ULTRIX

digital

Guide to Ethernet Communications Servers

Order Number: AA–ME98B–TE

ULTRIX

Guide to Ethernet Communications Servers

Order Number: AA-ME98B-TE

June 1990

Product Version: ULTRIX Version 4.0 or higher

This guide describes administrative tasks and procedures for setting up and maintaining the interfaces between the ULTRIX operating system and communications servers in an Ethernet local area network (LAN).

digital equipment corporation
maynard, massachusetts

Contents

About This Manual

1 Ethernet Communications Servers

2 Terminal Servers

A Remote Node Maintenance

B Program for Host-Initiated Connections

C Program to Replace getty for Special Services

Examples

About This Manual

This guide describes administrative tasks and procedures for setting up and maintaining the interfaces between the ULTRIX operating system and communications servers in an Ethernet local area network (LAN).

Audience

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

To use this guide, you must know how to use ULTRIX 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 Communications Servers Introduces communications servers and terminal servers.
Chapter 2	Terminal Servers Describes how to set up a terminal server to work with ULTRIX 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	Remote Node Maintenance Describes the procedures and commands you use to maintain remote ULTRIX service nodes in a LAN.
Appendix B	Program for Host-Initiated Connections 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/Telnet gateway.

Related Documents

For the technical information you need 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 operating system. This documentation includes:

- Terminal server documentation

  You need a 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 the commands `connect`, `logout`, and `show`.

- ULTRIX operating system guides

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

- ULTRIX Reference Pages Section 1

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

- ULTRIX Reference Pages Sections 4 and 5

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

- ULTRIX Reference Pages Section 8

  You need to know how to use the 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 for your particular processor, for example, `E` for examine and `T` for test, as defined in your VAX processor documentation.

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

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:

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.

#	A number sign is the default superuser prompt.
>>> CPU*nn*>>	The console subsystem prompt is two right angle brackets on RISC systems, or three right angle brackets on VAX systems. On a system with more than one central processing unit (CPU), the prompt displays two numbers: the number of the CPU, and the number of the processor slot containing the board for that CPU.
`Local>`	The Local Area Network (LAN) prompt. This is the prompt issued by a terminal server on an Ethernet LAN. The prompt for your LAN may be different.
`user input`	This bold typeface is used in interactive examples to indicate typed user input.
`system output`	This typeface is used in interactive examples to indicate system output and also in code examples and other screen displays. In text, this typeface is used to indicate the exact name of a command, option, partition, pathname, directory, or file.
UPPERCASE lowercase	The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
macro	In text, bold type is used to introduce new terms.
filename	In examples, syntax descriptions, and function definitions, italics are used to indicate variable values; and in text, to give references to other documents.
`cat(1)`	Cross-references to the *ULTRIX Reference Pages* include the appropriate section number in parentheses. For example, a reference to `cat(1)` indicates that you can find the material on the `cat` command in Section 1 of the reference pages.

New and Changed Information

This manual is a revision. New and changed information includes:

- Corrections to `lcp` command syntax for LAT/Telnet examples in Chapter 5, Chapter 6, and Appendix C
- Insertion of RISC processor information in Sections 2.1.2.1 and 2.1.5.3
- Clarification of using console mode for VAX processors in Section A.2.3

In addition, there are several editorial, format, and typographical changes.

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. In some cases, terminal servers can connect synchronous devices to service nodes.

- DECnet Router

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

- DECrouter 2000

  Transfers data packets between DECnet nodes on an Ethernet LAN and remote DECnet nodes or other Ethernet LANs by means of 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 operating system, refer to the *ULTRIX 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 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.

Terminal Servers 2

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

Before you set up the ULTRIX 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 to be either a load host for the terminal server, or a service node that is available to users of devices that are connected to the terminal server.

2.1 Tailoring an ULTRIX Load Host

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

An ULTRIX 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.

Follow these steps to tailor a load host:

1. Edit the system configuration file (the `config` file), if necessary.
2. Edit the `rc.local` file.
3. Install the terminal server load image.
4. Reboot the ULTRIX system, if necessary.
5. Downline-load the terminal server load image.

2.1.1 Edit the System Configuration File

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

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

When you add entries to the `config` file, you must rebuild the kernel to make the additions take effect. You can rebuild the kernel manually or by using the `doconfig`(8) command.

2.1.2 Edit the rc.local File

On the load host, you edit the `/etc/rc.local` file so the system will start the network interface to the load host and the `mop_mom` daemon whenever it is rebooted.

2.1.2.1 Start the Network Interface to the Load Host – To interface the hardware to the network, add an entry to the `/etc/rc.local` file for the specific hardware interface that you have. The entry must be positioned in the file so that it is executed before the `mop_mom` daemon commands are executed.

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) or RISC processors:

```
/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, add the following commands to the `/etc/rc.local` file after the entry that 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 database.

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 database. 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 database by setting the environment variable `LOADUMP_SECURE` to `on`. Then, only systems you have entered in the nodes database will be loaded.

You can set this environment 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 other `mop_mom` entry:

```
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 installed. This software is not included in the ULTRIX 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.

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, 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 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 downline-loading the terminal server:

- Turn power on to the terminal server.
- Use the `load` command.
- Use the `trigger` command.

The first two methods are described in this section. The `trigger` command is described in Appendix A.

2.1.5.1 Turn on Power to the Terminal Server – To downline-load the terminal server, turn on power to it. When the terminal server is 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 downline load, the ULTRIX system checks its nodes database for the terminal server requesting the load. If an entry for the terminal server is in the nodes database, the system volunteers to downline-load the software to the terminal server.

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

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

In most instances, the first system that volunteers to send the software downline-loads the image to the terminal server. The Ethernet address of the system performing the downline load appears on the terminal server console during the process.

Be sure the system that is downline-loading the image software to your terminal server has the current version of the LAT software. To find the server characteristics, enter this command 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 downline load was successful, look in the syslog file in the /usr/spool/mqueue directory on the system that downline-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 downline load failed because the load image software, for example 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 database on that system.

2.1.5.3 Use the load Command – Some terminal servers allow you to downline-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 database. To do this, use the dsvconfig setup script included with the terminal server software. The dsvconfig script is not included with the ULTRIX distribution, but comes with the terminal server's load image software. The terminal server's software installation guide tells you how to enter the nodes you want to downline-load using the dsvconfig script.

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 nodes database; the second updates the volatile database.

Once dsvconfig is installed, you need to run the following script:

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

Refer to the terminal server installation guide for more information.

You can enter information in the nodes database manually by using the addnode command with the -h, -l, and the -c options. For example:

```
# /etc/addnode system -h 08-00-2B-05-40-B7 -l pr0801eng -c qna-1
```

In the example, system is the name of the terminal server; 08-00-2B-05-40-B7 is the terminal server physical Ethernet address; and qna-1 is the service circuit ID of the local processor. The service circuit ID varies depending on the processor type, as follows (where *n* is the unit number):

qna-*n* For Q-bus–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 or for RISC processors

After you have added the terminal server information to your system's nodes database 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 when the prompt returns.

2.1.5.4 Determine If load Succeeded – To see if the downline 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 downline load failed because the system image software 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)
```

2.2 Tailoring an ULTRIX Service Node

A service node is any 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.

Follow these steps to tailor an ULTRIX service node:

1. Edit the system configuration file (the config file), if necessary.
2. Make the lta devices.

3. Edit the `ttys` file.
4. Edit the `rc.local` file.
5. Reboot the ULTRIX system, if necessary.

2.2.1 Edit the System Configuration File

If your system is going to be a service node, edit the system configuration 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 servers, use this entry instead:

```
pseudo-device    lta 32
```

The number you specify must be a multiple of 16, up to a maximum of 256.

When you add entries to the configuration file, rebuild the kernel to make the additions take effect. You can rebuild the kernel manually or by using the `doconfig`(8) 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:

```
# MAKEDEV lta1
```

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

You can produce a listing of which devices in `/dev` are the LAT devices (special files) by typing:

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

Note that all `tty` special files with a major number of 39 are for the LAT. A typical LAT special file line looks like this:

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

2.2.3 Edit the etc/ttys File

After you have created the new LAT special files, you need to add entries for those devices to the `/etc/ttys` file.

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 etc/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 add an entry to the `/etc/rc.local` file for the specific hardware interface that you have. This entry must be positioned in the file so that it is executed before the the `lcp` command in the file is executed.

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, add the following entry to 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, other host-initiated connections, or gateways on your LAT, be sure to study the instructions in Chapters 3 through 5 of this manual. The `lcp` entry for each is different from the entry for terminals.

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 node 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.

Before you set up the printer, it must be installed 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

To match the hardware settings of the printer and the terminal server, you need to determine your printer's character size, flow control, parity, and speed. Refer to your printer documentation for this information.

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 a command like 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 the following:

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 define a terminal server's port settings permanently, use the `DEFINE` command. For example:

```
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

You need 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 24 lines of the test data unless you press the BREAK key 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

You need to specify the name of the server and the name of the port for the printer to your ULTRIX operating system. There are two methods for doing this.

The first method is to specify the names through an `lcp` command when you select a LAT terminal line. The second method is to specify the names through an entry in the `/etc/printcap` file. You cannot use both methods for the same server/port name pair.

3.3.1 Select a LAT Terminal Line

To give a name to the server and the port, you can 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 in to 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
```

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 names of the tty port and the terminal server port must be made in the `/etc/printcap` file.

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, enter the appropriate form of the `lcp` command, for example, one of the following:

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

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

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 following command:

```
# kill -HUP 1
```

3.3.2 Set Up the /etc/printcap File

To give a name to the server and the port, you can make entries in the `/etc/printcap` file instead of using the `lcp` command to name the entries. In the `printcap` entries, the following parameters need to be defined for the LAT printer:

- The `:lp=` parameter, the LAT terminal line used to send data to the printer. For example:

  ```
  :lp=/dev/tty42
  ```

- The `:ts=` parameter, 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`. The parameter would then be:

  ```
  :ts=LAT_08002B0540B7:
  ```

- The `:op=` parameter, 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`. The parameter would then be:

  ```
  :op=PORT_7:
  ```

- The `:os=` parameter, the service name (supported on some terminal servers).

Following are four sample `/etc/printcap` entries for terminal server ports. Note that they contain `ts=` and `op=` parameters 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.

```
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:
```

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 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 system to associate a named port on a named terminal server with a specific tty device special file. As a result, users can develop 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 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.

Set the ttys being used for the host-initiated connections to `off nomodem` in the `/etc/ttys` file. If necessary, issue the `kill` command to make the changes you have made to `/etc/ttys` take effect:

```
# 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. See Section 3.1 for instructions.

4.2 The Program Interface

Applications developed to employ host-initiated connections 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 and 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 suggestions 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 the Telnet protocol, without having to log in first to a local ULTRIX system.

For example, a user traveling on business could use a terminal on a LAN to connect through Telnet to her home system and account, even though she does 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:

1. Edit the `ttys` file.
2. Start up the gateway.
3. Edit the `rc.local` file.
4. Connect to the gateway.

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 make the changes take effect:

```
# kill -HUP 1
```

5.1.2 Start Up the Gateway

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

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

The name and description of the default (login) LAT service must have been defined 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, enter the `CONNECT` command. For example, to connect to remote node named remote using a local node named LOCAL as a gateway, type:

```
Local> CONNECT TELNET NODE LOCAL DEST REMOTE
```

Alternatively, 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 the service 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

The `lcp` command allows 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 protocols without having to log in first to an ULTRIX system.

You can also write your own specialized applications and have them advertised to terminal services.

6.1 Programming the Service

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

Example 6-1: Simple Service Program

```
/*
 * 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 and 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
 *              and  15 you would need to edit /etc/ttys and change the
 *              lines for tty14 and 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 date:/dev/ddt14,/dev/tty15 \
 *                 -V "HOSTNAME" \
 *                 -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
```

Example 6-1: (continued)

```
#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);
}
```

A sample program that can be used to replace `getty` for a LAT/Telnet gateway is provided in Appendix C.

Other coding suggestions and restrictions are included in Appendix C.

6.2 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:

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

6.3 Using the Service

To use the service at a LAT terminal, issue the CONNECT command. For example:

```
Local> CONNECT DATE
```

Remote Node Maintenance A

This appendix discusses the control functions of ULTRIX host nodes and the following remote node maintenance functions:

- Obtaining a processor's hardware Ethernet address
- Downline-loading a terminal server
- Upline-dumping memory
- Enabling remote console capabilities

Because it does not have a mass storage device, a terminal server must downline-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 upline-dumps 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 software. An example of a target node is a DECserver 200.

A.1 Control Functions of ULTRIX Host Nodes

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

- Downline load bootstrap loaders and operating system images to a remote node such as a terminal server
- Receive an upline 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.

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 downline-load request using the `load` and `trigger` commands at a command node. These commands cause a target node to issue a downline load request.

  There are no commands that you can use to initiate an upline dump; only target nodes initiate upline dump requests.

- Host node

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

- Target node

  The target node receives the bootstrap loaders and the system image file. Target nodes issue downline-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 upline-dump request.

A.2 Obtaining the Hardware Ethernet Address

There are three ways to obtain the hardware Ethernet address for a processor: using the DECnet `ncp` program, using the ULTRIX `arp` command, and issuing commands 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
```

The symbol *dev-c* is the circuit ID, as specified in Section 2.1.5.3.

A.2.2 Using the arp Command

You can use the `arp` command to find the Ethernet address for a processor in the Address Resolution Protocol (ARP) table. For example, if the host name is `bangor`, type:

```
# arp bangor
```

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

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

If no entry exists for the processor's address, use the `rsh` command to resolve the entry. For example:

```
# rsh bangor who
```

Even if this command fails, it fills the ARP table entry for the processor's Ethernet address.

A.2.3 Using Console Mode (VAX Processors Only)

If the `ncp` and `arp` commands fail, you can use the console mode of VAX processors to obtain the Ethernet address. 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 sequence 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 `t50` 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 sequence at the console mode prompt:

```
>>> show ether
```

The system prints the hardware Ethernet address:

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

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

A.3 Downline Loading

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

A.3.1 Prerequisites for Downline Loading

Before attempting a downline-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 host system, because a host uses circuit-level access to load a target node.

- If the load is operator-initiated, 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 system. When `mop_mom` receives a program load request, it forks and executes the loader, `/usr/lib/dnet/mop_dumpload`, which performs the downline 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 database.

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

A.3.2 Operator-Initiated Downline Loads

You can initiate a downline 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 database 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 database with the `addnode` command. For security reasons, you can choose not to include a target node's service password in the database. 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 downline-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 database 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 downline 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 database 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 database with the addnode command. For security reasons, you can choose not to include a target node's service password in the database. 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 downline load using the trigger command:

1. Use the addnode command to define the required information in the nodes database for the target node, as described in Section A.3.2.2.
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 database, as done by the addnode command. 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 command exits, and auburn continues to carry out the procedure for a target-initiated downline load.

A.3.3 Target-Initiated Downline Loads

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

A target-initiated downline 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 database 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 downline-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 A.3.4) continues with the designated host node performing the downline load.

A.3.4 Downline 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 database.

Note

For a DECserver 200 terminal server, the first and only program load request message is for the system load file.

A.4 Upline Dumping of Memory

You can include certain parameters in the nodes database that allow a specified Ethernet target node to dump its memory into a file on your ULTRIX system. This procedure is called upline dumping. Upline dumping is a valuable tool for crash analysis.

When a target node that is capable of upline dumping detects an impending system failure, that system requests an upline dump.

Note

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

Upline dumping, unlike downline loading, is always initiated by the target node. There are no commands to initiate an upline dump. The ULTRIX system uses the Maintenance Operation Protocol (MOP) to perform an upline dump.

A.4.1 Prerequisites to Upline Dumping

Before attempting an upline dump operation, you must ensure that the nodes, lines, and circuits involved in the upline 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 system. When `mop_mom` receives an upline 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 database, and it must be able to create this file. The dump file is defined with the `addnode` command.

A.4.2 Upline Dump Sequence

The following steps outline the upline 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 downline-loaded it. If the host node is available, the upline 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 upline dump device type at the target node.

   Each node on the Ethernet checks its nodes database to determine if it can accept an upline 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 database 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 upline dump is completed, the host node sends a dump complete message to the target node and attempts to downline-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 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 your server products.

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 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.

Before you can use the `ccr` command, these requirements must be met:

- The host node (your ULTRIX system) and the remote node must be on the same Ethernet.
- The nodes database 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 database with the `addnode` command.

You issue the `ccr` command as follows:

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

This command connects to the remote console server on `dallas`. While in console carrier mode, you can press CTRL/B which operates as a break command that gets the attention of the console online debugging tool (ODT). 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 your system manager to determine the password.

Program for Host-Initiated Connections B

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

B.1 Program Listing

You can find the `dial.c` program in the following directory:

```
/usr/examples/lat
```

```
/*
 * 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();

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);
}

/*
 * 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);

    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;
            }
        }
     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);
}

/*
 *        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);
}

/*
 *        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;

        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, `latdlogin.c`, that can replace the `/usr/etc/getty` program for each tty to be used as a LAT/Telnet gateway in a local network. You can find `latdlogin.c` in the following directory:

`/usr/examples/lat`

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 latdlogin:/dev/tty7,/dev/tty8 \
    -V "nodnam lat service" \
    -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
```

In the code, note the use of the `ioctl` LIOCTTYI. 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 specified a destination string in the request to connect to the service, that string is returned in structure member `lta_dest_port`.

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, including the login prompt from the remote node REMOTE:

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

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

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

C.3 Program Listing

The following is the `latdlogin.c` program:

```
/*
 * 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>

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);
    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

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Guide to Ethernet
Communications Servers
AA-ME98B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

What do you like best about this manual? _____

What do you like least about this manual? _____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
Guide to Ethernet
Communications Servers
AA-ME98B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line