ULTRIX

digital

Guide to System Exercisers

Order Number: AA–ME96B–TE

ULTRIX

Guide to System Exercisers

Order Number: AA-ME96B-TE
June 1990

Product Version: ULTRIX Operating System, Version 4.0 or higher

This manual describes how to test your system using the system exercisers for file systems, memory, and peripherals.

digital equipment corporation
maynard, massachusetts

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause of DFARS 252.227-7013.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

digital
CDA
DDIF
DDIS
DEC
DECnet
DECstation
DECUS
DECwindows
DTIF
MASSBUS
MicroVAX
Q-bus
ULTRIX
ULTRIX Mail Connection
ULTRIX Worksystem Software
VAX
VAXstation
VMS
VMS/ULTRIX Connection
VT
XUI

IBM is a registered trademark of International Business Machines Corporation.

UNIX is a registered trademark of AT&T in the USA and other countries.

MICOM is a registered trademark of MICOM Systems, Inc.

Contents

About This Manual

1 Testing Your System with Exercisers

2 Exercising File Systems and Memory

3 Exercising Peripherals

About This Manual

The objective of this guide is to provide you with information on system exercisers. It describes how to run system exercisers and explains the exercisers you use for file systems and memory, peripherals, and network and communications.

Audience

The *Guide to System Exercisers* is written for the person responsible for managing and maintaining an ULTRIX operating system. It assumes that this individual is familiar with ULTRIX commands, the system configuration, and the system's controller/drive unit number assignments. You do not need to be a programmer to use this guide; however, you should be familiar with an ULTRIX editor, such as `vi` or `ed`. You should also be aware of the naming conventions used in this book.

Organization

This manual consists of three chapters and an index. The chapters are:

Chapter 1: Running System Exercisers
Provides an overview of system exercisers. It describes how to run system exercisers, how to exercise more than one part of the system simultaneously, and exerciser diagnostics.

Chapter 2: Exercising File Systems and Memory
Describes the exercisers that test file systems, system memory, and shared memory.

Chapter 3: Exercising Peripherals
Describes the exercisers that test disk drives, tape drives, and line printers.

Chapter 4: Exercising Network and Communications Systems
Describes the exercisers that test TCP/IP and terminal communications.

Related Documents

This document should be used in conjunction with the *ULTRIX Reference Pages*.

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.

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.

rlogin

In syntax descriptions and function definitions, this typeface is used to indicate terms that you must type exactly as shown.

. . .

In syntax descriptions and function definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.

.
.
.

A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.

`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.

RETURN

This symbol is used in examples to indicate that you must press the named key on the keyboard.

CTRL/*x*

This symbol is used in examples to indicate that you must hold down the CTRL key while pressing the key *x* that follows the slash. When you use this key combination, the system sometimes echoes the resulting character, using a circumflex (^) to represent the CTRL key (for example, ^C for CTRL/C). Sometimes the sequence is not echoed.

ESC X

This symbol is used in examples to indicate that you must press the first named key and then press the second named key. In text, this combination is indicated as ESC-X.

New and Revised Information

In Chapter 2, the description of the m option to shmx, the shared memory exerciser, now lists the location of the SMMAX system parameter for both VAX and RISC computers.

Chapter 3 includes a description of the new tape drive exerciser, tapex.

In Chapter 4, the description of exercising the terminal communication system includes an expanded list of letters that can be used as part of a pseudodevice name. Letters p through z can now be used.

Testing Your System with Exercisers 1

This chapter describes how to use system exercisers as a troubleshooting tool for your ULTRIX operating system. The exercisers are commands residing in the /usr/field directory that allow you to test all or part of your system by exercising specified parts. Note that to run the exercisers you must be logged in as superuser.

The system exercisers test the following system areas:

- System memory and file systems
- Peripherals
- Network and communications systems

Specific exercisers for each area are described in the following chapters.

Each of the exercisers has an on-line help facility that prints a description of how to use the exerciser. To access on-line help, use the -h option. For example, to access help for the dskx exerciser type:

```
# dskx -h
```

For additional information on each of the exercisers, see the *ULTRIX Reference Pages.*

Note

> If you are running exercisers in a diskless operating system environment, you must either use the -o option of the exerciser, or move the /usr/field directory to a writable area. Because the netx exerciser does not have the -o option, you must copy netx to a writeable area before exercising the TCP/IP network.
>
> By default, the /usr file system is mounted read-only in a diskless environment. Because some of the system exercisers in /usr/field require their log files to be local to the exerciser, the read-only restriction prevents the creation of the client's log file, causing that test to fail.
>
> To avoid this problem, copy the desired exerciser and its associated files to /var/tmp before executing them.

1.1 Running System Exercisers

To run any of the exercisers, you must be logged in as superuser and /usr/field must be your current directory. For example:

```
% su
password:
# cd /usr/field
```

All of the examples in this guide assume that you are logged in as the superuser and that your current directory is /usr/field.

All of the system exercisers, except netx, have the -o option. The -o option allows you to specify a file where diagnostic output is saved when the exerciser terminates.

All of the exercisers can be run in either the foreground or the background and can be canceled at any time by pressing CTRL/C in the foreground. You can run more than one exerciser at the same time. Keep in mind, however, that the more processes you have running, the slower the system will perform. Thus, before exercising the system extensively, make sure there are no other users on the system.

Section 1.2 describes how to run multiple exercisers using the syscript utility, and Section 1.3 describes exerciser log files.

1.2 Exercising More Than One Part of the System

To exercise more than one part of the system simultaneously, use the syscript maintenance command. The syscript command asks you which exercisers you want to run, how long you want to run each exerciser, and how many exercisers you want to run at one time. The syscript command allows you to exercise a device, a subsystem, or the entire system. For more information, refer to syscript(8) in the *ULTRIX Reference Pages.*

You can start each exerciser either manually, by specifying the time parameter (-t option) and by placing each command in the background before executing the next command, or you can use the syscript command. If you are using the syscript command, type:

```
# syscript
```

Once the syscript command is running, answer the questions that it asks. The syscript command then executes the individual exercisers and creates a file called testsuite, which contains all of the answers you entered when you first executed syscript. You can reexecute the commands in the testsuite file by entering:

```
# sh testsuite
```

This causes testsuite to execute using the original commands and parameters that you entered.

1.3 Using Exerciser Diagnostics

When an exerciser is halted (by either CTRL/C or timing out), it prints diagnostics on the terminal. These diagnostics are also stored in the exerciser's latest logfile. The diagnostics inform you of the tests' results.

Each time an exerciser is invoked, a new logfile is generated in the /usr/field directory. For example, when you execute the cmx command for the first time, a logfile named #LOG_CMX_01 is created. The logfiles are records of each exerciser's results and consist of the starting and stopping times, and of error and statistical information. The starting and stopping times of exercisers are also logged into the system errorlog file, /usr/adm/syserr/syserr.hostname. This file also contains information on errors reported by the device drivers or by the system.

The logfiles are especially useful if you have a video display terminal, because they provide a record of the diagnostics. However, after reading a logfile you should delete it, as no more than nine logfiles can accumulate for each exerciser. If you attempt to run an exerciser when it has already accumulated nine logfiles, the exerciser tells you to remove some of the old logfiles so that it can create a new one.

If the exercisers find errors, you can determine which device or area of the system has the difficulty by looking at the system errorlog file `/usr/adm/syserr/syserr.hostname`, using the appropriate `uerf`(8) command. For more information on the error logger, see the *Guide to the Error Logger System.*

For the meanings of the error numbers and signal numbers, see `intro`(2) and `sigvec`(2) in the *ULTRIX Reference Pages.*

The following chapters describe each of the system exercisers in detail.

Exercising File Systems and Memory 2

This chapter describes the system exercisers that test local file systems, system memory, and shared memory.

2.1 Exercising the Local File System

Use the `fsx` command to exercise the file systems locally. The `fsx` command exercises the specified file system by initiating multiple processes, each of which creates, writes, closes, opens, reads, validates, and unlinks a test file of random data. For more information on the `fsx` command, refer to `fsx`(8) in the *ULTRIX Reference Pages.*

Caution

Do not run `fsx` on a file system that is on the same partition that `dskx` is currently read/write exercising. The `dskx` exerciser will be overwriting the file system that the `fsx` exerciser is supposed to be using.

Do not test remotely mounted (NFS) file systems with `fsx`.

The format of the `fsx` command is:

fsx –h –o*file* –p*n* –f*path* –t*min*

–h Prints the help message for the `fsx` command.

–o*file* Saves the output diagnostics in *file.*

–p*n* Specifies the number of processes you want `fsx` to initiate. The maximum number of processes is 250. If you do not specify the `-p` option, the default of 20 is assumed.

–f*path* Specifies the pathname of the directory of the file system to test, for example, `-f/usr` or `-f/mnt`. If you do not specify the `-f` option, the default is `/usr/field`.

–t*min* Specifies how many minutes you want the `fsx` command to exercise the file system. If you do not specify the `-t` option, the `fsx` command runs until you terminate it by pressing CTRL/C in the foreground.

The following is an example of the `fsx` command:

```
# fsx -p5 -f/usr -t60 &
```

The example shows how to test the `/usr` file system with five processes running for 60 minutes in the background.

2.2 Exercising the System Memory

Use the memx command to exercise the system memory. The memx command exercises the system memory by initiating multiple processes. By default, the size of each process is defined as the total system memory in bytes divided by 20. The minimum allowable number of bytes per process is 4095. The memx command runs ones and zeros, zeros and ones, and random data patterns in the allocated memory being tested.

If you are running memx in a diskless environment, you must copy the memx file and its associated files from /usr/field to a writeable area. The files that you need to run the memx exerciser include the following:

- memx
- memxr
- shmx
- shmxb

For more information refer to memx(8) in the *ULTRIX Reference Pages.*

Note

The memx command is restricted by the amount of available swap space. The size of the swap space and the internal memory available determine how many processes can run on your system. For example, if there are 16 Mbytes of swap space and 16 Mbytes of memory, all the swap space would be used if all 20 (the default) initiated processes were running simultaneously. This would prevent any other process from executing.

Therefore, on systems with large amounts of memory and small amounts of swap space, you must use the -p or the -m options, or both, to restrict the number of memx processes or to restrict the size of the memory being tested.

The format of the memx command is:

memx –h –o*file* –s –m*n* –p*x* –t*min*

–h Prints the help message for the memx command.

–o*file* Saves the output diagnostics in *file.*

–s Disables automatic invocation of shmx, which is performed by default.

–m*n* Specifies the amount of memory in bytes (*n*) for each process to test. The default is the total amount of memory divided by 20, with a minimum size of 4095 bytes.

–p*x* Specifies the number of processes to initiate. The maximum number is 20, which is also the default.

–t*min* Specifies how many minutes you want the memx command to exercise the memory. If you do not specify the -t option, the memx command runs until you terminate it by pressing CTRL/C in foreground.

The following is an example of the memx command:

```
# memx -s -m4095 -p5 -t60 &
```

This example disables automatic execution of the shared memory exerciser, shmx, which is performed by default. It also initiates five processes that test 4095 bytes of memory, and it runs in the background for 60 minutes.

2.3 Exercising Shared Memory

Use the shmx command to exercise shared memory. The shmx command spawns a background process called shmxb, and together shmx and shmxb exercise the shared memory segments. They take turns writing and reading each other's data in the segments.

Using shmx, you can test the number and the size of memory segments and shmxb processes. The shmx exerciser runs until the process receives either a CTRL/C or a kill -15 *pid* command in foreground. For more information on the kill command refer to kill(1) in the *ULTRIX Reference Pages.*

You automatically invoke shmx when you start memx. You can also run shmx by itself. The format of the shmx command is:

shmx –h –o*file* –t*i* –m*j* –s*k* –v

–h Prints the help message for the shmx command.

–o*file* Saves diagnostic output in *file.*

–v Uses the fork system call instead of the vfork system call to spawn shmxb.

–t*i* Indicates the run time in minutes (*i*). The default is to run until the process receives either a CTRL/C or a kill-15 *pid* command in foreground.

–m*j* Specifies the memory segment size in bytes (*j*) to be tested by the processes; *j* must be greater than 0. The default is SMMAX/6. (SMMAX is a system parameter. On a VAX computer, SMMAX is set in the file /sys/conf/vax/param.c. On a RISC computer, it is set in the file /sys/conf/mips/param.c.)

–s*k* Specifies the number of memory segments (*k*). The default is 6. The maximum is also 6.

The following example tests six memory segments (default), each with a segment size of SMMAX/6. The test terminates when you either press CTRL/C or issue the command kill-15 *in the foreground.*

```
# shmx
```

Here is another example of the shmx command:

```
# shmx -t180 -m100000 -s3 &
```

This example runs three memory segments of 100,000 bytes for 180 minutes in the background.

Exercising Peripherals 3

This chapter describes the system exercisers that you use to test the following peripherals:

- Disk drives
- Tape drives
- Line printers

3.1 Exercising the Disk Drives

Use the `dskx` maintenance command to exercise the disk drives. The `dskx` maintenance command exercises specified partitions and file systems on the designated disk.

First, `dskx` checks to see if the specified file system is on the disk. If the file system exists and if you are using either the `-p` or `-c` options, `dskx` asks you if you want to continue or stop the program. These options overwrite data on the disk, destroying any file systems on the specified partition or device.

Caution

> The `-p` and `-c` options destroy data on the device you are testing. Use extreme caution when using either of these options.
>
> When using the `-p` option, ensure that the partition you are exercising does not overlap other partitions, because you will inadvertently destroy data on another partition. Use the `chpt` maintenance command with the `-q` option to see which partitions overlap on the device you are testing.

For more information on the `dskx` command, refer to `dskx`(8) in the *ULTRIX Reference Pages*.

The format of the `dskx` command is:

```
dskx –h –ofile –pdevpart –cdev –rdev –tmin –dn
```

–h Prints help message for the `dskx` command.

–o*file* Saves output diagnostics in *file*.

–p*devpart* Performs random seeks, writes, and reads on the specified partition (*part*) of the device (*dev*). Next, it validates the random data and block sizes. The *part* specifies the partition on the device and can be `a` through `h`.

You cannot test the `c` partition, because the test would corrupt the bad block information.

dev Specifies the raw or buffered device name and number such as rra2 for an RA device or hp0 for an RM or an RP device.

–c*dev* Performs random seeks, writes, and reads on all the partitions of the specified device (*dev*), except for the `c` partition. Partition `c` is not tested, because the test would corrupt the bad block information.

–r*dev* Performs random seeks and reads on all partitions of the device specified (*dev*), except for partition `c`. The `-r` option is safe to use on any disk containing file systems, because it will not overwrite data.

–t*min* Specifies how many minutes you want the `dskx` command to exercise the disk. If you do not specify the `-t` option, the `dskx` command runs until you terminate it by pressing CTRL/C in the foreground.

–d*n* Specifies in minutes how often you want the `dskx` command to print diagnostics to the terminal. The default is to print diagnostics upon completion of the exercise.

The following are examples of the `dskx` command:

```
# dskx -rra0 -t20 -d5 &
# dskx -pra1g -t60 -d10 &
```

The first example tests the first RA disk on the system for 20 minutes in the background. Diagnostics are printed to the terminal every five minutes. Note that this example shows how to seek and read the device; no file systems are overwritten.

The second example tests the `g` partition of the `ra1` device for 60 minutes. It runs in the background and writes diagnostics to the terminal every 10 minutes.

3.2 Exercising the Tape Drives

There are two tape drive exercisers, `mtx` and `tapex`. Both the `mtx` and `tapex` commands write, read, and validate random data on a tape device from the beginning of the tape (BOT) to the end of the tape (EOT). The `tapex` command performs additional tests, for example, positioning tests for records and files and tape transportability tests. For more information on `mtx` and `tapex`, refer to `mtx`(8) and `tapex`(8) in the *ULTRIX Reference Pages*.

3.2.1 Using mtx

The format of the `mtx` command is:

mtx –h –o*file* –r*n* –f*n* –s*dev#* –l*dev#* –v*dev#* –a*dev#* –t*min*

–h Prints help message for the `mtx` command.

–o*file* Saves diagnostic output in *file* .

–r*n* Specifies the record length in bytes for the long record exercise. The range of *n* is 100 through 20480 bytes. The default value is 10240 bytes.

–f*n* Specifies the length of the file in numbers of records. The variable *n* is the number of records. The default value is EOT (–1).

–s*dev#* Performs a short record test that writes, reads, and validates 512-byte records (the default) on device *dev*. The variable *dev#* is the raw device name and number, such as rmt0h for a high-density rewind tape device, or nrmt0l for a low-density no-rewind tape device.

–l*dev#* Performs a long record test that writes, reads, and validates 10240-byte records (the default) on device *dev#*. The variable *dev#* is the raw device name and number, such as rmt0h for a high-density rewind tape device, or nrmt0l for a low-density no-rewind tape device.

–v*dev#* Performs a variable record length test that writes, reads, and validates random record lengths from 512 to 20280 bytes on device *dev#*. The variable *dev#* is the raw device name and number, such as rmt0h for a high-density rewind tape device, or nrmt0l for a low-density no-rewind tape device.

–a*dev#* Performs short, long, and variable record length tests on device *dev#*. The variable *dev#* is the raw device name and number, such as rmt0h for a high-density rewind tape device, or nrmt0l for a low-density no-rewind tape device.

–t*min* Specifies how many minutes you want the `mtx` command to exercise the tape drive. If you do not specify the `-t` option, the `mtx` command will run until you terminate it by pressing CTRL/C in foreground.

The following is an example of the `mtx` command:

```
# mtx -r20480 -lrmt0h -t60 &
```

This example shows how to perform a long record test on the tape device rmt0h. The record length is 20480 bytes and the test runs for 60 minutes in the background.

3.2.2 Using tapex

Some `tapex` options cause specific tests to be performed, for example, an end-of-media test. Other options modify the tests, for example, by enabling caching. The format of the `tapex` command is:

tapex *–options –parameters*

The options are as follows:

–a Performance measurement test that calculates the tape transfer bandwidth for writes and reads to the tape by timing data transfers.

–b Continuously runs the write/read tests until the process is killed. This flag can be used in conjunction with the `-r` or `-g` flag.

–c Enables caching on the device, where supported. This does not specifically test caching, but it enables the use of caching on a tape device while running the other tests.

–C Disables caching on TMSCP tape devices. If the tape device is a TMSCP unit, then caching is the default mode of test operation. This flag causes the tests to be run in noncaching mode.

–d Tests the ability to append to the media. First, the test writes records to the tape. Then, it repositions back one record and appends additional records.

Finally, the test does a read verification. This test simulates the behavior of the `tar r` switch.

–e End-of-media test. This test first writes data to fill up a tape, which may take a long time for long tapes. It then does reads and writes past the end of media, which should fail. Next, it enables writing past end of media, writes to the tape, and reads back the records for validation.

–E Runs an extensive series of tests in sequential order. Due to the large number of tests, this option takes a long time to complete. Depending on tape type and cpu type, this series of tests can take up to 10 hours to complete.

–f /dev/rmt#?

Specifies the name of the device special file that corresponds to the tape unit being tested. The number sign (#) symbol represents the unit number. The question mark (?) can be the letter h for the high density device or l for the low density device. The default tape device is `/dev/rmt0h`.

–F File-positioning tests. First, files are written to the tape and verified. Next, every other file on the tape is read. Then, the previously unread files are read by traversing the tape backwards. Finally, random numbers are generated; the tape is positioned to those locations, and the data is verified. Each file uses a different record size.

–G File-positioning tests on already-written tape. This flag can be used in conjunction with the `-F` flag to run the file position tests on a tape that has already been written to by a previous version of the `-F` test. For this to work, the same test parameters, for example record size and number of files, must be used as when the the tape was written. No other data should have been written to the tape since the previous `-F` test.

–g Random record-size tests. This test writes records of random sizes. It reads in the tape, specifying a large read size; however, only the amount of data in the randomly-sized record should be returned. This test only checks return values and does not validate record contents.

–h Displays a help message describing the tape exerciser.

–i Interactive mode. Under this mode, the user is prompted for various test parameters. Typical parameters include the record size and the number of records to write. The following scaling factors are allowed:

k or K	for kilobyte (1024 * n)
b or B	for block (512 * n)
m or M	for megabyte (1024 * 1024 * n)

For example, 10k would specify 10240 bytes.

–j Write phase of the tape-transportability tests. This test writes a number of files to the tape, and then verifies the tape. After a successful verification, the tape is brought off line to be moved to another tape unit and read in with the `-k` option. The purpose of this test is to prove that a tape can be written on one drive and read on another drive. The test parameters for the `-k` phase of the transportability test must match the parameters of the `-j` test. Any changes of test parameters from the defaults should also be changed during the `-k` test.

–k Read phase of the tape-transportability tests. This test reads a tape that was written by the `-j` test and verifies that the expected data is read from the tape. Success of this test proves that a tape can be written on one drive and read on another. As stated in the description of the `-j` option, any parameters changed in the `-j` test must also be changed in the `-k` test.

–L Media loader test. For sequential stack loaders, the media is loaded, written to, and verified. Then, the media is unloaded, and the test repeats on the next piece of media. This verifies that all the media in the input deck is writable. To run this test in read-only mode, also specify the `-w` option.

–l End-of-file test. This test verifies that a zero byte count is returned when a tape mark is read and that another read will fetch the first record of the next tape file.

–m Displays tape contents. This is not a test; it reads the tape sequentially and prints out the number of files on the tape, the number of records in each file, and the size of the records within the file. The contents of the tape records are not examined.

–N Disables the usage of n-buffered I/O on tests that support its usage. (See `nbuf`(4) for a description of n-buffered I/O.)

–o Sends output to the specified filename. The default is to not create an output file and send output to the terminal.

–p Runs both the record-positioning and file-positioning tests. (See the `-R` and `-F` options.)

–q Command timeout test. This test verifies that the driver allows enough time for completion of long operations. First, the test writes files to fill up the tape. Next, a rewind is performed, followed by a forward skip to the last file. This test is successful if the forward skip operation completes without error.

–r Record-size test. A number of records are written to the tape and then verified. This process is repeated over a range of record sizes.

–R Record-positioning test. First, records are written to the tape and verified. Next, every other record on the tape is read. Then the other records are read by traversing the tape backwards. Finally, random numbers are generated; the tape is positioned to those locations, and the data is verified.

–s Record-size behavior test. Verifies that a record read will return at most one record or the read size, whichever is less.

–S Single record size test. This test modifies the record-size test (-r option) to use a single record size.

–T Copies output to standard output. This flag is useful if you want to log output to a file with the -o option and also have the output displayed on standard output. This flag must be specified after the -o flag in the command line.

–v Verbose mode. This option causes more detailed terminal output of what the tape exerciser is doing. For example, it lists operations the exerciser is performing, such as record counts, and more detailed error information.

–V Very verbose mode. This option causes more output to be generated than either the default mode or the -v flag. The output consists of additional status information on exerciser operation.

–w Opens the tape as read-only. This mode is only useful for tests that do not write to the media. For example, it allows the -m test to be run on a write-protected media.

–Z Initializes read buffer to the nonzero value 0130. This can be useful for debugging purposes. If the -Z flag is not specified, all elements of the read buffer will be initialized to 0. Many of the tests first initialize their read buffer and then perform the read operation. After reading a record from the tape, some tests validate that the unused portions of the read buffer remain at the value to which they were initialized. As a debugging tool, it may in some cases be useful to have this initialized value set to be nonzero. In those cases, the arbitrary character 0130 can be used.

You can change the default test parameters either by using the -i option described previously or by specifying the parameters in the command line. To specify a value, type the parameter name followed by a space and then the number. For example, the parameter -min_rs 512 specifies a minimum record size of 512 blocks. The following scaling factors are allowed:

k or K for kilobyte (1024 * n)

b or B for block (512 * n)

m or M for megabyte (1024 * 1024 * n)

For example, 10K would specify 10240 bytes.

These parameters are associated with the option -a:

–perf_num The number of records to write and read.

–perf_rs The size of records.

These parameters are associated with the option -d:

–tar_num The number of additional and appended records.

–tar_size The record size for all records written in this test.

These parameters are associated with the option -e.

Note that specifying too much data to be written past EOM could cause a reel-to-reel tape to go off the end.

–end_num The number or records written past EOM.

–end_rs The record size.

These parameters are associated with the option -F:

–num_fi The number of files.

–pos_ra The number of random repositions.

–pos_rs The record size.

–rec_fi The number of records per file.

This parameter is associated with the option -g:

–rand_num The number of records to write and read.

These parameters are associated with the options -j and -k:

–tran_file The number of files to write or read.

–tran_rec The number of records contained in each file.

–tran_rs The size of each record.

These parameters are associated with the option -R:

–pos_num The number of records.

–pos_ra The number of random repositions.

–pos_rs The record size.

These parameters are associated with the options -r and -S:

–inc The record increment factor.

–max_rs The maximum record size.

–min_rs The minimum record size.

–num_rec The number of records.

–t Sets a time limit in minutes on how long to run the record-size test (the -r option). The default is to run the test to completion.

These parameters are associated with the option `-s`:

–num_rec The number of records.

–size_rec The record size.

This parameter is used in any test which supports n-buffered I/O:

–num_nbuf The number of buffers to use.

This parameter is associated with all tests:

–err_lines The threshold on error printouts.

The following example runs an extensive series of tests on tape device rmtlh and sends all output to a file called `tapex.out`.

```
#  tapex -f /dev/rmtlh -E -o tapex.out
```

The next example performs random record-size tests and uses verbose mode in sending back output. By default, this test runs on the default tape device /dev/rmt0h and output is sent to the terminal.

```
#  tapex -g -v
```

This example performs read/write record testing using record sizes in the range 10k to a maximum record size of 20k. By default, this test runs on the default tape device /dev/rmt0h and output is sent to the terminal.

```
#  tapex -r -min_rs 10k -max_rs 20k
```

3.3 Exercising the Lineprinters

Use the `lpx` maintenance command to exercise the lineprinters. The `lpx` command exercises the line printers by printing a rolling character pattern repeatedly to the device. If the device is offline, `lpx` attempts to start the device every 60 seconds until it succeeds or until you terminate it. Once the line printer starts, the `lpx` command prints five pages of a rolling character pattern, pauses for 15 minutes, and then repeats the printing and pausing sequence until the the command terminates. For more information refer to `lpx`(8) in the ULTRIX Reference Pages.

Note

> To prevent other jobs from interfering with the test, you should disable the line printer queue of the printer you are testing before using the `lpx`. You may need to look in the `/etc/printcap` file to determine the spool queue and then use the line printer control command `lpc` to accomplish this task.

The format of the `lpx` command is:

lpx –h –o*file* –p*n* –d*dev* –t*min*

–h Prints help message for the `lpx` command.

–o*file* Saves diagnostic output in *file*.

–p*n* Specifies the pause period in *n* minutes. The default pause period is 15 minutes and the shortest pause can be zero (0) minutes.

–d*dev* Specifies the line printer you want to test such as lp or lp1. The device names are listed in the /dev directory.

–t*min* Specifies how many minutes you want the lpx command to exercise the line printer. If you do not specify the -t option, the lpx command runs until you terminate it by pressing CTRL/C in foreground.

The following is an example of the lpx command:

```
# lpx -t60 -dlp1 &
```

This example shows how to exercise the lp1 lineprinter for 60 minutes in the background.

Exercising Network and Communications Systems 4

This chapter explains the system exercisers used to test the following network and communications systems:

- TCP/IP
- Terminal communications

4.1 Exercising the TCP/IP Network

Use the netx maintenance command to exercise the TCP/IP network. The netx command sets up a stream socket connection with netx acting as the client and the miscd utility acting as the server in the TCP/IP internet domain. Using the connection, the netx command writes random data to the miscd server. The server loops the data back to netx, and then the data is read and verified against the original data. For more information on the netx command, refer to netx(8) in the *ULTRIX Reference Pages.*

The netx command uses the port number of the echo TCP service in the /etc/services file. The echo TCP service must be enabled in the /etc/inetd.conf file. The service is enabled if there are no number signs (#) in front of the echo service.

If you remove the number sign (#) from the echo TCP service in the /etc/inetd.conf file, you must start the echo daemon. To do this, first find the PID of inetd, using the ps -ax command. Then, type the command kill -HUP *pid.*

The *pid* argument is the PID of the inetd daemon.

The format of the netx command is:

netx –h –p*n nodename* –t*min*

–h	Prints the help message for the netx command.
–p*n*	Specifies the port number to use in the internet domain. The variable *n* must be less than 32768. By default, netx uses the port associated with the echo service, so you should not have to use this option.
nodename	The name of the remote or local system host running the server.
–t*min*	Specifies how many minutes you want the netx command to exercise the network connection. If you do not specify the -t option, the netx command runs until you terminate it by pressing CTRL/C in foreground.

The following is an example of the netx command:

```
# netx suez -t60 &
```

The example shows how to test the network connection to a remote system host named suez. The test runs for 60 minutes in the background.

4.2 Exercising the Terminal Communication System

Use the cmx maintenance command to exercise the terminal communications system. The cmx command writes, reads, and validates random data and packet lengths on the communications line or lines specified. The lines you exercise must have a loopback connector attached to the distribution panel or the cable. Otherwise, cmx repeatedly prints an error message to the terminal until its time expires or until you press CTRL/C. For more information on the cmx command, refer to cmx(8) in the *ULTRIX Reference Pages.*

Also, the lines you specify must be disabled in the /etc/ttys file. See the *Guide to System Environment Setup* for information about the /etc/ttys file.

The format of the cmx command is:

cmx –h –o*file* –t*min* –l *line-1 line-2 line-n...*

–h Prints help message for the cmx command.

–o*file* Saves output diagnostics in *file.*

–t*min* Specifies how many minutes you want the cmx command to exercise the communications system. If you do not specify the -t option, the cmx command runs until you terminate it by pressing CTRL/C in the foreground.

–l *line-n* Specifies the line or lines you want to test. The values for *line-n* are found in the /dev directory and are the last two characters of the tty device name. For example, if you want to test the communications system for a device named tty02, tty03, and tty14, *line-1* is 02, *line-2* is 03, and *line-3* is 14. Additionally, the *line-n* argument can specify a range of lines to test, such as 00-08.

You cannot test pseudodevice lines or lta device lines. Pseudodevices have a p, q, r, s, t, u, v, w, x, y, or z as the first character after tty, such as ttyp3. The lta devices have a major number of 39.

The following examples show how to use the cmx command. The first example exercises communications lines tty22 and tty34 for 45 minutes in the background. The second example exercises lines tty00 through tty07 until you press CTRL/C.

```
# cmx -l 22 34 -t45 &

# cmx -l 00-07
```

Index

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 System Exercisers
AA-ME96B-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 --

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 System Exercisers
AA-ME96B-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