68xxx UniFLEX® System Manager's Guide

MANUAL REVISION HISTORY

Revision	Date	Change
A	03/87	Original Release, 68xxx UniFLEX System Manager's Guide, for Version 2.1 of 68xxx UniFLEX

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

Contents

Preface xi

Chapter 1 Getting Started

Building an Operating System 1.1

Booting a System 1.5
Automatic-Boot Mode 1.6
Manual-Boot Mode 1.6

Initializations 1.9

Updating a System Disk 1.9

Adding UniFLEX Support Software to Your System 1.10

Updating UniFLEX Support Software 1.10

Chapter 2 The Init-Control File

Introduction 2.1

Executing a Shell Command 2.2

Flags 2.2

Control Commands 2.3

Essential Elements of the Init-Control File 2.5

Error Messages 2.6
Fatal Errors 2.7
Nonfatal Errors 2.8

Chapter 3 Starting the System

Day-to-Day Procedures 3.1
Booting to Single-User Mode 3.1
Booting Directly to Multi-User Mode 3.2

Setting the Date 3.2
Using a Built-in Clock 3.3
Setting the Date Manually 3.3

Message of the Day 3.4

Formatting a Disk 3.5

Contents

Sequestering Bad Blocks 3.6
Formatting with the 'l' or 'L' Option 3.7
The "badblocks" command 3.7

Routine Verification of the Logical Structure of the Medium 3.8

Maintaining Backup Files 3.8

Mounting Devices 3.9

Taking a Memory Dump 3.11

The History File 3.12

Adding New Programs 3.13

The File "/etc/startup" 3.13

Shutting Down the System 3.14
Step One 3.14
Step Two 3.15

Chapter 4 Using a Printer Spooler

Introduction 4.1

Configuring a Printer Spooler 4.1
Selecting a Device 4.1
Using a terminal driver for a serial printer 4.2
Creating a spooler command 4.2
Creating a spooler directory 4.2
Initiating a printer spooler 4.3

Using the Spooler Command 4.4

Shutting Down a Printer Spooler 4.5

Summary of Routine Spooler Use 4.6

Repairing a Damaged Printer Spooler 4.6

Chapter 5 The Password File

Introduction 5.1

Structure of the Password File 5.1
User Name 5.1
Password 5.2
User ID 5.3
Home Directory 5.3
Login Program 5.4

Original Password File 5.4

Adding a User to the System 5.4

Deleting a User 5.5

Chapter 6 Files and Devices

Introduction 6.1

Regular File 6.1

Contiguous File 6.1

Directory 6.2

Pipe 6.2

Device 6.2
Character Device 6.3
Block Device 6.3
Pseudoterminal Device 6.4
Network Devices 6.4

Creating Devices 6.4

Chapter 7 Important Directories

Introduction 7.1

/uniflex 7.1

/.badblocks 7.1

/act 7.2

/bin 7.2

Contents

/dev 7.2

/etc 7.3
/etc/ttylist 7.3
/etc/ttycap 7.5
/etc/termcap 7.5
/etc/.init.control 7.6
/etc/format.control 7.6
/etc/log 7.7
/etc/log/motd 7.7
/etc/log/password 7.7

/gen 7.7
/gen/errors 7.7
/gen/help 7.7
/gen/spooler 7.8

/lib 7.8
/lib/rel20.errs 7.8
/lib/rel68k.errs 7.8
/lib/Syslib68k 7.8
/lib/sysacct 7.9
/lib/sysdef 7.9
/lib/syserrors 7.9
/lib/sysfcntl 7.9
/lib/sysints 7.9
/lib/sysmessages 7.9
/lib/syspty 7.10
/lib/sysrump 7.10
/lib/sysstat 7.10
/lib/systime 7.10
/lib/systty 7.10
/lib/sys68881 7.10
/lib/std_env 7.11

/lost+found 7.11

/tmp 7.11

/usr 7.12

/usr0, /usr1, /usr2, /usr3 7.12

Chapter 8 Errors Fatal to the Operating System

Introduction 8.1

Errors during Initialization 8.1

After Loading the Operating System 8.2

Chapter 9 Fine-tuning the UniFLEX Operating System

Introduction 9.1

Invoking the "tune" Command 9.1
Arguments 9.1
Format for Arguments 9.2
Options Available 9.2
Modes of Operation 9.2
Read-only mode 9.2
Interactive mode 9.3
Automatic mode 9.3

Adjustable Parameters--an Overview 9.4
Functions of the Adjustable Parameters 9.4
Limits and Defaults for Adjustable Parameters 9.6

Parameters Adjustable in Automatic Mode 9.8
System Buffers 9.8
Lists of I/O Characters 9.9
Daylight-Savings-Time Flag 9.9
Last Day of Daylight Savings Time 9.9
First Day of Daylight Savings Time 9.10
Time of Day for Daylight Savings Time 9.10
Maximum Number of Open Files 9.10
Maximum Number of Locked Records 9.11
Maximum Number of Mounted Devices 9.11
Maximum Number of Message Buffers 9.11
Maximum Number of Message Exchanges 9.12
Size of Message Buffers 9.12
Paging Device 9.12
Default Paging Space 9.13
Pipe Device 9.13
Root Device 9.14
Seek Rate of the Floppy Disk Drives 9.14
Maximum Number of Tasks Supported 9.14
Shared-text Programs 9.15
Time Limit for Tasks 9.15
Number of Time-outs 9.15
Setting the Time Zone 9.15
Number of Tasks per User 9.16

Parameters Associated with "phys" Segments 9.16
Physical Address 9.17
Logical Address 9.17
Segment Size 9.17

Parameters Associated with Scheduling 9.17
Functions of the Scheduler 9.17
A Task's Personality 9.18
Determining Priorities 9.19
Max CPU utilization 9.20
CPU utilization increment per hit 9.20
CPU utilization decay 9.20
Determining Duration of Stay in the CPU 9.20
Max quantum 9.21
Quantum increment 9.21

Examples 9.21

Error Messages 9.22

Chapter 10 Repairing a Damaged Disk

Introduction 10.1
Structure of a UniFLEX Disk 10.2
Physical Errors on the Disk 10.3
Limitations of "diskrepair" 10.4

The Command Line 10.4
The 'a' Option 10.5
The 'b' Option 10.6
The 'B' Option 10.6
The 'f' Option 10.6
The 'm' Option 10.6
The 'M' Option 10.7
The 'n' Option 10.7
The 'p' Option 10.7
The 'q' Option 10.7
The 'r' Option 10.8
The 'u' Option 10.8
The 'v' Option 10.8

Preliminary Checks 10.8
Command-line Options 10.9
Specified Device 10.9
Backup Devices 10.10
Permissions 10.10
Unmounting a Mounted Disk 10.10
Checking the Root Device 10.11
Status of the Root Directory 10.12

Calling "blockcheck" 10.12
Abnormal Termination of "blockcheck" 10.13
Improper I/O Redirection 10.13

Preliminary Checks on the SIR 10.14
Accessing the SIR 10.14
Size of Disk 10.14
Fdn Count 10.15
First Block of Paging Space 10.15
First Block of Contiguous-File Space 10.16

The File "/.badblocks" 10.17
Accessing the Bad-Blocks File 10.17
Validating the Bad-Blocks File 10.17
Checking the Size 10.18

Phase 1--Check Allocated Blocks 10.18
File Size 10.18
Noncontiguous files 10.19
Contiguous files 10.19
Out-of-Range Blocks in Fdns 10.19
Blocks Duplicated in Fdns 10.20

Transition between Phase 1 and Phase 2 10.21
Calling "fdncheck" 10.21
Abnormal Termination of "fdncheck" 10.21
The File "/.badblocks" 10.22
Reading the Root Directory 10.22

Phase 2--Scan Directories 10.23
Size of Directory 10.23
Nesting Directories 10.24
Invalid File Name 10.24
File Size 10.25
Out-of-Range Blocks in Files 10.26
Blocks Duplicated in Files 10.27
The Files "." and ".." 10.28
Unknown File Type 10.28
Inactive Fdn 10.29
Out-of-range Fdns 10.29

Phase 3--Check Unreferenced Directories 10.30

Phase 4--Check File and Directory Links 10.32
Unreferenced Files 10.33
Link Count 10.34
In-core Fdn List 10.35

Phase 5--Check Free Lists 10.35
Volume Space 10.36
Missing blocks 10.36
Duplicate blocks 10.36
Out-of-range blocks 10.36
Out-of-range pointers 10.36
In-core block list 10.37

Contents

Summary of the status of the free list 10.37
Rebuilding the free list 10.37
Checking the Contiguous-File Free-List 10.38
Similarities to the free list for the volume space 10.38
Out-of-order free list 10.39
Extraneous data 10.39
Summary of the status of the contiguous-file free-list 10.39
Rebuilding the contiguous-file free-list 10.40

Phase Six--Check SIR Information 10.41
Free Fdn Count 10.41
Free Block Count 10.41
Checking the "Mount Flag" 10.41
State of the Disk 10.42
Updating the SIR 10.43

I/O Errors 10.43

Index of Error Messages 10.46

Chapter 11 Recovering from Problems

Introduction 11.1

After a Crash 11.1

Notes 11.8

Recovering Files Containing I/O Errors 11.11

Miscellaneous Repairs 11.12
Fixing Missing "." and ".." Files 11.12
Expanding the Directory "lost+found" 11.13

Who Owns What? 11.14

Setting the User ID Bit 11.14

Appendix A A Generic Init-Control File

Appendix B Program Interrupts

Index

Preface

This manual provides the system manager with the information necessary for building and booting a UniFLEX operating system, maintaining the system from day to day, updating the system, and recovering from problems that may damage the operating system.

The UniFLEX® Operating System is available in a variety of configurations. Each configuration consists of the kernel of the operating system plus a subset of all the modules that the operating system can support. These modules include features like contiguous files, pseudoterminals, interprocess communication, and a real-time task scheduler. The manual is written to accomodate the most complex system. You may therefore find yourself reading about features which your system does not appear to support. If you have any doubt about whether or not your system supports a particular feature, you can invoke the command

```
info /uniflex
```

which sends to standard output, among other things, a list of the optional features that your system does support.

Preface

Chapter 1

Getting Started

1.1 Building an Operating System

The UniFLEX Operating System is supplied either on a series of floppy disks or on a streaming tape. However, before the software can be of much use to you, you must copy it to your hard disk. This procedure, known as building an operating system, is described in this section. We strongly recommend that you read the entire section before trying to build your system.

When you are ready to build your system, carry out the following steps:

1. Turn on the machine and all associated hardware, such as the console (the terminal used to boot the system). The ROM assumes that the console is configured for 8 data bits, 1 stop bits, and no parity. It assumes a baud rate of 9600 or 19200 (see 68xxx Hardware Setup Notes).

2. If the following message appears on the screen of the console (the terminal used to boot the system)

   ```
   UniFLEX AUTO-BOOT - Hit ^C to Abort
   ```

   type control-C (to type a control character depress the key marked "control" or "ctrl" and hold it down while typing either the upper- or lowercase version of the appropriate character, in this case 'C' or 'c'). You have 30 seconds to type control-C before the system automatically boots. If for some reason you miss the chance to abort from auto-boot mode, a series of error messages will appear on the screen, and you will have to reset the system.

3. The following message appears either immediately after you turn the system on or after you type control-C in response to the previous message:

   ```
   <hardware_description> UniFLEX ROM <release_date>
   ?
   ```

   The question mark, '?', is a prompt from the ROM that indicates it is waiting for your instructions.

4. Insert the disk labeled "System Floppy" in the primary floppy disk drive (see 68xxx UniFLEX Hardware Setup Notes). Do not write protect this disk. When the disk is in place, type the sequence escape-B (type the key labeled "escape" or "esc" followed by an uppercase 'B').

5. In response, the ROM issues the following prompt:

```
$Boot from disk/tape
File name?
```

You should type

```
<dev_name>:uniflex
```

followed by a carriage return. The term <dev_name> is a hardware-specific designation for the device that contains the system disk or tape (see 68xxx UniFLEX Hardware Setup Notes); "uniflex" is the name of the file to load from the root directory of the system disk into main memory. As it loads the file, the ROM displays three messages:

```
Text segment <size> bytes at <address>
Data segment <size> bytes at <address>
Bss segment <size> bytes at <address>
```

The process of loading the system should take no more than a minute. If the system does not appear to be loading and you do not receive any message from the ROM describing the problem, check to be sure that the floppy disk is correctly inserted. On most systems the label on the floppy disk should face the lever on the floppy disk drive.

6. When the ROM finishes loading the file (indicated by the appearance of the ROM prompt ('?') below the message about the bss segment), type

```
control-A
```

This command starts the operating system. When it is ready for further input, it displays the following banner:

```
<hardware_description> UniFLEX Operating System
Copyright (C) [<year_list>] by
Technical Systems Consultants, Inc.

Version <version_number> - Created: <release_date>
Configuration:  <configuration_information>

Total user memory = <memory_after_loading_UniFLEX>
```

The banner is followed by the system prompt, "++", which indicates that the operating system is ready to accept commands.

The amount of memory left after loading the operating system depends on the particular configuration of the computer and the amount of memory installed in it. The operating system typically consumes between 128 and 512 K.

Record the version number and the release date in a safe place. Also record the serial number of the computer and the serial number of your copy of the UniFLEX Operating System (located on the label on the master disk). If you ever need to contact either the manufacturer of your hardware or Technical Systems Consultants, you will need this information.

7. Next, you must use a shell script (see "shell" in 68xxx UniFLEX Utility Commands) to format the hard disk, establish a file system on it, and copy the operating system to it. To execute this shell script type

```
/etc/crdisk[.ST] [<options_list>]<model_spec> 50
```

If you are building your system from a streaming tape, use the optional suffix, ".ST", when you invoke the command. The number 50 tells the operating system to reserve 50 cylinders for paging (you can alter the amount of paging space later if necessary with the "alter_page" command) and <model_spec> provides the operating system with coded information about the particular drive. The optional list of options can consist of any of the single-character options (those that do not take an argument) to the "formatw" command (see 68xxx UniFLEX Utility Commands). If you use the option list, be sure that you do not put a space character between the list and the model specification.

We recommend the 'v' option to check the disk for bad blocks and the 'L' option, which prompts you for the

location of any bad blocks you already know about either from previous experience with the disk or from a list supplied by the manufacturer. Including the 'v' option significantly increases the length of time required to build the system, but it provides a thorough check of the hard disk, which is well worth the extra time when you are building a system.

You may provide the model specification in one of three ways--a lowercase 'm' followed by an equals sign, '=', and a model code; an uppercase 'M'; or an uppercase 'P'. These letters each correspond to an option to the "formatw" command. See the documentation for this command for a complete explanation of these options. Note that your choice here should not be preceded by a plus sign, '+'.

8. When the system prompt (two plus signs) appears, stop the system with the following command:

```
stop
```

The system responds with the message

```
... System Shutdown Complete ...
```

9. Remove the master disk or tape from the drive and store it in a safe place. If you are building your system from a tape, the operating system on your hard disk is now complete.

10. If your machine is set to enable automatic-boot mode (see your hardware documentation), the system issues the following message after a delay of at most 5 to 10 seconds:

```
UniFLEX AUTO-BOOT - Hit ^C to Abort
```

Do not type a response; allow the system to boot automatically.

If your system has a switch which allows you to disable automatic boot and that switch is set, you must boot the system yourself. To do so, first type the sequence escape-B. When the ROM prompts you for the name of the file to load, type

<hard_disk_drive>:uniflex

followed by a carriage return. The term <hard_disk_drive> is a hardware-specific designation for the drive that contains the system disk (see 68xxx UniFLEX Hardware Setup Notes); "uniflex" is the name of the file to load from the root directory of the system disk (now the hard disk) into main memory. If you type only a carriage return in response to the prompt for a file name, the ROM loads the default file that is appropriate for your hardware (see 68xxx UniFLEX Hardware Setup Notes). In most cases this file is the one you want.

When the ROM finishes loading the file (indicated by the appearance of the ROM prompt below the message about the bss segment), type

control-A

11. When the system has booted, UniFLEX issues prompts that lead you through the procedure of putting the rest of the operating system on the hard disk. Follow the instructions, inserting each floppy disk it requests into the same drive you initially used to boot the system.

12. After copying the information from all the floppy disks to the hard disk, the operating system executes the "set_termcap" command, which prompts for information on the configuration of all the terminal ports on your system. Consult the documentation for "set_termcap" for more information. When it finishes configuring the ports, the operating system automatically shuts down.

13. Your UniFLEX Operating System is now complete.

1.2 Booting a System

Before you can use the operating system, you must boot it. Booting a system consists of executing a special program, called the boot program, which resides in the monitor ROM (read-only memory). The boot program loads the executable file "/uniflex", which is the kernel of the operating system, into main memory and starts to execute it.

In general you can boot the operating system in one of two ways: manually or automatically. However, when you are booting the system for the first time, or whenever you are rebuilding the operating system, you must boot the system manually.

1.2.1 Automatic-Boot Mode

By default, the operating system enters automatic-boot mode whenever you either turn the machine on or reset it. Unless your hardware allows you to bypass the automatic-boot mode, you see the following message whenever you start or reset the machine:

```
UniFLEX AUTO-BOOT - Hit ^C to Abort
```

If you do not type control-C within 30 seconds, the system boots itself.

Some machines have a switch which forces the system into manual-boot mode. Normally, however, automatic-boot mode is the mode of choice.

1.2.2 Manual-Boot Mode

If you set your system for manual-boot mode or if you type control-C to interrupt the automatic procedure, the system sends you the following message:

```
<hardware_description> UniFLEX ROM <release_date>
```

This message indicates that the ROM is waiting for your instructions. The ROM supports a variety of commands, which are described here. A carriage return must follow any command that is neither an escape sequence nor a control sequence.

a	Access memory one byte at a time.
control-A	Continue execution at the current program counter. The program counter points to either the entry point of a program just loaded with the "escape-B" command or to the last breakpoint.
escape-B	Load a file containing the operating system from a disk. The sequence escape-B is entered by typing

the "escape" key, followed by an uppercase 'B'. When the ROM receives this command, it prompts you for the name of the file it is to load:

```
$Boot from disk or tape
File name?
```

In response, you must specify both the name of the file and the device on which it is located. The format of the response is

<dev_name>:<file_name>

where <dev_name> is one of the device names that the ROM can recognize and <file_name> is the name of the file containing the appropriate version of the operating system. A device name is of the form "w<num>", "fd<num>", "smd<num>", or "st0"--for a mini-Winchester, a floppy disk, a storage module device, or a streaming tape. In all cases the minimum value of <num> is 0; the maximum is system-dependent (see 68xxx UniFLEX Hardware Setup Notes). The argument <file_name> specifies the name of an executable file to load into memory. Although the name of a file may, in general, contain up to 55 characters, the name used here may contain no more than 14 characters. The file referenced by this name must be in the root directory of the specified device. If you are loading an operating system, the name of the file is "uniflex". You may, however, load any executable file. Your response must be followed by a carriage return.

As it loads the file, the ROM displays three messages:

```
Text segment <size> bytes at <address>
Data segment <size> bytes at <address>
Bss segment <size> bytes at <address>
```

When the ROM finishes loading the file (indicated by completion of the message about the bss segment), it waits for another command.

escape-D Dump memory to a device. For a detailed explanation of how to take a memory dump, see Section 3.9.

e Access only bytes with even addresses.

o	Access only bytes with odd addresses.
l	Access memory a long word (32 bits) at a time.
control-S	Execute a single instruction.
escape-V	Compare the contents of memory to the contents of the specified file. The ROM prompts you for the name of the file to use in the comparison. You must specify both the name of the file and the device on which it is located. The format of the response is

<dev_name>:<file_name>

where <dev_name> is one of the device names that the ROM can recognize and <file_name> is the name of the file containing the appropriate version of the operating system. A device name is of the form "w<num>", "fd<num>", "smd<num>", or "st0"--for a mini-Winchester, a floppy disk, a storage module device, or a streaming tape. In all cases the minimum value of <num> is 0; the maximum is system-dependent (see 68xxx UniFLEX Hardware Setup Notes). The argument <file_name> specifies the name of the file to compare memory to. Although the name of a file may, in general, contain up to 55 characters, the name used here may contain no more than 14 characters. The file referenced by this name must be in the root directory of the specified device.

w	Access memory a word (16 bits) at a time. This mode of access is the default.

The ROM supports some additional commands, which are a subset of the commands supported by the machine-language debugging system ("qdb") that comes with the operating system. These commands are briefly described here. For more detailed information about these commands, see the documentation for "qdb" in 68xxx UniFLEX Utility Commands. A carriage return must follow a "qdb" command.

b	Set a breakpoint.
B	List the breakpoints that are currently set.
c	Clear one or all breakpoints.
d	Dump a section of memory.
g	Continue execution at the current program counter. The program counter points to either the entry point of a program just loaded with the "escape-B" command or to the last breakpoint (equivalent to control-A).
i	Disassemble instructions.

m Modify bytes in memory.
M Display current memory map.
r Display the contents of all registers.
R Set the contents of a register.
s Execute a single instruction (equivalent to control-S).

1.3 Initializations

Whether you boot your system automatically or manually, the boot procedure ends when the file "uniflex" is loaded into main memory and starts to run. When it starts running, it first allocates memory for the various tables it needs, then initializes the hardware associated with the operating system. After it completes these initializations, the operating system is completely functional.

1.4 Updating a System Disk

Periodically the operating system is revised either to improve its performance or to remove bugs. If your maintenance is current at the time of a revision, you can obtain a free version of the revised operating system by returning your set of master disks to Technical Systems Consultants. We will update the disks and return them to you as quickly as possible.

When you receive your updated disks, follow the procedure in this section to update your hard disk:

1. Place the new master disk in floppy drive 0.
2. Boot the system manually from the new master disk (see Section 1.2.2).
3. Type the following command:

```
/etc/update_system
```

4. From this point on, the master disk instructs you on updating your hard disk.

1.5 Adding UniFLEX Support Software to Your System

The procedure for adding UniFLEX Support Software, such as utilities packages and compilers, to your operating system is simple and is the same in all cases:

1. Boot from the existing system disk.
2. Login, if necessary, as the system manager.
3. Type the following command:

```
install
```

4. The operating system responds with a prompt telling you which drive to use. Insert the disk containing the support software into the appropriate drive and type a carriage return.
5. If the software spans more than one disk, the system tells you when to put the next disk into the drive.
6. When the system completes the installation procedure, it sends a message to the screen telling you that you may remove the disk from the drive. When you remove the master disk, store it in a safe place.
7. The product is now available on your system disk.
8. If for some reason the installation procedure fails, you will receive a message to that effect. In such a case telephone Technical Systems Consultants for assistance.

1.6 Updating UniFLEX Support Software

Periodically support software is revised either to improve its performance or to remove bugs. If your maintenance is current at the time of a revision, you can obtain a free version of the revised software by returning the corresponding master disk or disks to Technical Systems Consultants. We will send the updated version of the software to you as soon as possible.

Once you have the revised software, you can install it on your hard disk by following the procedure described in Section 1.5.

Chapter 2

The Init-Control File

2.1 Introduction

When the operating system finishes initializing the system (see Section 1.3), it executes the first task, "init". This task is the progenitor of all other tasks on the system. If the immediate parent of a task dies, "init" acts like the next of kin, becoming the task's foster parent.

The "init" program can do a wide variety of things. Precisely what it does is controlled by the contents of the file "/etc/.init.control" (also called the init-control file) when the system is booted. The operating system is provided with a standard version of "/etc/.init.control", but you can tailor the program to suit your needs, or rewrite it completely. When modifying the file, however, exercise extreme caution because the usefulness of a system with a damaged init-control file is limited (see Section 2.6.1). Always keep a copy of the standard init-control file so that you can run your system if you inadvertently damage the copy on your hard disk.

The init-control file is a series of commands that the "init" program automatically executes each time you boot the system. Each line of code is either a shell command (see 68xxx UniFLEX Utility Commands) or a specially constructed line beginning with a flag that indicates to "init" the purpose of that particular line. One of these flags, the plus sign ('+'), must be followed by one of several "control commands". This chapter explains the execution of a shell command from the init-control file and describes the meaning of each flag and of each control command. It also discusses the elements that are crucial to any init-control file as well as the error messages returned by "init".

Theoretically, the init-control file may consist of between 0 and 256 lines of code inclusive. Practically speaking, however, even a minimal init-control file contains fifty or sixty lines of code. Each line may contain a maximum of 80 characters. Appendix A provides a line-by-line description of a generic init-control file similar to the one shipped with your operating system.

2.2 Executing a Shell Command

The ability to execute a shell command from the init-control file allows you to perform routine tasks automatically when you boot the system. For instance, you can set the date, initiate any spoolers the system supports, and check the integrity of your disks by embedding the appropriate commands in the init-control file.

If a line of code in the init-control file does not begin with one of the flags discussed in the next section, the "init" program first opens the device "/dev/console" (or "/dev/tty00 if "/dev/console" does not exist) as file descriptors 0, 1, and 2 (standard input, standard output, and standard error), then executes the specified shell command. This method is the more common way of invoking a shell command from the init-control file.

You can also execute a shell command without opening any standard I/O channels. To do so, precede the shell command with an exclamation point, '!'. You should invoke a shell command in this fashion if the standard I/O channels are either unnecessary or unidentified (a possibility if your hardware allows you to change the location of "/dev/console").

2.3 Flags

The "init" program recognizes four flags: the colon (:), the plus sign (+), the exclamation point (!), and the arrow (->, a hyphen followed by a greater-than sign). The flag tells "init" how to interpret the code which follows the flag. If a line of code in the init-control file does not begin with one of these flags, "init" interprets that line as a shell command (see Section 2.2). Descriptions of the four flags follow:

:<label>	Assign the specified label to this line of the script. A label may contain between 1 and 32 characters inclusive.
-> <label>	Jump to the line referenced by the label. The label must be assigned to the line with the ':' command.
+<control_command>	Execute the specified control command (see Section 2.4).
!<shell_command>	Execute the specified shell command without opening any standard I/O channels (see Section 2.2). You should use this

form of invoking a shell command if the standard I/O channels are either unnecessary or unidentified (a possibility if your hardware allows you to change the location of "/dev/console").

2.4 Control Commands

The commands that can follow a plus sign (the control commands) are as follows:

*	Ignore this line; it is a comment.
c -> <label>	Branch to the specified label if the system disk is not "clean". A disk is clean if it has been properly unmounted since the last time it was mounted. Because a "dirty" disk may be corrupted, it must be cleaned by the "diskrepair" command before being used.
f -> <label>	Branch to the specified label if the last shell command failed (terminated with a nonzero status).
g <+_or_->	Start (with a plus sign, '+) or exit from (with a minus sign, '-') a "ghost" shell program. Executing a ghost shell program improves the performance of subsequent shell commands because once the system loads the text of the shell program, it is there for other shell programs to use (the shell is a shared-text program). If it is not running a ghost shell, the system must load the text of the shell program each time it executes a shell command.
h <history_entry>	Put the specified two-character sequence, followed by a time stamp, into the history file, "/act/history". The following table shows the valid entries.

Entry	Meaning
bt	System was booted.
su	System entered single-user mode.
mu	System entered multi-user mode.
st	System was stopped.
bd	Time and date just before date was set.
ad	Time and date set with "date" command.

The "history" command is indifferent to the entries, but the accounting programs balk at any entry that is not a terminal number or an entry from this table. Entries signifying logging in and logging out should not be made from the init-control file. The "login" program makes the appropriate entry when a user logs in (if the history file exists), and "init" makes the appropriate entry when a user logs out.

l <user_name>

Log the user in as the specified user, but stay in single-user mode. You may not invoke this command if the system is in multi-user mode. The name specified must be in the password file, "/etc/log/password". The system prompts for a password if that user has one.

m

Enter multi-user mode. The "init" program starts a "login" task for each terminal that is enabled for login (has a plus sign in the first column of its entry in the file "/etc/ttylist"). It watches these "login" programs, and if one of them terminates, it replaces it with another one.

o <+_or_->

Open (with a plus sign, '+') or close (with a minus sign, '-') the device "/dev/console" (or "/dev/tty00" if "/dev/console" does not exist) as file descriptors 0, 1, and 2 (standard input, standard output, and standard error). Opening the console device speeds the performance of the "init" program if it must print many messages on the console because "init" need not open the console each time it prints a message. You

should close the device before doing any interactive procedures, so that you do not inadvertently jeopardize the "init" program.

p <message>

Print the string specified by <message> on the console, followed by a carriage return (hexadecimal 0D) and a line-feed character (hexadecimal 0A).

s

Shut the system down. Shutting down the system consists of updating and unmounting all mounted disks (including the system disk), killing all tasks running on the system, printing a message on the console saying that the system shutdown is complete, and restarting the ROM boot process if automatic-boot mode is enabled.

<sig_num> -> <label>

Branch to the specified label when receiving the specified signal. The label must be assigned to the line with the ':' command. At present, "init" accepts only the numbers 1, 4, 6, and 8 as signal numbers. The "shutup" command, when invoked without a minus sign, generates the signal 4; when invoked without it, an 8. The "stop" command, when invoked without a minus sign, generates the signal 6; when invoked without it, a 1. A list of signals and their corresponding numbers appears in Appendix B.

t <+_or_->

Enable (with a plus sign, '+') or disable (with a minus sign, '-') the tracing of the execution of the control commands. When tracing the commands, "init" displays each control command on the console in the following format

INIT: EXEC <line_number> - <code>

before beginning to execute it. The default init-control file disables tracing feature.

u <seconds>	Update the file system at the interval specified by <seconds>. When the operating system updates the file system, it flushes each modified buffer to the appropriate disk. The value of <seconds> must be greater than or equal to 0. The default init-control file sets the interval to 30 seconds, which is appropriate for most situations. A value of 0 disables the update feature. We recommend that you use an interval between 10 and 60. Setting the interval below 10 is likely to impede system performance; setting it above 60 results in infrequent updates that leave you highly vulnerable in the event of a system crash.
w <seconds>	Pause for the specified number of seconds. The value of <seconds> must be greater than 0.

2.5 Essential Elements of the Init-Control File

Under no circumstances should you remove the code that automatically invokes the "diskrepair" command if the system disk has not been properly unmounted (see the descriptions of line 16 and lines 111-119 in Appendix A). Also, it is essential for the command that shuts the system down to appear in an appropriate place in the init-control file. Otherwise, you have no clean way of shutting down the system.

2.6 Error Messages

The "init" program processes the entire init-control file before it begins execution. If it finds an error, it sends an error message to standard error. A list of possible error messages and an explanation of each one follows.

2.6.1 Fatal Errors

If the "init" program encounters a fatal error, it executes a single-user shell program, giving you an opportunity to edit the "init-control" file and fix the problem. When this shell program terminates, the system automatically shuts down.

Bad command.
<line_number> - <code>
This error should not occur. If it does, contact Technical Systems Consultants.

Control file syntax error.
<line_number> - <code>
The "init" program could not parse the code at the specified line number.

Error allocating memory for a label: <reason>
<line_number> - <code>
The operating system returned an error when it tried to allocate memory for a label. This message is followed by an interpretation of the error returned by the operating system.

Error allocating memory: <reason>
The operating system returned an error when "init" tried to allocate memory for the code in the init-control file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "init" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error shutting down system : <reason>
The operating system returned an error when "init" tried to halt the system. This message is followed by an interpretation of the error returned by the operating system.

Invalid argument "<arg>".
This message indicates that something is wrong with the operating system. Contact Technical Systems Consultants for assistance.

Invalid option '<char>'.
This message indicates that something is wrong with the operating system. Contact Technical Systems Consultants for assistance.

Too many labels.
An init-control file may define no more than 32 labels.

Too many lines in "<file_name>".
An init-control file may not contain more than 256 lines of code.

Undefined label "<string>".
<line_number> - <code>
The specified string was used as a label, but was not defined with the ':' command.

2.6.2 Nonfatal Errors

Error executing the "login" program: <reason>
The operating system returned an error when "init" tried to execute the "login" program. This message is followed by an interpretation of the error returned by the operating system.

Error executing the shell program.
The operating system returned an error when "init" tried to execute the shell program. This message is followed by an interpretation of the error returned by the operating system.

Error getting parameters associated with "<dev_name>": <reason>
The operating system returned an error when "init" tried to adjust the parameters (e.g., the baud rate) associated with the specified device. This message is followed by an interpretation of the error returned by the operating system. Most likely, the specified device is not a terminal.

Error opening "<dev_name>": <reason>
The operating system returned an error when "init" tried to open the specified device. This message is followed by an interpretation of the error returned by the operating system. Most likely, one of the entries in the file "/etc/ttylist" is not in the directory "/dev".

Error opening "/etc/ttylist": <reason>
'm' command ignored.
The operating system returned an error when "init" tried to open the file "/etc/ttylist". This message is followed by an interpretation of the error returned by the operating system. The operating system does not enter multi-user mode.

Illegal baud rate <code> for "<dev_name>".
The baud rate (specified in the second column of each entry in "/etc/ttylist") must be a number from 1 to 9 inclusive, a character from 'a' to 'f' inclusive, or a blank (see Section 7.7.1).

Invalid argument to ´k´ command: <sig_num>
Command ignored.
The number specified for the ´k´ control command is not a valid signal number. At present, "init" accepts only the numbers 1, 4, 6, and 8 as signal numbers. A list of all signals supported by the operating system and their corresponding numbers appears in Appendix B.

No handling instructions for signal <sig_num>.
Signal ignored.
The "init" program received a signal for which it has no handling instructions. It ignores the signal. By default, "init" catches and ignores all catchable signals (see Appendix B).

Syntax error. Command ignored.
<line_number> - <code>
The control command ´g´, ´o´, or ´t´ was followed by a character other than a plus or minus sign. The "init" program ignores this line of code.

Too many lines in "/etc/ttylist".
The file "/etc/ttylist" may contain no more than 64 lines. The "init" program ignores any lines beyond line 64.

Chapter 3

Starting the System

3.1 Day-to-Day Procedures

Most operating systems support two modes of operation: single-user mode and multi-user mode. In single-user mode, the only active terminal is the console; in multi-user mode, a user may log in at any terminal that is enabled for login (see Section 7.7.1).

The standard init-control file, which comes with the operating system, boots the system into single-user mode. This mode of operation is useful at times when you want to be certain that you are the only user on the system. Such times may include the time immediately after booting the system when you perform certain daily operations and times when you are performing system maintenance.

Booting to single-user mode may jeopardize the security of your system. By default, when the system comes up in this mode, it logs the user in as "system" without requesting a password. Thus, any user who can get to the hardware and knows how to boot it can access all the software as system manager without knowing the system password. In an environment in which this presents a problem, you have two choices: you can direct the operating system to request a password in single-user mode, or you can boot directly to multi-user mode.

3.1.1 Booting to Single-User Mode

The standard init-control file executes a single-user shell program as the last step in establishing single-user mode (see line 51 of the sample init-control program in Appendix A). You can replace this line of code with the following command:

```
+1 system
```

Doing so logs the user in as "system", but if a password exists, the operating system asks for it. The operating system does not execute the shell program until it receives the correct password. Thus, a user who does not know the password cannot access the operating system.

Putting this command in the file at this point also ensures that if the system is taken to single-user mode during shutdown, the user who does so must know the system manager's password in order to proceed.

In single-user mode only one terminal, the console, is active. By default the console is "/dev/console". If no device by that name exists, the console is "/dev/tty00".

When you are ready to put the system into multi-user mode so that other users may log in, simply execute the following command:

```
log
```

3.1.2 Booting Directly to Multi-User Mode

Depending on your environment, you may prefer to have your system boot directly to multi-user mode. It is possible to bypass single-user mode entirely by performing routine operations and maintenance from commands in the init-control file. In multi-user mode each user must log in with a valid combination of a user name and password.

3.2 Setting the Date

The first thing you should do after booting the system (if you do not do so automatically in the init-control file) is to set the date and time. Most hardware comes equipped with a built-in clock from which the operating system can read the date and time. Other systems require you to set the date and time yourself. You can determine whether or not your system has a built-in clock by invoking the command

```
info /bin/date
```

If and only if the information field returned by this command contains a line of the form

```
-- Hardware configuration: <hardware_designation>
```

your system supports a built-in clock.

3.2.1 Using a Built-in Clock

If your system has a built-in clock, you can set the date directly from this clock with the following command:

```
date +s
```

If your system does not have a built-in clock, this form of the "date" command has no effect.

If for any reason you wish to bypass the built-in clock, you may set the date manually, following the directions in the next section. Setting the date manually not only sets the date for the system but also sets the date in the built-in clock.

3.2.2 Setting the Date Manually

The "date" command has two forms: one with an argument and one without. Any user may execute the "date" command without an argument. In response, the system returns the current date and time. The system manager may also use the "date" command with an argument. This form of the command sets the date both for the built-in clock, if one exists, and for the operating system. The syntax for this version of the command is as follows:

```
date [<mm>-<dd>-<yy>] <hr>:<min>[:<sec>]
```

where <mm> is a number from 1 to 12 inclusive representing the month, <dd> is a number from 1 to 31 inclusive representing the day, and <yy> is a two-digit number representing the last two digits of the year. The time must be 24-hour-clock time where <hr> is a number from 0 to 23 inclusive representing the hour, <min> is a number from 0 to 59 inclusive representing minutes, and <sec> is a number from 0 to 59 inclusive representing seconds. For example, the following command sets the date to 7:30 AM on April 17, 1987:

```
date 4-17-87 7:30:00
```

If the system has only been down a short time, it may not be necessary to set the month, day, and year. If you do, you may include just the day, the day and the month, or the day, month, and year. The operating system takes values for the day, month, and year from the disk if you do not specify them.

It is, however, always necessary to set the time (if you omit the seconds argument, the system assumes a value of 0). If you do not specify a time, the system responds with the message

```
Syntax: date [ [<mm>-<dd>-<yy>] <hr>:<min>[:<sec>] ]
```

Even though it is not always necessary, it is a good idea to specify both arguments to the "date" command. Because other parts of the operating system reference the date, it is important for the date to be correct. The consequences of having the date set incorrectly range from a minor inconvenience to a serious problem.

You should enter the local time when you set the date. The system stores the time internally in seconds since January 1, 1980, at the zeroth meridian (in Greenwich, England). As it makes this conversion, it adjusts the result for the local time zone and for Daylight Savings Time if it is in effect (see Section 9.4.3).

3.3 Message of the Day

Whenever a user logs in, in response to the login prompt, the login program reads the file "/etc/log/motd" and sends it to standard output before issuing the system prompt. Thus, if you want to send a message to all users, you can enter it in this file.

You can edit the message-of-the-day file just as you would edit any other file. An example follows:

```
++ chd /etc/log
++ edit motd +b
^d!
#i
    1.00=April 18, 1987:  A new Pascal compiler was
    2.00=installed on the system today.  Documents which
    3.00=describe its use are available in the main office.
    4.00=All users are welcome to use this new compiler.
    5.00=#s
++
```

The first line of this example changes your working directory from the root directory (your default location on booting) to the directory "/etc/log". The next line invokes the editor, telling it to edit the file "motd" without creating a backup file. The next command deletes

the contents of the entire file. (You may not always want to delete old messages. In such a case simply leave out the ´^d´ and position yourself at the bottom of the file with the command ´!´ instead.) The letter ´i´ in response to the editor´s prompt puts you in insert mode. Now you can enter your message.

It is a good idea to date your messages so that both you and the users know when they were first issued. When you finish typing your message, you can exit the editor by typing a pound sign, ´#´, as the first character on a line, followed by an ´s´ for "stop". (See the documentation on the editor in The UniFLEX Operating System for more detailed instructions on the use of the editor.)

3.4 Formatting a Disk

Any disk that is to be used with the UniFLEX Operating System must first be properly formatted. The "format" commands not only establish the physical boundaries of the sectors on the disk but also set up the UniFLEX file system by writing both the boot sector (sector 0) and the system information record (SIR), by establishing the root directory, by initializing the file descriptor nodes (fdns), and by reserving paging space. The precise function of a "format" command can be altered by the many available options. These commands are documented with other UniFLEX utilities in 68xxx UniFLEX Utility Commands.

If you are formatting a disk that is to be used as a system disk, be sure to use the ´r´ option to reserve some paging space. Paging space is a section of the disk that is reserved for storing portions of tasks that the operating system removes from memory to make room for tasks of higher priority. When the system runs out of memory, it selects as many 4K pages as it needs from tasks that are currently in memory, and writes them to the paging space.

A system disk cannot function without paging space. The amount of paging space you need depends on the individual nature of your system--in what ways it is used and how heavily it is used. It is impossible to give a single amount of paging space that serves all systems. However, the following factors should be considered when determining the amount of paging space to reserve:

1. The maximum task size the machine permits (see the documentation for your hardware).

2. A system is usually running a number of tasks which may not be obvious to any users. Such tasks include the login program, the initialization program ("init"), and extra shell programs.

3. A compiler may call the assembler and loader. Doing so may cause the compiler to be paged out.

4. The more random access memory (RAM) you have, the less paging space you need.

5. The more users on your system, the more paging space you need.

6. The argument to the 'r' option specifies how many cylinders to reserve for paging space. A cylinder consists of all the data tracks that can be accessed by all the read-write heads without mechanically moving the head assembly. The operating system always formats such a disk with 512 bytes per sector. The number of sectors per track is hardware-dependent. You can easily calculate the number of bytes per track. The following example assumes 17 sectors per track.

   17 sectors/track * 512 bytes/sector = 8704 bytes/track

   To convert to the number of bytes per cylinder, you must multiply by the number of tracks per cylinder. For a hard disk the number of tracks per cylinder is equal to the number of heads in the disk.

For further details about formatting a disk, see 68xxx UniFLEX Utility Commands.

3.5 Sequestering Bad Blocks

Almost inevitably, some parts of a disk become physically damaged. A block that is physically damaged is referred to as a bad block. The operating system cannot use a bad block. If the operating system tries to read from or write to such a block, it fails and sends the user an I/O error. Depending on the location of the bad block, an I/O error may range from a minor inconvenience to a serious problem that causes the system to crash. It is therefore wise to remove a bad block from use as soon as it is detected. The UniFLEX Operating System supports two methods of doing so: one during the formatting of a disk; the other

after the disk has been formatted. In either case the bad block is placed in a file named ".badblocks" in the root directory (also known as the bad-blocks file). The operating system knows not to allocate any of the blocks in this file.

3.5.1 Formatting with the 'l' or 'L' Option

The "format" commands support two options which allow you to specify that certain blocks should be placed in the bad-blocks file. The use of these options is described in detail in 68xxx UniFLEX Utility Commands. Basically, you should look at the list of bad blocks supplied by the manufacturer of the disk. This list should specify a head, a cylinder, and either a range or a sector number for each bad block. In order to put a block in the bad-blocks file, the "format" command must be told either the head, cylinder, and logical 512-byte sector of the bad block or its logical block-number. The tables in Appendix B and Appendix C of the format documentation enable you to convert the information supplied by the manufacturer to the information needed by the operating system.

If you know of additional bad blocks on the disk, you should specify them to the "format" command as well. Alternatively, you can format the disk without using either of these options and follow the procedure described in the next section.

3.5.2 The "badblocks" command

If at any time after the disk is formatted the system reports an I/O error, you should locate the block or blocks causing the problem and isolate them from the system by placing them in the bad-blocks file. The most thorough way to do so is first to invoke the following command:

```
/etc/devcheck <dev_name> +V
```

The "devcheck" command checks the entire disk for I/O errors. By default, it tries to read each block on the disk. It reports to standard error the address (in decimal) of each block it cannot read. With the 'V' option in effect, "devcheck" checks the disk more thoroughly by nondestructively reading from and writing to each block. This form of the command is extremely time-consuming, but it provides a meticulous check of the quality of the disk.

If the "devcheck" command reports any bad blocks, you should immediately invoke the "badblocks" command. This command, which is documented in 68xxx UniFLEX Utility Commands, reads a list of decimal addresses from the command line, places the corresponding blocks in the bad-blocks file, and by default, executes the "diskrepair" command (see Chapter 10) to clean up any damage to the disk caused by sequestering the bad blocks.

You can automatically invoke the "badblocks" command from "devcheck". For further details, see 68xxx UniFLEX Utility Commands.

3.6 Routine Verification of the Logical Structure of the Medium

Whenever you boot your system, it is wise to make certain that the logical structure of the disk or disks you are using is intact. Any errors are best detected before you start to use the system. You can check the logical structure of the disk with the "diskrepair" command (see Chapter 10).

3.7 Maintaining Backup Files

One of the hazards of working with computers is that they do not always behave perfectly. Power failures and overheating are just two of the things which may cause a system to malfunction. Whenever a system shuts down in any but the prescribed fashion (see Section 3.13), the structure of the system disk and any disks mounted on the system may be damaged. Although complete recovery from such an event is often possible (see Chapters 10 and 11), sometimes it it not. Some or all of the files on the system can be irreparably damaged.

Your best protection against disaster is the maintenance of a proper backup system, which consists of a copy of every file in use on your system. You will never know how important the maintenance of a good backup system is until you need, but do not have, one. It is a good idea to maintain two backup copies of everything and to store one copy at a different site. This precaution ensures you of an intact copy of the files even if a fire, flood, nuclear accident, or some other catastrophe destroys your primary site.

We strongly recommend that you back up your system every day. You can make backup copies by using the "backup" or "copy" command (see 68xxx UniFLEX Utility Commands).

It is particularly important for you to maintain an up-to-date copy of all important files if your hardware supports only one floppy disk drive and have neither a streaming tape or a second hard disk. With such a configuration you run a greater risk of being unable to salvage any data from a damaged disk. If your system includes a streaming tape, a second hard disk, or more than one floppy disk drive, you might be able to salvage some data, but it is far easier to back up regularly than to try to piece together data from a shattered disk.

3.8 Mounting Devices

By default, when you boot your system, you "come up" in the root directory on the root device. The root directory is the directory at the top of the directory tree. It has no parent directory; all other directories are descendants of the root directory.

You may need to use devices other than the root device. For instance, users may need to access information stored on another disk. In order to make that information available, you must mount the device containing the disk on a node of the directory tree with the following command:

```
/etc/mount <dev_name> <dir_name>
```

mounts <dev_name> at the directory <dir_name>.

As long as the device is mounted, any references to <dir_name> actually access the root directory of the disk in the device mounted there. Any files in the directory at which the device is mounted are inaccessible as long as the device is mounted. Only the system manager may execute the "mount" command.

When the operating system mounts a device, it sets a flag on the disk in the device. Normally, the "unmount" command

```
/etc/unmount <dev_name>
```

clears the flag. If you remove the disk from a mounted device without

first unmounting it, the flag remains set, and the "mount" command refuses any further attempts to mount a device containing that disk. Instead, it returns the message:

Error mounting "<dev_name>" in "<dir_name>": Device may be corrupted.

If you receive this error message, invoke the command "diskrepair +M" the problem disk. The 'M' option tells "diskrepair" to do nothing but clear the "mount flag" on the specified disk (see Section 10.2.6).

A block device can be mounted on any directory in the file system. However, because any files in a directory on which a device is mounted are inaccessible, it is good practice to mount a device on an empty directory. For convenience, it is also a good idea to mount the device on a directory in the root directory, in order to minimize the length of the reference to the mounted device. The operating system comes with four empty directories called "usr0", "usr1", "usr2", and "usr3". These directories, which reside in the root directory, are convenient nodes on which to mount devices. You can create additional directories for this purpose if you need them.

For instance, suppose that you have a file called "billing_info" in the root directory of a removable-cartridge disk and that users regularly need access to the information in this file. You can mount the disk by placing it in drive "rc0" and invoking the following command:

```
/etc/mount /dev/rc0 /usr0
```

This command tells the operating system to place the device "/dev/rc0" at the node "/usr0" in the root device. Users can now reference this file as "/usr0/billing_info".

The "mount" command supports one option, 'r', which tells the system to mount the device for reading only. Use of this option allows users to read files on the mounted device but prohibits them from writing to the device.

3.9 Taking a Memory Dump

A memory dump is essentially a picture of the contents of a portion of the system's random-access memory (RAM). You may find yourself in the unfortunate position of needing a memory dump because your operating system is nonfunctional. In such a case, we can help you most efficiently if you can take a memory dump and send it to us on floppy disks or tape.

The ROM can obtain a memory dump for you as long as you have the appropriate medium available. If your system supports streaming tape, you need not make any special preparations in advance because you can always dump to a tape. However, if your system does not support streaming tape, you should always have on hand a set of formatted floppy disks that can hold the entire contents of your RAM. The disks should be labeled and numbered so that all you need to do is take the dump and send it to us. It is unlikely that you will ever need to use these disks, but you should have them available as insurance. In some instances a memory dump provides the only means by which we can help you solve a problem.

If you do need to take a memory dump, follow this procedure:

1. Press the "abort" switch on your machine. Do not reset the machine! Resetting clears the RAM.

2. When you see the prompt from the ROM ("? "), type

   escape-D

   (that is, type the key labeled "escape" or "esc", followed by an uppercase 'D').

3. The ROM then prompts you for information about the device to dump to.

4. When you finish selecting the device, the ROM prompts you to insert the first volume of the medium into the specified device. Put your tape or floppy disk in the appropriate drive and type a carriage return. The ROM prompts you for additional volumes if necessary.

5. Send the dump to

Technical Systems Consultants
111 Providence Rd.
Chapel Hill, NC 27514

Please include a description of the circumstances that led to the failure of the system.

3.10 The History File

The root directory contains a directory called "act", which is initially empty. This directory is a place to store accounting information. (The optional software package, User/System Accounting System, makes extensive use of this directory.) As shipped, the only accounting procedure the UniFLEX Operating System performs is the maintenance of the history file, "/act/history", if it exists. The standard init-control file directs the operating system to make an entry in the history file each time the system is booted or stopped and each time it goes from single-user mode to multi-user mode or from multi-user mode to single-user mode. In addition, the "date" command writes an entry to the file each time the system manager sets the date, and the "login" command writes an entry each time any user logs in or out.

A listing of the history file is unintelligible. If you want to extract information from the file, you must execute the "history" command. For an explanation of the output from this command see 68xxx UniFLEX Utility Commands.

If you want to maintain a history file, all you need to do is create a file named "/act/history" with the following command:

```
create /act/history
```

After creating the file you should alter the permissions so that other users cannot accidentally destroy it by creating a file with the same name. To do so, you must deny other users write permission in the file:

```
perms o-w /act/history
```

The history file can rapidly grow and occupy a lot of space on your system disk. Therefore, you may want to start a new history file

periodically. If you want to save the old information, you can copy the file to a backup device. It is a good idea to rename the file in some way so that the name indicates the period of time covered by that particular file. For instance, if you start a new history file once a month, you may append the name of the month to the file as you back it up. When you have backed up the file, simply create the file "/act/history" again. This command truncates the existing file of that name to length 0.

3.11 Adding New Programs

When you add a new executable program to your system, you must decide what directory to put it in. The decision depends, in part, on who is to have access to the program. System-wide programs can conveniently go in one of two places--the directory "/bin" or the directory "/usr/bin". By default, the shell program automatically searches both these directories when it is looking for an executable file (see "addpath" and "setpath" commands in 68xxx UniFLEX Utility Commands). Commonly used programs should go in "/bin" because, by default, the shell program searches that directory before it searches "/usr/bin". Less commonly used programs belong in "/usr/bin".

3.12 The File "/etc/startup"

The standard init-control file is constructed so that when you type the "log" command to take the system from single-user mode to multi-user mode, the operating system looks for the file "/etc/startup" (the start-up file), and if it exists, executes all the commands it contains before entering multi-user mode. By default, if any of the commands in the file fails, the operating system terminates execution of the file and executes another single-user shell program. From this shell program you can edit the start-up file and fix the problem.

A sample start-up file, which initiates two printer spoolers, might look like this:

```
/etc/insp letter
/etc/insp ppr +f
```

The feature of automatically executing the start-up file during the transition from single- to multi-user mode mimics the behavior of the 6809 UniFLEX Operating System. With a 68xxx UniFLEX Operating System you can bypass this step by putting the commands directly into the init-control file and removing from it the line that executes the start-up file.

3.13 Shutting Down the System

You should routinely use one or more of the commands supplied with the operating system to shut it down. Do not reset the computer unless forced to by a system crash. Failure to use the software to shut down the system may result in damage to the system disk and any disks mounted on the system.

The standard init-control file establishes a procedure for shutting down the system that mimics the 6809 UniFLEX Operating System. That two-step procedure is described in this section. You can, however, proceed directly to the second step. In addition, if security at your site warrants it, you can alter the init-control file so that you cannot get to single-user mode from multi-user mode or so that you must give a password in order to do anything in single-user mode (see Section 2.4).

3.13.1 Step One

The "shutup" command, which only the system manager may invoke, takes the system from multi-user to single-user mode. The format for this command is

```
/etc/shutup [[-]<minutes>]
```

By default, the "shutup" command takes the system from multi-user to single-user mode by sending a hang-up interrupt to all tasks. This interrupt is followed by a 15-second delay before the system actually enters single-user mode. If any tasks are being executed when the "shutup" command is run, the hang-up interrupt permits many of them (those that recognize and honor the interrupt) to terminate cleanly without the loss of any data. You may suppress both the interrupt and the 15-second delay by using the optional minus sign.

The argument <minutes> is a number between 0 and 60 inclusive, which specifies the number of minutes the system should wait before beginning to shut down. The default argument is 15.

If you invoke the "shutup" command with an argument other than 0, the system executes the command in the background and sends the task ID to standard output so that you can subsequently terminate the "shutup" program with the "int" command if necessary. The "shutup" command sends a message announcing the impending shutdown to all terminals that are logged in (even if they have normal messages locked out). It repeats this message at intervals until the system actually shuts down. In the default case of fifteen minutes, the message is sent fifteen, ten, five, three, two, and one minute prior to shutdown.

3.13.2 Step Two

To bring the system to a halt from single-user mode, issue the following command:

```
/etc/stop [[-]<minutes>]
```

By default, before stopping the system the "stop" command sends a hangup interrupt to all tasks. This interrupt is followed by a 15-second delay before the system actually shuts down. If any tasks are running when the "stop" command is invoked, the hang-up interrupt permits many of them (those that recognize and honor the interrupt) to terminate cleanly without the loss of any data. You may suppress both the interrupt and the 15-second delay by using the optional minus sign. Only the system manager may invoke the "stop" command.

After you shut down your system, it is advisable to reset it in order to withdraw the read-write heads from the surface of the disk. If the auto-boot message appears, type control-C to interrupt the boot procedure. When the ROM prompt appears on your screen, shut off the power to the computer. Also shut off the power to any peripherals attached to the computer.

Chapter 4

Using a Printer Spooler

4.1 Introduction

The UniFLEX Operating System is shipped with a general-purpose printer-spooler which must be configured before use. A printer spooler coordinates the printing of files in response to requests by users. The operating system also supports a more sophisticated spooler, the 68xxx UniFLEX Enhanced Spooler, which is available as a separate product.

4.2 Configuring a Printer Spooler

Before you can use the general spooler you must do four things: select a device; create a spooler command; create a spooler directory to contain the files you send to the spooler; and initiate the spooler.

4.2.1 Selecting a Device

The first step in making a spooler functional is selecting the appropriate device from those in the file "/dev". A UniFLEX device is a special kind of file that channels the data sent to it to a particular device driver, in this case a printer or terminal driver. The driver processes the data and passes it to a physical device, in this case a printer. The links from a device to a particular driver and from a driver to a particular physical device are determined by the device's major and minor device numbers (see Section 6.6). Therefore, when you select the device to associate with your spooler, you select the port that you must connect your printer to. It is not essential to connect the printer to the port before you configure the spooler.

Each system comes with its own set of devices and drivers for various kinds of printers (see 68xxx UniFLEX Hardware Setup Notes). If the device you need is not in already on the system, you can create it with the "makdev" command (see 68xxx UniFLEX Utility Commands).

4.2.1.1 Using a terminal driver for a serial printer

All systems can support a serial printer on any terminal port except the console. If you do connect a serial printer to a terminal port, you may for the sake of convenience wish to give the port an alternative name that suggests to the user a printer rather than a terminal. You can do so by creating a link between the terminal port and a device with a more appropriate name. For example, to create the alternative name "spr" for terminal port "tty13", you would invoke the following command:

```
link /dev/tty13 /dev/spr
```

You can now access the port by either name, but you would most likely want to configure the spooler for the port based on the name "spr".

4.2.2 Creating a spooler command

Let us assume that you have decided to create a spooler for a parallel printer and to associate that spooler with the device "/dev/ppr".

The command you create to send things to the spooler is simply a link to the executable file "/etc/print", which is a part of the UniFLEX Operating System. However, the last component of the name of the file to which you link "/etc/print" must match the last component of the name of the device you have selected ("ppr"). Because many people may be using the command, it is convenient to place it in the directory "/bin" so that the users do not have to specify the full file specification or the path name every time they invoke the spooler. You can accomplish all this with the following command:

```
link /etc/print /bin/ppr
```

which puts into the directory "/bin" an entry which points to the file "/etc/print".

4.2.3 Creating a spooler directory

The next step is to create a directory for the spooler. The last component of the name of this directory must also match the last component of the name of the device driver, and the directory must reside in the directory "/gen/spooler". This is simply done:

```
crdir /gen/spooler/ppr
```

In general it is wise to set the permissions in this directory so as to deny access to all users except the system manager. The following command does so:

```
perms o-rwx /gen/spooler/ppr
```

Your system now contains a new command, "ppr", which spools files to the parallel printer. If you want to change the name of the command to something more memorable, such as an abbreviation for the name of the manufacturer of the printer, you can do so by renaming or linking the device to "/dev/<new_name>", renaming or linking the executable file "/bin/ppr" to "/bin/<new_name>", and creating a directory of the appropriate name in "/gen/spooler" (you should not rename or link a directory). If you choose to link rather than to rename the device and the executable file, you must be careful to avoid having different users accessing the same device by different names at the same time. If this should happen, the files may be sent to the same device simultaneously, resulting in the intermingling of the files as they are printed.

4.2.4 Initiating a printer spooler

Now you have a spooler command and a spooler directory, but before you can actually execute the command, you must initiate the spooler. In fact, each time you boot the system, you must initiate each standard spooler you want to use. (You can simplify this task by putting the relevant commands in the file "/etc/startup"—see Section 3.12) A spooler is initiated by a command of the following form:

```
/etc/insp <splr_name> [+f]
```

where <splr_name> must match the last component of the file specification of the corresponding device. The 'f' option suppresses the banner page and the form-feed character which are both normally printed before each print job. In this example the appropriate command is

```
/etc/insp ppr
```

When you execute this command, the system creates a background task, which runs continuously until you deliberately stop it with a "pstop" command. In addition, it creates a file in the spooler directory called

".mrk*splr?", which contains the task ID of the background task. The background task checks the contents of the spooler directory approximately every 20 seconds. It sends any files present to the printer and deletes them from the spooler directory. Note that the original copies of the files are still intact; only the references to them in the spooler directory are deleted.

4.3 Using the Spooler Command

To use this spooler a user simply types

```
ppr <file_name>
```

When a user invokes the spooler command, the command generates a name for the file being spooled. It does so by appending a three-digit number to the user name of the person who invoked the spooler. Once it has created the new name, the spooler command makes an entry in the corresponding spooler directory. This entry is like any other entry in a directory; it contains a pointer to the file descriptor node (fdn) of the file being spooled and the newly generated name of the file. If the file is on the same disk as the spooler directory and it is sent directly to the spooler, the entry in the spooler directory is simply a link to the original file. If the file is on another disk, the operating system makes a copy of the original file on the same disk as the spooler directory and makes an entry for that copy of that file in the spooler directory. Finally, if the user sends the file through a pipe, the spooler command copies the output from the pipe into a file and makes the corresponding entry in the spooler directory.

One consequence of having the entry in the spooler directory be a link to an existing file is that the linking process causes the date and time of the last modification of the file to change whereas the copying process does not. Thus, the spooler command updates the date and time of a file that is on the same disk as the spooler directory and is sent directly to the spooler. A user who wishes to avoid changing the date and time can always force the spooler to make a copy of the file by sending the file to the spooler through a pipe:

```
list <file_name> ^ <splr_name>
```

If you do send a file directly to a spooler, be careful not to make any changes in the text of the original file before printing is complete. Because the directory entry is a link in such a case, any changes in the

original are also changes in the file being spooled. Once again, if you wish to force the spooler to make a copy of the original file so that you can change it, send the file through a pipe as illustrated in the previous paragraph.

Once you have initiated a spooler, you can spool files even if a printer is not attached to the appropriate port. The files remain in the spooler directory until the printer is functional.

4.4 Shutting Down a Printer Spooler

Under normal conditions you should initiate a standard printer spooler each time you boot the system. Before shutting down the system you should send a "pstop" command to each active spooler. The spooler finishes printing any active job before it stops. The syntax of the "pstop" command is

```
pstop <splr_name>
```

The "pstop" command, like the "/etc/insp" command, acts on only one device at a time. You must issue a separate command for each spooler. In response to a "pstop" command the system sends an interrupt to the background task created by the "/etc/insp" command. The background task finishes printing the current file (if one exists), then deletes the ".mrk*splr?" file in the appropriate spooler directory.

If either you forget to issue the "pstop" command before shutting down the system or the system crashes, the spoolers may be left in a peculiar state. When the system goes down, through normal shut-down procedure or through a crash, all tasks are, of course, interrupted. Thus, even if you do not issue a "pstop" command, the background tasks associated with each spooler disappear whenever the system shuts down. However, the ".mrk*splr?" files are not automatically deleted when the system shuts down. You may, therefore, find yourself in the situation of having ".mrk*splr?" files, but no background tasks associated with the task IDs the files contain. If you try to issue the "/etc/insp" command in such a case, the operating system responds

```
File '.mrk*splr?' already exists - spooler not invoked.
```

If this happens, simply issue a "pstop" command to the spooler in question. In such a case, because there is no background task to receive the interrupt or to delete the appropriate ".mrk*splr" file, the

"pstop" command assumes the responsibility of finding and deleting the troublesome file. Once this file is deleted, the "/etc/insp" command can successfully invoke the spooler.

4.5 Summary of Routine Spooler Use

The following table summarizes the steps in making a printer spooler functional.

Table 4-1. Spooler-related Tasks for the System Manager

Task	Frequency	Command
Select a device	Once	"makdev" or "link"if necessary (See Section 4.2.1)
Create command	Once	link /etc/print /bin/<splr_name>
Create directory	Once	crdir /gen/spooler/<splr_name>
Activate spooler	Every boot	/etc/insp <splr_name>
Terminate spooler	Every shutdown	pstop <splr_name>

4.6 Repairing a Damaged Printer Spooler

Occasionally, the necessary links between the files that control your printer spoolers may be broken. This problem is most likely to manifest itself when you update the spooler program. You may find, after updating your system disk, that the bugs which were supposed to have been fixed are not fixed, or that the enhancements which were supposed to have been made are not on your system.

In either case you should first check to see if the links that are essential to the functioning of the spooler have been broken. You can establish this by doing a "dir +l" on the directory "/etc". The entries for the files "prcon" and "print" should look like this (of course, the date and time differ from those shown here):

```
prcon   25   6    rwx--x   system   14:14 Apr 16 1987
print   28  <N>   rwx--x   system   14:14 Apr 16 1987
```

The link count for each file is the second number in the entry. The link count for "prcon" should always be 6. The link count for "print", <N>, varies depending on the number of spooler commands you create. The number <N> should be 1 plus the number of spooler commands you have linked to that file. If either link count is incorrect, you should completely remove the spooler from the system and recreate it from the master disk. The following series of commands deletes the necessary files:

```
chd /etc
kill insp prcon print
chd /usr/bin
kill end idle next pstop purge rerun
```

In addition you must delete any spooler commands that you created. These commands are the ones in "/bin" that you linked to "/etc/print". It may be wise also to delete the spooler directories such as "/gen/spooler/ppr". Before you can delete them with the "kill +d" command, you need to delete all the files they contain.

Once you have completely removed the old spooler from your system, you must reconstruct the spooler. To do so, place the master system disk in a floppy drive and execute the following commands:

```
chd /
backup <drive_name> +lRnp
```

The "backup" command prompts you for permission before restoring to the hard disk any file on the floppy disk which is newer than the file of the same name on the hard disk or for which no corresponding file exists on the hard disk. These files may include files that are not at all related to the spooler and which you may or may not choose to restore. However, you should type a 'y' in response when it asks for permission to restore the following files:

```
/etc/insp
/etc/print
/etc/prcon
/usr/bin/purge
```

When the "backup" command terminates, you must perform the usual steps, described earlier in this section, to create your own individual spooler commands and spooler directories.

Chapter 5

The Password File

5.1 Introduction

As system manager you are responsible for maintaining the list of users, which is stored in the file "/etc/log/password". This file must contain an entry for each user.

5.2 Structure of the Password File

Each entry in the password file has following form:

<user_name>:[<password>]:<ID>:<home_dir>:[<login_program>]

To add a user to the system you must edit the password file and place an entry for that user in the file. This entry must contain a user name, a user ID, and a home directory. The other fields in the entry are optional.

5.2.1 User Name

A user name is the name that a user types in response to a login prompt in order to gain access to the system. It is best to assign a user name which actually identifies the person to whom it refers. Obvious choices are the user's first name, last name, or initials. The operating system imposes certain restrictions on a user name:

1. The name must consist entirely of lowercase letters.
2. The name must contain no more than thirty-one characters.
3. The name must be unique.

The operating system does not check the validity of a user name. However, it assumes that these restrictions have been honored. Violation of the rules may therefore cause unpredictable results.

You can assign a user name by editing the password file.

5.2.2 Password

A password is a safeguard against unauthorized access to the system. While a password is not essential, it is recommended that you assign one to each user. The user then has the option of changing the password by using the "password" command.

The operating system imposes no restrictions on passwords. The following guidelines, however, help ensure the security of the system:

1. Passwords should be five or six characters long. (Longer passwords are accepted, but in any case only the first sixteen characters are significant.)

2. Passwords should be a random mixture of letters and numbers.

3. The letters used should always be lowercase.

You assign a password to a user by invoking the "password" command with an argument. The syntax for the command is

```
password [<user_name>]
```

Only the system manager may use the optional argument. In response to this command the operating system asks for the password. When you type the password, the letters do not appear on the screen. They are deliberately suppressed in order to maintain the secrecy of the password. The operating system asks you to retype the password in order to verify the entry. If what you type the second time does not match what you typed the first time, it responds

```
Retry different - password unchanged.
++
```

If, however, you type the same password both times, the operating system enters an encrypted form of the password in the password file. Thus, when a user lists the password file, it is obvious which users have passwords because their password fields are not empty. However, because the passwords are encrypted, it is not obvious what they are.

If any user ever forgets his or her password, you can assign a new one by invoking the "password" command with an argument.

5.2.3 User ID

You must assign a user ID to each user. User IDs can be in the range from 1 to 32,000 inclusive. The system manager's ID must always be 0. The system comes with a second user, "bin", whose user ID is 1; however, that ID can be changed.

You can assign an ID by editing the password file.

5.2.4 Home Directory

The fourth field in each entry in the password file contains the name of the user's home directory. The home directory is the directory that a user enters by default upon logging in. Each user must have a home directory. In general, the home directory is named "/usr/<user_name>". If no home directory is in the password file, the user receives the following message when attempting to log in:

Cannot access login directory.

The operating system then sends another login prompt to the terminal.

You can assign a home directory by editing the password file. However, you must also create the home directory using the following command:

crdir /usr/<user_name>

It is a good idea to make the user the owner of the home directory with the following command:

owner <user_name> /usr/<user_name>

Otherwise, the user cannot set the permissions for the home directory. By default, when setting the permissions on a newly created directory, the system grants read, write, and execute permissions to all users. Whether or not the default permissions are acceptable depends entirely on your environment.

5.2.5 Login Program

The last field in an entry in the password file contains the name of the login program, which is the program that begins execution when the user logs in. If the login program does not exist, or if for any reason the operating system cannot execute the specified program, a message to that effect is sent to the terminal, followed by a login prompt. If the field is empty, the default is the shell program.

You can assign a login program by editing the password file.

5.3 Original Password File

As shipped the UniFLEX Operating System has two entries in the password file:

```
system::0:/:
bin::1:/:
```

Each of these entries contains only the required fields; the optional fields are absent. Thus, as created, your system recognizes two users: "system" and "bin". Neither user has a password. The user ID for system is 0 (as it must be); the user ID for bin is 1. The home directory for both users is the root directory. The login program for both is, by default, the shell program.

One of your first tasks as system manager should be to assign passwords to these users. Use the "password" command to make these assignments.

5.4 Adding a User to the System

To add a user to the system you must make an entry in the password file that contains three things: a unique user name, a unique user ID, and a home directory. In addition you must be sure to create the appropriate home directory, to set the permissions on that directory as befits your working environment, and to make the user its owner. It is also wise to assign a password to the new user.

Suppose you want to add a user with the user name "laurie" to the system. You can maintain complete control over the choice of user ID, the name of the home directory, and the permissions on the home directory by following the procedure described in this section. Alternatively, you can use the "addusr" command (see 68xxx UniFLEX Utility Commands).

To add a new user yourself, you must first make a new entry in the password file. Before adding the new user to the file, select a user ID which is not already in use. If you selected the number 100, you would add the following line to the password file (see UniFLEX Text Editor for information on editing a file):

```
laurie::100:/usr/laurie:
```

Next you should create the directory "/usr/laurie" and make "laurie" the owner of that directory:

```
crdir /usr/laurie
owner laurie /usr/laurie
```

The operating system creates a directory with read, write, and execute permissions for all users. Whether or not you feel a need to make the permissions more restrictive will depend on your environment. The "addusr" command denies other users write permissions in a user's home directory.

Now the new user can log in. Her home directory is "/usr/laurie"; her login program is, by default, the shell program.

To assign a password use the "password" command, which enters an encrypted form of the password in the password file.

5.5 Deleting a User

Eventually, for one reason or another, you will need to remove a user from your system. First, you must determine if you want to save any of the files in that user's directories. If you do, copy them to another directory. Next, delete the unwanted files and directories from the system. It is not actually essential to sort through the files this way before removing the user from the system. However, it is wise to do so

before much time has elapsed so that the content and purpose of the files are fresh in your mind.

The process of removing the user from the system is quite simple. All you need to do is edit the file "/etc/log/password" and delete the line which contains the relevant entry. You may want to use the "delusr" command to automate the whole procedure (see 68xxx UniFLEX Utility Commands).

Chapter 6

Files and Devices

6.1 Introduction

Information on the operating system is organized into files. A file is a collection of information or data kept on the system and given a name for later reference. You can store information in several kinds of files: regular files (also referred to as ordinary files), contiguous files, directories, pipes, and devices (also referred to as special files).

Whenever you create any kind of file, the operating system assigns to it a file descriptor node (fdn). The fdn contains all the information that the operating system needs to know about a file. This information includes but is not limited to the type of file, the owner of the file, the size of the file, and, if appropriate, the addresses of all the blocks that are a part of the file.

6.2 Regular File

A regular file is simply a collection of data which resides either in memory or on a mass-storage hardware-device. The operating system allocates space for the data one block (512 bytes) at a time, as data are added to the file. Thus, the data may be scattered about the disk, depending on the sequence in which blocks are released from the free list. You can create a regular file by invoking either the "create" command or the "edit" command, or by redirecting output from any command to a nonexistent file (see "shell" in 68xxx UniFLEX Utility Commands).

6.3 Contiguous File

A contiguous file is similar to a regular file, but the blocks allocated for data are contiguous. You specify how much space the operating system should reserve for contiguous files when you format the disk (see 68xxx UniFLEX Utility Commands). An assembly language program can create a contiguous file with the "create_contiguous" system call.

Not all versions of the operating system support contiguous files.

6.4 Directory

A directory is a specially constructed file that contains the names and identifies the location of other files. Directories provide a means of organizing a group of related files in one place. For instance, the directory "bin" contains the name and identifies the location of many of the binary files provided with the operating system. You can create a directory with the "crdir" command. When "crdir" creates a directory, it puts two entries in it: one called ".", which references the directory itself; one called "..", which references its parent directory.

6.5 Pipe

A pipe is a special kind of file that allows one-way communication between a task and one of its child tasks (see Section 6.3 of the 68xxx UniFLEX Programmer's Guide). A pipe takes output from one task and uses it as input to the other task connected to the pipe. An assembly language program uses the "crpipe" system call to create a pipe. You can create a pipe from a UniFLEX command line by redirecting output to a pipe with the symbol '^' or '|' (see "shell" in 68xxx UniFLEX Utility Commands). A pipe that is created from the command line uses the output from one task as the input to another task. For instance, the command

```
page test ^ spr
```

tells the operating system to use the output from the "page" command as the input to the "spr" command.

6.6 Device

A UniFLEX device is a special kind of file. Like other kinds of files, a device has an fdn, but the information in the fdn tells the operating system that this particular file does not contain any data. Rather, the first byte of the fdn of a device identifies the device as a character, block, pseudoterminal, or network device. This byte is followed by two

numbers: a major device number and a minor device number. The major device number identifies the type of device within the broader classification determined by the character--for example, a block device could be a hard disk or a floppy disk. The minor device number is an identification number which associates the device with one of several devices of the same type--for example, the third floppy disk device.

Each version of the UniFLEX Operating System is capable of supporting a particular array of devices. (Consult the table of standard devices that comes with the 68xxx Hardware Setup Notes to determine which devices your system can support.) Both the number and kind of devices that an operating system supports are system-dependent. This section describes each type of device supported by UniFLEX. Your operating system may not support all of them.

6.6.1 Character Device

A character device is a device which the operating system accesses one character at a time, such as a terminal. Printers and tape devices are also character devices. All versions of the UniFLEX operating system support character devices.

6.6.2 Block Device

A block device is a device which the operating system accesses a block (512 bytes) at a time. Each block device must have a character device associated with it so as to support some functions, such as formatting, which are not block-oriented.

In general, block devices are random-access mass-storage devices. For example, all disks are block devices. However, some versions of the operating system support a special kind of block device called a RAM disk that allows the operating system to use available random-access memory (RAM) to emulate a disk. The system can access a copy of a file that is on a RAM disk much more quickly than it can access the original copy on a hard disk.

All versions of the UniFLEX operating system support block devices, but not all versions support RAM disks.

6.6.3 Pseudoterminal Device

A pseudoterminal device is a device which allows one program to communicate with another task as if it were communicating with a terminal. The task which creates the pseudoterminal is the "master task"; the task or tasks with which the master task communicates are the "slave tasks" (see "create_pty" in 68xxx UniFLEX Introduction to UniFLEX System Calls). Not all versions of the UniFLEX operating system support pseudoterminal devices.

6.6.4 Network Devices

A network device is a device used by the operating system to support distributed file systems. Not all versions of the UniFLEX operating system support network devices.

6.7 Creating Devices

Some devices are "made" when you first create your system. The operating system places them in the device directory, "/dev". A long listing of these devices shows not only the type of device but also the major and minor device numbers. You can see a long listing by invoking the command

```
ls /dev +1
```

Every operating system can support more devices than it initially creates. You can, within system-dependent limits, add more devices to your system with the "makdev" command (see "makdev" in 68xxx UniFLEX Utility Commands). It is not enough to create a RAM disk; you must also format it with the "ramdisk" command (see 68xxx UniFLEX Utility Commands).

Chapter 7

Important Directories

7.1 Introduction

The procedure which builds a system disk creates several files and subdirectories in the root directory. This chapter describes the kind of material found in each of them.

7.2 /uniflex

The file "uniflex" contains the kernel of the operating system. In addition, it defines certain system parameters--such as the identities of the root device, the paging device, and the pipe device--which you can alter with the "tune" command (see Chapter 9).

The information field of this file details which UniFLEX modules are supported by your operating system (e.g., pseudoterminals, networks, RAM disks). You can read the information field by invoking the following command:

```
info /uniflex
```

For more information on the various UniFLEX modules see the Preface.

7.3 /.badblocks

The file ".badblocks" (the bad-blocks file) is created as an empty file with all permissions turned off. The "badblocks" command uses it while sequestering damaged blocks from the operating system (see Section 3.5.2). Because the bad-blocks file must always be in a particular location on the disk, the operating system will not let you delete, rename, or move it.

7.4 /act

The directory "act" contains files related to system accounting. By default, when you boot the system, the operating system creates a file in "/act" called "utmp". When the system is in multi-user mode, the operating system writes the name and terminal number of each user who is using the system in this file. It also enters the time at which the user logged in. When the system enters single-user mode, the file is truncated to a length of 0.

The "who" command reads this file to obtain its output. If you ever forget which mode you are in, you can simply execute the "who" command. If the only response is a system prompt, no information is in the file "utmp", so the system is in single-user mode. Or, you can execute the following command:

```
dir +1 /act
```

If the length of the file "utmp" is 0, the system is in single-user mode.

If you create a file named "history" in the directory "/act", the operating system uses it to maintain an account of the use of the system (see Section 3.10). Additional accounting software (available in the package User/System Accounting) makes extensive use of this directory.

7.5 /bin

The name "bin" is short for binary files. This directory contains the most commonly used UniFLEX commands, such as "ls", "copy", and "kill".

7.6 /dev

The directory "/dev" contains the names of all the devices that are available on your system. These devices include physical devices such as disks, terminals, and printers, as well as logical devices, such as pseudoterminals and RAM disks.

You can, within the limits of your operating system, add devices to this directory with the "makdev" command. A list of devices that your operating system can support accompanies the 68xxx Hardware Setup Notes.

Several devices are common to all systems:

DISK A link to the root device, which is defined in the file "/uniflex". If you want to change the identity of the root device, you must use the "tune" command (see Chapter 9).

null The system's bit bucket. You can always write to the bit bucket, but when you do, the data are lost from the system. If you try to read from the bit bucket, the operating system returns an end-of-file condition.

pmem A device which allows certain utility programs to access physical memory as if it were a file.

smem A device which allows certain utility programs to access system memory as if it were a file.

swap A link to the paging device, which is defined in the file "/uniflex". If you want to change the identity of the paging device, you must use the "tune" command (see Chapter 9).

7.7 /etc

Except for the commands "login" and "print", the commands in the directory "/etc" are intended for use only by the system manager. Such commands include "diskrepair", "makdev", and "shutup".

This directory also includes several important files and a subdirectory.

7.7.1 /etc/ttylist

The "ttylist" file, "/etc/ttylist", contains an entry for each terminal port on the system, up to a maximum of sixty-four entries. Each entry is created with the following format:

<sign><baud_rate><nn>:[<terminal_type>]:[<optional_info>]

A brief explanation of this format follows. For more information see "set_termcap" and "crt_termcap" in 68xxx UniFLEX Utility Commands.

<sign> A plus sign, '+', or a minus sign, '-'. A plus sign enables the port for login; a minus, disables it. When the operating system enters multi-user mode, the "init" program sends a login prompt to every terminal port that is enabled for login.

<baud_rate> A code specifying the baud rate for the terminal port in question. The code must be a value from the following table:

Code	Speed	Code	Speed
space	Hardware default	8	1200
1	75 or 38400	9	1800
2	110	a	2400
3	134.5	b	3600
4	150	c	4800
5	200	d	7200
6	300	e	9600
7	600	f	19200

nn A two-digit number representing the terminal port.

<terminal_type> The type of terminal attached to the port. Initially, this field is blank. In order to use display-oriented software on a terminal, you must define the terminal type as one of those defined in the file "/etc/ttycap" (see Section 7.7.2).

If you define a terminal as "modem", a "login" command from that terminal prompts for the terminal type after verifying the combination of user name and password. You may respond to the prompt in one of three ways: by typing one of the strings defined in the file "/etc/ttycap"; by typing a carriage return, which leaves the terminal type as "modem" (see Section 7.7.2); or by typing a question mark, '?', which tells the "login"

command to display a list of all terminals defined for the system.

Once you have logged in, you can change the terminal type by invoking the "env" command (see 68xxx UniFLEX Utility Commands).

<optional_info> Optional information, which is normally the name of the person most commonly using the terminal. This field has no functional meaning and need not be present.

Each time you add a terminal to the system you should edit this file so as to enable the port for login and to add an entry describing the type of the terminal.

7.7.2 /etc/ttycap

The file "ttycap" is a specially constructed file which defines the capabilities of a variety of terminals. As supplied with the operating system, this file contains entries only for the terminals we have experience with. If the type of terminal you need is not in the file, you can add it (see "crt_termcap" in 68xxx UniFLEX Utility Commands).

The default "ttycap" file contains an entry for the terminal type "modem", which it defines as an ANSI standard terminal.

7.7.3 /etc/termcap

The termcap file, "/etc/termcap", is a file created by combining the information in the files "/etc/ttycap" and "/etc/ttylist". This file makes it possible for the same program to operate on many different terminals regardless of their individual characteristics. The operating system creates this file when you build your original system. You can alter the file by using either the "crt_termcap" or the "set_termcap" command.

7.7.4 /etc/.init.control

The init-control file, "/etc/.init-control", controls the functioning of the first task ("init") executed by the operating system. The contents of the init-control file determines, for example, the interval at which the operating system updates all disks; the action the system takes on receiving a particular interrupt; and whether the system comes up in single-user mode or goes directly to multi-user mode. You can alter the init-control file supplied with your system to suit your needs, but you should do so only with extreme caution (see Chapter 2).

7.7.5 /etc/format.control

The format-control file, "/etc/format.control", defines a default format command for each standard disk likely to be used on the system. This file is used by the "backup" and "format" commands. You can always override the default.

The format-control file may contain either instructions for formatting all media in all drives the same way or different instructions for formatting media in different drives. In the first case, the first line of the file should contain the file specification of the formatting program (e.g., "/etc/formatfd"); the second, the model name. (For a discussion of file specifications see page 6 of The UniFLEX Operating System.) In the second case, the first character of the first line of the file must be an asterisk, '*'. The file specification for a device must immediately follow the asterisk. The second and third lines of the file should then contain the file specification of the formatting program and the model name, respectively, for formatting media in the specified device. This pattern of three lines may be repeated to specify formatting instructions for other devices. The first character of any line specifying a device must be an asterisk. A sample format-control file, which defines format commands for the devices "fd0" and "fd1", follows.

```
*/dev/fd0
/etc/formatfd
IFD-DD
*/dev/fd1
/etc/formatfd5
IFD-8-08
```

7.7.6 /etc/log

The directory "/etc" contains a subdirectory called "log", which contains several files used by the operating system in the course of its daily operations.

7.7.6.1 /etc/log/motd

The file "motd" contains the message of the day. Whenever a user logs in, in response to a login prompt, the "login" program reads this file and sends its contents to standard output before issuing the system prompt. Thus, if you want to send a message to all users, you can enter it in this file (see Section 3.3).

7.7.6.2 /etc/log/password

The password file, "/etc/log/password", contains a list of users, their passwords (encrypted), their user IDs, their home directories, and their login programs. Because many UniFLEX programs use the password file, it is essential to maintain its integrity (see Chapter 5).

7.8 /gen

The directory "/gen" contains three directories: "errors", "help", and "spooler".

7.8.1 /gen/errors

The directory "errors" contains one file, "system", which is a binary listing of UniFLEX error numbers and their corresponding messages.

7.8.2 /gen/help

The directory "help" contains brief descriptions of the usage of the UniFLEX commands. The "help" command uses the information in this directory.

7.8.3 /gen/spooler

The directory "/gen/spooler" is the directory in which you must create a directory for each of the spoolers on your system (see Chapter 4) in order for them to function properly.

This directory also contains the subdirectory "at", which is used by the "atexecute" command. The "at" directory contains a file called "holidays" which defines holidays for the "at" daemon (see 68xxx UniFLEX Utility Commands).

7.9 /lib

The directory "/lib" contains files which are sources containing definitions of both symbols and structures that can be used by assembly language programs. Run-time libraries for compilers are usually located in "/lib".

7.9.1 /lib/rel20.errs

The file "rel20.errs" is a binary listing of error messages used by the "rel20" command. It is present only on 68020 systems.

7.9.2 /lib/rel68k.errs

The file "rel68k.errs" is a binary listing of error messages used by the "rel68k" command.

7.9.3 /lib/Syslib68k

The file "Syslib68k" contains relocatable versions of all the files in "/lib" whose names begin with the string "sys". The linking-loader may search this file in a final attempt to resolve externals--see Section 8.1 of 68xxx Relocating Assembler and Linking-Loader.

7.9.4 /lib/sysacct

The file "sysacct" defines a structure for storing and retrieving accounting information.

7.9.5 /lib/sysdef

The file "sysdef" defines the correspondence between the names and numbers of UniFLEX system calls.

7.9.6 /lib/syserrors

The file "syserrors" defines the correspondence between the names and numbers of UniFLEX errors. It also contains a brief definition of the most general cause of each error.

7.9.7 /lib/sysfcntl

The file "sysfcntl" defines the correspondence between the names and numbers of the subfunctions used by the "fcntl" system call. It also defines the constants used by this system call.

7.9.8 /lib/sysints

The file "sysints" defines the correspondence between the names and numbers of UniFLEX interrupts. It also contains a brief definition of each interrupt.

7.9.9 /lib/sysmessages

The file "sysmessages" defines the structure of the buffer returned by the "msg_status" system call.

7.9.10 /lib/syspty

The file "syspty" defines the correspondence between the names and the numbers of the subfunctions used by the "control_pty" system call. It also defines the constants used by this system call.

7.9.11 /lib/sysrump

The file "sysrump" defines the correspondence between the names and the numbers of the subfunctions used by the "rump" system call.

7.9.12 /lib/sysstat

The file "sysstat" defines the structure of the buffer returned by the "ofstat" and "status" system calls. It also defines the file-permission flags.

7.9.13 /lib/systime

The file "systime" defines the structures of the buffers returned by the "time" and "ttime" system calls.

7.9.14 /lib/systty

The file "systty" defines the structure of the buffer returned by the "ttyget" and "ttyset" system calls. It also defines the constants used by these system calls.

7.9.15 /lib/sys68881

The file "sys68881" defines the structure of the buffer used by the "FPU_exception" system call.

7.9.16 /lib/std_env

The file "std_env" describes the standard hardware-specific environment of your particular system. It contains a series of options which the linking-loader ("load68k") automatically processes before it processes any options from the command line. The options specify information about such things as the hardware page-size and the starting address of the text and data segments. The linking-loader uses the file to get the basic information it needs in order to load any module. If necessary, you can override those options in the file that take arguments by specifying the same options with different arguments on the command line. You can negate the effect of a single-character option specified in the standard-environment by using the same option on the command line, preceded by a minus sign, '-' (see Chapter 7 in 68xxx UniFLEX Relocating Assembler and Linking-Loader).

7.10 /lost+found

The directory "/lost+found" is for use by the "diskrepair" command (see Chapter 7). If "diskrepair" finds any unreferenced files while it is checking the disk, it places them, by default, in "/lost+found". Although the files are no longer where they belong, the information they contain is preserved, as is the name of the owner.

7.11 /tmp

The directory "/tmp" is for use by programs which need to create temporary files--that is, files that you do not need after you execute the program. Some UniFLEX commands use this directory. Any programs you write may also use it. Thus, if the system happens to crash while any programs that use temporary files are running, the temporary files are all in one place and you can easily delete them all when you reboot the system. You can automatically do so by putting the following command in your startup file (see 3.12):

```
kill /tmp/*
```

Alternatively, you can put the equivalent command in your init-control file (see Chapter 2).

7.12 /usr

The directory "/usr" is the directory in which you will generally create a home directory for each user on the system (see Section 5.2.4). As created by the procedure that builds the operating system, however, "/usr" contains only one file--the subdirectory "/usr/bin". This subdirectory contains the less commonly used UniFLEX commands, such as "history", "free", and "info".

7.13 /usr0, /usr1, /usr2, /usr3

The empty directories "/usr0", "/usr1", "/usr2", and "/usr3" provide nodes for the mounting of devices on the system (see Section 3.8).

Chapter 8

Errors Fatal to the Operating System

8.1 Introduction

The errors documented in this chapter are fatal to the operating system--that is, you must reboot to gain control. Each error first sends an error message to the console then cleanly halts the system so that the disk remains intact.

8.2 Errors during Initialization

The following errors may occur when you try to boot the system:

Invalid license.
The file "/uniflex" is not licensed for the machine you are using.

I/O error on root device during initialization.
The operating system could not read the root device. First, verify that the disk in the root device is not write-protected. If it is not, boot from your master disk, mount the hard disk, and use the "tune" command (see Chapter 9) to determine whether or not the specification of the root device is correct (see Section 9.4.16). If it is incorrect, change it. If the specification is correct, the message indicates a hardware failure.

No paging space.
The disk in the paging device does not have any paging space. The operating system cannot function without paging space. Boot from your master disk, mount the hard disk, and invoke the "alterpage" command from the hard disk to adjust the size of the paging space (see 68xxx UniFLEX Utility Commands).

Out of memory during initialization.
The operating system was able to set up all the internal tables it needed, but it ran out of memory when it tried to initialize the first task. Either the system does not have sufficient memory or the sizes assigned to the various tables are unrealistically large. You can ascertain how much memory is available by using the 'M' command while you are still in the ROM (see Section 1.2.2). If sufficient memory is available, boot to your master disk and invoke the "tune" command to determine how much memory is allocated for the

tables. Then use "tune" to change the parameters--such as "buffers" and "locked_recs"--which control the tables' sizes (see Section 9.4).

Overflow in system table.
While the operating system was establishing its internal tables, it ran out of memory. Either the system does not have sufficient memory or the sizes assigned to the various tables are unrealistically large. You can ascertain how much memory is available by using the 'M' command while you are still in the ROM (see Section 1.2.2). If sufficient memory is available, boot to your master disk and invoke the "tune" command (see Chapter 9) to determine how much memory is allocated for the tables. Then use "tune" to change the parameters--such as "buffers" and "locked_recs"--which control the tables' sizes (see Section 9.4).

8.3 After Loading the Operating System

The errors described in this section occur only after the file "/uniflex" is successfully loaded in memory.

All pages are locked.
Certain parts of the operating system are always locked in memory. In addition, a task can lock pages in memory. This error occurs if the system needs memory but finds that all pages are locked. You can alleviate the problem by reducing the number of pages locked by tasks under your control. If the problem persists, please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Error reading system information record (SIR).
The operating system attempted to update a mounted device but was unable either to read from or to write to the SIR even though it had been able both to read from and write to it when mounting the device. This error is indicative of a hardware failure.

Error reading system information record (SIR) on paging device.
The operating system detected an I/O error when it tried to read the SIR on the disk in the paging device. Use the "tune" command (see Chapter 9) to determine whether or not the specification of the paging device is correct (see Section 9.4.13). If it is incorrect, change it. If the specification is correct, the message is indicative of physical damage on the disk in the paging device. You must reformat the disk.

File descriptor node (fdn) gone.
The internal data structure that the system uses to keep track of files is damaged. This error is indicative of a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Memory page already free.
One part of the operating system tried to release a page of memory that some task indicated was in use. However, another part of the operating system indicated that the memory was already free. This error is the result of a logical inconsistency and probably indicates a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Mount gone.
The operating system was following the file specification of a file that is on a mounted device, but the information about the mounted device was damaged. This error is indicative of a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

No buffer available for the name of the program.
When system accounting is enabled, the operating system must store the name of each program that is executing so that it can write the appropriate information to the accounting files. The operating system allocates buffers for this purpose--establishing as many buffers as the number of tasks that can simultaneously run on the system. This error indicates that all of these buffers are in use. It is the result of a logical inconsistency and probably indicates a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

No destination page during fork.
When the operating system executes a "fork" system call, it first makes a complete copy of the task. It puts as much of this copy as possible in memory; the rest goes in the paging space. This error indicates that the operating system was unable to locate the part of the task that it wrote to the paging space. It is the result of a logical inconsistency and probably indicates a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Out of memory and paging space.
No memory is available to the operating system, and no paging space is available to free up some memory.

Out of paging space.
The system needed to transfer some data from memory to the paging space and thought that paging space was available. However, it was unable to find any. You can either increase the size of the paging space with the "alterpage" command or use the "tune" command (see Chapter 9) to decrease the number of tasks the system can support at one time.

Overflow in task table.
The operating system thought it could create an entry in the task table for a task, but when it tried to do so, it found that the task table was full. This error is the result of a logical inconsistency and probably indicates a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Overflow in text table.
The users tried to execute too many shared-text programs simultaneously. You can change the number of shared-text programs allowed with the "tune" command (see Section 9.4.19).

Paging page already free.
One part of the operating system tried to free a page on the paging device, but another part of the operating system thought that that page was already free. This error is the result of a logical inconsistency and probably indicates a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Underflow in TMAT page counter.
One of the internal data structures for the task (the task memory allocation table) has been damaged. This error is indicative of a hardware failure. Please take a memory dump if possible (see Section 3.9) and send it to Technical Systems Consultants for analysis.

Chapter 9

Fine-tuning the UniFLEX Operating System

9.1 Introduction

The "tune" command alters certain parameters which govern the behavior and performance of the UniFLEX Operating System. Because different systems are used and stressed in different ways, the optimal settings for these parameters vary from site to site. Careful tuning of the operating system allows you to get the best performance from your system.

The "tune" command only changes the values of the parameters in the specified file. It does not alter the copy of that file that is in memory. Therefore, the changes have no effect until you boot the operating system from the modified version.

9.2 Invoking the "tune" Command

The syntax for the "tune" command is

```
/etc/tune <file_name> [<param_list>] [+pPq]
/etc/tune <file_name> [+pPr]
```

9.2.1 Arguments

The "tune" command takes one obligatory and one optional argument:

<file_name>	The name of a file that contains a copy of the operating system.
<param_list>	An optional list of the parameters to change and of the values to assign to them.

9.2.2 Format for Arguments

The format of each element of the optional argument, <param_list>, is as follows:

<param_name>=<num>

Sections 9.3 through 9.6 explain what parameters you can adjust.

9.2.3 Options Available

- r Operate in read-only mode. If you lack write permission for the specified file, the "tune" command automatically invokes this option. The 'r' option is incompatible with the use of a list of parameters. If you specify both, "tune" returns an error.
- p Include parameters associated with scheduling (see Section 9.6). You may change these parameters only if you invoke the "tune" command in interactive mode.
- P Include parameters associated with "phys" segments (see Section 9.5). You may change these parameters only if you invoke the "tune" command in interactive mode.
- q Operate in quiet mode--that is, suppress all messages. You can use this option only when you invoke the "tune" command in automatic mode.

9.2.4 Modes of Operation

The "tune" command operates in three modes: read-only, interactive, and automatic. The syntax you use to invoke the command determines the mode.

9.2.4.1 Read-only mode

In read-only mode "tune" displays the current value, in the specified file, of the parameters you select by your choice of options. You cannot adjust any parameters in read-only mode.

As system manager you can execute "tune" in read-only mode by specifying the 'r' option. A user who does not have write permission for the file being tuned can execute "tune" only in read-only mode. In such a case the 'r' option is not necessary; "tune" automatically executes in read-only mode.

9.2.4.2 Interactive mode

If you have write permission for the specified file and specify neither a parameter list nor the 'r' option, "tune" executes in interactive mode. In this mode it displays current values one by one. To change the value of a parameter, enter the new value and a carriage return following the display. To leave the value as is, type just a carriage return.

Two kinds of adjustable parameters, those associated with scheduling and those associated with "phys" segments, can only be changed from interactive mode. However, the "tune" command does not prompt for values for these parameters unless you use the appropriate options to request it to do so (see Section 9.2.3).

The "tune" command imposes certain restrictions on the values of the parameters it alters. If you try to set the value of a parameter outside restrictions limits from interactive mode, "tune" responds with an error message and does not let you proceed until you enter a valid value.

9.2.4.3 Automatic mode

Instead of going through the entire list of parameters interactively, you can specify values for most of them from the command line. (As mentioned previously, you cannot change parameters related to scheduling or to "phys" segments from the command line.) If you specify a value for any parameter on the command line, the values of parameters not specified do not change. After making the changes, "tune" displays a list of all parameters and the values you specified. If any value violates the restrictions mentioned previously, "tune" displays a message to that effect and sends a bell (control-G) to the terminal. Although the display shows whatever values you specified, "tune" does not change the value of a parameter in the file unless the change is valid.

9.3 Adjustable Parameters--an Overview

The object of adjusting parameters is to make your operating system perform optimally. This section summarizes the functions of all the adjustable parameters and gives the minimum, maximum, and default values for each one. Detailed descriptions of the parameters and of the advantages and disadvantages of changing them are discussed in the following sections.

9.3.1 Functions of the Adjustable Parameters

Table 9-1 briefly describes the parameters that the "tune" command can alter in automatic mode (from the command line). They are discussed in more detail later in Section 9.4.

Table 9-1. Parameters Adjustable in Automatic Mode

<param_name>	Description
buffers	Number of system buffers.
DST	Flag for the observation of Daylight Savings Time (0 indicates it is not observed locally; 1, that it is).
DST_end	The last day in the year that Daylight Savings Time can end.
DST_start	The first day in the year that Daylight Savings Time can start.
DST_time	The time of day at which the switch to or from Daylight Savings Time occurs.
files	Maximum number of files that can be open at one time.
iolists	Maximum number of lists of I/O characters.
locked_recs	Maximum number of entries allowed in the table of locked files.
mounts	Maximum number of devices that can be mounted at one time.
msg_buffers	Maximum number of message buffers that can be used at one time.
msg_exchanges	Maximum number of message exchanges.
msg_size	Maximum size of each message buffer.
page_dev	Device number or name of the paging device.
page_space	Default size of the paging space.
pipe_dev	Device number or name of the pipe device.
root_dev	Device number or name of the root device.
seek_rate	Seek rate of the floppy disk drive.
tasks	Maximum number of active tasks the system can support.
text_segs	Maximum number of unique shared-text programs that the operating system can execute at one time.
timeouts	Maximum number of pending timed-events.
time_limit	Maximum CPU time allowed per user task.
time_zone	Time difference in minutes between local time and Universal Time. A positive value of "time_zone" indicates the number of minutes west of Greenwich; a negative value, the number of minutes east.
user_tasks	Maximum number of active tasks each user can create.

In addition, you can change the parameters associated with "phys" segments if you invoke the "tune" command in interactive mode with the 'P' option. (If you specify the 'P' option with either the 'r' option or with a list of parameters to change from the command line, "tune" displays the values of these parameters but does not allow you to change them.) Table 9-2 gives a brief description of these parameters.

Table 9-2. "Phys" Parameters

Name	Description
Logical Address	The base logical-address of the segment.
Physical Address	The base physical-address of the segment.
Segment Size	The number of 4-Kbyte pages per segment.

Finally, you can change the values of certain parameters associated with scheduling. These parameters are used in determining two dynamic variables for each task: "CPU utilization", which directly influences a task's priority (see Section 9.6.3); and "quantum", the number of ticks a task have in the CPU before a task of equal priority can displace it (see Section 9.6.4). A tick is 10 milliseconds.

Table 9-3. Scheduling Parameters

Name	Description
Max CPU Utilization	The maximum value to which "CPU utilization" can increase.
CPU Utilization Increment/Hit	Measure of how fast a task's "CPU utilization" increases each time the system catches it in the CPU.
CPU Utilization Decay	Measure of how quickly the value for "CPU utilization" decays when a task is not using the CPU.
Quantum Increment	Number of ticks to add to "quantum" each time a task gains access to the CPU.
Max Quantum	The maximum value to which "quantum" can increase.

9.3.2 Limits and Defaults for Adjustable Parameters

The operating system originally sets the values for all adjustable parameters when it copies the file "/uniflex" from the master disk or tape to a system disk. Table 9-4 shows the original values for all parameters that you can tune in automatic mode, as well as the limits the operating system imposes on them. You can determine the default values for your particular system by running the "tune" command in read-only mode on the "uniflex" file on your master disk.

Table 9-4. Defaults and Limits for Parameters Adjustable in Automatic Mode

<param_name>	Default	Minimum	Maximum
buffers	sd	8	192
DST	0	0	1
DST_end	303	0	364
DST_start	96	0	364
DST_time	120	0	1439
files	sd	16	1024
iolists	sd	0	512
locked_recs	32	0	Value of "files"
mounts	5	2	32
msg_buffers	256	0	32767
msg_exchanges	64	0	256
msg_size	64	0	4096
page_dev	sd	0	sd
page_space	5000	256	1000000
pipe_dev	sd	0	sd
root_dev	sd	0	sd
seek_rate	0	0	sd
tasks	sd	8	128
time_limit	0	0	32767
text_segs	20	2	20
timeouts	32	2	256
time_zone	300	-1440	1440
user_tasks	10	5	Value of "tasks"

Notes: sd = system-dependent

The default value for every parameter associated with a "phys" segment is system dependent.

The operating system defines each of the scheduling parameters for each class of task it recognizes (see Section 9.6.2). The minimum, maximum, and default values for each of these parameters are shown in Table 9-5. The values shown as maximum values are not, strictly speaking, maxima. The "tune" command does allow you to exceed them. However, if you choose larger values, the behavior exhibited by the operating system may not be what you expect.

Table 9-5. Default Values and Limits for Scheduling Parameters

Name	Default	Minimum	Maximum
CPU Personality			
Max CPU Utilization	120	4	200
CPU Utilization Increment/Hit	5	1	20
CPU Utilization Decay	4	1	10
Quantum Increment	100	10	200
Max Quantum	100	10	400
TTY Personality			
Max CPU Utilization	35	4	200
CPU Utilization Increment/Hit	2	1	20
CPU Utilization Decay	2	1	10
Quantum Increment	50	10	200
Max Quantum	~~200~~ 100	10	400
DISK Personality			
Max CPU Utilization	75	4	200
CPU Utilization Increment/Hit	3	1	20
CPU Utilization Decay	3	1	10
Quantum Increment	75	10	200
Max Quantum	150	10	400
PIPE Personality			
Max CPU Utilization	50	4	200
CPU Utilization Increment/Hit	3	1	20
CPU Utilization Decay	4	1	10
Quantum Increment	125	10	200
Max Quantum	~~175~~ 375	10	400

9.4 Parameters Adjustable in Automatic Mode

9.4.1 System Buffers

The parameter "buffers" determines the number of blocks reserved for the buffer cache. The minimum number of blocks in the buffer cache is 8; the maximum, 192. The default is system-dependent. The number of blocks reserved for the buffer cache must be a multiple of 8.

When the operating system searches the buffer cache for a particular block, it must search sequentially. Thus, the larger the buffer cache, the greater the time spent searching it. Depending on how heavily your system is used, you may see a degradation in system response as you increase the size of the cache.

The advantage of increasing the size of the buffer cache is that the system does not need to access the disk as often. The overall speed of operation may therefore increase.

9.4.2 Lists of I/O Characters

The operating system buffers both input to and output from terminals. The parameter "iolists" refers to the maximum number of buffers of I/O characters the system can support. The number of I/O buffers supplied with the operating system is system-dependent. The minimum is 0; the maximum, 512.

A single terminal can access no more than 20 buffers. The maximum useful value for "iolists" is, therefore, the number of terminals on the system multiplied by 20. All systems, however, are limited to the maximum of 512. If your system slows down noticeably when terminal activity is high, increase the value of "iolists".

9.4.3 Daylight-Savings-Time Flag

The Daylight-Savings-Time flag, "DST", indicates whether or not Daylight Savings Time is observed locally. A value of 0 indicates that it is not; a value of 1, that it is. The default value is 0.

9.4.4 Last Day of Daylight Savings Time

The parameter "DST_end" specifies the latest day in the year on which Daylight Savings Time can end, ignoring the effect of the extra day in a leap year (the system automatically makes that adjustment if appropriate). The default value is 303. The minimum value is 0 (January 1); the maximum, 364 (December 31).

For example, the algorithm currently in use in the United States ends Daylight Savings Time on the last Sunday in October. The latest date that day can be is October 31, which is the 304th day of the year. Because the parameter is zero-based, the appropriate value for "DST_end" in this case is 303.

9.4.5 First Day of Daylight Savings Time

The parameter "DST_start" specifies the latest day in the year on which Daylight Savings Time can start, ignoring the effect of the extra day in a leap year (the system automatically makes that adjustment if appropriate). The default value is 96. The minimum value is 0 (January 1); the maximum, 364 (December 31).

For example, the algorithm currently in use in the United States starts Daylight Savings Time on the first Sunday in April. The latest date that day can be is April 7, which is the 97th day of the year. Because the parameter is zero-based, the appropriate value for "DST_start" in this case is 96.

9.4.6 Time of Day for Daylight Savings Time

The parameter "DST_time" specifies the time of day (expressed as then number of minutes past midnight) at which the switch to or from Daylight Savings Time occurs. The minimum value is 0; the maximum, 1439. The default is 120 (i.e., 2:00 A.M.).

9.4.7 Maximum Number of Open Files

The parameter "files" refers to the number of open files that the operating system can support at one time. The default value is system-dependent. The minimum value is 16; the maximum, 1024.

The operating system maintains a table that tells which files are open. As the number of entries in the table increases, the time required to search the table also increases. You may, therefore, see some degradation in system response as you increase the value of "files".

On the other hand, you may see an improvement in performance as you increase the value of this parameter. The information about an open file is stored in memory in a cache of file descriptor nodes (the fdn cache). Even when a user closes the file, the information about the file stays in this cache until it is full. When a user opens a file, the operating system first looks in the fdn cache for the information it needs about the file. If the information is already there, the system need not access the disk. Thus, as the size of the fdn cache increases, the speed of the operating system may increase.

The point at which the increase and decrease in system performance balance each other depends on the way in which a particular system is used.

Although you can change the number of open files that the system can support, you cannot change the number of files that one task can open. That number is restricted to 32.

9.4.8 Maximum Number of Locked Records

Whenever a user locks a record (by invoking the "lrec" system call), the operating system makes an entry in its table of locked records so that other users can check to see whether or not the records they want to access are locked. The parameter "locked_recs" determines how many entries this table can contain. If you try to lock a record when the table is full, the operating system returns an error.

If the users on your system lock records frequently, you may want to increase the value of "locked_recs" above the default value of 32. The minimum value you can use for this parameter is 0; the theoretical maximum, 128. However, for any given system the practical maximum is the same as the value of the parameter "files" (see Section 9.4.7).

9.4.9 Maximum Number of Mounted Devices

The value of the parameter "mounts" determines how many mounted devices the operating system can support at one time. The default value for "mounts" is 5. The minimum value is 2; the maximum, 32. Although you cannot unmount the root directory, it is a mounted device.

9.4.10 Maximum Number of Message Buffers

The value of the parameter "msg_buffers" determines how many buffers the operating system can use at one time for intertask communication (see the documentation for "msg_receive" in the Introduction to UniFLEX System Calls).

The default value is 256. The minimum value is 0; the maximum, 32767.

9.4.11 Maximum Number of Message Exchanges

The value of the parameter "msg_exchange" determines how many message exchanges the operating system can open at one time (see the documentation for "msg_attach" in the Introduction to UniFLEX System Calls).

The default value is 64. The minimum is 0; the maximum, 256. No matter how many exchanges the system can support, an individual task can attach to no more than thirty-two message exchanges at a time.

9.4.12 Size of Message Buffers

The value of "msg_size" determines the maximum length of a message that you can send from one task to another (see the documentation for "msg_receive" in the Introduction to UniFLEX System Calls).

The default value is 64. The minimum is 0; the maximum, 4096.

9.4.13 Paging Device

The value of the parameter "page_dev" describes which block device the operating system is to use for paging. It stores the number as a 2-byte hexadecimal number whose first byte is the major device number of the page device and whose second byte is the minor device number of the page device (see 68xxx UniFLEX Hardware Setup Notes). To specify the value in decimal, use the following format:

page_dev = <major_dev_num>/<minor_dev_num>

To specify the value in hexadecimal use the same format, but place a dollar sign, '$', before the major device number.

You can also specify the page device by name:

page_dev = <dev_name>

The default value for "page_dev" is system dependent. Reasonable minimum and maximum values are also system dependent (see 68xxx UniFLEX Hardware Setup Notes). The "tune" command, however, does not have enough information to determine whether or not the number you specify is a reasonable value. It is, therefore, safer to specify the device by name.

Changing the paging device to a device reserved for paging may improve system performance because the head of that disk drive always remains over the paging space.

9.4.14 Default Paging Space

The parameter "page_space" tells the operating system how many 4-Kbyte pages to allocate for paging in its internal tables. The minimum value is 256; the maximum 1,000,000. The default is 5,000.

The operating system builds the internal tables before it can read the disk to determine how much paging space is truly available. If "page_space" is greater than the amount of actual paging space, you get no more space but only build a table that is larger than necessary. If "page_space" is less than the actual paging space, you cannot access all the paging space on the disk. You should therefore tune your system so that "page_space" is the same as the amount of paging space on the disk.

9.4.15 Pipe Device

The parameter "pipe_dev" tells the operating system which device to use for creating pipes. You specify the value of "pipe_dev" just as you specify the value of "page_dev" (see Section 9.4.13). The default, minimum, and maximum values are also the same.

In order for the system to function properly, the pipe device must always be the same as either the root device or the paging device.

9.4.16 Root Device

The parameter "root_dev" tells the operating system which device contains the root directory. You specify the value of "root_dev" just as you specify the value of "page_dev" (see Section 9.4.13). The default, minimum, and maximum values are also the same. Changing the root device does not affect the performance of the operating system.

9.4.17 Seek Rate of the Floppy Disk Drives

The floppy disk driver tells the hardware how fast to try to move the heads of the floppy disk drives from one track to an adjacent track. The parameter "seek_rate" specifies this rate. If the rate is too fast, the system does not function properly. If it is too slow, the system wastes time.

The default value of "seek_rate" is 0. The minimum is also 0; the maximum is system-dependent, but the largest value for any system is 255. Consult the 68xxx UniFLEX Hardware Setup Notes for the correspondence between the value of "seek_rate" and the seek rate of your system.

9.4.18 Maximum Number of Tasks Supported

Although only one task can occupy the CPU at any given time, the operating system can support more than one "active task". An active task is simply a task to which the operating system has assigned a task ID. The operating system maintains a table of active tasks. The parameter "tasks" determines how many entries this table can hold. The default value of the number of tasks supported is system-dependent. The minimum is 8; the maximum, 128.

Changing the maximum number of tasks that the system can support incurs some overhead even if the number of active tasks does not change. As the number of tasks that can be supported increases, the system may slow down.

The parameter "tasks" determines the number of tasks that the system can support. The number of active tasks allowed to an individual user is determined by a different parameter, "user_tasks".

9.4.19 Shared-text Programs

Certain UniFLEX programs that are both moderately large and frequently used--such as the shell program and the editor--are shared-text programs. No matter how many people are using these programs, the operating system needs only one copy of the program in memory. The parameter "text_segs" determines how many shared-text programs the operating system can support. The default value is 20. The minimum is 2; the maximum, 20.

9.4.20 Time Limit for Tasks

The parameter "time_limit" determines the total number of seconds a task may spend in the CPU. If a task exceeds the time limit, the operating system terminates it abnormally.

The default value, which is 0, specifies that no time limit be imposed. The minimum value is 0; the maximum, 32767.

9.4.21 Number of Time-outs

The parameter "time_outs" specifies the maximum number of pending timed-events the system can support at one time. A pending timed-event is something that the system knows it must perform after a specific amount of time has passed. Waiting for a disk to come up to speed, for instance, establishes a pending timed-event.

The default value of "time_outs" is 32. The minimum is 2; the maximum, 256. If the number of time-outs is insufficient, the operating system may crash.

9.4.22 Setting the Time Zone

When you use the "date" command to set the date and time, the operating system converts the time you enter into the number of seconds that have passed since midnight, January 1, 1980, at the zeroth meridian (in Greenwich, England). As it makes this conversion it must adjust the result for the local time zone and for Daylight Savings Time, if it is in effect. The value of the parameter "time_zone" is the number of

minutes difference between local time and Greenwich Mean Time (Universal Time). The value must represent a nonfractional number of hours--that is, it must be a multiple of 60. A positive value of "time_zone" indicates the number of minutes west of Greenwich; a negative value, the number of minutes east.

The default value of "time_zone" is 300 (the correct value for Lafayette, Indiana, the birthplace of Technical Systems Consultants). The minimum is -1440; the maximum, 1440.

9.4.23 Number of Tasks per User

The parameter "user_tasks" determines the maximum number of active tasks an individual user can create. The default value is 10. The minimum is 5; the theoretical maximum, 25. However, for any given system the practical maximum is the same as the value of "tasks". The same number applies to all users except the system manager, who may run as many tasks as the system can support.

The cost of increasing the value of "user_tasks" is negligible. You may want to lower the number of tasks allowed to each user if your system seems to be overloaded by a particular user.

Although varying the parameter "user_tasks" allows you to vary the number of tasks each user can simultaneously execute, you cannot alter the fact that any one shell program supports a maximum of five background tasks.

9.5 Parameters Associated with "phys" Segments

If you invoke the "tune" command in interactive mode and specify the 'P' option, you can alter the parameters associated with "phys" segments. A "phys" segment is a section of memory that a task can readily access by a system-dependent code. The default values for all parameters associated with "phys" segments are system dependent, as is the number of "phys" segments a system can support. The "tune" command prompts you for information relating to each "phys" segment.

9.5.1 Physical Address

You can specify the physical address of a "phys" segment either as a decimal number or as a hexadecimal number preceded by a dollar sign, '$'.

9.5.2 Logical Address

If your system does not have a memory management unit (MMU), the value you specify as the logical address of a "phys" segment should be the same as the value of its physical address. If your system does have an MMU, you can use any value for the logical address because the system calculates the correct address when you boot it. You can specify the address either as a decimal number or as a hexadecimal number preceded by a dollar sign, '$'.

9.5.3 Segment Size

The value you specify as the size of a "phys" segment must be a number between 1 and 32,767 inclusive. This number is the number of 4-Kbyte pages in the "phys" segment. A value of 0 indicates that the "phys" segment is not defined.

9.6 Parameters Associated with Scheduling

If you invoke the "tune" command in interactive mode and specify the 'p' option, you can alter the adjustable parameters associated with task scheduling.

9.6.1 Functions of the Scheduler

The scheduler determines which task controls the CPU. In order to understand how the scheduler functions (and, therefore, how the adjustable parameters influence scheduling), you must be familiar with a bit of scheduler jargon. An active task is any task to which the operating system has assigned a task ID. The system assigns a priority to each task and periodically reevaluates the priorities of all active

tasks, giving control of the CPU to the task with the highest priority (only one task can control the CPU at a time). The task that controls the CPU is the currently executing task. While in control of the CPU a task can execute system calls-- instructions which are executed by the system on behalf of the user and which can access system resources reserved for the operating system--or user instructions--instructions written by the user which access only system resources that are accessible to that user. A task that is waiting to use the CPU is an executable task. A task that relinquishes the CPU to wait for the completion of a system call is a suspended task. A task remains suspended until the system call is complete; it cannot regain control of the CPU while it is suspended.

The primary functions of the scheduler are to evaluate priorities and to regulate the amount of time a task can spend in the CPU before being displaced by a task of equal priority. Both of these functions require the scheduler to assess the "personality" of each task in the system.

9.6.2 A Task's Personality

In order to make the scheduling algorithm more flexible, the scheduler classifies each active task according to the kinds of demands it makes on the system's resources. The classification reflects the scheduler's best guess about the kind of system resources a task is likely to use. The operating system recognizes four classes (personalities) of task: tty (terminal) intensive, disk intensive, pipe intensive, and CPU intensive. A tty-intensive task is one whose most recently executed system call performed a character-oriented function such as writing to a terminal. A disk-intensive task is one whose most recently executed system call performed a disk-oriented function such as retrieving a block of data from a disk. Similarly, a pipe-intensive task is one that most recently performed a pipe-oriented function such as writing to a pipe. A CPU-intensive task, on the other hand, is one that during its last stay in the CPU either executed only user instructions or executed system calls that accessed more than one type of system resource (e.g., disks and terminals).

The scheduler reevaluates, but does not necessarily change, the classification of a task each time the task gains control of the CPU and each time it executes a system call. Each time the scheduler changes a task's classification, it also recalculates that task's priority. Because priority calculations consume valuable system resources, the scheduler reclassifies a task a maximum of once during any given stay in the CPU. The scheduler follows these rules:

1. When a task gains control of the CPU for the first time, the scheduler classifies it as CPU-intensive.

2. If a CPU-intensive task executes a device-oriented system call (one that accesses a terminal, disk, or pipe), the scheduler immediately reclassifies the task according to the type of device accessed.

3. If a non-CPU-intensive task executes a system call that accesses a different type of device from the one on which the classification is based (e.g., if a disk-intensive task writes to a pipe), the classification does not immediately change. Instead, the scheduler notes that the task performed more than one kind of device-oriented system call while it was in control of the CPU.

4. When a task regains control of the CPU, the scheduler does not change its classification unless the task executed more than one kind of device-oriented system call during its last stay in the CPU. Because the scheduler has no way of guessing what kind of resources such a task is likely to use, it gives the task its original classification--CPU intensive.

9.6.3 Determining Priorities

Once each second the scheduler evaluates the priorities of all executable tasks. In addition, it evaluates the priority of a task whenever it switches from executing a system call to executing a user instruction. Of the factors that go into the determination of a task's priority, you can influence two: the bias and the "CPU utilization".

You can set the bias with either the "setpr" system call (see Introduction to UniFLEX System Calls) or the "nice" command (see 68xxx UniFLEX Utility Commands).

"CPU utilization" is a dynamic parameter associated with each task. A task begins with its value of "CPU utilization" equal to 0. However, this value changes depending on how much time the task spends in the CPU. An increase in the value of "CPU utilization" lowers a task's priority. You cannot directly alter the "CPU utilization", but you can adjust several parameters that determine how quickly it changes. Each of the parameters discussed here is defined four times--once for each personality (see Section 9.6.2).

9.6.3.1 Max CPU utilization

The adjustable parameter "max CPU utilization" sets an upper limit on the value of "CPU utilization" for tasks of a given personality and, therefore, on how much use of the CPU can affect the priority of those tasks. The larger this number is, the more adversely use of the CPU can affect the priority.

9.6.3.2 CPU utilization increment per hit

Each time the system clock ticks (every 10 milliseconds), the operating system increments the "CPU utilization" of the task that is in the CPU unless it has already reached the maximum. The value of the adjustable parameter "CPU utilization increment per hit" defines the size of the increment for tasks of a given personality. It is a measure of the price paid in terms of priority for use of the CPU. The larger this number is, the more adversely each use of the CPU can affect the priority.

9.6.3.3 CPU utilization decay

Whenever the operating system evaluates the priority of an executable task, it decrements the value of "CPU utilization" for that task. The value of the adjustable parameter "CPU utilization decay" defines the size of the decrement for tasks of a given personality. It is a measure of the reward to a task's priority of not using the CPU. The larger this number is, the more favorably inactivity affects the priority.

9.6.4 Determining Duration of Stay in the CPU

The length of time (measured in system ticks) that a task can stay in the CPU before being displaced by a task of equal priority is determined by its value of "quantum". "Quantum" is a dynamic parameter associated with each task. A task begins with its value of "quantum" equal to 0. The value increases, up to a maximum, each time the task gains access to the CPU. Each time the system clock ticks (every 10 milliseconds), the operating system decrements the value of "quantum" for the task in the CPU by 1 tick. When the value of "quantum" falls to 0, the task must relinquish the CPU to an executable task of equal priority if one exists. Although you cannot directly alter the value of "quantum", you can adjust the parameters that determine how quickly it increases and what its maximum value can be. Each of these parameters is defined four times--once for each personality.

9.6.4.1 Max quantum

The adjustable parameter "max quantum" sets an upper limit on the value of "quantum" for tasks of a given personality and, therefore, on how long those tasks can remain in the CPU before being displaced by a task of equal priority.

9.6.4.2 Quantum increment

Each time a task gains access to the CPU, the operating system increments its value of "quantum" unless it has already reached the maximum. The value of the adjustable parameter "quantum increment" defines the size of the increment for tasks of a given personality.

9.7 Examples

The following examples illustrate some uses of the "tune" command.

1. /etc/tune /uniflex +r
2. /etc/tune /usr2/uniflex tasks=32 page_dev=/dev/fd0 +pP

The first example displays a list of the items that "tune" can adjust. The current value of each item in the file "uniflex" in the root directory appears in parentheses.

The second example changes the specified parameters in the file "uniflex" in the directory "/usr2". Presumably, a system disk is mounted on "/usr2". This command sets the maximum number of tasks allowed on the system to 32 and defines floppy drive 0 as the paging device. In order for this particular version of the operating system to be able to perform paging, a floppy disk formatted with paging space must be in floppy drive 0. This command also allows you to modify the parameters related to both "phys" segments and task personality.

9.8 Error Messages

This section describes the error messages returned by "tune".

"<dev_name> is not a block device.
The device specified is not a block device.

Illegal character in number: <num>
The value specified contains a character that is neither a digit nor, if appropriate, a valid hexadecimal character.

Invalid option: '<char>'
The option specified by <char> is not a valid option to the "tune" command.

Invalid parameter: "<param>"
The parameter specified is not an adjustable parameter.

Invalid pipe device.
The pipe device must be the same as either the root device or the paging device.

Not a configurable 68xxx UniFLEX(R) file.
The file specified must contain a copy of the UniFLEX Operating System.

Syntax: /etc/tune <file_name> [<param_list>] [+pP]
/etc/tune <file_name> [+pPr]
The "tune" command requires exactly one argument. This message indicates that the argument count is wrong.

The 'q' option is incompatible with interactive and read-only modes.
You may specify the 'q' option only when you are changing parameters from the command line.

The 'r' option is incompatible with command-line parameters.
The 'r' option, which tells "tune" to operate in read-only mode, conflicts with the specification of parameters on the command line. The command is aborted.

***Value must be a multiple of <num>.
The value for the number of buffers in the system must be a multiple of 8. The value for the time zone must be a multiple of 60.

***Value out of range [num_1, num_2]
The value specified for a parameter is not within the range of acceptable values. The limits of the range are shown inside the square brackets.

You must be system manager to change or adjust values.
Only the system manager can alter an adjustable parameter.

Chapter 10

Repairing a Damaged Disk

10.1 Introduction

The UniFLEX Operating System includes a utility, "diskrepair", which checks the structure of the disk or disks specified on the command line. The structure of a disk refers to the layout of and the connections among files, directories, free space, paging space, and other information that makes up the file system. "Diskrepair" detects any inconsistencies in the structure of the disk and, optionally, repairs them. Although "diskrepair" does not methodically search for and repair media (I/O) errors, it can take care of any bad blocks it discovers. If the 'a' option is in effect when "diskrepair" encounters an I/O error, it calls the utility "/etc/badblocks", which places the offending block in the bad-blocks file, "/.badblocks" (see 68xxx UniFLEX Utilities Commands).

While it is operating, "diskrepair" calls two other utilities--"blockcheck" and "fdncheck", which are both located in the directory "/etc". "Blockcheck" is concerned with the allocation of blocks on the disk. It locates problems such as duplicate blocks, missing blocks, and invalid block addresses. "Fdncheck" is concerned with the directories on the disk. It locates problems such as unreferenced files, directory entries with invalid associated files, and so forth. These errors are discussed in more detail later in this chapter.

"Diskrepair" performs some tasks that are not directly related to the logical structure of the disk. These preliminary tasks are essential to the proper performance of the utility. The heart of the program, which actually checks the structure of the disk, consists of the following six phases:

Phase 1--Check allocated blocks
Phase 2--Scan directories
Phase 3--Check unreferenced directories
Phase 4--Check file and directory links
Phase 5--Check free lists
Phase 6--Check SIR information

This chapter discusses each of the phases, as well as the other tasks performed by "diskrepair", in chronological order. The appropriate error messages, with the exception of the messages resulting from physical errors, are documented with each section. Messages resulting from physical errors are documented in Section 10.14. In addition, Section 10.15 consists of an index of error messages which directs you to the page on which each message is explained.

10.1.1 Structure of a UniFLEX Disk

Before you can understand what "diskrepair" does, you must understand some things about the structure of a UniFLEX disk. When the operating system formats a disk, it writes the system information record (SIR) to the second block on the disk, block 1. The SIR contains information describing the layout of the remainder of the disk. This information is essential to the successful execution of the operating system. The rest of the disk consists of contiguous-file space, paging space, volume space, and file descriptor nodes.

Paging space is a section of the disk that is reserved for storing portions of tasks that the operating system removes from memory to make room for tasks of higher priority. Paging space is not necessary on a disk that is to contain only data, but every system disk needs some paging space (see Section 3.4).

Normally when a disk is formatted, most blocks on it are available for storing data in regular files and directories. The addresses of these available blocks are maintained in a "free list". When a noncontiguous file needs a block, the operating system removes an address from the free list and associates the block at that address with the file in question. The combination of the blocks used in files and the blocks in the free list is known as the volume space.

The addresses of blocks available for contiguous files are maintained in a separate free list. When a contiguous file needs a block, the operating system removes an address from the contiguous-file free-list and associates the block at that address with the file in question. The combination of the blocks used in contiguous files and the blocks in the contiguous-file free-list is known as the contiguous-file space. Not all operating systems support contiguous files (see Preface).

When you format a disk (using one of the versions of the "format" command), the operating system reserves a certain number of blocks (determined by the 'f' option) for file descriptor nodes (fdns). An fdn

contains all the information that the operating system needs to know about a file. This information includes but is not limited to the type of file, the owner of the file, the size of the file, and the address of each block that is part of the file.

Whenever the operating system creates a file, it makes an entry in the parent directory. The entry contains the name of the file and the number of the fdn assigned to that file. It is possible for more than one directory entry to point to the same fdn; each of these entries is called a link. Each link results in another name for a file which already exists. However, no matter how many links there are to a file, only one fdn describes the file itself. Thus, each file on the disk should correspond to exactly one fdn.

The information in the SIR and the fdns establishes the logical structure of the disk. "Diskrepair" checks this structure and, optionally, repairs the errors it finds. It is able to do so because some of the information on the disk is redundant. For instance, the link count for a file (the number of directory entries that point to the fdn for that file) is stored in the fdn itself for quick reference. However, as "diskrepair" looks at the structure of the disk, it checks every directory entry and keeps track of how many times each fdn is referenced. If the number of direct references (or links) to an fdn does not agree with the the number stored in the fdn itself, "diskrepair" can easily change the number in the fdn.

10.1.2 Physical Errors on the Disk

"Diskrepair" is not a substitute for maintaining proper backups. For one thing, it cannot repair physical errors. A physical error is an error from the hardware, usually caused by physical damage to the medium, which prevents the operating system from reading from or writing to the medium. If at any time "diskrepair" encounters a physical error on the disk, it prints one of the following messages:

```
Error reading block <block_num>.
Error writing block <block_num>.
Error reading fdn <fdn_num> in block <block_num>.
Error writing fdn <fdn_num> in block <block_num>.
```

followed by the prompt:

```
-----> Continue?
```

If you see one of these messages, your disk is probably physically damaged. In general, if you choose to continue with "diskrepair", the results are entirely unpredictable. They depend on precisely which block is damaged. Continuing with "diskrepair" may cause further damage to the disk, but in some cases it may be the desired course of action. If you choose not to continue, "diskrepair" aborts.

We suggest that you respond negatively to the prompt to continue the first time "diskrepair" reports an I/O error and that you immediately rerun "diskrepair". It is possible--though unlikely--that the I/O error is a soft one and will not recur. If the error does recur, respond negatively to the prompt to continue and immediately rerun "diskrepair" with the 'a' option (see Section 10.2.1).

In many cases if you choose to continue, you receive another message which describes what "diskrepair" was trying to do when it encountered the I/O error (see Section 10.14).

10.1.3 Limitations of "diskrepair"

"Diskrepair" cannot solve all the problems your disk may have. As mentioned in the preceding section, it cannot fix physical problems on your media (but see Section 10.2.1). As for problems with the logical structure of the disk, "diskrepair" can only repair an error if the damaged information is redundant--that is, if there is some way of determining what the information should be. It cannot, for example, repair a badly damaged SIR; nor can it repair a disk if the root directory is severely damaged.

Now that you have been warned that "diskrepair" cannot fix all the problems that may arise when a disk is damaged, let's look at how it functions and at the large number of things that it can do.

10.2 The Command Line

The syntax for "diskrepair" is as follows:

```
/etc/diskrepair [<dev_name_list>] [+<abBfmMnpqruv>]
```

where <dev_name_list> is a list of the names of the devices to check.

Brief descriptions of the options which are available follow:

a	Automatically place in the file "/.badblocks" any bad blocks encountered. Continue to run "diskrepair" until the disk is repaired or the program has executed ten times.
b	Perform "blockcheck" only.
B	Ignore sector zero.
f	Perform "fdncheck" only.
m	Ignore missing blocks is both free lists.
M	Check only to see whether or not the "mount flag" on the disk is set. If it is, clear it.
n	Do not attempt to fix errors.
p	Prompt for permission to repair.
q	Use quiet mode.
r	Rebuild both free lists whether or not they are in error.
u	Report on the usage of disk blocks.
v	Use verbose mode.

If you execute "diskrepair" without any options, it does what it can to repair structural errors on your disk. You can, however, modify its behavior by specifying various options on the command line. In particular, if you invoke the ´p´ and ´v´ options, "diskrepair" reports its progress in greater detail and prompts you for permission before making any repairs. In order to give you a more detailed explanation of the "diskrepair" command, this chapter assumes that you have specified both the ´p´ and ´v´ options on the command line. In the absence of the ´p´ option, "diskrepair" behaves as if it had prompted you for a response before each repair and you had answered positively.

Detailed descriptions of the options follow.

10.2.1 The ´a´ Option

The ´a´ option tells "diskrepair" to call the "badblocks" command whenever it encounters a bad block. "Badblocks" then sequesters the block in the file "/.badblocks". Each time it calls "badblocks", "diskrepair" sends the following message to standard error:

```
Calling "/etc/badblocks" to remove bad block.
```

The ´a´ option also tells "diskrepair" to run continuously until either the disk is fixed or the program has executed ten times. Each time "diskrepair" starts over, it sends the following message to standard error:

Rerunning "diskrepair" on "<dev_name>".

10.2.2 The 'b' Option

The 'b' option instructs "diskrepair" to run only the "blockcheck" portion of the utility. This procedure is often considerably faster, but still provides a fairly complete assessment of the validity of the structure of the disk.

10.2.3 The 'B' Option

Normally "diskrepair" tries to read sector zero to make sure that the disk being checked was not created by the "backup" command. You can, if necessary, bypass this check by specifying the 'B' option, which instructs "diskrepair" to ignore sector zero. You should only do so if "diskrepair" aborts because it cannot read sector zero.

10.2.4 The 'f' Option

The 'f' option instructs "diskrepair" to run only the "fdncheck" portion of the utility. This option is useful if you suspect a problem exists in the directory structure, but the result is by no means a thorough check of the structure of the disk.

10.2.5 The 'm' Option

The operating system maintains a list of blocks available for use by noncontiguous files called the free list and a list of blocks available for use by contiguous files called the contiguous-file free-list. A missing block is any block in the volume space or the contiguous-file space which is not a part of any file and is not in one of the free lists. The existence of such blocks is a harmless error in the structure of the disk. "Diskrepair" generally places missing blocks in the appropriate free list. The 'm' option, however, instructs "diskrepair" not to rebuild either free list solely on account of missing blocks. This option reduces the time required for "diskrepair" to run if missing blocks are the only problem in one or both of the free lists.

10.2.6 The ´M´ Option

The ´M´ option instructs "diskrepair" to bypass all but one of its usual functions. If you specify the ´M´ option, "diskrepair" simply checks to see whether or not the "mount flag" on the disk it is examining is set. If the flag is not set, "diskrepair" terminates. If it is set, "diskrepair" clears it, then terminates. This option may be useful if your system crashes (see Chapter 11).

10.2.7 The ´n´ Option

The ´n´ option tells "diskrepair" to report all errors but to make no attempt to fix them. Therefore, "diskrepair" opens the device for reading only. This option is useful for checking the structure of a disk without risking the loss of data during repairs.

10.2.8 The ´p´ Option

If you specify the ´p´ option, "diskrepair" reports each error, followed by a prompt requesting permission for the proposed repair. All prompts require an answer of either ´y´ ("yes") or ´n´ ("no").

Many repairs result in the loss of data. (You can generally infer what has been lost from the messages "diskrepair" displays.) Judicious use of the ´n´ and ´p´ options not only allows you to assess the damage to the disk and to decide which information you are willing to sacrifice during the repair process but also gives you the opportunity to try to salvage the data (if salvage is possible on your system) before repairing the disk. Methods of salvaging data from a damaged disk are discussed in Chapter 11.

10.2.9 The ´q´ Option

The ´q´ option suppresses certain warnings and messages from "diskrepair". Several conditions exist which, while not technically errors in the structure of the disk, may cause problems. These conditions usually result in a warning message; the ´q´ option suppresses such messages.

10.2.10 The 'r' Option

By default, if "diskrepair" finds that either free list is in error, it rebuilds it. The 'r' option instructs "diskrepair" to rebuild both free lists whether or not they contain errors. This option may save some time if you know that one of the free lists is bad. You can also use it to reduce fragmentation within the free lists.

10.2.11 The 'u' Option

The 'u' option generates a report on the block usage of the specified device. This report is printed at the end of the "diskrepair" operation. It contains statistics on (1) the number of each type of file in the file system and the total number of files in the system, (2) the number of unused blocks and the number of used blocks in the volume space, including a breakdown of how the used blocks are allocated, (3) the number of unused blocks and the number of used blocks in the contiguous-file space, and (4) the number of free fdns and the number of fdns in use.

10.2.12 The 'v' Option

"Diskrepair" operates in one of two modes: simple or verbose. Simple mode is selected by default; verbose mode is selected by the 'v' option. In simple mode "diskrepair" reports only those errors which require the deletion of either directory entries or files. In verbose mode "diskrepair" reports all errors. In addition, it sends to standard output informative messages telling you which phase it is in.

In verbose mode the 'p' option causes "diskrepair" to prompt for permission to make any changes to the disk. In simple mode "diskrepair" prompts you only for permission to make changes which require the deletion of either directory entries or files; it automatically repairs all other errors without prompting.

10.3 Preliminary Checks

Before "diskrepair" even begins to check the structure of the disk, it makes several preliminary checks, which are necessary to ensure that the utility can function properly.

10.3.1 Command-line Options

First of all, "diskrepair" checks the validity of each character specified as an option on the command line. If any character is not a valid option, "diskrepair" tells you

```
Invalid option: '<char>'.
```

If the options you specify conflict with each other, it tells you

```
Conflicting options.
```

In either case, "diskrepair" aborts.

10.3.2 Specified Device

Next, "diskrepair" looks at the device or devices you specified on the command line. If you specify a nonexistent device or if you fail to specify a device, "diskrepair" responds with the appropriate message:

```
No such device.
"<dev_name>" ignored.
```

or

```
No device specified.
```

The first of these messages is not fatal unless you specified only one device on the command line. The second is always fatal.

"Diskrepair" can only operate on a block device. Therefore, it must determine whether or not the device specified on the command line is a block device. If it is not, "diskrepair" issues the following message:

```
Not a block device.
```

This message, too, is fatal to "diskrepair".

10.3.3 Backup Devices

If you have correctly specified a device, "diskrepair" looks at the first sector of the disk to make sure that the disk was not created by the "backup" command. If that sector is damaged, "diskrepair" informs you

```
Cannot read sector zero.
```

It then aborts. If you are certain that the disk is was not created by the "backup" command, it is safe to invoke "diskrepair" with the ´B´ option, which tells it to ignore sector zero (see Section 10.2.3).

If "diskrepair" can read the first sector of the disk and discovers that it is a disk created by "backup", it sends the following message to standard error:

```
Cannot check a "backup" disk.
```

The program then aborts.

10.3.4 Permissions

If you execute "diskrepair" without the ´n´ option, you must have both read and write permission on the specified device; with the ´n´ option, you need only read permission. If you do not have the necessary permissions, "diskrepair" informs you:

```
Permission denied.
```

The program then aborts.

10.3.5 Unmounting a Mounted Disk

With the exception of the root device (see Section 9.4.16) "diskrepair" cannot alter a disk if it is in use. Therefore, when checking any other device, "diskrepair" determines whether or not the specified disk is mounted, and, unless you specify the ´n´ option, it unmounts a mounted disk before proceeding. If any user´s working directory is on the device or if any file on the device is being accessed when "diskrepair"

tries to unmount it, the unmount procedure fails and the following message appears on your screen:

Device is busy.

Although "diskrepair" can unmount a mounted disk, it cannot prevent a user from mounting the disk while "diskrepair" is in progress. You yourself must assume that responsibility.

If "diskrepair" encounters some other problem when it tries to unmount the device, it responds

Unmount error <error_num>.

where <error_num> is the number of the UniFLEX error that caused the failure. Consult the operating system manual for an explanation of the error.

If the 'n' option is in effect, "diskrepair" does not need to unmount a mounted disk because it cannot write to the disk. However, when running "diskrepair" with the 'n' option, you should make sure that no one else is using the disk that you are testing. The results of running "diskrepair" while someone is using the disk are unreliable.

10.3.6 Checking the Root Device

You can use "diskrepair" to check the structure of the root device, but in order to perform correctly, the utility must suspend all other tasks running on the system. After invoking the system call to do so, "diskrepair" waits for 5 seconds to give all tasks a chance to handle the suspension gracefully. During those 5 seconds any input to "diskrepair" (such as a control-C) is lost. We recommend that you refrain from typing (except to interrupt or respond to "diskrepair") until "diskrepair" releases all tasks from suspension. Although the operating system can continue to accept input, the suspended tasks cannot process it, and if you type more than 255 characters, you will lose some data.

If for some reason "diskrepair" cannot suspend all tasks, it returns the following message before either aborting or proceeding to the next device:

Cannot suspend running tasks.
"<dev_name>" ignored.

When "diskrepair" is done, it first updates the disk if necessary, then allows all suspended tasks to resume. Of course, if "diskrepair" shuts down the system, the suspended tasks cannot resume; they are lost.

10.3.7 Status of the Root Directory

As its final preliminary check, "diskrepair" tries to read the fdn which describes the root directory. If, for any reason, it cannot access this fdn, it reports

Cannot access fdn for root device.

This error is fatal to "diskrepair". It is unlikely that you will be able to salvage data from the disk (see Chapter 11), and you must eventually rebuild the system.

10.4 Calling "blockcheck"

At this point, "diskrepair" calls the utility "/etc/blockcheck". If it cannot read or execute that file, it tells you

Cannot call "/etc/blockcheck".

It then proceeds to the next device, if you specified one. Otherwise, it aborts.

After successfully accessing "blockcheck", "diskrepair" checks to make sure that it is the proper version of the utility. If it is not, it aborts after reporting:

"/etc/blockcheck" is invalid.

Next, "blockcheck" tries to open the specified device. If it fails, it reports

Cannot open device.

If you specified only one device on the command line, "diskrepair" aborts. Otherwise, it prints the message

"<dev_name>" ignored.

It then proceeds to the next device.

10.4.1 Abnormal Termination of "blockcheck"

If for any reason "blockcheck" terminates abnormally--that is, receives a program interrupt from the operating system--"diskrepair" issues the following message before it aborts:

"Blockcheck" terminated abnormally (status = <num>).
"Diskrepair" aborted for "<dev_name>".

Such a message is not indicative of a problem with either "diskrepair" or the device. You should try to run "diskrepair" again, for the problem may not recur. If the problem persists, it is probably caused by malfunctioning hardware. Contact Technical Systems Consultants for assistance.

10.4.2 Improper I/O Redirection

When testing the structure of a disk, it is impractical to try to redirect either standard output or standard error (the results of the test) to a file on the disk you are testing. If you do try to do so, you receive the following message:

Output directed to device under test.

In such a case, "diskrepair" aborts.

While it is checking the validity of any I/O redirection, "diskrepair" must access the fdns of whatever files are open as standard error and standard output. If for any reason "diskrepair" cannot read one or both of these fdns, it prints whichever of the following messages is appropriate:

```
Cannot access fdn for standard error.
Cannot access fdn for standard output.
```

In such a case you should reinvoke "diskrepair" with the terminal as the standard I/O channel that caused the problem.

10.5 Preliminary Checks on the SIR

10.5.1 Accessing the SIR

Before proceeding with the tests on the structural integrity of the disk, "blockcheck" tries to read the SIR. If the SIR has been damaged so badly that "blockcheck" cannot read it, it reports:

```
Cannot read System Information Record.
```

This error is fatal to "diskrepair". You may be able to salvage some information from the disk (see Chapter 11), but you must reformat it.

10.5.2 Size of Disk

"Diskrepair" checks to see that the size of the disk is within the range that it can handle. The current limit is approximately 400 Megabytes. If the data in the SIR indicate that the disk is larger than this limit, "diskrepair" issues the following message:

```
Disk too large or bad size in SIR.
"<dev_name>" ignored.
```

This error, too, is fatal to "diskrepair". You may be able to salvage some information from the disk (Chapter 11), but you must reformat it.

10.5.3 Fdn Count

"Diskrepair" reads the SIR to determine how many blocks on the disk are reserved for use as fdns. It checks to see that (1) the number of fdns does not exceed 65,528 (the maximum allowed by the operating system) and (2) the number of blocks allocated for fdns does not exceed the size of the disk. If either limit is exceeded, "diskrepair" issues the following message:

```
Too many fdn blocks: <num>.
"<dev_name>" ignored.
```

This error is fatal to "diskrepair". You may be able to salvage some information from the disk (see Chapter 11), but you must reformat it.

10.5.4 First Block of Paging Space

The SIR contains the address of the first block of the paging space. It also contains information that allows "diskrepair" to calculate independently what that address should be. If the calculated address is less than the address stored in the SIR, a contradiction exists: some blocks that are in the volume space are also in the paging space. If this situation arises, "diskrepair" issues the following message:

```
Volume space overlaps paging space.
-----> Assume volume space correct and fix?
```

If you respond 'n', "diskrepair" aborts with the message

```
"<dev_name>" ignored.
```

If you respond 'y', "diskrepair" assumes that the address it calculated is correct and rewrites the SIR with the calculated address as the address of the first block of the paging space.

If the calculated address is greater than the address stored in the SIR, the net result is a hole in the map of the disk: the volume space ends before the paging space begins. The intermediate blocks are inaccessible. Although this situation wastes some disk space, it does not cause any logical inconsistencies in the structure of the disk. Therefore, "diskrepair" considers the disk intact, but it issues the

following warning:

WARNING: <num> unassigned blocks between volume space and paging space.

10.5.5 First Block of Contiguous-File Space

The SIR contains the address of the first block of the contiguous-file space. It also contains information that allows "diskrepair" to calculate independently what that address should be. If the calculated address is less than the address stored in the SIR, a contradiction exists: some blocks that are in the paging space (or the volume space if the disk does not contain any paging space) are also in the contiguous-file space. If this situation arises, "diskrepair" issues one of the following messages:

Volume space overlaps contiguous-file space.
Paging space overlaps contiguous-file space.

The message is followed by whichever of the following prompts is appropriate:

-----> Assume volume space correct and fix?
-----> Assume paging space correct and fix?

If you respond 'n', "diskrepair" aborts with the message

"<dev_name>" ignored.

If you respond 'y', "diskrepair" assumes that the address it calculated is correct and rewrites the SIR with the calculated address as the address of the first block of the contiguous-file space.

If the calculated address is greater than the address stored in the SIR, the net result is a hole in the map of the disk: the volume or paging space ends before the contiguous-file space begins. The intermediate blocks are inaccessible. Although this situation wastes some disk space, it does not cause any logical inconsistencies in the structure of the disk. Therefore, "diskrepair" considers the disk intact, but it issues the one of the following warnings:

```
WARNING: <num> unassigned blocks between volume space and
         contiguous-file space.
WARNING: <num> unassigned blocks between paging space and
         contiguous-file space.
```

10.6 The File "/.badblocks"

You can effectively hide blocks that are known to be bad by placing them in the bad-blocks file, "/.badblocks", with either the "format" or "badblocks" command (see Section 3.5). The operating system knows not to allocate any of the blocks which are in this file. "Diskrepair", too, must be aware of the presence of this file, so that it does not inadvertently place the bad blocks in either free list. Therefore, the last thing it tries to do before it starts to check the structure of the disk is to read the bad-blocks file.

10.6.1 Accessing the Bad-Blocks File

If "diskrepair" encounters an I/O error while trying to read the bad-blocks file, it reports the following error and continues:

```
Error checking ".badblocks" file.  File ignored.
```

Although "diskrepair" continues to run, it does not know about the blocks in the bad-blocks file. In such a situation the results are unpredictable. If problems arise, you may be able to salvage some of your data, (see Chapter 11), but you must eventually reformat the disk.

10.6.2 Validating the Bad-Blocks File

If "diskrepair" can read the bad-blocks file but finds that (1) it is not a regular file or (2) the first block of the file is located where the boot sector (sector zero), the SIR, or the root directory should be, it issues the following message and tries to continue:

```
Bad ".badblocks" file.  File ignored.
```

Since "diskrepair" cannot read the bad-blocks file, it does not know which blocks are bad. It might, therefore, try to access a bad block for its own use. If it does so and if it detects the error, it notifies you of the physical error. In such a case you should deny permission for "diskrepair" to continue, salvage whatever data you can (see Chapter 11), and rebuild your system.

10.6.3 Checking the Size

The bad-blocks file cannot contain more than 16,522 blocks. It is unlikely that you will ever use a disk that contains that many bad blocks, but it is possible for the fdn associated with the bad-blocks file to be damaged in such a way that the operating system thinks that the bad-blocks file is too large. In such a case "diskrepair" returns the following message:

```
Bad-blocks file too large.
```

It then ignores the part of the file that appears to be beyond 16,522 blocks.

10.7 Phase 1--Check Allocated Blocks

During phase 1, "diskrepair" reads every fdn on the disk. From reading the fdns it can determine which ones are inactive (do not describe a file), which ones describe devices, and which ones describe regular files, directories, or contiguous files. If an fdn refers to a regular file, a directory, or a contiguous file, "diskrepair" continues reading the fdn to see how many blocks and which blocks are allocated to that file.

10.7.1 File Size

During phase 1, "diskrepair" evaluates the size of each file as recorded in its fdn. If the size is logically inconsistent, "diskrepair" reports an error. It does not attempt to fix the error until phase 2.

10.7.1.1 Noncontiguous files

Once "diskrepair" knows how many blocks are allocated to a particular noncontiguous file, it compares that number to the size that is written in the fdn. Because the size is measured in bytes, this comparison can only determine if the size and the number of allocated blocks are compatible. For instance, if two blocks are allocated to the fdn, you would normally expect the file to contain between 513 and 1,024 bytes. If the size is not in this range, "diskrepair" reports

File size error in fdn <fdn_num> (size <num_1>, found <num_2>).

10.7.1.2 Contiguous files

The only restriction that "diskrepair" places on the size of a contiguous file is that it must be a multiple of 512. If it is not, "diskrepair" reports:

Contiguous-file size error in fdn <fdn_num>.

10.7.2 Out-of-Range Blocks in Fdns

While "diskrepair" is reading fdns to determine which blocks are allocated to which files, it checks to make sure that the address of each block is a valid one. For noncontiguous files the addresses should correspond to blocks within the volume space on the disk; for contiguous files, to blocks within the contiguous-file space. If an address which appears in an fdn corresponds to a block on an inappropriate part of the disk, "diskrepair" reports

Out-of-range block in fdn <fdn_num>.

It does not, at this point, do anything to correct the problem (see Section 10.9.5). The problem will be corrected later; however, the only way to fix the structure of the disk in such a case is to delete the file containing the out-of-range block. Before doing so, however, you may be able to salvage most of the file (see Chapter 11).

"Diskrepair" maintains a list containing the addresses of the out-of-range blocks it encounters. Currently, the maximum length of this list is forty blocks. If "diskrepair" encounters more than forty out-of-range blocks, it continues to report each one, but it tells you

Too many out-of-range blocks.

This message means that "diskrepair" cannot add any more blocks to the list. Of course, under such conditions it cannot completely repair the disk. Therefore, when "diskrepair" finishes, it tells you to run "diskrepair" again.

10.7.3 Blocks Duplicated in Fdns

A given block from the volume space should be allocated either to the free list or to exactly one fdn (hence, one file). If "diskrepair" finds that a block is allocated to more than one fdn, it keeps track of the number of the fdn in which the second, and any subsequent, allocations occur. If it does find any blocks that are duplicated in fdns, it enters phase 1B, in which it scans the fdns again in order to determine which fdn first claimed a duplicate block. It then prints a message for each fdn which contains a duplicate block. The message has the following form:

Duplicate block <block_num> in fdn <fdn_num>.

At this point "diskrepair" does not do anything to correct the problem (see Section 10.9.6). The problem will be corrected later; however, the only way to fix the structure of the disk is to delete all but one of the files that contain the duplicate block. Before doing so, however, you may be able to salvage most of the data in the files (see Chapter 11).

"Diskrepair" maintains a list containing the addresses of the duplicate blocks it encounters. Currently, the maximum length of this list is sixty blocks. If "diskrepair" encounters more than sixty duplicate blocks, it continues to report each one but also includes a message saying

Too many duplicate blocks.

This message means that "diskrepair" cannot add any more blocks to the list. Of course, under such conditions it cannot completely repair the disk, and when it finishes, it tells you to run "diskrepair" again.

10.8 Transition between Phase 1 and Phase 2

During the transition from phase 1 to phase 2 "diskrepair" calls the utility "fdncheck", which does some preliminary checking of its own before proceeding with the evaluation of the logical structure of the disk.

10.8.1 Calling "fdncheck"

If "diskrepair" cannot read or execute "fdncheck", it tells you

```
Cannot call "/etc/fdncheck".
```

It then proceeds to the next device, if you specified one. Otherwise, it aborts.

After successfully accessing "fdncheck", "diskrepair" checks to make sure that it is the proper version of the utility. If it is not, it aborts after reporting

```
"/etc/fdncheck" is invalid.
```

10.8.2 Abnormal Termination of "fdncheck"

If for any reason "fdncheck" terminates abnormally--that is, receives a program interrupt from the operating system--"diskrepair" issues the following message:

```
"Fdncheck" terminated abnormally (status = <num>).
"Diskrepair" aborted for "<dev_name>".
```

It then proceeds to the next device, if you specified one. Otherwise, it aborts.

Such a message is not indicative of a problem with either "diskrepair" or the device. You should try to run "diskrepair" again, for the problem may not recur. If the problem persists, it is probably caused by malfunctioning hardware. Contact Technical Systems Consultants for assistance.

10.8.3 The File "/.badblocks"

The utility "fdncheck", like "blockcheck", tries to read the file ".badblocks" in the root directory. It is possible for "fdncheck" to have trouble reading the bad-blocks file even if "blockcheck" does not. If it does encounter a problem, it returns one of the error messages described in Section 10.6.1.

10.8.4 Reading the Root Directory

Before entering the next phase, "fdncheck" tries to read the root directory. If the root is so badly damaged that it cannot be read, "diskrepair" issues the following message:

```
Cannot read root directory.
```

If the length of the root directory has been truncated to 0, you receive a message to that effect:

```
Length of root directory is zero.
```

These messages are both fatal to "diskrepair". In either case the disk cannot be repaired. Such a disk cannot be mounted; therefore, you cannot even attempt to salvage the information on it.

If "fdncheck" tries to read the root directory and discovers that the root fdn describes it as a regular file rather than a directory, it reports

```
Root fdn is not a directory.
-----> Force into directory?
```

A negative response causes "diskrepair" to abort. Depending on the extent of the damage, you may be able to salvage some of the information on the disk (see Chapter 11). A positive response causes "diskrepair" to force the fdn to describe a directory and to prompt for permission to continue:

```
-----> Continue?
```

If you respond negatively, "diskrepair" aborts. If you respond positively, it tries to continue although the amount of success it has depends on the exact nature of the damage. If the only part of the fdn which is damaged is the part that describes the type of file associated with the fdn, this repair should solve the problem. If, however, much more of the fdn is damaged, other problems will arise. In a case where the root directory is badly damaged, "diskrepair" cannot fix the disk. Nor can you salvage any information from it. Your only choice is to reformat the disk.

10.9 Phase 2--Scan Directories

During phase 2, "diskrepair" scans each directory for problems in structure. A directory consists of a series of entries, each containing the name of a file and the number of the fdn that describes that file. The number of bytes in an entry is always a multiple of 16. Each entry uses at least 16 bytes--two for the fdn number and fourteen or more for the file name--and at most 64 bytes (see Section III of The UniFLEX Operating System).

10.9.1 Size of Directory

Currently "diskrepair" cannot handle a directory of more that 4,416 entries. However, this theoretical limitation on the size of a directory is unlikely to be a practical limitation. If you do have a directory that appears to be too large, "diskrepair" prints the following message:

```
Directory too large:  "<dir_name>".
-----> Truncate and continue?
```

If you receive this message and you know that you do not have a directory with more than 4,416 entries, chances are that the part of the fdn for the directory which contains the size is damaged. In such a case, it is safe to truncate because "diskrepair" simply sets the size to 4,416. If, on the other hand, you do have a directory that really is too large, you cannot successfully run "diskrepair" without losing the files beyond the 4,416th entry in the directory.

If the size of a directory is not a multiple of 16, "diskrepair" reports

```
Odd size for directory "<dir_name>".
   fdn=<fdn_num>     type=directory          size=<bytes>
   owner=<owner_name>  time=<time_and_date>
-----> Truncate?
```

"Diskrepair" corrects this error by truncating the size stored in the fdn to the nearest multiple of 16. This repair should cause no harm.

10.9.2 Nesting Directories

"Diskrepair" can only function to the level of twelve subdirectories. If your directory structure is more deeply nested than that, "diskrepair" informs you

```
Nesting too deep at "<dir_name>".
```

The program then aborts.

10.9.3 Invalid File Name

While reading a directory, "diskrepair" checks the validity of the file name that appears in each entry. If the file name is invalid, it reports:

```
Invalid file name: "<file_name>".
-----> Fix?
```

When it displays an invalid file name, "diskrepair" uses the caret, '^', followed by the appropriate character, to indicate any control character except the null character. It uses the symbol "^@" (a caret followed by an "at" symbol) to indicate the null character. It displays a character with the high-order bit set as the same character without the high-order bit set. Thus, an invalid name may appear to be valid if the only problem is that it contains one or more characters with the high-order bit set.

File names are discussed in detail in Section III of The UniFLEX Operating System. Basically, a file name is invalid if the directory entry which contains it (1) contains more than 55 characters, (2) contains a slash character, '/', within the name, (3) does not end with a null character (unless the name of the file is exactly 14 characters long) (4) contains a control character, (5) contains only null characters, (6) or contains a character with the high-order bit set (unless mandated by the size of the file name). A file name is also invalid if the way in which it is stored indicates that it should contain more than 14 characters but it does not.

Although an invalid file name is extremely unlikely, the consequences are serious. The operating system simply cannot access a file with an invalid name. Unless you fix the error, the file remains on the disk, inaccessible, until you reformat the disk. We therefore recommend that you allow "diskrepair" to rename the file. Such a change has no effect on anything else on the disk.

"Diskrepair" uses the following rules when fixing an invalid file name:

1. If the name contains more than 55 characters, "diskrepair" truncates it to 55 characters.
2. If the way in which the name is stored indicates that it should contain more than 14 characters but it does not, "diskrepair" shortens the name.
3. If the name contains an invalid character, "diskrepair" replaces it with an 'X'.
4. If the name contains only null characters, "diskrepair" names the file "file<fdn>".

It is possible for "diskrepair" to create a file name that matches the name of another file. If a file name should happen to be duplicated, you can change the name of one of the files with the "rename" command (see 68xxx UniFLEX Utility Commands) as soon as "diskrepair" terminates.

10.9.4 File Size

During phase 1, "diskrepair" reports all errors in file size (see Section 10.7.1) by telling you which fdn the error is in. It then stores the fdn number in a list, which it refers to during phase 2 when it is able to determine the name of the file associated with each fdn. The list has room for twenty fdns. During phase 2, "diskrepair" repeats its messages about errors in file size (for up to twenty files), but now the message includes the name of the file and a prompt for permission to

fix it:

```
File size error in fdn <fdn_num>, file "<file_name>".
-----> Fix?
```

When "diskrepair" corrects a file size error in a noncontiguous file, it counts the number of blocks allocated to the file, multiplies that number by 512 (the number of bytes in a block), and puts that number in the appropriate place in the fdn. Therefore, unless the file completely uses the last block allocated to it, the number "diskrepair" writes to the fdn is greater than the number of bytes in the file. Thus, the file is likely to contain some extraneous data. If the file is a text file, you can remove the extra data with an editor. When you exit from the editor, the operating system writes the correct size to the fdn. If the file is a binary file, however, you may not be able to remove the extraneous data and, if the file is not backed up, you may have to sacrifice it.

When "diskrepair" corrects a file size error in a contiguous file, it rounds the size up to the next multiple of 512.

If "diskrepair" encountered file-size errors in more than twenty files on your disk, it is unable to tell you the names of all of the files. In such a case you should fix the errors and rerun "diskrepair" so that you can fix the remaining errors.

10.9.5 Out-of-Range Blocks in Files

During phase 1, "diskrepair" makes a record of all the fdns that contain block addresses that are out of range. As it reads the directories during phase 2, it watches for each of these fdns. When it encounters one, it prints the following message:

```
Out-of-range block in "<file_name>".
   fdn=<fdn_num>     type=<file_type>     size=<bytes>
   owner=<owner_name>  time=<time_and_date>
-----> Delete?
```

If you do not delete a file containing an out-of-range block, the structure of the disk remains damaged. However, you may want to defer deleting the file, trying first to salvage what you can (see Chapter 11). If neither the 'n' nor the 'p' option is in effect, "diskrepair" automatically deletes all files that contain one or more out-of-range

blocks unless the file is the root directory or the bad-blocks file.

If an out-of-range block occurs in either the bad-blocks file or in the root directory, "diskrepair" cannot delete the offending file. Rather, the program prints one of the following messages:

```
Cannot delete ".badblocks".
Cannot delete root directory.
```

To correct either of these situations you must eventually reformat the disk. Of course, if the disk is not backed up, you should try to salvage as much information as possible (see Chapter 11) before reformatting it.

Now consider the case where more than one directory entry points to an fdn which contains an out-of-range block (multiple links to a file). The first time "diskrepair" encounters the out-of-range block in an fdn, it asks if you want to delete the corresponding file. If you do delete the file, "diskrepair" changes the fdn number in the directory entry to 0 and changes the fdn itself so that it is inactive. Thus, the fdn numbers in the directory entries for the other files which were linked to the file you deleted now point to an inactive fdn. "Diskrepair" handles this problem as it encounters each file (see Section 10.9.9).

Suppose, however, that you deny permission when "diskrepair" asks to delete the first file corresponding to the fdn that contains an out-of-range block. In that case, "diskrepair" finds the same out-of-range block when it encounters the next file linked to that fdn. Once again, it asks for permission to delete the file. If you deny permission, the pattern repeats itself. If, on the other hand, you grant permission to delete one of the linked files after having denied it to one or more of them, you create a new problem. "Diskrepair" changes the fdn number in the directory entry to 0 and makes the fdn itself inactive. If it encounters any more files which had been linked to the deleted file, it tells you, as it should, that their directory entries point to an inactive fdn. It can fix this problem. However, the directory entries for the files you did not delete that also contain the out-of-range block now also point to the same inactive fdn. It is too late for "diskrepair" to fix this problem because it has already checked those files. It can, nevertheless, alert you to the situation and does so by printing the following message:

```
WARNING: Previous links to fdn <fdn_num>.
```

In addition, when "diskrepair" finishes, you receive a message telling

you to run "diskrepair" again.

10.9.6 Blocks Duplicated in Files

During phase 1, "diskrepair" makes a record of all the fdns that contain duplicate block addresses. As it reads the directories during phase 2, it watches for each of these fdns. When it encounters one, it prints the following message:

```
Duplicate block in "<file_name>".
   fdn=<fdn_num>     type=<file_type>     size=<bytes>
   owner=<owner_name>  time=<time_and_date>
    Duplicated in fdn <fdn_num>.
-----> Delete?
```

Until you delete all but one file containing a duplicate block, the structure of the disk remains damaged. If neither the 'n' option nor the 'p' option is in effect, "diskrepair" inactivates every fdn except the last one which claims a duplicated block, unless the corresponding file is either the root directory or the bad-blocks file. The overall effect is to delete the files described by these fdns and to return to the free list all blocks they contain which are not duplicated in other files. If, however, you first execute the program with the 'n' option, "diskrepair" informs you which files contain duplicate blocks but deletes no files. If you run it with the 'p' option, you can respond 'n' to all prompts for deleting files. Once you know which files contain duplicated blocks, you can execute "diskrepair" again with the 'p' option, knowing which files you want to delete. In this way you can save the file you want to save even if it is not the last one to claim the block. In addition, you may be able to salvage some material before deleting it (see Chapter 11).

If a duplicate block occurs in either the file ".badblocks" or in the root directory, "diskrepair" cannot delete the offending file. Rather, the program prints one of the following messages:

```
Cannot delete ".badblocks".
Cannot delete root directory.
```

Because you can keep one of the files that contains a duplicate block, this situation may resolve itself cleanly--if the duplicate block really belongs in the file in question. If it does not, you will encounter other problems as well.

10.9.7 The Files "." and ".."

The first two entries in every UniFLEX directory should be the files "." and "..". The name ".." refers to the parent directory of the directory in question; the name "." refers to the directory itself. If "diskrepair" finds a directory that does not contain exactly one of each of these files, it issues one of the following messages:

```
WARNING: Too many "." entries for "<dir_name>".
WARNING: Too many ".." entries for "<dir_name>".
WARNING: No "." entry for "<dir_name>".
WARNING: No ".." entry for "<dir_name>".
```

These messages are only warnings. "Diskrepair" makes no effort to correct any of these problems. You can usually fix the case of a missing "." or ".." directory after "diskrepair" is finished (see Section 10.5.1). However, the case of multiple "." or ".." files cannot be repaired without deleting the directory. After deleting it, you should rerun "diskrepair", which will locate an unreferenced directory corresponding to the extra "." or ".." file. Do not place this directory in the lost-and-found directory. Instead, delete it.

If the entries "." and ".." are present in the directory, "diskrepair" checks their fdn numbers. They should match the fdns of the directory itself and its parent directory. If they do not, "diskrepair" prints whichever of the following messages is appropriate:

```
Bad "." in "<dir_name>".
Bad ".." in "<dir_name>".
```

followed by the prompt

```
-----> Fix entry?
```

It is perfectly safe to fix the entry; the change has no other effect on the disk.

10.9.8 Unknown File Type

The fdn contains information describing the type of file it refers to. Nine valid types of file exist: regular file, directory, block device, character device, network node, pseudoterminal, inactive, pipe, contiguous file. If "diskrepair" does not recognize the file as one of

these nine types, it notifies you:

```
Unknown file type for "<file_name>".
   fdn=<fdn_num>     type=unknown          size=<bytes>
   owner=<owner_name>  time=<time_and_date>
-----> Delete?
```

If you see this message, you may as well delete the file because you cannot salvage it.

10.9.9 Inactive Fdn

An inactive fdn is one that does not refer to any file; it is therefore a "free" fdn. If a directory entry points to an inactive fdn, "diskrepair" reports

```
Inactive fdn for "<file_name>".
   fdn=<fdn_num>     type=inactive     size=<bytes>
   owner=<owner_name>  time=<time_and_date>
-----> Delete directory entry?
```

This message indicates that (1) the fdn describing the file has been severely damaged, (2) the directory entry points to the wrong fdn--which happens to be inactive, or (3) the fdn was previously cleared to fix another problem, such as an out-of-range block. In any case, you may as well delete the directory entry because "diskrepair" cannot determine what the fdn should be. If the fdn is damaged, you cannot recover the file; however, if the entry merely points to the wrong fdn, the correct fdn is almost certainly unreferenced and will, therefore, show up later in the course of repairing the disk (see Sections 10.10 and 10.11.1).

10.9.10 Out-of-range Fdns

The "format" command reserves a certain number of blocks for fdns. The fdns are numbered from 1 to the appropriate number, which varies depending on the exact form of the "format" command used (see 68xxx UniFLEX Utility Commands). If the fdn number in any directory entry is outside this range of numbers, "diskrepair" reports

```
Out-of-range fdn for "<file_name>".
```

and prompts for permission to delete the directory entry. You may as well give permission because the fdn that should have been pointed to is now probably an unreferenced fdn and will show up at some other time in the process of repairing the disk (see Sections 10.10 and 10.11.1).

10.10 Phase 3--Check Unreferenced Directories

During phase 3, "diskrepair" looks for unreferenced directories. An unreferenced directory is a directory whose fdn is not pointed to by an entry in any directory in the file system. If "diskrepair" finds an unreferenced directory it sends you the message

```
Unreferenced directory.
   fdn=<fdn_num>     type=directory    size=<bytes>
   owner=<owner_name>  time=<time_and_date>
Put in "lost+found"?
```

If you respond to the prompt positively, "diskrepair" names the directory "file<fdn_num>" and tries to add it to the directory "lost+found". If it succeeds, it informs you

```
"<file_name>" put in "lost+found".
Parent fdn was <fdn_num>.
```

After "diskrepair" is finished, you can look at all the files in the lost-and-found directory, decide where they belong, and reconstruct your file system.

The directory "lost+found" is normally created by the command "crdisk" when it creates a system disk. Of course, if for any reason this directory is not on the damaged disk, "diskrepair" cannot put any files in it. If the lost-and-found directory does not exist, "diskrepair" tells you

```
Cannot connect: "lost+found" directory is missing or full.
```

It then prompts you for permission to delete the file.

"Diskrepair" returns this same message if it discovers that the lost-and-found directory is full except for the first time it makes the discovery, when it prints the following message instead:

No room in "lost+found".

It then prompts for permission to delete the file. Once "diskrepair" discovers that the directory "lost+found" is full, it no longer asks you if it should put an unreferenced file in "lost+found". Rather, it simply asks for permission to delete the file.

If the lost-and-found directory is inaccessible (either nonexistent or full), "diskrepair no longer asks you if it should put an unreferenced directory in "lost+found". Rather, it simply asks for permission to delete the file.

If your lost-and-found directory does fill up, you should let "diskrepair" finish, but do not allow it to delete any unreferenced files. Mount the disk if possible and see what files are in "lost+found". If the files are not backed up, you can try to copy them to a backup device if one is available. Once you have successfully copied them, you can delete all entries in "/lost+found" and run "diskrepair" again. Once the disk is fixed, you may want to increase the size of the lost-and-found directory (see Section 11.5.2).

If your disk does not have a lost-and-found directory, you cannot reliably salvage the information in the unreferenced directory or in any of the files that are descendants of that directory. If you do not have backup copies of the unreferenced files, you may try, as a last resort, to create a "lost+found" directory on the damaged disk. If you are successful, you will be able to salvage the unreferenced files. However, creating a file on a damaged disk may simply make the situation worse. You should not try to create a lost-and-found directory until you have salvaged as much information as possible from the disk and are willing to reformat it if your attempt to create it fails.

10.11 Phase 4--Check File and Directory Links

During phase 2, "diskrepair" builds a table which contains an entry for each fdn. The entry shows whether or not the fdn is active and, if it is, what type of file it describes. It also contains space for the link count (the number of directory entries that point to that fdn). During phase 4, "diskrepair" checks this table for certain inconsistencies.

10.11.1 Unreferenced Files

Once the table of fdns is made, it shows whether or not any unreferenced files exist. If the table shows that the link count for a file is 0, that file is unreferenced--that is, no directory entry points to that fdn. Any unreferenced files and any unreferenced directories which were not previously placed in the lost-and-found directory are reported to the user as follows:

```
Unreferenced <file_or_directory>.
   fdn=<fdn_num>     type=<file_type>     size=<bytes>
   owner=<owner_name>  time=<time_and_date>
```

If the file is a directory, "diskrepair" asks if you want to delete it (you had a chance to put it in the lost-and-found directory in phase 3). If it is a regular file and the disk contains a lost-and-found directory that is not yet full, "diskrepair" asks

```
-----><Put in lost+found?
```

If you respond negatively, "diskrepair" asks if it should delete the file. If you neither delete it nor place it in the lost-and-found directory, the structure of the disk remains damaged. If you respond positively to the prompt to insert the file in the lost-and-found directory, "diskrepair" names the file "file<fdn_num>" and adds it to the directory. After "diskrepair" is finished, you can look at all the files in the lost-and-found directory, decide where they belong, and reconstruct your file system.

The directory "lost+found" is normally created by the command "crdisk" when it creates a system disk. Of course, if for any reason this directory is not on the damaged disk, "diskrepair" cannot put any files in it. Instead of asking for permission to put the file in the lost-and-found directory, it tells you

```
Cannot connect: "lost+found" directory is missing or full.
```

It then prompts you for permission to delete the file.

"Diskrepair" returns this same message if it discovers that the lost-and-found directory is full except for the first time it makes the discovery, when it prints the following message instead:

```
No room in "lost+found".
```

It then prompts for permission to delete the file. Once "diskrepair" discovers that the directory "lost+found" is full, it no longer asks you if it should put an unreferenced file in "lost+found". Rather, it simply asks for permission to delete the file.

If your lost-and-found directory does fill up, you should let "diskrepair" finish, but do not allow it to delete any unreferenced files. Try to mount the disk and see what files are in "lost+found". If the files are not backed up, you can try to copy them to a backup device. Once you have successfully copied them, you can delete all entries in "/lost+found" and run "diskrepair" again. Once the disk is fixed, you may want to increase the size of the lost-and-found directory (see Section 11.5.2).

If your disk does not have a lost-and-found directory, you cannot reliably salvage the information in an unreferenced file. If you do not have backup copies of the unreferenced files, you may try, as a last resort, to create a "lost-and-found" directory on the damaged disk. If you are successful, you will be able to salvage the unreferenced files. However, creating a file on a damaged disk may simply make the situation worse. You should not try to do so until you have salvaged as much information as possible from the disk and you are willing to reformat it if your attempt to create the directory fails.

10.11.2 Link Count

While it is building the table of fdns, "diskrepair" keeps track of the link count for each file. If the actual link count as tallied by the process of making the table does not agree with the link count appearing in the fdn, "diskrepair" offers to fix the discrepancy:

```
Link count is <num_1>, should be <num_2>.
-----> Fix?
```

In response to a 'y', it changes the link count in the fdn to match the one determined by the tally. This change has no effect on the rest of the disk.

10.11.3 In-core Fdn List

During phase 4, "diskrepair" also examines the in-core fdn list, which is a partial list of fdns that are available for use (inactive). It tries to verify the number of free fdns in the list. If this number is in error, "diskrepair" responds

```
Bad in-core fdn count.
```

At this time "diskrepair" also checks to see if any in-core fdn which is supposedly free is actually in use. In such a case "diskrepair" tells you

```
Free in-core fdn in use.
```

Finally, "diskrepair" checks for in-core fdns that are duplicated in the list or whose fdn numbers are outside the range of permissible numbers. These errors are reported by the following messages:

```
Duplicate in-core fdns.
Out-of-range in-core fdn.
```

Each of these four messages is followed by the prompt

```
-----> Fix in-core fdn list?
```

There is no reason to deny permission to fix the in-core fdn list because "diskrepair" can do so without affecting the rest of the disk.

10.12 Phase 5--Check Free Lists

During phase 5, the "diskrepair" command makes several checks on the structure of the two free lists--one for the volume space and one for the contiguous-file space. (Because not all systems support contiguous files, the free list for the volume space is usually referred to simply as "the free list".) You can fix any problems encountered in either free list by rebuilding it (see Sections 10.12.1.7 and 10.12.2.5).

10.12.1 Volume Space

"Diskrepair" first validates the structure of the free list for the volume space.

10.12.1.1 Missing blocks

During phase 5, "diskrepair" uses the existing free list to complete the table that it started in phase 1. This table consists of a bit map of the volume space on the disk. Each block in the volume space is represented by a particular bit. When "diskrepair" encounters a claim to a block, it sets the corresponding bit to 1. When the map is complete, all bits should be set to 1. Any bit that is still set to 0 represents a "missing" block--that is, it is allocated neither to a file nor to the free list. "Diskrepair" does not inform you of any missing blocks until it summarizes the status of the free list (see Section 10.12.1.6).

10.12.1.2 Duplicate blocks

If, while completing the bit map, "diskrepair" finds a bit that is already set to 1, it knows that the block represented by that bit is claimed by both the free list and a file. Each time "diskrepair" encounters such a block it reports:

    Block <block_num> duplicated in free list.

10.12.1.3 Out-of-range blocks

While "diskrepair" is evaluating the free list, it checks to make sure that the address of each block is a valid one. All addresses should correspond to blocks within the volume space on the disk. If "diskrepair" discovers any out-of-range blocks, it informs you of them when it summarizes the status of the free list (see Section 10.12.1.6).

10.12.1.4 Out-of-range pointers

Occasionally, "diskrepair" reports

    Out-of-range pointer in free list.

The pointer referred to has to do with the way the operating system maintains the free list. If "diskrepair" finds an out-of-range pointer, it stops checking the free list and immediately prints a summary of the information in the bit map (see Section 10.12.1.1). Because the summary is printed before the table is complete, the number of missing blocks is apt to be large.

10.12.1.5 In-core block list

While it is checking the free list, "diskrepair" determines whether or not the number in the SIR representing the number of free in-core blocks is correct. If it is incorrect, "diskrepair" issues the following message:

```
Bad in-core block count.
```

If the in-core block count is incorrect, "diskrepair" stops checking the free list and immediately prints a summary of the information in the bit map (see Section 10.12.1.1). Because the summary is printed before the table is complete, the number of missing blocks is apt to be large.

10.12.1.6 Summary of the status of the free list

If "diskrepair" finds any problems during phase 5, it summarizes the status of the free list with the following messages:

```
Missing blocks = <num>.
Duplicate blocks in free list = <num>.
Out-of-range blocks in free list = <num>.
```

10.12.1.7 Rebuilding the free list

After reporting any errors found in phase 5, "diskrepair" prompts for permission to rebuild the free list:

```
Invalid free list.
-----> Rebuild?
```

"Diskrepair" reconstructs the free list from the information in the bit map. Rebuilding the free list does not affect any data on the disk. Thus, it is a safe procedure unless "diskrepair" was unable to read the

file ".badblocks" (see Section 10.6). On a large disk, however, the process may take a considerable amount of time (as much as fifteen minutes). If the only problem is missing blocks, which are harmless, you may wish to wait and rebuild the free list at a time that will not inconvenience too many people.

If you give permission for "diskrepair" to rebuild the free list, it enters phase 5B. During this phase it places in the free list every block that is in the volume space but is not allocated to a file and is not in the bad-blocks file. When it finishes rebuilding the free list, it prints the message

Free list rebuilt (<num> blocks).

where <num> is the number of blocks in the reconstructed free list.

If "diskrepair" encounters an I/O error while it is rebuilding the free list, it omits the block from the free list, sending you the following message:

Omitting block <block_num> from free list.

If you specified the 'a' option, "diskrepair" will call the "badblocks" command in order to place the block in the bad-blocks file. Otherwise, the block becomes a missing block (see Section 10.12.1.1). In such a case, you should invoke the "badblocks" command as soon as "diskrepair" terminates.

10.12.2 Checking the Contiguous-File Free-List

After checking and, if necessary, rebuilding the free list for the volume space, "diskrepair" checks the structure of the contiguous-file free-list. Because the two free lists are maintained in different ways, the checks are not identical.

10.12.2.1 Similarities to the free list for the volume space

First, "diskrepair" checks the contiguous-file free-list for missing, duplicate, and out-of-range blocks. If it encounters any duplicate blocks, it reports:

Block <block_num> duplicated in contiguous-file free-list.

It reports the total number of duplicated blocks, as well as any other problems, when it summarizes the status of the contiguous-file free-list (see Section 10.12.2.4).

10.12.2.2 Out-of-order free list

The map of the contiguous-file free-list is an ordered list--that is, the numbers of each block in the list should be greater than the preceding one and less than the following one. If any block is out of order, "diskrepair" sends the following message to standard output:

Contiguous-file free-list map is out of order.

An out-of-order free-list map may cause other problems in the free list, such as a large number of missing blocks. You should allow "diskrepair" to rebuild the free list.

10.12.2.3 Extraneous data

A portion of the contiguous-file free-list map should contain only zeros. If "diskrepair" finds any other data in this part of the free list, it returns the following message:

Extraneous data in contiguous-file free-list map.

You should allow "diskrepair" to rebuild the free list.

10.12.2.4 Summary of the status of the contiguous-file free-list

When "diskrepair" completes its check of the contiguous-file free-list, it summarizes the status of this free list as follows:

Missing contiguous-file blocks = <num>.
Duplicate blocks in contiguous-file free-list = <num>.
Out-of-range blocks in contiguous-file free-list = <num>.

10.12.2.5 Rebuilding the contiguous-file free-list

After reporting any errors it finds in the contiguous-file free-list, "diskrepair" prompts for permission to rebuild it:

```
Invalid contiguous-file free-list.
-----> Rebuild?
```

Rebuilding the contiguous-file free-list is a safe procedure unless "diskrepair" was unable to read the bad-blocks file (see Section 10.6). However, depending on the size of the contiguous-file space, the process may take a considerable amount of time. If the only problem is missing blocks, which are harmless, you may wish to wait and rebuild the free list at a time that will not inconvenience too many people.

If you give permission for "diskrepair" to rebuild the contiguous-file free list, "diskrepair" places in the list every block that is in the contiguous-file space but is not allocated to a file and is not in the bad-blocks file. When it finishes rebuilding the free list, it prints the message

```
Contiguous-file free-list rebuilt (<num> blocks).
```

As you use the contiguous-file free-space, it may become fragmented. If fragmentation is excessive, the operating system cannot maintain a complete map of the free-list. In such a case, the problem becomes apparent to "diskrepair" as it tries to rebuild the contiguous-file free-list, and it reports

```
Contiguous-file free-list too fragmented to fit in map.
<num> blocks lost.
```

This degree of fragmentation is not considered a logical error on the disk; however, you will not be able to access all the blocks in the contiguous-file space.

If "diskrepair" encounters an I/O error in one of the blocks allocated for the map of the contiguous-file free-space, it tells you

```
Error encountered rebuilding contiguous-file free-list map.
```

Although this problem is unlikely to arise, its consequences are serious, for you must reformat the disk to correct the problem. You may, however, be able to salvage some data first (see Chapter 11).

10.13 Phase Six--Check SIR Information

During its final phase of operation, "diskrepair" makes several more checks on the system information record (SIR).

10.13.1 Free Fdn Count

If the number stored in the SIR which represents the total number of free fdns on the disk does not agree with the number calculated by "diskrepair" as it moves through the fdns one by one, "diskrepair" reports

```
Incorrect count of free fdns in SIR.
----->Fix?
```

In response to a positive answer, "diskrepair" changes the number in the SIR. This change has no effect on anything else on the disk.

10.13.2 Free Block Count

Similarly, if the number stored in the SIR which represents the total number of free blocks on the disk does not agree with the number calculated by "diskrepair" as it checks allocated blocks, "diskrepair" reports

```
Incorrect count of free blocks in SIR.
----->Fix?
```

In response to a positive answer, "diskrepair" changes the number in the SIR. This change has no affect on anything else on the disk.

10.13.3 Checking the "Mount Flag"

Each time you mount a device, the operating system sets a flag on the disk in the mounted device (see Section 3.8) Normally, the flag is cleared by the "unmount" command. If someone removes a disk from a mounted device without first unmounting it, the flag remains set. If "diskrepair" finds that this "mount flag" is set, it reports

```
"Mount flag" in SIR should be cleared.
-----> Clear "mount flag" in SIR?
```

In response to a positive answer, "diskrepair" clears the flag. This change has no affect on anything else on the disk.

10.13.4 State of the Disk

If "diskrepair" modifies the disk, it prints the following message on completion of its tests:

```
=== Disk modified. ===
```

If it does not modify the disk and the 'v' option was not in effect, it prints whichever of the following messages is appropriate:

```
=== Disk OK. ===

=== Disk needs repair! ===
```

If you did specify the 'v' option, "diskrepair" assumes that its output has already indicated whether or not you need to repair the disk.

"Diskrepair" may encounter more problems than it can fix during one run. For example, it can only handle a certain number of duplicate or out-of-range blocks (see Sections 10.7.2 and 10.7.3). If "diskrepair" cannot fix all the errors it encounters, or if it encounters a read or write error but you choose to continue operation, it prints the following message when it is finished:

```
=== Problems encountered.  Rerun "diskrepair".
```

If appropriate, this message occurs no matter what options you specify.

If the 'a' option is in effect and the disk is so badly damaged that all the iterations of "diskrepair" fail to fix it, the following message appears:

```
=== Repair of "<dev_name>" not complete after <num> tries. ===
```

10.13.5 Updating the SIR

When it is finished modifying the disk, "diskrepair" must update the SIR so that it corresponds to the new structure of the disk. If the device being checked is not the root device, "diskrepair" simply rewrites the SIR with the correct information. If, however, the SIR of the root device must be updated, "diskrepair" kills all tasks running on the system and locks up the system so that no new tasks can begin. It then modifies the SIR. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory. After updating the SIR, "diskrepair" stops the system and prints the following message:

```
=== Intentional system stop.  Reboot UniFLEX. ===
```

If you receive this message, you must reboot the system before you can proceed.

If "diskrepair" encounters an I/O error when it tries to make any changes in the SIR, it prints the following message:

```
ERROR UPDATING SIR.  DISK IS BAD!
```

This error is not only fatal to "diskrepair", it also means that you must rebuild your system disk and that you cannot salvage any data from it.

10.14 I/O Errors

If "diskrepair" encounters an I/O error, it first prints a generic message telling you that it could not read from or write to the disk. It then prompts you for permission to continue and, if you respond positively, usually gives you a more detailed message which tells you what it was trying to do when it encountered the error (see Section 10.14). We suggest that you respond negatively to the prompt to continue the first time "diskrepair" reports an I/O error and that you immediately rerun "diskrepair". It is possible--though unlikely--that the I/O error is a soft one and will not recur. If you receive the same error message again, rerun "diskrepair" with the 'a' option.

An alphabetic list of the I/O error messages follows. An explanation accompanies each message.

Cannot fix duplicated block.
"Diskrepair" found a duplicated block in an fdn, but when it tried to read the fdn to find out the owner, the size, and so forth, it encountered an I/O error.

Cannot fix error in file size.
"Diskrepair" encountered an I/O error when it tried to change the file size in the fdn of the file in question.

Cannot fix error in link count.
"Diskrepair" encountered an I/O error while trying to read the link count from an fdn.

Cannot fix out-of-range block.
"Diskrepair" found an out-of-range block in a file, but was unable to read the fdn to find out the owner, size, and so forth.

Cannot fix unknown file.
"Diskrepair" encountered an I/O error trying the read the fdn of a file whose type was unknown.

Cannot fix unreferenced fdn.
"Diskrepair" found an unreferenced file or directory, but got an I/O error while trying to read the fdn to determine the owner, size, and so forth.

Cannot truncate directory.
"Diskrepair" tried to truncate the size of a directory, but it encountered an I/O error when it tried to write the new size in the fdn.

Connect to "lost+found" unsuccessful.
"Diskrepair" was unable to write to the directory ".lost+found" and therefore could not put the unreferenced fdn into that directory. If you receive this error message, "diskrepair" cannot put any files in "/lost+found". Eventually, you must delete the unreferenced files, but you may be able to salvage some data first (see Chapter 8).

Error opening "<dir_name>".
Directory completely ignored!
"Diskrepair" was trying to scan all the directories on the disk, but due to an I/O error it was unable to open the directory <dir_name>. All files and subdirectories in that directory, as well as any files in the subdirectories (and so forth) are now unreferenced.

Directory partially ignored.
"Diskrepair" was trying to scan all directories on the disk, but due to an I/O error in one of the blocks in <dir_name> it was unable to read the entire directory. It therefore lacks information on the directory entries that were maintained in that particular block and in all subsequent blocks in <dir_name>. Any descendants of these files are now unreferenced.

Fdn not updated.
"Diskrepair" encounted an I/O error when it tried to write to an fdn.

Fdns <fdn_num_1> to <fdn_num_2> skipped.
"Diskrepair" encountered an I/O error while reading a block of fdns. It is, therefore, unable to read any of the fdns in that block. Since each block contains eight fdns, as many as eight files may be inaccessible. Such a situation usually causes other problems, but you may be able to salvage some data. If the error is in the first fdn block, you must reformat the disk. You cannot salvage any data from the disk.

File or directory not deleted.
"Diskrepair" tried to delete a file or a directory but was unable to write to the corresponding fdn.

Free list check aborted.
"Diskrepair" encountered an I/O error while trying to read the free list. It stops trying to read the free list, prints the message, "Invalid free list", and tries to rebuild the free list. The free list may, however, contain a bad block. When "diskrepair" is done, you should rerun it with the ´a´ option (see Section 10.2.1).

Links for connected file may be bad.
When "diskrepair" puts a file in the lost-and-found directory, it tries to correct the link count. This message appears if "diskrepair" encountered an I/O error while trying to fix the link count.

Omitting block <block_num> from free list.
In the process of rebuilding the free list, "diskrepair" encountered an I/O error. By default, it leaves the offending block out of the free list; therefore, that block becomes a missing block. (If you specify the ´a´ option, "diskrepair" calls "badblocks" to place the block in the bad-blocks file.) This situation causes no immediate problems; however, the next time you run "diskrepair" it may put the bad block back in the free list, thus creating the potential for an I/O error during normal operation. You should use the "badblocks" command to put the offending block in the file "/.badblocks".

Part of file may be ignored.
The operating system encountered an I/O error while trying to read an fdn during phase 1. As a result "diskrepair" may not be aware of all the blocks that are supposed to be in the file and may release them to the free list.

10.15 Index of Error Messages

This section contains an index to the error messages which are not caused by I/O errors.

Bad "." in "<dir_name>", 10.29
Bad ".." in "<dir_name>", 10.29
Bad ".badblocks" file. File ignored, 10.17
Bad in-core block count, 10.37
Bad in-core fdn count, 10.35
Bad-blocks file too large, 10.18
Block <block_num> duplicated in contiguous-file free-list, 10.39
Block <block_num> duplicated in free list, 10.36
"Blockcheck" terminated abnormally (status = <num>), 10.13

Cannot access fdn for root device, 10.12
Cannot access fdn for standard error, 10.14
Cannot access fdn for standard output, 10.14
Cannot call "/etc/blockcheck", 10.12
Cannot call "/etc/fdncheck", 10.21
Cannot check a "backup" disk, 10.10
Cannot connect: "lost+found" directory is missing or full, 10.31, 10.33
Cannot delete ".badblocks", 10.27, 10.28
Cannot delete root directory, 10.27, 10.28
Cannot open device, 10.13
Cannot read root directory, 10.22
Cannot read sector zero, 10.10
Cannot read System Information Record, 10.14
Cannot suspend running tasks, 10.12
Conflicting options, 10.9
Contiguous-file free-list map is out of order, 10.39
Contiguous-file free-list too fragmented to fit in map, 10.40
Contiguous-file size error in fdn <fdn_num>, 10.19

Device is busy, 10.11
Directory too large: "<dir_name>", 10.23
Disk modified, 10.42
Disk needs repair, 10.42
Disk too large or bad size in SIR, 10.14
Duplicate block <block_num> in fdn <fdn_num>, 10.20

Duplicate block in "<file_name>", 10.28
Duplicate blocks in contiguous-file free-list = <num>, 10.39
Duplicate blocks in free list, 10.37
Duplicate in-core fdns, 10.35

Error checking ".badblocks" file. File ignored, 10.17
Error encountered rebuilding contiguous-file free-list map, 10.40
ERROR UPDATING SIR. DISK IS BAD, 10.43
"/etc/blockcheck" is invalid, 10.12
"/etc/fdncheck" is invalid, 10.21
Extraneous data in contiguous-file free-list map, 10.39

"Fdncheck" terminated abnormally (status = <num>), 10.21
File size error in fdn <fdn_num> (size <num_1>, found <num_2>), 10.19
File size error in fdn <fdn_num>, file "<file_name>", 10.26
"<file_name>" put in "lost+found", 10.31
Free in-core fdn in use, 10.35

Inactive fdn for "<file_name>", 10.30
Incorrect count of free blocks in SIR, 10.41
Incorrect count of free fdns in SIR, 10.41
Intentional system stop. Reboot UniFLEX, 10.43
Invalid contiguous-file free-list, 10.40
Invalid file name: "<file_name>", 10.24
Invalid free list, 10.37
Invalid option: '<char>', 10.9

Length of root directory is zero, 10.22
Link count is <num_1>, should be <num_2>, 10.34

Missing blocks = <num>, 10.37
Missing contiguous-file blocks = <num>, 10.39
"Mount flag" in SIR should be cleared, 10.42

Nesting too deep at "<dir_name>", 10.24
No device specified, 10.9
No room in "lost+found", 10.32, 10.34
No such device, 10.9
Not a block device, 10.9

Odd size for directory "<dir_name>", 10.24
Omitting block <block_num> from free list, 10.38
Out-of-range block in "<file_name>", 10.26
Out-of-range block in fdn <fdn_num>, 10.19
Out-of-range blocks in contiguous-file free-list = <num>, 10.39
Out-of-range blocks in free list = <num>, 10.37
Out-of-range fdn for "<file_name>", 10.30
Out-of-range in-core fdn, 10.35
Out-of-range pointer in free list, 10.36
Output directed to device under test, 10.13

Paging space overlaps contiguous-file space, 10.16
Parent fdn was <fdn_num>, 10.31
Permission denied, 10.10
Problems encountered. Rerun "diskrepair", 10.42

Repair of "<dev_name>" not complete after <num> tries, 10.42
Root fdn is not a directory, 10.22

Too many duplicate blocks, 10.20
Too many fdn blocks: <num>, 10.15
Too many out-of-range blocks, 10.20

Unknown file type for "<file_name>", 10.30
Unmount error <error_num>, 10.11
Unreferenced <file_or_directory>, 10.33
Unreferenced directory, 10.31

Volume space overlaps contiguous-file space, 10.16
Volume space overlaps paging space, 10.15

WARNING: <num> unassigned blocks between paging space and contiguous-file space, 10.17
WARNING: <num> unassigned blocks between volume space and contiguous-file space, 10.17
WARNING: <num> unassigned blocks between volume space and paging space, 10.16
WARNING: No "." entry for "<dir_name>", 10.29
WARNING: No ".." entry for "<dir_name>", 10.29
WARNING: Previous links to fdn <fdn_num>, 10.27
WARNING: Too many "." entries for "<dir_name>", 10.29
WARNING: Too many ".." entries for "<dir_name>", 10.29

Chapter 11

Recovering from Problems

11.1 Introduction

When a system crashes or a program malfunctions, some parts of the system disk may be damaged. You can always rebuild the original UniFLEX system by executing the "crdisk" command. If the damage is extensive, you may have no other choice, but remember that "crdisk" reformats the hard disk, removing any files you have added to the system. You may, therefore, prefer to try to recover your damaged system.

This chapter describes how to proceed in the event of a system crash. It also presents solutions for several other kinds of problems.

11.2 After a Crash

When your system crashes, your course of action depends on how badly the system disk is damaged. This section provides a step-by-step procedure for you to follow in assessing and, if possible, fixing the damage. As you become more familiar with the system and with the "diskrepair" command in particular, you will no doubt find shortcuts through this procedure. However, it is presented here as a guide to the system manager who is unfamiliar with the operating system.

Unfortunately, we cannot anticipate every possible failure, but in constructing this procedure we have tried to anticipate the most likely ones. Should something happen to your system that is not covered by this procedure, feel free to call us for technical assistance.

1. Boot from your system disk (see Note 1).
   A. If you cannot boot, go to 2.
   B. If you can boot, go to 40.

2. Boot from your master floppy (see Notes 2 and 1)
   Does the operating system return a prompt ("++")?
   A. No--go to 3.
   B. Yes--go to 6.

3. Check your hardware.
   A. If the hardware is not faulty, go to 4.
   B. If the hardware is faulty, go to 5.

4. Replace your master floppy (see Note 3).
   Go to 2.

5. Shut down the operating system (see Note 4).
   Fix the hardware.
   Go to 1.

6. Run "diskrepair" on the system disk from the master floppy (see Note 5).
   Does "diskrepair" enter phase 1 (see Note 6)?
   A. No--go to 7.
   B. Yes--go to 16.

7. Is your existing backup sufficient?
   A. No--go to 8.
   B. Yes--go to 15.

8. Mount the system disk (see Note 7).
   A. If you cannot mount the system disk, go to 9.
   B. If you can mount the system disk, go to 11.

9. Clear the "mount flag" on the system disk (see Note 8).
   Mount the system disk (see Note 7).
   A. If you cannot mount the system disk, go to 10.
   B. If you can mount the system disk, go to 11.

10. Rebuild your system disk (see Section 1.1). You cannot salvage any data from the damaged disk.

11. Is a spare device available (see Note 9)?
    A. No--go to 12.
    B. Yes--go to 14.

12. Copy the kernel of the operating system from the master floppy to the system disk (see Note 10).
    A. If the command fails, go to 10.
    B. If the command is successful, go to 13.

13. Tune the copy of the operating system that is on the system disk (see Note 11).
    Shut down the operating system (see Note 4).
    Boot from your system disk (see Note 1).
    A. If you still cannot boot from the system disk, go to 10.
    B. If you can boot from the system disk, go to 40.

14. Prepare fresh backup medium.
    Mount the system disk if it is not already mounted (see Note 7).
    Back up files from the system disk to the backup device (see Note 12).
    A. If the backup procedure fails, go to 12.
    B. If the backup procedure is successful, go to 15.

15. Rebuild your system disk (see Section 1.1).
    Restore your own files from the backup (see Note 13).

16. Does "diskrepair" return an error as it tries to rewrite the system information record? (The message is "ERROR UPDATING SIR. DISK IS BAD!")
    A. No--go to 17.
    B. Yes--go to 10.

17. Does "diskrepair" need to delete data in order to repair the disk?
    A. No--go to 18.
    B. Yes--go to 19.

18. Let "diskrepair" fix the disk (see Note 14).
    Shut down the operating system (see Note 4).
    Boot from the system disk (see Note 1).
    A. If you still cannot boot from the system disk, go to 11.
    B. If you can boot from the system disk, go to 40.

19. Let "diskrepair" make any changes to the disk that it requests permission for except changes that require deletions (see Note 15).
    Make a list of files that "diskrepair" needs to delete in order to repair the disk.
    Is your existing backup sufficient?
    A. No--go to 20.
    B. Yes--go to 30.

20. Is a spare device available (see Note 9)?
    A. No--go to 21.
    B. Yes--go to 29.

21. Copy the kernel of the operating system from the master floppy to the system disk (see Note 10).
    A. If the command fails, go to 22.
    B. If the command is successful, go to 25.

22. Let "diskrepair" fix the disk (see Note 14). You cannot salvage the files it needs to delete.
    Shut down the operating system (see Note 4).
    Boot from the system disk.
    A. If you still cannot boot from the system disk, go to 10.
    B. If you can boot the system disk, go to 23.

23. Does the operating system return a prompt ("++")?
    A. No--go to 41.
    B. Yes--go to 24.

24. The operating system should now be intact except for the files that "diskrepair" deleted (see Note 13).

25. Tune the copy of the operating system that is on the system disk (see Note 11).
    Shut down the operating system (see Note 4).
    Boot from the system disk (see Note 1).
    A. If you still cannot boot from the system disk, go to 26.
    B. If you can boot from the system disk, go to 27.

26. Reset the hardware.
    Boot from the master floppy (see Notes 2 and 1).
    Go to 22.

27. Does the operating system return a prompt ("++")?
    A. No--go to 28.
    B. Yes--go to 41.

28. Back up the files that "diskrepair" needs to delete in order to repair the disk (see Note 12).
    Let "diskrepair" fix the disk (see Note 14).
    Restore files from your backup (see Note 13).
    The operating system should now be intact.

29. Prepare fresh backup medium (floppy disk or tape).
    Back up the files that "diskrepair" needs to delete in order to repair the disk (see Note 12).
    A. If the backup procedure fails, go to 22.
    B. If the backup procedure is successful, go to 30.

30. Let "diskrepair" fix the disk (see Note 14).
    Restore files from backup (see Note 13).
    Shut down the operating system (see Note 4).
    Boot from the system disk (see Note 1).
    A. If you still cannot boot the from system disk, go to 31.
    B. If you can boot the system disk, go to 37.

31. Boot from the master floppy (see Notes 2 and 1).
    Is your existing backup sufficient?
    A. No--go to 32.
    B. Yes--go to 36.

32. Copy the kernel of the operating system from the master floppy to the system disk (see Note 10).
    A. If the command fails, go to 33.
    B. If the command succeeds, go to 34.

33. If your existing backup is insufficient, prepare fresh backup medium (tape or floppy disk) and back up the system disk (see Note 12).
    Go to 15.

34. Tune the copy of the operating system that is on the system disk (see Note 11).
    Shut down the operating system (see Note 4).
    Boot from the system disk (see Note 1).
    A. If you still cannot boot from the system disk, go to 35.
    B. If you can boot from the system disk, go to 37.

35. Reset the hardware.
    Boot from the master floppy (see Notes 2 and 1).
    Go to 33.

36. You can either go to 15, or, if you prefer, you can try to make your system disk bootable by going to 32.

37. Does the operating system return a prompt ("++")?
    A. No--go to 38.
    B. Yes--system should be intact.

38. Reset the hardware.
    Boot from the master floppy (see Note 12).
    Mount the system disk (see Note 7).
    Copy the files "/etc/init", "/etc/login", "/bin/shell", and "/dev" from the master floppy to the system disk (see Note 16).
    A. If the copy procedure fails, go to 33.
    B. If the copy procedure is successful, go to 39.

39. Shut down the operating system (see Note 4).
    Boot from the system disk (see Note 1).
    Does the operating system return a prompt ("++")?
    A. No--go to 33.
    B. Yes--system should be intact.

40. Does the operating system return a prompt ("++")?
    A. No--Go to 41.
    B. Yes--go to 55.

41. Reset the hardware.
    Boot from the master floppy (see Notes 2 and 1).
    Does the operating system return a prompt ("++)"?
    A. No--go to 42.
    B. Yes--go to 45.

42. Check your hardware.
    A. If the hardware is not faulty, go to 43.
    B. If the hardware is faulty, go to 44.

43. Replace your master floppy (see Note 3).
    Go to 41.

44. Shut down the operating system (see Note 4).
    Fix the hardware.
    Go to 1.

45. Run "diskrepair" on the system disk from the master floppy (see Note 5).
    Does "diskrepair" enter phase 1 (see Note 6)?
    A. No--go to 46.
    B. Yes--go to 51.

46. Is your existing backup sufficient?
    A. No--go to 47.
    B. Yes--go to 15.

47. Is a spare device available (see Note 9)?
    A. No--go to 10.
    B. Yes--go to 48.

48. Mount the system disk (see Note 7).
    A. If you cannot mount the system disk, go to 49.
    B. If you can mount the system disk, go to 50.

49. Clear the "mount flag" (see Note 8).
    Mount the system disk (see Note 7).
    A. If you still cannot mount the system disk, go to 10.
    B. If you can mount the system disk, go to 50.

50. Prepare fresh backup medium.
    Back up files from the system disk to the backup device (see Note 12).
    Go to 15.

51. Does "diskrepair" return an error as it tries to rewrite the system information record? (The message is "ERROR UPDATING SIR. DISK IS BAD!")
    A. No--go to 52.
    B. Yes--go to 10.

52. Let "diskrepair" make any changes to the disk that it requests permission for except changes that require deletions (see Note 15). Do not allow it to delete any files except "/etc/init", "/etc/login", "/bin/shell", and any files in "/dev".
    Mount the system disk (see Note 7).
    Copy "/etc/init", "/etc/login", "/bin/shell", and "/dev" from the

master floppy to the system disk (see Note 16).
A. If the copy procedure fails, go to 46.
B. If the copy procedure is successful, go to 53.

53. Shut down the operating system (see Note 4).
Boot from the system disk (see Note 1).
Does the operating system return a prompt ("++")?
A. No--go to 54.
B. Yes--go to 55.

54. Reset the hardware.
Boot from the master floppy (see Notes 2 and 1).
Go to 2.

55. Run "diskrepair" on the system disk from the system disk (see Note 5).
Does "diskrepair" enter phase 1 (see Note 6)?
A. No--go to 56.
B. Yes--go to 58.

56. Is your existing backup sufficient?
A. No--go to 57.
B. Yes--go to 15.

57. Prepare fresh backup medium.
Back up files from the system disk to the backup device (see Note 12).
A. If the backup procedure fails, go to 10.
B. If the backup procedure is successful, go to 15.

58. Does "diskrepair" return an error as it tries to rewrite the system information record? (The message is "ERROR UPDATING SIR. DISK IS BAD!")
A. No--go to 59.
B. Yes--go to 56.

59. Does "diskrepair" need to delete data in order to repair the system disk?
A. No--go to 60.
B. Yes--go to 61.

60. Allow "diskrepair" to fix the system disk (see Note 14).
System should be intact.

61. Is your existing backup sufficient?
A. No--go to 62.
B. Yes--go to 64.

62. Let "diskrepair" make any changes to the disk that it requests permission for except changes that require deletions (see Note 15).
Make a list of files that "diskrepair" needs to delete in order to repair the disk.
Prepare fresh backup medium.
Back up the files "diskrepair" needs to delete.
A. If the backup procedure fails, go to 63.
B. If the backup procedure is successful, go to 64.

63. Let "diskrepair" fix the system disk (see Note 14).
You cannot salvage the files it needs to delete.
System should be intact except for the deleted files (see Note 13).

64. Let "diskrepair" fix the system disk (see Note 14).
Restore the deleted files from your backup.
System should be intact.

11.3 Notes

1. The system disk is the hard disk which you usually boot from. You have succeeded in booting your system if the banner message tells you how much user memory is available. If the system fails to respond, if the ROM informs you that it cannot find the file "/uniflex", or if the banner message stops short of telling you how much user memory is available, the boot procedure has failed.

2. The master floppy is the disk from which you booted your system for the first time. Its label designates it as the "System Floppy".

3. If you need to replace your master disk, send the disk and a note describing the problem to

Technical Systems Consultants
111 Providence Rd.
Chapel Hill, NC 27514

4. To shut down the system execute the following command:

```
stop
```

5. In this situation you are running "diskrepair" only to assess the extent of the damage to the system disk. You should therefore invoke the command with the 'n' and 'v' options as shown here:

```
diskrepair /dev/w0 +nv
```

6. When the 'v' option is in effect, "diskrepair" sends to standard output messages telling you what phase it is in. If "diskrepair" aborts before printing any such messages, it did not enter phase 1.

7. Mount the system disk with the following command:

```
mount /dev/w0 /usr0
```

8. To clear the "mount flag", invoke "diskrepair" with only the 'M' option:

```
diskrepair /dev/w0 +M
```

9. Your system supports a spare device if it supports either a streaming tape or enough hardware so that you can use one floppy disk drive to boot from your master floppy and still have a disk drive (for a floppy or a hard disk) available as a backup device.

10. To copy the kernel of the operating system from the master disk to your system disk, invoke the following command:

```
new_system
```

11. Before you can try to boot to your system disk, you must tune the copy of the operating system that is on that disk so that the root, pipe, and paging devices are all the system disk. You can do so with the following command:

```
tune /usr0/uniflex root_dev=/dev/w0 page_dev=/dev/w0 swap_dev=/dev/w0
```

12. Rebuilding a system destroys all the data on the system disk. If you need to rebuild your system and you are not well backed up, you will want to try to copy all the data you need from your system disk to the backup device. You should not copy files that are part of the operating system itself because the process of building a system will reconstruct them in their original form.

    You must therefore perform a selective backup. You can do so either by specifying every directory you want to back up or by invoking the "backup" command with the 'P' option so that it prompts you for permission to back up each file. Detailed instructions on the use of the "backup" command are in the manual entitled 68xxx UniFLEX Utility Commands.

    If "diskrepair" failed to enter phase 1, you can have no idea of which files contain out-of-range blocks or other I/O errors. The "backup" command cannot continue to copy a file once it encounters an I/O error in that file. Instead, it returns an I/O error. In

such a case your backup includes only part of the damaged file--all blocks up to but not including the damaged block. Section 11.4 provides detailed instructions on trying to recover a file that contains an I/O error.

13. If you need to restore files that are part of the UniFLEX Operating system, you should restore them directly from your master floppy disks. The first of these floppy disks (the one labeled as System Floppy) is a bootable disk; the others are backup disks--that is, they were created by the "backup" command and can only be read by that command. In order to determine which files are on the System Floppy, mount it and invoke the following command:

```
dir +lda
```

In order to determine which files are on the rest of the disks, invoke the "backup" command in catalog mode. Once you have located the files you need to restore, you can copy those on the System Floppy to your system disk with the "copy" command. To copy files on any of the other floppy disks, you must invoke the "backup" command in restore mode.

If a file you need to restore resides on both the disk labeled "System Utilities for <type_of_hardware>" and on one of the disks labeled "Standard System Utilities", restore the version on the machine-specific disk. In addition, if you must restore your init-control file, restore the version on the machine-specific disk.

You restore your own files to the system disk by using the "backup" command in restore mode (see Note 11). If any of the files you backed up contained duplicate blocks, the restored versions of some of those files will contain incorrect information. If the file is a binary file, you probably cannot recover it. If it is a text file, you may be able to reconstruct (from memory or a hard copy) the part of the file that is incorrect.

If you were able to run "diskrepair" on the damaged disk, you will know which files contained duplicate blocks and can check each one to assess the damage. If "diskrepair" failed to enter phase 1, you will have no idea which files may cause problems.

14. When you need to let "diskrepair" repair the disk completely, you can either invoke the command without any options, or invoke it in the following way:

```
/etc/diskrepair /dev/w0 +pv
```

and respond with a 'y' whenever it asks for permission to make a repair.

15. To prevent "diskrepair" from deleting any data from the system disk, invoke the command as follows:

```
/etc/diskrepair /dev/w0 +pv
```

and respond with an 'n' to any request to delete a file or a directory.

16. To copy these files to the system disk, invoke the following commands:

```
copy +P /etc/login /usr0/etc/login
copy +P /etc/init /usr0/etc/init
copy +P /bin/shell /usr0/bin/shell
copy +dP /dev /usr0/dev
```

11.4 Recovering Files Containing I/O Errors

If a file contains an out-of-range block or any other I/O error, you cannot access the entire file. At the very least, that part of the file that is associated with the out-of-range block is inaccessible. If the file is a binary file, you cannot salvage any of it. If, however, the file is a text file, you may be able to recover most of it with the "head" and "tail" commands.

In order for you to be able to salvage a file containing an out-of-range block, you must have a free disk device available to use as a backup device. A streaming tape is insufficient.

If your system does not have a spare backup device, you must boot from the damaged disk (see Section 1.2) and use your floppy disk drive as a backup device. If you have a spare disk device, you should boot from your master disk and mount the damaged disk for reading only with the following command:

```
/etc/mount <dev_name> <dir_name> +r
```

To salvage the file, use the "head" command repeatedly with different arguments until you find the largest argument that you can successfully use. Then, execute the command with this argument and redirect the output to a temporary file on the undamaged disk.

Repeat the procedure for the "tail" command. Note that, because a file always starts at the beginning of a block, the "head" command fails with an argument whose value is a multiple of 512 (the number of bytes in a block) plus 1. You cannot predict, however, where the "tail" command will fail because the end of the file is not necessarily the end of a block. Once you have found the limit of the "tail" command, repeat that command and append the output to the temporary file containing the first part of the file.

The temporary file now contains all but one block of the original file. You cannot salvage the data associated with the out-of-range block, but you have recovered the rest of the file. Now you can rerun "diskrepair" and allow it to delete the file containing the out-of-range block. When "diskrepair" is complete, rename your temporary file with the name of the original file and move it back to the appropriate directory on the newly repaired disk. The file is not complete, but it is a lot better than no file.

If you have 68xxx UniFLEX Utilities Package I, you can simplify this recovery procedure by invoking the "skipbad" command.

11.5 Miscellaneous Repairs

11.5.1 Fixing Missing "." and ".." Files

If "diskrepair" finds a missing "." or ".." file, it reports the error (see Section 10.9.7) but makes no attempt to fix it. In general, however, the situation is simple to fix. To create the correct "." file you simply link the directory in question to the file "." in that directory. Similarly, to create the correct ".." file you link the parent of the directory in question to the ".." file in the directory.

For example, if the directory "/usr/larry/tests" is missing both the "." and ".." entries, you can probably correct the situation with the following two commands:

```
link /usr/larry/tests /usr/larry/tests/. +d
link /usr/larry /usr/larry/tests/.. +d
```

11.5.2 Expanding the Directory "lost+found"

When the operating system creates a directory, it allocates one block for it. Because an entry in the lost-and-found directory uses 16 bytes, (each name is of the form "file<fdn_num>"), the first block of this directory can hold thirty-two entries. However, the first two entries in the first block of any directory must be "." and "..". Thus, a newly created lost-and-found directory has room for thirty entries before another block must be allocated to that directory.

The directory "lost+found" is created by the program "crdisk". If you wish to increase beyond thirty the number of entries it can hold, use the following procedure. You should not try to increase the size of the lost-and-found directory on a damaged disk unless the free space is intact.

1. Boot the system with the disk containing the lost-and-found directory whose size you want to increase.

2. Change directories using the command

   ```
   chd /lost+found
   ```

3. Create as many files as necessary to force the operating system to allocate another block to the directory. If the directory is empty (except for the "." and ".." entries) and is one block long, you must add thirty-one files to allocate another block to the directory. After that you must add thirty-two files for each block you want to add.

4. Verify that the directory is as large as you want by issuing the command

   ```
   dir +1 /
   ```

   Included in the information shown by this command is the size of the directory in blocks.

5. Delete all the files you just created. The number of unreferenced files that "diskrepair" can fit into the directory is

(<size_in_blocks> * 32) - 2

11.6 Who Owns What?

Sometimes a program fails to perform as expected because the proper user is no longer the owner. In the original version of the operating system (the version shipped on the master floppies), the user "system" owns all files.

11.7 Setting the User ID Bit

In order to function properly, some commands must have the user ID bit set. Setting this bit grants to any user executing the program the same privileges as the owner of the program for the duration of the task. You can set or clear this bit, which is called the 's' permission bit, with the "perms" command.

If you rebuild any files, be sure that you use the set the 's' bit on the following files:

1. /bin/copy
2. /bin/copy-dir
3. /bin/crdir
4. /etc/init
5. /etc/insp
6. /etc/login
7. /etc/print
8. /usr/bin/at
9. /usr/bin/backup
10. /usr/bin/info
11. /usr/bin/mail
12. /usr/bin/newuser
13. /usr/bin/password
14. /usr/bin/relinfo
15. /usr/bin/su

The correct form of the command is

perms s+ <file_name> [<file_name_list>]

Appendix A

A Generic Init-Control File

This appendix contains a line-by-line description of a generic init-control which is similar to the init-control file shipped with your operating system. Lines consisting solely of the sequence "+*" (comments without text) are not documented.

Line 1: +* Keep terminal open
  Comment describing line 2.

Line 2: +o +
  Open the console as standard input, standard output, and standard error.

Line 4: +* Example /etc/.init.control
  Comment describing this file.

Line 6: +* Automatically update all disks every 30 seconds
  Comment describing line 7.

Line 7: +u 30
  Update the file system every 30 seconds.

Line 9: +* Set up exit conditions
  Comment describing lines 11-14.

Line 11: +01 -> shut_soft
  On receipt of a HANGUP signal (signal number 1) branch to the line labeled "shut_soft" (line 88).

Line 12: +04 -> shut_down
  On receipt of a SIGEMT signal (signal number 4) branch to the line labeled "shut_down" (line 97).

Line 13: +06 -> shut_down
  On receipt of a SIGPIPE signal (signal number 6) branch to the line labeled "shut_down" (line 97).

Line 14: +08 -> shut_soft
  On receipt of a SIGTRACE signal (signal number 8) branch to the line labeled "shut_soft" (line 88).

Line 15: +* Check if root device needs cleaning
  Comment describing line 16.

Line 16: +c -> diskrepair
  Branch to the line labeled "diskrepair" (line 113) if the system disk is not clean.

Line 18: :boot
Assign the label "boot" to line 18.

Line 20: +* Update the history file
Comment describing line 22.

Line 22: +h bt
Put the entry "bt" (system was booted), followed by a time stamp, into the file "/act/history".

Line 24: +* Set date from hardware clock
Comment describing line 26.

Line 26: date +s
Execute the "date" command, taking the time from the system's hardware clock. If the system does not have a hardware clock, this command has no effect.

Line 28: +* Create the mount table
Comment describing line 30.

Line 30: create /etc/mtab
Create the file that contains the mount table for the system. The operating system uses the mount table to keep track of which devices are mounted.

Line 32: +* Enter single-user mode
Comment describing lines 34-51.

Line 34: :single_user
Assign the label "single_user" to line 34.

Line 35: +* Keep terminal open
Comment describing line 36.

Line 36: +o +
Open the console as standard input, standard output, and standard error.

Line 38: +* Set up exit conditions
Comment describing lines 40-44.

Line 40: +01 -> shut_soft
On receipt of a HANGUP signal (signal number 1) branch to the line labeled "shut_soft" (line 88).

Line 41: +04 -> shut_down
On receipt of a SIGEMT signal (signal number 4) branch to the line labeled "shut_down" (line 97).

Line 42: +06 -> shut_down
On receipt of a SIGPIPE signal (signal number 6) branch to the line labeled "shut_down" (line 97).

Line 43: +08 -> shut_soft On receipt of a SIGTRACE (signal number 8) branch to the line labeled "shut_soft" (line 88).

Line 44: +* Update the history file
Comment describing line 45.

Line 45: +h su
Put the entry "su" (system entered single-user mode), followed by a time stamp, into the file "/act/history".

Line 46: +* Establish single-user environment
Comment describing line 47.

Line 47: create /act/utmp
Create the file "/act/utmp". When the system is in multi-user mode, the operating system writes the name and terminal number of each user who is using the system to this file. It also enters the time at which the user logged in. If the length of the file "utmp" is 0, as it is at the time of its creation, the system is in single-user mode.

Line 48: +* Close console to disassociate "init"
Comment describing line 49.

Line 49: +o -
Close the console.

Line 50: +* Run single-user shell program
Comment describing line 51.

Line 51: shell
Execute an interactive shell program. When this shell program terminates, "init" proceeds to line 52.

Line 53: Enter multi-user mode
Comment describing lines 55-67.

Line 55: :multi_user
Assign the label "multi_user" to line 55.

Line 56: +* Run system "startup" script
Comment describing line 57.

Line 57: shell /etc/startup
Use the shell program to execute the file "/etc/startup".

Line 58: +f -> single_user
If the shell program fails to execute the system startup script, branch to the line labeled "single_user" (line 34). You will know that the shell program failed if, on terminating the single-user shell program, you receive only a system prompt, "++", rather than a log-in banner.

Line 59: +* Update the history file
Comment describing line 60.

Line 60: +h mu
Put the entry "mu" (system entered multi-user mode), followed by a time stamp, into the file "/act/history".

Line 62: +* Modify exit conditions for multi-user mode
Comment describing lines 64-65.

Line 64: +01 -> hang_up
On receipt of a HANGUP signal (signal number 1) branch to the line labeled "hang_up" (line 72).

Line 65: +04 -> no_multi
On receipt of a SIGEMT signal (signal number 4) branch to the line labeled "no_multi" (line 79).

Line 66: +* Enter multi-user mode
Comment describing line 67.

Line 67: +m
Enter multi-user mode.

Line 68: -> single_user
Branch to the line labeled "single_user". If the file "/etc/ttylist" does not exist, or if "init" cannot read the file, it cannot enter multi-user mode and, instead, executes this line of code.

Line 70: +* Exit multi-user mode
Comment describing lines 70-77.

Line 72: :hang_up
Assign the label "hang_up" to line 72.

Line 73: +p Issuing HANGUP, expect 15 second delay
Display the message, "Issuing HANGUP, expect 15 second delay", followed by a carriage return and a line-feed character, on the console.

Line 74: +* Send HANGUP signal to all tasks
Comment describing line 75.

Line 75: +k 01
  Send a HANGUP signal to all tasks running on the system.

Line 76: +* Wait for them to terminate cleanly
  Comment describing line 77.

Line 77: +w 15
  Pause for 15 seconds to give all tasks a chance to catch and handle the HANGUP signal.

Line 79: :no_multi
  Assign the label "no_multi" to line 79.

Line 80: +* Kill any outstanding tasks
  Comment describing line 81.

Line 81: +k
  Send a SIGKILL signal to all tasks running on the system.

Line 82: +* Return to single-user mode.
  Comment describing line 83.

Line 83: -> single_user
  Branch to the line labeled "single_user" (line 34).

Lines 85: +* Begin system shut down -- soft entry
  Comment describing lines 88-93.

Line 88: :shut_soft
  Assign the label "shut_soft" to line 88.

Line 89: +p Issuing HANGUP, expect 15 second delay
  Display the message, "Issuing HANGUP, expect 15 second delay", followed by a carriage return and a line-feed character, on the console.

Line 91: +k 01
  Send a HANGUP signal to all tasks running on the system.

Line 93: +w 15
  Pause for 15 seconds to give all tasks a chance to catch and handle the HANGUP signal.

Line 95: +* Shut down system
  Comment describing lines 97-101.

Line 97: :shut_down
  Assign the label "shut_down" to line 97.

Line 98: +* Disable automatic-update feature
  Comment describing line 99.

Line 99: +u 0
  Disable the automatic-update feature.

Line 100: +* Update the history file
  Comment describing line 101.

Line 101: +h st
  Put the entry "st" (system stopped), followed by a time stamp, into the file "/act/history".

Line 103: +* Stop_system
  Comment describing lines 105-109.

Line 105: :stop_system
  Assign the label "stop_system" to line 105.

Line 106: +* Kill all tasks
  Comment describing line 107.

Line 107: +k
  Send a SIGKILL signal to all tasks running on the system.

Line 109: +s
  Shut the system down.

Line 111: +* Run "diskrepair" to clean root device
  Comment describing lines 113-118.

Line 113: :diskrepair
  Assign the label "diskrepair" to line 113.

Line 115: +p
  Send the null string, followed by a carriage return and a line-feed character, to the console.

Line 116: +p Root device may be corrupted -- running "diskrepair"
  Send the message "Root device may be corrupted - running "diskrepair", followed by a carriage return and a line-feed character, to the console.

Line 117: +p
  Send the null string, followed by a carriage return and a line-feed character, to the console.

Line 118: /etc/diskrepair /dev/DISK +vp
  Execute the "diskrepair" command on the root device. Use verbose mode, and prompt for permission to make any repairs (repairs may result in the loss of data from the disk--see Chapter 7).

Line 119: +f -> stop_system
  If the "diskrepair" command fails, branch to the line labeled "stop_system" (line 105).

Line 120: -> boot
  Branch to the line labeled "boot" (line 18).

Appendix B

Program Interrupts

Table B-1 contains the name, number, and a brief description of each of program interrupt. The table also notes various attributes of each interrupt. The file "/lib/sysints" defines these interrupts.

If not caught or ignored, the default behavior of each program interrupt (except SIGDEAD and SIGDUMP) is to terminate the task to which it is sent. As shown in the table, some also produce a "core dump". A core dump is a file called "core" in the working directory which contains the task's image of the contents of memory. Each byte in the program and stack space is written to the core file immediately after receipt of the interrupt. The user can examine this file to determine the state of memory at the time the interrupt was received. A core file is often useful for diagnostic pruposes. The operating system will not create a core file if the working directory contains a file named "core" which denies write permission to the current effective user or if the working directory denies write permission to the current effective user.

The default action for the SIGDUMP interrupt is to create a core dump and return control to the task. The task is not terminated.

A vendor may use a TRAP instruction with a number greater than 6. In such a case the user should not issue the instruction.

User-defined interrupts are available to the end user.

For further information on program interrupts see Section 6.4 of the 68xxx UniFLEX Programmer's Guide.

Table B-1. Table of Program Interrupts

Name	Number	Description	A	C	D	I	R
SIGHUP	1	Hangup	+	+	-	+	+
SIGINT	2	Keyboard	+	+	-	+	+
SIGQUIT	3	Quit	+	+	+	+	+
SIGEMT	4	A-line (Axxx) emulation trap	+	+	+	+	+
SIGKILL	5	Task kill	+	-	-	-	+
SIGPIPE	6	Broken pipe	+	+	-	+	+
SIGSWAP	7	Swap error	+	+	-	-	+
SIGTRACE	8	Trace	+	+	-	+	-
SIGTIME	9	Time limit	+	+	+	-	+
SIGALRM	10	Alarm	+	+	-	+	+
SIGTERM	11	Task terminate	+	+	-	+	+
SIGTRAPV	12	TRAPV instruction	+	+	+	+	+
SIGCHK	13	CHK instruction	+	+	+	+	+
SIGEMT2	14	F-line (Fxxx) emulation trap	+	+	+	+	+
SIGTRAP1	15	TRAP #1 instruction	+	+	+	+	+
SIGTRAP2	16	TRAP #2 instruction	+	+	+	+	+
SIGTRAP3	17	TRAP #3 instruction	+	+	+	+	+
SIGTRAP4	18	TRAP #4 instruction	+	+	+	+	+
SIGTRAP5	19	TRAP #5 instruction	+	+	+	+	+
SIGTRAP6	20	TRAP #6-14 instruction	+	+	+	+	+
SIGPAR	21	Parity error	+	+	+	-	+
SIGILL	22	Illegal instruction	+	+	+	-	+
SIGDIV	23	Division by 0	+	+	+	+	+
SIGPRIV	24	Privileged instruction	+	+	+	-	+
SIGADDR	25	Address error	+	+	+	-	+
SIGDEAD	26	A child task terminated	-	+	-	+	+
SIGWRIT	27	Write to read-only memory	+	+	+	-	+
SIGEXEC	28	Data or stack space violation	+	+	+	-	+
SIGBND	29	Segmentation violation	+	+	+	-	+
SIGUSR1	30	User-defined interrupt #1	+	+	-	+	+
SIGUSR2	31	User-defined interrupt #2	+	+	-	+	+
SIGUSR3	32	User-defined interrupt #3	+	+	-	+	+
SIGABORT	33	Program abort	+	-	-	-	+
SIGSPLR	34	Spooler signal	+	+	-	+	+
SIGINPUT	35	Input is ready	+	+	-	+	+
SIGDUMP	36	Take memory dump	0	+	+	+	+
	37-41	System-defined interrupts					
SIGUNORDERED	42	MC68881 branch or set on unordered operand	+	+	-	+	+
SIGINEXACT	43	MC68881 inexact result	+	+	-	+	+
SIGFPDIVIDE	44	MC68881 division by 0	+	+	-	+	+
SIGUNDERFLOW	45	MC68881 underflow	+	+	-	+	+
SIGOPERAND	46	MC68881 invalid operand	+	+	-	+	+
SIGOVERFLOW	47	MC68881 overflow	+	+	-	+	+
SIGSNAN	48	MC68881 signaling not-a-number	+	+	-	+	+
	49-63	Vendor-defined interrupts					

Notes: A = Default state is "abort" (otherwise, "ignore")
C = Interrupt can be caught
D = Produces a core dump
I = Interrupt can be ignored
R = Resets to default state when triggered
0 = See text

Accounting directory, 7.2
Accounting information, storing and retrieving, 7.9
Accounting software, 7.2
/act. See Accounting directory
/act/history. See History file
Active tasks
  classification of, 9.18-19
  definition of, 9.14, 9.17
  killed by "diskrepair" command, 10.43
  maximum number per user. See Tune command, adjustable parameters, user_tasks
  maximum supported. See Tune command, adjustable parameters, tasks
  priority of, reevaluating, 9.18
  suspended by "diskrepair" command, 10.11
  table of, 9.14
/act/utmp, 7.2
Adding a user, 5.1, 5.4-5
Addpath command, 3.13
Addresses, invalid, 10.1
Addusr command, 5.5
Alterpage command, 1.3
ANSI standard terminal, 7.5
Arrow, in init-control file, 2.2
Asterisk, in format-control file, 7.6
At daemon, 7.8
At directory, 7.8
At symbol, with caret to indicate null character in invalid file name, 10.24
Atexecute command, 7.8
Automatic-boot mode
  aborting from, 1.1
  disabling, 1.4
  enabling, 1.4
  entering, 1.6

Background tasks, supported by shell program, 9.16
Backup command, 3.9, 4.7, 7.6, 10.6, 10.10, 11.9, 11.10
  handling I/O error, 11.9-10
  with 'P' option, 11.9
Backup device, 10.10, 11.9
Backup files
  creating selectively, 11.9
  maintaining, 3.8-9
Bad block
  checking the disk for during formatting, 1.3
  definition of, 3.6
  locating, 3.7
  sequestering, 3.6-8
  specifying location to "formatw" command, 1.3-4
/.badblocks. See Bad-blocks file.
Badblocks command, 3.7-8, 7.1, 10.1, 10.17, 10.38
  called by "diskrepair", 10.5
  function of, 3.8
Bad-blocks file, 3.6-8, 7.1, 10.1, 10.5, 10.17-18, 10.37, 10.40
  duplicate block in, 10.28
  function of, 7.1
  I/O error in, 10.17
  location of, 7.1
  maximum size of, 10.18
  out-of-range blocks in, 10.27
  read by "fdncheck" command, 10.22
  validating, 10.17
Banner message, from operating system, 1.2-3, 11.8
Banner page, from printer spooler, 4.3
Baud rate
  of console, 1.1
  specifying, 7.4
  table of codes, 7.4
Bias, 9.19
/bin, 3.13, 4.2, 6.2, 7.2
Binary file
  duplicate blocks in, 11.10
  error in file size, 10.26
  out-of-range block in, 11.11-12
/bin/shell. See Shell program
Bit bucket, 7.3
Bit map, 10.36, 10.37
Block device
  definition of, 6.3
  with "diskrepair" command, 10.9

Block usage, report on, 10.5, 10.8
Blockcheck command, 10.1, 10.5, 10.6, 10.14
  abnormal termination of, 10.13
  called by "diskrepair" command, 10.12
Blocks
  allocation of, 10.1
  checking allocated, 10.18-20
  duplicated. See Duplicate block
  free, count of, 10.41
  inaccessible
    in contiguous-file space, 10.40
    in non-contiguous-file space, 10.15-16
  in-core list of, 10.37
  missing. See Missing block
  number unused
    in contiguous-file space, 10.8
    in volume space, 10.8
  number used
    in contiguous-file space, 10.8
    in volume space, 10.8
  out-of-range. See Out-of-range block
  size of, 6.1
  unassigned, 10.16, 10.17
Boot mode, automatic. See Automatic-boot mode
Boot mode, manual. See Manual-boot mode
Boot procedure, end of, 1.9
Boot program, 1.5
Boot sector, 3.5, 10.17. See also Sector zero
Booting an operating system, 1.5
  default location on 3.4, 3.9
  indication of success, 11.8
Breakpoint, 1.8
Bss segment
  address of, 1.2, 1.7
  size of, 1.2, 1.7
Buffer cache, 9.5, 9.8-9
Buffers, for I/O, 9.9
Built-in clock, 3.2-4
  bypassing, 3.3-4
Built-in clock (cont.)
  setting, 3.3-4
  using, 3.3

Cache, of file descriptor nodes, 9.10-11
Caret
  with "at" symbol in invalid file name, 10.24
  to create pipe, 6.2
  in invalid file name, 10.24
Carriage return, 2.5, 7.4
Character device, 6.3
Child task, 6.2
Clock. See Built-in clock
Colon, in init-control file, 2.2
Communication between tasks, 6.2, 6.4, 9.5. See also Intertask communication
Console, 2.5, 3.1, 4.2
  baud rate assumed by the ROM, 1.1
  closing from init-control file, 2.4
  configuration assumed by the ROM, 1.1
  default device, 3.2
  definition of, 1.1
  opening from init-control file, 2.4
  opened as standard I/O channels, 2.2
Contiguous file
  creating, 6.1
  definition of, 6.1
  restriction on size, 10.19
Contiguous-file free-list. See Free list, for contiguous-file space
Contiguous-file space, 10.2, 10.19, 10.40
  first block of, 10.16
  fragmentation of, 10.40
  inaccessible blocks in, 10.40
  specifying, 6.1
Control character, 1.6
  in file name, 10.25
  typing, 1.1
Control commands, 2.2, 2.3
  branch
    on abnormal termination, 2.3

Control commands, branch (cont.)
if disk is "dirty", 2.3
on signal, 2.5
close console, 2.4
comment, 2.3
enter multi-user mode, 2.4
log in in single-user mode, 2.4
make entry in history file, 2.3-4
open console, 2.4
pause, 2.6
print a string, 2.5
trace, 2.5
update file system, 2.6
Control sequence. See Control character
Control-A, 1.2
Control-C, 1.1, 1.6, 10.11
Control_pty system call, 7.10
Copy command, 3.9, 7.2
Core dump, definition of, B.1
CPU. See also Max CPU utilization; CPU utilization increment per hit; CPU utilization decay; CPU utilization
regulating a task's time in, 9.6, 9.15, 9.18, 9.20, 9.21
relinquishing to a task of equal priority, 9.20
time allowed in per user task. See Tune command, adjustable parameters, time_limit
CPU utilization, 9.6, 9.19, 9.20
decrementing, 9.20
definition of, 9.19
effect on priority, 9.19
maximum value of, 9.6, 9.20
rate of decrease of, 9.6
rate of increase of, 9.6
CPU utilization decay, 9.6, 9.8, 9.20
CPU utilization increment per hit, 9.6, 9.8, 9.20
CPU-intensive task, definition of, 9.18
Crdir command, 6.2
Crdisk command, 1.3, 11.1, 11.13
creating "/lost+found", 10.31, 10.33
Create command, 6.1
Create_contiguous system call, 6.1
Create_pty system call, 6.4
Crpipe system call, 6.2
Crt_termcap command, 7.4, 7.5
Currently executing task, 9.18
Cylinder, 3.6

Data, salvaging from system disk, 11.9-10
Data segment
address of, 1.2, 1.7, 7.11
size of, 1.2, 1.7
Date
determining, 3.3
setting, 2.2, 3.2-4
Date command, 3.3, 9.15
bypassing built-in clock, 3.3-4
making entries in the history file, 3.12
with built-in clock, 3.3
without built-in clock, 3.3-4
Daylight savings time, 3.4, 9.5, 9.15
first possible day, 9.5, 9.10
last possible day, 9.5, 9.9
time of day of switch, 9.5, 9.10
Daylight-savings-time flag, 9.9
Deleting a user, 5.5-6
Delusr command, 5.6
/dev. See Device directory
Devcheck command, 3.7-8
/dev/console. See also Console, 2.2, 2.3
Device, 6.1, 6.2
adding, 7.3
backup. See Backup device
created by operating system, 6.4
created by system manager, 6.4
definition of, 4.1, 6.2
kinds of, 6.2, 6.3
logical, location of, 7.2
mounted, number allowed, 9.11
physical, location of, 7.2
selecting for printer spooler, 4.1-2
spare. See Backup device

Device (cont.)
standard with a particular system, 6.3
Device directory, 6.4, 7.2, 11.11
Device driver, 4.1
Device number, major, 4.1, 6.3, 6.4
Device number, minor, 4.1, 6.3, 6.4
/dev/tty00, 2.2, 3.2. See also Console
Dir command, 4.6
Directory,, 6.1, 10.2
contents of, 10.23
creating, 6.2
definition of, 6.2
entry in, 10.3, 10.23
nesting, limit for "diskrepair", 10.24
ownership of, 11.14
parent, 6.2, 10.29
purpose of, 6.2
scanned by "diskrepair" command, 10.23
size of, 10.19, 10.23-24, 11.13
unreferenced, 10.29, 10.31-32
Disassembling instructions, 1.8
DISK, 7.3
Diskrepair command, 2.3, 2.6, 3.8, 3.10, 7.11, 10.1-48
for assessing damage, 11.8
called by "badblocks" command, 3.8
checking file size of contiguous files, 10.19
checking file size of noncontiguous files, 10.19
checking free lists, 10.35-40
for contiguous-file space, 10.38-40
for volume space, 10.36-38
checking links, 10.32-35
checking system information record, 10.41-43
checking unreferenced directories, 10.31-32
file size error, 10.25-26
having to rerun, 10.20, 10.26, 10.27, 10.42
inactivating fdns, 10.28
Diskrepair command (cont.)
index of error messages, 10.46-48
I/O errors
handling, 10.1, 10.38, 10.40, 10.43
returned by, 10.3-4, 10.44-46
I/O redirection during, 10.13
limit on errors in file size, 10.26
limitations of, 10.4
list of out-of-range blocks, 10.20
location of, 7.3
maximum size of directory, 10.23
maximum size of disk, 10.14
nesting directories, limit on, 10.24
with options
'a', 10.1, 10.4, 10.5-6, 10.38, 10.42, 10.43
'b', 10.6
'B', 10.6, 10.10
brief descriptions of, 10.5
checking validity of, 10.9
conflicting, 10.9
'f', 10.6
'm', 10.6
'M', 10.7
'n', 10.7, 10.10, 10.11, 10.26, 10.28, 11.8
'p', 10.5, 10.7, 10.8, 10.26, 10.28, 11.10
'q', 10.7
'r', 10.8
'u', 10.8
'v', 10.5, 10.8, 11.8, 11.10, 10.42
without options, 10.5
phase 1, 10.18-20, 10.21, 10.25, 10.26, 10.28, 10.36, 11.8, 11.9
phase 1B, 10.20
phase 2, 10.21, 10.23-31, 10.25, 10.26, 10.28, 10.32
phase 3, 10.31-32
phase 4, 10.32-35
phase 5, 10.35-40
phase 5B, 10.38

Diskrepair command (cont.)
phase 6, 10.41-43
phases
messages concerning, 11.9
synopsis of, 10.1
permissions required, 10.10
preventing deletion of data, 11.10-11
prompting for permission to repair, 10.5, 10.7
read-only mode, 10.7
rebuilding free list for contiguous-file space, 10.40
renaming a file with invalid name, 10.25
rewriting system information record, 10.41, 10.43
simple mode, 10.8
specifying device to, 10.9
suspending tasks, 10.11
syntax, 10.4
transition, phase 1 to phase 2, 10.21-23
unmounting a disk, 10.10
unreferenced directory, handling of, 10.31
unreferenced file, handling of, 10.33-34
updating the system information record, 10.43
verbose mode, 10.5, 10.8
warnings from, 10.7
Disk
accessing a nonsystem disk, 3.9-10
block usage, report on, 10.8
causes of damage to, 3.8
checking integrity of, 2.2
formatting, 3.5-6, 6.1, 10.2
irreparable damage to, 10.22, 10.23
locating I/O errors on, 3.7
logical structure of,, 10.1, 10.2-10.3
checking without changing, 10.7
logical verification of, 3.8, 10.1-48
maximum size for "diskrepair", 10.14
physical damage to, 3.6-8
physical errors on, 10.3-4
Disk (cont.)
repairing. See Diskrepair command
state of, 10.42
updating, 7.6
Disk-intensive task, definition of, 9.18
Dollar sign, 9.12, 9.17
Duplicate block, 10.1
in bad-blocks file, 10.28
in fdns, 10.20
in files, 10.28, 11.10
fixing, 10.20
in free list for contiguous-file space, 10.38, 10.39
in free list for volume space, 10.36, 10.37
length of list of, 10.20
in root directory, 10.28

Edit command, 6.1
Editor, 9.15
End-of-file condition, 7.3
Env command, 7.5
Environment, hardware-specific, 7.11
Error
correspondence between name and number, 7.9
fatal to operating system, 8.1-4
during initialization, 8.1-2
after loading UniFLEX, 8.2-4
Error file, 7.7
Error messages
from "diskrepair" command, index of, 10.46-48
from operating system
binary listing, used by "rel20", 7.8
binary listing, used by "rel68k", 7.8
listing of, 7.7
from "tune" command, 9.22-23
Error numbers, 7.7
Escape sequence, typing, 1.2, 1.6
Escape-B, 1.2, 1.4
/etc, 7.3-7, 10.1

/etc/badblocks. See Badblocks command
/etc/blockcheck. See Blockcheck command
/etc/format.control. See Format-control file
/etc/.init.control. See Init-control file
/etc/log, 7.7
/etc/login, 11.11
/etc/log/motd. See Message of the day
/etc/log/password. See Password file
/etc/prcon, links to, 4.7
/etc/print, 4.2, 4.7
/etc/startup. See Start-up file
/etc/termcap. See Termcap file
/etc/ttycap. See Ttycap file
/etc/ttylist. See Ttylist file
Exclamation point
  in init-control file, 2.2
  in text editor, 3.5
Executable file, loading into memory, 1.6-7
Executable task, 9.18
External reference, resolving, 7.8
Extraneous data
  in file, 10.26
  in free list for contiguous-file space, 10.39

Fcntl system call, 7.9
Fdn cache, 9.10-11
Fdncheck command, 10.1, 10.5, 10.6, 10.21-23
  abnormal termination of, 10.21
  reading bad-blocks file, 10.22
  reading root directory, 10.22
Fdn. See File descriptor node
File, 6.1-4. See also Files
File descriptor node, 6.1, 10.2
  contents of, 6.1, 10.3
  of a device, 6.2
  free, count of, 10.41
  inactive, 10.27, 10.28, 10.30
  initializing, 3.5
  in-core list of, 10.35
  location of, 10.23
File descriptor node (cont.)
  maximum allowed by the operating system, 10.15
  number unused, 10.8
  number used, 10.8
  out-of-range, 10.30
  read by "diskrepair" command, 10.18
  reserving space for, 10.2, 10.30
  of root directory, 10.12
  of standard error, 10.13
  of standard output, 10.13
  table of, 10.32
  of unreferenced directory, 10.31
  of ".", 10.29
  of "..", 10.29
File name
  changed by "diskrepair" command, 10.25
  invalid, 10.24-25
  location of, 10.23
  size of, 1.7, 10.25
File size, error in, 10.25-26
File system
  establishing, 1.3, 3.5
  updating from init-control file, 2.6
File type, 10.29-30
Files
  contiguous. See Contiguous file
  definition of, 6.1
  directory. See Directory
  intermingling of, 4.3
  with I/O errors, recovering, 11.11-12
  kinds of, 6.1
  locked. See Tune command, adjustable parameters, locked_recs
  maximum number open at once. See Tune command, adjustable parameters, files
  number of each type in system, 10.8
  number open on system, 9.10-11
  number open per task, 9.11
  number (total) in system, 10.8

Files (cont.)
ordinary. See Regular file
ownership of, 11.14
pipe. See Pipe
regular. See Regular file
restoring, 11.10
special. See Device
unreferenced, 7.11, 10.1, 10.30, 10.33-34
Fine-tuning the UniFLEX Operating System, 9.1-24
First task, 2.1
Flags. See Init-control file, flags; Tune command, adjustable parameters, DST; Permission flag; Mount flag
Floppy disk drive, seek rate for. See Tune command, adjustable parameters, seek_rate
Format command, 3.5-6, 7.6, 10.2, 10.17, 10.30
using with bad blocks, 3.7
Formatting a disk, 1.3-4, 3.5-6, 6.1
Formatw command, 1.3, 1.4
Format-control file, 7.6
FPU_exception system call, 7.10
Fragmentation
of contiguous-file space, 10.40
within free lists, 10.8
Free command, location of, 7.12
Free list, 6.1, 10.20, 10.28
for contiguous-file space, 10.2, 10.5, 10.6, 10.17
checking, 10.38-40
duplicate blocks in, 10.38, 10.39
extraneous data in, 10.39
fragmentation in, 10.8
map of, 10.39
missing blocks in, 10.38, 10.39
out of order, 10.39
out-of-range blocks in, 10.38, 10.39
rebuilding, 10.5, 10.6, 10.8, 10.40
summary of status, 10.39
for volume space, 10.2, 10.5, 10.6, 10.17
Free list, volume space (cont.)
duplicate blocks in, 10.36, 10.37
fragmentation in, 10.8
I/O error during rebuilding, 10.38
missing blocks in, 10.37, 10.38
omitting block from, 10.38
out-of-range blocks in, 10.36, 10.37
out-of-range pointers in, 10.36
rebuilding, 10.5, 10.6, 10.8, 10.37-38
summary of status, 10.37
validated by "diskrepair" command, 10.36-38

/gen, 7.7
/gen/errors. See Error file
/gen/help. See Help directory
/gen/spooler. See Spooler directory
/gen/spooler/at. See At directory
Ghost shell program, 2.3
Greater-than sign, in init-control file, 2.2

Hang-up interrupt, 3.14, 3.15
Hard disk, 1.3. See also System disk
Hardware
initializing, 1.9
page size, 7.11
Head command, 11.12
Help directory, 7.7
High-order bit, set in file name, 10.24, 10.25
History command, 2.4, 3.12, 7.12
History file, 3.12-13, 7.2
altering permissions, 3.12
creating, 3.12
entries made by "init" program, 2.4
extracting information from, 3.12
making entry from init-control file, 2.3-4
Holidays file, 7.8
Home directory, 5.1, 5.3, 5.4,

Home directory (cont.)
7.7
assigning, 5.3
creating, 5.3
definition of, 5.3
location of, 7.12
owner of, 5.3
Hyphen, in init-control file, 2.2

Info command, 3.2, 7.1, 7.12
Init program, 2.1, 3.6, 7.4, 7.6, 11.11
error messages from, 2.6-9
fatal, 2.7-8
nonfatal, 2.8-9
making entries in history file, 2.4
signals accepted by, 2.5
Initializations, 1.9
Initializing hardware, 1.9
Init-control file, 2.1-9, 3.2, 3.14, 7.11
altering, 7.6
contents of, 2.1
essential elements of, 2.6
example, A.1-7
executing shell command from, 2.2
flags, 2.2-3
assign a label, 2.2
branch to a label, 2.2
exclamation point, 2.2. See also Shell command, executing from init-control file
execute control command, 2.2. See also Control commands
execute shell command, 2.2. See also Shell command, executing from init-control file
plus sign, 2.2. See also Control commands
length of file, 2.1
length of line in, 2.1
purpose of, 7.6
standard, 3.1, 3.12, 3.13, 3.14
when executed, 2.1
Insp command, 4.3
Install command, 1.10
Instruction, user, 9.18
Int command, 3.15
Interrupts, B.1-4
correspondence between name and number, 7.9
sent by operating system, 10.13, 10.21
table of, B.1
where defined, 7.9
Intertask communication, 9.11-12
I/O characters, lists of. See Tune command, adjustable parameters, iolists
I/O error, 10.3
in badblocks file, 10.17
cause of, 3.6
from "diskrepair", 10.4, 10.43-46
handling by "diskrepair" command, 10.1, 10.3-4, 10.17
location of, 3.7, 11.9-10
in map of contiguous-file space, 10.40
recovering files containing, 11.11-12
salvaging file, 11.11-12
while rebuilding free list for volume space, 10.38
while updating system information record, 10.43
I/O redirection, 6.1, 6.2, 10.13

Kernel, 1.5, 7.1, 11.9
Kill command, 7.2

Leap year, 9.9, 9.10
/lib, 7.8-11
/lib/rel20.errs, 7.8
/lib/rel68k.errs, 7.8
/lib/std_env. See Standard-environment file
/lib/sysacct, 7.9
/lib/sysdef, 7.9
/lib/syserrors, 7.9
/lib/sysfcntl, 7.9
/lib/sysints, 7.9, B.1
/lib/Syslib68k, 7.8
/lib/sysmessages, 7.9
/lib/syspty, 7.10
/lib/sysrump, 7.10
/lib/sysstat, 7.10

/lib/systime, 7.10
/lib/systty, 7.10
/lib/sys68881, 7.10
Line-feed character, 2.5
Link command, 11.13
Link count, 10.32, 10.34
   from "dir" command, 4.7
Linked files, out-of-range blocks in, 10.27
Linking-loader, 7.8, 7.11
Links
   checked by "diskrepair" command, 10.32-35
   definition of, 10.3
Load68k command. See Linking-loader
Log command, 3.2, 3.13
Logging in, in single-user mode, 2.4
Login command, 3.12, 7.3
Login program, 3.6, 5.1, 5.4, 7.7
   assigning, 5.4
   default, 5.4
   definition of, 5.4
Login task, starting, 2.4
Long listing, of directory entry, 6.4
Lost-and-found directory, 7.11, 10.29, 10.31
   creating
      on damaged disk, 10.32, 10.34
      normally, 10.31, 10.33
   expanding, 11.13-14
   full, 10.31, 10.32, 10.33, 10.34
   nonexistent, 10.31, 10.33
/lost+found. See Lost-and-found directory
Lost+found directory. See Lost-and-found directory
lrec system call, 9.11
Ls command, location of, 7.2

Major device number, 9.12-13
Makdev command, 4.1, 6.4, 7.3
Manual-boot mode, 1.6
Master floppy, 11.8
Master task, 6.4
Max CPU utilization, 9.6, 9.8, 9.20
Max quantum, 9.6, 9.8, 9.21
Medium, logical verification of, 3.8, 10.1-48
Memory
   accessing from ROM, 1.6, 1.7, 1.8, 1.9
   allocation of for tables, 1.9
   amount available to users, 1.3
   dumping, 1.7, 1.8
   modifying, 1.9
   physical, 7.3
   system, 7.3
Memory dump, 1.7, 1.8, 3.11-12
Memory management unit, 9.17
Memory map, displaying, 1.9
Message, sending to all users, 3.4, 7.7
Message buffers. See Tune command, adjustable parameters, msg_buffers
Message of the day, 3.4-5, 7.7
Message exchanges. See Tune command, adjustable parameters, msg_exchanges
Message size. See Tune command, adjustable parameters, msg_size
Minor device number, 9.12-13
Minus sign
   in init-control file, 2.3, 2.4, 2.5
   with "shutup" command, 3.14
   in standard environment file, 7.11
   with "stop" command, 3.15
   in ttylist file, 7.4
Missing block, 10.1, 10.5, 10.6, 10.36
   in contiguous-file space, 10.38, 10.39, 10.40
   handling by "diskrepair" command, 10.6
   in volume space, 10.37, 10.38
MMU. See Memory management unit
Mode of operation of operating system, 7.6. See also Multi-user mode; Single-user mode
   changing from multi-user to single-user, 3.14-15
   changing from single-user to multi-user, 3.2

Mode of operation (cont.)
determining, 7.2
Model code for "formatw" command, 1.4
Modem, 7.4, 7.5
Modules supported by the system, xi, 7.1
Mount command, 3.9-10, 11.9, 11.11
Mount flag
checking, 10.7, 10.41-42
clearing
by "diskrepair" command, 3.10, 10.5, 10.7, 10.42, 11.9
by "unmount" command, 3.9, 10.41
setting, 3.9
Mounting a device, 3.9-3.10
maximum number. See Tune command, adjustable parameters, mounts
nodes for, 7.12
for reading only, 3.10, 11.11
.Mrk*splr? file, 4.4, 4.5
Msg_status system call, 7.9
Multi-user mode, 3.1, 7.6
booting directly to, 3.2
entering
from init-control file, 2.4, 3.2
from single-user mode, 3.2, 3.13

Network device, function of, 6.4
New programs, adding to the system, 3.13
New_system command, 11.9
Nice command, 9.19
Null character, 10.24, 10.25
Null device, 7.3

Ofstat system call, 7.10
Operating system
booting, 1.5
building, 1.1-5
from tape, 1.3, 1.4
copying to a hard disk, 1.1-5
kernel, 1.5, 7.1, 11.9
loading into memory, 1.2
making entries in the history file, 3.12
Operating system (cont.)
medium supplied on, 1.1
modes of operation, 3.1-2
optimizing performance, 9.1
prompt from, 1.3
release date, 1.3
setting date for, 3.3-4
shut down procedure, 2.5, 3.14-15
size of, 1.3
starting execution of, 1.2
tuning, 9.1
version number, 1.3
Ordered list, 10.39
Output, redirecting, 6.1, 6.2
Out-of-range block
in bad-blocks file, 10.27
in fdns, 10.19-20
in files, 10.26-28, 11.9-10, 11.11-12
fixing, 10.19
in free list for contiguous-file space, 10.38, 10.39
in free list for volume space, 10.36, 10.37
in linked files, 10.27
list maintained by "diskrepair" command, 10.20
recovering files containing, 11.11-12
in root directory, 10.27
salvaging file, 11.11-12
Out-of-range file descriptor nodes, 10.30
Out-of-range pointers, in free list for volume space, 10.36

Page size, hardware, 7.11
Paging device. See also Tune command, adjustable parameters, page_dev
changing identity of, 7.3, 9.12-13
defining, 9.12-13
definition of, 7.3
identifying, 7.1
link to, 7.3
setting on system disk, 11.9
Paging space, 10.2. See also Tune command, adjustable parameters, page_space

Paging space (cont.)
changing size of, 1.3
default, 9.13
definition of, 10.2
determining amount, 3.5
first block of, 10.15
overlapping contiguous-file space, 10.16
required, 10.2
reserving, 1.3, 3.5
Parent directory, 6.2, 10.29
Parent task, 6.2
Password, 5.1, 5.2-3, 7.7
assigning, 5.2
forgotten, 5.3
guidelines for, 5.2
restrictions on, 5.2
in single-user mode, 3.1, 3.2
Password command, 5.2
Password file, 2.4, 5.1-6
original, 5.4
structure of, 5.1-4
Pause, from init-control file, 2.6
Pending timed-event. See Timed event, pending; Tune command, adjustable parameters, time_outs
Permission flag, 7.10
Permissions
default for a directory, 5.3
s+, 11.14
Perms command, 11.14
Personality of a task. See Active tasks, classification of
Phys parameters, 9.3, 9.7, 9.16-17
Phys segments, 9.2, 9.5
definition of, 9.16
logical address of, 9.6, 9.17
physical address of, 9.6, 9.17
size of, 9.6, 9.17
Physical error. See I/O error
Pipe, 6.1, 6.2
Pipe device. See also Tune command, adjustable parameters, pipe_dev
defining, 9.13
identifying, 7.1
restrictions on, 9.13
setting on system disk, 11.9
Pipe-intensive task, definition of, 9.18
Plus sign, 1.4
in init-control file, 2.1, 2.2, 2.3, 2.4, 2.5
in ttylist file, 7.4
Pmem, 7.3
Print command, 7.3
Printer, 4.2, 6.3
Printer driver, 4.1
Printer spooler, 4.1-8
banner page from, 4.3
changing modification time of spooled file, 4.4
changing the name of, 4.3
configuring, 4.1-4
creating directory for, 4.2-3
creating directory entry for spooled file, 4.4
creating spooler command, 4.2
damage to, 4.6-7
the 'f' option, 4.3
function of, 4.1
initial form-feed character, 4.3
initiating, 4.3
links between files, 4.6-7
naming the spooled file, 4.4
reconstructing, 4.7
removing from the system, 4.7
selecting device for, 4.1-2
shutting down, 4.5-6
summary of use, 4.6
symptoms of damage to, 4.6
using the spooler command, 4.4-5
Printing, a string from init-control file, 2.5
Priority, 9.6, 9.17
determining, 9.19
evaluating, 9.18, 9.20
price paid for using CPU, 9.20
reward to for not using CPU, 9.20
rules for calculating, 9.18
Program counter, 1.6, 1.8
Program space, B.1
Prompt
from operating system, 1.3
from ROM, 1.1

Pseudoterminal device,
definition of, 6.4
Pstop command, 4.3, 4.5

Quantum, 9.6
decrementing value of, 9.20
definition of, 9.20
maximum value of, 9.6, 9.21
rate of increase of, 9.6, 9.21
Quantum increment, 9.6, 9.8, 9.21
Question mark, 1.1, 7.4-5

RAM, 3.6, 6.3
clearing, 3.11
looking at the contents of, 3.11
RAM disk, 6.3, 6.4
Ramdisk command, 6.4
Random access memory. See RAM
Read-only memory. See ROM
Rebuilding a system, 11.9, 11.10
Records, locked, 9.11
Register
displaying contents of, 1.9
setting contents of, 1.9
Regular file, 6.1, 10.2
checking size of, 10.19
definition of, 6.1
Release date, 1.3
Rel20 command, 7.8
Rel68k command, 7.8
Rename command, 10.25
Resetting the computer, 3.14, 3.15
Restoring files, 11.10
ROM, 1.5, 11.8
accessing memory from, 1.6
assumption about console, 1.1
commands supported by, 1.6-9
comparing contents to a file, 1.8
loading a file into, 1.5, 1.6-7
prompt from, 1.1
using to take memory dump, 3.11
Root device. See also Tune command, adjustable parameters, root_dev
changing identity of, 7.3
Root device (cont.)
checking structure of, 10.11
definition of, 7.3
with "diskrepair" command, 10.10
identifying, 7.1
link to, 7.3
setting on system disk, 11.9
specifying, 9.14
updated by "diskrepair" command, 10.43
Root directory, 3.4, 3.9, 5.4, 9.11, 9.14, 10.17
badly damaged, 10.4
checking status of, 10.12
duplicate block in, 10.28
establishing, 3.5
out-of-range block in, 10.27
read by "fdncheck" command, 10.22
Rump system call, 7.10
Run-time libraries, 7.8

Salvaging data
from file with I/O error, 11.11-12
from file with out-of-range block, 11.11-12
from system disk, 11.9-10
Scheduler, 9.17-18
Scheduling parameters, 9.2, 9.3, 9.6, 9.17-21
default values for, 9.7
maximum values for, 9.7
minimum values for, 9.7
Search path, 3.13
Sector
physical boundaries of, 3.5
size of, 3.6
Sector zero, 10.5, 10.6, 10.7, 10.10, 10.17
ignoring, 10.6
writing, 3.5
Seek rate of floppy drives, 9.14. See also Tune command, adjustable parameters, seek_rate
Serial number
of hardware, 1.3
of software, 1.3
Serial printer, attaching to a terminal driver, 4.2

Setpath command, 3.13
Setpr system call, 9.19
Set_termcap command, 1.5, 7.4, 7.5
Shared-text programs, 9.15
  maximum number supported. See Tune command, adjustable parameters, text_segs
Shell command, 2.2-3
  abnormal termination of, 2.3
  executing from init-control file, 2.2
Shell program, 3.6, 5.4, 6.1, 6.2, 9.15, 11.11
  background tasks supported by, 9.16
  ghost, 2.3
  search path of, 3.13
  single-user, 2.7, 3.13
Shell script, 1.3
Shutting down the system, 2.5, 3.14-15
Shutup command, 2.5, 3.14-15, 7.3
Signal, branching on in init-control file, 2.5
Single instruction, executing, 1.8, 1.9
Single-user mode, 3.1, 7.2, 7.6
  booting to, 3.1-2
  bypassing, 3.2
  exiting, 3.2, 3.13, 3.15
  password in, 3.1, 3.2
  when to use, 3.1
SIR. See System information record.
Skipbad command, 11.12
Slash character, in file name, 10.25
Slave task, 6.4
Smem, 7.3
Spooled file, naming, 4.4
Spooler. See Printer spooler
Spooler command. See Printer spooler
Spooler directory, 7.8
  creating, 4.2-3
  entries in, 4.4
Stack space, B.1
Standard error, 2.2, 2.4, 2.6, 10.5, 10.13
Standard input, 2.2, 2.4
Standard I/O channels. See Standard input; Standard output; Standard error
Standard output, 2.2, 2.4, 7.7, 10.13, 11.9
Standard-environment file, 7.11
Start-up file, 4.3, 7.11
  abnormal termination of, 3.13
Status system call, 7.10
Stop command, 2.5, 3.15
Streaming tape, building an operating system from, 1.3, 1.4
Support software
  adding to a system, 1.10
  updating, 1.10
Suspended task, 9.18
Swap, 7.3. See also Paging device; Paging space
System buffers, 9.8-9
System call
  correspondence between name and number, 7.9
  definition of, 9.18
System crash, 11.1
  course of action following, 11.1-11
  effect on printer spooler, 4.5
System disk
  building, 7.1
  copying operating system to, 1.3
  definition of, 11.8
  formatting, 1.3-4, 3.5-6, 6.1
  mounting, 11.9
  salvaging data from, 11.9-10
  setting paging device on, 11.9
  setting pipe device on, 11.9
  setting root device on, 11.9
  updating, 1.9
System floppy, 1.2. See also Master floppy
System information record, 10.17, 10.37
  access by "diskrepair", 10.14
  badly damaged, 10.4
  checked by "diskrepair", 10.41-43
  contents of, 10.2
  preliminary checks by "diskrepair", 10.14

System information record (cont.)
rewritten by "diskrepair" command, 10.15, 10.16, 10.41, 10.43
updated by "diskrepair" command, 10.43
writing, 3.5, 10.2
System-wide programs, home for, 3.13
S+ permission, 11.14

Tail command, 11.12
Tape, 6.3, 11.9, 11.11
Task
abnormal termination of, 9.15
active. See Active tasks
currently executing, 9.18
executable, 9.18
number supported by system, 9.14
suspended, 9.18
Task ID, 3.15, 4.4, 9.14, 9.17
Temporary directory, 7.11
Temporary files
deleting, 7.11
location for, 7.11
Termcap file, 7.5
Terminal, 6.3. See also Terminal port, 7.4
ANSI standard, 7.5
buffering input to, 9.9
buffering output from, 9.9
defining capabilities of, 7.5
maximum number supported, 7.3
as standard I/O channel, 10.14
Terminal driver, 4.1, 4.2
Terminal port
configuration of, 1.5
disabling for login, 7.4
enabling for login, 7.4
number for, 7.4
type of terminal associated with, 7.4-5
Text editor, 9.15
Text file
duplicate blocks in, 11.10
out-of-range block in, 11.11-12
Text segment
address of, 1.2, 1.7, 7.11
size of, 1.2, 1.7
Tick, definition of, 9.6
Time
determining, 3.3
internal storage of, 3.4
setting, 3.2-4
Time limit for tasks, 9.15
Time system call, 7.10
Time zone, 3.4, 9.7, 9.15-16
Timed event, pending, 9.15
Time-outs, number of, 9.15
/tmp. See Temporary directory
Tracing, control commands, 2.5
Track
moving head from one to another, 9.14
number of sectors in, 3.6
Trouble-shooting after a system crash, 11.1-11
Ttime system call, 7.10
Ttycap file, 7.4, 7.5
Ttyget system call, 7.10
Ttylist file
contents of, 7.3-5
format of, 7.4-5
Ttyset system call, 7.10
Tty-intensive task, definition of, 9.18
Tune command, 7.1, 7.3, 9.1-24, 11.9
adjustable parameters, 9.4-21
from automatic mode, 9.8-16
buffers, 9.5, 9.7, 9.8-9
default values of, 9.7
determining current values of, 9.21
DST, 9.5, 9.7, 9.9
DST_end, 9.5, 9.7, 9.9
DST_start, 9.5, 9.7, 9.10
DST_time, 9.5, 9.7, 9.10
files, 9.5, 9.7, 9.10-11
functions of, 9.4
iolists, 9.5, 9.7, 9.9
locked_recs, 9.5, 9.7, 9.11
maximum values, 9.7
minimum values, 9.7
mounts, 9.5, 9.7, 9.11
msg_buffers, 9.5, 9.7, 9.11-12
msg_exchanges, 9.5, 9.7, 9.12

Tune, adjustable parameters (cont.)
msg_size, 9.5, 9.7, 9.12
page_dev, 9.5, 9.7, 9.12-13
page_space, 9.5, 9.7, 9.13
pipe_dev, 9.5, 9.7, 9.13
root_dev, 9.5, 9.7, 9.14
seek_rate, 9.5, 9.7, 9.14
tasks, 9.5, 9.7, 9.14, 9.16
text_segs, 9.5, 9.7, 9.15
time_limit, 9.5, 9.7, 9.15
time_outs, 9.5, 9.7, 9.15
time_zone, 9.5, 9.7, 9.15-16
user_tasks, 9.5, 9.7, 9.14, 9.16
adjusting parameters in interactive mode, 9.3
arguments, 9.1-2
automatic mode, 9.3, 9.8
default values for parameters, 9.6, 9.7
error messages from, 9.22-23
examples, 9.21
function of, 9.1
interactive mode, 9.2, 9.3, 9.5, 9.16, 9.17
limits on values of parameters, 9.7
list of parameters, 9.3
modes of operating, 9.2-3
options, 9.2. See also individual names
'p' option, 9.2, 9.17
'P' option, 9.2, 9.5, 9.16
parameters associated with scheduling. See Scheduling parameters
parameters associated with "phys" segments. See Phys parameters
'q' option, 9.2
quiet mode, 9.2
'r' option, 9.2, 9.3, 9.5
read-only mode, 9.2-3
restrictions imposed on parameters by, 9.3
syntax for, 9.1

/uniflex, 7.1, 7.3, 9.6, 9.21, 11.8
UniFLEX commands, location of, 7.2, 7.12
Unmount command, 3.9, 10.41
Unreferenced files, 7.11, 10.1, 10.30, 10.33-34
Updating support software, 1.10
Updating system disk, 1.9
User ID, 5.1, 5.3, 7.7
assigning, 5.3
for bin, 5.3
range of, 5.3
for system manager, 5.3
User ID bit, 11.14
User instruction, 9.18
User name, 5.1-2
Users, list of, 7.7
/usr, 7.12
/usr/bin, 3.13, 7.12
/usr0, 3.10, 7.12
/usr1, 3.10, 7.12
/usr2, 3.10, 7.12
/usr3, 3.10, 7.12
Utmp file, 7.2

Verification of the medium, logical, 3.8
Version number, 1.3
Vertical bar, to create pipe, 6.2
Volume space, 10.2, 10.19, 10.20
bit map of, 10.36, 10.37
overlapping contiguous-file space, 10.16
overlapping paging space, 10.15

Who command, 7.2

"."
definition of, 6.2
file descriptor node of, 10.29
missing entry, 10.29, 11.12-13
multiple entries, 10.29

".."
definition of, 6.2
file descriptor node of 10.29
missing entry, 10.29, 11.12-13
multiple entries, 10.29