où `algo` désigne l'algorithme de cryptage employé. La clé privée n'a pas d'extension, la clé publique a l'extension `.pub`.

Pour générer ces fameuses clés, nous pouvons utiliser le programme ssh-keygen :

```
$ ssh-keygen -t dsa -b 4096
```

Cette commande suffira pour générer une paire de clés DSA d'une taille de 4096 bits. L'algorithme RSA peut également être utilisé, mais le DSA reste en général préféré.

La génération prend un certain temps, après lequel il est demandé dans quel fichier sauvegarder la clé privée (laissez les valeurs par défaut), ainsi qu'une phrase de passe (*passphrase*). C'est cette phrase qui permettra d'utiliser la clé.

Elle constitue une sécurité supplémentaire, car dans le cas où votre clé privée se fait subtiliser, l'auteur du forfait ne pourra rien en tirer s'il ne connaît pas la phrase de passe. Choisissez-la bien. Vous pouvez également décider de vous en passer... à vos risques et périls.

Une fois l'exécution de ssh-keygen terminée, vous vous retrouvez donc avec deux fichiers `id_dsa` et `id_dsa.pub` dans le répertoire `~/.ssh`. Comment en tirer parti ? Pour commencer, nous allons nous authentifier sur le serveur non plus avec notre mot de passe utilisateur, mais avec notre clé SSH.

Pour cela, nous allons nous connecter sur le serveur et créer ou éditer le fichier `~/.ssh/authorized_keys` pour y ajouter notre clé publique (par copier/coller de l'intégralité du fichier ou en utilisant `scp`).

Ce fichier est censé contenir une clé publique par ligne, et indique que ces clés peuvent être utilisées pour authentifier l'utilisateur qui souhaite se connecter. Attention, comme les clés sont longues, à ne pas insérer de retours à la ligne lors du copier/coller !

Maintenant, comment activer l'authentification par clé lors de la connexion ? Eh bien, il n'y a strictement rien à faire. SSH essaie successivement différentes méthodes d'authentification pour s'assurer de l'identité d'un utilisateur, et la méthode consistant à demander son mot de passe sur le système arrive en dernier.

L'authentification par clé est l'étape précédente. Il existe encore une autre méthode d'authentification par hôte, venant avant l'authentification par clé. Elle est souvent désactivée par défaut et nous ne la couvrirons donc pas ici.

Réessayons de nous connecter, en passant l'option `-v` à notre client pour voir ce qui se passe :

```
$ ssh -v percival
...
debug1: Authentications that can continue: publickey,keyboard-interactive
debug1: Next authentication method: publickey
...
debug1: Trying private key: /home/moi/.ssh/id_rsa
debug1: Offering public key: /home/moi/.ssh/id_dsa
debug1: Server accepts key: pkalg ssh-dss blen 818
debug1: PEM_read_PrivateKey failed
debug1: read PEM private key done: type <unknown>
Enter passphrase for key '/home/moi/.ssh/id_dsa':
```

Oh ! On nous demande cette fois le mot de passe de la clé privée. Une fois tapé, l'accès à `percival` est donné.

Comment fonctionne l'authentification par clé ? Le serveur SSH, par le biais du fichier `authorized_keys`, dispose d'une liste de clés publiques qu'il peut utiliser pour authentifier l'utilisateur.

Pour ce faire, il va envoyer au client un « challenge », c'est-à-dire un court message chiffré avec la clé publique. Si le client est capable de le déchiffrer et de répondre au challenge, c'est qu'il dispose de la clé privée, et donc qu'il est autorisé à se connecter.

Tout cela est bien joli, mais il faut tout de même taper la phrase de passe de sa clé pour accéder à notre hôte distant... Pas forcément, car on peut utiliser ssh-agent pour autoriser l'utilisation de notre clé privée une fois pour toute.

Le gardien des clés

ssh-agent est un petit programme qui gardera nos clés privées ouvertes pour les utiliser à chaque fois que nécessaire, sans redemander leur mot de passe. Si vous le lancez en ligne de commande, vous ne manquerez pas d'être perplexe par son comportement :

```
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-sciaTo7933/agent.7933; export SSH_AUTH_SOCK;
SSH_AGENT_PID=7934; export SSH_AGENT_PID;
echo Agent pid 7934;
$
```

C'est déjà fini ? Oui, et en plus, ça n'a servi à rien ! :)

ssh-agent est en fait un programme qui est prévu pour être lancé en début de session, avec comme paramètre le programme initial de la session. Si vous travaillez en mode console, ce pourra être bash, si vous êtes sous X, ce sera sans doute votre gestionnaire de fenêtres.

Comme sa sortie l'indique, ssh-agent définit quelques variables d'environnement, puis lance le programme passé en paramètre. Si ce programme est celui qui va gérer la session, tous les programmes lancés par la suite seront des fils de ssh-agent et auront donc accès à ces variables d'environnement qui leur permettront de communiquer avec lui.

De nombreuses distributions lancent le gestionnaire de fenêtre avec ssh-agent. Si vous êtes logué via l'interface graphique, il y a de grandes chances qu'il soit déjà lancé. Pour vérifier :

```
$ ps x |grep ssh-agent
 7313 ?        Ss     0:00 /usr/bin/ssh-agent /usr/bin/dbus-launch
--exit-with-session x-session-manager
```

Ce résultat variera selon ce que vous utilisez, mais si vous voyez ssh-agent apparaître, c'est qu'il est déjà lancé. Si vous ne le voyez pas et que vous voulez l'essayer rapidement, vous pouvez lancer ssh-agent bash dans une console par exemple. Tous les programmes que vous lancerez à partir de cette console auront alors accès à votre agent.

Une fois ssh-agent lancé, il suffit de lui indiquer que nous voulons garder notre clé privée ouverte :

```
$ ssh-add
Enter passphrase for /home/moi/.ssh/id_dsa:
```

Entrez votre phrase de passe, et voilà ! Vous pouvez dorénavant vous connecter sur `percival` sans taper votre mot de passe utilisateur ou la phrase de passe de votre clé. Lors de la phase d'authentification par clé, le client SSH vérifie si ssh-agent est lancé et si oui, lui demande l'accès à la clé privée. Comme nous l'avons ouverte avec `ssh-add`, elle est disponible sans plus de formalités. `ssh-add` peut prendre en paramètre le fichier de la clé privée à ajouter. S'il est absent, votre clé par défaut sera ouverte. Il reconnaît également les options suivantes :

- `-D` ferme toutes les clés ouvertes ;
- `-d` ferme la clé dont le fichier est donné en paramètre ;
- `-t durée` ne garde les clés ouvertes que pendant une durée limitée (en secondes).

Par ailleurs, ssh est également capable de rediriger les requêtes d'authentification par agent. Si vous créez une chaîne de connexions ssh pour atteindre, par exemple, un hôte se trouvant dans un réseau privé, vous pouvez demander à ce que les requêtes d'authentification par agent soient redirigées jusqu'à votre connexion d'origine. Cela vous permet d'éviter d'avoir à relancer l'agent sur chaque machine de la chaîne. Pour cela, rajoutez l'option -A à vos lignes de commande ssh.

Un bon contrôle de vos clés SSHvous garantit non seulement sécurité et confidentialité, mais aussi confort d'utilisation.

SSH et les tubes

Une grande partie de la puissance du shell réside dans l'utilisation des tubes, permettant de relier la sortie d'une commande à l'entrée d'une autre. Le fait de passer par SSH ne rompt pas cette chaîne, et il est ainsi possible de brancher la sortie de commandes locales sur des commandes distantes, et vice-versa. Tout d'abord, il est possible d'utiliser SSH pour exécuter une simple commande sur la machine distante et récupérer sa sortie. Il suffit de passer la commande à exécuter après le nom d'hôte (de préférence entre guillemets pour la protéger) :

```
moi@lancelot$ ssh lui@percival "ls"
Password:
Desktop bin monfichier.txt
moi@lancelot$
```

Notez que la connexion est coupée dès que la commande distante est exécutée. Voyons maintenant comment nous pouvons tirer parti de cette forme d'appel à SSH. Un problème récurrent pour bon nombre d'entre nous consiste à faire des sauvegardes de ses données les plus importantes. Par sécurité, il est pertinent de stocker ces données sauvegardées sur une autre machine. En combinant bash et SSH, nous allons rendre cette opération possible en une seule ligne de commande qui pourra facilement être définie comme tâche périodique à l'aide de *cron*.

Lorsque l'on demande à SSH d'exécuter une simple commande distante, cette commande peut être branchée avec une commande locale. SSH redirige en effet son entrée standard vers l'entrée standard de la commande distante, et la sortie standard de la commande distante vers la sienne.

Voici la ligne permettant de faire une sauvegarde du répertoire `Documents` sur `percival` :

```
$ tar czf - Documents |ssh percival "cat >sauvegarde.tar.gz"
```

Le paramètre `-` donné à `tar` lui demande de sortir le flux compressé sur la sortie standard. Ce flux est récupéré par SSH qui le transfère sur `percival` et l'utilise comme entrée standard de `cat`. Tel qu'il est appelé, `cat` ne fait que copier son entrée standard vers le fichier `sauvegarde.tar.gz`. Ainsi, nous avons réalisé une sauvegarde distante et compressée en une seule ligne.

On peut également, si la bande passante n'est pas une ressource critique, réaliser la compression côté serveur :

```
$ tar cf - Documents |ssh percival "gzip -c >sauvegarde.tar.gz"
```

La restauration de cette sauvegarde pourra se faire de la manière suivante :

```
$ ssh percival "cat sauvegarde.tar.gz" |tar xzf
```

On peut donc faire passer les sorties standards de commandes à travers une connexion SSH. Ce serait dommage de s'arrêter là – on peut en effet y faire passer bien d'autres choses !

Tunnels SSH

Pour ce scénario, nous allons compléter notre schéma réseau : `lancelot` et `percival` ne sont pas dans le même réseau local, mais appartiennent à deux réseaux différents connectés par internet. `percival` est derrière un pare-feu qui filtre tous les ports à l'exception du port 22 (le port utilisé par SSH). Derrière `percival` se trouve `arthur`, qui est un serveur POP3 (email). À partir de `lancelot`, nous voudrions bien récupérer nos mails qui se trouvent sur `arthur`. Il y a plusieurs problèmes ici. Premièrement, `arthur` n'est pas directement accessible depuis `lancelot`. Seul `percival` a une adresse internet publique, `arthur` n'étant pas visible de l'extérieur du réseau. Deuxièmement, le pare-feu filtre tous les ports à l'exception du port 22. Or, le serveur POP3 écoute sur le port 110... SSH va nous permettre de nous affranchir de ces contraintes.

L'option `-L` de SSH permet de créer un *tunnel*. Sous ce nom se cache le fait d'employer un port de la machine locale pour transporter des données à travers la connexion SSH et les rediriger où l'on veut à partir de la machine distante. Le tunnel est également utilisable en sens inverse. Cette option est très puissante. Dans notre cas, nous voulons

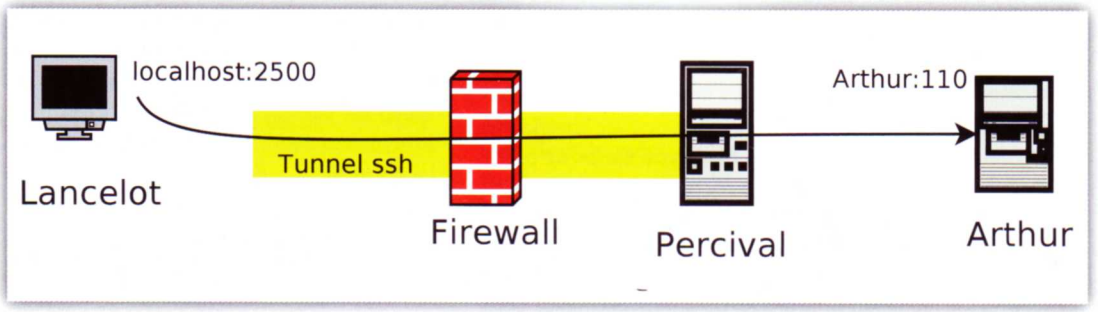


Fig. 2 : Utilisation d'un tunnel SSH pour atteindre une machine inaccessible à partir du réseau visible. Du point de vue d'arthur, ce sera percival qui se connecte, pas lancelot !

nous servir de percival comme intermédiaire pour nous connecter à arthur, comme le montre la figure 2. Nous allons ouvrir notre tunnel avec la commande suivante :

```
$ ssh -L 2500:arthur:110 percival
```

Nous demandons à SSH de nous connecter à percival. Jusque-là tout est normal. Mais en plus, l'option -L demande à rediriger le port local 2500 vers le port 110 de l'hôte arthur. Celui-ci est spécifié du point de vue de percival – rappelons qu'arthur n'est même pas visible pour lancelot. Le tunnel se fermera avec la session ssh. Du côté du client mail, il suffira de d'indiquer localhost comme serveur de mail, avec le port 2500. Et voilà ! Tant que la connexion SSH restera ouverte, le fait de nous connecter sur le port 2500 de lancelot équivaudra à se connecter sur le port 110 de arthur, et nous pourrons récupérer nos mails à partir de ce dernier, qui transiteront cryptés dans le tunnel SSH.

Intégration et transparence

Maintenant que nous connaissons les principes de SSH et que nous maîtrisons l'outil en ligne de commande, nous allons voir quelques exemples d'intégration dans des applications modernes qui rendent son utilisation plus confortable.

Intégration à KDE et Gnome

Les deux principaux environnements de bureau libres offrent la possibilité de passer par un serveur SSH pour accéder à des fichiers distants. KDE et Gnome proposent une couche d'accès à différents systèmes de fichiers (les KIO pour KDE, gnome-vfs pour Gnome) qui gèrent (entre autres) le protocole SFTP. Cette fonctionnalité très puissante mériterait un article à elle seule. Pour accéder à vos fichiers distants, il suffit de préciser un emplacement sous une forme similaire à celle utilisée pour scp, préfixé de sftp://. Cette syntaxe peut être utilisée dans n'importe quelle boîte d'ouverture/d'enregistrement de fichiers, ou tout simplement dans la barre d'URL de Konqueror ou Nautilus (figure 3). Par exemple : sftp://moi@percival/home/moi/ vous placera dans le répertoire /home/moi de percival, en vous authentifiant en tant que moi.

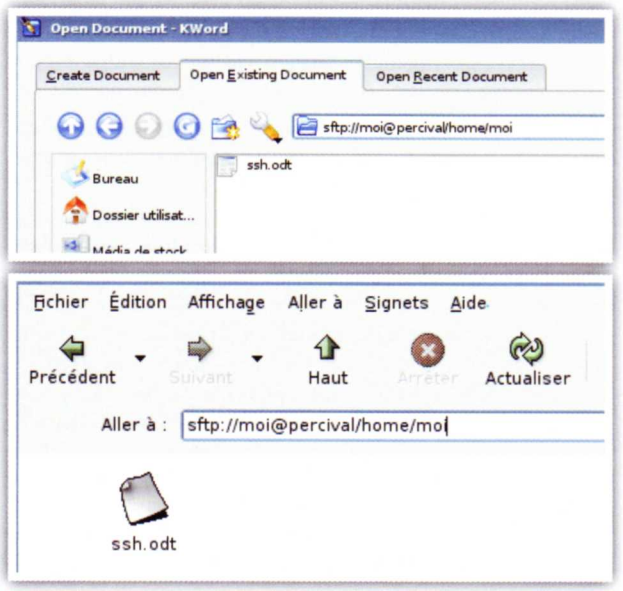


Fig. 3 : KDE et Gnome intègrent très bien l'accès à des fichiers distants par SFTP.

Si un mot de passe est nécessaire (celui du compte utilisateur distant ou de votre clé SSH), il vous sera demandé au travers d'une boîte de dialogue. KDE propose également un protocole supplémentaire : fish://. Celui-ci n'utilise pas le protocole SFTP, mais passe par une connexion SSH classique assistée d'un script perl. Il peut se révéler utile pour vous connecter à un serveur qui a désactivé le protocole SFTP.

SSHFS

Les intégrations que nous venons de couvrir sont très pratiques et puissantes, mais ne constituent néanmoins pas une solution universelle : elles sont limitées aux applications KDE ou Gnome. Et si je veux ouvrir un fichier distant avec Emacs ? Ou regarder un film avec Mplayer sans avoir à le copier préalablement ? Un autre désavantage de ces couches

d'accès est qu'elles traitent les fichiers distants comme des flux et ne supportent pas (du moins à l'heure actuelle) le déplacement. Impossible par exemple de se déplacer dans un fichier son en cours de lecture...

La solution à tous ces problèmes consiste à placer l'accès distant à un niveau plus bas que le *middleware* applicatif : au niveau du noyau. SSHFS est un système de fichiers utilisant FUSE, le module noyau permettant de développer des systèmes de fichiers au niveau utilisateur.

Beaucoup de systèmes de fichiers existent pour FUSE, comme le fameux `GmailFS`, ou ce module permettant de monter une KIO... FUSE et SSHFS sont disponibles dans certaines distributions (Ubuntu Breezy notamment). Si la vôtre ne les fournit pas, vous pouvez vous reporter sur la page du projet pour la procédure d'installation.

Attention ! Il faut en général être dans le groupe `fuse` pour pouvoir monter un système de fichiers avec ce dernier. Une fois installé, vous pouvez invoquer SSHFS comme vous invoqueriez mount :

```
$ mkdir remote
$ sshfs moi@percival:/home/moi remote
```

Cette commande montera le répertoire `/home/moi` sur la machine `percival` dans le répertoire `remote`. L'authentification SSH se fera pour l'utilisateur `moi`. Pour démonter le système de fichiers :

```
$ fusermount -u percival
```

SSHFS vous offre ainsi un accès distant totalement transparent et sécurisé et un moyen simple d'accéder à vos données de n'importe où !

Vous pouvez par exemple laisser votre collection musicale, vos albums photo ou vos documents de travail sur votre serveur personnel et les avoir toujours à disposition, pourvu que vous disposiez d'une connexion internet.

Conclusion

Nous avons rapidement découvert l'utilisation de base de SSH, ses méthodes d'authentification ainsi que le *tunneling*. L'utilisation de ce protocole comme moyen d'accès à des fichiers distants, au travers de KDE, Gnome ou FUSE, reste cependant l'utilisation la plus pratique (et aussi la plus méconnue) de ce protocole.

Particulièrement adapté à l'utilisation contemporaine des réseaux, SSH permet d'accéder à ses données personnelles de manière simple et sécurisée à partir de n'importe quelle connexion réseau.

LIENS

- Openssh : http://www.openssh.org/
- FUSE : http://fuse.sourceforge.net/

Alexandre Courbot,

http://www.gnurou.org

Port Forwarding iptables et authentification d'hôte par SSH

A chaque première connexion SSH à un hôte distant, `ssh`, sur le poste client, demande à l'utilisateur de confirmer l'identité de la machine en fournissant l'empreinte de sa clef RSA.

Dans la plupart des cas, avouez-le, vous répondez simplement `yes` sans vérifier l'empreinte. C'est une très mauvaise chose puisque, si un attaquant est déjà en lice, vous pourriez littéralement accepter une attaque *Man-In-the-Middle* et toutes celles qui suivront. L'acceptation entraîne l'enregistrement de la clef publique dans un fichier `~/.ssh/known_hosts`.

Celle-ci sera vérifiée à chaque connexion par la suite. Encore récemment, les entrées dans ce fichier débutaient par l'adresse IP ou le nom de la machine distante.

A présent, cette information est chiffrée. Un problème survient lorsque, pour une même adresse IP, nous avons plusieurs hôtes serveur SSH à contacter. Le cas typique est celui d'une passerelle/NAT faisant fonctionner un serveur SSH sur le port 22 mais faisant suivre son port 2222 sur le port 22 d'une machine du LAN.

Ainsi, en utilisant l'option `-p` de `ssh` on accèdera soit à la passerelle, soit à la machine du LAN. Cependant, dans `~/.ssh/known_hosts`, une seule clef sera stockée et un message d'alerte sera affiché si la vérification échoue.

Il existe une solution permettant de régler le problème et de différencier les deux hôtes. Il suffit pour cela de créer des profiles dans votre `~/.ssh/config` :

```
Host gateway
        Hostname truc.dom.fr
        CheckHostIP no
        Port 22
        HostKeyAlias gateway
```

```
Host mamachine
        Hostname truc.dom.fr
        CheckHostIP no
        Port 2222
        HostKeyAlias mamachine
```

On se connectera alors en utilisant `ssh gateway` ou `ssh mamachine`. C'est `HostKeyAlias` qui permet de créer des entrées différentes dans `known_hosts`. Notez également la directive `CheckHostIP` annulant la vérification IP pour une passerelle ADSL se voyant attribuer une adresse dynamiquement.

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

BitlBee : Lorsque IRC communique avec MSN

Christophe Buffenoir

EN DEUX MOTS **Les messageries instantanées permettent de communiquer aisément avec un correspondant. Cependant, divers protocoles existent dont MSN, ICQ, AIM, Jabber ou encore Yahoo Messenger. Ces cinq réseaux facilitent les communications de deux personnes. Ils ne sont pas prévus pour des groupes de personnes souhaitant parler d'un sujet. Pour cela, il existe encore un autre protocole : IRC. Il serait donc utile de pouvoir utiliser l'ensemble de ces services avec un même et unique logiciel. La méthode classique consiste à utiliser une application spécialisée et compatible avec l'ensemble. Nous verrons dans cet article comment utiliser IRC et seulement IRC pour parler à une personne d'un de ces réseaux.**

BitlBee est une passerelle entre IRC et les grands protocoles de messagerie instantanée (MSNP pour MSN, Oscar pour AIM et ICQ, Yahoo Messenger Protocol pour Yahoo, et enfin XMPP pour Jabber et maintenant Google Talk). Des questions diverses sont régulièrement posées aux développeurs, comme pour tout logiciel sortant de l'ordinaire. Régulièrement, des personnes demandent pourquoi elles ne voient pas les autres utilisateurs dans le *channel*. La confusion avec un serveur IRC est rapidement réalisée.

Wilmer van der Gaast était plus habitué à utiliser un client IRC complet qu'un logiciel spécialisé en messagerie instantanée. Il voulait donc communiquer sur MSN par IRC. Peu de logiciels existaient pour cela. Un support de ICQ était disponible pour irssi, mais il ne gérait que l'ancienne version du protocole. Alors, en 2002, il écrit la première version de BitlBee. Mais d'où vient ce nom si spécial ? Il s'agit d'un mélange de « het Bijtje » et de « the little bee ». Ces deux expressions signifient « la petite abeille ». La première est en néerlandais et la seconde en anglais. L'abeille a deux couleurs distinctes, ce qui, ici, représente deux mondes différents : IRC et la messagerie. Le but de Wilmer n'était pas de réinventer la roue. Il a donc préféré prendre du code existant, à savoir celui de Gaim. Mais de lourds changements de ce côté vont intervenir. En effet, la partie gérant Yahoo se sert désormais de la bibliothèque `libyahoo2`. Le module MSN, quant à lui, a tout simplement été réécrit. Il reste donc quelques parcelles de code de Gaim. Ne vous affolez donc pas si vous voyez une erreur du programme Gaim. Il s'agit bel et bien de BitlBee. Bien sûr, le développement ne s'est pas déroulé sans difficulté. Un bug important a empêché de sortir la version 1.0. Il a été corrigé pour la version 0.99. Les anciennes versions de BitlBee peuvent alors se bloquer complètement et ne pas répondre aux requêtes. Dans tous les cas, il faut le tuer avec la commande `kill` pour reprendre la main. Ceci oblige à utiliser inetd. Le problème provenait d'une erreur sur un contrôle d'erreur.

Et si nous l'installions ?

Vous trouverez BitlBee directement sur le site web **http://www.bitlbee.org**. Il existe deux versions majeures : `bitlbee 0.92` avec des patchs pour MSN et `bitlbee 1.0.1` ([NDLR : Pour Debian, 0.92 est en stable alors que 1.0.1 est encore en *unstable*. Solution intermédiaire, 1.0 dans *testing*). La première nécessite un patch fourni à part pour Jabber et n'est pas considérée comme stable. Les extensions MSN ne sont pas très appréciées par Wilmer. Il recommande donc la dernière version, à jour, avec des bugs importants en moins. Cet article décrit l'installation de la version 0.92. Le procédé est le même pour la 1.0.1 avec l'application du patch en moins.

BitlBee a des dépendances. Il faut en effet une bibliothèque SSL. Vous avez le choix entre GnuTLS, OpenSSL ou NSS. GnuTLS est recommandée pour sa licence, mais elle est lourde de dépendances. Il vaut mieux l'installer par un paquet prêt à l'emploi plutôt que de la compiler. OpenSSL est beaucoup plus simple à mettre en œuvre mais ne permet pas d'avoir un exécutable distribuable. En effet, il y a une incompatibilité de licence. NSS combine l'avantage des deux.

Cette bibliothèque est livrée avec la suite Mozilla, Firefox ou encore Thunderbird. Elle est cependant susceptible de changer lors d'une mise à jour du navigateur ou du client mail. Cet article montre l'installation avec OpenSSL déjà installé sur le serveur. Bien sûr, vous aurez également besoin d'un client IRC.

Il en existe une multitude, que ce soit en graphique (XChat, Konversation, etc.), pour un navigateur web (l'*applet* PJIRC entre autres) ou encore en ligne de commande (irssi, ircii, etc.). Le plus connu du grand public est entièrement propriétaire et payant. Il s'agit de mIRC. Ce dernier n'est disponible que pour Windows.

La première étape consiste à décompresser le fichier `bitlbee-0.92-msn6-akke.tgz`. Il contient la version stable de BitlBee et les patchs MSN. Vous le trouverez en téléchargement dans la partie *MSN extensions* du site web. Il possède de bons avantages par rapport à la version standard comme la gestion des avatars et la modification de la police d'affichage.

Afin de réaliser une installation « propre », la décompression se fait dans le répertoire /usr/src.

```
tar -xzvf bitlbee-0.92-msn6-akke.tgz -C /usr/src
```

Pour bénéficier du support de Google Talk, il faut également appliquer le patch bitlbee-jabberserver.

```
cp bitlbee-jabberserver.patch /usr/src/bitlbee-0.92-msn6-akke/
cd /usr/src/bitlbee-0.92-msn6-akke/
patch -p1 < bitlbee-jabberserver.patch
```

Lors de l'utilisation, le contrôle du logiciel se fait dans un channel appelé #bitlbee. Le système est représenté par l'utilisateur *root*. Il est possible de changer ces deux valeurs avant la compilation. Pour cela, il suffit d'éditer le fichier bitlbee.h et de changer les deux lignes suivantes :

```
#define ROOT_NICK "root"
#define ROOT_CHAN "#bitlbee"
```

Ici, nous mettrons le nom de l'ordinateur comme utilisateur de contrôle, à savoir Elrond. Il est alors possible de passer à la compilation proprement dite. Les modules Jabber, Oscar, MSN et Yahoo seront activés. Tapez simplement la commande suivante pour procéder à la configuration et à la compilation.

```
./configure --prefix=/usr --etcdir=/etc/bitlbee --ssl=openssl && make
```

Contrairement à d'autres logiciels, l'installation est décomposée en deux. La première commande ci-dessous s'occupe des binaires et la deuxième de la configuration.

Cela permet de conserver des fichiers existant lors d'une mise à jour. Enfin, le répertoire /var/lib/bitlbee doit être créé à la première installation. Il contiendra l'ensemble des données des utilisateurs.

```
make install
make install-etc
mkdir -p /var/lib/bitlbee
chown -R bitlbee:daemon /var/lib/bitlbee
```

BitlBee ne doit pas s'exécuter avec l'utilisateur root. D'ailleurs, ceci devrait être le cas de l'ensemble des serveurs, que ce soit pour le web, pour IRC ou pour autre chose. Pour un serveur en production, il aurait même été préférable d'enfermer BitlBee dans un *chroot*.

La configuration qui va suivre n'est pas idéale non plus. En effet, en raison du caractère instable de certaines versions de BitlBee, le serveur sera lancé par inetd ou xinetd.

Or, ces deux derniers démons sont souvent désactivés sur les serveurs pour limiter le nombre de failles.

La commande ci-dessous crée l'utilisateur bitlbee du groupe daemon. En attribuant le *shell* /bin/false, il devient impossible de se connecter avec cet utilisateur.

```
useradd -g daemon -s /bin/false bitlbee
```

Le fichier /etc/bitlbee/motd.txt contient le texte affiché à la connexion sur le serveur. Il est tout à fait possible d'ajouter une citation aléatoire. Pour cela, il suffit de créer un petit script et de l'ajouter à la configuration de cron.

Le fichier /etc/bitlbee/bitlbee.conf permet de modifier la configuration. Les valeurs initiales conviennent parfaitement, il est donc inutile de l'éditer. Il ne reste plus qu'à ajouter le service bitlbee dans /etc/services et à l'enregistrer dans inetd (ou xinetd selon la distribution Linux).

Sur le serveur de test, bitlbee s'exécutera sur le port 6665. Normalement, il devrait être sur le 6667, mais un serveur IRC classique l'utilise déjà. Il faut donc éditer le fichier /etc/services et ajouter la ligne suivante:

```
bitlbee 6665/tcp #Passerelle IRC / IM
```

Il reste à configurer le lancement du serveur. Il faut donc savoir si vous utilisez inetd ou xinetd. Pour cela, utilisez simplement la commande ps comme ceci :

```
ps -aux | grep inet
```

Avec inetd

Si inetd est en fonctionnement, la ligne ci-dessous doit être ajoutée au fichier /etc/inetd.conf.

```
bitlbee   stream   tcp   nowait   bitlbee   /usr/sbin/bitlbee
```

Il ne reste alors qu'à relancer inetd ainsi :

```
pkill -HUP inetd
```

Avec xinetd

Pour l'utilisation de xinetd, créez le fichier /etc/xinetd.d/bitlbee. Il contiendra le code suivant :

```
service bitlbee
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = bitlbee
        server          = /usr/sbin/bitlbee
}
```

L'installation est alors terminée. Vous allez enfin pouvoir profiter de votre nouveau serveur.

La configuration

Tout d'abord, il faut ouvrir un client IRC. Quelques *plugins* pour irssi sont disponibles sur le site de BitlBee. Ils permettent de changer certains messages en notices, de faire une recherche sur les contacts ou encore de gérer les avertissements de frappe. Lancez donc votre client IRC à l'adresse de votre serveur et sur le port 6665. Bien sûr, il est possible de le faire par la commande `/connect`.

Un channel `#bitlbee` s'ouvre. Le maître Elrond vous souhaite alors la bienvenue. Le channel en question vous permettra de taper les commandes. La première à apprendre est bien sûr `help`. Plus de précisions sont données sur un point précis avec l'utilisation d'un paramètre. La liste des commandes s'affiche avec `help commands`.

Une gestion de session existe. Ainsi, lorsque vous quitterez le client IRC, vous pourrez récupérer tous les paramètres. Il faut cependant s'enregistrer. Pour cela, il suffit de taper `register un_mot_de_passe`. Il vous faudra vous identifier par la commande `identify le_mot_de_passe` à la prochaine connexion. Les paramètres seront alors chargés.

Il est désormais temps de configurer les comptes. Les étapes indiquées ci-dessous impliquent que vous ayez un compte dans les services correspondant. MSN sera largement traité avec les extensions. D'ailleurs, pour configurer un compte, il suffit juste d'utiliser la commande `account add`. Les six lignes suivantes montrent respectivement la création des comptes MSN, AIM, ICQ, Yahoo Messenger, Jabber.org, et Google Talk.

```
account add msn buffenc@yahoo.fr mot_de_passe
account add oscar buffenc mot_de_passe login.oscar.aim.com
account add oscar 319800069 mot_de_passe login.icq.com
account add yahoo buffenc mot_de_passe
account add jabber buffenc mot_de_passe jabber.org
account add jabber buffenc@gmail.com mot_de_passe talk.google.com:5222:ssl
```

Comme vous avez pu le voir ci-dessus ou dans l'aide via la commande `help account add`, le premier paramètre correspond au protocole utilisé. Le second est l'identifiant sur ce service, le troisième est le mot de passe et le quatrième le serveur. Pour Google Talk, le port doit être mis à 5223 chez certaines personnes. À chaque ajout de compte, Elrond doit répondre `"Account successfully added"`. Vous obtiendrez la liste de tous les comptes par `account list`. Cette dernière commande est très importante. En effet, lors de l'ajout d'un contact, il est nécessaire de préciser le numéro du compte concerné.

Pour lancer les connexions, tapez simplement `account on`. Vous pourrez également vous déconnecter d'un protocole particulier avec `account n off` où `n` est le numéro du compte. La commande `save` permet de sauvegarder la configuration.

BitlBee offre également d'autres options intéressantes. Il suffit d'utiliser la commande `set` sans paramètre pour les lister. La modification d'une valeur s'effectue en tapant `set paramètre valeur`.

Par exemple, pour ne pas se connecter automatiquement après l'identification, ce sera `set auto_connect "false"`. Le paramètre `auto_reconnect` détermine si BitlBee se reconnecte automatiquement après une déconnexion. Le délai entre deux tentatives est défini par `auto_reconnect_delay`. L'option `charset` est importante.

Elle correspond à l'encodage des caractères. La valeur par défaut est `iso-8859-1`. Cela signifie que le caractère « euro » n'est pas disponible. Il vous faudra donc changer la valeur en `iso-8859-15` ou en `utf-8` selon la configuration de votre client IRC. Si vous voyez du code HTML apparaître dans les messages de votre correspondant, il faudra passer l'option `html` à `strip`.

Lorsque `away_devoice` est activé, les contacts absents n'ont pas l'attribut `voice`. Ce dernier est généralement noté par le signe « + » devant le pseudo. Si l'option `private` a pour valeur `true`, une fenêtre est ouverte lorsqu'un contact vous parle. Dans le cas contraire, la discussion débute dans le channel `#bitlbee`. Il faudra alors utiliser la commande `/msg` du client IRC ou encore utiliser la souris pour ouvrir la discussion privée. Une autre solution consiste tout simplement à parler depuis le channel.

Dans ce cas, vous devrez donner le nom du contact au début de chaque message. Une chaîne de caractères doit être ajoutée juste après le pseudo. Elle est définie par l'option `to_char`. Bien sûr, elle est à définir en fonction du client IRC. Il s'agit en fait des caractères ajoutés automatiquement par la complétion de la touche de tabulation. Par exemple, pour parler à Arnaud, il suffit d'écrire `Arnaud: mon message` si `to_char` vaut « : ».

Avec la messagerie instantanée, il est possible d'écrire plusieurs lignes à la suite. Ce n'est pas le cas d'IRC. Pour cela, l'option `buddy_sendbuffer` permet d'envoyer les lignes écrites durant le délai `buddy_sendbuffer_delay` d'un bloc. Les options commençant par `msn_notify_` activent ou désactivent les notices sur certains événements comme l'ouverture ou la fermeture de la fenêtre.

Vous pourrez modifier la police utilisée par MSN avec tous les paramètres `msn_font_*`. Pour finir sur la commande `set`, les extensions MSN comprennent trois autres valeurs. `msn_images_path_buddy` et `msn_images_path_emoticon` sont les répertoires où seront respectivement enregistrés les avatars et les *émoticônes*. Votre avatar est défini par `msn_images_mybuddyimage`. L'image sera obligatoirement au format PNG.

Abonnez-vous !

11 Numéros de Linux Magazine

1 an de bonne lecture, bien UNIX, bien technique... Bref...

LES 3 BONNES RAISONS DE VOUS ABONNER !

- Ne manquez plus aucun numéro
- Recevez Linux Magazine chaque mois chez vous, ou dans votre entreprise
- Economisez 12,45 €/an ! (soit 2 magazines offerts !)

➥Des offres de couplage sont disponibles
➥Retrouvez les Tarifs étrangers hors France Métro sur www.ed-diamond.com

BON D'ABONNEMENT À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*LINUX MAGAZINE - BP 20142 - 67603 SELESTAT CEDEX

11 Numéros de Linux Magazine à 53€ Offre France Métro

Soit **Votre Linux Magazine à 4,82€** (Tarif au numéro dans le cadre d'un abonnement France Métro)

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

LES 4 FAÇONS DE VOUS ABONNER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

○ Oui, je souhaite m'abonner à Linux Magazine pour 11 numéros

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

☐ **Pour avoir un suivi par e-mail de vos abonnements, merci de nous indiquer votre adresse e-mail** :

**En application des articles 27 et 34 de la loi dite «Informatique et libertés» n° 78-17 du 6 janvier 1978, vous disposez d'un droit d'accès et de rectification aux données vous concernant.

Offres collectionneurs

Les anciens numéros !

TOUJOURS DISPONIBLES !

LES 4 FAÇONS DE COMMANDER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

BON DE COMMANDE À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*LINUX MAGAZINE - BP 20142 - 67603 SELESTAT CEDEX

Bon de commande Linux Magazine

Référence	Prix / N°s	Qté.	Total
Linux Magazine 72	6,40 €		
Linux Magazine 73	6,40 €		
Linux Magazine 74	6,40 €		
Linux Magazine 75	6,40 €		
Linux Magazine 76	6,40 €		
Linux Magazine 77	6,40 €		
Linux Magazine 78	6,40 €		
Linux Magazine 79	5,95 €		
Linux Magazine 80	5,95 €		
Linux Magazine 81	5,95 €		

Bon de commande Linux Magazine Hors Série

LM HS 08 Introduction à la crypto	5,95 €		
LM HS 09 Installer son serveur Web à la maison	5,95 €		
LM HS 10 Installation de votre serveur internet	5,95 €		
LM HS 11 GIMP par la pratique	5,95 €		
LM HS 12 Firewall votre meilleur ennemi Acte 1	5,95 €		
LM HS 13 Firewall votre meilleur ennemi Acte 2	5,95 €		
LM HS 14 Maitrisez Blender	5,95 €		
LM HS 15 GIMP et la Photo	5,95 €		
LM HS 16 Kernel (1)	5,95 €		
LM HS 17 Kernel (2)	5,95 €		
LM HS 18 Haute Disponibilité	5,95 €		
LM HS 19 The Gimp 2.0	5,95 €		
LM HS 20 PHP 5	5,95 €		
LM HS 21 Recyclez vos PC	6,40 €		
LM HS 22 GIMP et le Web	6,40 €		
LM HS 23 Linux et electronique	6,40 €		
	sous TOTAL		
	Frais de port France Metro		+ 3,81 €
	Frais de port Etranger		+ 5,34 €
	TOTAL		

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

GNU LINUX MAGAZINE / FRANCE

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

L'art de communiquer

Bien sûr, pour communiquer, il faut avoir des contacts. Pour cela, il faut les ajouter à la liste du protocole. Vous n'avez pas besoin de vous occuper d'importer ou d'exporter quoi que ce soit. En effet, les contacts sont enregistrés sur le serveur et non pas sur le client. L'ajout se fait par la commande `add` ainsi :

```
add 1 mon_contact
```

Le 1 correspond au numéro du compte. Pour le connaître, il faut utiliser la commande `account list`. Bien sûr, un jour, vous pourrez avoir envie de supprimer un de vos contacts. BitlBee gère cela très bien.

```
remove mon_contact
```

La même syntaxe est appliquée aux commandes `block` et `allow`. La première bloque un contact. Ce dernier ne pourra plus vous parler sans que vous ne commenciez la discussion. La seconde commande annule cet état et réautorise le contact. Enfin, si vous souhaitez renommer une personne, procédez ainsi :

```
rename mon_contact nouveau_nom
```

Cependant, certaines opérations nécessitent l'accord du contact concerné. Il s'agit principalement de l'ajout. De même, vous devrez autoriser ou interdire l'ajout de votre adresse chez une autre personne. La question est tout simplement posée dans le channel `#bitlbee`. Il faudra alors répondre par `yes` ou par `no`. Par exemple, pour une connexion IRC, la question peut être celle-ci :

```
Question on ICQ connection (handle 9876543210):
The user 0123456789 wants to add you to their buddy list for the following reason:

No reason given.
You can use the yes/no commands to answer this question.
```

La liste des questions est disponible via la commande `qlist`. Il manque cependant une notion importante. Comment pouvons-nous discuter avec ce système ? Pour entamer une discussion, ouvrez tout simplement une fenêtre avec le contact. Si ce dernier a l'attribut `voice`, il est présent. Dans le cas contraire, il est absent. Un contact déconnecté n'est tout simplement pas présent sur le channel `#bitlbee`.

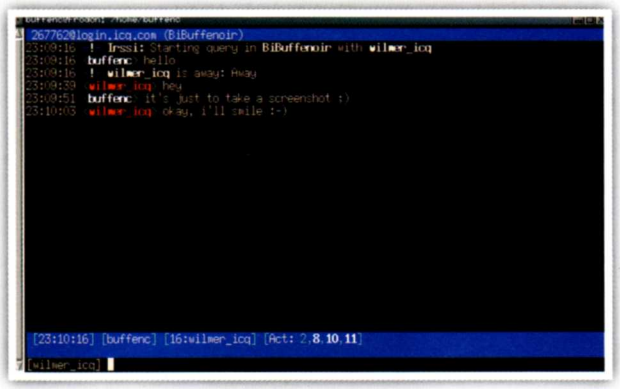


Fig. 1 : Une petite discussion avec Wilmer sur ICQ, en utilisant irssi.

La commande `blist` vous permettra de lister les différents contacts. Une autre solution permet d'envoyer des messages. En effet, il est tout à fait possible de parler depuis le channel `#bitlbee`. Dans ce cas, vous devez faire précéder chacun des messages du pseudo et de la chaîne `to_char`.

Certains protocoles gèrent les discussions privées. Bien sûr, tous les contacts impliqués dans une telle fonctionnalité ne peuvent être sur deux comptes différents. Les channels correspondant portent le nom `#chat_xxx`. Les trois `x` représentent un nombre à trois chiffres. Pour créer un groupe de discussion, il faut d'abord rejoindre le channel portant le même nom que le premier contact. Cela se réalise avec la commande `/join #mon_contact`. Une erreur sera reportée par le client IRC. En effet, le channel demandé n'existe pas. Mais BitlBee s'occupe de créer le groupe de discussion virtuel. Il suffit alors d'inviter les contacts à ajouter. Pour cela, vous pouvez utiliser la commande `/invite mon_autre_contact #chat_xxx`.

Ce système de communication a de lourds défauts. Vous ne pourrez pas utiliser la webcam ou encore transférer des fichiers. Mais, en contrepartie, il est possible d'utiliser diverses autres fonctionnalités. En effet, le couplage avec un client IRC permet d'utiliser une multitude de scripts. Ainsi, vous pourrez facilement connaître le temps depuis lequel un contact s'est déconnecté ou encore envoyer des citations. L'ensemble des tests a été réalisé avec irssi.

Ce client IRC en console dispose de nombreux atouts. Par exemple, il peut être utilisé via SSH. Cela permet de garder la même configuration pour chaque utilisation. Mieux encore, un utilitaire nommé screen permet de garder des applications en console ouverte et de récupérer la sortie.

Ainsi, il vous suffit de lancer screen dans une connexion SSH sur votre serveur. Exécutez alors irssi. Lorsque vous devrez changer d'ordinateur, il suffira de détacher l'affichage avec la combinaison de touches [Ctrl]+[D].

Tapez alors `screen -r` à la prochaine connexion SSH. Vous verrez apparaître irssi avec toutes les discussions ouvertes. Quant aux avatars MSN, rien ne vous empêche de les enregistrer dans un répertoire utilisé par Apache. Ils seront visibles avec un simple navigateur web.

Christophe Buffenoir,

http://www.buffenoir.org

Exploration du module Scene

Olivier Saraja

EN DEUX MOTS **Dans cet article, nous allons approfondir notre connaissance du module Scene. Nous en avons déjà fait usage dans le passé, notamment pour lier des objets à la scène courante, mais ce module nous réserve de nombreux autres usages, comme nous allons le découvrir.**

Jusqu'à présent, nous avons fait appel au module `Scene` à chaque fois que nous avons souhaité insérer un objet dans notre scène au travers d'un script Python. La démarche était alors la suivante, par exemple, pour la création d'une nouvelle caméra :

- (1) création d'un nouveau bloc de données :

```
cam = Camera.New('persp')
```

- (2) récupération de la scène courante :

```
scn = Scene.GetCurrent()
```

- (3) création d'un nouvel objet :

```
camera = Object.New('Camera')
```

- (4) établissement de la liaison entre le bloc de données de caméra et l'objet caméra :

```
camera.link(cam)
```

- (5) établissement d'une liaison entre l'objet caméra et la scène courante :

```
scn.link(camera)
```

La procédure est la même pour l'insertion d'un objet de type `lampe`, d'un maillage ou de n'importe quel autre type d'objet : elle fait systématiquement appel au module `Scene`, qui en devient pratiquement incontournable.

Mais ce module couvre d'autres applications, très différentes, de la duplication de scène à la gestion des calques actifs, en passant par la possibilité de piloter le rendu de vos scènes ou les réglages du moteur de radiosité.

Mais nous découvrirons tout cela très progressivement.

ATTENTION

Dans cet exemple, nous avons respecté une certaine logique hiérarchique dans la structure des données, et en particulier, nous noterons que la liaison de l'objet à la scène est établie en dernier lieu. En effet, en supposant que nous aurions d'abord établi une liaison entre un objet et la scène, nous aurions immédiatement généré un bloc de données vide pour garnir cet objet. Au moment de l'établissement de la liaison entre bloc de données et objet, le bloc spécifié va écraser le bloc vide, mais celui-ci restera quelque part dans la mémoire de Blender, l'encombrant légèrement. Si cela n'a certainement aucune incidence lors de la création et de l'usage de scripts simples, cela peut avoir une forte incidence sur la mémoire utilisée par Blender lors de la génération automatique d'un grand nombre d'objets (un script de génération automatique de gazon, par exemple ?). Respecter cet ordre de liaison garantit donc une réservation optimale de la mémoire.

1. Le Module Scene

Comme tout autre module, le module `Scene` doit être importé, généralement dans la deuxième ligne de votre script:

```
01:     import Blender
02:     from Blender import Scene
03:     ...
```

Il s'agit du premier module dont nous approfondissons l'étude qui propose également des sous-modules : `Radio` (pour la simulation d'un illumination par solution de radiosité) et `Render` (pour contrôler les paramètres de rendu d'une scène). Optionnellement, nous importerons ces deux sous-modules grâce à une ligne de type :

```
03:     from Blender.Scene import Radio, Render
04:     ...
```

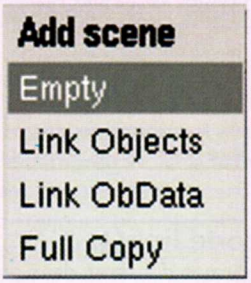
Le module `Scene` propose quatre fonctions essentielles, permettant de récupérer les données relatives à une scène existante, à récupérer la scène courante, à créer une nouvelle scène, et enfin à supprimer une scène.

NOTE

Un fichier Blender peut contenir plusieurs scènes ; chaque scène peut être strictement indépendante l'une de l'autre, ou bien partager certains éléments comme des blocs de données ou des objets.

Dans Blender, la gestion des scènes est simple et très limitée. Dans la barre de menu principale se trouve un bouton menu SCE portant le nom de la scène courante (par défaut : `Scene`). Sur sa gauche, se trouve un petit ascenseur permettant de naviguer entre les différentes scènes existantes, ou d'en créer de nouvelles grâce à l'option `ADD NEW`.

Dans ce dernier cas, Blender propose différents types de scènes : `Empty` crée simplement une nouvelle scène, totalement vierge ; les autres options seront détaillées lors de la description de la méthode `copy()` en 2.1.



Les options d'ajout d'une scène dans Blender

1.1 La fonction Get():

Cette fonction permet de récupérer une scène particulière en utilisant son nom, ou bien de récupérer la liste de toutes les scènes. Par exemple :

```
scn = Scene.Get()
print scn
```

permet d'imprimer dans la console la liste de toutes les scènes disponibles :

```
[[Scene "Scene"], [Scene "Scene.001"]]
```

et d'en récupérer ensuite une en particulier. Si vous connaissez le nom de la scène qui vous intéresse, vous pouvez également récupérer celle-ci simplement en la nommant :

```
scn = Scene.Get('Scene.001')
```

1.2 La foncion GetCurrent():

Très similaire à la fonction précédente, celle-ci permet de récupérer la scène couramment active. Par exemple :

```
scn = Scene.GetCurrent()
```

1.3 La fonction New([nom]):

Cette fonction permet de créer une nouvelle scène et de lui donner un nom. Par exemple :

```
scn = Scene.New('NouvelleScene')
```

créera une nouvelle scène sobrement nommée... `NouvelleScene` !

1.4 La fonction Unlink([nom]):

Celle-ci agit à l'opposé de la précédente, car au lieu de créer une scène nommée, elle l'efface ! Attention à sa syntaxe particulière, car elle n'admet pas comme argument un nom de scène. Il vous faudra donc passer par une variable ou une fonction pour spécifier la scène à effacer. Par exemple :

```
scn_del = Scene.Get('NouvelleScene')
scn = Scene.Unlink(scn_del)
```

Une formulation plus économe et strictement équivalente serait :

```
scn = Scene.Unlink(Scene.Get('NouvelleScene'))
```

2. Les méthodes propres au module Scene

2.1 La méthode copy():

Cette méthode permet de produire une copie d'une scène. L'argument passé à la méthode `copy()` permet de déterminer la façon dont les objets « enfants » de la scène sont dupliqués.

L'argument `0` lie les objets aux objets originaux; l'argument `1` lie les blocs de données aux blocs de données des objets originaux ; enfin, l'argument `2` crée une copie intégrale et indépendante de la scène.

Par exemple, supposons et exécutons le script suivant :

```
01:  import Blender
02:  from Blender import Scene
03:  scn = Scene.GetCurrent()
04:  print scn
05:  scn0 = scn.copy(0)
06:  scn0.setName('LinkObjects')
07:  scn1 = scn.copy(1)
08:  scn1.setName('LinkObjectData')
09:  scn2 = scn.copy(2)
10:  scn2.setName('FullCopy')
11:  scn = Scn.Get()
12:  print scn
```

Nous avons maintenant quatre scènes : `Scene`, `LinkObjects`, `LinkObjectData` et `FullCopy`. Pour comprendre la différence entre ces différents modes de copie, faites l'expérience suivante : dans Blender, assurez-vous d'être dans la scène d'origine nommée `Scene`. Éditez ensuite le cube (touche [TAB]) et déplacez un ou plusieurs de ses points de contrôle (par exemple grâce à la touche [G]).

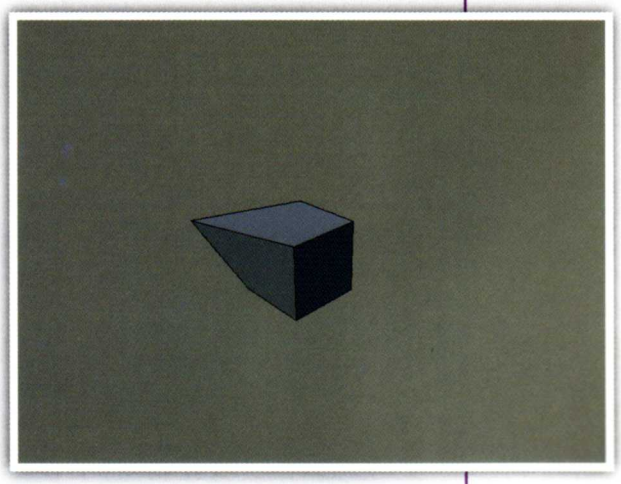


Fig. 1 : En mode édition, nous modifions le maillage du cube de la scène d'origine

Une fois ceci fait, sortez du mode édition, et redimensionnez (touche [S]) le cube d'un facteur 2.

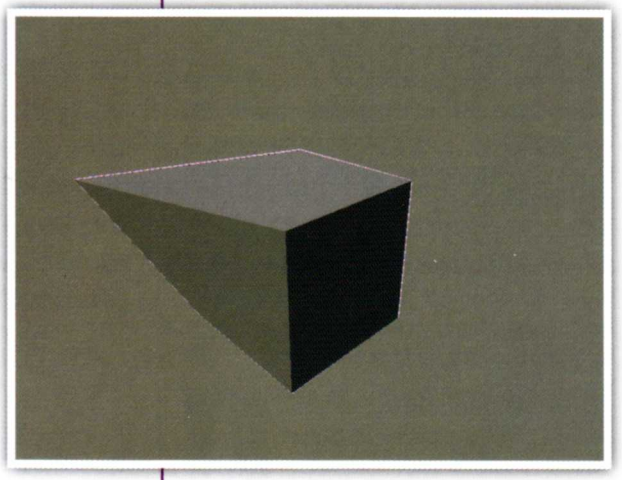


Fig. 2 : Hors du mode édition, nous redimensionnons le « cube » ainsi retouché.

Basculez successivement de scène en scène. Vous constaterez que dans la scène `LinkObject`, le cube a suivi à la fois le changement de forme et le changement de dimension que celui de la scène d'origine : l'objet de la nouvelle scène est directement lié à celui de la scène d'origine.

Dans la cène `LinkObjectData`, seul le changement de forme a suivi celui de la scène d'origine : le cube de la nouvelle scène et celui de la scène d'origine partagent le même bloc de données, et donc le même maillage, mais les transformations géométriques (déplacement, rotation, dimensions) appliquées à l'objet (hors du mode édition, donc !) ne sont pas transmises.

Enfin, dans la scène `FullCopy`, le cube n'a suivi ni la modification du maillage, ni les transformations géométriques; cette scène est strictement indépendante de la scène d'origine, aucune modification ou transformation n'a été transmise de scène à scène.

2.2 La méthode makeCurrent():

Cette méthode permet de déterminer la scène courante, par exemple dans le cas où, par l'usage d'autres méthodes ou fonctions, vous créez ou récupérez de multiples scènes. Par exemple, nous pourrions continuer le petit script précédent par les lignes :

```
13:     scn = Scene.Get('LinkObject')
14:     scn.makeCurrent()
```

pour faire de la scène `LinkObject` la scène couramment active de Blender.

2.3 La méthode link():

Cette méthode permet d'insérer dans une scène un objet en provenance d'une autre scène ; l'objet importé est lié à l'objet d'origine : les modifications de maillage comme les transformations géométriques seront répercutées d'une scène à l'autre. Après avoir créé de multiples scènes comme dans le paragraphe 2.1 au sujet de la méthode `copy()`, dans Blender, retournons dans la scène d'origine `Scene` et ajoutons une `UVSphere`. Créons ensuite un nouveau script (dans la fenêtre d'édition de texte : *File* > *New*) et saisissons le code suivant :

```
01:     import Blender
02:     from Blender import Scene, Object
03:     scn = Scene.Get('Scene')
04:     scn.makeCurrent()
05:     ob = Object.Get('Sphere')
06:     scn = Scene.Get('FullCopy')
07:     scn.makeCurrent()
08:     scn.link(ob)
```

Lorsque nous l'exécuterons, le script basculera automatiquement dans la scène d'origine `Scene` si nous n'y étions pas déjà, récupérera l'objet nommé `Sphere` sous le nom de variable `ob`, rebasculera dans la scène `FullCopy`, et grâce à la méthode `link()`, liera l'objet `ob` à la scène.

2.4 La méthode unlink():

A l'opposé de la méthode précédente, celle-ci supprime un objet désigné de la scène courante. Pour poursuivre sur notre exemple précédent, ajoutons les lignes suivantes au script précédent :

```
09:     scn = Scene.Get('Scene')
10:     scn.makeCurrent()
11:     ob = Object.Get('Cube')
12:     scn.unlink(ob)
```

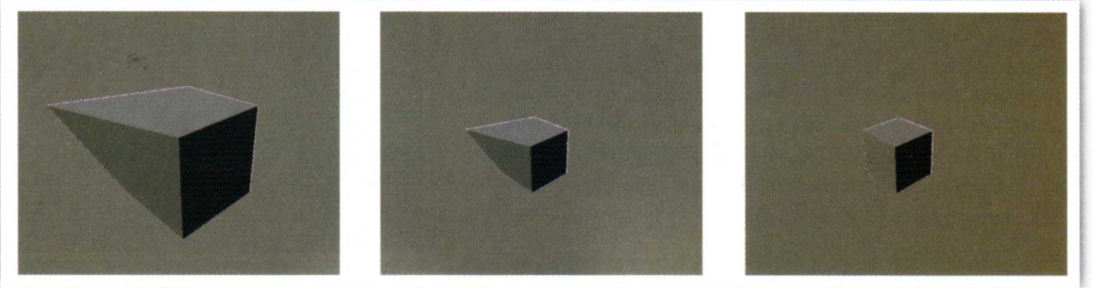


Fig. 3 : Respectivement, la scène LinkObjects (strictement identique à la scène Scene), la scène LinkObjectData (le maillage est identique, mais pas les dimensions de l'objet) et enfin la scène FullCopy (qui ne partage ni bloc de données, ni objet, avec la scène d'origine)

Lorsque nous l'exécutons, ce script agit à l'identique du précédent, à l'exception qu'il retourne dans la scène d'origine `Scene`, qu'il sélectionne l'objet nommé `Cube`, et le supprime de la scène d'origine. En conclusion, nous n'avons plus que la sphère dans la scène d'origine, tandis que dans la scène `FullCopy`, nous avons le cube et la sphère, récupérée de la première scène.

2.5 La méthode getActiveObject():

Cette méthode permet de récupérer l'objet actif de la scène. Il est important de comprendre ici la différence entre objet **actif** et objet **sélectionné**. Même quand aucun objet n'est sélectionné, il y a toujours un objet actif ; généralement, il s'agit du dernier objet sélectionné, ou du dernier maillage ou objet pour lequel vous seriez entré en mode édition.

REMARQUE

La méthode `Object.GetSelected()[0]` permet de récupérer l'objet actif si celui-ci est sélectionné : l'objet actif apparaît en tête de liste. Il y a donc une différence fonctionnelle fondamentale entre les méthodes `getActiveObject()` du module `Scene` et `getSelected()` du module `Object`.

Par exemple :

```
01:     import Blender
02:     from Blender import Scene
03:     scn = Scene.getCurrent()
04:     active = scn.getActiveObject()
```

vous montre comment récupérer simplement le dernier objet actif et en stocker la référence dans la variable nommée `active`. Il s'agira bien sûr de l'objet actuellement sélectionné, s'il existe une telle sélection, ou du dernier objet sélectionné (ou édité) si aucune sélection n'est disponible.

2.6 La méthode getChildren()

Grâce à cette méthode, il est possible de récupérer la liste de tous les objets d'une scène particulière.

REMARQUE

Le module `Object` propose une méthode très similaire, `Object.Get()`, qui récupère toutefois la liste de tous les objets existants de Blender, toutes scènes confondues. La méthode `Scene.getChildren()` se révèle plus pratique dans la mesure où elle ne retourne que la liste des objets de la scène courante, ce qui est pratique dans le cadre de scripts d'export, par exemple.

Essayons le script suivant avec la scène par défaut de Blender :

```
01:     import Blender
02:     from Blender import Scene
03:     scn = Scene.getCurrent()
04:     children = scn.getChildren()
05:     print children
```

On obtient dans la console la liste des objets de la scène :

```
[[Object "Cube"], [Object "Lamp"], [Object "Camera"]]
```

2.7 La méthode getCurrentCamera():

Cette méthode permet de récupérer la caméra active de la scène. A noter que dans Blender, n'importe quel objet peut être une caméra, et que cette méthode peut donc récupérer un objet en lieu et place d'une vraie caméra ! Par exemple, sélectionnez la lampe de la scène par défaut de Blender, et appuyez sur la combinaison de touches `[Ctrl]+[0]` du pavé numérique. Lancez maintenant le petit script suivant :

```
01:     import Blender
02:     from Blender import Scene
03:     scn = Scene.getCurrent()
04:     cam = scn.getCurrentCamera()
```

Le résultat, dans la console, sera alors le suivant :

```
[Object "Lamp"]
```

Vous pouvez également utiliser la méthode `setCurrentCamera()` afin de définir, au travers de Python, la caméra active. Par exemple, le script :

```
01:     import Blender
02:     from Blender import Scene, Object
03:     scn = Scene.getCurrent()
04:     cam = Object.Get('Camera')
05:     scn.setCurrentCamera(cam)
```

récupère l'objet nommé `Camera` dans l'objet `cam`, puis la méthode `setCurrentCamera()` fait de cet objet `cam` la caméra active de la scène `scn`.

2.8 La méthode getName():

Sans surprise, cette méthode permet de récupérer le nom de la scène courante. Par exemple :

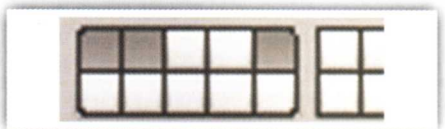
```
04:     print scn.getName()
```

renverra `Scene` dans la console pour la scène par défaut de Blender. Bien sûr, vous pouvez également utiliser la méthode `setName()` pour changer ce nom si cela ne vous convient pas :

```
05:     scn.setName('SceneAlternative')
```

2.9 La méthode getLayers():

Supposons que les calques actifs soient les suivants, dans votre scène Blender :



Les calques 1, 2 et 5 sont activés

Nous pouvons utiliser la méthode `getLayers()` pour afficher les numéros de calque actifs dans la console grâce au petit script suivant :

```
01:     import Blender
02:     from Blender import Scene
03:     scn = Scene.getCurrent()
04:     liste_calques = scn.getLayers()
05:     print liste_calques
```

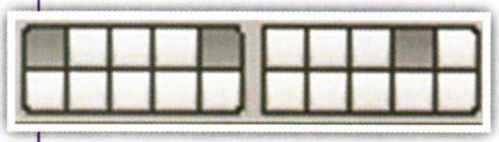
Ce qui se traduit par l'affichage effectif suivant, dans la console :

```
[1, 2, 5]
```

Bien sûr, nous pouvons également spécifier grâce à la méthode `setLayers()` les numéros de calques à activer. Par exemple :

```
04:     calques_actifs = scn.setLayers([1, 5, 9])
```

activera les calques suivants de Blender:



Désormais, ce sont les calques 1, 5 et 9 qui sont actifs

REMARQUE

La méthode `setLayers()` sert seulement à déterminer les calques actifs de la scène courante, mais aucunement à déterminer sur quel calque se trouve un objet particulier. Cette dernière action est du ressort du module `Object`, et de la variable `layers`. Par exemple, reprenons la scène par défaut de Blender. Elle contient une caméra, un cube et une lampe dans le calque numéro un. Exécutons maintenant le script suivant :

```
01:     import Blender
02:     from Blender import Object
03:     ob_list = Object.Get()
04:     item = 0
05:     for obj in ob_list:
06:             obj = ob_list[item]
07:             obj.layers = [5]
08:             item = item + 1
```

Il récupère les noms de chaque objet (toutes scènes et tous calques confondus) et les stocke dans une liste nommée `ob_list`. On récupère ensuite chaque objet de la liste et on le stocke dans une variable temporaire `obj`. On définit alors le cinquième calque comme étant le calque de destination de l'objet en question grâce à la ligne `obj.layers = [5]`.

2.10 Les méthodes getRadiosityContext() et getRenderingContext():

Ces deux méthodes permettent de récupérer les réglages propres aux sous-modules `Radio` et `Render`. Par exemple :

```
Rndr = scn.getRenderingContext()
Rndr.render()
```

permet d'ordonner à Blender d'effectuer un rendu de la scène courante. Nous verrons dans de prochains articles les possibilités et les méthodes propres aux sous-modules `Radio` et `Render`.

3. Exemples de scripts

Vous trouverez ci-après deux exemples de scripts faisant appel au menu `Scene`. Ils n'ont pas de grande prétention, à part d'automatiser certaines tâches qui peuvent être longues et répétitives lorsque vos scènes contiennent un très grand nombre d'objets, et de remettre en pratique les fonctions élémentaires `For` et `if`.

3.1 Fusion des calques actifs

Le petit script qui suit permet, pour la scène courante et celle-ci seulement, de déplacer vers un même calque tous les objets figurant sur les calques actifs au moment où il est lancé. Les calques dont le contenu a été déplacé deviennent donc vides.

Ce script illustre la différence importante entre la méthode `setLayers()` du module `Scene` et la variable `Layers` du module `Object` : la première définit le calque actif affiché dans la fenêtre 3D, tandis que le second indique le ou les calques sur lesquels un objet est visible.

```
01:  import Blender
02:  from Blender import Scene, Object
03:  print '=============='
04:  print 'Debut du script'
05:  scn = Scene.GetCurrent()
06:  print 'Nom de la scene: ', scn
07:  children = scn.getChildren()
08:  print 'Liste des objets de la scene: ', children
09:  liste_calques = scn.getLayers()
10:  print 'Liste des calques actifs: ', liste_calques
11:  for lc in liste_calques:
12:          premier_calque = lc
13:          break
14:  print 'Calque de fusion: ', premier_calque
15:  item=0
16:  for obj in children:
17:          obj = children[item]
18:          for ln in obj.layers:
19:                  for lc in liste_calques:
20:                          if (ln == lc):
21:                                  fusion = 1
22:                                  obj.layers = [premier_calque]
23:                                  break
24:                          else:
25:                                  fusion=0
26:          print obj, 'fusion = ', fusion
27:          item = item + 1
28:  print 'Fin du script'
```

3.2 Fusion vers le bas

Ce script est très inspiré du précédent, mais au lieu de fusionner tous les calques actifs ensemble (dans le premier calque), il fusionne la calque actif avec le calque immédiatement situé sous lui. Par exemple, le cinquième calque est fusionné vers le bas, avec le quatrième calque.

Toutefois, rien n'est prévu dans le cas d'un objet visible sur plusieurs calques à la fois : après une fusion, il n'apparaît que sur le calque (inférieur) de fusion. Il mériterait donc d'être amélioré mais cela n'a pratiquement plus rien à voir avec le module Scene.

Ici, les points intéressants sont d'une part le changement de calque, après la fusion, pour montrer le calque immédiatement inférieur présentant les objets « fusionnés » grâce à la méthode setLayers(), et l'usage de la fonction Redraw() pour réactualiser le contenu de la fenêtre 3D.

```
01:     import Blender
02:     from Blender import Scene, Object
03:     print '=============='
04:     print 'Debut du script'
05:     scn = Scene.GetCurrent()
06:     print 'Nom de la scene: ', scn
07:     children = scn.getChildren()
08:     print 'Liste des objets de la scene: ', children
09:     active_layer = scn.getLayers()[0]
10:     print 'Le calque actif porte le numero: ', active_layer
11:     item=0
12:     mergeddown=0
13:     for obj in children:
14:             obj = children[item]
15:             ln = obj.layers[-1]
16:             print obj, 'layer is: ', ln
17:             if (ln == active_layer):
18:                     mergeddown = 1
19:                     obj.layers = [active_layer - 1]
20:             print obj, 'Fusion vers le bas = ', mergeddown
21:             item = item + 1
22:     scn.setLayers([active_layer-1])
23:     Blender.Redraw()
24:     print 'Fin du script'
```

Conclusion

Nous venons d'explorer les fonctions essentielles du module Scene. Ce sont les plus basiques, mais elles peuvent jouer un rôle essentiel dans vos scripts dès lors que vous souhaitez insérer de nouveaux objets, mais aussi réaliser l'export des calques visibles d'une scène vers une autre scène, par exemple.

C'est toutefois un module à tiroirs, puisqu'il propose également deux sous-modules ; le premier, Radio, permet de contrôler le contexte de radiosité de vos scènes.

Pour sa part, le second, Render, permet de contrôler le contexte de rendu de vos scènes (à ce sujet, il est à noter que chaque scène d'un même fichier blend peut présenter des contextes de rendu différents).

Mais ces deux sous-modules ne seront étudiés que dans les prochains articles, et vous noterez alors qu'ils ne manquent pas non plus d'intérêt pratique.

ANNEXE

Blender 2.41

Au moment où vous lirez ces lignes, Blender 2.41 sera sorti depuis déjà quelques semaines.

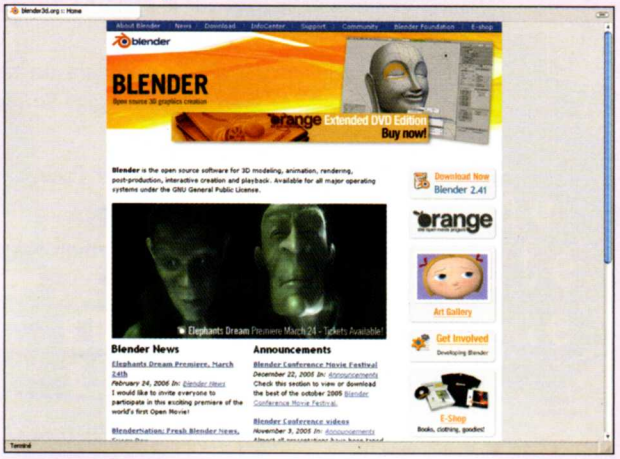
Par rapport à la 2.40, peu de nouvelles fonctionnalités du point de vue de l'utilisateur.

Cette nouvelle version a en revanche vu les efforts des développeurs s'axer sur le moteur de jeu, mais aussi sur les modules Python qui peuvent faciliter l'écriture de scripts d'import ou d'export d'animations.

En particulier, cette version voit le grand retour du module Pose, qui avait été mis entre parenthèse le temps de la réécriture du système d'animation, qui hisse désormais Blender un cran plus haut dans la liste des applications incontournables d'animation 3D.

LIENS

- La page de Blender (version actuelle : v2.41) : www.blender.org



- La docmentation officielle de Python pour Blender v2.40 : **http://www.blender.org/documentation/240PythonDoc/index.html**
- La documentation de Python : http://docs.python.org/download.html

Olivier Saraja,

olivier.saraja@linuxgraphic.org

Listes de lecture pseudo-intelligentes

Sébastien Munch

EN DEUX MOTS **Dans cet article, nous allons voir comment améliorer la façon dont Freevo accède aux fichiers musicaux.**

Les logiciels de lecture musicale historiques (XMMS, Winamp, etc.) basent leurs listes de lectures sur le système de fichiers : l'utilisateur doit rechercher dans l'arborescence des fichiers le morceau qu'il veut écouter.

Les lecteurs multimédias évolués actuels (Rhythmbox, Amarok, JuK, iTunes...) permettent au contraire de simplifier cette tâche, et d'effectuer une sélection musicale par artiste, par genre ou encore par album.

Freevo ne propose pas de liste de lecture intelligente : il ne permet que de naviguer dans le système de fichiers.

Il est possible d'implémenter ce principe au niveau du système de fichiers, par le biais de répertoires et liens symboliques.

Bien que moins performante que le support natif de la fonctionnalité, cette astuce permet de l'émuler avec n'importe quel logiciel multimédia basé sur le système de fichiers, dont Freevo.

Nous allons créer pour cela un script Python. Le choix de ce langage se justifie ici par deux points :

- C'est le langage utilisé par Freevo ;
- Il est simple à comprendre.

Pour plus d'informations sur Python, consultez http://docs.python.org.

NOTE

Ce principe n'est pas limité à Freevo, il est applicable à tout lecteur audio basé sur le système de fichiers.

Structure du script

Nous allons tout d'abord créer une structure de base pour ce script, dans le fichier `/home/freevo/bin/freevo_make_music_links.py`. Ce squelette parcourt la liste des fichiers à traiter, et affiche leur nom :

```
#! /usr/bin/env python
FILESROOT = '/home/freevo/Media/Musique'
LINKSROOT = '/home/freevo/Media/MusiqueLiens'
import os, sys

def main():
    for d in os.walk(FILESROOT):
        for f in d[2]:
            do_file(os.path.join(d[0], f))

def do_file(f):
    print f

if __name__ == '__main__':
    main()
```

Quand on exécute directement un script Python, la variable `__name__` contient la chaîne `"__main__"`. Quand le fichier est importé par un autre script Python, son contenu est différent.

Le test que l'on fait ici permet donc d'exécuter la fonction `main()` lorsque l'on exécute le script. Il est placé à la fin du script pour que toutes les fonctions soient définies avant exécution.

La fonction `os.walk` permet de parcourir un répertoire (dans notre cas, celui qui est pointé par `FILESROOT`) et de retourner tous les fichiers trouvés.

La fonction `do_file` sera replacée par le code de traitement de chacun des fichiers. Pour l'instant, elle ne fait qu'afficher le nom du fichier.

Lorsque le fichier est créé, n'oubliez pas de lui donner les droits d'exécution (`chmod u+x /home/freevo/bin/freevo_make_music_links.py`). Si on exécute ce script, il affichera la liste des fichiers musicaux disponibles.

Traitement des fichiers

Nous allons nous baser sur les méta-informations contenues dans les fichiers MP3 et OGG Vorbis afin de créer notre base de données musicale.

Après différenciation des fichiers .ogg et .mp3, leurs informations sont récupérées à l'aide de programmes externes : ogginfo et mp3info ; ils sont disponibles dans la plupart des distributions. La fonction `do_file` est donc remplacée :

```
def do_file(f):
    infos = {'artist':'', 'title':'', 'album':'',
             'tracknumber':'', 'genre':''}
    fname = f.replace('"', '\\"')
    if f[-4:].lower() == '.ogg':
        answ = os.popen('ogginfo "%s"' % fname)
        for i in answ.readlines():
            for type in infos.keys():
                if '%s=' % type in i.lower():
```

```
                data = i.split('=', 1)[1].strip()
                infos[type] = data
        answ.close()
    elif f[-4:].lower() == '.mp3':
        answ = os.popen('mp3info -p "%%a\n%%t\n%%l\n%%n\n%%g\n"
"%s"' % fname)
        i = [ j.strip() for j in answ.readlines() ]
        answ.close()
        infos = {'artist':i[0], 'title':i[1], 'album':i[2],
                 'tracknumber':i[3], 'genre':i[5]}

    print f, ':', infos
```

L'expression `f[-4:]` retourne les quatre derniers caractères de la chaîne `f`, c'est-à-dire dans notre cas l'extension du fichier. Ceci permet de définir le programme à exécuter pour accéder au fichier. La fonction `os.popen` exécute une commande externe et en récupère la sortie standard. La fonction `readlines` transforme la sortie du programme, ligne par ligne, en une liste de chaînes de caractères.

Dans le cas de `ogginfo`, on parcourt les informations retournées, afin de retrouver celles qui nous intéressent et de remplir le dictionnaire `infos`. Dans le cas de `mp3info`, on a la possibilité de faire formater la sortie du programme comme on veut, ce qui simplifie grandement le traitement.

Si on exécute `freevo_make_music_links.py` à cette étape, il retournera la liste des fichiers suivis de leurs informations sous forme d'un dictionnaire Python.

Pour plus d'informations sur les dictionnaires, vous pouvez vous référer à la section 5.5 du didacticiel sur http://docs.python.org/tut/tut.html.

Création des liens

Maintenant que nous avons les méta-informations concernant tous les fichiers, il nous suffit de créer des liens symboliques dans une arborescence spéciale. Nous allons remplacer la ligne `print f, ':', infos` par les suivantes :

```
    # The file extension
    ext = f[-3:]

    link = '%s.%s' % (infos['title'], ext)

    # Tracks from an artist
    if infos['artist']:
        dir = os.path.join(LINKSROOT, 'artist',
infos['artist'])
```

```
        lnk(f, dir, link)

    # Artists, by genre
    if infos['genre'] and infos['artist']:
        dir = os.path.join(LINKSROOT, 'genre+artist',
                           infos['genre'], infos['artist'])
        lnk(f, dir, link)

    link = '%s - %s.%s' % (infos['tracknumber'],
infos['title'], ext)

    # Albums
    if infos['album']:
        dir = os.path.join(LINKSROOT, 'album', infos['album'])
        lnk(f, dir, link)

    # Albums, by artist
    if infos['album'] and infos['artist']:
        dir = os.path.join(LINKSROOT, 'artist+album',
                           infos['artist'], infos['album'])
        lnk(f, dir, link)

    # Albums, by genre
    if infos['genre'] and infos['album']:
        dir = os.path.join(LINKSROOT, 'genre+album',
                           infos['genre'], infos['album'])
        lnk(f, dir, link)

    # Albums, by genre and artist
    if infos['genre'] and infos['album'] and infos['artist']:
        dir = os.path.join(LINKSROOT, 'genre+artist+album',
infos['genre'],
                           infos['artist'], infos['album'])
        lnk(f, dir, link)

def lnk(f, dir, link):
    try:
        os.makedirs(dir)
    except:
        pass

    link = link.replace('/', '_')
    fullpath = os.path.join(dir, link)
    try:
        os.symlink(f, fullpath)
    except OSError:
        if os.readlink(fullpath) != f:
            linklist = link.split('.')
            linklist[-1] = '%s_' % linklist[-1]
            link = '.'.join(linklist)
            lnk(f, dir, link)
```

Pour chacune des informations de classement, nous testons si cette information existe (`if infos['XXX']`) ; si tel est le cas, la fonction `lnk` est exécutée.

Cette fonction crée le répertoire voulu s'il n'existe pas déjà (`os.makedirs`), puis crée un lien symbolique vers le fichier réel (`os.symlink`).

Si un tel lien existe déjà, un second lien est créé en ajoutant le caractère _ au nom de fichier, ceci pour gérer les cas où deux pistes auraient le même nom.

Résultat

Après l'exécution de ce script, le répertoire `/home/freevo/Media/MusiqueLiens` contient une arborescence logique, permettant de naviguer plus facilement parmi les pistes musicales.

Il nous suffit ensuite de paramétrer Freevo pour accéder de manière élégante à ces répertoires. Dans le fichier `/home/freevo/.freevo/local_conf.py`, la variable `AUDIO_ITEMS` est modifiée :

```
AUDIO_ITEMS = [
   ('Album',
      '/home/freevo/Media/MusiqueLiens/album'),
   ('Artiste',
      '/home/freevo/Media/MusiqueLiens/artist'),
   ('Artiste / Album',
      '/home/freevo/Media/MusiqueLiens/artist+album'),
   ('Genre / Album',
      '/home/freevo/Media/MusiqueLiens/genre+album'),
   ('Genre / Artiste',
      '/home/freevo/Media/MusiqueLiens/genre+artist'),
   ('Genre / Artiste / Album',
      '/home/freevo/Media/MusiqueLiens/genre+artist+album'),
   ('Accès direct', '/home/freevo/Media/Musique'),
]
```

Après relancement de Freevo, le menu *Ecouter de la musique* contient ces différentes catégories.

Cette méthode a bien sûr plusieurs inconvénients, le principal étant que ces informations ne sont pas mises à jour en temps réel. Cependant, une collection musicale n'étant généralement pas très dynamique, il suffira de faire des mises à jour de temps en temps : soit manuellement après avoir modifié ou ajouté des fichiers, soit automatiquement, toutes les nuits par exemple. Par ailleurs, ce script n'efface pas les liens pointant vers des fichiers inexistants.

Mise à jour automatique

Le meilleur moyen de mettre à jour ces informations de manière automatique est d'ajouter une ligne dans `crontab`, avec la commande `crontab -e` :

```
0 5    * * *   /home/freevo/bin/freevo_make_music_links.py
30 5   * * *   /usr/bin/freevo cache >/dev/null 2>&1
```

Ces lignes font exécuter le script à 5h00 tous les jours et mettre à jour le cache de Freevo à 5h30.

Améliorations

Il est bien sûr possible d'améliorer cette procédure. Il est par exemple envisageable de garder les informations de mises à jour dans une base de données à part, afin de ne pas analyser les fichiers déjà traités ; ou encore de gérer les dates de sortie des albums pour les classer par ordre chronologique. Il peut également être intéressant de créer un script qui nettoie les liens devenus inutiles... Mais maintenant, c'est à vous de jouer !

Sébastien Munch,

yeiazel@yeiazel.net

Offres d'emplois LOLIX

Liste des offres au 06 Mars 2006

En partenariat avec LOLIX

Offre Numéro : 5397
Société : OXYD
Référence : TECH-BL
Contact : RH - rh2005@oxyd.fr
Titre : Technicien hotline bilingue
Date de mise en ligne : 05/03/2006

Dans le cadre du développement de la société OXYD, nous recherchons actuellement un technicien support bilingue Anglais / Français. Vous rejoindrez l'équipe technique OXYD et vous interviendrez directement pour nos clients français et étrangers. Vous traitez leur demande par téléphone et par email. Vous devez absolument maitriser un anglais courant, des connaissances sur plateforme Linux sont nécessaires, des connaissances sur plateforme Windows Server sont appréciées.

Offre Numéro : 5396
Société : Appli-Box
Référence : devAppli-Box
Contact : Didier Galland - didier.galland@appli-box.fr
Tél. : 04 74 62 04 29
Titre : Développeur web php/mysql
Date de mise en ligne : 04/03/2006

Agence web de taille humaine en pleine croissance (+50% CA par an), située à Villefranche-sur-Saône (15' de Lyon) recherche un développeur PHP/Mysql H/F

Mission : Développement technique d'applications Internet, Intranet, extranet. Gestion technique des projets Internet. Vous aurez en charge également de conseiller nos clients dans le choix de logiciels libres, et procéderez à leur intégration. Le poste est évolutif et pourra s'orienter vers l'encadrement et la gestion de projets. Diplôme type Ingénieur ou DESS (de préférence mais pas obligatoire).

Expérience : 4-5 ans minimum en agence WEB ou SSII. Une expérience significative PHP et MySql

Connaissances spécifiques demandées :
Expertise : php (objet), MySql, HTML, CSS, javascript
Bonnes connaissances : réseaux, LINUX, logiciels libres
Connaissances appréciées:XUL, ActionScript, Flash
Anglais technique courant

Offre Numéro : 5391
Société : SUBSTANTIEL SAS
Contact : Delmotte Brice - brice@substantiel.fr
Tél. : 01 42 15 10 82
Titre : Développement applications debian
Date de mise en ligne : 03/03/2006

Vous avez l'esprit d'initiative et vous souhaitez effectuer votre stage dans une petite structure dynamique où vous pourrez vous épanouir. Substantiel est une jeune entreprise réactive vendant depuis peu l'ORDISSIMO, un ordinateur simple grand public pour ceux qui ne savent pas se servir d'un ordinateur.Nous développons un système d'exploitation ergonomique et simple, basé sur le système GNU Linux et sur des logiciels open source, tels firefox, thunderbird, openoffice, mplayer, qui sont tous simplifiés ergonomiquement, et qui peuvent être mis à jour à distance. Vous participerez au développement d'applications Open source, à leur simplification et à leur intégration dans notre OS. Une bonne maitrise de Linux est donc impérative !

Offre Numéro : 5392
Société : SUBSTANTIEL SAS
Contact : Delmotte Brice - brice@substantiel.fr
Tél. : 01 42 15 10 82
Titre : Développement applications debian
Date de mise en ligne : 03/03/2006

Vous avez l'esprit d'initiative et vous souhaitez effectuer votre stage dans une petite structure dynamique où vous pourrez vous épanouir. Substantiel est une jeune entreprise réactive vendant depuis peu l'ORDISSIMO, un ordinateur simple grand public pour ceux qui ne savent pas se servir d'un ordinateur. Nous développons un système d'exploitation ergonomique et simple, basé sur le système GNU Linux et sur des logiciels open source, tels firefox, thunderbird, openoffice, mplayer, qui sont tous simplifiés ergonomiquement, et qui peuvent être mis à jour à distance. Vous participerez au développement d'applications Open source, à leur simplification et à leur intégration dans notre OS. Une bonne maitrise de Linux est donc impérative !

Offre Numéro : 5393
Société : ATOUT LIBRE
Référence : TF06102
Contact : Thierry FABIE - drh@atoutlibre.fr
Tél. : 01 69 18 39 90
Titre : Administrateur linux debian+solaris
Date de mise en ligne : 03/03/2006

- 3 ans d'expérience minimum en ingéniérie sous Linux Debian.
- Poste longue durée (CDI).
- Expérience confirmée sous Linux Debian et Unix Solaris.

Offre Numéro : 5394
Société : ATOUT LIBRE
Référence : TF06103
Contact : Thierry FABIE drh@atoutlibre.fr 0169183990
Titre : Administration weblogic/websphere
Date de mise en ligne : 03/03/2006

- Mission Longue durée (12 mois mini.)
- Des compétences Nagios seraient un plus.
- Forte expérience demandée.

Offre Numéro : 5390
Société : Deviant Network
Contact : Jérémie Bouiges - recrutement-webmaster@deviantnetwork.com
Tél. : 08 71 10 02 00
Titre : Webmaster / webdesigner
Date de mise en ligne : 03/03/2006

Deviant Network recherche un développeur maitrisant XHTML et CSS et possédant une vraie patte artistique, pour un projet de refonte de l'ensemble de nos sites web. La maitrise de Flash n'est pas du tout requise.

Offre Numéro : 5389
Société : Telefun / Skyrock
Référence : DEVMOB/JA
Contact : Jerome Aguesse - job@skyrock.com
Tél. : 01 44 88 82 00
Titre : Développeur mobile
Date de mise en ligne : 03/03/2006

Telefun, société éditant skyrock.com et skyblog.com, recherche un développeur maîtrisant les technologies : PHP, SMS/MMS, WAP/I-Mode et J2ME. De formation Ingénieur, vous témoignez d'une expérience solide dans le développement de projets liés aux technologies mobiles, acquise dans le mobile business, au sein d'un portail ou chez un prestataire (SSII, éditeur de services mobiles).

La maîtrise du développement d'applications avec PHP et MySQL est requise, ainsi qu'une bonne connaissance du langage Java (MIDP 2.0, Doja). Vous devez également connaître les contraintes et les formats liés aux différents terminaux mobiles (WML, cHTML, XHTML Mobile Profile, vidéo 3GPP).

Organisé et autonome, vous attachez autant d'importance au contenu et à sa mise en forme qu'à la qualité du code développé. Curieux des nouvelles technologies mobiles et web, vous êtes au courant des dernières innovations et êtes à l'aise dans des environnements hétérogènes de type Linux, OpenBSD et Windows.

Offre Numéro : 5382
Société : evistel
Référence : Admin/06/03
Contact : Marie Munier - marie.munier@evistel.com
Tél. : 01 53 62 95 51
Titre : Administrateur réseaux
Date de mise en ligne : 03/03/2006

Evistel développe des solutions de télécommunications dans un environnement SS7/GSM/TDMA/CDMA/GPRS. Au sein d'une petite équipe d'environ 20 personnes basées à Paris dans le 13ème vous aurez pour mission de vous occuper de l'administration réseaux, avec les tâches suivantes :

- Optimisation des infrastructures réseaux (PPP, VPN...),
- Installation de solutions de travail collaboratif (suivi bugs, groupware...),
- Mise en place de serveurs,
- Scripts de production des machines,
- Support du responsable technique (web + base adresses)

Offre Numéro : 5386
Société : Ava Telecom
Référence : 02032006
Contact : richard Bitton - consultation@aura-group.com
Titre : Architecte open source
Date de mise en ligne : 03/03/2006

AVA TELECOM, filiale d'AURA GROUP (www.aura-group.com), est une SSII spécialisée en Ingénierie Systèmes et Réseaux. AURA GROUP fort de 200 collaborateurs, nous intervenons depuis plus de 14 ans auprès de nos clients Grands Comptes.

- Nous recherchons actuellement un Architecte ou Chef de projet Open Source.
- De formation Bac + 2 minimum, vous avez une expérience d'au moins 4 ans en architecture open source (Redhat et/ou Debian), avec des connaissances en gestion de projet (de migration).
- Disponibilité : 1 mois
- Poste basé sur Paris.
- Devenez partenaire de notre réussite et adresser votre candidature à AVA Telecom, Département Recrutement, à l attention de Mme Virginie DAHIREL, 8 cours Louis Lumière 94300 VINCENNES ou par e-mail: drh@aura-group.com

Merci de préciser votre disponibilité et vos prétentions salariales dans votre dossier de candidature.

Offre Numéro : 5381
Société : Ingénieurs et Consultants
Référence : ic2006-03
Contact : Benedicte SOUPIZET - bso@ingenieursetconsultants.com
Tél. : 01 44 88 24 93
Titre : Expert php -ez publish
Date de mise en ligne : 03/03/2006

Profil :
- Formation ingénieur ou BAC+5, dans le domaine des systèmes d'information et des nouvelles technologies.
- Très bon niveau en PHP, MySQL
- Bonne connaissance du système de gestion de contenu en eZ publish
- Connaissances des technologies XML, CSS2, XHTML, PostgreSQL, Linux, Apache, CVS
- 3 ans d'expérience au minimum

Poste :
L'ingénieur participera à deux types de projet:
- Projets de gestion de contenu réalisés au forfait pour le compte de client grands comptes
- Réalisation d'une application open source de travail collaboratif

Offre Numéro : 5384
Société : Telefun / Skyrock
Référence : devweb
Contact : Jerome Aguesse - job@skyrock.com
Tél. : 01 44 88 82 00
Titre : Développeur (xhtml/php)
Date de mise en ligne : 03/03/2006

De formation informatique, tu possèdes une solide expérience professionnelle en développement de sites web dynamiques. Tu maîtrises MySQL, le langage PHP et le HTML 4 strict + CSS ou XHTML. La connaissance du moteur de template Smarty, de cvs, de subversion serait un plus.

Envoie ton CV + photo avec tes références par e-mail ou par courrier (e-mail prioritaire) à job@skyrock.com

Adresse : Skyrock - Téléfun (Ref : devweb) 37 bis rue Grenéta 75002 Paris

Offre Numéro : 5387
Société : Telefun / Skyrock
Référence : DEVIM/JA
Contact : Jerome Aguesse - job@skyrock.com
Tél. : 01 44 88 82 00
Titre : Développeur applicatif
Date de mise en ligne : 03/03/2006

Vous avez en charge le développement et l'exploitation de services d'Instant Messenging soit au niveau client, soit au niveau serveur. Organisé et autonome, vous êtes au courant des dernières innovations en matière d'instant messenging tant client que serveur.Vous maitrisez les technologies suivantes :

- C/C++ / C# VisualC++ /API Win32/.Net
- Python
- XMPP/Jabber
- Postgresql

Offre Numéro : 5388
Société : Metanext
Référence : DDR-NN-0306
Contact : Silvia Beaudenon - drh@metanext.com
Titre : Responsable d'exploitation linux
Date de mise en ligne : 03/03/2006

Mission :
- Coordonner l'équipe Système et Réseau en environnement LAMP (Linux /Apache / MySQL / Perl)
- Être garant du Maintien en Conditions Opérationnels des systèmes d'infrastructure,
- Coordonner la résolution des problèmes et des escalades,
- Coordonner les différents projets d'évolution de l'infrastructure du SI.
- Poste basé à Paris

Profil recherché :
- Profil d'ingénieur système Linux - 5 ans d'expérience,
- Expérience du management d'équipe technique,
- Expérience de la gestion de projet.

Offre Numéro : 5383
Société : Technologies et Services
Référence : ADMJ2EE-060302
Contact : Jude JEAN - jude.jean@etechnoserv.com
Tél. : 01 46 26 36 02
Titre : H/f administrateur j2ee
Date de mise en ligne : 03/03/2006

Nous recherchons pour un de nos clients un administrateur de plate-forme J2EE qui aura à réaliser les tâches suivantes :

- Installation et configuration des applicatifs serveurs (serveurs HTTP, serveurs d'applications, load balancer)
- Déploiement des applications J2EE (jsp, servlet, ejb)
- Définition des architectures J2EE
- Suivi de production des environnements Internet et Intranet
- Tunning des architectures et applicatifs serveur (Jonas, Websphere, Tomcat, Apache)
- Industrialisation et optimisation des processus d'alimentation et de mise en production
- Rédaction de documentations techniques
- Développement d'applications Intranet (jsp, servlet, ejb, flash, xml)
- Support des équipes de développement et veille.

Les compétences techniques exigées :
- Environnements J2EE
- Serveurs : JONAS, Websphere, Netscape, Apache, IIS.
- Outils : Load Runner, OPEN Sta.
- Langage : SQL, Shell, Java

Expérience :
- 2 à 4 ans
- Etude : Bac + 4/5
- Langue : Anglais

RIM-Linux suite et fin

William Daniau

EN DEUX MOTS **Dans un précédent numéro du magazine (80), un système GNU/Linux complet fonctionnant entièrement en mémoire a été construit. Cet article lui fait suite en détaillant l'installation d'applicatifs sur cette base.**

1 Mon troisième RIM-Linux

1.1 Un peu de cosmétique : bootsplash

Cachez ces logs que je ne saurais voir ! Bootsplash est un patch cosmétique qui permet d'avoir un beau *boot* graphique avec une barre de progression. Bootsplash, c'est un patch, mais aussi des utilitaires que nous allons compiler, Bootsplash permet d'avoir des animations illustrant les différentes phases de boot, mais nous n'irons pas jusque-là, nous nous contenterons d'une barre de progression et de messages textes. Il nous faudra ajouter la bibliothèque `libfreetype` nécessaire à `fbtruetype`, l'utilitaire qui nous permet d'afficher ces messages.

```
tar xjf bootsplash-3.2.tar.bz2
cd Utilities
make
cp splash ../../rootbase/sbin
cp fbtruetype ../../rootbase/sbin
strip ../../rootbase/sbin/{splash,fbtruetype}
cp -d /usr/lib/libfreetype.so* rootbase/usr/lib
strip rootbase/usr/lib/libfreetype.so*
```

Pour pouvoir utiliser Bootsplash, il faut démarrer en mode *framebuffer* (paramètre `vga=xxx`; voyez la documentation du noyau pour les valeurs de paramètres à passer).

Avec Bootsplash, on peut gérer le fait de booter en différentes résolutions. Dans notre cas nous nous limiterons au mode 800x600 16 bits qui présente l'avantage d'exister sur 99,9% des combinaisons ordinateurs+écran.

Ce mode s'active avec l'option `vga=788` que nous avons déjà placé dans notre `isolinux.cfg`. Bootsplash peut se placer dans deux modes différents : le mode `silent` où l'on masque la console en affichant une belle image et le mode `verbose`, où l'on affiche la console.

Le mode de départ est déterminé par le paramètre `splash=silent` (ou `verbose`) dans les paramètres du noyau. Au cours de l'exécution des scripts de démarrage, on pourra passer de l'un à l'autre facilement via

```
echo "silent" > /proc/splash
```

et

```
echo "verbose" > /proc/splash
```

On utilise en général deux images, l'une pour le mode `silent` et une autre qui sert de fond d'écran pour le mode `verbose`. Tout cela est défini dans un fichier de configuration.

Nous allons avoir besoin d'un thème. Sur le site allemand de Bootsplash [1], on trouvera les derniers patches, utilitaires et de nombreux thèmes. La documentation de Bootsplash n'est pas très fournie mais en examinant des exemples, on arrive à s'en sortir assez facilement. Le thème que nous allons utiliser (dans `(chemin)/bootsplash/rimlinux`) est composé simplement d'une belle photo JPEG et d'un fichier de configuration, on utilise la même image pour le mode `silent` et pour le mode `verbose`. Copions donc le contenu de notre thème dans notre `rootbase` :

```
cp -a ./Configuration/bootsplash ./rootbase/etc/
```

Afin que le *kernel* puisse afficher très tôt l'image `silent`, c'est-à-dire avant ses initialisations, l'utilitaire `splash` permet d'ajouter une ou plusieurs images à un `initrd`. Pour un `initramfs`, on procèdera de la façon suivante :

```
splash -s -f config-file > rootbase/bootsplash
```

Cela va créer un fichier `bootsplash` que le kernel saura traiter. Comme l'utilitaire `splash` prend comme argument le fichier de configuration contenant le chemin absolu vers l'image utilisée, le plus simple est de créer un lien symbolique sur votre système (on suppose que le répertoire `/etc/bootsplash` existe) :

```
cd rootbase/etc/bootsplash
ln -s $PWD/rimlinux /etc/bootsplash/rimlinux
```

De cette façon, l'instruction complète que nous ajouterons à notre script de création de l'iso sera :

```
./bootsplash-3.2/Utilities/splash -s -f \
 /etc/bootsplash/rimlinux/config/bootsplash-800x600.cfg >
rootbase/bootsplash
```

On pourra utiliser le script fourni `make_new_iso_bootsplash.sh` . Mais un écran fixe pendant la séquence de boot, ce n'est pas terrible parce qu'on ne sait pas ce que fait le système, d'où la barre de progression.

L'affichage de la barre de progression se fait de la façon suivante :

```
echo "show x" > /proc/splash
```

où x représente un entier compris entre 0 et 65534, c'est aussi simple que ça. On peut aussi afficher un texte avec une taille et une couleur donnée avec fbtruetype. En utilisant tous ces outils, on obtient les scripts fournis rcS-iso3 et setHardware-iso3.pl que l'on va recopier dans notre rootbase :

```
cp ./Configuration/rcS-iso3 rootbase/etc/init.d/rcS
cp ./Configuration/setHardware-iso3.pl rootbase/sbin/setHardware.pl
```

Nous modifierons également le fichier isolinux.config afin d'ajouter splash=silent dans les paramètres du noyau. On aura alors une belle séquence de boot graphique comme en figure 1. Au passage, l'arbre est un magnifique olivier centenaire poussant sous le soleil du Portugal chez ma belle-sœur... :-)

Fig. 1 : Séquence de boot avec Bootsplash

1.2 Un script de configuration de plus : configHardware.pl

A la suite du script setHardware.pl qui détecte le matériel, nous allons lancer dans le rcS un autre script qui va effectuer quelques configurations :

- Montage par subfs des CD-ROM/DVD et *floppys*
- Montage en lecture seule des partitions locales si l'utilisateur en fait la demande.
- Configuration d'interface réseau par DHCP si l'utilisateur en fait la demande.
- Effacement des modules si l'utilisateur en fait la demande.
- Montage automatique des CD et floppys

On examine le contenu de /proc/sys/dev/cdrom/info et on en ressort la liste des CD-ROM. On crée les points de montage et on monte par subfs :

```
# On obtient la liste des cds dans /proc/sys/dev/cdrom/info
@cdrominfo=`cat /proc/sys/dev/cdrom/info`;
chop @cdrominfo;
# La 3ème ligne contient
# drive name:            hda sdb  sdc etc
$tmp=$cdrominfo[2];
@liste_cdroms=split(/[\s\t]+/,$cdrominfo[2]);
shift @liste_cdroms; # drive
shift @liste_cdroms; # name:
foreach (@liste_cdroms)
 {
 #Create the mount point
 system("mkdir -p /localmounts/cd/$_");
 # Mount with subfs
 system("mount /dev/$_ -t subfs /localmounts/cd/$_ -o
fs=cdfss,ro");
 }
```

Pour les floppys, on fait à peu près le même travail, mais on en obtient la liste en examinant /dev/fdx :

```
# Pour les floppys on regarde /dev/fdx
# Ca ne marche pas avec les floppys usb -> TODO
@liste_flop=`cd /dev ; ls fd*`;
chop @liste_flop;
foreach (@liste_flop)
 {
 #Create the mount point
 system("mkdir -p /localmounts/fd/$_");
 # Mount with subfs
 system("mount /dev/$_ -t subfs /localmounts/fd/$_ -o
fs=floppyfss,rw");
 }
```

- Montage automatique des partitions locales

Si l'utilisateur passe localmount=yes, on monte les partitions locales en lecture seule, libre ensuite à l'utilisateur de les remonter en lecture/écriture. On cherche les partitions dans /proc/partitions, on élimine les périphériques *loop* (c'est pour plus tard...), on crée les points de montage et on monte :

```
#
# Montage des partitions locales en lecture seule
# on obtient la liste des partitions dans /proc/partitions
@listepart=`cat /proc/partitions`;
chop @listepart;
# parse the list
# take away first two lines
shift @listepart;
shift @listepart;
foreach (@listepart)
 {
 ($nothing,$major,$minor,$size,$name)=split(/[\s\t]+/,$_);

 if ( ($name =~/\d$/) && !( $name=~/^loop/) ) # This is a
partition
  {
  # Create mountpoint
  system("mkdir -p /localmounts/discs/$name");
  # Mounting
  system("mount /dev/$name /localmounts/discs/$name -o ro");
  }
 }
```

A noter que l'utilisateur peut lancer `localmount.pl` a posteriori pour monter les partitions.

▶ Configuration réseau

Si l'utilisateur passe `ethx=dhcp` sur la ligne de commande du kernel, on tente de configurer cette interface par DHCP avec le client DHCP `udhcpc` :

```
udhcpc -i eth0
```

Je n'ai pas fait d'interface de configuration réseau. Il faudra donc le faire à la main mais ce n'est pas très compliqué. Par exemple, on veut configurer `eth0` avec l'adresse 192.168.0.145, masque 255.255.255.0, la passerelle par défaut est 192.168.0.253 et le DNS est 192.168.0.252 :

```
ifconfig eth0 192.168.0.145 netmask 255.255.255.0
route add default gw 192.168.0.253
echo "nameserver 192.168.0.252" > /etc/resolv.conf
```

▶ Effacement des modules

Si l'utilisateur passe `keepmods=no`, on efface le répertoire des modules du noyau, cela permet de gagner de la place dans les environnements à faible mémoire. Par contre, tout ce qui est `hotplug` et qui n'a pas encore été détecté ne fonctionnera pas.

▶ En résumé

En résumé, on va copier les scripts fournis `localmount.pl` et `configHardware.pl` :

```
cp ./Configuration/configHardware.pl ./rootbase/sbin
cp ./Configuration/localmount.pl ./rootbase/sbin
```

1.3 Montage de partages NFS

Afin de pouvoir monter des partages NFS, il faut lancer le programme `portmap` que nous allons copier depuis notre distribution (il sera lancé par `rcS`) :

```
cp /sbin/portmap rootbase/sbin/portmap
strip rootbase/sbin/portmap
```

On pourra ensuite monter les partages NFS. Pour monter par exemple `machine1:/partage`, on va créer un point de montage, par exemple `/localmounts/nfs/machine1/partage` et exécuter :

```
mount machine1:/partage -t nfs /localmounts/
nfs/machine1/partage
```

1.4 Montage de partages Windows avec samba

Pour les partages samba [2], nous allons avoir besoin de `smbmount`. Le problème est qu'en général, `smbmount` est lié à beaucoup trop de bibliothèques pour être copié tel quel dans notre RIM-Linux. Par exemple, sur ma SuSE 9.3, un `ldd` sur `smbmount` donne :

```
ldd /usr/bin/smbmount
[...]
 libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0x4014a000)
 libldap-2.2.so.7 => /usr/lib/libldap-2.2.so.7 (0x40150000)
[...]
```

On voit que samba a été compilé avec un support LDAP, un support kerberos. C'est vraiment trop pour nous qui souhaitons faire des montages simples de partages Windows... Eh oui, on est bon pour recompiler !

```
tar xzf samba-3.0.20b.tar.gz
cd samba-3.0.20b/source
./configure --without-ldap --without-winbind --without-utmp \
--without-sys-quota --with-smbmount --disable-cups
make
cp bin/{smbmnt,smbmount} ../../rootbase/usr/bin/
strip ../../rootbase/usr/bin/{smbmnt,smbmount}
```

Un `ldd` sur le `smbmount` nouvellement compilé donne quelque chose qui est beaucoup mieux, puisque nous n'aurons pas à rajouter de bibliothèques.

Si on veut monter `//machine1/partage` avec comme *login* `alfred` et comme mot de passe `alfred25`, on créera d'abord un point de montage, par exemple `/localmounts/smb/machine1/partage` et on exécutera :

```
smbmount //machine1/partage /localmounts/smb/machine1/partage \
    -o username=alfred,password=alfred25
```

1.5 Un serveur XXX

1.5.1 Xserver : Buildit

Lorsque Keith Packard était encore dans le groupe XFree86, il avait développé Kdrive une version de X compacte, peu consommatrice en mémoire et ne nécessitant aucun fichier de configuration ni polices annexes pour démarrer. Compiler Kdrive était un peu pénible.

Il fallait télécharger les sources de XFree86, modifier quelques fichiers de configuration après quelques heures laborieuses de recherche de documentation.

Lors du changement de licence de XFree86, Keith Packard est parti en emmenant Kdrive dans son sac et c'est devenu le projet Xserver de freedesktop.org [3]. Xserver a été grandement amélioré par rapport au Kdrive de XFree86-4.2 que j'avais utilisé dans Womp. Il gère plus d'extensions, les polices TrueType avec l'extension Render, la transparence avec l'extension Composite, etc.

Une grande amélioration est qu'il utilise les gnu *autotools*. Plus besoin de `xmkmf` et compagnie. Il n'y a pas de *package* téléchargeable, il faut utiliser le CVS. On trouve sur le site un script permettant d'automatiser la construction du serveur (**http://www.freedesktop.org/wiki/XserverBuildScript**). Je fournis néanmoins un *tarball* du CVS au 17 octobre 2005.

On pourra l'utiliser ou préférer la version CVS. On trouvera dans ce tarball le script original `XserverBuildScript` et deux scripts séparant les deux fonctions du premier, à savoir obtenir les sources d'une part (`Getsources`) et les compiler (`Buildit`) d'autre part.

Le script Buildit (de même que l'original XserverBuildScript) utilise sudo qui vous permet de compiler en tant qu'utilisateur normal. Si vous ne voulez pas utiliser sudo, éditez le fichier Buildit, supprimez tous les sudo et compilez en *root*.A noter un petit problème dans les noms de packages, résolu par la création de liens symboliques dans Buildit. L'installation se fera dans le répertoire /opt/fdo :

```
tar xzf Xserver-cvs-17-10-2005.tar.gz
cd Xserver-cvs-17-10-2005
./Buildit
```

La compilation est relativement longue... *Go take a coffee*. Une fois terminée, nous avons dans /opt/fdo/bin plusieurs serveurs. Nous allons utiliser uniquement le *driver* Xvesa qui fonctionnera avec le plus de matériels différents. Il possède une option -shadow qui accélère son fonctionnement et le rend utilisable même à de très hautes résolutions (je m'en suis servi en 1400x1050 avec confort). Nous allons donc recopier le serveur et les bibliothèques créés dans notre rootbase. Nous copions également la bibliothèque libz de notre distribution nécessaire au serveur X et créons un lien de /usr/X11R6/lib/X11 vers /usr/lib/X11, le serveur cherchant ses polices dans /usr/lib/X11/fonts.

Les informations de locale seront prises par contre dans l'installation locale de Xorg. Certains fichiers étant manquants dans l'installation que nous venons de faire.

```
mkdir -p rootbase/usr/X11R6/lib/X11/locale
mkdir -p rootbase/usr/X11R6/lib/X11/fonts
ln -s ../X11R6/lib/X11 rootbase/usr/lib/X11
mkdir -p rootbase/usr/X11R6/bin
cp /opt/fdo/bin/Xvesa rootbase/usr/X11R6/bin
strip rootbase/usr/X11R6/bin/Xvesa
cp -d /opt/fdo/lib/*.so* rootbase/usr/X11R6/lib
strip rootbase/usr/X11R6/lib/*
cp /opt/fdo/share/X11/XErrorDB rootbase/usr/X11R6/lib/X11
cp /opt/fdo/share/X11/XKeysymDB rootbase/usr/X11R6/lib/X11
cp -a /usr/X11R6/lib/X11/locale/{C,iso8859-1} \
 rootbase/usr/X11R6/lib/X11/locale
cp /usr/X11R6/lib/X11/locale/{compose.dir,locale.dir,locale.alias} \
 rootbase/usr/X11R6/lib/X11/locale
cp -d /lib/libz.so* rootbase/lib
strip rootbase/lib/libz.so*
```

On pourra aussi copier xcompmgr (*composite manager*) pour jouer avec. On copiera aussi une petite sélection de polices, bien que sous certaines conditions, du moins en local, le serveur puisse fonctionner sans. Il est nécessaire d'en avoir quelques-unes en mode terminal.

```
cd rootbase/usr/X11R6/lib/X11/fonts
tar xzf (chemin)/x11-fonts.tgz
```

Au final, nous avons maintenant une installation de X qui prend... 3,1 Mo !

1.5.2 Xserver : Use it

Nous allons décrire sommairement le fonctionnement de ce serveur X. Au préalable, en lançant le serveur avec l'option -listmodes, nous allons obtenir la liste des modes VESA disponibles. Notez que les modes affichés ne présupposent rien sur l'écran.

Ce n'est pas parce que la carte sait faire du 1280x1024x16 que l'écran saura le faire. Voici un type de sortie obtenu :

```
VBE version 3.0 (NVIDIA)
DAC is fixed, controller is VGA compatible, RAMDAC causes snow
Total memory: 32768 kilobytes
[...]
0x0115: 800x600x24 TrueColor [8:8:8:8]
0x0117: 1024x768x16 TrueColor [5:6:5:0]
0x0118: 1024x768x24 TrueColor [8:8:8:8]
[...]
```

Attention ! Les modes affichés dépendent de la carte vidéo. Ce n'est pas parce que dans cet exemple 0x0117 représente le mode 1024x768 16 bits qu'il en sera de même avec une autre carte, il est nécessaire d'effectuer cette opération de listage des modes. Ensuite on lance le serveur en lui spécifiant le mode à utiliser :

```
Xvesa -mode 0x0117
```

Si on a sur le réseau une machine acceptant les connexions Xdmcp[1], on peut d'ores et déjà se servir de notre RIM-Linux tel quel en tant que Terminal X en tapant :

```
Xvesa -mode 0x0117 -query machine
```

On obtiendra alors l'écran de connexion de la machine en question.

Les autres arguments qui vont nous intéresser sont -ac qui équivaut à effectuer un xhost + préalable, -shadow qui accélère le fonctionnement du serveur X, et -mouse qui permet de spécifier le périphérique de la souris avec en option, le nombre de boutons, la molette étant prise en compte en spécifiant 5 boutons :

```
Xvesa -ac -mode 0x0117 -shadow -mouse /dev/input/mice,5 -query machine
```

1.6 Gestionnaire de fenêtre : Boîte noire

Pour une utilisation en local et non en terminal X, il va nous falloir un gestionnaire de fenêtres. Il existe des gestionnaires de fenêtre très compacts et néanmoins fonctionnels.

Parmi eux citons : swm [4], windowlab [5] et blackbox [6]. Les deux premiers sont très petits, moins de 100 ko !

On les utilisera pour des applications très contraintes en taille mémoire. Pour cet exemple, j'ai préféré utiliser blackbox qui allie une taille raisonnable et des fonctionnalités confortables.

[1] *Il faut qu'un gestionnaire de connexion fonctionne, par exemple xdm,kdm ou gdm. Dans les distributions modernes, l'écoute des connexions distantes est souvent désactivé par défaut, l'activer est plus ou moins facile suivant la distribution. Pour kdm, le fichier kdmrc doit contenir Enable=true dans le groupe [Xdmcp].*

Nous utilisons dans la suite une méthode très simple permettant de faire des pseudo-packages, consistant à installer dans un répertoire `/usr/local` vide. Ce n'est pas forcément très élégant mais c'est pratique. On aurait pu aussi préférer utiliser la forme `make DESTDIR=/some/where install` lors de l'installation, mais il faut packager de toute façon après.

```
tar xjf blackbox-0.70.0.tar.bz2
cd blackbox-0.70.0
./configure
make
mv /usr/local /usr/local.sav
mkdir /usr/local
make install
mv /usr/local /usr/local.blackbox
mv /usr/local.sav /usr/local
```

Un `ldd` sur notre blackbox nous indique les bibliothèques que nous allons devoir ajouter. Il nous manque :

```
libstdc++.so.5
libXft.so.2
libfontconfig.so.1
libexpat.so.0
libgcc_s.so.1
```

Nous voyons que blackbox utilise le package `fontconfig`, un package dont le but est de centraliser la configuration des polices (et entre autres les polices TrueType) et dont l'auteur n'est autre que... Keith Packard.

Nous allons utiliser l'installation locale de `fontconfig` et ajouter un répertoire de polices TrueType au même endroit que sur la distribution locale. On peut y mettre les polices qu'on veut mais attention à ne pas abuser, car cela prend de la place. Pour ma part, j'y mets les polices luxi* fournies avec Xorg. Cela prend moins d'un Mo.

```
cp -a /etc/fonts rootbase/etc
cp /usr/bin/{fc-cache,fc-list,fc-match} rootbase/usr/bin
strip rootbase/usr/bin/{fc-cache,fc-list,fc-match}
cp -d /usr/lib/libfontconfig.so* rootbase/usr/lib
strip rootbase/usr/lib/libfontconfig.so*
mkdir rootbase/usr/X11R6/lib/X11/fonts/truetype
cp /usr/X11R6/lib/X11/fonts/truetype/luxi*.ttf rootbase/
usr/X11R6/lib/X11/fonts/truetype
```

Copions maintenant les autres bibliothèques et blackbox lui-même.

```
cp -d /usr/lib/libstdc++.so.5* rootbase/usr/lib
strip rootbase/usr/lib/libstdc++.so.5*
cp -d /usr/X11R6/lib/libXft.so.2* rootbase/usr/X11R6/lib
strip rootbase/usr/X11R6/lib/libXft.so.2*
cp -d /usr/lib/libexpat.so* rootbase/usr/lib
strip rootbase/usr/lib/libexpat.so*
cp /lib/libgcc_s.so.1 rootbase/lib
strip rootbase/lib/libgcc_s.so.1
mkdir -p rootbase/usr/local/share
cp -a /usr/local.blackbox/share/blackbox rootbase/usr/local/share
mkdir -p rootbase/usr/local/bin
cp /usr/local.blackbox/bin/{blackbox,bsetroot} rootbase/usr/local/bin
strip rootbase/usr/local/bin/{blackbox,bsetroot}
```

On pourra personnifier le menu de blackbox dans `rootbase/usr/local/share/blackbox/menu`. Copiez le fichier fourni `menu.blackbox`.

```
cp ./Configuration/menu.blackbox rootbase/usr/local/share/
blackbox/menu
```

1.7 Emulateur de terminal : rxvt-unicode

Toujours pour travailler en local, il nous faut un émulateur de terminal. On peut en recopier un depuis sa distribution ou en recompiler un, ce que je propose ici avec `rxvt-unicode` [7], mon préféré :

```
tar xjf rxvt-unicode-5.7.tar.bz2
cd rxvt-unicode-5.7
./configure --enable-rxvt-scroll --enable-mousewheel \
--with-name=rxvt --enable-xim --with-term=xterm \
--with-res-name=rxvt --with-res-class=Rxvt
make
cp src/rxvt ../rootbase/usr/bin
strip ../rootbase/usr/bin/rxvt
```

1.8 Midnight Commander : un gestionnaire de fichiers en mode texte

Gnu Midnight Commander [9] est un gestionnaire de fichier en mode texte très performant, qui sera très utile pour effectuer des copies/transferts de fichiers sur les partitions locales, CD et montages réseau. Il comporte entre autres fonctionnalités un *pager* (`mcview`) et un éditeur de texte (`mcedit`)[2].

```
tar xzf mc-4.6.1.tar.gz
cd mc-4.6.1
./configure --with-screen=ncurses --with-glib12 \
--without-ext2undel \
--without-gpm-mouse
make
mv /usr/local /usr/local.sav
mkdir /usr/local
make install
mv /usr/local /usr/local.mc
mv /usr/local.sav /usr/local
```

Intégrons `mc` dans notre `rootbase` en épurant un peu, notamment les fichiers de locale :

```
cp -d /usr/local.mc/bin/{mc,mcedit,mcview,mcmfmt} rootbase/usr/
local/bin
strip rootbase/usr/local/bin/{mc,mcmfmt}
cp -a /usr/local.mc/lib/mc rootbase/usr/local/lib
strip rootbase/usr/local/lib/mc/cons.saver
cp -a /usr/local.mc/share/mc rootbase/usr/local/share
rm rootbase/usr/local/share/mc/mc.hlp.*
rm rootbase/usr/local/share/mc/mc.menu.sr
```

[2] *Petite remarque concernant la distribution Suse : quand on fait un su, /opt/gnome/bin et /opt/kde3/bin ne sont pas dans le PATH, or configure a besoin par exemple pour gtk d'accéder à gtk-config. Si on veut faire le configure en root, il faut au préalable faire «export PATH=/opt/gnome/bin:/opt/kde3/bin:$PATH ou bien alors travailler carrément connecté en root.*

```
rm rootbase/usr/local/share/mc/mc.hint.*
mkdir -p rootbase/usr/local/share/locale/fr/LC_MESSAGES
cp /usr/local.mc/share/locale/fr/LC_MESSAGES/mc.mo rootbase/usr/
local/share/locale/fr/LC_MESSAGES
```

Nous avons également besoin de la Glib 1.2 (libglib,libgmodule,libgthread) que nous prenons dans la distribution locale.

```
mkdir -p rootbase/opt/gnome/lib
cp -d /opt/gnome/lib/libglib-1.2.so* rootbase/opt/gnome/lib
strip rootbase/opt/gnome/lib/libglib-1.2.so*
cp -d /opt/gnome/lib/libgmodule-1.2.so* rootbase/opt/gnome/lib
strip rootbase/opt/gnome/lib/libgmodule-1.2.so*
cp -d /opt/gnome/lib/libgthread-1.2.so* rootbase/opt/gnome/lib
strip rootbase/opt/gnome/lib/libgthread-1.2.so*
```

1.9 Serveurs de son

Pour utiliser notre RIM-Linux en tant que terminal X avec le maximum de confort, il est intéressant de pouvoir rediriger le son vers notre terminal, de la même façon que l'on redirige l'affichage. Il n'y a malheureusement pas de méthode aussi standard que X pour effectuer cela. Il existe principalement 3 serveurs de son permettant cela :

- **artsd** le serveur de son de KDE [10].
- **esd** le serveur de son de Enlightement et Gnome [11].
- **nasd** un serveur de son à vocation générale [12].

Nous allons installer ces trois serveurs dans notre RIM-Linux afin de laisser le choix.

1.9.1 artsd

artsd est maintenant complètement intégré à KDE, mais il existe une version *stand-alone* que nous allons utiliser. Pour pouvoir compiler (du moins avec l'emplacement des *headers* de la glib sur ma distribution), il faut modifier le fichier arts/gmcop/giomanager.h en remplaçant la ligne :

```
#include <gmain.h>
```

par

```
#include <glib/gmain.h>
```

La compilation se fait ensuite classiquement :

```
cd arts-0.5.4
./configure
make
mv /usr/local /usr/local.sav
mkdir /usr/local
make install
mv /usr/local /usr/local.arts
mv /usr/local.sav /usr/local
```

Il ne reste plus qu'à l'intégrer dans notre rootbase :

```
cp /usr/local.arts/bin/artsd rootbase/usr/local/bin
strip rootbase/usr/local/bin/artsd
mkdir -p rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libsoundserver_idl.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libkmedia2_idl.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libartsflow.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libartsflow_idl.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libmcop_mt.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libmcop.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libartsc.so* rootbase/usr/local/lib
cp -d /usr/local.arts/lib/libartscbackend.* rootbase/usr/local/lib
strip rootbase/usr/local/lib/*
cp -a /usr/local.arts/lib/mcop rootbase/usr/local/lib
```

Il nous faut aussi copier libaudiofile de notre distribution :

```
cp -d /usr/lib/libaudiofile.so* rootbase/usr/lib
strip rootbase/usr/lib/libaudiofile.so*
```

Sur le terminal X (c'est-à-dire le RIM-Linux), on va lancer artsd de la façon suivante :

```
artsd -n -u -p 16000
```

Sur la machine distante, on va définir la variable d'environnement ARTS_SERVER sur <IP du terminal>:16000, par exemple si l'IP de notre RIM-Linux est 192.168.0.192 :

```
export ARTS_SERVER=192.168.0.192:16000
```

Les applications utilisant arts (Xmms ou Mplayer par exemple) comme sortie se connecteront au serveur artsd sur le RIM-Linux.

1.9.2 esd

La compilation se fait classiquement.

```
tar xjf esound-0.2.36.tar.bz2
cd esound-0.2.36
./configure
make
mv /usr/local /usr/local.sav
mkdir /usr/local
make install
mv /usr/local /usr/local.esd
mv /usr/local.sav /usr/local
```

On copie dans notre rootbase :

```
cp /usr/local.esd/bin/esd rootbase/usr/local/bin
strip rootbase/usr/local/bin/esd
cp -d /usr/local.esd/lib/libesd.so* rootbase/usr/local/lib
strip rootbase/usr/local/lib/libesd.so*
```

Sur le terminal X (c'est-à-dire le RIM-Linux), on va lancer esd de la façon suivante (notre esd ne fonctionne qu'avec alsa, bien qu'il soit censé fonctionner également avec OSS).

```
esd -public -tcp -port 16001
```

Sur la machine distante, on va définir la variable d'environnement ESPEAKER sur <IP du terminal>:16001, par exemple si l'IP de notre RIM-Linux est 192.168.0.192 :

```
export ESPEAKER=192.168.0.192:16001
```

Les applications utilisant esd (Xmms ou Mplayer par exemple) comme sortie se connecteront au serveur esd sur le RIM-Linux.

1.9.3 nasd

nas se compile un peu différemment. Il n'utilise pas les autotools, mais le système d'auto-configuration de X11 bien que n'utilisant absolument pas X.

```
tar xzf nas-1.7.src.tar.gz
cd nas-1.7
xmkmf
make World
```

Si vous souhaitez compiler des applications avec le support natif nas, il faudra installer nas sur votre système en tapant `make install`. Personnellement, je me suis fait un RPM pour nas (à ce propos, il existe un utilitaire qui s'appelle `checkinstall` [8] qui vous crée un RPM en interceptant les appels d'un `make install`). Copions donc `nasd` dans notre `rootbase` :

```
cd nas-1.7
cp server/nasd ../rootbase/usr/X11R6/bin
strip ../rootbase/usr/X11R6/bin/nasd
mkdir ../rootbase/etc/nas
cp server/nasd.conf.eg ../rootbase/etc/nas/nasd.conf
cp -d lib/audio/libaudio.so* ../rootbase/usr/X11R6/lib
strip ../rootbase/usr/X11R6/lib/libaudio.so*
```

Sur le terminal X (c'est-à-dire le RIM-Linux), on va lancer nasd de la façon suivante :

```
nasd :0 -aa -b
```

Le port utilisé est spécifié avec un offset de 8000, `:0` correspond à 8000, `:1` à 8001 etc.

Sur la machine distante, on va définir la variable d'environnement `AUDIOSERVER` sur <IP du terminal>:0, par exemple si l'IP de notre RIM-Linux est 192.168.0.192 :

```
export AUDIOSERVER=192.168.0.192:0
```

Si l'application a le support natif nas (mais il y en a assez peu), ce sera la même chose que précédemment. Par contre, pour les autres applications, il faudra utiliser une bibliothèque supplémentaire qui redirige le son oss sur nas (pour arts et esd, ce sont des programmes séparés artsdsp et esddsp qui effectuent cette opération). Du côté de la machine distante, il nous faudra compiler et installer audiooss :

```
tar xzf audiooss-1.0.0.tar.gz
cd audiooss-1.0.0
xmkmf
make
make install
```

Avant de lancer une application son, il faudra positionner la variable d'environnement `LD_PRELOAD` sur la bibliothèque installée :

```
export LD_PRELOAD=/usr/X11R6/lib/libaudiooss.so.1.0:$LD_PRELOAD
```

Ca ne fonctionne pas avec toutes les applications son, mais ça fonctionne entre autres avec Xmms utilisant oss comme pilote de sortie.

1.10 Navigateur grand luxe : Firefox

Pour l'installation de Firefox [13], nous allons utiliser simplement l'installateur de Firefox et l'installer directement dans notre `rootbase/usr/local`, comme l'installation ne *hardcode* pas dans les fichiers l'emplacement de l'installation, cela ne pose pas de problèmes. On pourra aussi simplement recopier une installation locale faite à partir du tarball.

```
tar xzf firefox-1.0.7.installer.tar.gz
cd firefox-installer
./firefox-installer
```

Suivre les instructions habituelles en choisissant `rootbase/usr/local/firefox` comme répertoire d'installation. Choisissez l'installation personnalisée et désélectionnez *Agent de rapport de qualité*.

Comme nous l'apprend un `ldd` sur `firefox-bin` (les bibliothèques marquées `not found` sont les bibliothèques incluses dans l'installation de Firefox, le script de démarrage repositionnant `LD_LIBRARY_PATH`). Il va nous falloir ajouter pas mal de choses pour que Firefox puisse fonctionner, dont les bibliothèques Gtk2, Atk, Pango et Glib2. Ces bibliothèques désignées par `ldd` et fichiers de configurations associés vont être pris dans notre distribution.

▶ gtk2 :

```
/etc/opt/gnome/gtk-2.0
/etc/opt/gnome/gtk-2.0/gdk-pixbuf.loaders
/etc/opt/gnome/gtk-2.0/gtk.immodules
/opt/gnome/bin/gdk-pixbuf-query-loaders
/opt/gnome/bin/gtk-query-immodules-2.0
/opt/gnome/bin/gtk-update-icon-cache
/opt/gnome/share/locale/fr
/opt/gnome/share/locale/fr/LC_MESSAGES
/opt/gnome/share/locale/fr/LC_MESSAGES/gtk20-properties.mo
/opt/gnome/share/locale/fr/LC_MESSAGES/gtk20.mo
```

▶ atk :

```
/opt/gnome/share/locale/fr
/opt/gnome/share/locale/fr/LC_MESSAGES
/opt/gnome/share/locale/fr/LC_MESSAGES/atk10.mo
```

▶ glib2 :

```
/opt/gnome/share/locale/fr
/opt/gnome/share/locale/fr/LC_MESSAGES
/opt/gnome/share/locale/fr/LC_MESSAGES/glib20.mo
```

▶ pango :

```
/etc/opt/gnome/pango
/etc/opt/gnome/pango/pango.modules
/etc/opt/gnome/pango/pangox.aliases
/opt/gnome/bin/pango-querymodules
```

▶ bibliothèques X11 :

```
/usr/X11R6/lib/libXt.so.6
/usr/X11R6/lib/libXp.so.6
/usr/X11R6/lib/libXi.so.6
/usr/X11R6/lib/libXinerama.so.1
/usr/X11R6/lib/libXfixes.so.3
/usr/X11R6/lib/libXcursor.so.1
/usr/X11R6/lib/libSM.so.6
/usr/X11R6/lib/libICE.so.6
```

Je ne détaille pas toutes les opérations de copie et *strip*, qui ne devraient pas poser de problèmes. On copie les répertoires entiers avec `cp -a`, et les fichiers avec `cp -d` pour conserver les liens symboliques, et on strippe les binaires et les bibliothèques.

On créera également un lien symbolique dans `rootbase/usr/local/bin` :

```
cd rootbase/usr/local/bin
ln -s ../firefox/firefox
```

On pourra aussi ajouter par exemple le *plugin* Flash composé des fichiers `flashplayer.xpt` et `libflashplayer.so` dans le répertoire `plugin` de Firefox. Attention toutefois ! Il y a un bug soit dans Flash soit dans le serveur X. Flash fait planter Firefox dans les modes 16 bits du serveur X. Pour utiliser Flash, il faut être en 24 bits.

1.11 Multimédia : MPlayer

Mplayer [14] est un des meilleurs lecteurs multimédias sous Linux. Il est capable de lire à peu près tous les fichiers multimédias existants, dont une grande part sans l'ajout de fichiers de codecs supplémentaires. Codecs supplémentaires éventuels qu'il suffit de placer dans `/usr/lib/codecs`.

```
tar xjf MPlayer-1.0pre7try2.tar.bz2
cd MPlayer-1.0pre7try2
./configure --enable-gui --enable-largefiles --disable-lirc \
--disable-lircc --disable-tv --disable-live --disable-sdl \
--enable-runtime-cpudetection --disable-aa --disable-caca \
--disable-jpeg --disable-gif --disable-gl --disable-pnm \
--disable-dga --disable-tga --disable-mga \
--disable-directfb --disable-xv
make
mv /usr/local /usr/local.sav
mkdir /usr/local
make install
mv /usr/local /usr/local.mplayer
mv /usr/local.sav /usr/local
```

Un `ldd` sur Mplayer va nous permettre d'identifier les bibliothèques qui nous manquent. Copions donc celles-ci :

```
cp -d /usr/lib/libmad.so* rootbase/usr/lib
cp -d /usr/lib/libdv.so* rootbase/usr/lib
cp -d /usr/lib/libtheora.so* rootbase/usr/lib
cp -d /usr/lib/libogg.so* rootbase/usr/lib
cp -d /usr/lib/liblzo.so* rootbase/usr/lib
cp -d /usr/lib/libdivxdecore.so* rootbase/usr/lib
cp -d /usr/lib/libmp3lame.so* rootbase/usr/lib
cp -d /usr/lib/libxvidcore.so* rootbase/usr/lib
cp -d /usr/lib/libpng.so* rootbase/usr/lib
cp -d /usr/lib/libcdda_interface.so* rootbase/usr/lib
cp -d /usr/lib/libcdda_paranoia.so* rootbase/usr/lib
strip rootbase/usr/lib/*
cp -d /opt/gnome/lib/libgdk-1.2.so* rootbase/opt/gnome/lib
cp -d /opt/gnome/lib/libgtk-1.2.so* rootbase/opt/gnome/lib
strip rootbase/opt/gnome/lib/*
cp -d /usr/X11R6/lib/libXxf86vm.so* rootbase/usr/X11R6/lib
strip rootbase/usr/X11R6/lib/libXxf86vm.so*
```

Puis l'installation de Mplayer :

```
cp -d /usr/local.mplayer/bin/{mplayer,gmplayer} rootbase/usr/
local/bin
strip rootbase/usr/local/bin/mplayer
cp -a /usr/local.mplayer/lib/mplayer rootbase/usr/local/lib
strip rootbase/usr/local/lib/mplayer/vidix/*
mkdir -p rootbase/usr/local/share/mplayer/Skin
mkdir -p rootbase/usr/local/share/mplayer/font
mkdir -p rootbase/etc/mplayer
cd rootbase/usr/local/share/mplayer
ln -s ../../../X11R6/lib/X11/fonts/truetype/luximb.ttf subfont.ttf
```

Nous aurons également besoin d'un revêtement. Nous utiliserons `Blue-1.4.tar.bz2` qui est le revêtement par défaut de MPlayer.

```
cd rootbase/usr/local/share/mplayer/Skin
tar xjf ../../../../../../Blue-1.4.tar.bz2
mv Blue default
```

1.12 Bonus : mplayerplug-in

Mplayerplug-in [15] est un excellent plugin pour `mozilla/netscape/firefox`, qui intègre Mplayer dans le navigateur pour visualiser tous les fichiers que sait lire Mplayer.

```
tar xzf mplayerplug-in-3.11.tar.gz
cd mplayerplug-in
./configure
make
strip *.so
cp *.so ../rootbase/usr/local/firefox/plugins/
cp *.xpt ../rootbase/usr/local/firefox/plugins/
```

Mplayerplug-in est de plus assez intelligent dans la gestion du son, puisqu'il teste la présence de différents serveurs de son, dans l'ordre : `artsd`, `esd` puis se rabat sur OSS.

1.13 Touche finale

Le script fourni, `startx.pl`, permet de simplifier le lancement de X. Il commence par proposer une liste de modes ordonnés par profondeur et par taille, puis demande si on veut travailler en terminal X ou en local. Si on travaille en terminal X, il propose de lancer un serveur de son puis demande le nom ou l'IP de la machine distante. Si on travaille en local, il lance le serveur X puis le gestionnaire de fenêtres.

```
cp ./Configuration/startx.pl rootbase/usr/X11R6/bin
```

`mixer.pl` est un script tout simple qui lance `alsamixer` si le son est configuré avec alsa et lance `rexima` sinon.

```
cp ./Configuration/mixer.pl rootbase/usr/bin
```

J'ai ajouté également Vncviewer, dont je me sers assez souvent, directement de ma distribution ainsi que les bibliothèques supplémentaires nécessaires.

```
cp /usr/X11R6/bin/vncviewer rootbase/usr/X11R6/bin
strip rootbase/usr/X11R6/bin/vncviewer
cp -d /usr/X11R6/lib/libXaw.so.8* rootbase/usr/X11R6/lib
cp -d /usr/X11R6/lib/libXmu.so* rootbase/usr/X11R6/lib
cp -d /usr/X11R6/lib/libXpm.so* rootbase/usr/X11R6/lib
cp -d /usr/lib/libjpeg.so.62* rootbase/usr/lib
strip rootbase/usr/X11R6/lib/libXaw.so.8*
strip rootbase/usr/X11R6/lib/libXmu.so*
strip rootbase/usr/X11R6/lib/libXpm.so*
strip rootbase/usr/lib/libjpeg.so.62*
```

1.14 Création de l'iso

Encore un peu de cosmétique avec Isolinux. Lorsque l'on boote, on a un écran noir avec un prompt `boot:` ce qui n'est pas très plaisant quand on s'est embêté à configurer un beau Bootsplash pour la suite. Nous allons donc voir comment ajouter une image dès l'amorçage d'Isolinux.

Avec The Gimp, nous allons éditer une image que l'on souhaite faire apparaître au démarrage. Pour notre exemple, il s'agit encore de la photo de l'olivier avec un texte ajouté : `F1 : Help`. Tout d'abord, nous allons la redimensionner en 640x480 ou plutôt à peine plus petit pour laisser la place pour le prompt d'Isolinux, soit en 640x450. Puis nous allons changer le mode de l'image et l'amener en mode indexé : *Menu Image -> Mode -> Indexe* en choisissant une palette optimale 16 couleurs (eh oui je sais, ce n'est pas beaucoup). A la suite de quoi nous allons enregistrer cette image sous le format ppm avec formatage des données brut.

Nous avons maintenant un fichier `isolinux.ppm` qu'il va falloir convertir en un format spécifique à Isolinux. Pour ce faire, nous utiliserons le script perl `ppmtolls16` inclus dans le package `syslinux` :

```
ppmtolss16 < isolinux.ppm > isolinux.lss
```

Il faut ensuite que nous fabriquions un fichier `isolinux.msg` contenant :

```
<CAN>isolinux.lss<newline>
```

où `<CAN>` représente le caractère ASCII 24 ou encore le caractère de contrôle <Ctrl-X>.

Cela peut se faire facilement avec la ligne de commande suivante :

```
perl -e "printf(\"\cXisolinux.lss\n\");" > isolinux.msg
```

Pour finir, il va nous falloir modifier notre `isolinux.cfg` de façon à ce qu'il pointe sur ce fichier `.msg` en y ajoutant la ligne `display isolinux.msg`.

En ce qui concerne le texte `F1 : Help` ajouté à l'image, il semble nous souffler l'idée que si on appuie sur [F1] on aura de l'aide... Cela se fait très simplement en ajoutant la ligne `F1 help.txt` à notre fichier `isolinux.cfg` et en ajoutant bien sur un fichier `help.txt` qui va décrire les différentes options du boot.

On trouvera fourni les fichiers `isolinux.lss`, `isolinux.msg`, `help.txt` et `isolinux.cfg` pour cette troisième iso dans un répertoire `isolinux-iso3` :

```
cp Confifuration/isolinux-iso3/* cdrom_base/isolinux
```

La création de l'iso se fait comme d'habitude. On utilisera le script `make_new_iso_bootsplash.sh`. La taille de l'image est de 33 Mo et la taille du `rootbase` décompressé en mémoire est de 68 Mo.

2 C'est beau mais c'est trop gros ! Let's shrink !

Si on examine la taille de notre `rootbase` en tapant `du -sh rootbase`, on voit que la taille de celui-ci est de 68 Mo. Pas question de pouvoir charger cette iso sur une machine ne faisant que 64 Mo donc et même difficilement avec 128 Mo. Il faut dire qu'on a particulièrement chargé la mule, en ajoutant de grosses applications : Firefox, Mplayer. Ceci dit, on se dit d'abord que si on ne sert pas des deux en même temps, ce serait pas mal d'avoir l'autre sous forme compressée...

Et puis tant qu'à faire, en entrant dans le détail, ce serait même pas mal si tous les fichiers du système qui ne servent pas à un instant donné soit conservé sous forme compressée ! C'est ce que réalise l'excellent module `squashfs` que nous avons intégré à notre noyau. Il s'agit d'un système de fichier compressé en lecture seule qui décompresse les fichiers à la demande. Bon ! Ca c'est un outil qui est déjà pas mal, le seul problème c'est qu'étant en lecture seule, on ne peut quand même pas faire ce qu'on veut.

D'où l'utilisation conjointe du non moins excellent module `unionfs` que nous avons également intégré à notre noyau. `Unionfs` (qui est utilisé maintenant par plusieurs distributions *live* dont Knoppix) est un système de fichiers permettant d'avoir réuni en une seule arborescence plusieurs branches de systèmes de fichiers, dont certaines en lecture seule, d'autres en lecture-écriture. Ainsi, on peut éditer et modifier des fichiers qui sont en lecture seule dans leur branche, la version modifiée se trouvant dans une branche en lecture-écriture, et tout cela de manière complètement transparente !

Nous allons donc couper en deux notre `rootbase`. Dans l'une des deux parties, nous allons mettre le strict nécessaire pour amorcer le système. Dans l'autre, nous « squashfs-iserons »... tout le reste !

2.1 Compilation et utilisation des utilitaires squashfs

Nous n'avons pas encore compilé les utilitaires `squashfs` :

```
cd squashfs2.2-r2/squashfs-tools
make
```

Copiez le programme mksquashfs quelque part dans le chemin de recherche des exécutables (`/usr/bin` par exemple). mksquashfs possède de nombreuses options, mais son utilisation la plus simple se résume à celle-ci :

```
mksquashfs repertoire1 repertoire2 ... destination
```

où `destination` représente soit un périphérique réel, soit un fichier que l'on montera ensuite en tant que loop. A noter une différence de comportement sensible suivant que l'on ait un unique répertoire source ou plusieurs. Dans le cas d'un unique répertoire source, le `squashfs` résultant contiendra le contenu du répertoire en éliminant le nom de celui-ci, une

Lisez-vous RÉGULIÈREMENT :

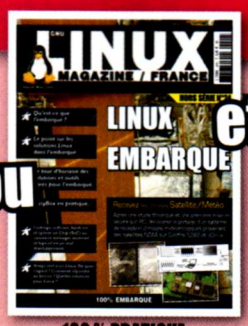
Offres de couplage

Le magazine 100 % Linux

et / ou

Le magazine 100 % Sécurité

et / ou



100 % PRATIQUE

et / ou

Apprivoisez votre pingouin !

Si OUI, alors ces offres d'abonnement à tarif préférentiel vous sont destinées...

En kiosque(1) ~~103,85 €~~ **79 €**

soit une économie de 24,85 €

En kiosque(3) ~~148,55 €~~ **105 €**

soit une économie de 43,55 €

En kiosque(2) ~~110,15~~ **83 €**

soit une économie de 27,15 €

En kiosque(4) ~~184,85 €~~ **129 €**

soit une économie de 55,25 €

(1) Pour 11 Nos Linux Magazine + 6 Nos Linux Mag HS - (2) Pour 11 Nos Linux Magazine + 6 Nos MISC - (3) Pour 11 Nos Linux Magazine + 6 Nos MISC + 6 Nos Linux Mag. HS - (4) Pour 11 Nos Linux Magazine + 6 Nos MISC + 6 Nos Linux Mag. HS + 6 Nos Linux Pratique

OFFRE DE COUPLAGE À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*LINUX MAGAZINE - BP 20142 - 67603 SELESTAT CEDEX

☐ **OUI, je m'abonne et désire profiter des offres spéciales de couplage**

Référence de l'offre :	Prix	Qté.	Total
☐ 11 Nos Linux Mag. + 6 Nos Linux Mag HS	79 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC	83 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC + 6 Nos Linux Mag HS	105 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC + 6 Nos Linux Mag HS + 6 Nos Linux Pratique	129 €		
OFFRES VALABLES UNIQUEMENTS EN FRANCE MÉTRO.		TOTAL	

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

LES 4 FAÇONS DE VOUS ABONNER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

Boostez votre collection !

Avez-vous L'ÂME du COLLECTIONNEUR ?

Vous recherchez un magazine en particulier? Allez sur www.ed-diamond.com pour voir le sommaire détaillé de chaque magazine et ensuite... Boostez votre collection avec les "POWER PACKS x5", soit 5 Linux Magazine pour 15€ et les "POWER PACKS x10", soit 10 Linux Magazine pour 25€ à choisir dans la liste ci-dessous :

LES 4 FAÇONS DE COMMANDER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

Choisissez vos numéros dans le tableau ci-dessous*

*Seuls les numéros ci-dessous sont disponibles pour une commande de POWER PACKS par x5 ou x10

- N°06 GNOME - The Gimp
- N°07 Dopez Linux
- N°08 Le futur résolument objet
- N°09 Prêt pour le jeu !
- N°10 The HURD : 100% GNU
- N°11 Exclusif : l'avenir de G.N.O.M.E
- N°12 NT et Linux : Guerre ou complément ?
- N°13 Cryptage : la clé de la sécurité
- N°14 XFree 4.0 : le futur à notre portée
- N°15 Passez à la vitesse supérieure
- N°16 OpenSources : Est-ce suffisant?
- N°17 Linux : Système embarqué
- N°18 Spécial interview : l'avenir de Linux
- N°19 Dossier spécial : Postgre SQL 7.0
- N°21 Le protocole Internet du 21e siècle : IPv6
- N°22 Le multi-hreading : Une manière moderne de programmer le Multitâche
- N°23 Débugger sous Linux
- N°24 Palm et Linux
- N°25 Kernel 2.4.0
- N°26 <Dossier> XML </Dossier>
- N°27 Les systèmes de fichiers journalisés
- N°28 Scripting : la force d'Unix
- N°29 L.F.S. Linux From Scratch
- N°30 Le chiffrement des données
- N°31 VPN et tunneling
- N°32 Changez de coquille
- N°34 XSL - FO : TeX Killer ?
- N°35 QoS et iproute : optimisation et contrôle du trafic IP
- N°36 Linux embarqué : Le projet mGlinux
- N°37 L'impression sous Linux
- N°38 Le desktop Shell : Enlightenment
- N°39 Sécurité : Patchez votre noyau !
- N°40 MySQL : la base de donnée OpenSource
- N°41 Steganographieou l'art de la dissimulation de données
- N°42 Développez vos pilotes de périphérique
- N°43 Administrez facolement votre réseau SNMP
- N°44 Comprenez NetBios pour Maîtriser l'interopérabilité windows GNU Linux
- N°45 Cohabitation : UnDNS Bind dans un reseau Windows 2000
- N°46 Debian : Utilisez Samba avec le support ACL
- N°47 GNUstep : le petit frère de Mac OS X ?
- N°48 Caudium, votre prochain serveur Web !
- N°49 Après MySQL & PostgreSQL SAP DB : La base de données libre & puisante
- N°50 Créez un album Photo avec PHP ...et sans MySQL
- N°51 Boostez votre site Web avec XML grâce à XSLT, CSS & XPath
- N°52 Linux Temps réel où en est-on aujourd'hui ?
- N°53 Linux sur PDA : Linux dans votre poche !
- N°54 Maîtrisez LVM
- N°55 Intelligence Artificielle : Principes & programmation de jeux de stratégie classique
- N°56 Développez vos applications Mozilla avec XPFE & XPCOM
- N°57 Maîtrisez la gestion... Slots & Signaux ... des événements en C++
- N°58 Djbdns enfin une alternative viable à BIND !
- N°59 Zopix, Créez un CD "Live" Zope en 10 minutes !
- N°60 JBoss serveur d'applications J2EE OpenSource
- N°61 Découvrez MySQL 5 et les procédures stockées
- N°62 Créez votre OS, principe et implémentation
- N°63 Les threads : kernel 2.6 et 2.4
- N°64 Adamoto
- N°65 Théorie et pratique : Supervision avec Nagios
- N°66 Créez votre Distribution Live
- N°67 C# .NET
- N°68 Le crash disque vous guette
- N°69 La réponse de Sun à Linux ? SOLARIS 10
- N°70 Découvrez et comprenez la technologie GRID
- N°71 Présentation et installation du Hurd

NUMÉROS LINUX MAGAZINE ÉPUISÉS

N°01, N°02, N°03, N°04, N°05, N°20, N°33.

BON DE COMMANDE POWER PACKS À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*LINUX MAGAZINE - BP 20142 - 67603 SELESTAT CEDEX

	OUI, je désire acquérir un POWER PACK X5	1er 1PP* X5	2ème 2PP* X5	3ème 3PP* X5
Cochez ici ▲ POWER PACKS X5	1, Linux Magazine N°			
	2, Linux Magazine N°			
	3, Linux Magazine N°			
	4, Linux Magazine N°			
	5, Linux Magazine N°			
	Total par série de POWER PACKS X5 :	15 €	30 €	45 €
	OUI, je désire acquérir un POWER PACK X10	1er 1PP* X10	2ème 2PP* X10	3ème 3PP* X10
Cochez ici ▲ POWER PACKS X10	1, Linux Magazine N°			
	2, Linux Magazine N°			
	3, Linux Magazine N°			
	4, Linux Magazine N°			
	5, Linux Magazine N°			
	6, Linux Magazine N°			
	7, Linux Magazine N°			
	8, Linux Magazine N°			
	9, Linux Magazine N°			
	10, Linux Magazine N°			
	Total par série de POWER PACKS X10 :	25 €	50 €	75 €
	Les Hors Séries et numéros spéciaux sont exclus des POWER PACKS. Montant TOTAL 15€ + 3,81€ de frais de port. Le TOTAL s'élève à 18,81€ pour l'achat d'un POWER Pack x5.	TOTAL :		
	SEULEMENT EN FRANCE MÉTROPOLITAINE !	Frais de port :		+3,81€
	*PP= POWER PACK	TOTAL :		

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions**

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

sorte de *chroot* quoi. Dans le cas de plusieurs répertoires sources, le `squashfs` résultant contiendra à sa racine les répertoires source.

2.2 Séparation du rootbase en deux parties

Nous allons donc couper en deux notre `rootbase`. Nous allons avoir deux répertoires : `rootbase` et `rootbase-squash`. A l'aide de `mksquashfs`, nous créerons un fichier squash à partir de `rootbase-squash` qui sera notre fichier squash principal. Par la suite, nous introduirons les rim-modules qui seront simplement des fichiers squash supplémentaires que nous intègrerons à notre RIM-Linux au moment de la création de l'iso, ce qui nous fera un système modulaire (on pourrait par exemple enlever Firefox et Mplayer du squash principal et faire deux fichiers rim-modules séparés, l'un pour Firefox, l'autre pour Mplayer. Au moment de la création de l'iso, on choisit suivant l'application les modules qu'on ajoute, etc.). Le mécanisme du boot sera celui décrit en figure 2.

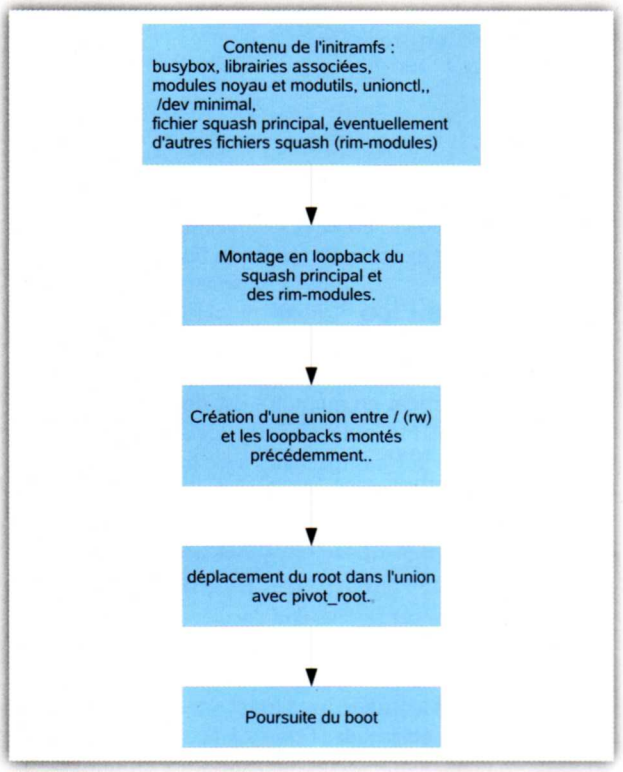


Fig. 2 : Mécanisme de boot avec le squash

Dans un premier temps, renommons `rootbase` en `rootbase-squash` et créons un nouveau répertoire `rootbase`. De quoi avons-nous besoin strictement pour booter ? Eh bien tout simplement à quelque chose près ce que nous avions dans notre ridiculissime première iso plus quelques fichiers de configuration : Busybox, ses liens symboliques, le fichier `inittab`, les premiers fichiers du `/dev`, le fichier de démarrage `/etc/init.d/rcS`, les modules (qui sont compressés ça tombe bien), les utilitaires de modules, les bibliothèques utilisées par ces programmes, `unionctl` pour la gestion de l'union, quelques répertoires vides pour la structure et... ça devrait suffire. Le script suivant devrait faire le travail (`do_split_rootbase.sh`) :

```
#!/bin/sh
# On renomme rootbase en rootbase-squash
# et on crée un nouveau rootbase vide
mv rootbase rootbase-squash
mkdir rootbase
# On efface bootsplash, il sera recréé
# lors de la création de l'iso
rm rootbase-squash/bootsplash
# Repertoire point de montage pivot
mkdir rootbase/oldroot
# repertoires pour gestion squash+union
mkdir rootbase/rimmodules
mkdir rootbase/union
# Repertoires complets
mv rootbase-squash/{dev,proc,sys,tmp,var} \
rootbase
# Repertoires partiels
mkdir rootbase/{etc,bin,sbin,usr,lib}
mkdir rootbase/usr/{bin,sbin}
mv rootbase-squash/lib/modules rootbase/lib
mv rootbase-squash/lib/\
{ld-linux.so.2,libc.so.6,libcrypt.so.1,libm.so.6} \
rootbase/lib
mv rootbase-squash/etc/bootsplash rootbase/etc
mv rootbase-squash/etc/\
{fstab,group,hosts,inittab,networks} \
rootbase/etc
mv rootbase-squash/etc/\
{nsswitch.conf,passwd,profile,shadow} \
rootbase/etc
mv rootbase-squash/etc/init.d rootbase/etc
# programmes
mv rootbase-squash/sbin/\
{insmod,rmmod,lsmod,modprobe} \
rootbase/sbin
mv rootbase-squash/sbin/unionctl rootbase/sbin
# copie de busybox et de ses liens
find rootbase-squash/bin -lname *busybox \
-exec mv '{}' rootbase/bin \;
find rootbase-squash/sbin -lname *busybox \
-exec mv '{}' rootbase/sbin \;
find rootbase-squash/usr/bin -lname *busybox \
-exec mv '{}' rootbase/usr/bin \;
find rootbase-squash/usr/sbin -lname *busybox \
-exec mv '{}' rootbase/usr/sbin \;
mv rootbase-squash/init rootbase
mv rootbase-squash/bin/busybox rootbase/bin
```

On a déplacé le répertoire `bootsplash` car on a un lien symbolique qui pointe dessus dans notre système. Il va de plus falloir créer les *devices loopback* dans notre `/dev`, car on va les utiliser en tout début de boot, avant même que udev soit démarré :

```
cd rootbase/dev
for i in {0,1,2,3,4,5,6,7,8,9,10};
do mknod loop$i b 7 $i;
done
```

Enfin, le début de notre script `rcS` va devoir être modifié pour effectuer le travail supplémentaire (script fourni `rcS-iso4`) :

```
#!/bin/sh

# Mount squashed files
# and move to union
modprobe unionfs
modprobe squashfs

# We need proc for proper operation of
# our squash mounts + union
mount -n proc -t proc /proc

# mount union with only root for the moment
mount -n unionfs -t unionfs -o dirs=/=rw
/.union

# We look in /.rimmodules for squashed files,
#  mount it and add to the union
for i in `ls /.rimmodules`
do
mkdir /.rimmodules/$i-mnt
mount /.rimmodules/$i -o loop,ro \
-t squashfs /.rimmodules/$i-mnt
/sbin/unionctl /.union --add --mode ro \
--after / /.rimmodules/$i-mnt
done

# umount proc before pivoting
umount /proc

# moving into union
cd /.union
/sbin/pivot_root . .oldroot

cd /

mount -n -a
```

Le reste du script est identique à celui de l'iso 3. On commence par charger les modules `squashfs` et `unionfs`, on monte `proc` temporairement, on crée l'union dans le répertoire `.union` avec comme branche unique `/`.

On regarde ensuite dans le répertoire `.rimmodules` et pour chaque fichier qu'on y trouve (s'il s'y trouve, il est censé être un fichier `squash`), on le monte en loopback et on l'ajoute à l'union. Enfin, on démonte notre `proc` temporaire et on effectue un `pivot_root` dans l'union.

On notera que tel que c'est fait, on peut avoir plusieurs fichiers et donc une certaine modularité. On pourra avoir par exemple un fichier principal, un fichier pour Firefox, etc.

2.3 Création de l'iso

Pour la création de l'iso, nous utiliserons le script `make_new_iso-sq.sh` qui effectue en plus l'opération de fabrication du fichier squash :

```
#!/bin/bash

echo "Adding splash picture"
./bootsplash-3.2/Utilities/splash -s -f \
 /etc/bootsplash/rimlinux/\
config/bootsplash-800x600.cfg > rootbase/bootsplash

echo "Making rootbase-squash.rim"
rm rootbase/.rimmodules/rootbase-squash.rim
./squashfs2.2-r2/squashfs-tools/mksquashfs \
 rootbase-squash rootbase/.rimmodules/rootbase-squash.rim


echo "Creating rootbase.gz from rootbase directory"
cd rootbase
find . -print | cpio -o -Hnewc | gzip > ../rootbase.gz

echo "Copying rootbase.gz to cdrom_base"
cd ..
cp rootbase.gz cdrom_base/isolinux/

echo "Creating new iso"
mkisofs -o rimlinux.iso -b isolinux/isolinux.bin \
 -c isolinux/boot.cat -no-emul-boot \
 -boot-load-size 4 -boot-info-table \
 cdrom_base

echo "Done"
```

On notera qu'avant de créer le fichier `squash`, on prend soin d'effacer le précédent, en effet `mksquashfs` n'écrase pas la destination si elle existe mais ajoute au bout.

2.4 Tests de l'iso

Les fonctionnalités de cette quatrième et dernière iso sont les mêmes que celles de la troisième, à ceci près que la place occupée statiquement en mémoire est deux fois inférieure (35 Mo contre 68 Mo pour l'iso 3). Cela veut dire qu'on pourra s'en servir sur des machines avec moins de mémoire ou bien... ajouter encore plus de fonctionnalités !!

2.5 To infinity and beyond... [16] ou comment on rejoint les live-CD ?

Avant de conclure cet article, une dernière remarque. Vu les techniques qu'on a utilisées à la fin, `unionfs` + `squashfs`, on n'est pas très loin des techniques de live-CD. Les deux petits scripts qui suivent ajoutent cette fonctionnalité à notre RIM-Linux. On pourra utiliser des fichiers squash se trouvant sur un support quelconque du moment qu'il est monté :

```
#!/usr/bin/perl

# Usage : use-livemodule.pl squash module
#
#

# Directory to inspect
$livemodule=shift @ARGV;

# First we need to know the last item on the union
@union=`unionctl / --list`;
$lastelement=pop @union;
$lastelement=~/[\s\t]*(.*)[\s\t]+\(.+\)/;
$lastelement=$1;
```

```
system("mkdir /.rimmodules/$livemodule-mnt");
system("mount $livemodule -o loop,ro -t squashfs "/.
"/.rimmodules/$livemodule-mnt");
system("unionctl / --add --mode ro --after ".
$lastelement" /.rimmodules/$livemodule-mnt");

script use-livedir.pl
#!/usr/bin/perl

# Usage : use-livedir.pl directory
#
# The specified directory must contain
# squashed files and only that -> no check
#
# All the files will be mounted and add to the union
#

# Directory to inspect
$dir=shift @ARGV;
print("Dir : $dir\n");

# First we need to know the last item on the union
@union=`unionctl / --list`;
$lastelement=pop @union;
$lastelement=~/[\s\t]*(.*)[\s\t]+\(.+\)/;
$lastelement=$1;

# Look in the specified directory
@listfiles=`ls $dir`;
chop @listfiles;

# make mountpoints, mount loop and add to union
foreach (@listfiles)
  {
  system("mkdir /.rimmodules/$_-mnt");
  system("mount $dir/$_ -o loop,ro -t ".
    "squashfs /.rimmodules/$_-mnt");
  system("unionctl / --add --mode ro --after ".
    $lastelement." /.rimmodules/$_-mnt");
  }
```

Le premier script `use-livemodule.pl` prend comme argument un nom de fichier. Ce fichier sera considéré comme un fichier `squash`. On crée un point de montage, on le monte en loopback et on l'ajoute à l'union. Le second script `use-livedir.pl` prend comme argument un nom de répertoire, tous les fichiers de ce répertoire sont considérés comme des fichiers `squash`, montés en loopback et intégrés à l'union. On trouvera deux exemples de fichiers d'extension : `firefox-java-plugin.rim` (31 Mo) et `mplayer-codecs.rim` (10Mo).

A noter que si l'on place ces fichiers dans le répertoire `rootbase/.rimmodules` avant la création de l'iso, ils seront intégrés directement dans le Rim-Linux. Mais alors il faudra beaucoup de mémoire pour le faire tourner...

Conclusion

Voilà, nous arrivons au bout de cet article qui je l'espère vous aura intéressé. En le suivant, nous avons vu comment réaliser (de manière *quick and dirty* sous certains aspects, notamment dans la partie applicative) une mini-distribution fonctionnant entièrement en mémoire vive et comportant de nombreuses fonctionnalités. Cette mini-distribution n'avait pas d'autre fin que de servir de support à cet article, mais elle nous a permis de passer en revue de nombreux aspects sous-jacents d'un système Linux auxquels on ne touche pas vraiment quand on est simple utilisateur d'une distribution moderne où tout est fait pour nous simplifier la vie.

William Daniau,

RÉFÉRENCES

- [1] http://www.bootsplash.de/files/
- [2] http://www.samba.org/
- [3] http://www.freedesktop.org/wiki/Software_2fXserver
- [4] http://www.informatik.hu-berlin.de/~sperling/prog/swm.html
- [5] http://www.nickgravgaard.com/windowlab/
- [6] http://blackboxwm.sourceforge.net/
- [7] http://dist.schmorp.de/rxvt-unicode/
- [8] http://asic-linux.com.mx/~izto/checkinstall/
- [9] http://www.ibiblio.org/mc/
- [10] http://www.arts-project.org/doc/arts-0.5.4.html
- [11] http://ftp.gnome.org/pub/gnome/sources/esound/0.2/
- [12] http://radscan.com/nas.html
- [13] http://www.mozilla.org/firefox
- [14] http://www.mplayerhq.hu/
- [15] http://mplayerplug-in.sourceforge.net/
- [16] Buzz l'éclair dans *Toy Story* !

Des courbes harmonieuses avec Gnuplot 4.0

Cyril Buttay, Florent Morel,

EN DEUX MOTS **Gnuplot fait partie de ces logiciels libres dont l'origine remonte à la nuit des temps (1986 !). Pourtant, malgré son âge avancé, il fait l'objet d'une activité «brûlante» : sa version 4.0 est sortie en avril 2004 (la précédente remontait à un peu plus de 10 ans...). La version 4.1 est actuellement en développement, et sortira peut-être pour les 20 ans de cet outil de visualisation indémodable !**

Gnuplot permet de tracer des données scientifiques en deux ou trois dimensions, sous forme de courbes, surfaces, etc. Cet article est basé sur l'utilisation de Gnuplot par les auteurs dans le cadre de publications scientifiques.

Comme tous les logiciels de cette époque, Gnuplot possède une interface un peu rebutante, mais il mérite qu'on s'intéresse à lui (ce ne sont pas les inconditionnels de LaTeX, Vi ou Emacs qui me contrediront). C'est un outil en ligne de commande [1], qui fonctionne comme LaTeX : on lui passe une série de directives, on compile, et on récupère le résultat sous forme de fichier graphique. Fonctionnant sur les mêmes principes, Gnuplot et LaTeX sont donc tout naturellement destinés à travailler ensemble (les graphiques produits par Gnuplot peuvent bien entendu être inclus dans un autre outil de mise en page). Cependant, si LaTeX génère par défaut des documents irréprochables (dans leur forme, hein... pour le fond, c'est votre boulot), Gnuplot nécessite un peu plus de travail de finition. Dans cet article, nous allons donc voir comment le dompter pour en tirer la substantifique moelle. Nous allons nous consacrer à la mise en forme d'une figure 2-D qui pourrait être destinée, par exemple, à une revue scientifique. Sachez cependant que Gnuplot est capable de générer bien d'autres formes de tracés, auxquels les concepts étudiés ici s'appliquent tout autant.

Premier exemple

L'équivalent du `Hello world` pour un logiciel de visualisation, c'est de tracer la fonction `sin(x)` ! Pour cela, il vous suffit, en ligne de commande, de taper la commande `gnuplot` pour lancer l'interpréteur :

```
$ gnuplot

        G N U P L O T
        Version 4.0 patchlevel 0
        last modified Thu Apr 15 14:44:22 CEST 2004
[..]
        Send bugs, suggestions and mods to
        <gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'
```

Notez la dernière ligne du message d'accueil (`Terminal type set to 'x11'`), qui signifie que les tracés seront dirigés vers votre serveur graphique ; c'est-à-dire qu'ils seront directement affichés au lieu d'être enregistrés dans des fichiers. Entrez alors la commande suivante : `plot sin(x)`, ce qui devrait produire le tracé de la figure 1. Vous pouvez remarquer que, par convention, la variable `x` représente l'abscisse. Si vous préférez écrire `sin(t)`, il vous faut d'abord déclarer *t* comme variable libre par `set dummy t`.

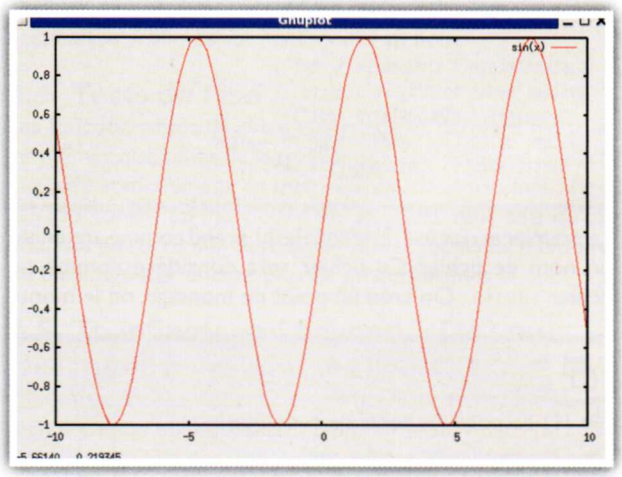


Fig 1 : Tracé de sin(x) sur le terminal X11.

Utilisation avancée

Maintenant que les présentations sont faites, nous allons pouvoir aller plus loin. Dans cette partie, nous allons chercher à tracer l'évolution de la population mondiale, d'après les données disponibles sur le site du bureau américain du recensement (http://www.census.gov/ipc/www/worldpop.html). Le fichier récupéré sur ce site est enregistré sous `population.dat` et possède la forme suivante :

```
1950    2556517137      1.47    37798160
1951    2594315297      1.61    42072962
1952    2636388259      1.71    45350197
1953    2681738456      1.77    47979452
1954    2729717908      1.87    51465740
```

```
1955        2781183648          1.89        52974870
1956        2834158518          ...
```

La première colonne correspond à l'année du recensement, la seconde à la population mondiale correspondante, la troisième est le taux d'accroissement annuel, et la quatrième la variation nette de population. Nous ne nous occuperons ici que des deux premières colonnes. Toutes les personnes qui trouvent qu'il est idiot d'afficher un résultat numérique avec autant de chiffres significatifs sont invitées à en faire part au bureau américain du recensement.

Les terminaux

Nous avons vu que par défaut, après le lancement de Gnuplot, les tracés sont dirigés vers le serveur graphique. Sous Linux, il s'agit en général du terminal X11, mais Gnuplot en connaît bien d'autres. Pour avoir un aperçu des terminaux existants, tapez `set terminal` à l'invite de Gnuplot. Dans la longue liste qui se déroule alors, on retrouve d'autres serveurs d'affichage, des imprimantes, mais aussi des formats de fichiers (`*.png`, `*.pbm`, etc.). Ce sont ces derniers qui vont nous intéresser, puisqu'un tracé dirigé vers ces terminaux sera en fait simplement enregistré dans un fichier du format correspondant.

Reste donc à choisir le terminal adapté à notre usage. Pour l'utilisation avec LaTeX, il vaut mieux se tourner vers le format Postscript, qui permet l'enregistrement sous forme vectorielle et qui est bien accepté par LaTeX. Il faut de plus savoir que tous les terminaux n'offrent pas les mêmes possibilités sous Gnuplot (certains n'autorisent pas les lettres accentuées, d'autres la couleur, etc.). De ce point de vue, le terminal `postscript` est le plus avancé. Il est même parfois plus simple de générer une sortie Postscript et de la convertir après coup (avec ImageMagick par exemple). Notons tout de même que certains terminaux sont conçus spécifiquement pour LaTeX (`psLaTeX`, `epsLaTeX`, `emtex`), et permettent de profiter de la qualité typographique offerte par ce dernier [2].

Tracé de données

Dans l'exemple de la figure 1, nous avons tracé une fonction. Mais Gnuplot peut également (et avec la même commande) tracer des données. Dès que le volume de commande devient important, il est préférable de passer par un fichier texte, plutôt que de taper les commandes en mode interactif. Voici le contenu de `population.plt` (vous pouvez prendre l'extension de votre choix, Gnuplot s'en moque) :

```
1 # fichier Gnuplot pour tracer l'évolution de la population mondiale
2 # entre 1950 et 2050
3
4 set term post eps enhanced
5 set output "population.eps"
6 plot 'population.dat' using 1:2
```

Pour exécuter ce fichier, vous pouvez taper `gnuplot population.plt` en ligne de commande ou `load 'population.plt'` si vous avez déjà lancé Gnuplot. Il y a une différence fondamentale entre les deux méthodes : dans la première, Gnuplot se lance, exécute votre commande, puis se termine. À chaque exécution de votre script, Gnuplot repart à zéro et se comporte donc de la même manière. Dans la seconde, il reste ouvert après l'exécution, et garde en mémoire les actions effectuées. Le résultat des commandes que vous pourrez entrer par la suite dépend donc du script que vous avez exécuté auparavant. Pour vous en convaincre, faites la manipulation suivante (en ligne de commande) :

```
$ gnuplot
..
gnuplot>load 'population.plt'
gnuplot>plot sin(x)
```

Et comparez avec le résultat du premier exemple, en début d'article : le terminal vers lequel est dirigé le résultat de `plot sin(x)` est le dernier utilisé (`X11` pour le premier exemple, `postscript` ici). Pour obtenir un comportement répétable, il est donc conseillé d'exécuter vos scripts par `gnuplot nom_du_script`.

Décortiquons maintenant le contenu du script `population.plt` :

- Lignes 1 et 2, des commentaires débutant par le classique caractère `#` ;
- Ligne 4, on sélectionne le terminal `postscript`, dans sa version `eps enhanced`. Les fichiers `eps` sont identiques aux fichiers `ps` (Postscript), mis à part qu'il ne peuvent contenir qu'une seule page, et qu'ils sont destinés à être incorporés dans un document. L'ajout de la directive `enhanced` signifie que l'on fait appel aux fonctions de typographie « avancée » du terminal, comme les exposants et les indices. Remarquez aussi que lorsqu'il n'y a pas d'ambiguïté, il est possible de raccourcir les instructions de Gnuplot (`term` au lieu de `terminal`, `post` pour `postscript`...). Attention cependant à ne pas perdre en lisibilité ce que vous gagnez en rapidité de frappe ;
- Ligne 5, on définit le nom du fichier de sortie (ici `population.eps`) ;
- Enfin, ligne 6, on demande le tracé du fichier de données `population.dat`, en utilisant la colonne 1 du fichier pour les abscisses et la colonne 2 pour les ordonnées.

À la fin de l'exécution de ce script, vous devriez obtenir dans votre répertoire courant le fichier `population.eps`, dont le contenu correspond à la figure 2.

Dans ce script très court, nous avons quasiment fait le tour de toutes les commandes de Gnuplot que nous utiliserons. D'une manière générale, un script consiste à définir les valeurs de propriétés à l'aide de la commande `set`, puis à effectuer le tracé par `plot`.

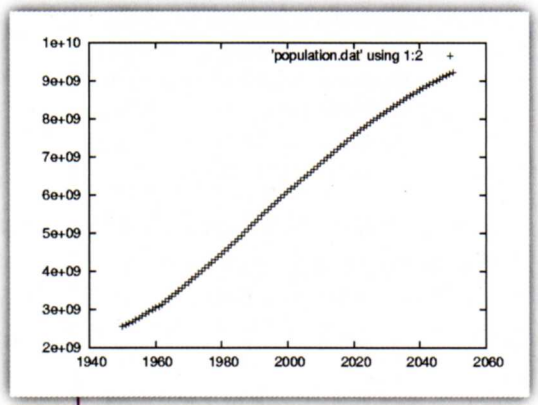


Fig. 2 : Le résultat obtenu « brut de décoffrage ».

La complexité et la puissance de Gnuplot résident dans la multitude des propriétés modifiables (pour en obtenir la liste, il vous suffit de taper `help set`). C'est à ces propriétés que nous allons maintenant nous intéresser en détail.

Commandes de mise en forme

Le tracé de la figure 2 est clair, mais il est trop brut pour être acceptable. Commençons par mettre en forme les axes.

```
1  # fichier Gnuplot pour tracer l'évolution de la population mondiale
2  #  entre 1950 et 2050
3
4  set term post eps enhanced
5  set output "population.eps"
6  set encoding iso_8859_15
7  set xlabel "Année"
8  set ylabel "Population mondiale (en milliards)"
9  set format y "%.0s"
10 set mxtics
11 set grid ytics xtics mxtics
12 plot [1950:2050][0:]'population.dat' using 1:2
```

Par rapport au script précédent, nous avons ajouté ici les lignes 6 à 11. Voyons leur effet :

- Par la ligne 6, on spécifie l'encodage de texte utilisé pour le script. Cette fonction est très importante dès que l'on utilise des caractères autres qu'anglais (comme les caractères accentués, ç, œ, etc.). L'`iso_8859_15` est le plus utilisé pour un Linux francisé ;
- Ligne 7 et 8, les mots-clés `xlabel` et `ylabel` correspondent aux titres des axes ;
- Ligne 9, on définit le format d'affichage de l'échelle des ordonnées. Pour obtenir de plus amples renseignements sur ces formats tapez `help format specifiers` dans Gnuplot. Le format défini par «%.0s» correspond à la partie entière de la mantisse de y en notation scientifique (c'est-à-dire avec des puissances multiples de 3). Bref, regardez plutôt la différence dans l'affichage de l'axe des ordonnées entre la figure 2 et la figure 3 !
- La ligne 10 définit les graduations de l'axe des abscisses. En langage Gnuplot, les « tics », ce sont les graduations des axes. Pour chaque axe, il y a les graduations majeures (`tics`), en face desquelles sont affichées des valeurs numériques, et les graduations mineures (`mtics`) qui sont muettes. Ces dernières sont désactivées par défaut, mais nous allons les afficher (grâce à la ligne 10) pour remplir un peu l'espace entre deux graduations majeures de l'axe des abscisses.
- La commande `set grid` permet d'afficher une grille en pointillés. Par défaut, cette grille ne correspond qu'aux graduations majeures des axes. Ici, nous avons décidé d'inclure les graduations mineures de l'axe des abscisses en rajoutant `ytics, xtics mxtics` en fin de commande.

Il faut enfin noter que nous avons modifié l'échelle d'affichage du graphe en rajoutant les options `[1950:2050][0:]` à la suite de la commande `plot`, ligne 12. Cette échelle est donnée sous la forme $[x_{min}:x_{max}][y_{min}:y_{max}]$ (suivi de z dans le cas d'un graphe 3D), toute valeur manquante étant calculée par Gnuplot. Dans l'exemple de la figure 3, nous avons donc décidé de restreindre l'affichage des abscisses aux années 1950 à 2050 (par défaut Gnuplot ajoute un peu d'espace au début et à la fin des courbes), et de forcer la limite y_{min} à 0, de manière à mieux voir les variations absolues.

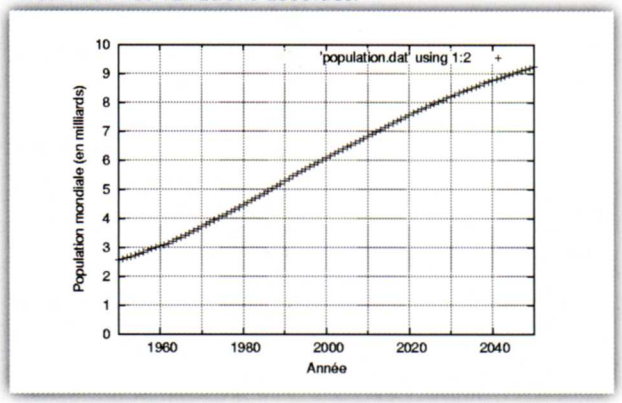


Fig 3 : Modification du style des axes.

Finitions

Maintenant que les axes sont présentables, passons à la dernière étape de notre travail. Il nous reste à changer les polices du graphique (il est destiné à un support écrit, où une police à empattement est plus lisible), puis à mettre en forme la courbe elle-même.

```
1  # fichier Gnuplot pour tracer l'évolution de la population mondiale
2  # entre 1950 et 2050
3
4  set term post eps enhanced "Times-Roman" 18
5  set clip points
6  set style line 1 pointtype 9 pointsize 1
7  set style line 2 linewidth 2 linetype 2
8  set output "population.eps"
9  set encoding iso_8859_15
10 set xlabel "Années"
```

Les polices (et leurs variantes) utilisables par défaut dans le terminal postscript

- Times-Roman, Times-Bold, Times-Italic, Times-BoldItalic ;
- Helvetica, Helvetica-Oblique, Helvetica-Bold, Helvetica-BoldOblique ;
- Courier, Courier-Oblique, Courier-Bold, Courier-BoldOblique ;
- Symbol ;
- AvantGarde-Book, AvantGarde-BookOblique, AvantGarde-Demi, AvantGarde-DemiOblique ;
- Bookman-Light, Bookman-LightItalic, Bookman-Demi, Bookman-DemiItalic ;
- Helvetica-Narrow, Helvetica-Narrow-Oblique, Helvetica-Narrow-Bold, Helvetica-Narrow-BoldOblique ;
- NewCenturySchlbk-Roman, NewCenturySchlbk-Italic, NewCenturySchlbk-Bold, NewCenturySchlbk-BoldItalic ;
- Palatino-Roman, Palatino-Italic, Palatino-Bold, Palatino-BoldItalic ;
- ZapfChancery-MediumItalic ;
- ZapfDingbats.

```
11  set ylabel "Population mondiale (en milliards)"
12  set format y "%.0s"
13  set mxtics
14  set grid ytics xtics mxtics
15  set key bottom right
16  set label 1 "nous {^{e_n}} sommes {/Courier-Bold là}" at 1980, 8.5e9
17  set arrow 1 from 2005,8e9 to 2005, 6.6e9 lw 2
18  plot [1950:2050][0:]'population.dat' using 1:($1=2005?$2:1/0) with lines \
21      title "prévisions" ls 2
```

L'effet de ce script est visible figure 4. Par rapport au listing précédent, nous avons ajouté ou modifié les lignes 4 à 7 et 15 à 21. Ligne 4, nous avons simplement spécifié la police utilisée (`Times-Roman`, suivie de sa taille). Le nom peut être choisi parmi les 35 polices et variantes de base Postscript données dans l'encadré 1, mais il est possible d'en utiliser d'autres, comme Computer modern, qui est utilisée par défaut pour les documents LaTeX (voir `help postscript fontfile` pour plus de détails). Avec le changement d'échelle horizontale que nous avons effectué, certains points se retrouvent à cheval sur les limites du graphe (voir figure 3). Pour éviter cela, nous utilisons la commande `set clip points` (ligne 5) qui a pour effet de supprimer les points qui dépassent. Ligne 6 et 7, nous définissons deux styles de ligne (les types 1 et 2). Une ligne possède quatre caractéristiques : le type de point (`pointtype`), le type de trait (`linetype`), la taille du point (`pointsize`) et la largeur de ligne (`linewidth`). Comme vous pouvez le voir dans l'exemple, il est possible de ne définir qu'une partie de ces paramètres, les autres prenant alors leur valeur par défaut.

Pour obtenir la liste des types de traits et de points offerts par un terminal donné, ainsi que leurs numéros correspondants, vous pouvez utiliser le fichier script `ps_symbols.gpi`, livré avec Gnuplot [3]. Il génère, lorsque vous l'exécutez par `gnuplot ps_symbols.gpi`, un fichier Postscript semblable à celui de la figure 5. Vous pouvez bien entendu modifier ce script, destiné par défaut au terminal postscript monochrome, pour l'adapter aux terminaux qui vous intéressent. En langage Gnuplot, `key` est la légende. La ligne 15 permet de déplacer la légende en bas à droite du tracé (`bottom right`), de manière à ce qu'elle n'interfère pas avec la courbe. `help key` vous permet de connaître les autres mots-clefs de positionnement, et si vous ne voulez pas de légende, il suffit d'utiliser la commande `unset key` ! Le contenu de la légende est défini plus bas.

Il est possible de rajouter des éléments graphiques sur le tracé. C'est ce que nous faisons avec les lignes 16 et 17 : ajout d'une zone de texte (`label`), et d'une flèche (`arrow`). Plusieurs remarques :

- Un numéro (optionnel, ici 1) est attribué à la zone de texte et à la flèche. Ce numéro sert d'identifiant et permet des manipulations ultérieures (cacher, déplacer, modifier les attributs de l'élément) ;
- La position des éléments (instruction `at` pour la zone de texte, `from` et `to` pour la flèche) peut se faire dans plusieurs systèmes de coordonnées. Par défaut, il s'agit des coordonnées affichées par les axes. C'est très pratique dans notre cas puisque si l'on modifie l'échelle de la figure, notre flèche pointera toujours au bon endroit ! Il est également possible d'utiliser des coordonnées relatives à la zone de tracé (et donc indépendantes des échelles) ou relatives à la figure complète (directives `graph` et `screen` respectivement). Comme d'habitude, tapez `help coordinates` pour plus de détails ;
- Bien d'autres paramètres peuvent être ajustés, comme la pointe de flèche, la justification du texte, etc. Ici, nous avons décidé d'utiliser un trait épais pour la flèche, et donc modifié le paramètre `linewidth` (`lw` en abrégé) en conséquence.

Les chaînes de caractères du graphe peuvent recevoir des directives de mise en forme. C'est le cas ici pour le `label` : nous avons localement changé la police du « là » et placé « e_n » en exposant. Les instructions de mise en forme sont encadrées par des accolades, et le nom de police est précédé d'un « / ». Le changement de police est surtout utile pour afficher les caractères grecs contenus dans la police Symbol.

Encadré 2 : Fichier permettant d'obtenir la figure 6

```
1  #fermeture d'un transistor MOS de puissance
2  set terminal postscript eps enhanced color solid "Times-Roman" 20  portrait
3  set grid
4  set style data lines
5
6  set mxtics 5
7  set mytics 5
8  set line style 1 lt 10 lw 3
9  set line style 2 lt 3 lw 3
10 set border 15 lt 7 lw 4
11
12 set arrow 1 from first 11e-6,graph 1 to first 11e-6,graph 0 ls 2 nohead
13 set label 1 "{t{_0}}" at first 10.95e-6,graph 0.9 center
14 set arrow 2 from first 11.04e-6,graph 1 to first 11.04e-6,graph 0 ls 2 nohead
15 set label 2 "{t{_1}}" at first 11.09e-6,graph 0.9 center
16 set arrow 3 from first 11.2e-6,graph 1 to first 11.2e-6,graph 0 ls 2 nohead
17 set label 3 "{t{_2}}" at first 11.25e-6,graph 0.9 center
18 set arrow 4 from first 11.44e-6,graph 1 to first 11.44e-6,graph 0 ls 2 nohead
19 set label 4 "{t{_3}}" at first 11.49e-6,graph 0.9 center
20
21 set output "multiplot.eps"
22 set multiplot
23     set bmargin 4
24         set size 1,0.4
25         set origin 0,0.6
26         set key bottom right
27         set format x ""
28         set format y "%3.0f"
29         unset xlabel
30         set ylabel "Tension grille source (V)"
31         plot [10.8e-6:12e-6][-3:20]"temporel.txt" using 1:3 title "simulation" ls 1
32
33         set origin 0,0.3
34         unset key
35         set ylabel "Tension drain source (V)"
36         plot [10.8e-6:12e-6][0:30]"temporel.txt"  using 1:2 ls 1
37
38         set ylabel "Courant de drain (A)"
39         set format x "%3.1s%c"
40         set xlabel "Temps (s)"
41         set origin 0,0
42         plot [10.8e-6:12e-6][-10:130]"temporel.txt" using 1:(-$4) ls 1
43 unset multiplot
```

Continuons à détailler le script ci-dessus. Lignes 18 à 21, nous retrouvons la commande `plot`, qui a pris un peu d'embonpoint par rapport aux exemples précédents. L'objectif poursuivi ici est de tracer de manière différente les données issues de recensements (celles qui sont antérieures à 2005), et celles estimées (postérieures à 2005).

Gnuplot n'accepte qu'une seule occurrence de `plot` par graphe. Il est par contre possible, lorsque l'on veut tracer plusieurs courbes sur un même graphe, de les enchaîner sur une même ligne de la manière suivante :

```
plot 'fichier1.dat' [options], 'fichier2.dat' [options],...
```

C'est de cette façon que nous avons procédé dans l'exemple. Notez que lorsque l'on réutilise plusieurs fois le même fichier de données, il n'y a besoin de le spécifier qu'une fois, puis de remplacer les appels suivants par `''` (comme dans notre exemple). Il est également possible de mêler tracé de données et de fonctions mathématiques. Dans le cas qui nous intéresse, nous allons effectuer un petit traitement lors de la lecture du fichier de données de manière à obtenir deux jeux de données, l'un antérieur à 2005, l'autre postérieur. Pour cela, nous utilisons la notation `using 1:($1`. La présence de parenthèses indique une fonction à évaluer. À l'intérieur de ces parenthèses, les numéros de colonne du fichier doivent être précédés d'un « `$` ». Le point d'interrogation et les deux points correspondent à la structure if/then/else. La condition peut donc s'exprimer comme suit : si la valeur dans la colonne 1 est inférieure à 2005, on renvoie la valeur correspondante dans la colonne 2, sinon, on renvoie la valeur « 1/0 », qui correspond à une absence de valeur numérique (NaN, ou Not a Number).

Nous attribuons ensuite un des styles de ligne définis plus haut (lignes 6 et 7 du script) aux deux tracés de la commande `plot`, ainsi qu'un titre (qui apparaît dans la légende). Enfin, nous spécifions que la seconde courbe doit être tracée en utilisant une ligne (`with line`) plutôt qu'avec des points (valeur par défaut). Il est également possible de mêler ligne et points avec `with linespoints` et encore bien d'autres possibilités (la liste complète avec `help with`). Enfin, dernier raffinement, nous avons réduit le nombre de points utilisés pour tracer la première courbe, de manière à ce qu'ils ne se chevauchent pas. Pour cela, nous avons fait appel à la directive `every 3` (ligne 19) pour ne tracer qu'un point sur 3.

J'veux de la couleur !

Nous nous sommes limités jusqu'ici à la génération de figures monochromes, comme la plupart de celles que l'on utilise

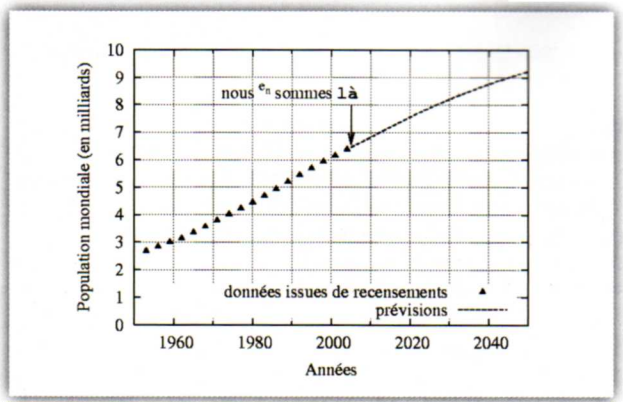


Fig. 4 : Notre figure terminée.

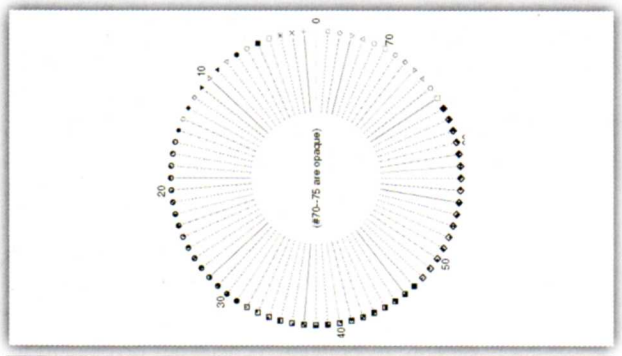


Fig. 5 : Les types de traits et de points du terminal postscript.

sur un document papier. Dans certains cas, l'utilisation de la couleur peut être intéressant (pour un affichage écran ou une présentation au vidéo-projecteur). Il suffit, pour en bénéficier dans l'exemple précédent, de modifier le terminal (ligne 4) comme suit :

```
set term post eps enhanced color solid "Times-Roman" 18
```

La directive `solid` est utilisée pour que les styles de ligne fassent appel à la couleur plutôt qu'aux pointillés. C'est en effet à notre sens la plus grande faiblesse de Gnuplot : les couleurs de ligne et les styles de pointillés ne sont pas indépendants. Il n'est pas possible de spécifier « trait bleu en pointillés » et « trait rouge continu ». La solution la plus simple consiste à n'utiliser les pointillés qu'en terminal monochrome et ne jouer que sur les couleurs de traits en terminal couleur. Même si elle est peu satisfaisante, cette solution se révèle en général peu contraignante.

Tracé plus complexe

Pour finir ce tour d'horizon de Gnuplot, nous vous proposons de décortiquer la figure 6, qui regroupe plusieurs concepts importants. Cette figure représente l'évolution de trois grandeurs électriques d'un transistor MOS durant sa fermeture. Ces trois grandeurs étant différentes (deux tensions et un courant) mais synchronisées, il est intéressant de les tracer dans trois graphes superposés. Nous allons pour cela utiliser l'environnement `multiplot` qui permet de faire appel plusieurs fois à la commande `plot` dans une même figure. Attention, cette commande sert à tracer plusieurs graphes (qui peuvent se superposer partiellement ou totalement) dans une même figure. Pour tracer plusieurs courbes dans un seul graphe, il faut utiliser la commande `plot` (voir lignes 18 à 21 dans l'exemple précédent). Le code permettant d'obtenir la figure 6 est donné dans l'encadré n°2. Regardons-le de plus près. Ligne 2, nous faisons appel au terminal postscript couleur, et nous définissons une utilisation verticale (`portrait`) de la figure. La ligne 4 permet de spécifier de manière globale que les données seront tracées par des lignes. Il n'y aura donc pas besoin de spécifier `with lines` lors de chaque appel à `plot` comme dans les exemples précédents.

Avec les lignes 6 et 7, nous affichons les graduations secondaires des axes d'abscisse et d'ordonnée, et nous en définissons le nombre entre les graduations principales (5). Nous définissons ensuite (lignes 8 et 9) deux types de lignes de tracé (la propriété `lt` étant la version abrégée de `linetype`). La ligne 10 permet de modifier le style de la bordure des graphes. L'utilisation de la commande `set border` est un peu spécifique, car elle est suivie d'un nombre définissant les côtés auxquels elle s'applique (15 pour tous les côtés, `help border` pour la liste complète de ces nombres). Le reste de la commande est simple, avec le choix d'un type (7 pour le noir dans le terminal postscript couleur), et d'une largeur de trait. Lignes 12 à 19, nous définissons quatre flèches et zones de texte (correspondant aux traits bleus et aux étiquettes « t_x » sur la figure 6). Nous avons utilisé ici un mélange de coordonnées : les coordonnées horizontales sont prises relativement à l'échelle des graphes (mot-clef `first`), alors que les coordonnées verticales sont prises relativement au graphe lui-même (mot-clef `graph`). De cette manière, les traits resteront toujours synchronisés aux tracés (puisqu'ils utilisent la même échelle horizontale), et ils couvriront la totalité de la hauteur du graphe (puisque leurs coordonnées verticales sont liées à ce dernier). Notez également la directive `nohead` qui permet de tracer des flèches sans pointes (pour faire des traits !).

Nous passons ensuite dans l'environnement `multiplot` (lignes 21 à 44).

Pour tracer nos trois graphes sur la même figure, il faut bien entendu réduire leur taille. C'est le rôle de `set size`, ligne 24. Attention, les valeurs numériques définissant la taille sont normalisées par rapport à la figure complète. Il faut donc lire la ligne 24 comme « La largeur d'un graphe sera égale à la largeur de la figure (1), et sa hauteur sera de 0,4*la hauteur totale de la figure ».

Si vous avez du mal avec les positions et les positions des graphes...

Lorsque vous allez commencer à jouer avec les tailles ou les positions des graphes, il vous arrivera fréquemment de « sortir du cadre », c'est-à-dire que certains éléments du graphe dépasseront et seront coupés à l'affichage. S'il est préférable de comprendre le fonctionnement de Gnuplot, dans certains cas désespérés, vous pouvez faire appel à `ps2eps`. Véritable caisse à outils du Postscript, cet utilitaire (en ligne de commande) permet – entre autres – de recalculer les limites d'une figure (pour que tous les éléments qu'elle contient soient visibles). Pour recalculer ces limites (appelées « Bounding Box »), il faut utiliser l'option `-B`: `ps2eps -B ma_figure.ps` (un fichier eps peut également être utilisé).

Il ne suffit pas que les graphes aient la même taille pour qu'il soit possible de les aligner. En effet, pour Gnuplot, la taille que nous avons spécifiée correspond à un rectangle englobant tous les éléments du graphe, c'est-à-dire la zone de tracé, mais également les échelles, les graduations (si elles sont à l'extérieur de la zone de tracé), les titres d'axes et de graphe. En fait, Gnuplot considère que la taille du graphe est égale à la taille de la zone de tracé plus des marges de chaque côté. Pour pouvoir aligner les graphes, il faut donc qu'ils aient la même taille et les mêmes marges. Par défaut, les marges sont calculées automatiquement par Gnuplot. Si nous prenons l'exemple de la marge gauche des trois graphes, nous voyons que sa taille est définie par la présence d'un titre d'axe et des valeurs numériques d'échelles. Cependant, les trois échelles ne sont pas identiques. Les deux premières n'ont que deux chiffres significatifs, alors que la troisième en a trois. Pour forcer Gnuplot à considérer le même espace pour chacune des trois échelles, nous avons spécifié un format d'affichage à trois chiffres (ligne 28 dans l'encadré n°2). De cette manière, l'espace nécessaire est réservé par Gnuplot, et les marges gauches ont toutes la même largeur.

La marge basse présente un autre problème, puisque le graphe du bas possède un titre d'axe et une d'échelle, ce qui n'est pas le cas des deux autres. Dans ce cas, la méthode la plus simple consiste à forcer la largeur de la marge. C'est le but de la commande `set bmargin 4` (ligne 23). On définit ainsi une valeur de marge du bas (`bmargin` pour « bottom margin », `help margin` pour la liste) de 4 caractères dans la police courante. La valeur de 4 a été choisie par essais successifs. Cette méthode n'est cependant pas très robuste, puisqu'elle dépend de la taille de police choisie pour le graphe. Enfin, dernière étape pour aligner les graphes, il faut les positionner dans la figure. Nous définissons pour cela l'origine de chaque graphe (`set origin` avant chaque appel à `plot`). Comme pour `set size`, les valeurs des coordonnées sont normalisées par rapport à la figure, avec le point 0,0 situé en bas à gauche.

Dans cette figure, nous avons également utilisé la commande `unset`, qui permet d'annuler l'action d'un `set`. On retrouve `unset` pour terminer le `multiplot` (ligne 44), mais aussi pour masquer la légende (ligne 34).

En effet, lorsqu'un élément est modifié ou créé par `set`, il reste en mémoire jusqu'à ce qu'il soit supprimé (`unset`) ou modifié (autre `set` sur le même élément). C'est ce qui se passe avec les quatre traits et zones de texte, définis lignes 12 à 19, et qui apparaissent sur les trois graphes. En ce qui concerne la légende, on la fait apparaître sur le premier graphe (ligne 26), puis on la supprime (ligne 34), pour ne pas la voir sur les deux autres graphes. On aurait aussi pu utiliser ce comportement de la commande `set` pour les échelles des graphs :

- L'échelle sur l'axe des abscisses étant la même pour tous, on aurait pu ne l'indiquer qu'une fois à l'aide de la commande `set xrange [10.8e-6:12e-6]` ;
- On aurait pu ensuite indiquer l'échelle des ordonnées de chaque tracé en insérant la commande `set yrange [...]` avant chaque appel à la commande `plot`.

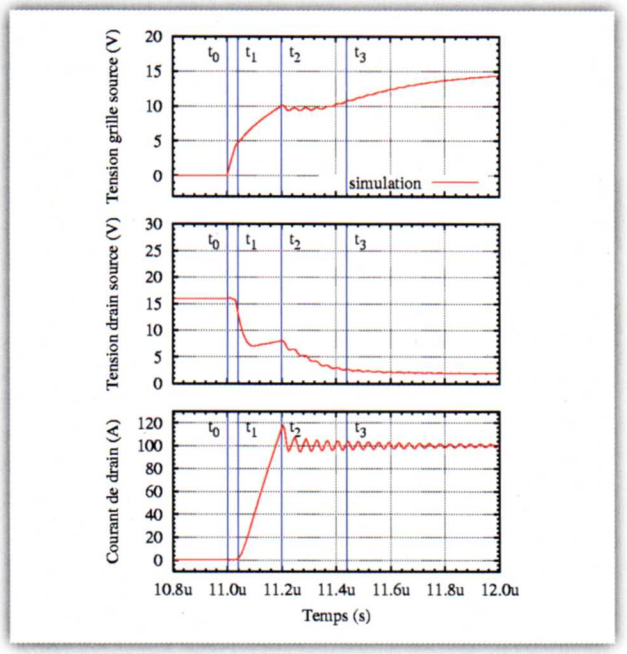


Fig. 6 : Une figure faisant appel à multiplot.

Toujours plus loin...

Nous arrivons au terme de cette introduction. Le but était ici de voir comment utiliser Gnuplot pour obtenir des figures à inclure dans un document LaTeX, et nous avons donc mis

l'accent sur la mise en forme des graphes. Gnuplot offre encore bien d'autres possibilités, comme les graphes 3-D, les graphes bâtons, l'ajout de barres d'erreurs, des fonctions de « fitting » de données...

Nous espérons qu'à l'issue de cet article, vous serez armés pour affronter la documentation de Gnuplot qui est très complète, mais un peu rebutante au premier abord. Elle est disponible sous trois formes (avec le même contenu) : en tapant `help` après avoir lancé Gnuplot, sous forme de fichier PDF (livré avec Gnuplot, ou disponible sur www.gnuplot.info), et au format HTML sur le même site.

Bibliographie

Le site de Gnuplot, www.gnuplot.info, propose la documentation complète, ainsi qu'un grand nombre d'exemples.

De nombreux autres exemples sur un site très complet (même s'il ne s'agit pas de la dernière version de Gnuplot) : http://t16web.lanl.gov/Kawano/Gnuplot/index-e.html

Vous trouverez dans Gnuplot plusieurs fichiers de documentation, comme le manuel, la liste des styles de ligne et de points disponibles [3], mais aussi un guide très court des commandes typographiques : `ps_guide.ps`.

Enfin, pour les cas désespérés, vous pouvez poser des questions sur le newsgroup Usenet `comp.graphics.apps.Gnuplot` (accessible par exemple *via* Google groups ou un lecteur dédié). Sur ce groupe de discussion anglophone, les développeurs de Gnuplot sont très actifs.

NOTES

- [1] Des interfaces graphiques existent pour les allergiques à la ligne de commande. Certaines sont recensées sur la page http://www.Gnuplot.info/links.html à la rubrique **front-ends**.
- [2] Le paquetage LaTeX `psfrag` permet de remplacer des chaînes de caractères d'une figure Postscript. Il est donc également possible de modifier la typographie d'une figure générée par le terminal postscript.
- [3] Pour une Mandrake 10, ce fichier se trouve dans `/usr/share/doc/Gnuplot-4.0.0/psdoc/ps_symbols.gpi`.

Cyril Buttay, Florent Morel,
cyril.buttay@free.fr
florent.morel@insa-lyon.fr

→ Smalltalk : Le pouvoir et la simplicité du tout objet

H. Fernandes, S. Ducasse

EN DEUX MOTS ***Smalltalk est un langage, un environnement de développement, qui mérite à être connu. Avec un modèle objet simple, clair et cohérent, des mécanismes réflexifs sophistiqués (c'est-à-dire la possibilité pour le programme de modifier sa structure et son comportement lors de l'exécution) et des outils de développement évolués, il permet de se concentrer sur l'essentiel : écrire rapidement un code de qualité, facile à déboguer et à faire évoluer. Avec cet article, nous commençons par un tour d'horizon historique et quelques aspects syntaxiques du langage.***

Une vision, un environnement et un langage

La genèse de Smalltalk commence au début des années 1970 sous l'influence de Simula **[Sim]**, Sketchpad **[Ske]** et LISP. Le premier introduit l'idée d'« objets » – bien que le terme programmation objet fût introduit avec Smalltalk – comme nouvelle approche de programmation pour conceptualiser et résoudre des problèmes. Le deuxième inventa l'interface graphique pour manipuler et interagir avec des objets informatiques. Enfin de LISP, Smalltalk reprend son ramasse-miettes (*garbage collector*) et son interactivité (recompilation incrémentale). Ces aspects sont au cœur de Smalltalk, à la fois un langage à objet pur et un environnement graphique pour interagir avec les classes et les objets. Le tout constitue la base d'un environnement informatique non plus réservé aux spécialistes, mais accessible au plus grand nombre. C'était la vision de l'informatique personnelle et du Dynabook d'Alan Kay, dont les idées sont réalisées dans le 100$ PC récemment annoncé par le MIT.

Alan Kay est l'initiateur de Smalltalk et, au début des années 70, il rejoignit le centre de recherche Xerox Parc en Californie **[Dea]**. Avec d'autres dont Dan Ingalls et Adele Goldberg, l'équipe développa les concepts de programmation objet et d'interface graphique. Elle inventa et développa la souris, le système de fenêtrage moderne, l'utilisation d'interface à bitmap, les dispositifs de transferts de bits (BitBlt), le paradigme MVC de développement d'interface utilisateur, le *byte-code*.

Dans les années 70 et jusqu'aux débuts des années 80, plusieurs versions successives virent le jour : Smalltalk-72, Smalltalk-74, Smalltalk-76 et enfin Smalltalk-80. Ces versions furent développées sur des ordinateurs personnels Alto **[Xerox]**. L'intégration d'un environnement graphique au langage était pour Alan Kay essentiel. Il s'agissait d'un amplificateur d'idées pour l'utilisateur afin de lui permettre d'expérimenter librement avec l'environnement et d'en tirer le meilleur. D'autres aspects forts du langage sont le tout objet, l'uniformité du modèle, une capacité complète d'introspection – réflexivité du langage – et le paradigme de la compilation incrémentale qui décuple la vitesse de développement.

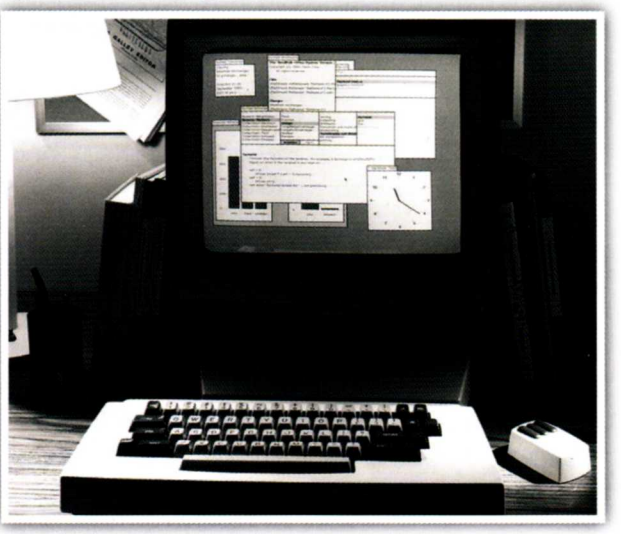


Fig. 1 : Alto III avec Smalltalk-76. On reconnaît le navigateur de classes au centre.

Bien que datant du début des années 80, Smalltalk-80 – que nous appellerons dorénavant Smalltalk – reste un langage d'actualité par la modernité de ses concepts (pureté, uniformité du modèle, simplicité, puissance d'extensions) et l'avance technologique qu'il avait en 1980. En fait, Smalltalk sans son environnement graphique ne serait plus vraiment Smalltalk, car alors on perdrait les outils graphiques de développement (navigateurs de classes, *workspace*, débogueur...) et de compilation incrémentale ou de déboguage/recompilation à la volée d'une application en fonctionnement.

Une émanation du laboratoire XeroxParc, ParcPlace, commercialisa un Smalltalk professionnel, maintenant nommé VisualWorks **[VW]**. D'autres sociétés développèrent leur propre Smalltalk (Dolphin Smalltalk, VisualAge Smalltalk d'IBM avec lequel VisualAge Java était développé). En 1996, la propagation de Smalltalk dans les entreprises américaines était de 60% par an. La déferlante Java qui convertit une partie

des programmeurs C et C++ ralentit considérablement la croissance du langage. Mais depuis les années 2002, Smalltalk revient au premier plan : par exemple, des sociétés telles que AMD, JPMorgan, UPS, Les Mutuelles du Mans... l'utilisent pour des applications très sensibles : chez AMD toute la chaîne de production de gravure de leurs microprocesseurs est développée en Smalltalk ! C'est une application qui ne peut être arrêtée et où il est nécessaire de modifier le code à l'exécution. Pour les personnes qui apprécient de connaître d'où vient une idée, il faut savoir que les *frameworks* de tests unitaires tels que JUnit proviennent de SUnit, l'idée des « *Design Patterns* » (Schémas de conception), la notion de « frameworks » (cadres applicatifs), la méthodologie XP (e*Xtreme Programming*), le modèle MVC (Modèle-Vue-Contrôleur), les *refactorings*, ont d'abord été développés en Smalltalk.

En 1996, une partie de l'équipe d'Alan Kay alors chez Apple, développa une version libre de Smalltalk, nommée « Squeak » **[Squ] [SquFr]**. Cette implantation de Smalltalk intègre en outre des fonctions graphiques et multimédias relativement importantes. Il est développé par une communauté très active. GNUSmalltalk **[Gnu]** est une autre implantation libre de Smalltalk, celui-ci est spécifiquement destiné à l'écriture de scripts, il est également activement développé et offre des possibilités de développement d'interfaces graphiques avec le toolkit TK. Finalement Smalltalk/X est une autre implantation de Smalltalk gratuite **[StX]**. VisualWorks est aussi disponible gratuitement pour toute application non commerciale.

Dans cet article et dans la série qui suivra nous utilisons Squeak, le Smalltalk *open source* multimédia. Les explications données sont applicables à toutes les implantations disponibles.

La philosophie de Smalltalk

Smalltalk a une approche qui, encore aujourd'hui, est très novatrice sur bien des aspects. Il nous semble donc opportun d'énumérer certains d'entre eux pour déjà mieux se familiariser avec ces principes souvent radicaux :

- **Une abstraction forte par rapport au matériel (hardware).** Ceci est réalisé en utilisant une machine virtuelle interprétant le byte-code Smalltalk et un environnement utilisateur entièrement défini en Smalltalk (i. e. en byte-code). Le premier, la machine virtuelle – VM – est un exécutable dépendant de la plate-forme matérielle, celui-ci interprète le byte-code Smalltalk ou le traduit en langage machine (pour les machines virtuelles ayant un traducteur tardif (*Just in Time Translator*) comme VisualWorks ou ST/X). Le deuxième, l'environnement, est le fichier image – appelé `Image` – qui est exécuté par la VM. L'image est indépendante de la plate-forme *hardware*. Elle peut être vu comme un instantané mémoire où l'on aurait gelé tous les objets manipulés par l'utilisateur (à l'image de la fonctionnalité de mise en veille des ordinateurs portables). Le source de l'image est présent dans un troisième fichier, le `Source`, celui-ci est accessible depuis l'environnement et il peut être modifié, les modifications sont enregistrées dans un quatrième fichier, le `Change` qui fonctionne comme un enregistreur de toutes les actions faites par l'utilisateur. Il est quasiment impossible (sauf corruption du disque) de perdre du code avec Smalltalk.
- **Le tout objet, l'ensemble des composants du système sont des objets.** Les opérateurs (multiplications, additions...) sont alors des méthodes qui opèrent sur des instances de classes de nombres, de chaînes de caractères ou de tout autre objet. De même les structures de contrôle sont des messages envoyés sur des instances de classe de Boolean. Quelques conséquences immédiates : (1) une très grande uniformité (facilité de compréhension du code source, si on ajoute à cela le choix de l'héritage simple, et d'un seul type d'accès aux attributs et messages) ; (2) une possibilité totale de surcharger les opérateurs et blocs de contrôle de l'ensemble des classes (puisque ce sont en fait des méthodes).
- **Une syntaxe minimale.** On a l'habitude de dire que sa définition tient sur une carte postale. La simplicité est une règle de conception très souvent adoptée en Smalltalk.
- **Compilation incrémentale.** Elle permet de recompiler – en byte-code – une classe ou même uniquement une méthode lorsque celle-ci a été modifiée. En cours de développement, ce mécanisme permet de modifier/recompiler une partie du code source de l'application sans avoir à la quitter. Ainsi le contexte lié au débogage n'est pas perdu et les bogues sont supprimés très rapidement. De même la compilation incrémentale est présente dans le débogueur : celui-ci permet non seulement de repérer les erreurs mais aussi de les corriger au vol. Ainsi dans une session type, une erreur – ou un signal – qui fait apparaître le débogueur, permet au programmeur de corriger la méthode problématique et de poursuivre le test de l'application sans aucune interruption brutale. La compilation incrémentale est le mode naturel de développement de Smalltalk.
- **Réflexivité du langage.** D'une part, Smalltalk est écrit en Smalltalk (la machine virtuelle, le *parsing*, la compilation en byte-code) et il est capable d'observer ses structures de données depuis le langage de la même façon que l'on interagit avec les objets habituels : par exemple, pour obtenir la liste des instances d'une classe `Rectangle`. D'autre part, il est capable de modifier son comportement et sa sémantique : par exemple détection automatique d'erreurs d'accès à des attributs et modification du code source pour ajouter automatiquement, à la

volée, les accesseurs correspondants. La réflexivité est naturelle en Smalltalk et les programmeurs l'emploient très souvent.

- **Outils de développement de haut niveau.** Ceux-ci sont non seulement graphiques – et conviviaux – mais aussi de très haut niveau, car ils tirent parti de la réflexivité du langage. C'est en effet elle qui permet d'avoir de tels outils : le navigateur de classes pour l'édition, la compilation, l'organisation des classes et méthodes, la navigation dans les hiérarchies, etc. ; l'inspecteur pour étudier les attributs de tout objet ; le chercheur de méthodes (*method finder*) capable de trouver les méthodes répondant à un comportement donné (par exemple `(2 . 3 . 8)` donnera la méthode puissance d'un nombre), etc.
- **Le ramasse-miettes (garbage collector).** C'est un système automatique de collecte des objets inutilisés. Tant qu'une référence vers un objet existe, celui-ci reste en mémoire. Lorsqu'un objet n'est plus référencé – par exemple en affectant `nil` à la seule variable référençant l'objet – son espace mémoire est réclamé par le ramasse-miettes.

Le modèle objet de Smalltalk

Le modèle objet de Smalltalk est simple et uniforme. Il est décrit par un ensemble de 7 règles :

- **Règle 1** – Tout est un objet.
- **Règle 2** – Un objet est instance d'une classe.
- **Règle 3** – Une classe définit la structure (variables d'instance) des instances et spécifie les méthodes qui seront exécutées en réponse aux messages envoyés aux instances de la classe. Les variables d'instance sont privées à l'objet lui-même et toutes les méthodes sont publiques.
- **Règle 4** – Une classe peut hériter d'une autre classe par héritage simple (encore le choix de la simplicité !).
- **Règle 5** – Les objets communiquent entre eux par des échanges de messages. Lorsqu'un objet reçoit un message, la méthode correspondante de sa classe est recherchée, si elle n'existe pas, la recherche est étendue à sa super-classe.
- **Règle 6** – Toutes les classes appartiennent au même arbre d'héritage. La classe `Object` est la racine de cet arbre.
- **Règle 7** – Les classes sont des instances de classes appelées méta-classes.

C'est tout ! Smalltalk inclut des méthodes anonymes ou plus communément appelées « lambda expressions », « fermetures lexicales » en LISP ou « blocs » dans le jargon Smalltalk. Ces blocs permettent d'exprimer simplement des itérateurs et des conditions de manière élégante **[Book]**.

Quelques exemples

Avec Smalltalk tout commence avec l'espace de travail (*workspace*). C'est un peu comme le terminal virtuel du Linuxien bidouilleur. Il permet de tester en direct des fragments de code. Il est par exemple très pratique pour tester le comportement d'un objet que l'on ne connaît pas encore bien. Donc dès que son environnement Smalltalk (ici Squeak) est lancé, on commence par ouvrir un worskspace (espace de travail) : depuis le menu du monde (clic sur le fond), choisir *open->workspace* ou plus rapidement le raccourci [Alt]+[k]. Pour une version francisée de Squeak, consulter **[SquFr]**. Dans la suite, nous vous proposons d'expérimenter quelques exemples. Pour exécuter ces fragments de code, sélectionnez-les avec la souris et faites [Alt]+[d] (`do-it`) ou [Alt]+[p] (`print-it`) pour, en plus, afficher la valeur retournée par le bout de code. Il est aussi possible d'exécuter à partir de *do it* ou *print it* du menu contextuel (bouton milieu).

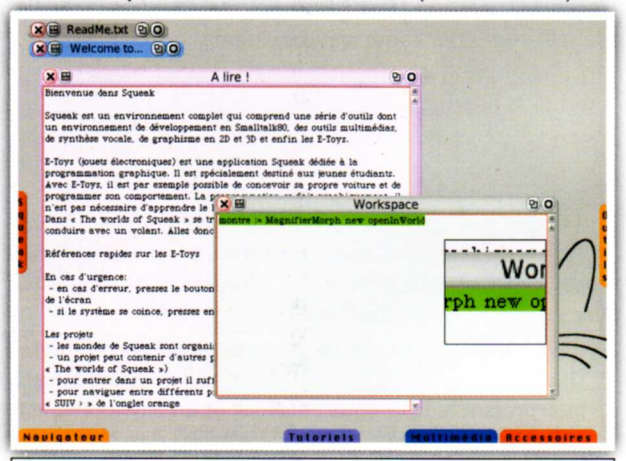


Fig. 2 : Un workspace. Un fragment de code qui, exécuté avec [Alt]+[d], a créé une loupe rectangulaire.

Création d'un objet

```
pi := -1 arcCos.  "la variable pi référence une instance d'un Float"
```

En sélectionnant `pi` et par un [Alt]+[i], nous appelons un inspecteur d'objet. Notez la forme de l'envoi du message `arcCos` à l'objet `-1`. Sélectionnez l'expression suivante et exécutez-la.

```
monEtoile := StarMorph new.  "la variable monEtoile référence une
instance d'un StarMorph"
```

Premier envoi de messages

Les exemples précédents contiennent déjà des exemples d'envoi de messages, poursuivez en suivant les exemples ci-dessous.

```
monEtoile openInWorld.    "Envoi du message unaire openInWorld à
monEtoile qui a pour effet d'afficher l'étoile"
monPoint := 50@10.        "Envoi du message binaire @ à l'objet 50
```

```
avec comme argument 10, le résultat de ce message est une instance
de Point, affectée à monPoint"
```

Ouvrez un inspecteur d'objet sur la variable monPoint. Vous pouvez observer les attributs x et y de l'objet.

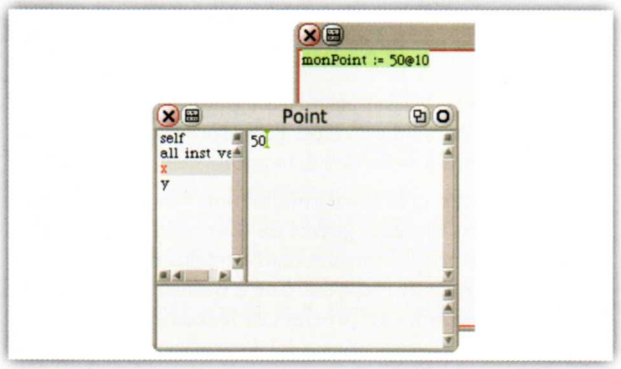


Fig. 3 : Un inspecteur ouvert sur une instance de Point

```
monEtoile position: 100@100.   "Envoi du message à mot-clé
position: à monEtoile avec comme argument le point 100@100 "
monEtoile color: Color orange.
```

Dans ce dernier exemple, il y a deux messages. Exécuté en premier, le message (de classe) unaire orange envoyé à la classe Color retourne une instance de cette classe définissant la couleur orange. Le deuxième message à mot-clé, color:, est envoyé à monEtoile et avec comme argument le résultat du message précédent. Vous pouvez sélectionner toutes les instructions pour exécuter d'un coup le script et obtenir une étoile de couleur orange.

Structure de contrôle

```
pi > 3 ifTrue: [Beeper beep].    "Beep si pi effectivement plus grand que 3"
```

pi > 3 est un message binaire > envoyé à pi avec comme argument 3. Ce message retourne un objet booléen. Ce booléen reçoit ensuite le message ifTrue: avec comme argument le bloc de code [Beeper beep]. Ce dernier bloc de code est exécuté si le receveur du message, pi > 3, retourne un boolean de valeur true. Il est ainsi possible de créer, depuis le langage lui-même, d'autres structures de contrôle. Il suffit pour cela de définir de nouvelles méthodes dans la classe Boolean. Notez cependant que ce besoin est rare car les structures de contrôle de Smalltalk sont riches et par défaut optimisées par le compilateur, donc avantageuses en coût à l'exécution.

Itérateurs

```
#(1 2 3) collect: [:x | x * x].  "Un print-it retourne la collection #(1 4 9)"
```

#(1 2 3) définit un tableau, une collection statique d'objets, ici des nombres. Le message à mot-clé collect: est un itérateur sur la collection, avec comme argument le bloc de code. Ce bloc de code est en fait une fonction anonyme avec comme argument x qui prend les différentes valeurs de la collection. Le caractère | dans le bloc sépare la déclaration de l'argument et le code du bloc qui est exécuté à chaque itération et dont le résultat est collationné. Cette collection est retournée à la fin de l'itération.

```
#(100 200 300 400) do: [:i | monEtoile copy position: i@i ;
openInWorld].  "crée une série de copies de monEtoile"
```

Ce dernier exemple reprend différents éléments des exemples précédents. Le caractère ; permet d'envoyer en cascade des messages à un même receveur, ici l'instance retournée par le message unaire monEtoile copy.

Les articles à venir

Notre présentation de Smalltalk s'achève ici, nous avons découvert l'historique du langage ainsi que quelques éléments de sa conception objet et de sa syntaxe. Nous espérons avoir aiguisé votre curiosité pour la suite. Même si vous n'envisagez pas de programmer en Smalltalk dans le futur, sa connaissance permet de mieux appréhender les mécanismes fondamentaux de la programmation par objet et par la même de devenir plus productif avec d'autres langages de programmation comme C++, Java ou Python. Dans les prochains articles, nous vous ferons découvrir la syntaxe et le modèle objet plus précisément, Seaside un époustouflant framework pour développer des applications web dynamiques **[Sea]**, les outils de développement et la programmation incrémentale, la réflexivité du langage, la machine virtuelle et plein d'autres surprises. En attendant, vous pouvez déjà consulter les quelques références proposées.

RÉFÉRENCES

- **[Sim]** Simula l'ancêtre des langages objet, http://fr.wikipedia.org/wiki/Simula
- **[Ske]** L'ancêtre des CAD, http://en.wikipedia.org/wiki/Sketchpad
- **[Xerox]** Le premier ordinateur personnel avec interface graphique, http://en.wikipedia.org/wiki/Xerox_Alto
- **[Squ]** Site de référence de Squeak http://www.squeak.org,
- **[SquFr]** Site francophone de Squeak, http://community.ofset.org/wiki/squeak
- **[Gnu]** GNU Smalltalk, http://www.gnu.org/software/smalltalk/smalltalk.html
- **[StX]** Un Smalltalk multiplateforme *gratuity*, http://www.exept.de/
- **[Dea]** Michael A. Hiltzik, *Dealers of Lightning : Xerox PARC and the Dawn of the Computer Age.*
- **[Sea]** Seaside, un framework pour développer des applications web dynamiques, http://www.seaside.st/
- **[Book]** Recueil de livres gratuits sur Smalltalk, http://www.listic.univ-savoie.fr/~ducasse/FreeBooks.html

H. Fernandes, S. Ducasse,
hilaire@ofset.org, ducasse@iam.unibe.ch,

Petite introduction théorique au temps réel !

Christophe Buffenoir

EN DEUX MOTS **Nous entendons souvent parler du temps réel. Mais que cela signifie-t-il exactement ? Les concepts associés à cette technologie sont utilisés pour les logiciels embarqués dans les avions, les voitures, les trains, mais également les usines, les imprimantes, etc. Nous verrons au fil de ce dossier ce que cette notion apporte et comment la mettre en pratique.**

Contrairement aux légendes que nous pouvons entendre autour de nous, le temps réel ne correspond pas à un logiciel qui répond le plus rapidement possible aux requêtes. En réalité, un logiciel temps réel répond selon des contraintes de temps déterminées.

Ainsi, pour répondre aux perturbations lors du pilotage d'un drone, les contraintes seront fortes et nécessiteront une certaine rapidité.

Au contraire, pour surveiller le niveau d'une cuve de 60 mètres cube avec une arrivée et une sortie d'eau ne pouvant dépasser 10 L/min, le logiciel de contrôle disposera d'un temps bien plus long.

Les exemples donnés ici montrent l'intérêt du temps réel : savoir réagir à un stimuli extérieur au système que nous appelons « STITR » (pour Système de Traitement de l'Information Temps Réel). Celui-ci peut être embarqué, c'est-à-dire incrusté dans un système comme par exemple une voiture ou un avion.

Le STITR permet l'automatisation d'un processus. Avant, les diverses opérations d'une usine étaient réalisées à la main, en tirant des cordes, en tournant des manivelles, etc. Aujourd'hui, un calculateur se charge d'actionner des éléments de contrôle (vannes automatiques, moteurs, pompes).

L'homme contrôle uniquement le bon fonctionnement et demande des opérations précises (réglage d'une consigne et arrêt du système entre autres). Non seulement, cette solution permet de réaliser des produits plus fiables et précis mais, en plus, elle limite les risques d'accident.

Les informations à traiter, quant à elles, ont une validité en fonction du temps. Elles ne doivent surtout pas surgir trop tôt ou trop tard. C'est donc à ce niveau que nous parlons de contraintes temps réel. Nous pouvons les classer parmi trois grandes catégories. Tout d'abord les contraintes strictes correspondent à l'invalidité des informations à la fin de l'échéance.

Les contraintes critiques concernent les informations provoquant des troubles graves de fonctionnement – voire provoquer une mort humaine dans certains cas particuliers – si le délai n'est pas respecté. Cette dernière situation est évidement à éviter à tout prix lors de la conception. Enfin, les contraintes dites « relâchées » gardent une certaine validité, moins importante, pour l'information après l'échéance.

La conception

Le développement de logiciel temps réel peut se réaliser selon trois approches différentes. La conception synchrone, la plus utilisée, consiste en l'écriture d'une simple boucle infinie. Les instructions sont exécutées les unes à la suite des autres. Aucun mécanisme bloquant ne doit être présent sous peine de geler le système.

L'approche multitâche préemptive sera détaillée plus bas. Dans ce cas, plutôt que de concevoir une seule boucle, l'application est décomposée en plusieurs boucles s'exécutant en parallèle. Un noyau temps réel est alors nécessaire pour gérer l'activation des différentes tâches. Enfin, l'approche objet est la plus récente. Elle reprend les principes de la POO.

Pour la conception d'un STITR multitâche préemptif, nous devons définir les différents agents actifs. Ces agents représentent différentes activités indépendantes pouvant communiquer entre elles. Chacun d'entre eux doit en plus répondre à des contraintes temporelles précises. En effet, après l'évènement déclencheur, l'agent doit réagir dans un temps minimal et surtout avant la limite de temps maximum. De même, nous pouvons déterminer un temps minimal avant lequel l'agent doit avoir fini son activité.

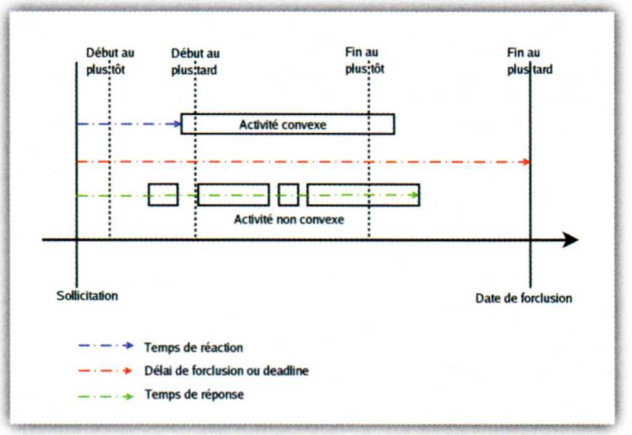


Fig. 1 : Les contraintes temporelles, la base du temps réel

Bien sûr, la principale contrainte reste la nécessité de terminer l'action avant la limite ultime, la date de forclusion. Toutes ces contraintes sont valables que l'activité soit convexe (elle se déroule alors sans interruption) ou non. La figure 1 nous montre plus clairement la succession de ces différentes échéances.

Comme nous venons de le voir, une activité a des contraintes en fonction d'un événement déclencheur, d'une condition d'activation. Mais sous quelle forme se présente cet événement ?

Cela peut être tout simplement une communication avec une autre activité. Ou encore une interruption déclenchée matériellement, provoquée par l'arrivée d'un signal spécifique sur le processeur appelé « interruption ». Mais cet événement peut également être la fin d'un blocage volontaire.

Attention au vocabulaire ! Nous parlons bien ici d'activité et non pas de tâche. Une activité peut effectivement être une interruption. Dans ce cas, elle arrête l'exécution du programme en cours : elle est prioritaire.

Nous parlons alors de préemption. L'interruption permet par exemple d'informer une tâche qu'un capteur a changé d'état. La gestion d'une horloge est un exemple typique d'utilisation d'une interruption. Cette dernière accède alors à un compteur afin d'activer les tâches périodiques au bon moment.

La gestion des activités nécessite aussi des mécanismes de synchronisation. Par exemple, si un signal sonore doit être émis lors de la pression d'un bouton. Une première tâche gère le *buzzer* et une seconde l'attente de pression.

Une synchronisation entre les tâches permet d'éviter le retentissement de l'alarme au mauvais moment. La solution généralement employée est le sémaphore. Celui-ci est constitué d'un entier et d'une file FIFO (*First In, First Out*).

L'entier ne doit surtout pas être modifié en dehors des deux opérations prévues à cet effet. La première consiste à « prendre » le sémaphore, c'est-à-dire à se placer au bout de la file d'attente.

La seconde opération, « donner », signifie que la partie protégée du programme s'est achevée et libère la ressource. Le listing 1 présente l'algorithme générique d'un sémaphore.

Listing 1

```
Structure Semaphore
{
     Valeur : entier
     Attente : file
}

Fonction Donner(Sem : Semaphore)
{
     Si Sem.Valeur < 0
     Alors Débloquer_suivant (Sem.Attente)
     FinSi
     Sem.Valeur = Sem.Valeur + 1
}

Fonction Prendre(Sem : Semaphore)
{
     Sem.Valeur = Sem.Valeur - 1
     Si Sem.Valeur < 0
     Alors Attendre(Sem.Attente)
     FinSi
}
```

La synchronisation peut se réaliser avec un sémaphore binaire. Elle ne concerne alors que deux tâches entre elles.

La valeur du sémaphore est initialisée à zéro. L'une des deux tâches sera bloquée tant que le sémaphore ne sera pas donné par l'autre tâche.

Une autre pratique concerne le partage des ressources. Un driver matériel, une variable commune, ou autre mécanisme ne pouvant

être utilisé par plusieurs tâches simultanément, sont des ressources non partageables.

Dans ce cas, soit l'opération d'accès est atomique (c'est-à-dire non divisible comme l'écriture d'un booléen) soit un sémaphore s'impose.

Ce dernier s'appelle alors « Mutex » ou « sémaphore d'exclusion mutuelle ». Sa valeur est initialisée à 1.

Avant d'accéder à une ressource, la tâche doit prendre le Mutex. Elle le donnera à la fin de l'opération. Cependant, le blocage doit être le plus court possible afin de pouvoir satisfaire les contraintes temporelles.

Non seulement les tâches se synchronisent entre elles mais, en plus, des moyens de communication inter-tâches sont disponibles. Si un tampon est nécessaire, une lecture et une écriture simultanées possible ou encore une synchronisation obligatoire, une file de messages est recommandée.

Cette dernière est également appelée « boîte aux lettres » pour son fonctionnement. Les messages sont envoyés par un ou plusieurs expéditeurs. L'unique destinataire récupère alors les « lettres » dans leur ordre d'envoi, c'est-à-dire en FIFO.

Par contre, si notre communication ne nécessite aucune des trois propriétés énumérées ci-dessus l'emploi d'un drapeau (une variable globale) protégé par un sémaphore binaire suffit amplement.

Souvent, le RTOS intègre directement le sémaphore dans les primitives du drapeau. Le rendez-vous, quant à lui, permet l'échange de données avec synchronisation mutuelle.

Le rôle du RTOS

Les différentes notions et mécanismes ci-dessus ne peuvent avoir lieu sans un noyau central orchestrant ce système.

Le RTOS (*Real-Time Operating System*) intervient alors. Ce dernier offre les différents services (comme les communications entre tâches et les mécanismes de synchronisation) et assure l'ordonnancement des tâches, l'exécution des interruptions, la gestion du temps, mais aussi les ressources système. Beaucoup de RTOS utilisent une HAL (*Hardware Abstraction Layer*), c'est-à-dire une couche basse permettant d'utiliser les fonctionnalités du RTOS sans se préoccuper des spécificités matérielles de la plate-forme.

L'ordonnanceur (*scheduler* en anglais) est l'élément central du RTOS. Il sert à déterminer la tâche à exécuter.

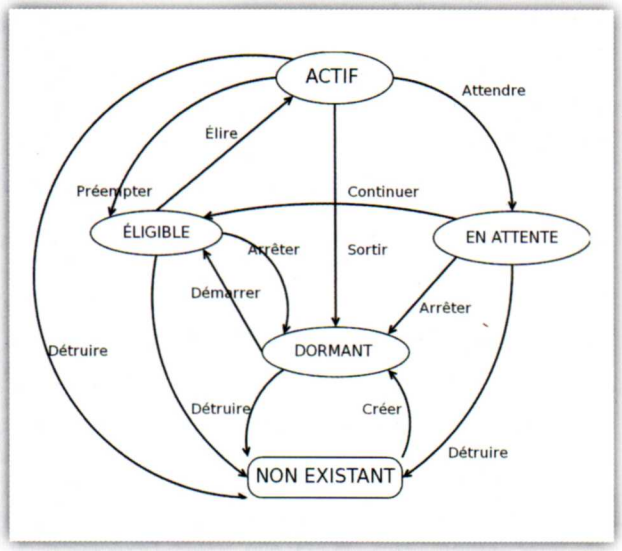


Fig. 2 : Les différents états possibles pour une tâche

En plus de cela, il attribue un état à chaque tâche : active, en attente, éligible à l'exécution ou encore inexistante.

Certains états sont spécifiques au RTOS utilisé, mais le principe reste le même. Tout d'abord une tâche doit être créée. Elle est alors dormante. Elle sera démarrée par une autre portion de code. Elle atteint alors l'état d'éligibilité et elle est prête pour s'exécuter.

Cependant, le processeur est une ressource non partageable gérée par l'ordonnanceur. La tâche devra donc attendre que la place se libère.

À la fin de son exécution, elle retournera soit à l'état d'éligibilité, soit à celui d'attente : la tâche a terminé son traitement pour atteindre une instruction bloquante.

Seulement, l'ordonnanceur choisit une tâche éligible selon des règles strictes.

Des stratégies différentes existent afin de satisfaire les spécificités des systèmes conçus. Certaines d'entre elles se basent sur des critères temporels, comme la stratégie EDF (*Earliest Deadline First*) activant la tâche dont le délai de forclusion se termine le plus tôt. La LLF (*Least Laxity First*) active la tâche ayant la plus petite laxité (c'est-à-dire au plus petit temps de marge entre la fin de la tâche et la fin du délai de forclusion).

Mais en général, les stratégies basées sur des priorités sont préférées pour leur déterminisme. La priorité peut être fixe ou changée dynamiquement au cours de l'exécution selon un mécanisme simple (aléatoire ou cyclique) ou basé sur une autre stratégie d'ordonnancement.

Une panoplie d'outils est ainsi disponible pour obtenir le comportement souhaité par le système. Le *Rate Monotonic* est la stratégie à priorités fixes la plus utilisée. Les priorités sont attribuées à la conception. Elles sont généralement inversement proportionnelles à la période d'activation.

Il subsiste cependant un problème non négligeable. Prenons le cas d'une ressource non partageable. Une tâche de très

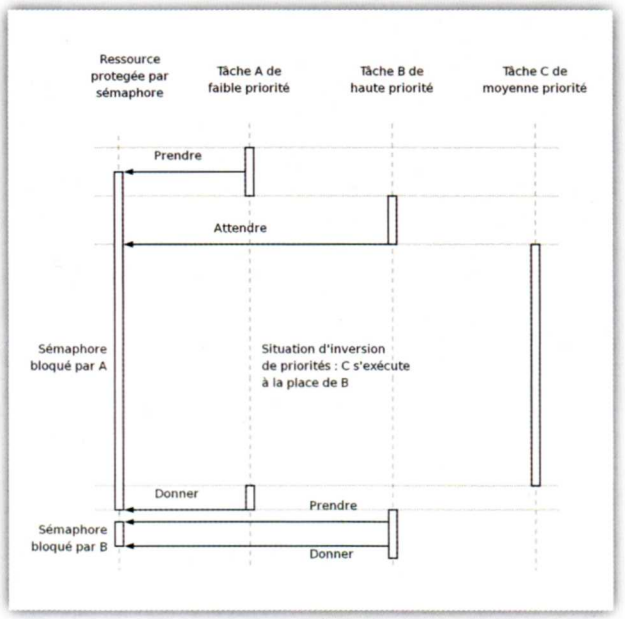


Fig. 3 : Une inversion de priorité dangereuse

basse priorité peut la bloquer alors qu'une autre, de priorité plus forte, tente d'y accéder à cette même ressource. Nous assistons alors à un blocage bien trop long de l'application, c'est-à-dire le temps que la tâche de basse priorité finisse le traitement. Elle sera régulièrement interrompue. Ce problème est connu sous le nom d'inversion de priorité : une tâche prioritaire est soumise à l'exécution d'une autre moins prioritaire. Deux solutions courantes sont proposées. La première, appelée « forçage de priorité », attribue la plus grande priorité existante à la tâche bloquante. La seconde, « héritage de priorité », attribue la plus forte des priorités des tâches bloquées à la tâche bloquante. Au lieu d'écrire soi-même l'algorithme du sémaphore, il est fortement conseillé d'utiliser les services fournis par l'éditeur du RTOS. En effet, ce genre de subtilités est implanté d'origine.

De même, une API générique et indépendante de la plate-forme permet d'utiliser facilement des périphériques matériels comme les ports séries et parallèles, l'écran ou tout autre grâce aux pilotes. Le plus simple et le plus sûr reste de bien étudier la documentation. Se pose alors la question fatidique : quel RTOS utiliser pour mon application ? Il n'existe pas de réponse universelle mais, dans tous les cas, des renseignements doivent être pris afin de respecter les normes.

Des réglementations existent et influent fortement sur le choix du système. Particulièrement pour l'aéronautique, la norme DO-178B doit être respectée à la lettre. Pour l'automobile, il s'agit de l'OSEK. Le développement temps réel peut nécessiter des contraintes très fortes en fonction de la criticité de l'application. Pour des besoins plus généraux, nous trouvons de nombreux systèmes. Le plus connu s'appelle VxWorks, édité par la société Wind River (cette entreprise est également citée pour ses actions sur les systèmes BSD, notamment FreeBSD).

Malheureusement, ce RTOS de qualité est également réputé pour son prix. Les dérivés de Linux, comme RTAI, LynuxWorks et d'autres, sont l'alternative à la mode. Cependant, pour ces derniers, un véritable noyau temps réel tourne en avant-plan et exécute Linux en tâche de très basse priorité. Ce principe exige un développement vertical contraignant. Deux autres RTOS, libres, peuvent attirer l'attention : RTEMS et eCos (présenté dans l'article ci-après).

Christophe Buffenoir,

http://www.buffenoir.org,

→ eCos, un RTOS adaptatif

Christophe Buffenoir

EN DEUX MOTS **Les systèmes nécessitant des contraintes temps réel possèdent souvent une architecture spécifique avec un système d'exploitation adapté. Pour s'adapter à ce marché, les RTOS passent donc par une phase configuration. Les performances et surtout l'empreinte de l'application dépendent fortement de cette étape cruciale.**

eCos signifie *Embedded Configurable Operating System*. Ce RTOS a été développé par Cygnus Solutions dès 1997. RedHat rachète la société en 1999, soit environ un an après la sortie de la première version d'eCos, et hérite alors de la propriété du système. Mais en 2002, la société au chapeau rouge décide de concentrer au maximum son activité sur Linux et procède à la dissolution de l'équipe eCos.

Seulement, un tel produit ne peut être abandonné aussi vite. La société eCosCentric est alors fondée peu après pour assurer une maintenance et fournir divers services pour eCos. Aujourd'hui, le système a fait ses preuves et se retrouve dans les produits de nombreux organismes comme le programme d'expérimentation spatial de nano-satellites Canadien CanX-2 ou encore chez Brother, Epson et divers constructeurs. Les applications d'eCos n'ont pas de limites et peuvent toucher aussi bien à l'automobile, à l'impression qu'aux jeux, aux routeurs réseau ou même aux applications multimédias.

Les principes de mise en œuvre

Comme pour tout système d'exploitation, eCos possède un noyau. Celui-ci est codé en C++ et permet de créer des applications dans ce langage ou bien en C. Plusieurs ordonnanceurs de tâches (*schedulers*) sont disponibles.

Nous les choisissons bien sûr en fonction des besoins spécifiques de l'application. Le noyau tourne au-dessus d'une HAL (*Hardware Abstraction Layer*) permettant d'utiliser tous les services du système indépendamment du matériel. Mais bien sûr, dans la limite des fonctionnalités de ce dernier. On retrouve aussi divers pilotes pour plusieurs architectures telles que x86 (386, 486, Pentium et assimilés), PowerPC, ARM, M68K et bien d'autres. La concurrence entre les constructeurs se ressent plus fortement dans l'embarqué que dans le monde de l'informatique personnelle : le but n'est pas d'avoir la machine la plus performante possible, mais la plus adaptée au besoin. Un RTOS se doit donc de gérer le maximum de plateformes possible pour répondre à des demandes bien précises. Pour en revenir au noyau d'eCos, celui-ci offre tous les services classiques (*timers*, compteurs, alarmes, files, tâches, interruptions et exceptions) en respectant les standards POSIX, µITRON ou non selon la configuration réalisée. La taille de l'OS reste raisonnable, entre 50ko et 14Mo selon les options. Un noyau simple avec scheduler a une empreinte d'environ 1 Mo.

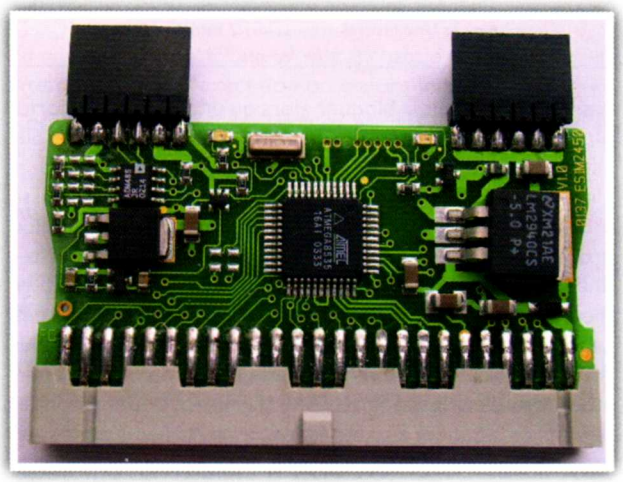


Fig. 1 : Un micro-automate industriel

Le développement d'une application avec eCos suit un cheminement bien précis. Après avoir téléchargé et installé eCos, il faut passer à la fameuse étape de configuration de l'OS. Pour cette tâche, Cygnus a réalisé un système très modulaire qui permet de paramétrer la moindre option de chacun des composants du RTOS. Ce large choix de configuration entraîne bien sûr une grande complexité et demande beaucoup de temps avant d'obtenir un système pleinement utilisable. Le résultat sera meilleur avec des paramètres choisis judicieusement. Après cette étape, il vous faudra compiler eCos pour le système cible. Les bibliothèques nécessaires à l'application sont embarquées dans le système. La dernière étape consiste à compiler notre application et à transférer le fichier binaire final sur la cible. Cette dernière opération n'est pas forcément aisée, tout dépend du système cible. Pour une cible avec un lecteur de disquette, il n'y a aucun problème, la plupart des ordinateurs individuels disposant eux aussi de ce type de lecteurs. Par contre, un système embarqué léger

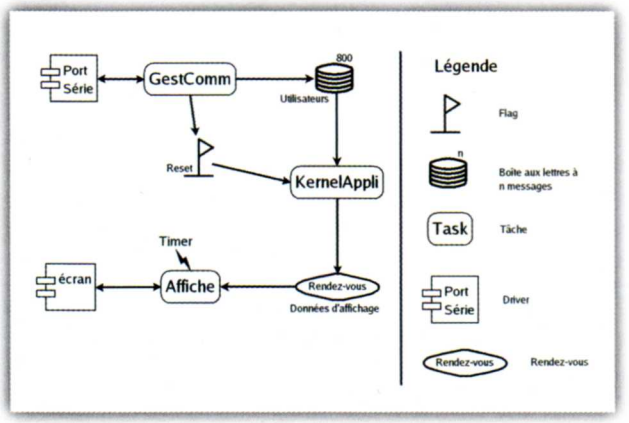


Fig. 2 : L'architecture de notre application

peut nécessiter l'achat de logiciel et de matériel spécifique pour la communication avec la station de développement (comme une sonde JTAG).

Notre exemple traitera du premier cas. Nous utiliserons donc RedBoot, le moniteur de ROM d'eCos. Son but est de démarrer sur la cible, de lancer le système d'exploitation ou encore de faire quelques opérations spécifiques (effacer la mémoire flash, etc.). Il émule également un serveur pour `gdb` et permet un débogage sur la cible.

Un peu de pratique

Prenons dans notre cas un cahier des charges simple. Nous allons concevoir un STITR qui dispose d'un port série et d'un écran (voire de deux ports série puisque la gestion de l'écran est identique). Les trames seront envoyées à partir d'un autre ordinateur. Nous avons deux types de messages possibles. Le premier est `RESET n` et le second `HELLO user`. Chacun d'eux est suivi d'un retour chariot `\r`. Le `n` correspond au nombre de secondes entre chaque rafraîchissement de l'écran et `user` à un nom de 32 caractères maximum. N'importe quel utilisateur peut dire `HELLO`, même s'il n'est pas référencé. Lors de la réception d'un message `HELLO`, l'utilisateur sera créé s'il n'existe pas et son compteur sera initialisé à 1. Si le nom a une longueur supérieure à la limite de 32 caractères, il sera tout simplement tronqué. Si l'utilisateur existe déjà, son compteur sera augmenté de 1 à chaque nouveau message `HELLO`.

L'écran devra alors faire apparaître le nombre de trames valides et invalides reçues (le message n'est pas cohérent, une erreur syntaxique empêche son traitement). Il affichera également le nombre de connexions de chaque utilisateur. Attention cependant, l'affichage sera une tâche autonome qui ne progressera pas à chaque trame reçue, mais à chaque rafraîchissement déclenché par le timer. Évidemment, même pour une petite application comme celle-ci, il nous faut un minimum de conception. Tout d'abord décomposons notre application en tâches et en interruptions. Dans le cas présent, trois tâches seront nécessaires. La première, que nous nommerons `GestComm`, réalisera l'acquisition des données. La seconde, `KernelAppli`, assurera la gestion des variables et des utilisateurs. La dernière, `Affiche`, permettra comme son nom l'indique l'affichage des données.

Ensuite, la localisation des ressources critiques obligera à placer des *mutex* (ou sémaphores d'exclusion mutuelle). En l'occurrence, nous avons le port série et l'écran. Ceux-ci ne doivent pas être utilisés en même temps par deux tâches différentes. Cela ne posera pas de problème particulier, puisque ces ressources ne sont utilisées que par une tâche chacune. Nous devons ensuite définir les moyens de communication entre les tâches. Dans notre application, `GestComm` doit envoyer les noms des utilisateurs reçus lors de chaque connexion à `KernelAppli`, ainsi que L'ordre de Reset. Il s'agit en fait de deux communications différentes. Enfin, après traitement, `KernelAppli` transmettra les diverses statistiques à `Affiche`.

Mais quel outil utiliser pour chacune des communications ? La transmission des noms des utilisateurs vers `KernelAppli` sera une file non bloquante de 800 éléments. Pourquoi mettre autant d'éléments ? La réponse est simple. `KernelAppli` tournera en boucle avec pour seule interruption les communications avec les deux autres tâches. En considérant qu'une itération de boucle peut durer 500ms, que le port série a un débit de 56kbps et qu'un flot constant de noms de 1 caractère parvient au système (il s'agit bien évidement du pire des cas), nous enregistrerons environ 600 noms. En rajoutant une marge pour éviter le débordement, la sécurité de l'application est garantie. De plus, en utilisant une machine relativement puissante comme un ordinateur de bureau, il est tout à fait possible de générer une telle quantité d'informations. Pour la communication entre `GestComm` et `KernelAppli`, nous utiliserons un drapeau (*flag*), c'est-à-dire une information booléenne. Il permettra d'informer de la remise à zéro des statistiques. Enfin, il reste la dernière communication qui se charge de transférer les informations d'affichage vers la tâche `Affiche`. Dans ce cas, l'utilisation d'un rendez-vous est conseillé. Malheureusement, eCos ne semble pas implémenter cette fonctionnalité dans la version actuelle. Il reste néanmoins possible de réaliser ce mécanisme avec un drapeau. La figure 1 récapitule ces différents mécanismes de communication. Après cette première phase d'analyse, nous n'avons encore choisi aucune stratégie d'ordonnancement. Chaque stratégie a des exigences propres, déterminant l'aptitude à gérer le système. Pour le *Rate Monotonic Scheduling*, le taux d'occupation du processeur au pire des cas ne doit pas dépasser 69%. Enfin, cette vision est plutôt simpliste. En réalité une formule permet d'obtenir la limite supérieure en fonction du

nombre de tâches et le résultat progresse en décroissant. Comme sa limite à l'infini est de 69%, ce nombre est acceptable dans tous les cas de figure. Le taux d'occupation total est la somme des taux d'occupation de chaque tâche. Ces derniers correspondent à la fraction du temps d'exécution au pire des cas sur le délai de forclusion au pire des cas. Ainsi, pour `KernelAppli`, la forclusion a lieu au bout de 500ms pour une durée d'exécution de 5ms au pire des cas. De même, `GestComm` doit se terminer avant 50ms pour 2ms d'exécution et Affiche 1000ms pour 5ms. Ainsi, le résultat du calcul est de 5,5%, soit très inférieur aux 69%. Le Rate Monotonic Scheduling est donc validé pour cette application. Les priorités des tâches doivent être définies selon la règle suivante : plus la période d'activation est courte, plus la priorité est élevée. Mais cette règle n'est pas immuable !

En effet, ici, `KernelAppli` devrait être plus prioritaire que `GestComm`. Mais aucun blocage n'intervient dans `KernelAppli`, donc cette tâche ne quitterait pas l'état d'éligibilité. Étant la tâche la plus prioritaire, les autres ne seraient tout simplement jamais exécutées. Ainsi, contrairement aux recommandations, `KernelAppli` sera la tâche la moins prioritaire. `Affiche` se place en deuxième position et `GestComm` sera la tâche la plus prioritaire. Ce qui semble entièrement logique aussi sur le plan des fonctionnalités. En effet, la récupération des données sur le port série est la phase la plus critique, alors que la tâche `Affiche` ne présente qu'une contrainte relâchée. De plus, l'utilisateur à l'écran ne verra pas de différences si l'affichage se produit 300 ms en retard. Au pire des cas, aucune information n'est écrite au cours de cette étape, cela ne sera que partie remise pour le prochain tour. Gardez à l'esprit qu'avec eCos la tâche la plus prioritaire est celle ayant le plus petit indice de priorité (comme pour `nice` sous Linux). L'algorithme de chaque tâche est relativement simple. Commençons par regarder la tâche `KernelAppli` dont vous trouverez le listing dans le fichier `kernelappli.c`, disponible sur le site du magazine. La boucle principale comportera tout d'abord une remise à zéro. Ensuite, les éléments de la file des noms passés par `GestComm` sont traités. Bien sûr, cette étape sera ignorée si la file est vide. Enfin, nous effectuons un test sur le rendez-vous avec `Affiche`. Si celui-ci est pris, les données d'affichage sont déposées écrasant ainsi les anciennes valeurs.

Nous plaçons alors le drapeau à « vrai » pour avertir de la fin de l'opération et libérer le

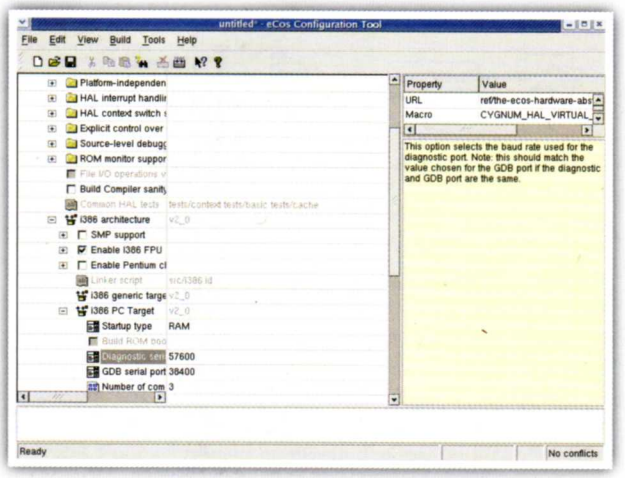


Fig. 3 : La configuration d'eCos

rendez-vous. La boucle se termine ici et reprend au début. La tâche `Affiche` (listing 1) sera encore plus simple. Dans une boucle infinie nous mettons en place une attente qui prendra fin lorsque le délai de rafraîchissement sera terminé. Lorsque cette étape s'activera, `Affiche` commencera le rendez-vous et se placera en attente des données. Une fois ces dernières récupérées, le rendez-vous sera rendu et les données affichées à l'écran jusqu'à la prochaine itération. Enfin, la tâche `GestComm` (fichier `gestcom.c` sur le site) écoutera le port série, en prenant les données caractère par caractère. Lorsqu'une chaîne est terminée par le retour de chariot, le traitement débute. Par contre, si le temps de saisie dépasse le *timeout*, l'information est retournée, l'ordinateur distant en est informé et l'attente d'une nouvelle chaîne commence. La structure de données de la file entre `GestComm` et `KernelAppli` sera une chaîne de caractères dont le premier élément aura une signification particulière.

S'il s'agit d'un `I`, le compteur des connexions invalides sera incrémenté. Au contraire, s'il s'agit d'un `U`, l'utilisateur doit être traité. Il reste cependant une fonction à étudier : `cyg_user_start`. Elle sert de point d'entrée pour notre application et contiendra donc toute la gestion, la création des mutex, des drapeaux, l'initialisation de certaines données et le lancement des tâches. Elle est la seule exécutée dans l'application si elle ne fait appel à aucune autre fonction ou tâche. Le code est disponible dans le listing 2.

Le rendez-vous reste assez simple à implémenter. Deux tâches ou plus doivent absolument s'arrêter en même temps pour qu'un transfert de données ait lieu. Pour cela, il nous faut un drapeau d'attente. En utilisant un bit par tâche, il est facile de contrôler que l'ensemble des tâches en attente soient prêtes. Lorsque la condition est réalisée, la tâche en charge du transfert réalise ses opérations puis active un dernier bit pour notifier la fin de la mise à jour. La tâche vide donc l'ensemble des bits du drapeau excepté celui de fin de mise à jour. Il sera retiré avant la prise du rendez-vous par cette même tâche. L'exécution reprend alors son cours. Évidemment, une et une seule tâche peut utiliser une synchronisation non bloquante, c'est obligatoirement elle qui assurera le transfert puisqu'elle est la seule à attendre les autres tâches. Ces

dernières attendent le signal donné par le bit de la tâche qui dirige les opérations, puis le bit de fin de mise à jour.

eCos, enfin...

Après cette étape de conception, et avant de continuer, passons par l'installation de notre RTOS. Vous pourrez récupérer eCos sur le site officiel, **http://ecos.sourceware.org** ou encore sur le site du magazine. Vous y trouverez la dernière version stable, c'est-à-dire la 2.0 à l'heure où nous écrivons ces lignes. À partir du site web, il vous suffira d'exécuter le fichier de quelques kilo-octets. Celui-ci téléchargera le RTOS et l'installera sur votre ordinateur hôte. Pour l'installation, il vous faudra extraire deux archives sur votre disque dur. La première contient le RTOS et le second les GNUtools, c'est-à-dire le compilateur, le débogueur, etc. Afin d'utiliser simplement les outils sans vous soucier du répertoire d'installation, vous pourrez inclure les répertoires `ecos-2.0/tools/bin et gnutools/i386-elf/bin` dans le `PATH` de votre Shell. La version de `gcc` ne rentrera pas en conflit avec celle installée sur votre système. En effet, un préfixe a été défini dans ce but à la création du binaire par eCosCentric. Passons alors à l'étape de configuration. Deux méthodes existent. La première utilise un outil graphique. Certes, cette méthode est la plus simple mais le logiciel présente encore beaucoup d'instabilité dans la version livrée avec eCos. Vous pourrez néanmoins télécharger un patch à cette adresse : **http://www.ecoscentric.com/devzone/configtool.shtml**. La deuxième méthode, pour les plus courageux, se réalise à l'aide d'un outil en ligne de commande. Tout d'abord, pour établir la configuration en mode graphique, lancez l'utilitaire `configtool`. Si vous avez indiqué le `PATH` comme conseillé ci-dessus, il vous suffira de taper `configtool` dans un terminal.

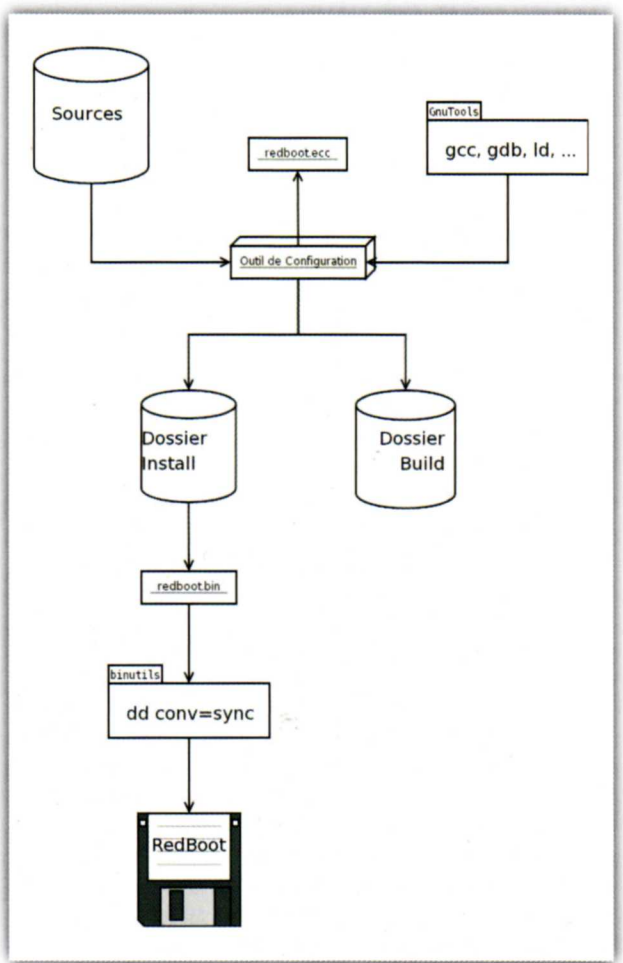


Fig. 4 : Le principe d'installation de RedBoot

La fenêtre apparaissant présente une hiérarchie d'options. Mais avant de changer un quelconque paramètre, il est nécessaire de spécifier l'architecture et les paquets souhaités. Allez dans le menu *Build* et choisissez *Templates*. Vous pourrez alors choisir votre cible. Ici, nous prendrons i386, la machine de test étant un Pentium III 750MHz. Notez cependant l'existence de la cible *i386 with Gigabit Ethernet*. Un peu plus bas, vous devez également sélectionner un set de paquets. Choisissez *Redboot* et validez. Le but est de créer une première disquette bootable qui servira à télécharger ensuite notre application. Cela est bien plus pratique lors de la phase de test. Cependant, il reste possible de recompiler l'application à la fin des tests afin de l'intégrer directement sur la disquette. Quelques messages d'erreur peuvent subvenir. Il s'agit de problèmes de conflits à résoudre. En général, l'outil de configuration vous propose des solutions standards que vous pouvez accepter sans crainte. Sur le coin inférieur droit de la fenêtre, la mention *No conflicts* gage d'une configuration cohérente. Cependant, même si une pré-configuration a été établie par défaut, il faut modifier certains paramètres.

Par exemple, vous pouvez activer l'option *eCos HAL-> i386 architecture-> Enable Pentium Class CPU features* si le processeur utilisé correspond et reste compatible. C'est le cas de notre machine de test. De même, ici nous avons activé la FPU en mode *Lazy Switch*. Dans la catégorie parente, le menu *i386 PC Target* mérite une attention particulière. Il permet d'affiner les options de démarrage et le débit des flux sur les ports série. Le type de démarrage doit être changé en *floppy* dans notre cas. En effet, le programme ne sera pas chargé en RAM avant son exécution, et de plus, la cible ne dispose pas de ROM pour l'application. L'option *GRUB* permet de démarrer à partir d'un disque dur. Afin d'assurer le fonctionnement souhaité de RedBoot, activez l'option *eCos HAL-> ROM Monitor support-> Behave as*

a ROM monitor. Ensuite, portez les débits des ports série à 57600bps.

Ceci est important car l'exemple n'est pas prévu pour une vitesse supérieure.Attention ! Le nombre de ports série par défaut est de 3. Il est vivement recommandé d'utiliser le nombre de ports de la cible plus un. En effet, le pilote d'affichage sur écran est considéré comme un port série !

Définissez également la console comme étant le port 2 (écran et clavier). Enfin, il faut désactiver une dernière option concernant la RAM. RedBoot ne détecte pas la mémoire étendue, ce qui pose problème lorsque plus de 640 ko sont disponibles. Enlevez donc la coche de l'option *Redboot ROM Monitor-> Validate RAM adresses during load*. Vous pourrez alors charger votre application sans difficultés. Sauvegardez votre travail avant de passer à la compilation de RedBoot. Choisissez dans la barre de menu *Build-> Generate Build Tree* et *Build-> Library*. Attendez la fin de la compilation (que vous pourrez suivre dans la partie inférieure de la fenêtre). Passez ensuite sur un terminal et insérez une disquette.Vous remarquerez trois répertoires nouvellement créés. Le plus important est celui terminant par `install`. Il contient les fichiers exécutables de Redboot ou les librairies eCos à inclure selon le cas. Tapez alors la commande suivante (à modifier selon le nom des répertoires) :

```
dd if=redboot-floppy-i386_install/bin/redboot.bin of=/dev/fd0 conv=sync
```

Une fois terminé, insérez la disquette dans la cible et démarrez-la. Des points devraient apparaître avant de terminer sur un écran d'information avec une invite de commande. Si ce n'est pas le cas, lisez bien la documentation et tentez de trouver les options qui posent problème. Testez en priorité sans les fonctionnalités spécifiques du Pentium. Vous trouverez les fichiers de configuration de la machine de test sur le site du magazine.Il existe aussi un outil de configuration en ligne de commande : `ecosconfig`. Il gère les diverses options offertes par l'interface graphique, les plantages en moins. Une documentation sur le site d'eCos vous permettra d'apprendre à l'utiliser. Vous pourrez également modifier les fichiers de configuration avec un simple éditeur de texte. Ils sont écrits dans un langage nommé CDL. En fait, il s'agit d'une description de chaque option.Vous y trouverez pour chacune d'elle le nom, la valeur par défaut, la valeur actuelle et un court paragraphe pour l'aide en ligne. La connaissance de ces fichiers est obligatoire pour activer l'instrumentation du kernel. En bref, si vous voulez aller plus loin, développer vos pilotes ou vos paquets, vous devrez les rajouter en utilisant le langage CDL dans les fichiers de configuration.

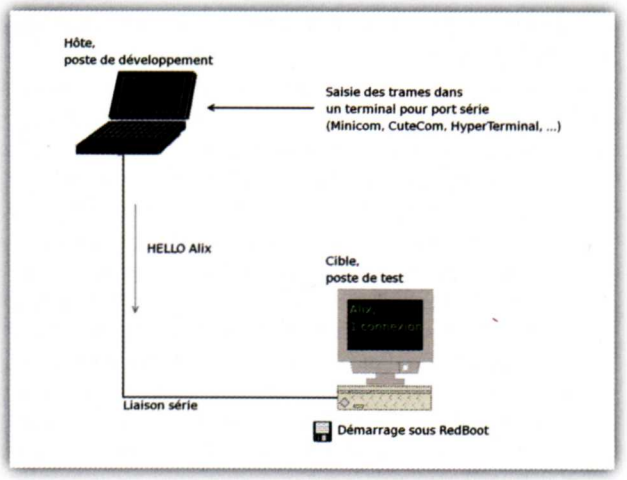


Fig. 5 : L'application de l'article en action !

Et un tour de plus...

Afin de poursuivre, il faut recommencer toutes les étapes de la configuration. Nous venons seulement de configurer RedBoot pour la cible. Ce moniteur de ROM se basant sur eCos suit les mêmes étapes de configuration que pour la création des bibliothèques eCos. Créez donc un nouveau fichier de configuration eCos, sélectionnez le *template* de l'architecture (le même que tout à l'heure) avec le set de paquets « *default* ». Vous devrez sélectionner les mêmes options que précédemment. Décochez cependant *Behave as a ROM monitor*. En effet, l'application n'est pas un moniteur de ROM, mais tourne au-dessus de RedBoot. Laissez l'option *Work with a ROM monitor : GDB_stubs*. De même, le type de démarrage doit être « RAM » et non pas « floppy ».

Une option, dans la catégorie *eCos Kernel-> Synchronization Primitives* permet de donner le nombre d'éléments dans les boîtes aux lettres, à savoir 800 dans notre cas. Enfin, vous devrez activer les pilotes des ports série matériels et les configurer sur 57600 bps.L'application comportera 5 fichiers C différents sans compter le `Makefile`.Attention à ce dernier, il contient un lien vers le répertoire des bibliothèques d'eCos pour la compilation. Une fois en place et le fichier `Makefile` correctement configuré, tout devrait se compiler aisément via la commande `make`. Mais le fait d'avoir un binaire ne suffit pas, il vous faudra également le tester. Un des fichiers contient la procédure principale, celle qui lance les diverses tâches. Ces dernières ne s'activent pas par défaut, il faut les créer et les débloquer. Ce fichier doit contenir également toutes les initialisations et la création des diverses entités. Il reste le point délicat du transfert de l'application vers la cible. Certes, avec RedBoot et un port série, il n'y a aucun problème, à part bien sûr une mauvaise configuration du port. Malheureusement, ceci est un cas peu fréquent dans l'embarqué. Beaucoup d'applications ne disposent pas de lecteur de disquette et encore moins de disque dur.

Le programme doit être transféré dans une ROM voire dans une mémoire flash.

Diverses interfaces existent et assurent non seulement pour le transfert, mais également le débogage. Cependant, elles sont bien trop souvent propriétaires. Par exemple, il faut débourser plusieurs milliers d'euros pour l'achat des logiciels sachant tirer parti d'une interface comme le Wiggler. Cette dernière se connecte au port parallèle de l'ordinateur. De plus, ces logiciels restent fortement dépendants de la plate-forme, rendant ainsi difficile tout portage vers d'autres architectures. Cependant, il existe une interface standardisée : le BDM (pour *Background Debug Mode*). La particularité de ce fonctionnement tient dans l'implémentation sur le processeur lui-même. Le port BDM correspond à 8 broches d'une carte mère sur lesquelles se branche une sonde spécifique. Cette dernière peut être un JTAG, un Wiggler ou une autre interface. Le fonctionnement est décrit dans le standard IEEE 1149.1. Il existe actuellement des sondes JTAG compatibles avec `gdb`. Néanmoins, cet équipement coûte cher malgré les services que cela peut rendre. Une autre difficulté de notre étape de transfert reste évidement la configuration de la communication entre l'ordinateur et la cible.

Effectivement, même avec un JTAG, il faut définir certains paramètres comme les registres utilisés. Pire encore, certains processeurs ont plusieurs jeux de registres, obligeant parfois à rentrer plusieurs configurations et à utiliser la bonne en fonction du mode de fonctionnement. Et même si la communication avec cette puce fonctionne, il n'en est pas forcément de même avec la RAM, la flash ou la ROM. Tout ceci nécessite encore du temps supplémentaire pour contourner la difficulté. L'étape la plus simple peut donc se transformer en vrai casse-tête. Elle oblige d'ailleurs certaines entreprises à se concentrer sur un matériel spécifique et éviter d'en changer sous peine de devoir repenser leur cycle de développement à chaque changement matériel, et parfois même de devoir acheter des outils très coûteux. Mais concrètement, comment faut-il transférer le fichier pour notre exemple ? Pour notre application un simple câble série croisé suffira pour assurer le transfert et également le débogage.

D'abord, allumez la cible avec votre disquette RedBoot. Tapez la commande `help` pour avoir un minimum d'aide à l'invite. Il vous faudra changer de console en écrivant `channel 0` (si le port série utilisé est le COM1). Il faut aussi exécuter un logiciel de terminal sur port série de l'ordinateur hôte (la station de développement). Vous pourrez utiliser `minicom` sous Linux/BSD. Si vous avez respecté la configuration des ports décrite dans cet article, vous devrez utiliser les valeurs : `57600 8N1`. La console change alors de terminal. RedBoot se pilote à partir de l'ordinateur hôte. Tapez `load -m yModem` et envoyez l'application avec le protocole yModem. Passez ensuite sur le channel 2. Il vous suffira alors de taper la commande `go` pour lancer l'application. La solution précédente, bien pratique, permet de tester l'application en visualisant les résultats. Mais il est également possible d'ouvrir une session `gdb` distante, par laquelle se déroule le téléchargement de l'application et son exécution. De même, il est possible d'instrumenter le noyau eCos, c'est-à-dire d'analyser le fonctionnement même du système et d'apercevoir les durées d'exécutions. Ce point est primordial pour les applications critiques.

En effet, il est très important de déterminer ainsi si une contrainte critique n'est pas respectée à un moment donné lors de l'observation.

Nous ne le ferons pas ici. De plus, eCos gère les exceptions et les assertions. Ces deux concepts fortement utiles permettent de garder le contrôle du système. Il faut noter un point pour les assertions. Lors de l'arrêt du logiciel, l'équipement est censé récupérer sa position mécanique au repos. Ceci est important par exemple pour une usine, car cette situation correspond à un arrêt total de la production mais surtout à la configuration la plus sûre. Les vies humaines et le matériel sont importants, il faut donc les préserver.

Après une assertion, le système pourra redémarrer selon les procédures normales. Ceci ne veut pas dire que la livraison d'un logiciel bogué est permise mais plutôt qu'il faut toujours prévoir une issue en cas de bogue, que ce soit logiciel, électrique ou mécanique. L'application présentée ici aurait pu être écrite autrement, en synchrone ou encore en objet avec le langage C++. La première de ces deux approches aurait même été préférable pour la taille et surtout l'occupation du processeur.

Cependant, cette solution est tout aussi valable. Elle permet une présentation succincte des possibilités d'un tel système. Il vous sera aisé d'approfondir les connaissances acquises dans cet article. En plus de la documentation très claire et complète, vous trouverez un livre disponible au format PDF sur le site officiel d'eCos. Tous ces documents ne sont bien sûr disponibles qu'en anglais.

Certains des principes évoqués ici peuvent également être appliqués dans du développement plus traditionnel, comme une application de gestion ou même un site web. Dès lors que plusieurs processus s'exécutent en parallèle, les besoins de blocage et de synchronisation spécifiques se font sentir. Prenez l'exemple d'un site web dynamique écrit en PHP. À chaque fois qu'un internaute consulte une page, une instance de PHP s'exécute et crée la page demandée.

Celle-ci peut nécessiter l'écriture de données et si plusieurs internautes accèdent aux mêmes variables sans aucune protection, cela peu vite entraîner la corruption du système. Se frotter aux contraintes du monde du temps réel aide aussi à mieux appréhender les problèmes de synchronisation dans l'informatique classique.

La gestion des tâches

cyg_thread	type du *thread*
cyg_thread_entry_t	entrée d'un thread
cyg_thread_create	fonction de création d'une tâche
cyg_thread_resume	fonction de réveil d'une tâche
cyg_thread_delay	fonction de mise en attente de la tâche courante sur un délai

Syntaxe des fonctions

```
cyg_thread_create(cyg_shed_info priorité,
          cyg_thread_entry_t point_d_entrée,
          cyg_addrword_t data donnees_a_transmettre,
          char* nom, void* pile,
          cyg_ucount32 taille_de_la_pile,
          cyg_handle_t* poignée, cyg_thread* tâche);
cyg_thread_resume(cyg_handle_t poignée);
cyg_thread_delay(cyg_tick_count_t délai);
```

Les tâches doivent tout d'abord être déclarées avec le type cyg_thread. Les points d'entrée correspondants devront être déclarés localement avec le même nom que la fonction utilisée. Il s'agit en fait du code exécuté de la tâche. Une poignée permet de manipuler la tâche tout au long du programme. Après les différentes déclarations, les tâches devront être créées puis réveillées. Rien ne vous empêche d'utiliser un même point d'entrée pour plusieurs tâches. Vous pourrez même créer un tableau de poignées pour réaliser des traitements itératifs.

La gestion des ports série

cyg_io_handle_t	type de la poignée
cyg_serial_info_t	type des paramètres de configuration
cyg_io_lookup	fonction d'attribution du pilote matériel à la poignée
cyg_io_set_config	fonction d'attribution des paramètres
cyg_io_read	fonction bloquante de lecture sur le périphérique

Syntaxe des fonctions

```
Cyg_ErrNo cyg_io_lookup(char* chemin,
          cyg_io_handle_t poignée);
Cyg_ErrNo cyg_io_setconfig(cyg_io_handle_t poignée,
          cyg_uint32 clé, void *paramètres,
          cyg_uint32* taille);
Cyg_ErrNo cyg_io_read(cyg_io_handle_t poignée,
          void* buffer, cyg_uint32* taille);
```

La gestion des boîtes aux lettres

cyg_mbox	type de la boîte aux lettres
cyg_mbox_create	fonction de création de la boîte aux lettres
cyg_mbox_put	fonction bloquante d'envoi de données
cyg_mbox_get	fonction bloquante de réception de données
cyg_mbox_tryget	fonction non bloquante de réception de données

Syntaxe des fonctions

```
void cyg_mbox_create(cyg_handle_t* poignée,
          cyg_mbox* boîte_aux_lettres);
cyg_bool_t cyg_mbox_put(cyg_handle_t* poignée,
          void* item);
void* cyg_mbox_get(cyg_handle_t* poignée);
void* cyg_mbox_tryget(cyg_handle_t* poignée);
```

La gestion des drapeaux

cyg_flag_t	type du drapeau
cyg_flag_init	fonction d'initialisation du drapeau
cyg_flag_setbits	fonction de mise à 1 de bits du drapeau
cyg_flag_maskbits	fonction de mise à 0 de bits du drapeau
cyg_flag_wait	fonction bloquante de test du drapeau
cyg_flag_poll	fonction non bloquante de test du drapeau

Syntaxe des fonctions

```
void cyg_flag_init(cyg_flag_t* drapeau);
void cyg_flag_setbits(cyg_flag_t* drapeau,
          cyg_value_t bits_concernés);
void cyg_flags_maskbits(cyg_flag_t* drapeau,
          cyg_value_t bits_concernés);
cyg_flag_value_t cyg_flag_wait(
          cyg_flag_t* drapeau,
          cyg_value_t pattern,
          cyg_flag_mode_t mode);
cyg_flag_value_t cyg_flag_poll(
          cyg_flag_t* drapeau,
          cyg_value_t pattern,
          cyg_flag_mode_t mode);
```

Listing 1 : fichier affiche.c

```
#include "commun.h"

void Affiche (cyg_addrword_t data){
    while(1)
    {
        int i=0;
        // Attente de rafraîchissement
        cyg_thread_delay(AffRafraichissement
                    * RESOLUTION);

        // Prise de rendez-vous
        cyg_flag_setbits(&RendezVous, 2);
        cyg_flag_wait(&RendezVous,
                    1, CYG_FLAG_WAITMODE_AND);
        // Attente de la fin de rendez-vous
        cyg_flag_wait(&RendezVous,
                     4, CYG_FLAG_WAITMODE_AND);

        // Affichage
        for(i=0; i < 30; i++)
            printf("\n\r");

        printf("Statistiques, connexions invalides : %d\n",
                     AffNbInvalides);
        printf("Nombre de connexions réussies : %d\n",
                     AffNbTotal - AffNbInvalides);
        printf("Utilisateurs et nb de connexions :\n");

        for(i=0; i < AffNbUtils; i++)
        {
            printf("%d : %s, %d connexion(s)\n",
                     i+1, AffUtilisateurs[i],
AffNbConnexions[i]);
            free(AffUtilisateurs[i]);
        }
    }
}
```

Listing 2 : Fichier main.c

```
#include "commun.h"
void cyg_user_start(void)
{
        cyg_thread_entry_t GestComm;
        cyg_thread_entry_t Affiche;
        cyg_thread_entry_t KernelAppli;
        cyg_mbox_create(&hdlFileUtils, &fileUtils);
        cyg_flag_init(&RendezVous);
        cyg_flag_setbits(&RendezVous, 0);
        cyg_flag_init(&Reset);
        cyg_flag_setbits(&Reset, 0);
        AffNbInvalides = 0;
        AffNbTotal = 0;
        AffNbUtils = 0;
        AffUtilisateurs = (unsigned char **)
                    malloc(sizeof(unsigned char *));
        AffNbConnexions = (int *) malloc(sizeof(int));
        AffRafraichissement = 30;
        // Création des threads, plus le nombre
        // en premier paramètre
        // est élevé, plus la priorité est basse
        cyg_thread_create(10, GestComm,
                    (cyg_addrword_t) 0,
                    "GestComm", (void *)stackGestComm, 131072,
                    &hdlGestComm, &PtGestComm);
        cyg_thread_create(11, Affiche, (cyg_addrword_t) 1,
                    "Affiche", (void *)stackAffiche, 131072,
                    &hdlAffiche, &PtAffiche);
        cyg_thread_create(12, KernelAppli,
                    (cyg_addrword_t) 1, "KernelAppli",
                    (void *)stackKernelAppli, 131072,
                    &hdlKernelAppli, &PtKernelAppli);
        // Exécution des deux threads
        cyg_thread_resume(hdlAffiche);
        cyg_thread_resume(hdlKernelAppli);
        cyg_thread_resume(hdlGestComm);
}
```

Christophe Buffenoir,

http://www.buffenoir.org,

→ Perles de Mongueurs

Stéphane Payrard

Les nouvelles de la communauté Perl d'avril – Mehari : un miniPerl6 sur une puce

Nous apprenons que Sun Microsystems échantillonne Mehari, un processeur SPARC supportant MiniPerl6, un sous-ensemble de Perl6.

Mehari est un processeur qui tourne le dos à l'architecture RISC pure : certains de ses *opcodes* sont de haut niveau. Comme les architectures SPARC traditionnelles, Mehari propose un environnement programmable avec un shell interactif avant de *booter* le système d'exploitation.

Mais cet environnement est MiniPerl6 au lieu de Forth. MiniPerl6, c'est aussi un *bytecode* de haut niveau. Il y aura donc des compilateurs pour les langages modernes de script autres que Perl6.

Extension du Risc par MiniPerl6

Des analyses ont montré que dans un système Unix, les mêmes structures de données sont réimplantées dans beaucoup de librairies et souvent de manière peu efficiente.

Le processeur Mehari factorise certaines de ces opérations en les implantant dans le *hardware*. Il gère « en dur » les chaînes Unicode, le moteur de règles Perl6, les tables associatives (connues sous le nom de *hash* en Perl), et les tableaux creux. Les tableaux creux permettent l'accès par index rapide comme les tableaux normaux et l'insertion rapide comme une liste, proposant ainsi le meilleur des deux mondes.

Mehari peut utiliser les librairies compilées pour du code SPARC traditionnel.

Mais pour un programme utilisant ces librairies, l'éditeur de lien remplace certains points d'entrée par l'appel correspondant en microcode miniperl6 résidant en ROM sur la puce.

Microprogrammation MiniPerl6

Les algorithmes décrits ci-dessus sont bien connus et peuvent donc résider en *firmware*.

Mais Perl6 proposera des opérations de haut niveau encore mal spécifiées. Il s'agit du moteur de règles et du système de ventilation des multiméthodes. Ces systèmes sont aussi microprogrammables, mais ce microcode là réside en RAM.

Le système de règles est un descendant du système d'expressions régulières de Perl5. Il sera plus propre et plus puissant et supportera la définition de grammaires. Le moteur qui l'implante est une synthèse des moteurs pour grammaire d'opérateurs, d'automates à la Yacc et de moteurs d'analyseur récursif descendant.

De même, Perl6 propose un système de ventilation de multiméthodes. En programmation objet classique, la méthode appelée est celle de la classe de l'objet contenu dans le premier paramètre. Les multiméthodes étendent ce système de ventilation à plusieurs paramètres.

En Perl6, l'inférence de type est encore à définir. Elle est trop complexe pour être implantée en *hard*. Ce sera un système de greffons qui ne modifiera pas la sémantique du langage. Donc MiniPerl6 et Mehari ne sont pas concernés.

Un peu d'histoire

Mettre dans le hardware des opérations de haut niveau grâce à la microprogrammation n'est pas nouveau, mais contraste avec l'approche qui a conduit aux premiers succès de Sun.

Les premières stations Sun à base d'un processeur standard, le 68000 de Motorola, avaient balayé les stations Symbolics qui utilisaient du matériel propriétaire.

De plus, Symbolics avait transformé en système propriétaire des logiciels développés au MIT. Cela avait d'ailleurs provoqué l'ire de Richard Stallman.

Quelques années plus tard, Sun, alors leader dans le monde Unix, a provoqué la colère de ses premiers clients, en particulier les universités, en décidant de vendre séparément ses compilateurs.

Cela a certainement contribué à la popularité de GCC. Notons que c'est Michael Tiemann, maintenant directeur technique de RedHat qui a porté le dorsal de GCC pour supporter l'architecture SPARC.

Stratégie, DRM et open source

Ces réflexions historiques et l'actualité récente (et plus spécifiquement la mise à disposition d'une bonne partie des sources de Solaris) amènent à penser que Sun va mettre Mehari en *open source*.

Steve Jobs a failli choisir Mehari, mais a renoncé quand Sun a refusé l'inclusion d'un système de gestion des droits (DRM) au sein du processeur, condition indispensable à sa stratégie multimédia.

Quelques détails sur les processeurs Mehari

Le processeur sera *multicore*, mais avec une seule unité de traitement de règles. Sun prépare une autre version avec un processeur asynchrone, car propager le signal d'horloge consomme beaucoup d'énergie sur un processeur de cette complexité. Son nom de code est « Ichtyus ».

Stéphane Payrard (cognominal),

Paris.pm – stef@mongueurs.net

À vous !

Envoyez vos perles à perles@mongueurs.net. Elles seront peut-être publiées dans un prochain numéro de Linux Magazine.

LIENS

Quelques pistes pour comprendre les technologies évoquées :

- Le système de règles en Perl6 : http://dev.perl.org/perl6/doc/design/syn/S05.html
- Les trois types d'analyseurs de grammaire dont le moteur de règles de Perl6 est la synthèse : http://en.wikipedia.org/wiki/Operator-precedence_parser, -http://en.wikipedia.org/wiki/Recursive_descent_parser, http://en.wikipedia.org/wiki/LALR_parser
- L'objet et le *multidispatch* en Perl6 : http://dev.perl.org/perl6/doc/design/syn/S12.html
- Vérification de systèmes de types ou d'inférence de types comme greffons optionnels : http://pico.vub.ac.be/~wdmeuter/RDL04/papers/Bracha.pdf
- Précédentes occasions manquées entre Sun et Apple : http://www.theregister.co.uk/2006/01/12/sun_apple_snapple/
- Analyse de Gilles Gravier, responsable Stratégie « Technologies de Sécurité » de la société Sun Microsystems à propos du DRM et de l'open source : http://blogs.sun.com/roller/page/gravax?entry=drm_in_an_open_source
- Shell interactif en environnement mininal : http://en.wikipedia.org/wiki/Busybox, http://fr.wikipedia.org/wiki/Microcode, http://fr.wikipedia.org/wiki/Open_Firmware
- Deux présentations des principes des processeurs asynchrones : http://research.sun.com/features/async/, http://user.it.uu.se/~eh/courses/tic/Papers/Lectures5-6/p1.pdf
- Douglas Adams : http://en.wikipedia.org/wiki/So_Long,_and_Thanks_for_All_the_Fish

→ Mise en place et étude du développement Dreamcast

Jonathan Muller

EN DEUX MOTS **C'est lors d'une install party à Metz que j'ai rencontré une personne qui lançait un rendu de Tux sur un écran géant à partir de sa Dreamcast. C'est de là que m'est venue l'idée de développer pour cette console que j'adore.**

1. Introduction à la Dreamcast

Qui ne connaît pas des titres comme SoulCalibur, Skies of Arcadia, Grandia 2, Sonic adventure ou encore le magnifique Shenmue? Vous me suivez, je vais vous parler de la Dreamcast. Dernière console en titre de Sega(tm), cette merveilleuse console à la mort prématurée fait encore parler d'elle au sein d'une communauté très active sur Internet. En effet, de nombreuses personnes ont décidé de programmer des jeux, des utilitaires, des émulateurs sur cette console. Je pourrais citer le très célèbre ScummVM disponible également sur PC et qui permet de lancer la plupart des jeux de Lucas Art(tm) (Monkey Island, Day of the tantacle, Sam & Max...) ou encore des émulateurs (cpc, megadrive, néo-géo en cours de développement). Parmi les jeux *Homebrew*, pour ma part, deux se détachent du lot : Feet of fury [1] et Alice Dreams [2].

On peut dire que la communauté est vraiment productive et motivée. J'en arrive au but de cet article : mettre en place un environnement de compilation/développement pour Dreamcast au sein d'un système GNU/Linux. Des packs Windows existent pour l'environnement de développement dev-cpp, mais je n'aborderai pas ce type d'installation. Avant toute explication concernant l'installation de l'environnement, passons en revue quelques informations techniques de la console.

2. Historique et Hardware

Sortie en novembre 1998 au Japon, la Dreamcast est disponible en Europe et aux États-Unis à partir de Noël 1999. Sega(tm) frappe fort avec sa nouvelle console 128 bits (contre 32 pour la Playstation).

Fig. 1 : La Dreamcast, sa manette et la carte mémoire

Composant	Description
Microprocesseur	RISC Hitachi SH4 128 bits 200 Mhz, 360 MIPS, 1,4 GFLOPS
Processeur graphique	NEC PowerVR (plus de 3 millions de polygones par secondes)
Processeur sonore	Yamaha RISC 32 bits (64 canaux ADPCM)
Mémoire	16 Mo (64Mbit SDRAMx2)
Système d'exploitation	Version optimisée de Windows CE
Lecteur CD	12x
Couleurs	16,7 millions
Carte mémoire	V. M. S.
Taille	190 x 195 x 78 mm
Poids	2 kg

Tableau 1

A ce sujet, il existe un très bon article paru en 2004, disponible sur le site de jeux vidéo : jeuxvideos.com [3].

3. Introduction

Le développement amateur pour les différentes consoles s'appelle le « Homebrew ». Il est bien évidemment possible de programmer sur Dreamcast à condition de disposer des bons outils :

- Un kit de développement ;
- Un moyen de transférer les données du PC vers la console afin de tester les programmes que nous aurons écrits.

Les différentes manipulations seront détaillées ci-après. Fort heureusement, il existe de nombreux logiciels disponibles sur Dreamcast. Le mécanisme de création d'un logiciel est le suivant :

- Compiler son projet avec la bibliothèque KOS (Kallisti OS) [4] écrite par Dan Potter. Des *snapshots* Subversion sont disponibles ici [5] (cf. la procédure d'installation détaillée dans le chapitre suivant) ;
- 3 choix s'offrent à nous pour tester notre programme :
  - Envoyer le programme via le port série de la Dreamcast. Pour ce faire, il faudra se fabriquer ou s'acheter un *coder cable* ;
  - Rendre le binaire « bootable » au format Dreamcast, le graver et mettre le CD dans le lecteur de la console ;
  - Rendre le binaire « bootable » et le tester sur un émulateur. Malheureusement, il existe peu d'émulateurs. Un seul est disponible : Chankast [6] qui apparemment fonctionnerait plutôt bien, l'inconvénient majeur étant qu'il n'est disponible que sous Windows. J'ai bien essayé de le lancer via Wine [7], mais sans réel succès.

Avant de passer à la partie technique de l'article, quelques informations pécuniaires pour ceux qui souhaitent se lancer dans le développement Dreamcast :

- Une console d'occasion (manette incluse) se trouve pour une cinquantaine d'euros chez n'importe quel revendeur ;
- Le coder cable se trouve sur différents sites spécialisés pour vingt, vingt-cinq euros environ.

Au final, on peut se construire une plate-forme de développement on ne peut plus intéressante pour soixante dix euros environ. Passons maintenant aux choses sérieuses : l'installation de l'environnement compilation/développement sur notre GNU/Linux préféré.

4. Kallisti Operating system (KOS)

KOS est l'acronyme de *Kallisti Operating System* qui sera désigné ainsi dans la suite de l'article. Ce système d'exploitation est considéré comme étant **LA** référence pour le développement Dreamcast. C'est un pseudo système temps réel pour les consoles de jeux vidéo :

- GBA (Game Boy Advance);
- PS2 ;
- Dreamcast.

Son état d'avancement étant le plus important, principalement grâce au grand nombre de gens investis sur cette console, le port Dreamcast est le plus utilisé. Il peut être assimilé à un noyau, comme Linux ou BSD, que nous allons utiliser via une bibliothèque (`libkallisti.a`) qui sera liée à nos propres programmes. Il est aussi possible d'écrire de nouveaux modules qui seront liés dynamiquement à notre programme. Nous pouvons alors envisager d'ajouter de nouveaux systèmes de fichiers, des pilotes de périphériques, etc. Les possibilités sont infinies.

KOS étant un pseudo système temps réel, il est évident qu'il possède des systèmes de fichiers particuliers. En effet, on peut :

- Lire sur le CD-ROM (accès par `/cd`) ;
- Écrire sur l'ordinateur relié par le coder cable (accès par `/pc` ; pratique pour faire des impressions d'écran) ;
- Lire sur le romdisk (accès par `/rd`). Ce système de fichiers est directement lié au binaire du programme lors de la compilation (sa taille est limitée à 16 Mo).

KOS dispose aussi de pilotes de périphériques pour le clavier, la souris, les manettes et la carte réseau (10/100Mbits). Assez de théorie, passons à la pratique avec l'installation et l'utilisation sur quelques exemples de cette fameuse bibliothèque.

Tout d'abord, récupérons les sources de KOS :

- `kos-ports` : portage de différentes bibliothèques (jpeg, png, ogg, mp3...) **http://gamedev.allusion.net/svn/snapshots/kos-ports-snapshot-20050618.tar.bz2**
- `kos-snapshot` : sources de Kos **http://gamedev.allusion.net/svn/snapshots/kos-snapshot-20050618.tar.bz2**

Il faut tout d'abord mettre en place un système de compilation croisée (*cross-compiler* ou *toolchain* en anglais) afin d'offrir la possibilité de compiler sur PC et d'exécuter sur Dreamcast. Deux compilateurs croisés seront nécessaires :

- Le premier afin de générer du code pour le processeur SH-4 ;
- Le second afin de générer du code pour le processeur ARM chargé de s'occuper du son.

ATTENTION

Un problème connu des versions de gcc supérieures à 3.0.4 nous oblige à utiliser une version 3.0.4 ou inférieure pour le cross-compilateur à destination du processeur ARM. En ce qui concerne le compilateur croisé à destination du processeur SH-4, j'ai utilisé une version 3.4.3 avec des patchs concernant la gestion des threads (ces derniers seront fournis dans une archive `.tar.gz`). Une mise en place incorrecte de ces deux compilateurs croisés vous assurera de magnifiques `kernel panic`, pour peu que vos programmes utilisent les bibliothèques sonores.

Liste des archives à récupérer [8] :

- `gcc-core-3.4.3.tar.bz2`
- `gcc-g++-3.4.3.tar.bz2`
- `gcc-core-3.0.4.tar.gz` (pour ARM)
- `newlib-1.12.0.tar.gz`
- `binutils-2.15.tar.bz2`
- `binutils-2.11.2.tar.gz` (pour ARM)
- `kos-ports-snapshot-20050618.tar.bz2`
  `kos-snapshot-20050618.tar.bz2`

Je vous mets à disposition un script servant à construire les compilateurs croisés [9] à partir de ces archives : `dc-chain-0.1.tgz` (les patchs y sont inclus).

Copiez alors ce script dans le répertoire de votre convenance, par exemple /tmp/testKOS.

```
tar xzf dc-chain-0.1.tgz
dc-chain-0.1
```

Copiez gcc-core-3.4.3.tar.bz2, gcc-g++-3.4.3.tar.bz2, newlib-1.12.0.tar.gz, binutils-2.15.tar.bz2, kos-ports-snapshot-20050618.tar.bz2 et kos-snapshot-20050618.tar.bz2 dans ce répertoire.

```
for in `ls -1 *.bz2`; do tar xjf $i; done
tar xzf newlib-1.12.0.tar.gz
```

4.1 Modification du Makefile

Modifiez le *Makefile*, notamment les chemins d'installation des compilateurs croisés : variables sh_prefix et arm_prefix, par exemple :

```
sh_prefix  := /tmp/testKOS/dc-chain-0.1/dc/$(sh_target)
arm_prefix := /tmp/testKOS/dc-chain-0.1/dc/$(arm_target)
```

Renseignez les bonnes versions des logiciels à compiler pour les variables : binutils_ver, gcc_ver, newlib_ver. Enfin, mettez à jour la variable : kos_root (cette variable représente le chemin où se trouvent les répertoires de KallistiOS. En l'occurrence, si vous avez suivi la manipulation ci-dessus, la valeur de cette variable est : /tmp/testKOS/dc-chain-0.1.

ATTENTION

J'ai noté des problèmes de compilation lorsque j'essayais de construire le toolchain en utilisant gcc-4.0 ou supérieur. C'est pour cela que je force l'utilisation de gcc-3.4 et g++-3.4 en assignant les variables CC et CXX dans le shell courant de la manière suivante :

```
export CC=/usr/bin/gcc-3.4
export CXX=/usr/bin/g++-3.4
```

Je vous laisse le soin d'adapter ces manipulations en fonction de la configuration de votre système.

Une fois ces différents réglages effectués, entrez les commandes suivantes :

```
mkdir dc
# (répertoire qui va accueillir
#  les binaires des compilateurs)
make patch
make build-sh4
```

Vous pouvez aller vous chercher un petit café ou un thé. Il m'a fallu environ 15 minutes sur un Athlon 3000).

Pour construire le toolchain ARM, il suffit de changer dans le Makefile, les versions de gcc de 3.4.3 vers 3.0.4 et de binutils de 2.15 à 2.11.2 et de lancer :

```
make arm-build
```

(environ 5 minutes sur un Athlon 3000+).
Il n'y a pas besoin d'appliquer de patch.

Les deux toolchains sont maintenant construits. Nous allons passer à la compilation proprement dite de KOS (ouf !) :

```
cd kos
cp doc/environ.sh.sample ./environ.sh
```

Éditez environ.sh avec votre éditeur de texte préféré :

- Placez dans la variable KOS_BASE le chemin où est décompressé le snapshot de KOS, par exemple : export KOS_BASE="/tmp/testKOS/dc-chain-0.1/kos/"
- Placez dans la variable KOS_CC_BASE le répertoire d'installation du compilateur sh-4, par exemple : export KOS_CC_BASE="/tmp/testKOS/dc-chain-0.1/dc/sh-elf"
- KOS_CC_PREFIX= sh-elf
- DC_ARM_BASE = export DC_ARM_BASE="/tmp/testKOS/dc-chain-0.1/dc/arm-elf"
- DC_ARM_PREFIX = arm-elf

```
./environ.sh
```

Pour inclure toutes ces variables dans le shell courant.

Les variables que nous avons renseignées sont utilisées par KOS non seulement pour la compilation de la bibliothèque, mais aussi à chaque compilation de projets utilisant KOS. Il faudra donc réaliser cette manipulation à chaque compilation ou alors inclure ces variables dans son shell.

4.2 Compiler KOS

Il suffit de taper make et vérifier que tout se passe bien ;)

4.3 Compiler les bibliothèques

Pour être standard, l'idéal est de placer l'arborescence de ports de bibliothèques au même niveau que le répertoire d'installation de KOS. Dans mon exemple, j'ai compilé KOS dans /tmp/testKOS/dc-chain-0.1/kos/, j'ai donc placé les bibliothèques dans /tmp/testKOS/dc-chain-0.1/kos-port.

```
tar xjf kos-ports-snapshot-20050409.tar.bz2
cd kos-ports
ln -s ../kos/addons/Makefile
make
```

Après avoir construit un environnement de développement à destination de la console, nous allons mettre en place les outils de communication PC/Dreamcast.

4.4 Transfert de fichiers vers la Dreamcast

Le moyen le plus facile pour transférer des données entre le PC et la Dreamcast est de se procurer ou de fabriquer ce que l'on appelle un « coder cable ». Des informations sur sa fabrication sont disponibles sur cette page [10].

N'ayant aucune connaissance en électronique, je me le suis acheté. C'est un simple câble série, nous aurons donc besoin

d'outils logiciels de transfert entre le PC et la console, à savoir : dcload/dc-tool. Le site [11] sur lequel se trouvent ces programmes est un site majeur de l'activité autour de notre console bien aimée. On y trouve toutes formes de sources, binaires, documentation ayant un proche rapport avec la console. La première phase est de graver un CD-ROM et de démarrer dessus avec sa Dreamcast. Ce CD-ROM met la console en écoute sur le port série. Cette dernière s'occupe une fois le transfert terminé, d'exécuter le programme. La Dreamcast joue alors le rôle de serveur et le PC le rôle de client. Téléchargeons alors les programmes respectifs.

4.4.1 Côté serveur :

```
wget http://www.boob.co.uk/files/dcload-1.0.3-1st_read.zip
unzip dcload-1.0.3-1st_read.zip
wget http://www.boob.co.uk/files/dcload-1.0.3.tar.gz
unzip dcload-1.0.3.tar.gz
cd dcload-1.0.3/make-cd
```

Ajustez le Makefile. Attention ! Le fichier Makefile est assez ancien. Pour créer les pistes du CD-ROM, une commande basée sur CDRecord est utilisée. Il va falloir mettre à jour ce Makefile, car sur les noyaux de la génération 2.6, l'émulation SCSI n'est plus utilisée. Personnellement, j'ai modifié ainsi : `CDRECORD = cdrecord dev=ATAPI:0,0,0 speed=8` et j'ai ajusté le chemin du fichier `1st_read.bin`. N'hésitez pas à lire la section de l'article concernant la gravure de binaires afin de comprendre ce que nous sommes en train de faire ;).

```
make
```

Les fichiers `.bin` sont dans la première archive précédemment téléchargée. Pour ceux qui n'ont pas envie de faire cette manipulation, je mets à disposition [9] l'image ISO de ce CD-ROM de boot.

4.4.2 Côté client :

```
wget http://www.boob.co.uk/files/dc-tool-serial-1.0.3-linux.gz
gunzip dc-tool-serial-1.0.3-linux.gz
chmod +x dc-tool-serial-1.0.3-linux
```

Pour utiliser le programme côté client : `./dc-tool-linux -x monFichier.elf`. Vous pouvez maintenant transférer des données du PC vers la Dreamcast.

ATTENTION

Il faut que l'utilisateur exécutant le programme `dc-tool-serial-1.0.3-linux` ait des droits d'écriture sur le périphérique `/dev/ttyS0` (périphérique utilisé par défaut). Pour voir les paramètres utilisés par `dc-tool-serial-1.0.3-linux`, il suffit de le lancer sans aucun argument, ce qui fait apparaître ceci :

```
john@Odyssee$ dc-tool-serial-1.0.3-linux
dc-tool 1.0.3 by <andrewk@napalm-x.com>
-x <filename> Upload and execute <filename>
-u <filename> Upload <filename>
-d <filename> Download to <filename>
-a <address>  Set address to <address>
              (default: 0x8c010000)
-s <size>     Set size to <size>
-t <device>   Use <device> to communicate
              with dc (default: /dev/ttyS0)
-b <baudrate> Use <baudrate>
              (default: 57600)
-e            Try alternate 115200
              (must also use -b 115200)
-n            Do not attach console and fileserver
-p            Use dumb terminal rather than
              console/fileserver
-q            Do not clear screen before download
-c <path>     Chroot to <path>
              (must be super-user)
-i <isofile>  Enable cdfs redirection using iso
              image <isofile>
-h            Usage information (you're looking at it)
```

La partie installation de l'environnement est enfin achevée. La première fois, j'en ai eu pour quelques heures, mais quel bonheur lorsque j'ai lancé mon premier programme sur la console !

Nous allons en créer quelques-uns dans la partie suivante. Nous arrivons enfin aux choses passionnantes :).

5. Exemples

5.1 Hello World

Un premier exemple classique, qui ne va pas faire grand-chose. Ce programme va être lu par la console, et utiliser une propriété de KOS pour écrire sur la sortie standard de notre shell.

KOS fournit une galerie assez complète d'exemples en partant de ce "Hello World" vers des choses beaucoup plus intéressantes avec de la 3D, du son...

```
01: #include <kos.h>
02:
03: /*
04: INIT_NONE                  -- Pas d'auto init
05: INIT_IRQ                   -- Activation des IRQs
06: INIT_THD_PREEMPT  -- Activation des threads
                         préemptif
07: INIT_NET                   -- Activation du réseau
08: INIT_MALLOCSTATS  -- Activation d'un appel à
                        malloc_stats() avant la fin de
                          l'exécution du programme
09: Vous pouvez faire des combinaisons de OR entre
    ces variables ou ne rien faire du tout.
10: */
11:
12: KOS_INIT_FLAGS(INIT_DEFAULT | INIT_MALLOCSTATS);
13:
14:
15: /* point d'entree principal du programme */
16: int main(int argc, char **argv)
17: {
18:   printf("Hello world!\n");
19:   return 0;
20: }
```

Le Makefile associé à ce projet va nous servir de base pour tous les autres exemples. C'est exactement le Makefile disponible dans kos/examples/dreamcast/hello.

```
01:# Put the filename of the output binary here
02:TARGET = hello.elf
03:
04:# List all of your C files here,
   # but change the extension to ".o"
05:# Include "romdisk.o" if you want a rom disk.
06:OBJS = hello.o
07:
08:# If you define this, the Makefile.rules will
   # create a romdisk.o for you
09:# from the named dir.
10:KOS_ROMDISK_DIR = romdisk
11:
12:# The rm-elf step is to remove the target
   # before building, to force the
13:# re-creation of the rom disk.
14:all: rm-elf $(TARGET)
15:
16:include $(KOS_BASE)/Makefile.rules
17:
18:clean:
19:        -rm -f $(TARGET) $(OBJS)
20:
21:rm-elf:
22:        -rm -f $(TARGET)
23:
24:$(TARGET): $(OBJS)
25:        kos-cc -o $(TARGET) $(OBJS)
26:
27:run: $(TARGET)
28:        $(KOS_LOADER) $(TARGET)
29:
30:dist:
31:        rm -f $(OBJS)
32:        $(KOS_STRIP) $(TARGET)
33:
```

5.2 Chargement d'une image

Passons tout de suite à l'exemple suivant qui est déjà beaucoup plus intéressant. En effet, cet exemple va charger une image au format png, et va effectuer un fondu en partant d'un écran noir jusqu'à l'apparition de l'image complète.

```
01: #include <kos.h>
02: #include <png/png.h>
03: #include <zlib/zlib.h>
04:
05: /* Structure décrivant une texture dans KOS*/
05: pvr_ptr_t tex;
06:
07: /*
08: Fonction de chargement de l'image au format png
    et transformation en structure KOS
09: Accès à un système de fichiers spécifique :
    Le RomDisk. Ce dernier est linke directement à
l'exécutable.
10  Attention les images doivent être carrées.
11: */
12: void img_init()
13: {
14:   team_tex = pvr_mem_malloc(512*512*2);
15:   png_to_texture("/rd/team.png",
          tex, PNG_FULL_ALPHA);
16: }
17:
18:/*
19: Fonction qui se charge de faire l'affichage
    de l'image sur tout l'écran.
20: Elle prend en paramètre un flottant qui
    représente  l'indice de transparence de l'image.
21:*/
22:void draw_tr_img(float alpha)
23:{
24:    pvr_poly_cxt_t cxt;
25:    pvr_poly_hdr_t hdr;
26:    pvr_vertex_t vert;
27:
28:    /*
29:     PVR_LIST_TR_POLY    : texture transparente,
          utiliser PVR_LIST_OP_POLY si elle est opaque
30:     PVR_TXRFMT_ARGB4444 : couleur de texture transparente,
          utiliser PVR_TXRFMT_RGB565 si elle est opaque
31:     tex                           : notre objet texture
32:     512,512             : taille de l'image
33:     PVR_FILTER_BILINEAR : filtre bilinéaire,
          mettre 0 si on n'en veut pas
34:    */
35:    pvr_poly_cxt_txr(&cxt, PVR_LIST_TR_POLY,
                  PVR_TXRFMT_ARGB4444, 512, 512,
                  tex, PVR_FILTER_BILINEAR);
36:    pvr_poly_compile(&hdr, &cxt);
37:    pvr_prim(&hdr, sizeof(hdr));
38:
39:
40:    /*
41:      L'API pvr de KOS donne une interface pour dessiner
         des points (vertex) sur l'écran.
42:      PVR_PACK_COLOR : définition de la couleur du point.
         On lui passe une valeur alpha en paramètre
           qui est notre valeur de transparence.
43:    */
44:    vert.argb = PVR_PACK_COLOR(alpha, 1.0f, 1.0f, 1.0f);
45:    vert.oargb = 0;
46:
47:    /*
48:      On définit tous les vertex
49:    */
50:    vert.flags = PVR_CMD_VERTEX;
51:
52:    /*
53:      Notre but est d'afficher, à chaque image,
         une image carrée de la taille de l'écran.
           On a donc 4 points à définir ayant pour coordonnées :
        (0,0,0), (0,480,0), (640,0,0), (640,480,0)
54:    */
55:    vert.x = 0;
56:    vert.y = 0;
57:    vert.z = 1;
58:    vert.u = 0.0;
59:    vert.v = 0.0;
60:    pvr_prim(&vert, sizeof(vert));
61:
62:    vert.x = 640;
63:    vert.y = 0;
64:    vert.z = 1;
65:    vert.u = 1.0;
66:    vert.v = 0.0;
67:    pvr_prim(&vert, sizeof(vert));
68:
69:    vert.x = 0;
70:    vert.y = 480;
71:    vert.z = 1;
72:    vert.u = 0.0;
```

```
73:    vert.v = 1.0;
74:    pvr_prim(&vert, sizeof(vert));
75:
76:    vert.x = 640;
77:    vert.y = 480;
78:    vert.z = 1;
79:    vert.u = 1.0;
80:    vert.v = 1.0;
81:
82:    /*
83:      On termine la définition des vertex
84:    */
85:    vert.flags = PVR_CMD_VERTEX_EOL;
86:    pvr_prim(&vert, sizeof(vert));
87:}
88:
89:void draw_intro(void)
90:{
91:  float alpha=0.0;
92:  int i = 0;
93:
94:  for(; i<100; i++)
95:  {
96:      /* On spécifie à PVR de se préparer */
97:      pvr_wait_ready();
98:
99:         /* On spécifie à PVR qu'on commence la
           scène */
100:     pvr_scene_begin();
101:
102:      /* On doit séparer l'affichage des textures
            transparentes des textures opaques. Si on avait sur
               notre scène des textures opaques, on rajouterait
               un bloc : pvr_list_begin(PVR_LIST_OP_POLY) */
103:       pvr_list_begin(PVR_LIST_TR_POLY);
104:      draw_tr_img(alpha);
105:       /* On a terminé ce qu'on voulait faire,
                    on le dit donc &grave; pvr. */
106:     pvr_list_finish();
107:       / *De même qu'on a dit qu'on commençait
                    une scène, on prévient quand
                    on l'a terminée */
108:     pvr_scene_finish();
109:     alpha+=0.01;
110:     thd_sleep(50);
111:  }
112:  thd_sleep(2000);
113:}
114:
115:
116: /* Définition du romdisk */
117:extern uint8 romdisk[];
118:KOS_INIT_ROMDISK(romdisk);
119:
120:int main(int argc, char **argv)
121:{
122:
123:  pvr_init_defaults();
124:
125:
126:  img_init();
127:  draw_intro();
128:  return 0;
129:}
```

Dans cet exemple, nous utilisons un *romdisk*. C'est une archive qui va être compilée et liée au binaire. Pour la créer, il suffit de créer un répertoire `romdisk` dans celui des sources du projet, et de modifier le Makefile de la sorte :

```
OBJS = hello.o romdisk.o

$(TARGET): $(OBJS)
         kos-cc -Wall -o $(TARGET) $(OBJS) -lpng -lm -lz
```

Il ne nous reste plus qu'à l'exécuter de la façon de notre choix.

5.3 Gestion des manettes

Voilà le dernier exemple abordé dans cet article. Nous allons nous attarder sur la gestion des manettes, grâce à un exemple qui illustre la gestion de la croix directionnelle.

```
01:#include <kos.h>
02:
03:
04:int check_pad()
05:{
06:  uint8 c;
07:   /* Structure permettant de représenter un contrôleur */
08:  static cont_cond_t cond;
09:   /* Récupération de la première manette */
09:  c = maple_first_controller();
10:
11:   /* Remplissage de la structure cond, en fonction de c.
12         C'est cette structure qui va contenir
                    les informations importantes.*/
13:  if (cont_get_cond(c, &cond) < 0)
14:    return 0;
15:
16:  /* On a appuyé à gauche de la croix ? */
16:  if (!(cond.buttons & CONT_DPAD_LEFT))
17:  {
18:    printf("Pressed left\n");
19:    return 1;
20:  }
21:
22:  /* On a appuyé à droite de la croix ? */
23:  if (!(cond.buttons & CONT_DPAD_RIGHT))
24:  {
25:    printf("Pressed right\n");
26:    return 1;
27:  }
28:
29:
30:  /* On a appuyé en haut de la croix ? */
30:  if (!(cond.buttons & CONT_DPAD_UP))
31:  {
32:    printf("Pressed up\n");
33:    return 1;
34:  }
35:
36:  /* On a appuyé en bas de la croix ? */
37:  if (!(cond.buttons & CONT_DPAD_DOWN))
38:  {
39:    printf("Pressed down\n");
40:    return 1;
41:  }
42:
43:  return 0;
44:}
45:
46:
47:int main(int argc, char **argv)
48:{
49:  int done = 0;
50:
51:  while(!done)
```

```
52:  {
53:    /*
54:      Macro qui permet d'itérer sur tous les contrôleurs
                    branchés sur la console.
55:      Si un contrôleur a appuyé sur start, le programme s'arrête
56:    */
57:    MAPLE_FOREACH_BEGIN(MAPLE_FUNC_CONTROLLER, cont_state_t, st)
58:      if (st->buttons & CONT_START)
59:        done = 1;
60:    MAPLE_FOREACH_END()
61:
62:    check_pad();
63:    /* On stoppe le thread principal pour 60 milli secondes
64:    thd_sleep(60);
65:  }
66:  return 0;
67:}
```

Pour le Makefile, nous allons réutiliser celui fourni dans le premier exemple. Il suffit amplement.

La présentation de l'interface de développement est terminée. Cette dernière est vraiment très riche. Je pourrais en parler durant de nombreuses pages et j'espère avoir choisi des exemples suffisamment caractéristiques pour vous donner envie de vous plonger dedans. Je vais terminer l'article par un paragraphe sur la gravure de nos binaires. Pour information, n'ayant pas trouvé les bons tutoriaux sur Internet, j'ai mis très longtemps avant de réussir cette étape. Je vais bien détailler la manipulation.

6. Rendre un binaire bootable

6.1 Préparation

Nous allons utiliser un fichier binaire `.elf` que nous allons nommer `hello.elf`. Nous allons le manipuler pour le rendre acceptable par la console, via un CD-ROM.

```
sh-elf-strip hello.elf
sh-elf-objcopy -O binary -R .stack hello.elf hello.bin
```

Notre binaire est alors quasiment prêt à être accueilli par la console. Il ne reste plus qu'une seule étape : le *scrambling*. *Scrambler* un binaire Dreamcast peut s'interpréter par « chiffrer » un binaire. Pour que ce binaire soit lu sur la console à partir d'un CD-ROM, il faut impérativement le *scrambler*. Pour ce faire, il faut télécharger le programme qui se charge de cette opération [12] et le compiler.

```
gcc -o scramble scramble.c
./scramble hello.bin 1ST_READ.BIN
```

Une fois ces manipulations effectuées, nous pouvons passer à l'étape de préparation du CD-ROM.

6.2 Explications Générales

Je vais expliquer brièvement le mécanisme pour rendre un binaire fraîchement compilé bootable sur un CD-ROM. Tout d'abord, la Dreamcast permet de lire des CD-R (pour les RW, c'est une autre paire de manches [13]). Pour qu'un CD-ROM soit bootable sur Dreamcast, ce dernier doit être composé de deux sessions. La première est une session audio normale (une piste de silence convient très bien) et la seconde doit être de type CD/XA (mode 2 form 1). Cette piste doit contenir une archive de type ISO9660 classique, dont les 16 premiers secteurs sont dédiés au *Bootstrap* (`IP.BIN`). Vous pouvez soit créer votre propre `IP.BIN` [14], soit en utiliser un existant, par exemple celui fourni dans l'archive de `dc-loader` téléchargée précédemment.

6.3 Création et gravure de la première piste

```
dd if=/dev/zero bs=2352 count=300 of=audio.raw
cdrecord dev=ATAPI:x,y,z speed=8 -multi -audio audio.raw
```

Faire un `cdrecord dev=ATAPI -scanbus` pour déterminer les valeurs de `x`, `y`, et `z`. Un conseil ! Gravez à une vitesse de 8x, une gravure plus rapide risquant de rendre le CD illisible sur la console. J'ai déjà perdu quelques CD dans cette manipulation ;). A noter que le CD a été gravé en multisession afin qu'il puisse accueillir la piste du programme à lancer.

6.4 Création et gravure de la seconde piste

Comme la première piste a été gravée en multissession, nous avons besoin de savoir quand commence cette piste et quand elle se termine. Pour ce faire :

```
cdrecord dev=ATAPI:x,y,z -msinfo
```

Cette commande retourne deux nombres sous la forme : `s,e`. Pour ma part, les valeurs sont : 0,11702. Nous allons procéder à la création de l'image du binaire :

```
mkisofs -l -r -C s,e -G IP.BIN -o tmp.iso 1ST_READ.BIN
```

puis la graver :

```
cdrecord dev=ATAPI:x,y,z speed=8 -multi -xa tmp.iso
```

ATTENTION

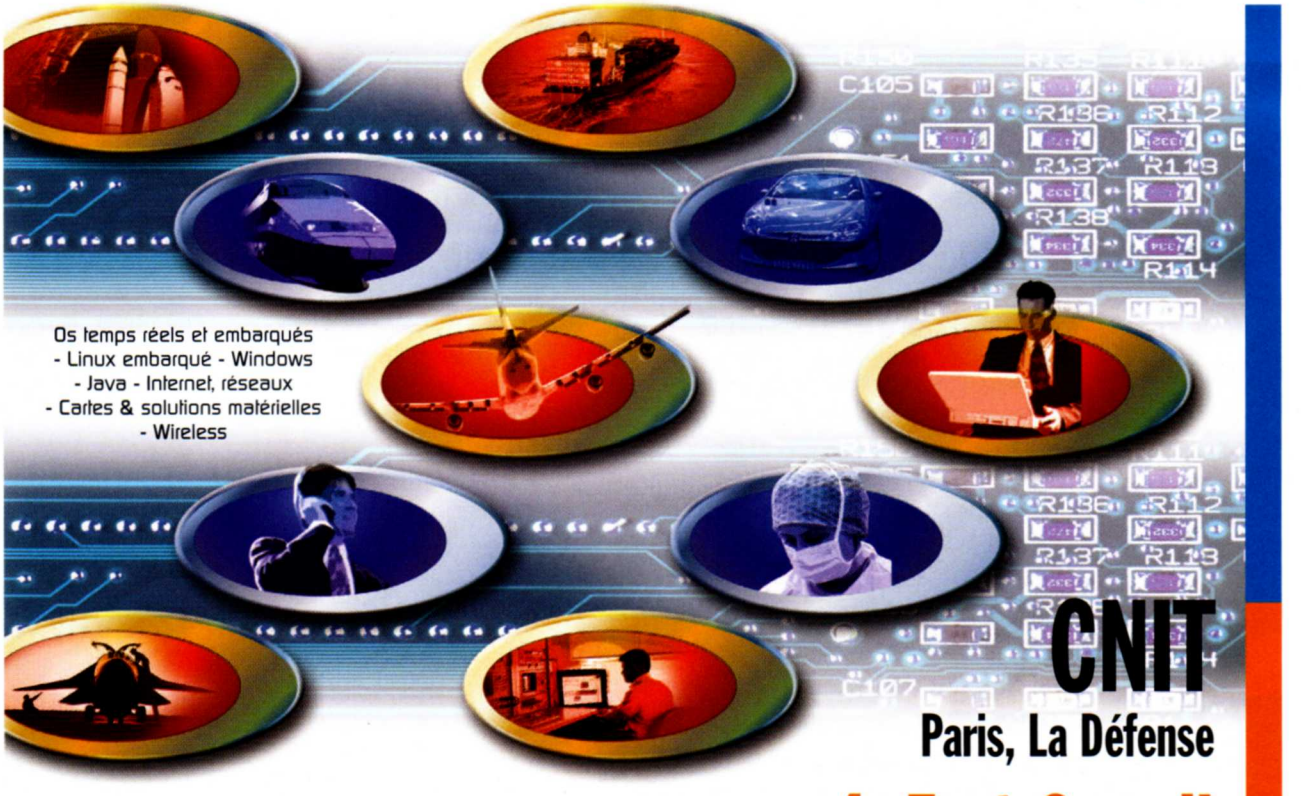
Vous devez vérifier dans le manuel de CDRecord ce que représente l'option `-xa`. En effet, la Dreamcast attend une taille de secteur de 2048 bytes. Historiquement, ainsi qu'on peut le lire dans la majorité des documents sur le sujet, CDRecord était appelé avec l'argument `-xa1`. Voici un extrait de mon manuel de CDRecord :

> *xa If this flag is present, all subsequent tracks are written in CD-ROM XA mode 2 form 1 format. The data size is a multiple of 2048 bytes. The XA sector sub headers will be created by the drive. With this option, the write mode is the same as with the -multi option.*

RTS 14e édition

EMBEDDED SYSTEMS 2006

Le salon des solutions informatiques temps réel et des systèmes embarqués



Au cœur de l'embarqué

Le monde des Systèmes Temps Réel évolue rapidement. Les applications se multiplient, utilisent des solutions plus matures et davantage dédiées aux spécificités de chaque secteur. Des innovations se font jour. Depuis 14 ans, RTS Embedded Systems rend compte de ces évolutions.

L'édition 2006 sera comme chaque année la vitrine de cette offre matérielle et logicielle, le rendez-vous de la profession rassemblée, et la caution des meilleurs experts dans la présentation d'un cycle de conférences pointues.

RTS 2006 proposera aux visiteurs son lot de nouveautés en terme de composants, de cartes, de systèmes d'exploitation, de logiciels enfouis...

Sans oublier, bien sûr, toutes les technologies émergeantes, à l'image des nouvelles techniques de véri-fication et de débogage des logiciels, de l'alternative des nouveaux sous-systèmes multiprocesseurs et de l'arrivée en force des FPGA dans les systèmes embarqués.

Pour exposer, visiter l'exposition ou s'inscrire aux conférences : www.birp.com/rts

Salon strictement réservé aux professionnels.

Organisation BIRP 97, rue du Cherche-Midi 75006 Paris - France - Tel : +33 (0)1 44 78 99 30 - Fax : +33 (0)1 44 78 99 49 - e-mail : rts@birp.fr

> *xa1 If this flag is present, all subsequent tracks are written in CD-ROM XA mode 2 form 1 format. The data size is a multiple of 2056 bytes. The XA sector sub headers are part of the user data and have to be supplied by the application that prepares the data to be written.*

Sur ma distribution, d'après ce manuel, je dois utiliser l'option `-xa` *et non* `-xa1` *pour graver le CD avec une taille de secteur de 2048 bytes. Il se peut que sur votre distribution, ce soit différent, alors faîtes bien attention.*

Voilà, nous avons gravé notre programme sur un CD. Il suffit maintenant de l'insérer dans la console et de procéder au test ;).

Un dernier mot en ce qui concerne la gravure de binaires : il existe des logiciels [15] qui permettent de graver plusieurs binaires en multisession sur le même CD, ce qui a pour effet un gain non négligeable de CD. Ces logiciels ne sont à disposition que sur plate-forme Windows. Je n'ai jamais eu l'occasion de tester.

Conclusion

Ainsi se termine l'article de présentation de l'interface de développement sur Dreamcast. J'espère que vous prendrez plaisir à porter vos jeux sur cette console que j'apprécie énormément. Je tiens à remercier d'une part les développeurs de KOS, qui ont fait un travail magnifique et d'autre part toute la communauté francophone du Homebrew Dreamcast.

Jonathan Muller,

jonathan.muller@yahoo.fr,

LIENS

- [1] Feet of fury : **http://www.feetoffury.com/**
  C'est un jeu commercial développé par l'équipe de Dan Potter, basé sur KOS.
- [2] Alice Dreams : **http://www.alicedreams.com**
  C'est un jeu encore en développement, par 2 français : Poche et Patbier.
- [3] Article sur la vie de la console :
  http://www.jeuxvideo.com/articles/0000/00003899_dossier.htm
- [4] KOS : **http://gamedev.allusion.net**
  Site officiel de KallistiOS, avec notamment l'historique, la motivation, les sources, les tutoriels pour configurer la bibliothèque...
- [5] Snapshots de KOS :
  http://gamedev.allusion.net/softprj/kos/howtogetit.php
- [6] Chankast : **http://www.chanka.org/**
- [7] Wine : **http://www.winehq.org/**
- [8] Liste de sites pour les outils de base du compilateur croisé :
  ftp://ftp.lip6.fr/pub/gcc/releases/gcc-3.4.3
  ftp://ftp.lip6.fr/pub/gcc/releases/gcc-3.0.4
  ftp://sources.redhat.com/pub/newlib/index.html
  http://ftp.gnu.org/gnu/binutils/
- [9] Scripts d'automatisation de la construction du compilateur croisé :
  http://muller.john.free.fr/dreamcast
- [10] Page d'explication sur la fabrication du coder cable :
  http://www.waz.org.uk/gamedev/dc/serialcable.php
- [11] Boob : **http://boob.co.uk/**
  Site de référence Dreamcast. On y trouve énormément de liens, binaires, documentation.
- [12] sramble.c : **http://boob.co.uk/files/scramble.c**
- [13] CD-RW sur Dreamcast :
  http://semicolo.free.fr/Dreamcast/cdrw/indexfr.html
- [14] Construire son propre IP.BIN : **http://mc.pp.se/dc/ip.bin.html**
- [15] Demomenu : **http://dcreload.free.fr/tuto_demomenu.php**
- Forum francophone principal sur le développement Dreamcast
  http://dcreload.free.fr/(lien forum)
- Portail francophone sur la Dreamcast : **http://www.dc-france.com/index.php**
- Forum anglophone très actif : **http://www.dcemulation.com/phpBB/index.php**

www.sstic.org

31 mai/1-2 juin 2006 Rennes

Les limites de la sécurité

SSTIC

SYMPOSIUM
SUR LA SÉCURITÉ
DES TECHNOLOGIES
DE L'INFORMATION
ET DES COMMUNICATIONS

misc · cea · Supélec · EADS CCR · ESA · Telindus Arche · france telecom R&D