ULTRIX-32
Guide to Ethernet
Communications Servers

Order No. AA-ME98A-TE

ULTRIX-32 Operating System, Version 3.0

Digital Equipment Corporation

This manual was written and produced by the ULTRIX Documentation Group in Nashua, New Hampshire.

Contents

About This Manual

1 Ethernet Communications Servers

2 Terminal Servers

6 Creating Your Own Service

A Server Maintenance Functions

Example

Figure

About This Manual

This guide describes administrative tasks and procedures for setting up and maintaining the interfaces between the ULTRIX-32 operating system and communications servers in a Ethernet Local Area Network (LAN).

Audience

This guide is meant for the person whose function includes maintaining networks on an ULTRIX-32 operating system.

To use this guide, you must know how to use ULTRIX-32 commands, understand system configurations, know system naming conventions, and be able to use an editor such as vi or ed. In addition, you need to know the names and network addresses of the other systems on the LAN.

Organization

This manual consists of six chapters, three appendixes, and an index.

Chapter 1 Ethernet Communication Servers
Introduces communications servers, terminal servers, and the user information you need to accomplish the tasks in this manual.

Chapter 2 Terminal Servers
Describes how to set up a terminal server to work with ULTRIX-32 systems.

Chapter 3 Printers on a Terminal Server
Describes how to set up a printer on a terminal server.

Chapter 4 Host-Initiated Connections
Describes how to set up host-initiated connections to terminal servers.

Chapter 5 LAT/Telnet Gateway
Describes how to set up the LAT/telnet gateway service.

Chapter 6 Creating Your Own Service
Describes how to set up your own service.

Appendix A	Ethernet Remote Node Maintenance Functions Describes the procedures and commands you use to maintain remote ULTRIX-32 service nodes.
Appendix B	Host-Initiated Connection Example Contains a sample C-language program that employs host-initiated connection capabilities.
Appendix C	Program to Replace getty for Special Services Contains a sample C-language program that replaces the /etc/getty program for any device you define as a lat/dlogin gateway.

Related Documents

You should have available the hardware documentation for your system and the other documents in the current ULTRIX-32 documentation set.

In addition, you should have the terminal server installation guide for your terminal server.

Note

For an introduction to Ethernet communications servers and other DIGITAL networks and communications products, obtain the current version of the *Networks and Communications Buyer's Guide* from your local DIGITAL sales representative.

Conventions

The following conventions are used in this manual:

special	In text, each mention of a specific command, option, partition, pathname, directory, or file is presented in this type.
command(x)	In text, cross-references to the command documentation include the section number in the reference manual where the commands are documented. The reference cat(1), for example, indicates that you can find the material on the cat command in Section 1 of the reference pages.
literal	In syntax descriptions, this type indicates terms that are constant and must be typed just as they are presented.
italics	In syntax descriptions, this type indicates terms that are variable.

[ ]	In syntax descriptions, square brackets indicate terms that are optional.
. . .	In syntax descriptions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
UPPERCASE	The ULTRIX-32 system differentiates between lowercase and uppercase characters. Enter uppercase characters only where specifically indicated by an example or a syntax line.
example	In examples, computer output text is printed in this type.
example	In examples, user input is printed in this bold type.
%	This is the default user prompt in multiuser mode.
#	This is the default superuser prompt. In some examples of file entries, the pound sign is a special character indicating a comment.
>>>	This is the console subsystem prompt.
Local>	This is the LAT server prompt.
.	In examples, a vertical ellipsis indicates that not all of the
.	lines of the example are shown.
.	
<KEYNAME>	In examples, a word or abbreviation in angle brackets indicates that you must press the named key on the terminal keyboard.
<CTRL/x>	In examples, symbols like this indicate that you must hold down the CTRL key while you type the key that follows the slash. Use of this combination of keys may appear on your terminal screen as the letter preceded by the circumflex character. In some instances, it may not appear at all.

Ethernet Communications Servers 1

Ethernet communications servers are dedicated, special purpose subsystems that allow resource sharing across many host systems within a local area network (LAN).

1.1 Types of Communications Servers

There are five types of communications servers:

- Terminal Server

  Connects terminals and other bit-serial, asynchronous devices to service nodes in an Ethernet LAN.

- DECnet Router

  Transfers data packets between DECnet nodes on an Ethernet LAN and remote DECnet nodes or other Ethernet LANs.

- DECrouter 2000

  Transfers data packets between DECnet nodes on an Ethernet LAN and remote DECnet nodes or other Ethernet LANs via asynchronous lines.

- DECnet Router/X.25 Gateway

  Connects DECnet/Ethernet LANs to X.25 packet-switched data networks and to remote DECnet systems.

- DECnet/SNA Gateway

  Connects Ethernet LANs to IBM hosts in an SNA network.

This guide focuses on terminal servers. For a definition of the types of communications servers supported by the ULTRIX-32 operating system, refer to the *ULTRIX-32 Software Operating System Software Product Description.*

1.2 Terminal Servers

Each terminal connected to a terminal server can access services running on service nodes that are connected to the same Ethernet. Terminal servers connect asynchronous terminals at speeds up to 19,200 bits per

second to local Ethernet hosts. Terminal servers do not have mass storage. Thus, server software, including diagnostics, is downline-loaded into terminal servers from a load host. In the event of a server-detected hardware or software malfunction, the server unit attempts to upline-dump its memory image (for later analysis), and automatically initiates a reload of its software.

1.3 Terminal Server Information

For the technical information you need in order to understand and use the material in this document, refer to the documentation shipped with your Ethernet terminal server, with your local area network, and with your ULTRIX-32 operating system. This documentation includes:

- Terminal server documentation

  You need a terminal server hardware installation guide. You also need software documentation about managing the terminal server software, the terminal image software file, and dsvconfig, a script for setting up a terminal server. In addition, you need to know how to use these commands: connect, logout, and show.

- ULTRIX-32 operating system guides

  You need to refer to the procedures in the Guide to System Configuration File Maintenance, the Guide to System Environment Setup, and the Guide to Networking.

- ULTRIX-32 Reference Pages, Section 1

  You need to know how to use these commands: chown, kill, lpc, lpr, mkdir, and ps.

- ULTRIX-32 Reference Pages, Sections 4 and 5

  You need to know about maintaining the LAT and devices using lta(4) and ttys(5).

- ULTRIX-32 Reference Pages, Section 8

  You need to know how to use these commands: addnode, arp, ccr, doconfig, getnode, ifconfig, lcp, load, MAKEDEV, mop_mom, netsetup, remnode, shutdown, and trigger.

- Processor Documentation

  Finally, you need to know how to use the processor console commands E (for examine) and T (for test), as defined in your VAX processor documentation.

Terminal Servers 2

This chapter describes how to set up ULTRIX-32 systems as load hosts or service nodes for a terminal server.

However, before you set up the ULTRIX-32 systems, you must attach the terminal server hardware to the LAN and test that the connection works. This hardware installation is described in your terminal server documentation.

Then, you tailor each system you have selected to be either a load host for the terminal server or a service node that is available to users of devices which are connected to the terminal server.

2.1 Tailoring An ULTRIX-32 Load Host

A load host is a Phase IV DECnet or an Internet host on the LAN. You use a load host (or load hosts, if you specify more than one) to downline load the terminal server software, define a place for upline dumps to be sent from the terminal server, and to maintain the software associated with the terminal server.

An ULTRIX-32 system must be running with either the DECnet/ULTRIX or the Internet software installed, configured, and enabled before it can function as a load host.

The procedures to tailor a load host are:

- Edit the config file, if necessary
- Edit the rc.local file
- Install the terminal server load image
- Reboot the ULTRIX-32 system, if necessary
- Downline-load the terminal server load image

2.1.1 Edit the Configuration File

If your system is to be a load host, edit the config file and add the following entries, if they are not already there:

```
options          DLI
pseudo-device    dli
```

Note

When you do add entries to the config file, rebuild the kernel to make the additions take effect. You can rebuild the kernel manually or by using the doconfig command.

2.1.2 Edit the rc.local File

On a load host, you edit the /etc/rc.local file because you want the system, when it is rebooted, to start:

- The network interface to the load host
- The mop_mom daemon

2.1.2.1 Start the Network Interface to the Load Host – To interface the hardware to the network, you must edit an entry into the /etc/rc.local file for the specific hardware interface that you have. In addition, you must have this entry take effect before executing the mop_mom daemon.

The following examples show representative interface entries.

For systems supporting a UNIBUS (DEUNA, DELUA):

```
/etc/ifconfig de0 `/bin/hostname`
```

For systems supporting a Q-BUS (DEQNA,DELQA):

```
/etc/ifconfig qe0 `/bin/hostname`
```

For systems supporting the busless, small VAX processors (DESVA):

```
/etc/ifconfig se0 `/bin/hostname`
```

For systems supporting a BI (DEBNT,DEBNA):

```
/etc/ifconfig ni0 `/bin/hostname`
```

2.1.2.2 Start the mop_mom Daemon – Next, you must add the following commands to the /etc/rc.local file after the entry which starts the network interface (if the commands are not already there):

```
if [ -f /etc/mop_mom ]; then
    /etc/mop_mom >& /dev/console
fi
```

When a target system like a terminal server uses mop_mom to load software, the process that mop_mom forks, mop_dumpload, does not always search the nodes data base.

For example, if mop_dumpload can get the name of the load image from the load message, it will load the file directly, without checking the nodes data base. A side effect of this direct load is that servers will be loaded even if they are not registered with the load host. However, you can force mop_dumpload to search the nodes data base by setting the environmental variable LOADUMP_SECURE to on. Then only systems which you have entered in the nodes data base will be loaded.

You can set the environmental variable automatically each time you bring your system to multiuser mode by placing the following entry in the /etc/rc.local file instead of the mop_mom entry described previously in this section.

```
if [ -f /etc/mop_mom ]; then
   LOADUMP_SECURE=on /etc/mop_mom  >& /dev/console
fi
```

2.1.3 Install the Terminal Server Load Image

Next, you must have the terminal server load image software (pr0801eng.sys for a DECserver 200) installed. This software is not included in the ULTRIX-32 distribution and must be ordered separately. For further information, see the terminal server manual.

The terminal server installation procedure places the load image software (for example, the pr0801eng.sys file in the /usr/lib/dnet directory). If this file is not in the proper directory, the DECserver cannot be downline loaded. Install the server software according to the installation instructions.

Note

DECnet-ULTRIX does not need to be running or installed to load the system. The mop_mom program handles the downline loading.

2.1.4 Reboot the System

If you rebuilt your kernel (refer to Section 2.1.1), you need to move the newly-configured kernel to the root(/) directory and reboot your system.

To reboot the system, use the shutdown command. The following example shows how to shut down and reboot the system in one command:

```
# /etc/shutdown -r "System going down for a quick reboot"
```

If you did not rebuild the kernel, you can cause the ULTRIX-32 system to begin functioning as a load host without rebooting by manually executing the commands you have added to the /etc/rc.local file.

2.1.5 Downline-Load the Terminal Server

There are three methods for down-line loading the terminal server. The first is to power on the terminal server, the second is to use the load command, and the third is to use the trigger command. The trigger command is described in Appendix A.

2.1.5.1 Power On the Terminal Server – To down-line load the terminal server, power it on. When the terminal server is powered on, it broadcasts over the Ethernet that it is ready to be loaded. Any load host running mop_mom and with the environmental variable LOADUMP_SECURE set to off (the default), can send the system image software needed to the terminal server, for example, pr0801eng.sys for a DECserver 200. If the LOADUMP_SECURE variable is set to on for mop_mom on a system that can do a down-line load, the ULTRIX-32 system checks its nodes data base for the terminal server requesting the load. If an entry for the terminal server is in the nodes data base, the system volunteers to down-line load the software to the terminal server.

If an entry for the terminal server does not exist in the nodes data base, then that ULTRIX-32 system cannot down-line load to the terminal server. In this case, the DECserver must get the load image software down-line loaded from another system on the Ethernet.

Note

> Because the terminal server installation placed a copy of the load image software in the /usr/lib/dnet directory, your system could down-line load the software to the terminal server. Your system is like any other host on the Ethernet.

In most instances, the first system to volunteer to send the software down-line loads the image to the terminal server. The Ethernet address of the system performing the down-line load appears on the terminal server

console during the process.

Figure 2-1 shows the relationship between the terminal server and the ULTRIX-32 systems connected by an Ethernet. In Figure 2-1, system and systema can down-line load the load image software because the LOADUMP_SECURE variable is set to off. Notice that systemb cannot down-line load the load image software because the LOADUMP_SECURE variable is set to on and the terminal server DS2MYS is not entered into the nodes database on Bsystem. Csystem can down-line load the image software when the LOADUMP_SECURE variable is set to on because DS2MYS has been entered in Csystem's nodes database.

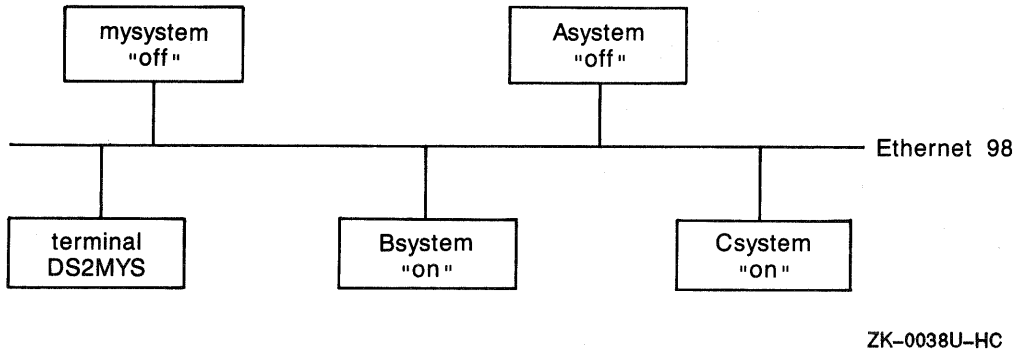

Figure 2-1: A Local Area Network with terminal server

Note

Be sure the system that is down-line loading the image software to your terminal server has the current version of the LAT software. To find the server characteristics, type a command similar to the following from the terminal server console:

```
Local> show server status
```

This command should show the version of LAT software loaded.

2.1.5.2 Determine If the Power On Succeeded – To determine if the down-line load was successful, look in the syslog file in the /usr/spool/mqueue directory on the system that down-line loaded the software (the host system). You should see an entry similar to the following, if the load succeeded:

```
Mar 11 09:02:14 localhost: 177 mop_dumpload:  sending volunteer
assistance for system load, (target node Ethernet address =
08-00-2B-03-C5-90)

Mar 11 09:02:16 localhost: 177 mop_dumpload:  sending system
image, (target node Ethernet address = 08-00-2B-03-C5-90)
```

If the down-line load failed because the load image software (pr0801eng.sys) was not installed in the /usr/lib/dnet directory, you could see an entry like this:

```
Mar 14 11:21:20 localhost:  302 mop_dumpload: load file
PR0801ENG not found, (target node Ethernet address =
08-00-2B-03-C5-90)
```

If the host system has the environmental variable LOADUMP_SECURE set to on for mop_mom, the syslog file should have entries for only those terminal servers (target nodes) that have entries in the nodes data base on that system.

2.1.5.3 Use the load Command – Some terminal servers allow you to down-line load using the load command, some do not. For example, the DECserver 100 does not support the load command.

To use the load command, you need to enter some terminal server information in your system's nodes data base. To do this, use the dsvconfig automatic setup included with the terminal server software. The dsvconfig script is not included with the ULTRIX-32 distribution, but comes with the terminal server's load image software, for example, pro801eng.sys. The terminal server's software installation guide tells you how to enter the nodes you want to down-line load using dsvconfig.

Note

If you can, use the dsvconfig script to enter the nodes instead of manually using the addnode command. Otherwise, you must issue the addnode command twice for a DECnet load host: once with the -P option and once without the -P option. The first command updates the permanent data base, the second updates the volatile database.

Once dsvconfig is installed, you need to run:

```
/usr/lib/dnet/dsvconfig
```

Refer to the terminal server installation guide for more information.

Information can be entered in the nodes database manually by using the addnode command. The format would be similar to:

addnode *system* -h *address* -l pr0801eng -c *circuit_ID*

The system is the name of the terminal server. The address is your terminal server's physical Ethernet address. The circuit_ID is the service circuit ID and is as follows, where *n* is the unit number:

qna-*n* For QBus based processors, such as the MicroVAX II, VAXstation II, MicroVAX 3000 family

bnt-*n* For BI based processors, such as the VAX 8200, 8300, 8500, 8550, 8700 and 8800

una-*n* For VAX processors with an Ethernet interface on a UNIBUS adapter (such as a VAX-11/750, VAX-11/780 and VAX-11/785)

sva-*n* For small, busless VAX processors, such as the VAXstation 2000

After you have added the terminal server information to your system's nodes data base by using dsvconfig or addnode, load the image software. For example, if the name of the terminal server is harvey, type:

```
# /etc/load harvey
```

If a hexadecimal password is needed, supply it here. See Appendix A for more information about the password.

2.1.5.4 Determine If load Succeeded – To see if the down-line load was successful, look in the /usr/spool/mqueue/syslog file on your system for an entry similar to this:

```
Mar 11 09:02:16 localhost: 291 mop_dumpload:  sending system
image, (target node Ethernet address = 08-00-2B-05-40-B7)
```

If the down-line load failed because the system image software was not installed in the /usr/lib/dnet directory, you could see an entry like:

```
Mar 14 11:21:20 localhost:  302 mop_dumpload: load file
PR0801ENG not found, (target node Ethernet address =
08-00-2B-03-C5-90)
```

2.2 Tailoring An ULTRIX-32 Service Node

A service node is any ULTRIX-32 system on the LAN to which you want users to connect through terminal servers using the LAT protocol. A load host can also be a service node if you tailor it to be one.

The procedures to tailor a service node are:

- Edit the config file, if necessary
- Make the lta devices
- Edit the ttys file
- Edit the rc.local file
- Reboot the ULTRIX-32 system, if necessary

2.2.1 Edit the Configuration File

If your system is going to be a service node, edit the config file and add the following entries if they are not already there:

```
options          LAT
pseudo-device    lat
pseudo-device    lta
```

The pseudo-device lta specification creates a default of 16 LAT lines. If you require more, then specify the number required. For example, if you need 32 LAT lines for use with four DECserver 200's, use this entry instead:

```
pseudo-device     lta 32
```

The number you specify must be a multiple of 16, for example, 16, 32, 48, or 256.

Note

When you do add entries to the config file, rebuild the kernel to make the additions take effect. You can rebuild the kernel manually or by using the doconfig command.

2.2.2 Make the lta Devices

To make the appropriate LAT devices, first change directory to /dev, then use the MAKEDEV command. For example, to create 16 LAT devices, type:

```
# cd /dev
# MAKEDEV lta0
```

The lta0 option creates 16 devices numbered sequentially. The device numbers will be tty00 through tty15, if there are no previously-built terminal special files.

If there were previously-built terminal special files (from whatever source), the lta0 command would create devices numbered sequentially after the

highest-numbered special file. For example, if there were already tty00 through tty07 terminal special files, the lta0 command would create 16 files, tty08 through tty23.

To create an additional 16 LAT devices, issue the MAKEDEV command again, using the lta1 option to specify the next set of 16 devices, for example:

```
# cd /dev
# MAKEDEV lta1
```

The lta1 option creates 16 additional devices numbered tty16 through tty31, if there are no previously built terminal special files.

You can see which devices in /dev are the LAT devices (special files) by typing:

```
# file /dev/tty* | grep LAT
```

All tty special files with a major number of 39 are for the LAT. A typical LAT special file looks like:

```
crw--w--w- 1 root      39, 4 MAR 10 11:29 tty12
```

2.2.3 Edit the ttys File

After you have created the new LAT special files, you need to add entries for those devices to the /etc/ttys file. To do this, type:

```
# cd /etc
# vi ttys
```

If you have created 32 new LAT devices for terminals and the special files created were tty00 through tty31, add 32 entries to the /etc/ttys file in this format:

```
tty00    "/etc/getty std.9600"    vt100    on nomodem    #LAT
                    .
                    .
                    .
tty31    "/etc/getty std.9600"    vt100    on nomodem    #LAT
```

To make the new entries in the /etc/ttys file take effect, reboot the system or type the following:

```
# kill -HUP 1
```

2.2.4 Edit the rc.local File

You need to place LAT entries in the /etc/rc.local file. These entries start up the network interface to the service node and the LAT on the service node.

2.2.4.1 Start the Network Interface to the Service Node – To interface the hardware to the network, you must edit an entry into the /etc/rc.local file for the specific hardware interface that you have. In addition, you must have this entry take effect before executing the lcp command.

Representative entries are shown in Section 2.1.2.1.

2.2.4.2 Start Up the LAT on the Service Node – To have LAT started automatically each time the system is brought to multiuser mode, edit the following entry into the /etc/rc.local file after the local daemons entries, and after the DECnet ncp entry if DECnet is installed (DECnet changes the controller Ethernet physical address):

```
if [ -f /etc/lcp ]; then
        /etc/lcp -s  & echo -n ' lat' >/dev/console
fi
```

Additional switches might be needed for the command. For example, if you have set up group access codes for the terminal server, you must include a -g switch in the lcp command.

Note

If you are planning on setting up printers, reverse LAT devices, or gateways on your LAT, the lcp entry will be different from this entry for terminals. See the other chapters in this manual for further information.

2.2.5 Reboot the System

If you rebuilt your kernel (refer to Section 2.2.1), you need to move the newly-configured kernel to the root(/) directory and reboot your system.

To reboot the system, use the shutdown command. The following example shows how to shut down and reboot the system in one command:

```
# /etc/shutdown -r "System going down for a quick reboot"
```

If you did not rebuild the kernel, you can cause the system to begin functioning as a service note without rebooting by manually executing the commands you added to the rc.local file.

Printers on a Terminal Server 3

This chapter describes how to set up a printer on a terminal server.

Prerequisite hardware setup consists of installing a printer on a serial interface on a terminal server.

The tasks discussed in this chapter are:

- Matching printer and server hardware settings
- Testing the port configuration
- Specifying server and port names
- Setting Up the spool directories
- Testing the printer

3.1 Matching Printer and Server Hardware Settings

The first task is to match the hardware settings of the printer and the terminal server. First, you need to determine your printer's characteristics. Specifically, you need to know your printer's character size, flow control. parity, and speed. Refer to your printer documentation for the information you need.

After you have determined your printer's characteristics, compare them to the terminal server's port settings. Be sure the settings correspond. You can see the settings on the terminal server console by using commands similar to the following:

```
Local> show port 7 characteristics
```

This shows the characteristics for port 7. At a minimum, the terminal server should have settings for the port similar to:

Character Size: *printer's character size*
Flow Control: XON (or –CTS/RTS, for some printers)
Speed: *Printer's Speed*
Access: Remote
Alternate Speed: None
Dedicated: None
Autobaud: Disabled
Autoconnect: Disabled

To permanently define a terminal server's port settings, using the define command type:

```
Local> DEFINE PORT 7 SPEED 9600 ALTERNATE SPEED NONE
```

After all the settings for the port have been defined, log out of that port. This initializes the new settings. For example:

```
Local> LOGOUT PORT 7
```

3.2 Testing the Port Configuration

The next task is to test the port configuration to verify that the printer characteristics match in the printer and in the terminal server port.

First, connect the printer to the terminal server. Then you can verify the configuration of the port by using the TEST PORT command on the terminal server. For example, if the configuration is correct, the following command running on a DECserver 200 causes a test pattern of characters to print on a printer attached to port 7:

```
Local> TEST PORT 7
```

The printer prints the test data until the BREAK key is pressed at the terminal server console. If data does not print or if it appears to be incorrect, then the port or the printer is incorrectly set, or there is a hardware problem.

3.3 Defining Names for the Server and Port

Next, you specify the name of the server and the name of the port for the printer to your ULTRIX-32 operating system. There are two ways to do so: through an entry in the /etc/printcap file, or through an lcp command. These ways of naming the server and the port occur when you select a LAT terminal line and set up the /etc/printcap file.

3.3.1 Select a LAT Terminal Line

The next task is to select a LAT terminal line to use with your printer or printers. (The major number for the device special file is 39.)

To prevent anyone from logging into the system by way of the printer's LAT terminal line, use the lcp command to turn off all but host-initiated connections for the device. After you have selected a line, place an lcp entry for it in the /etc/rc.local file. Be sure the new entry appears after the local daemons section. The following entries illustrate the two alternative forms of the lcp command:

```
if [ -f /etc/lcp ]; then
      /etc/lcp -s -h /dev/tty42:T1301A:PORT_6 >/dev/console
fi


if [ -f /etc/lcp ]; then
     /etc/lcp -s -h /dev/tty42 >/dev/console
fi
```

Of course, the lcp command in this file is executed the next time the system is booted (or brought to multiuser mode).

If you want to have the lcp command take effect immediately, type one of the following:

```
# /etc/lcp -s -h /dev/tty42:T1301A:PORT_6

# /etc/lcp -s -h /dev/tty42
```

The first example uses the lcp command to associate tty42 with port 6 on the terminal server T1301A. The second example merely reserves the tty for host-initiated connections; the correlation between the tty and the terminal server port must be made in the printcap file.

Note

You can establish the correlation with either the lcp command or the printcap file, but not both. You must use one method or the other.

Next, you must turn off all LAT printer terminal lines listed in the /etc/ttys file. To do this, first edit the /etc/ttys file. For example, if the printer terminal line is tty42, this entry should appear in the /etc/ttys file:

```
tty42 "/etc/getty T9600" vt100  off nomodem   # LAT connect tty
```

Then, have the modifications to the /etc/ttys file take effect by typing the command:

```
# kill -HUP 1
```

3.3.2 Set Up the /etc/printcap File

The next task is to place a proper entry for your printer in the /etc/printcap file. In the entry, the following parameters need to be defined for the LAT printer:

lp=/dev/tty*nn* The LAT terminal line used to send data to the printer, for example:

```
/dev/tty42
```

ts=*name* The name of the terminal server connected to your printer. To find the terminal server name, type the following at the terminal server console:

```
Local> show server characteristics
```

The entry in the Name: field is the one needed in the /etc/printcap file, for example, LAT_08002B0540B7.

op=PORT_*n* The name of the terminal server port connected to your printer. For example, to find the port name where 7 is the port number, type the following at the DECserver 200 console:

```
Local> show port 7
```

The entry in the Name: field is the one needed in the /etc/printcap file, for example, PORT_7.

os=PRINTME The service name supported on some terminal servers.

Following are four sample /etc/printcap entries for terminal server ports. Please note that each entry contains ts= and an op= lines to specify the server name and the port name. An alternative way to specify these names would have been in the lcp command, as described in Section 3.3.1. However, in that case, the entries would be the same except that the op and os entries would not be used.

```
laser|portrait printing on ln03:\
     :lp=/dev/tty04:\
     :sd=/usr/spool/laser:\
     :ts=T1301A:\
     :op=PORT_6:\
     :br#300:\
     :fc#0177777:fs#023:\
     :if=/usr/lib/ln03of:\
     :pw#80:\
     :pl#66:\
     :mc#20:\
     :vf=/usr/lib/ln03vf:

lp|lp0|local line printer:\
     :sh:\
     :fs#023:\
     :fc#0177777:\
     :br#4800:\
     :ts=T1301A:\
     :lp=/dev/tty05:\
     :pw#80:\
     :op=PORT_5:\
     :sd=/usr/spool/lp:


lp11|la50|line printer on LAT:\
     :fc#0177777:\
     :fs#023:\
     :lp=/dev/tty11:\
     :op=PORT_5:\
     :ts=T1301A:\
     :of=/usr/lib/lpf:\
     :sd=/usr/spool/la50:

lp3|ln03|laser printer on LAT: :lp=/dev/tty00:\
     :sd=/usr/spool/lpd:\
     :ts=LAT_08002B0540B7:\
     :op=PORT_7:\
     :br#19200:\
     :fc#0177777:fs#023\
     :xc#0177777:xs#040\
     :of=/usr/lib/lpdfilters/ln03of:\
     :if=/usr/lib/lpdfilters/ln03of:\
     :lf=/usr/adm/lpd-errs:
```

Read the instructions on remote printer setup contained in the documentation for your DIGITAL terminal server configuration. Then carefully execute the instructions.

3.4 Setting Up the Spool Directories

The next task is to set up the printer spool directories. The printer spool directories correspond directly to the entries in the /etc/printcap file for the sd option. For example, if you have the entry, sd=/usr/spool/lp3, then type the following to create the appropriate spool directory:

```
# cd /usr/spool
# mkdir lp3
# chown daemon lp3
```

3.5 Testing the Printer

After the printer is set up, you should try printing a file to be sure everything works properly. For example, if the printer name is lp3 and test is a text file, type:

```
# lpr -Plp3 test
```

If the printer does not work, check to make sure all the settings are correct. If the printcap entry has an lf entry defined, you can check the corresponding file for information on errors that could have occurred.

Host-Initiated Connections 4

This chapter describes how you set up an ULTRIX-32 system for host-initiated connections to any bit-serial, asynchronous device functioning off a terminal server. Examples of such devices are terminals, modems, communications ports on other host computer systems, and printers. The printer connections discussed in Chapter 3 of this manual are one instance of a host-initiated connection. Sometimes, host-initiated connections are called reverse LAT connections.

This feature allows the manager of an ULTRIX-32 system to associate a named port on a named terminal server with a specific tty device special file. As a result, users can code applications that connect to the port through the LAT. The type of device the target represents is transparent to the LAT protocol.

4.1 Setting Up The ULTRIX-32 System

To define the connection between the host tty and the terminal server port service, you run the LAT control program, lcp, using the -h option. In the command, you specify the tty, the lat server name and the lat port number, in that order. For example:

```
lcp -h /dev/tty42:T1301A:PORT_6
```

The protection bits, the owner, and the group of the tty should be set appropriately for the intended use of the connection. For example, ttys are normally owned by root and are readable only by their owner. If you intend to let ordinary users open and read the tty, you would need to make the tty world readable.

The ttys being used for the host-initiated connections should be set to "off" in the /etc/ttys file. If necessary, issue the kill command to effect the changes you have made to /etc/ttys:

```
kill -HUP 1
```

Next, you must set up the server port characteristics to match the characteristics of the device connected to the port and to allow host-initiated connections. Refer to Section 3.1 for instructions.

4.2 The Program Interface

Applications written to employ this service are much like applications for any tty device. However, there are some programming considerations.

- The programs interface with the LAT database through the LAT driver. When the host program issues an open call to the tty, the LAT driver attempts to establish a connection to the target port on the target system. The driver reports success/failure codes in the variable errno.
- When the open call is successful, the user program issues read and write system calls to handle data transfers, and normal ioctl processing for the device control information.
- A close system call on the device terminates the LAT connection.
- Other coding hints and restrictions are included in the comments to the sample program shown in Appendix B.

LAT/Telnet Gateway 5

This chapter describes how to set up and use the LAT/telnet gateway service. By employing this service, a user on a LAT terminal can connect directly to remote hosts through telnet without having to login first to a local ULTRIX-32 system.

For example, a user could be travelling on business. She could use a terminal on a LAN to connect through telnet to her home system and account, even though she did not have an account on any system in the LAN.

5.1 Setting up the Gateway

The steps you take to set up the gateway service are similar to those you take to tailor a service node for a terminal server.

5.1.1 Edit the ttys File

Select the LAT ttys to dedicate to the gateway, for example, tty20, tty21, and tty22. The number of ttys selected determines the maximum number of simultaneous LAT/telnet gateway sessions the system can deliver.

Then, edit the system's /etc/ttys file to replace getty with lattelnet for the selected devices. For example:

```
tty20    "/usr/etc/lattelnet std.9600"  vt100    on nomodem #lat/telnet gate
tty21    "/usr/etc/lattelnet std.9600"  vt100    on nomodem #lat/telnet gate
tty22    "/usr/etc/lattelnet std.9600"  vt100    on nomodem #lat/telnet gate
```

Then, use the kill command to effect the changes:

```
kill -HUP 1
```

5.1.2 Start up the Gateway

Use the lcp command to start up the gateway. For example:

```
lcp -v HOSTNAME \
    -V "HOSTNAME lat service" \
    -v telnet:/dev/tty20,/dev/tty21/,/dev/tty22 \
    -V "lat/telnet gateway service"
```

Remember, you must define the name and description of the default (the login) LAT service before you specify the gateway service name and its description.

5.1.3 Edit the rc.local File

Change the entry for the lcp command line in the /etc/rc.local file to reflect your new service. The command entry should duplicate the startup lcp command.

5.2 Using the Gateway

To use the gateway at a LAT terminal, a person enters the connect command. For example, to connect to remote node named remote using a local node named local as a gateway, the command line would be:

```
Local> connect telnet node local dest remote
```

Alternatively, a user could enter the service name (telnet) and wait to be prompted for the remote node desired. For example, the following represents what occurs when a user on local node PRINTF connects to telnet and waits for a login prompt from remote node NETRIX:

```
Local> connect telnet
Local -101- 5 other session(s) active
Local -010- Session 6 to TELNET on node PRINTF established


LAT to TELNET gateway on printf
telnet> open netrix
Trying...
Connected to netrix.
Escape character is '^]'.
netrix login:
```

Creating Your Own Service 6

ULTRIX-32 systems now can offer more than one service to the users on a LAN. Previous to ULTRIX-32, Version 3.0, the ULTRIX-32 systems running as service nodes on an Ethernet LAN offered a single service: a login service known by the node name.

In Version 3.0, the lcp command is enhanced (through the -V and -v switches), to allow service nodes to offer multiple services. One such service, a component of the operating system software, is the LAT/telnet gateway, as described in Chapter 5. By employing this service, a user on a LAT terminal can connect directly to a remote node through telnet without having to login first to an ULTRIX-32 system.

Users can also write their own specialized applications and have them advertized to terminal services.

6.1 Setting Up the Service

The steps you take to set up a service are similar to those you take to set up the LAT/telnet gateway discussed in Chapter 5. Here is a summary of the steps:

- Select the LAT ttys to be dedicated to the service.
- Edit the system's /etc/ttys file to replace getty with the name of your service.
- Use the kill command to effect the changes.
- Use the lcp command to set up the service.
- Add the lcp command for your service as an entry in the /etc/rc.local file.

For more information, please review Chapter 2 of this manual.

6.2 Using the Service

To use the service at a LAT terminal, a person issues the connect command. For example:

```
Local> CONNECT DATE
```

6.3 Programming the Service

Programming for a service can be as simple or as complicated as the service you have designed. An example of a simple service is shown in Figure 6-1, Service Example.

Example 6-1: Service Example

```
/*
 * l a t d a t e
 *
 * Description: This sample program illustrates the use of multiple
 *              lat services.  When a user at a terminal connected
 *              to a terminal server issues a "CONNECT DATE" command
 *              the date & time will be printed on his terminal.
 *
 * Setup:       It is necessary to dedicate one or more lat ttys
 *              to the service.  For example, to dedicate ttys 14
 *              & 15 you would need to edit /etc/ttys & change the
 *              lines for tty14 & tty15 to look like:
 *
 *              tty14 "/etc/latdate std.9600" vt100 on
 *              tty15 "/etc/latdate std.9600" vt100 on
 *
 *              Then do a "kill -HUP 1" for the change to take effect.
 *              Then issue an lcp command to advertise the latdlogin
 *              gateway service:
 *
 *              lcp -v hostname \
 *                  -V "HOSTNAME" \
 *                  -v date:/dev/tty14,/dev/tty15 \
 *                  -V "lat date & time service"
 *
 * To compile:  cc -o latdate latdate.c
 *
 * Example:     CONNECT DATE
 *
 */

#ifndef lint
static char *sccsid="@(#)chapter6.multiple  1.4  9/3/88";
#endif
```

(continued on next page)

```
#include <errno.h>
#include <sys/file.h>
#include <sys/ioctl.h>

struct sgttyb ttyb;
char dev[256] = "/dev/";
int latfd;

main(argc, argv)
int argc;
char *argv[];

{

    strcat(dev, argv[argc-1]);

    chown(dev, 0, 0);
    chmod(dev, 0622);

    if( (latfd = open(dev, O_RDWR)) < 0 ) {
    perror(dev);
    exit(1);
    }

    ttyb.sg_flags = CRMOD;
    ioctl(latfd, TIOCSETP, &ttyb);

    dup2(latfd, 0);
    dup2(latfd, 1);
    dup2(latfd, 2);

    execl("/bin/date", "lat-date", (char *)0);
}
```

After completing a program for the service, set up the service following the same steps as outlined in this chapter.

A sample program, one that can be used to replace getty for a lat:dlogin gateway, is provided in Appendix C.

A new ioctl, LIOCTTYI, has been created. This ioctl returns information specific to lat ttys in a structure of type ltattyi. The structure is defined in the header file, /sys/h/ltatty.h.

Within the ltattyi structure, the server name and the port associated with a given lat tty are returned in structure members lta_server_name and lta_port_name as null-terminated ASCII strings. If the user at the lat terminal specifieds a destination string in his request to connect to the service, that string is returned in structure member lta_dest_port.

Other coding hints and restrictions are included in the comments to the program in Appendix C.

Server Maintenance Functions A

This appendix discusses the control functions of ULTRIX-32 host nodes and remote node maintenance functions, such as:

- Obtaining a Processor's Hardware Ethernet Address
- Downline-Loading a Terminal Server
- Upline-Dumping of Memory
- Enabling Remote Console Capabilities

Note

Because it does not have a mass storage device, a terminal server must load the system software it uses to communicate with the terminals and the Ethernet network from one of the processors on the network. If the terminal server software fails, it sends its crash dump image over the network to an available processor.

In the context of this chapter, a node is equivalent to a system. A node can be a terminal server or a system running the ULTRIX-32 software. An example of a target node is a DECserver 200.

A.1 Control Functions of ULTRIX-32 Host Nodes

On an Ethernet network, processors that are running ULTRIX-32 software can act as host nodes for unattended remote nodes. A host system (node) can perform these functions:

- Down-line load bootstrap loaders and operating system images to a remote node such as a terminal server
- Receive an up-line dump of a memory image from a remote node
- Connect to a console server on a remote node and allow a local terminal to act as a console for that remote node

A host node can be the primary or a backup host node. Backup host nodes perform host functions if a primary host is unavailable.

To understand down-line loading and up-line dumping, you must understand the nodes involved in a loading or dumping sequence. In the following node descriptions, the command node and the host node can be either the same or different nodes, but neither can be the target node (the unattended remote node).

- Command Node

  You can initiate a down-line load request using the load and trigger commands at a command node. These commands cause a target node to issue a down-line load request.

  Only target nodes initiate up-line dump requests. There are no commands that you can use to initiate an up-line dump.

- Host Node

  The host node actually performs a down-line load, receives an up-line dump, or connects to a remote console server. The host node must be on the same Ethernet network as the target node, because down-line loads, up-line dumps, and connections to remote console servers are performed using circuit-level access instead of logical links.

- Target Node

  The target node receives the bootstrap loaders and the system image file. Target nodes issue down-line load requests either in response to requests from command nodes or in the course of the hardware bootstrap routine. If a target node senses an impending system failure, it can initiate an up-line dump request.

A.2 Obtaining the Hardware Ethernet Address

There are three ways to obtain the hardware Ethernet address for a processor: through the DECnet ncp program, using the ULTRIX arp command, and at the system console when the processor is in console mode.

A.2.1 Using the ncp Program

To use the ncp program to find the Ethernet address for a processor, type:

```
# ncp show line dev-c characteristics
```

where dev-c is the circuit ID of the circuit, as specified in Chapter 2 of this document.

A.2.2 Using the arp Command

You can use the arp command to find the Ethernet address for a processor. For example, if the host name is bangor, type:

```
# arp bangor
```

The Ethernet address for bangor is displayed, for example, as:

```
bangor (128.47.40.94) at 8:0:2b:3:f4:a2
```

Note

Should no entry exist for the processor's address, use the rsh command to resolve the entry. Type, for example:

```
# rsh bangor who
```

Even if this command fails, the arp table entry for the processor's Ethernet address is filled.

A.2.3 Using Console Mode

If the ncp and arp commands fail, you can use the console mode. Follow these instructions, which assume your hardware has been installed at the default control status register (CSR) addresses:

First, obtain the console mode prompt:

```
>>>
```

What you then do depends on your processor type.

If your processor is a MicroVAX II, VAXstation II, or VAXstation II/GPX, type the following at the console mode prompt:

```
>>> e/p/w 20001920
    P 20001920  FF08

>>> e
    P 20001922  FF00

>>> e
    P 20001924  FF2B

>>> e
    P 20001926  FF03

>>> e
    P 20001928  FF05

>>> e

    P 2000192A  FF8B
```

The hardware Ethernet address is made up of the last two characters from each line displayed by the system. In the previous example, the hardware Ethernet address is:

```
08-00-2B-03-05-8B
```

If your processor is a VAXstation 2000 or MicroVAX 2000, type the following at the console mode prompt:

```
>>> t 50
```

The system prints the hardware Ethernet address on the second line of the t 50 printout. Note that ID is not part of the hardware Ethernet address:

```
.
.
.
ID 08-00-2B-02-F0-36
```

If your processor is a MicroVAX or VAXstation 3000 series system, type the following at the console mode prompt:

```
>>> show ether
```

The system prints the hardware Ethernet address:

```
08-00-2B-02 -f0-36
```

Note

If the target node is a DECserver 200, the Ethernet hardware address is on a label on the back of the unit.

A.3 Down-Line Loading

Ethernet nodes running ULTRIX-32 software can down-line load an operating system image to a remote node. For example, you can down-line load a DECserver 200 system load image file from your ULTRIX-32 system to a DECserver 200. In this case, the load image file is pr0801eng.sys.

A.3.1 Prerequisites to Downline Loading

Before attempting a down-line load operation, you must ensure that the nodes, lines and circuits involved in the load meet the following requirements:

- The target node must be on the same Ethernet as the ULTRIX-32 host system, because a host uses circuit-level access to load a target node.
- If the load is operator-initieated, the bootstrap at the target node must be capable of both recognizing trigger messages and sending program requests.
- The physical hardware devices must be set up correctly to support the load.
- For target-initiated loads, the host node device involved in the load operation must be enabled to perform service functions. To enable the host node device, run /etc/mop_mom as a background task on the ULTRIX-32 system. When mop_mom receives a program load request, it forks and executes the loader, /usr/lib/dnet/mop_dumpload, which performs the down-line load.
- The host node must have access to the load files. The location of the files can be specified in the target node's program load request or can default to the information contained in the nodes data base.

Down-line loading can be initiated by an operator or by a target node, such as a DECserver 200.

A.3.2 Operator-Initiated Down-Line Loads

You can initiate a down-line load using the load or the trigger command, depending on whether the operation is initiated by the host node or the target node.

A.3.2.1 The load Command – You can use the load command to cause the host node to load the specified target node, such as a DECserver. Before you can issue a load command, the target node must support the load command, and the nodes data base must contain these definitions:

- The service circuit over which the load is performed
- The Ethernet hardware address of the target node
- The service password needed to gain access to the target node, if not specified in the load command
- The name of the image file to use, if not specified in the target node's program load request

You can define an entry in the nodes data base with the addnode command. For security reasons, you can choose not to include a target node's service password in the data base. In that case, you must specify the service password in the load command line, using the -p option.

The load command sends a Maintenance Operation Protocol (MOP) bootstrap message to a target node, and then waits for the target node to send its program load requests. The load command honors requests for secondary, tertiary, and system loaders, in that order, starting from any stage in the loading sequence. The load command waits for program requests from the target node until the operating system has been sent to the target and then the load request exits. The file /usr/spool/mqueue/syslog contains information created by load requests.

A.3.2.2 Using the load Command – Use the following procedure for down-line loading a target node using the load command:

1. You can use the dsvconfig script, if available, or the addnode command to define the required information in the nodes data base for the target node. In this example, the target node is a DECserver 200 called auburn:

```
# addnode auburn -h 08-00-2b-02-40-4e \
-p 12345 \
-c una-0 \
-l ps0801eng
```

   The addnode command in this example specifies one load file for auburn, because the node is loaded in a single stage.

2. Issue the load command:

```
# /etc/load auburn
```

The load command in the example sends a MOP bootstrap message to node auburn, which responds by sending a program load request for the primary loader to the host node.

A.3.2.3 The trigger Command – The trigger command directly triggers the bootstrap mechanism of a target node, causing the target to send a program load request to the Ethernet dump/load assistance multicast address. The trigger command has the same result as pushing the BOOT switch on a target node. It initiates a down-line load to the target node from the first host node to respond to the request.

Before you can issue a trigger command, the target node must support the trigger command, and the nodes data base must contain these definitions:

- The service circuit over which the load is performed
- The Ethernet hardware address of the target node
- The service password needed to gain access to the target node, if not specified in the trigger command

- The name of the image file to use, if not specified in the target node's program load request

You can define an entry in the nodes data base with the addnode command. For security reasons, you do not need to include a target node's service password in the data base. In that case, you must specify the service password in the trigger command line, using the -p option.

A.3.2.4 Using the trigger Command – Use the following procedure to initiate a down-line load using the trigger command:

1. Use the addnode command to define the required information in the nodes data base for the target node.
2. Issue the trigger command:

```
# /etc/trigger auburn
```

Note that this example assumes that the service circuit, the Ethernet hardware address, and the service password for auburn are defined in the nodes data base. The command in the example sends a MOP bootstrap message to the target node auburn. This boot message causes auburn to send a program load request message to the Ethernet load assistance multicast address. The trigger exits, and auburn continues to carry out the procedure for a target-initiated down-line load.

A.3.3 Target-Initiated Down-Line Loads

A target node initiates a down-line load by triggering its bootstrap ROM and issuing a program load request.

A target-initiated down-line load occurs when a target node does not have a specific host node from which to request a program load (for example, if a target's host node crashes, or when the BOOT switch on a target node is pressed). Target-initiated loads proceed as follows:

1. The target node sends a program load request message to the Ethernet load assistance multicast address AB-00-00-01-00-00. This message is a request for any node on that Ethernet to perform the load.
2. Each node on the Ethernet whose circuits are enabled for service operations searches its nodes data base for an entry corresponding to the information in the program load request. When a node finds a node entry with an Ethernet hardware address matching the hardware address of the requesting target node, the node determines if it can down-line load the target. The node then sends the secondary loader to the target node, if requested, or a message volunteering to perform

the load, if the target is requesting the tertiary loader or system load file. This information is logged in /usr/spool/mqueue/syslog.

3. The target chooses the node that responds first to proceed with the loading sequence. It does not send a message to any other node. The loading sequence (described in Section 4.3.3) continues with the designated host node performing the down-line load.

A.3.4 The Down-Line Load Sequence

The load sequence is the same, whether a load request is initiated by the system manager or by a target node.

The first program to run at the target node is the primary loader. Typically, this program is executed directly from the target node's bootstrap ROM or is in the microcode of the load device (such as una or qna). The target node's primary loader is triggered, and the target node sends a request program load message to the host node. Usually, the primary loader requests a secondary loader program, which, in turn, requests a tertiary loader. The last program to be loaded is the operating system.

In this sequence, each program requests the next one until the operating system is loaded. After the load sequence is complete, the target receives a message with the name of the host and places the name in its volatile data base.

Note

For a DECserver 200 terminal server, the first and only program load request message is for the system load file, such as pr0801eng.sys or ps0801eng.sys.

A.4 Up-Line Dumping of Memory

You can include certain parameters in the nodes data base that allow a specified Ethernet target node to dump its memory into a file on your ULTRIX-32 system. This procedure is called up-line dumping. Up-line dumping is a valuable tool for crash analysis.

When a target node that is capable of up-line dumping detects an

impending system failure, that system requests an up-line dump.

Note

You should check the /usr/spool/mqueue/syslog file after a power failure or severe weather storm. Many up-line dump requests can cause the syslog file to grow quite large in a relatively short amount of time.

Up-line dumping, unlike down-line loading, is always initiated by the target node. There are no commands to initiate an up-line dump. The ULTRIX-32 system uses the Maintenance Operation Protocol (MOP) to perform an up-line dump.

A.4.1 Prerequisites to Upline Dumping

Before attempting an up-line dump operation, you must ensure that the nodes, lines, and circuits involved in the up-line dump operation meet these requirements:

- The target node must be on the same Ethernet as the host node, because the host uses circuit level access to dump the target node.
- The host node device involved in the dump operation must be enabled to perform service functions. To enable the host node device, run /etc/mop_mom as a background task on your ULTRIX-32 system. When mop_mom receives an up-line dump request, it forks and executes the loader, /usr/lib/dnet/mop_dumpload, which performs the dump.
- The target must supply a memory size value and a starting memory address in the request memory dump message that it sends to the host.
- The host must have a dump file for the target node specified in its nodes data base, and it must be able to create this file. The dump file is defined with the addnode command.

A.4.2 Up-Line Dump Sequence

The following steps outline the up-line dump process:

1. When a target node senses a system failure, it sends a request dump service message to its host node, the node that originally down-line loaded it. If the host node is available, the up-line dump proceeds as described in Step 2. If the host node is unavailable, the target node sends a memory dump request to the Ethernet dump/load assistance

multicast address AB-00-00-01-00-00. This message contains information about the memory size and the up-line dump device type at the target node.

Each node on the Ethernet checks its nodes data base to determine if it can accept an up-line dump from the target node. The nodes that can accept dumps respond to the target node. The target node chooses the first node that responds to continue the dumping sequence. It does not send a message to any other node. The dumping sequence then continues as described in Step 2.

2. From the dump request message sent by the target node, the host retrieves the memory dump count and the memory address from which to start dumping. It also retrieves the name of the file where the target's memory image will be stored from the target's nodes data base entry. The host node then sends to the target node a MOP request memory dump message with the starting address and buffer-size values.
3. Using the values it receives from the host, the target returns the requested block of memory in a MOP memory dump data message. The host receives the block of dump data, places it in the dump file, increments the memory address by the number of locations sent, and sends another request memory dump message to the target. This sequence is repeated until the amount of memory dumped matches the memory size specified in the dump request.
4. When the up-line dump is completed, the host node sends a dump complete message to the target node and attempts to down-line load the target by sending a trigger message.

A.5 Remote Console Capabilities

The console carrier requester command, ccr, sets up a logical connection between your ULTRIX-32 system and the console carrier server on a remote node. The ccr command enables a terminal to act as the console for a remote unattended node. For example, your terminal can act as the console for the DECserver 100 terminal server and its resident software.

You can use the ccr command to force a crash if a server node becomes unresponsive. To determine how to force a crash, see the documentation for the server product.

A.5.1 The ccr Command

When you use the ccr command, the remote console carrier server is in one of these states:

- Loaded and unreserved
- Loaded and reserved
- Not loaded

If the console carrier server is loaded and unreserved, issuing the ccr command reserves it, and this message appears on your terminal:

```
ccr: Remote console reserved
```

If the console carrier server is loaded and reserved by another user, the following message appears on your terminal:

```
ccr: Remote console already in use
```

If the console carrier server is not loaded, the ccr command loads the server. To load a server, an ULTRIX-32 system may need to have the console carrier server image file and its loader file present in the directory /usr/lib/dnet. When the server is loaded, the ccr command reserves the console and proceeds.

A.5.2 Requirements of the ccr Command

To use the ccr command, these requirements must be met:

- The host node (your ULTRIX-32 system) and the remote node must be on the same Ethernet.
- The nodes data base must contain these definitions, or they must be specified in the ccr command line:
  - The service circuit to the target node
  - The Ethernet hardware address of the target node
  - The service password needed to gain access to the target node

You can define an entry in the nodes data base with the addnode command.

A.5.3 Using the ccr Command

Use the following procedure to connect to a remote console server with the ccr command:

1. Make sure that all of the ccr requirements (outlined above) have been met.

2. Issue the ccr command:

```
# /etc/ccr dallas
ccr: Remote console reserved
# (enter DECserver remote access password)
      .
      .
      .
<CTRL/D>
ccr: Remote console released
#
```

In the example, this command connects to the remote console server on dallas. Your terminal remains in console carrier mode until you press CTRL/D to terminate the ccr command.

Note

In some cases you may have to enter a service password to access the remote console. See the remote console's documentation or system manager to determine the password.

A.5.4 The ccr Command Special Characters

The ccr command uses the following special characters:

- CTRL/B operates as a break command to get the attention of the console on-line debugging tool (ODT).
- CTRL/D exits from console carrier mode and terminates the ccr command.

Program for Host-Initiated Connections B

This appendix contains a sample program that employs a LAT host-initiated connection, commonly called a reverse lat.

The program is on the software distribution, in the ULTRIX-32 examples directory. The pathname is: /usr/examples/lat/dial.c

```
/*
 * d i a l
 *
 * Description: This sample program illustrates the use of a LAT Host
 *              Initiated Connection.  It connects /dev/ttyxx to a DEC
 *              SCHOLAR modem that is attached to the port "LAT_PORT"
 *              on the DECserver 200 "LAT_SERVER".  After a successful
 *              open, it autodials a phone number to a host computer
 *              and emulates a terminal connected to the host computer.
 *
 * Setup:       Before invoking 'dial', LAT_SERVER and LAT_PORT must be
 *              defined by the lcp command:
 *
 *              lcp -h /dev/ttyxx:LAT_SERVER:LAT_PORT
 *
 *              Access to '/dev/ttyxx' must be Read/Write for the user
 *              of 'dial'.
 *
 * To compile:  cc -o dial dial.c
 *
 * Usage:       dial phone# /dev/ttyxx
 *
 * Comments:    In terminal emulation:
 *                ^](CTRL/]) for escape character
 *                ^]? for help
 *                ^]b to send break signal
 */

#ifndef lint
static char *sccsid="@(#)appb.revlatexample 1.4  9/3/88";
#endif

#include <stdio.h>
#include <ctype.h>
#include <signal.h>
#include <sgtty.h>
#include <sys/types.h>
#include <sys/file.h>
#include "/sys/h/ioctl.h"


/*
 * For DEC SCHOLAR modem (See SCHOLAR 2400 Modem Owner's Manual)
 * byte 1:      1 (CTRL/A) - autodialer
 * byte 2:      P - pulse dialing  T - tone dialing
 * last byte:     ! - start dialing
 */
u_char nl[20]={0x01, "P123-4567!"};

int fd;
void nodial();
```

(continued on next page)

```
main(argc,argv)
int argc;
char *argv[];
{
    char buf[BUFSIZ];    /* Read/write buffer */
    int len;

    /*
     * Open reverse LAT device.  Set the O_NDELAY bit so
     * that we get an EBUSY error if the LAT_PORT is busy.
     * Without this, our request might get queued by the
     * terminal server (if the port is busy & queuing is on)
     * and we might sit waiting for a long time.
     */
    if ( (fd = open(argv[2],O_RDWR|O_NDELAY)) < 0 )
    {
    perror(argv[0]);
        goto doneonerror;
    }

    len = strlen(argv[1]);          /* get phone # */
    strcpy(&nl[2], argv[1]);
    nl[len+2] = '!';                /* ! for start dialing */
    write(0, "Dialing ", 8);        /* print 'Dialing phone#, wait...' */
    write(0, argv[1], len);
    write(0, ", wait... ", 10);
    write(fd, nl, len+3);           /* send phone # to modem for autodial */

    signal(SIGALRM, nodial);            /* Give call 60 seconds to go thru */
    alarm(60);
    read(fd, buf, 80);                  /* get echo of phone # */
    signal(SIGALRM, SIG_IGN);
    len = read(fd, buf, 80);            /* get return status */
    buf[len] = 0x00;
    printf("%s", buf);                  /* print return status */
    if (buf[0] == 'A') termmain();      /* act as terminal emulator if */
                                        /* 'Attached', exit otherwise */

doneonerror:
    printf("Try later\n");
    exit(1);
}

void nodial()
{
    char buf[BUFSIZ];    /* Read/write buffer */

    printf("\nDial out failed\n");
    exit(1);
}
```

(continued on next page)

```
/*
 * The remainder of the this program is a terminal emulator.
 */

struct sgttyb Isgttyb, sgttyb, sgttyb1;
struct tchars Itchars, tchars1;
struct ltchars Iltchars, ltchars;
int fd, readfd, writefd, exception, outfile, ret, ret1;

void resettty();


termmain()
{

    char buf[BUFSIZ];    /* Read/write buffer */
    char *bufptr;
    int on = 1;

    ioctl(0, TIOCGETP, &Isgttyb);
    ioctl(0, TIOCGETC, &Itchars);
    ioctl(0, TIOCGLTC, &Iltchars);

    /*
     * Set the terminal into CBREAK | NOECHO | -CRMOD mode so
     * that we can handle character buffering and echo ourselves. We will
     * also disable all special character handling except ^S and ^Q.
     */
    sgttyb = Isgttyb;
    sgttyb.sg_flags |= CBREAK;
    sgttyb.sg_flags &= ~(ECHO | CRMOD);
    ioctl(0, TIOCSETP, &sgttyb);
    tchars1 = Itchars;
    tchars1.t_intrc = tchars1.t_quitc = tchars1.t_eofc = tchars1.t_brkc = -1;
    ioctl(0, TIOCSETC, &tchars1);
    ltchars.t_suspc = ltchars.t_dsuspc = ltchars.t_rprntc = ltchars.t_flushc
                    = ltchars.t_werasc = ltchars.t_lnextc = -1;
    ioctl(0, TIOCSLTC, &ltchars);

    ioctl(fd, TIOCGETP, &sgttyb1);
    sgttyb1.sg_flags |= RAW;
    sgttyb1.sg_flags &= ~ECHO;
    ioctl(fd, TIOCSETP, &sgttyb1);
    ioctl(fd, FIONBIO, &on);

    signal(SIGHUP, resettty);
    signal(SIGINT, resettty);
    signal(SIGQUIT, resettty);
    signal(SIGBUS, resettty);
    signal(SIGSEGV, resettty);
```

(continued on next page)

```
printf("escape character: ^];    help: ^]?\r\n\n");
for (;;)
{
    readfd = exception = (1 << fd) + (1 << 0);
    if ((select(fd+1, &readfd, 0, &exception, 0)) > 0)
    {
        if (readfd & (1 << fd))
        {
            if ((ret = read(fd,buf,BUFSIZ)) <= 0)
            {
                printf("ret: %d\n", ret);
                goto done;;
            }
            ret1 = write(0,buf,ret);
            ret -= ret1;
            bufptr = buf + ret1;

            while (ret)
            {
                writefd = 1 << 0;
                select(fd+1, 0, &writefd, 0, 0);
                if (writefd & (1 << 0))
                {
                    ret1 = write(0,bufptr,ret);
                    ret -= ret1;
                    bufptr = bufptr + ret1;
                }
            }
        }
        if (readfd & (1 << 0))
        {
            ret = read(0,buf,BUFSIZ);
            if (*buf == 0x1d)
            {
        if ( !(*buf = esccommands()))
                    continue;
            }
            write(fd,buf,ret);
        }
        if (exception & (1 << fd))
        {
            printf("exception: \n");
            goto done;
        }
    }
```

(continued on next page)

```
        else
            {
                perror("select: \n");
                goto done;
            }
        }

done:
    printf("\nEXIT! ");
    resettty();
}

void resettty()
{
    int off = 0;

    /*
     * Restore the terminal characteristics to their state before the
     * current session was entered.
     */
    ioctl(0, TIOCSETP, &Isgttyb);
    ioctl(0, TIOCSETC, &Itchars);
    ioctl(0, TIOCSLTC, &Iltchars);
    close(fd);
    printf("\nUltrix LAT dial out disconnected\n\n");
    exit(0);
}
```

(continued on next page)

```
/*
 *        e s c c o m m a n d s
 *
 * for input chatacter:
 * ?:         this menu
 * p:          escape to local command mode
 * b:          send a break
 * esc:      send ^]
 * all others:      exit esacape mode
 *
 */
esccommands()
{
    char ch;
    int ret;

    ret = read(0,&ch,1);
    switch(ch)
    {
    case 'p':
        localcommands();
        break;

    case 'b':
            ioctl(fd, TIOCSBRK, 0);
        break;

    case 0x1b:
        return (0x1d);

    case '?':
        printf("\t?\tthis menu\r\n");
          printf("\tp\tescape to local command mode (? for help)\r\n");
        printf("\tb\tsend a break\r\n");
        printf("\tescape\tsend ^]\r\n");
        printf("\tothers\texit escape mode\r\n");

    }
    return(0);
}
```

(continued on next page)

```
/*
 *        l o c a l c o m m a n d s
 */
extern char **environ;
localcommands()
{
    char command[512];
    int notdone = 1,pid;

    /*
     * Reset the terminal to its original state.
     */
    ioctl(0, TIOCSETP, &Isgttyb);
    ioctl(0, TIOCSETC, &Itchars);
    ioctl(0, TIOCSLTC, &Iltchars);

    printf("\n");
    while (notdone)
    {
    printf("local command> ");
    if (gets(command) == NULL)
    {
            printf("\nEXIT! ");
        resettty();
    }
    switch (command[0])
    {
        case '?':
        printf("\tsuspend\tsuspends lat\n");
        printf("\texit\texits\n");
        printf("\t^D\texits\n");
        printf("\tcmd\tinvoke shell to execute command\n");
        printf("\t\tblank line resumes lat\n\n");

        case '\0':
        notdone = 0;
        break;
```

(continued on next page)

```
        default:
        /*
         * Check for special commands that we handle locally.
         */
        if (strcmp(command, "suspend") == 0)
        {
            kill(getpid(), SIGTSTP);
            break;
        }
        if (strcmp(command, "exit") == 0)
        {
                    printf("\nEXIT! ");
                resettty();
        }
        pid = fork();
        if (pid < 0)
        {
            perror("lat server - fork failed");
            break;
        }
        if (pid == 0)
        {
            if (execle(getenv("SHELL"), getenv("SHELL"), "-c",
                                command, 0, environ) < 0)
            {
            perror("lat server - unable to exec shell");
            exit(1);
            }
        }
        wait(0);
        break;

    }
    }

    /*
     * Reset the terminal to its state on entry.
     */
    ioctl(0, TIOCSETP, &sgttyb);
    ioctl(0, TIOCSETC, &tchars1);
    ioctl(0, TIOCSLTC, &ltchars);
}
```

Program to Replace getty for Special Services C

This appendix contains a sample program. The program can be used to replace the /usr/etc/getty program for each tty to be used as a LAT/dlogin gateway in a local network. The program is on the software distribution, in the ULTRIX-32 examples directory. The pathname is: /usr/examples/lat/latdlogin.c

C.1 Setting Up the Program

Before running the program, you must execute an lcp command to define the service and the tty devices to be used for the service. For example:

```
lcp -v nodnam -V "nodnam lat service" -v latdlogin:/dev/tty7,/dev/tty8 \
-V "lat/dlogin gateway service"
```

The example lcp command reserves the LAT ttys with minor devices 7 and 8 for use as lat/dlogin gateways. In addition, you must change the tty for entries in /etc/ttys, replacing getty with the name of the program (latdlogin). For example:

```
tty07    "/etc/latdlogin 2"    vt100 on nomodem  #lat/dlogin gateway
tty08    "/etc/latdlogin 2"    vt100 on nomodem  #lat/dlogin gateway
```

C.2 Using the Program

To use the service, a person at a terminal can use the connect command to establish a connection to a remote node. For example:

```
Local> connect latdlogin node local dest remote
```

As a result, the server would return the following message to the user:

```
Local> LAT TO DLOGIN GATEWAY ON LOCAL CONNECTING TO REMOTE>
```

Followed by the login prompt from remote node REMOTE.

When a user logs out from the remote session, the message returned to the user is:

```
dlogin -- session terminated
local -011- session x disconnected from LATDLOGIN
```

```
/*
 * l a t d l o g i n
 *
 * Description: This sample program acts as a LAT to DLOGIN gateway.
 *              With it, a user at a terminal connected to a terminal
 *              server can log into remote DECnet nodes without
 *              having to log into (or even have an account on) the
 *              local system.
 *
 * Setup:       This program requires that DECnet be installed on
 *              your system.  It is necessary to dedicate one or
 *              more lat ttys to the service.  For example, to
 *              dedicate ttys 14 & 15 you would need to edit
 *              /etc/ttys & change the lines for tty14 & tty15 to look like:
 *
 *              tty14 "/etc/latdlogin std.9600" vt100 on
 *              tty15 "/etc/latdlogin std.9600" vt100 on
 *
 *              Then do a "kill -HUP 1" for the change to take effect.
 *              Then issue an lcp command to advertise the latdlogin
 *              gateway service:
 *
 *              lcp -v hostname \
 *                  -V "HOSTNAME" \
 *                  -v latdlogin:/dev/tty14,/dev/tty15 \
 *                  -V "lat/dlogin gateway"
 *
 * To compile:  cc -o latdlogin latdlogin.c
 *
 * Example:     To access DLOGIN service from LAT terminal:
 *              CONNECT dlogin NODE hostname DEST DECnet_nodename
 *
 * Comments:    More extensive tty set up could be added (such as
 *              for the parameters defined in gettytap & termcap).
 *              See getty(8).  See 'Guide to Ethernet Communication
 *              Servers' for LAT service set up.
 */

#ifndef lint
static char *sccsid="@(#)appc.gatewayexample    1.5  9/7/88";
#endif

#include <sys/ltatty.h>
#include <sys/ioctl.h>
#include <sgtty.h>
#include <ctype.h>
#include <sys/file.h>
#include <stdio.h>
```

(continued on next page)

```
struct  sgttyb mode = { 0, 0, CERASE, CKILL, CRMOD|ECHO };
char    hostname[256];
char    tty[256] = "/dev/";
struct  ltattyi ltainfo;
long    flags = LCRTERA | LCRTBS | LPRTERA;
int     latfd;
char    *np;


main(argc, argv)
int argc;
char *argv[];

{
    gethostname(hostname, sizeof(hostname));

    /* generate full path name to device special file */
    strcat(tty, argv[argc-1]);

    /* change mode & owner of tty */
    chown(tty, 0, 0);
    chmod(tty, 0622);

    /* open LAT line */
    latfd = open(tty, O_RDWR);

    /* get DESTINATION field */
    ioctl(latfd, LIOCTTYI, &ltainfo);

    /* make tty stdin, stdout, & stderr */
    dup2(latfd, 0);
    dup2(latfd, 1);
    dup2(latfd, 2);

    /* set tty flags & mode */
    ioctl(0, TIOCLSET, &flags);
    ioctl(0, TIOCSETP, &mode);
```

(continued on next page)

```
    if (ltainfo.lta_dest_port[0] != 0)
    {
        /* A destination was specified in the connect request. */
        /* Upper-case it, then exec dlogin. */
        printf("\nLAT to DLOGIN gateway on %s connecting to %s\n",
               hostname,ltainfo.lta_dest_port);
        for (np = ltainfo.lta_dest_port; *np; np++)
        {
            if (isupper(*np))
                *np = tolower(*np);
        }
        execl("/usr/bin/dlogin","dlogin",ltainfo.lta_dest_port,0);
    }
    else
    {
        /* No destination specified.  Print usage & exit. */
        printf("\nLAT to DLOGIN gateway usage: ");
        printf("CONNECT dlogin NODE %s DEST DECnet_host\n", hostname);
        close(latfd);
        exit(0);
    }
}
```

Index

U

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
and New Hampshire,
Alaska or Hawaii
call **800-DIGITAL**

In Canada
call **800-267-6215**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473

*Any prepaid order from Puerto Rico must be placed
with the Local Digital Subsidiary:
809–754–7575

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

Zip Code

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line