ALTOS
COMPUTER SYSTEMS

ACS 8600 Computer System

Xenix Operating System Supplement

Revision D
October 6, 1982

Part Number 690-11857-001

2360 Bering Drive, San Jose, CA 95131
(408) 946-6700 Telex: 171562 ALTOS SNJ or 470642 ALTO UI

ACS 8600 COMPUTER SYSTEM

XENIX OPERATING SYSTEM

SUPPLEMENT

REVISION D

OCTOBER 6, 1982

Part Number: 690-11857-002

TABLE OF CONTENTS

Section 3. UTILITY REFERENCE SECTION (cont.)

APPENDICES

APPENDICES (cont.)

Section 1.

INTRODUCTION

OVERVIEW

The XENIX* supplement to the User Manual for the ACS 8600 provides two kinds of information. First, it has information about the Altos implementation of XENIX on the ACS 8600. This information consists of both operational and reference material.

Secondly, this supplement has information about other documentation available. XENIX refers to the Microsoft implementation of the UNIX* operating system, which was developed by Bell Laboratories. Information in this document supplements documentation provided by Bell Laboratories, and Microsoft, and other software suppliers.

SUPPLEMENT ORGANIZATION

The Introduction contains general explanatory and background material, and lists documentation furnished to you as part of your XENIX system.

The next section, Operating Procedures, gives information on the following:

1. Installation Procedures

   How to install, or re-install, the XENIX operating system on your ACS 8600 Computer System.

2. Starting Up XENIX

   How to start up the XENIX system for regular use.

---

*XENIX is a trademark of Microsoft, Inc.
UNIX is a trademark of Bell Laboratories.

3. Shutting Down XENIX

   How to shut XENIX down gracefully, making sure that no other users are caught by surprise.

4. Saving and Restoring Files

   How to save and restore files using the tar utility.

The Utility Reference Section gives the following information.

1. Useful XENIX Utilities

   A UNIX Manual (described below under the heading Bell Laboratories Manuals) gives full information about UNIX utilities. This section provides some guidance on the most important utilities. They are described briefly to help guide you through the Bell reference material.

2. UNIX Manual Changes and Additions

   Altos also provides some variations on UNIX utilities and some new utilities from various sources. These variations and additions are described in the standard Bell Laboratories format for documenting utilities. The Altos documentation can be left in this supplement or can be inserted into the Bell reference manual as desired.

The Appendices contain general reference material.

Appendix A. Hard Disk Organization

A brief reference describing how the XENIX system and user files are allocated on the hard disk.

Appendix B: Floppy Diskette Organization

A brief reference describing how files are allocated on floppy diskettes.

Appendix C. Diskette Handling

Information on proper handling and storage of floppy diskettes.

Appendix D. The Serial Line Printer and Spooler

Information regarding the serial line printer default options and how to change them.

Appendix E. The Parallel Line Printer and Spooler

Information regarding the parallel line printer default options and how to change them.

Appendix F. Cartridge Tape Organization

The configuration of the tape drive upgrade for the ACS 8600.

Appendix G. List of Terminal Capabilities

A data base listing special capabilities of all terminals supported by Altos XENIX.

Appendix H. Numeric Formats, C and Fortran 77

Reference information on the internal format used for numerical representation in these languages.

Appendix I. List of XENIX Utilities

A sample list of utilities furnished with your system.

Appendix J. 8086 Assembly Language

A description of the XENIX 8086 Assembly Language.

DOCUMENTATION FURNISHED

The following documentation is furnished with your XENIX system.

Bell Laboratories Manuals

UNIX Programmer's Manual, Seventh Edition. This is a three-volume set.

Volume 1 gives general information about UNIX and about the manual set. It has the reference information on utilities and system calls, organized into sections.

Volume 2A contains supplementary and tutorial information.

Volume 2B contains additional reference material, and includes advanced topics and languages. For example, this includes the C language, and yacc, which is a tool for writing compilers for other languages.

UNIX Reference Card

A thirty-six page concise reference booklet, loosely

bound in order to lie flat.

Other Documentation

Using the UNIX System, by Richard Gauthier.

A commercial book which is a tutorial for the beginning user of the UNIX system.

Addendum One to the XENIX Operating System Supplement

This Altos manual contains reference and tutorial material dealing with UNIX utilities furnished by the University of California, Berkeley. Both the documentation and the utilities are furnished under license by the Regents.

XENIX "SHELLS"

The UNIX operating system in designed in layers. It can be thought of as somewhat like an onion; with the outermost layer being what is visible and touchable, and the inner layers supporting the outer ones. The outermost layer is called, in UNIX, a "shell." This particular implementation of XENIX supports more than one shell. As a user, you can choose the shell you wish to use whenever you start up the system. In fact, you can change the shell at any time. Two shells are of particular interest:

The UNIX Shell

This is the shell explained in the book by Richard Gauthier.

The Business Shell

The Business Shell is a simplified, menu-driven command interpreter, with extensive tutorial and guidance material. It is designed for use in a learning environment or an application environment where the emphasis is on business use of XENIX.

More information on the Business Shell is given in the Utility Reference Section, where utilities pertaining to it are documented. The following utilities in that section relate particularly to the Business Shell: bsh(1), digest(1), menus(1).

The explanation of how to choose the shell you wish to use is given in the Operating Procedures Section, under "Starting Up XENIX." and "Getting Started with XENIX."

ADDING PERIPHERALS

The ACS 8600 can support an additional hard disk, an optional cartidge tape drive, and can use either or both a serial and a parallel printer. Here is some information about these additional peripherals.

Additional Hard Disk

The additional hard disk drive can have a capacity of twenty or forty megabytes. See the Appendix on **Hard Disk Organization** for information. If it is present at the time of installation, it is automatically included in your XENIX system. If you add it later, see the add.hd utility in the Utility Reference Section for information on including it in XENIX.

Optional Tape Drive

The cartridge tape drive can be used for saving and restoring hard disk files. For information, see the Appendix on **Cartridge Tape Organization**, and also "Saving and Restoring Files," in the Operating Procedures Section. If it is present at the time of installation, it is automatically included in your XENIX system. If you add it later, see the add.ct utility in the Utility Reference Section for information on including it in XENIX.

Printers

The standard printer is assumed to be a serial printer. However, a parallel printer can be used, or both a serial and a parallel printer. For information, see the Appendices on **Serial Line Printer and Spooler**, and **Parallel Line Printer and Spooler.**

SCOPE OF THIS REVISION

Revision D of this supplement pertains to XENIX Version 2.2c and later. Version 2.2c differs from XENIX 2.2b in two capabilities; it supports the addition of an optional cartridge tape drive, and has the format utility which allows the formatting of diskettes while running XENIX.

Revision 2.2b introduced a number of new utilities, and some improvements to the tar utility which saves hard disk files to diskette and restores from diskette. The improvements allow handling files without regard for diskette boundaries. Revision 2.2c added the capability of using tar with either magnetic tape or diskettes. See "Saving and Restoring Files" in the Operating Procedures Section.

Corrected System Deficiencies

In prior releases of XENIX certain deficiencies in the hardware or the system software were documented as "Known System Deficiencies." All prior deficiencies are corrected in this release.

Section 2.

OPERATING PROCEDURES

DOCUMENTATION CONVENTIONS

In the documentation of these procedures, all information the user enters is shown in **bold face.** Variable information is shown using these three small letters: n, a, and x. They mean, respectively, any number, any letter, and any character, either letter or number. For example:

#1 of n Version n.na Filename xxxxxx

INSTALLING XENIX

Prior Steps

Before installing the operating system, two tasks are done. The diagnostics must be run, and copies must be made of the installation diskettes. When the diagnostics are run, if any bad sectors are found on the hard disk that have not been flagged, they must be flagged using the procedures described in the Diagnostic Supplement. (ADX Diagnostic Supplement, HARD86 Test.) The XENIX installation procedure automatically handles any bad sectors that have been flagged.

Note:

At the time of this release, the ADX diskette did not have the COPY utility for making copies of diskettes. This utility will be released soon. In the meantime, it may be necessary to use some other means of copying the installation diskettes, or to forgo making copies.

Before starting this procedure, assemble the the copied set

of XENIX installation diskettes. The copies are used for installation while the originals are stored in a safe place. There are several installation diskettes, labeled as follows:

1. XENIX Root File System
2. XENIX Utilities #1 of n
3. XENIX Utilities #2 of n
4. XENIX Utilities ...
5. XENIX Utilities #n of n

You may wish to have a copy of **UNIX Programmer's Manual**, Volume 1, handy during installation. The manual is not necessary, but it has background information on certain utilities used in the installation procedure.

If you have an installed XENIX system, see "Re-Installing XENIX," later in this section, which explains how to preserve current user files while installing the latest release. Unless the re-installation procedures are followed, installing a new release may, in effect, erase all existing files on your hard disk.

System Hardware Checkoff

This is a checkoff list of items concerning the system hardware. If the system has been unpacked and assembled according to the instructions in Section 2 of the **ACS 8600 Computer System User Manual**, items 1 and 2 should be good. Check item 4 even though the system diagnostics have just been run; the diagnostics can be run with the terminal connected to any serial port.

If you have difficulty during the beginning phases of installation, consult this list.

1. Check that the disk heads and motor are unlocked.
2. Check that the fuse is in place.
3. Check that the power cord is connected.
4. Connect a terminal to serial port 1. (RS 232C connection.) If not connected to port 1, XENIX will not work!

The terminal should be configured as follows:

1. ASCII character set
2. 9600 baud asynchronous

3. 8 bits data, no parity, 1 start bit and 1 stop bit

4. The terminal must raise DTR on pin 20 of the RS-232 connection in order to be recognized by the ACS 8600. If the terminal does not, this signal can be achieved by strapping pin 20 to pin 6, which is always raised by the ACS 8600.

For information on configuring your terminal, consult the vendor manual.

If you experience system errors during installation, call Altos Field Engineering.

Procedure for Installing XENIX

Be sure to allow sufficient time to complete this procedure. If you have to abandon it after you have begun working with the XENIX installation diskettes, when you start over you will get a message saying the system was not shut down properly. In such a case, see the notes at the end of this procedure for information on how to respond.

Allow approximately one hour for this procedure the first time you do it. When you have done it before, you can do it more quickly.

If you are not familiar with handling floppy diskettes, the Appendix on Handling Diskettes will be useful. It covers such topics as preventing data loss when working with diskettes and proper storage of diskettes.

If you wish to upgrade an earlier version XENIX already installed, go to "Re-Installing XENIX", which follows this procedure, and save your files. Resume at step 4 of the procedure below.

An error made when entering information can be corrected by using the Backspace, Rubout, or Delete key.

Operator Actions

System Response

1. Turn on power switch on front panel.

2. ALTOS COMPUTER SYSTEMS - 8600
   Monitor Version n.nn
   Press any key to interrupt boot
   (This is the monitor sign-on message)

3. Disk Boot Failed, Code: xx
   *
   . (period)

Operator Actions

System Response

(This appears because the floppy drive is empty.)

If you do not get the monitor sign-on message, go to the System Hardware Checkoff procedure, above, and check those items.

4. Load the diskette labeled "XENIX ROOT FILE SYSTEM" (For information on how to load a diskette, and other useful information on diskettes, see the Appendix on **Diskette Handling.**)

NOTE

The XENIX root file diskette must be writeable, that is, the write-protect notch described under Diskette Handling must not be open; if this diskette is not writeable, the XENIX prompt, #, will not be displayed.

5. Press the Reset button on front panel.

6. ALTOS COMPUTER SYSTEMS - 8600
   Monitor Version n.nn
   Press any key to interrupt boot

7. After a delay of 35-40 seconds, the following message appears.

   Xenix Vn.na

   mem=nnnK (denotes available 'user' memory)

   # (XENIX prompt)

8. Enter **fsck /dev/fd0.root<CR>**

(This validates the consistency of the file system on the floppy disk.)

9. /dev/fd0.root
   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Pathnames
   ** Phase 3 - Check Connectivity
   ** Phase 4 - Check Reference Counts
   ** Phase 5 - Check Free List
   ** nn files nn blocks nnnnn free
   #

Operator Actions

System Response

If Step 9 is unsuccessful, the "XENIX root file system" diskette is defective and must be replaced. Start over at Step 1 with a new diskette.

10. If Step 9 was successful, enter:

make.hd<CR>

(This is a "shell script" to initialize the hard disk.)

11. The system will ask:

What size disk do you have (20 or 40)?

This refers to whether you have a 20-megabyte integral hard disk (-12), or a 40-megabyte integral hard disk (-14).

12. Reply **20** or **40** accordingly.

13. At this time, the operation to initialize the hard disk for XENIX operation begins. You will begin to see messages of the form:

```
Checking block nnn
```

If the procedure finds bad disk sectors that have been flagged, they are automatically located and mapped to an alternate-sector area using the layout(1) and map(1) utilities.

For each bad sector encountered, you will see an informational message similar to this:

```
err on dev 0/0
bn = xxxx  cmd = xxxx  sts = xxxx

Block nnnn is bad
Continuing on block nnnn+1
```

You may also see messages of the form:

```
Spare block nnnn is also bad
```

if there is a bad spot in the alternate-sector area.

Operator Actions

System Response

With the exception of cylinder zero, bad spots may be located anywhere on your hard disk. The preceding messages are informative only and indicate that XENIX is initializing the hard disk correctly.

After 10-15 minutes, when the initialization process is complete, XENIX displays the alternate-sector map in four column format as follows:

Cylinder number/Head number/Sector number mapped to block number in spare-block area.

The following message is then displayed:

```
isize = xxxxx
m/n   = x xx
```

After several minutes, you will see the following output:

```
/dev/hd0b
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
** nn files nn blocks nnnnn free
***** FILE SYSTEM WAS MODIFIED *****
2 + 0 records in
2 + 0 records out
Now boot the hard disk and run 'load.hd' to
get the rest of the utilities.
** Normal System Shutdown **
```

NOTE

If you see any error messages other than those described above, your hard disk has not been initialized correctly.

If you cannot eliminate the problem by repeating the installation process from Step 1, consult your Altos Dealer or call Altos Customer Service.

Operator Actions

System Response

14. When you have reached this step, the "XENIX Root File System" is installed on your hard disk. Remove the "Root File System" diskette to a safe place and proceed to Step 15. We will now boot from the hard disk.

15. Press the Reset button on the front panel, and be ready to press the Space Bar when you see the following message.

16. ALTOS COMPUTER SYSTEMS - 8600
Monitor Version n.nn
Press any key to interrupt boot

17. Press the Space Bar. If you miss the three second margin, an error message will appear. If that happens, return to Step 15.

18. Enter (1) to boot from Hard Disk
Enter (2) to boot from Floppy Disk
Enter (3) to boot from Tape
Enter (4) to boot diagnostics
Enter (5) to enter Altos monitor

Enter option:

19. Enter **1**

20. After a delay of about 15 seconds, this message appears:

Xenix Vn.na

mem = nnnK (indicates available "user" memory)
# (XENIX prompt)

21. If you are re-installing XENIX, go to the next section "Re-installing XENIX". Restore your files before executing **load.hd**.

22. Enter **load.hd<CR>**

23. The system will ask:

Do you have a magnetic tape unit?

24. Respond **y** or **n**, as appropriate.

Operator Actions

System Response

25. The system will display either:

    Installing system with magnetic tape driver

    or

    Installing system without magnetic tape driver

    Please insert the diskette labelled XENIX utilities #1 of n and press RETURN

26. Insert the diskette and press Return.

27. The system will prompt you to load the remaining XENIX installation diskettes in sequence. When you see see each prompt message, remove the prior diskette, load the specified diskette, and press Return when ready.

    As files are copied from diskette to hard disk, you will see messages of the form:

    x "filename", nnnnn bytes, nn tape blocks

    In addition, there are link messages, stating that certain files are linked to other files.

    After the last diskette has been copied to hard disk, the display of filenames will stop. There is a message,

    Remove diskette and store in a safe place.

    At this point, several minutes of internal processing begin. After several minutes, you will see the XENIX prompt:

    #

Note:

Be sure this prompt has appeared before you continue with the next step.

Operator Actions

System Response

28. Enter **fsck /dev/root<CR>**

(This validates the consistency of the file system on the hard disk.)

29. /dev/root
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
** nn files, nn blocks, nnnnn free
#

This indicates that XENIX has been successfully installed. If you see errors other than those mentioned above, return to step 15 and repeat the installation procedure.

Remember, if during your retry you see a message saying the "system was not shut down properly", consult the notes at the end of this procedure.

30. If step 29 was successful, XENIX is correctly installed. Shut the system down gracefully, as follows:

Enter **/etc/haltsys<CR>**

31. ** Normal System Shutdown **

Congratulations on successfully installing XENIX. If you wish to bring XENIX up, see "Starting Up XENIX," later in this section.

Resuming Interrupted Installation

You may receive the following message because you had to interrupt the installation procedure for some reason, or your system was shut down by a power failure or a system crash:

Operator Actions	System Response
	1. The system was not shut down properly, and the root file system should be cleaned. Proceed (y/n)?
2. ALWAYS enter y	

XENIX validates the consistency of the disk file system, which may have been corrupted, and automatically repairs any damage. If there is no damage, you will see the following output:

```
/dev/fd0.root
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
** nn files nn blocks nnnnn free

root structure was ok - proceeding with
bootup

#          (XENIX prompt)
```

If the file system was corrupted, XENIX will repair it automatically, and display a log of the necessary corrections. If you have any doubt whether the file system has been repaired satisfactorily, we recommend you start over with a new set of installation diskettes.

For more information, refer to the FSCK(1) documentation given in the Utility Reference Section of this supplement.

RE-INSTALLING XENIX

Some prior versions of XENIX (Versions 2.0a through 2.0e) have files laid out on the hard disk in a manner that is not compatible with version 2.2a and later versions. The new layout allows the assignment of alternate sectors for bad sectors, and is a major system enhancement. Further information is available in Section 3, "UNIX Manual Changes and Additions". (See LAYOUT(1) and MAP(1) Utilities.)

Note:

If XENIX 2.2a or later is simply installed on top of a prior version, all existing files are destroyed.

The procedure which follows allows the installation of XENIX 2.2a or later, while preserving user files from the prior version. It accomplishes the following:

1. Structuring the new hard disk layout.
2. Replacing prior system files.
3. Preserving non-system (user) files.

Information Regarding Re-installing XENIX

In brief, the procedure is to:

1. Log in as Super-User.
2. Use the tar utility to copy the relevant parts of the system to backup diskettes.
3. Go through the regular installation procedure (described in the preceding section) through the point where **make.hd** has been run to lay out the hard disk.
4. Use the **tar** utility to copy the backup diskettes back to the hard disk.
5. Proceed with the regular installation, which then uses **load.hd** to load all the new system files.

The result is: Where new files have the same designation as prior files, the new files will replace prior files. Where the designation of prior files is unique, as with user libraries, those files will remain.

The number of single-sided, double-density floppy diskettes required depends on how full the hard disk is. The approximate maximum for a fully-loaded hard disk is:

1. Approximately 35 diskettes for a 20-megabyte hard disk.

2. Approximately 74 diskettes for a 40-megabyte disk.

Procedure for Re-installing XENIX

1. Log in as "root" on the system console, in single-user mode.

   If necessary, use the wall utility to ask other users to log off. The who utility will inform you if any others remain. Do not proceed until you are the single user.

2. Use tar to copy all sub-directories of / (the root directory), except /dev. A typical command for this is:

   cd /<CR>
   tar cv bin etc lib lost+found tmp usr<CR>

   Saving will require approximately three minutes per diskette. The number of diskettes and the total time depends on the size of your file system.

   Be sure to label and number each diskette in order as you unload it.

3. Proceed with the regular installation given in the preceding section through **make.hd**. A note in that procedure directs you back to the next task here.

4. AFTER booting from the hard disk and BEFORE running **load.hd**, restore the saved files from the floppies:

   tar xv<CR>

   Insert the diskettes in the order they were made, according to the numbers you placed on their labels. Restoring will take approximately three minutes per diskette.

5. Return to the regular installation procedure and complete it. You will resume the procedure by running **load.hd** to load the contents of the installation diskettes.

GETTING STARTED WITH XENIX

The Altos distribution of XENIX has four login names that require no password. Each has a particular use:

root	logs into the Unix shell, as Super-User
unix	logs into the Unix shell, as a regular user
admin	logs into the Business shell, as Super-User
user	logs into the Business shell, as a regular user

You may set any password you choose with the **passwd** command. (See the **UNIX Programmers Manual, Volume 1**, Section 1).

There are two ways to add new user accounts to the system. Use the ua utility, documented in the Utility Reference Section, or follow the more laborious procedure in Gauthier, Section 9.1.2.

The names "unix" and "user" require no password, and log you in without super-user privileges. If you are new to UNIX systems, and want to explore UNIX concepts, log in as "unix" and consult the book by Gauthier to learn something about the system.

Logging as "user" brings you into the Business Shell, which is a simplified, menu-driven command interpreter with tutorial and guidance facilities.

Before you make regular use of the Business Shell, it may be necessary for a system programmer to make some system adaptations. These adaptations describe the terminal capabilities to the Business Shell, and are explained below. The Business Shell will run without them, but the terminal screen may appear confused.

Customizing the Business Shell

The Business Shell makes use of special terminal capabilities. To find out the capabilities available, it accesses the file /etc/ttytype, which defines the type of the terminal attached to each serial port. It may be necessary for an experienced programmer to edit this file to provide the correct information.

Instructions and reference material for doing this are given in the Appendix entitled **List of Terminal Capablities.**

System Peripherals

As distributed by Altos, the first seven terminal ports are initialized to 9600 baud, asynchronous operation. To change these default baud rates, consult GETTY(8) and TTYS(5) in the **UNIX Programmer Manual, Volume 1.**

Port 8 is configured for a serial line-printer, with asynchronous operation at 9600 baud. For further information, including how to change the default baud rate, see the Appendix entitled **Serial Line Printer and Spooler.**

If you have a parallel printer, see the Appendix entitled **Parallel Line Printer and Spooler** for information on incorporating it in XENIX.

STARTING UP XENIX

After XENIX has been installed on your hard disk, you may start it at any time. (This is known as "bootstrapping" or "booting" for short.) The procedure is below.

An error made when entering information can be corrected by using the Backspace, Rubout, or Delete key.

Operator Actions | **System Response**

1. Press Power On Switch or, with power on, press Reset Button. Be ready to press the Space Bar when you see this message:

2. ALTOS COMPUTER SYSTEMS-8600
   Monitor Version n.nn
   Press any key to interrupt boot

3. Press the Space Bar. If you miss the three second margin, an error message will appear. If that happens, return to step 1.

4. Enter (1) to boot from Hard Disk
   Enter (2) to boot from Floppy Disk
   Enter (3) to boot from Tape
   Enter (4) to boot Diagnostics
   Enter (5) to enter Altos Monitor

   Enter option:

5. Enter **1**

6. After a delay of 20 seconds or so, this message will appear:

   Xenix Vn.na

   mem = nnnK

7. XENIX checks whether the system was shut down gracefully the last time it was used (i.e., with the **haltsys** or **shutdown** command). If so, XENIX assumes the disk file system is in a consistent state, and prompts with the Super-User prompt:

   # (XENIX prompt)

   If you see this, proceed to Step 9.

Operator Actions

System Response

If the system was not shut down gracefully, (for example, shut down by a power failure or a system crash), XENIX prompts as follows:

The system was not shut down properly, and the root file system should be cleaned.
Proceed (y/n)?

8. ALWAYS enter **y**

XENIX validates the consistency of the disk file system, which may have been corrupted, and automatically repairs any damage. If there is no damage, you will see the following output:

```
/dev/root
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
** xx files xxx blocks xx free

root structure was ok - proceeding with
bootup

#
```

If the file system was corrupted, XENIX will repair it automatically, and display a log of the necessary corrections. If you have any doubt whether the file system has been repaired satisfactorily, we recommend you reload the hard disk from a recent dump. (See "Saving and Restoring Files," later in this section.)

Refer to the FSCK(1) documentation in the Utility Reference Section of this supplement for more information.

9. XENIX is now running in "single-user" mode (i.e. only the console is active) and you have all the powers of the Super-User. This means that the File System protection mechanism is temporarily disabled, and therefore you can modify or delete any files you wish. Obviously, this power should be exercised with extreme care. The "#" prompt is a constant reminder of your Super-User status.

Operator Actions

System Response

10. Enter **multiuser<CR>**.

11. The system asks you to enter the current date and time, and then comes up "multiuser". The login prompt appears on all terminals.

    If "multiuser" is not available, see the note at the end of this procedure.

12. Respond with your user name and password.

If you are starting up for the first time, enter either **user<CR>** or **unix<CR>**. No password is required. "User" brings up the Business Shell, "unix" the UNIX shell.

* * * * *

Note: If for some reason, "multiuser" is not available, enter the date and time with **date yymmddhhmm<CR>**, where

yy = current year
mm = current month
dd = current day of month
hh = hour (24 hour clock)
mm = minutes

Follow this with **Control-D** to bring XENIX up "multi-user".

SHUTTING DOWN XENIX

Before powering off your ACS 8600, use one of the following procedures to shut down XENIX gracefully. There is one procedure for the Business Shell, and two for the UNIX shell, depending on whether or not other users are on the system.

An error made when entering information can be corrected by using the Backspace, Rubout, or Delete key.

Business Shell Procedure

Operator Actions

System Response

1. Log in as "admin" on the system console.
2. Select the "Shutdown" entry on the "System Administration" menu.

3. Minutes till shutdown? (0-15):

4. Enter the desired number of minutes. The system will send a message to all users to finish and log off because the system will shut down in so many minutes (the number you specified).

5. The shutdown procedure terminates with the message:

   ** Normal System Shutdown **

6. If there is a floppy diskette in your system's drive, remove it.
7. Turn the power switch to OFF.

UNIX Shell: Procedure if there are Other Users

Operator Actions

System Response

1. Log in as "root" on the system console. This establishes Super-User mode.
2. Enter **/etc/shutdown<CR>**

Operator Actions

System Response

3. Minutes till shutdown? (0-15):

4. Enter the desired number of minutes. The system will send a message to all users to finish and log off because the system will shut down in so many minutes (the number you specified).

6. The shutdown procedure terminates with the message:

** Normal System Shutdown **

7. If there is a floppy diskette in your system's drive, remove it.

8. Turn the power switch to OFF.

UNIX Shell: Procedure if You are the Only User

Operator Actions

System Response

1. Log in as "root" on the system console. This establishes Super-User mode.

2. Enter the commands:

sync<CR>
/etc/haltsys<CR>

3. XENIX responds:

** Normal System Shutdown **

4. If there is a floppy diskette in your system's drive, remove it.

5. Turn the power switch to OFF.

SAVING AND RESTORING FILES

Altos recommends that you back up the file system on your hard disk regularly. The XENIX operating system has several utilities for copying files from hard disk to floppy disk or magnetic tape, and restoring them back to hard disk. This discussion briefly describes several of these programs, and gives some detailed information about one of them, the tar utility.

The utilities most commonly used to back up files are: tar, dd, and dump with its associated restor. For information on these programs, see the entries for TAR(1), DD(2), DUMP(1) and RESTOR(1) in the **UNIX Programmer's Manual**, Seventh Edition, Volume 1. In addition, see also the Utility Reference Section of this supplement, under TAR(1).

Of these programs, tar is the smartest and easiest to use. XENIX supplies an I/O device file named /dev/tar so that file transfers to and from floppy disk (which is the standard device that /dev/tar "talks" to) are simple. Tar can also use the optional cartridge tape.

Dump is a facility for the dated back-up and restore of files. This program is used to archive files whose date of modification is more recent than that of a previous dump, a technique known as an "incremental" dump. Restor is the converse of dump. An additional program, dumpdir, is used to list the names of files captured in a single dump. Dd is a highly-optioned program that allows the user a great deal of autonomy when files are being transferred. All these programs, dd, dump/restor, and dumpdir, require that the user should reasonably familiar with the XENIX system.

USING tar

The tar (Tape ARchive) utility permits backup and restore of single files and whole directories, nested to arbitrary depth. Originally designed for 9-track magnetic tape, tar is equally at home with floppy disk and cartridge tape. It is used in three major ways; to back up a list of files and/or directories, to restore a list of files and/or directories, and to obtain a list of files currently on a particular disk or tape.

Operation with Floppy Diskette

To archive a file or file system on one or more floppy diskettes, proceed as follows. You will need a sufficient number of floppy diskettes that have been formatted for double-density operation. The format utility (described in the Utility Reference Section) can be used to format diskettes. The ADX diskette also has a format utility, but this cannot be run with

XENIX in operation.

As an aid in estimating the number of diskettes you will need, remember that each diskette holds 900 blocks. Files are saved across diskette boundaries.

Insert a blank, double-density formatted floppy diskette, and enter the command:

tar cv "files"<CR>

where "files" is a list of any XENIX files, or directories, separated by spaces. All files specified in the list are archived, including sub-directories and associated files of any directories you have specified.

The system will prompt you to remove and replace diskettes if the material will not fit on the current diskette. Label and number the diskettes. The sequence of diskettes is important. They must be restored in the order they were written.

For example, the command:

tar cv /usr/john<CR>

may be used to archive all files under /usr/john.

Similarly, the commands:

cd /usr<CR>
tar cv *<CR>

may be used to archive all the files under /usr.

To restore a directory, with all its nested directories and files, proceed as follows. Insert the first floppy disk in the sequence, and change directory (**cd**) to the name of the directory to be restored. Enter

tar xv<CR>

For example, to restore the files under /usr/john, check that there is a directory entry for /usr/john. Use the mkdir command to create one, if necessary.

Load the first dump diskette, and enter

cd /usr/john<CR>

Enter

tar xv<CR>

Here is a typical file back-up with the options explained.

tar cv *<CR>

The two options are "c" and "v". Tar is directed to back up all files and directories in the current working directory, "*", onto floppy disk. The options "c" and "v" mean "create" and "verbose", respectively. The create option directs tar to copy the specified files from the system disk to the standard back-up medium, floppy diskette. The verbose option directs tar to display a line on the terminal for each file backed up.

To restore those files and directories, the following may be used:

tar xv<CR>

In this example, the "x" option (for eXtract) and a null file list directs tar to copy every file from the floppy disk onto the system. The "v" (verbose) option operates as in the previous example; tar displays a line on the terminal for each file restored.

The next example obtains a list of files on a floppy diskette.

tar tv<CR>

The "t" option displays a line of information about each file on a floppy. This line is similar to the information obtained from the use of the -l option with ls. The name of the file, its permissions, owner name, date of last modification, and size in bytes are displayed.

Default Actions for tar.

The tar utility saves files to, and restores files from, a device called /dev/tar. This device is assumed as a default by tar, so the command:

tar cv /usr/john<CR>

is equivalent to:

tar cvf /dev/tar /usr/john<CR>

The device /dev/tar is normally equivalent to the floppy diskette device fd0.dd, which is a double-density diskette file having 128-byte sectors. See the appendix entitled **Floppy Disk Organization** for further information.

Operation with Cartridge Tape

With cartridge tape you must specify the tape, /dev/ct0, and a blocksize of eight. (Eight specifies a block of 4096 bytes.) For example, to dump the directory /usr/john:

tar cvbf 8 /dev/ct0 /usr/john<CR>

To restore the same directory, enter:

tar xvbf 8 /dev/ct0 /usr/john<CR>

The invocation

tar cvbf 8 /dev/ct0 *<CR>

performs the same action as "tar cv *", except that the cartridge tape is used in place of the floppy disk. The "b" and "f" options are used to specify the data blocksize and the file device. The blocksize is 8 and the device is /dev/ct0, the cartridge tape.

To restore these files:

tar xvbf 8 /dev/ct0 *<CR>

To obtain a list of files:

tar tvbf 8 /dev/ct0<CR>

See the Appendix entitled **Cartridge Tape Organization** for further information.

Considerations in Using tar

The ordering of options on the tar command line is important; tar will complain if they are not consistent. A good rule of thumb is to learn a particular ordering, such as those above. A skeleton tar line looks like:

tar optionlist option1 ... optionN filelist<CR>

In the previous example, the "b" option is the first option in the options list, hence the specification for the "b" option should be the first item following the options list. In a similar fashion, "f" is second in the option list and its corresponding specification is supplied after the specification for "b".

The form of file and directory names used in the tar command is important. If a file or directory is specified with a complete path name, such as

/usr/john/tempest.c<CR>

tar will remember that a complete pathname was used and when time comes to restore this file, it will be put back as /usr/john/tempest.c, regardless of the working directory of the restorer. Tar will overwrite any previous file of that name, and, if necessary, create a directory in which to put it. A complete pathname is one which begins with "/".

It is sometimes useful to be able to tar off a file or files from a specific directory and restore them in another directory. If, for example, there was a file named "tar.doc" in /usr/john, one might like to be able to place it in /usr/jim. If the file is specified without a path when creating it, it will be restored to the current working directory of the restorer. For example,

tar cv alpha beta gamma<CR>

will archive the files alpha, beta and gamma from the current working directory of the creator, and

tar xv alpha beta gamma<CR>

will restore those same files into the working directory of the restorer.

Remember that diskettes and tapes should be labelled with a description of the files saved and the date, and a sequence number if more than one diskette or tape is used. For information on diskettes, see the Appendix **Diskette Handling.**

Section 3

UTILITY REFERENCE SECTION

USEFUL UTILITIES FOR GETTING STARTED

This section provides a summary of some especially useful utilities that are supplied with the Altos distribution of XENIX. This list is not intended to be complete, but merely a summary of those utilities you will find useful in getting started with XENIX. A complete specification for all utilities may be found in the **UNIX Programmer's Manual**, Volume 1, and a tutorial introduction is presented in the Gauthier book, **Using the UNIX System.**

You may list the full set of utilities supplied with any particular release of XENIX by displaying the contents of the /bin, /usr/bin and /etc directories. The Appendix contains a sample list of utilities.

The Altos implementation of XENIX provides some utilities which differ from standard UNIX, and also some new utilities from various sources. This section documents the changed and new utilities, as "UNIX Manual Changes and Additions." The material supplied in this section may be kept in this supplement or inserted in the **UNIX Programmer's Manual**, as desired.

A List of Useful Utilities for Getting Started

ar	Object library manager and archiver
as	XENIX 8086 relocatable assembler
cat	Display a file
cc	"C" compiler

cd	Change directory. Changes your current position in the File System hierarchy.
chmod	Change mode. Changes file protection attributes
chown	Change file ownership
cmp	Compare two files
cp	Copy a file
diff	Display the differences between two files
ed	The standard UNIX editor
ld	XENIX linkage editor
ls	List. Displays the contents of the current directory
mkdir	Make a new directory
mv	Move. Renames files and directories
od	Displays an octal dump of a file
ps	Display system status
pwd	Print working directory. Displays current position in the directory hierarchy
rm	Remove. Deletes a file
rmdir	Delete a directory
stty	Set terminal options, such as baud rate
tar	File system archiver. May be used for file system dumps and restores

Additional Useful Utilities

In the following pages, "UNIX Manual Changes and Additions," many useful utilities are documented. See the list on the following page for a quick reference to these utilities. Note in particular: format, fcopy, multiuser, and ua, and the new version of tar. The Business Shell, bsh, has two accompanying utilities, menus and digest.

See also the **Addenda One to the XENIX Operating System Supplement** for reference and tutorial material on other utilities, such as the screen editior vi.

UNIX MANUAL CHANGES AND ADDITIONS

Introduction

The material in this section may remain in this supplement or be inserted in Sections 1 through 5 of Volume 1 of the UNIX PROGRAMMERS MANUAL, as you wish. If you insert these documents into the manual, place them in the sections corresponding to the number in parentheses after the utility name. (Entries within sections are in alphabetic order.)

Some of the utilities are enhancements or variations of existing Bell Laboratories UNIX utilities. Others are completely new.

The origin of each utility is specified (in abbreviated form) in column 2 of the table on the immediately following pages.

Utilities labelled (altos) are provided by Altos Computer Systems.

Utilities labelled (bell) were developed by Bell Laboratories after their current manual was published.

Utilities labelled (msoft) were developed by Microsoft, Inc.

There are additional utilities developed at the University of California, Berkeley, which are supplied under license from the Regents of the University. These utilities are documented in the Altos Publication ADDENDUM NUMBER ONE TO THE ACS 8600 COMPUTER SYSTEM - XENIX OPERATING SYSTEM INSTALLATION SUPPLEMENT. That publication also contains other XENIX tutorial and reference material provided by the University.

List of UNIX Manual Changes and Additions

Utility	Source	Description
add.ct(1)	(altos)	Add cartridge tape to system.
add.hd(1)	(altos)	Add additional hard disk to system.
bsh(1)	(altos)	Business Shell. A user system with special guidance and convenience features.
digest(1)	(altos)	Create menu systems for the Business Shell.
fcopy(1)	(altos)	Copy a floppy diskette, while in XENIX.

List of UNIX Manual Changes and Additions (cont.)

Utility	Source	Description
flagbad(1)	(msoft)	Report and fix up physical bad sectors in a file system. This utility is available but not necessary in Version 2.2a or later.
format(1)	(altos)	Format a floppy diskette, while in XENIX.
fsck(1)	(bell)	File system consistency check and interactive repair.
layout(1)	(altos)	Configure a hard disk.
map(1)	(altos)	Create an alternate sector map for a hard disk drive.
multiuser(1)	(altos)	Bring the system up multi-user.
sizefs(1)	(altos)	Determine the size of a logical device from the layout information associated with a hard disk.
tar(1)	(bell)	Tape or floppy archiver. Dumps and restores hard disk files.
ua(1)	(altos)	User administration. Adds and deletes user accounts on the system.
locking(2)	(msoft)	Lock or unlock a record of a file.
rdchk(2)	(msoft)	Check if there is data to be read.
menus(5)	(altos)	Develop menus for Business Shell.
ttytype(5)	(altos)	Data base for defining terminal type associated with each ACS 8600 serial port.

NAME

add.ct - add a cartridge tape drive

SYNOPSIS

add.ct

DESCRIPTION

Add.ct is a shell script which assists the installer of a cartridge tape drive under XENIX. This script requires no interaction with the installer.

The purpose of this script is to produce a device entry for the cartridge tape drive in the /dev directory. When this script is invoked, a device named /dev/ctØ will be created in /dev.

NAME

add.hd - add a second hard disk

SYNOPSIS

add.hd

DESCRIPTION

Add.hd is a shell script which helps the user to install a second hard disk under XENIX. The first thing that the script does is prompt the user for the size of the second drive. It asks you whether you have a 20 or a 40 megabyte drive. Once you reply with a correct number it will tell you that it is making the appropriate sized disk.

Part of the process of making the extra disk is to run the layout(1) program, which divides the disk into two areas. One area is reserved for spare sectors (in case of bad spots), and the other area is ready to be made into a file system. The layout program is immediately followed by the map(1) program, which checks the second drive for bad spots. If there are any, it maps them into the spare area. When the map(1) program is complete (10-20 minutes), a file system is created on the second drive and checked.

As its final act the script creates the directory /usr2, and tells you how to insert the second drive into the XENIX directory hierarchy.

When the shell script is completed and you see the XENIX prompt again, you should add to the file /etc/rc a line which mounts the second drive as a subdirectory, such as:

```
/etc/mount /dev/hd1a /usr2
```

This means that each time you bring the system up multi-user, files and directories created in the directory /usr2 will be physically located on the second drive.

SEE ALSO

layout(1)

BUGS

Add.hd runs significantly slower (3-4 times), when running multi-user.

NAME

bsh -- Altos Computer Systems Business Shell

SYNOPSIS

bsh [-fhas] [menusystem]

DESCRIPTION

Bsh is a menu-driven command language interpreter. It may be installed as the "login shell" in the password file, or it may be invoked directly by the user.

The command is implemented using the termcap and curses facilities from UC Berkeley. It must be run from a terminal which is defined within /etc/termcap.

This command should only be run interactively. A user's terminal may be left in a very strange state if bsh is run in the background.

In the options described below, either "line feed" or "return" performs the newline function.

Options

-f Start bsh in "fast" mode. In this mode, a prompt whose first letter is a lower-case alphabetic or numeric character is executed immediately when the first letter is typed. The system does not wait for a terminating newline. Prompts whose first letter is upper-case alphabetic wait for a terminating newline before executing the requested actions. Fast mode is the default initial mode, if not over-ridden by the command line or the BSHINIT variable (see below). The current mode may be changed during execution through use of the "?mode" command (described below).

-h displays a short help message describing how to invoke bsh.

-a displays a one-line descriptive summary of the syntax used to invoke bsh.

-s Start bsh in "slow" mode. In this mode, all prompts must be terminated by newline before execution occurs. The current mode may be changed during execution through use of the "?mode" command (described below).

A menu system may be specified if desired. In this case, bsh utilizes the designated menu system instead of the default one (/etc/menusys.bin). Prior to use by bsh a menu system must be "digested" using the digest(1) utility. If the specified menu system does not exist or if it is not read-accessible, bsh issues an error message and terminates.

How to create a new menu system and how to update or modify an existing menu system is described in menus(5).

Commands

prompts
Typing any of the prompts on the current menu screen immediately causes the actions associated with the prompt to be executed. It is the responsibility of the menu designer to ensure that reasonable actions exist for each prompt. Selecting a prompt with no associated action causes an error message to be displayed.

An action may be any one of the following:

> Go to a specified menu
> Execute a sh(1) script
> Execute a bsh internal command (e.g. chdir(1))

menuname
Typing the name of a menu causes it to immediately become the current menu. If the menuname is misspelled, or if it does not exist in the current menu system, an error message is displayed.

newline
Typing a newline causes the immediately preceding menu to become the current one. If there is no previous menu, an error message is displayed. Bsh does not distinguish between "line feed" and "return" -- both generate a newline.

? Typing a question mark (?) causes the "help" menu associated with the current menu to be displayed. Help menus are no different from normal menus (except, perhaps, in the type of information they contain). When the current menu is named "xyz", typing a question mark is entirely equivalent to typing "xyz?"

?? Typing a pair of question marks (??) causes the bsh system help information to be displayed. It contains much the same information as is presented here.

menuname?
Typing the name of a menu followed by question mark causes the designated help menu to become the current one.

manualpage??
Typing the name of an entry in the Unix manual followed by two question marks causes the designated manual page to be displayed. Thus, to see the entry for bsh one

may type "bsh??" This is precisely equivalent to typing "!man bsh."

!command
The exclamation point (!) allows the user to "escape" to the standard shell (sh(1)). The command must follow the usual rules as described in the sh(1) documentation. In particular, the command may consist of a sequence of shell commands separated by semicolons -- thus several actions may be invoked. If the command is absent, sh(1) is invoked as a sub-shell with no arguments. In this case, bsh will be resumed as soon as the sub-shell terminates. (Usually, this is accomplished by sending the sub-shell an end-of-file. End-of-file is Control-d on most terminals.) You may escape to the Berkeley C shell (csh(1)) by typing "!csh."

?index
This special command causes bsh to display its internal "index" for the current menu system. The index contains the names of every accessible menu.

?mode
This special command allows the user to change from "slow" mode to "fast" mode and vice versa. The user is asked if he wishes to change to the alternate mode. If your response begins with "y" or "Y", the change is made, otherwise the current mode remains in effect.

interrupt
Bsh will immediately return to the top-level command interpreter upon receipt of an interrupt signal. Such a signal is usually generated via the DEL, RUBOUT or BREAK key.

backspace
Bsh understands the Backspace function (as obtained from /etc/termcap).

CANcel
Bsh interprets the CANcel key to mean "restart input." The CANcel key is Control-x on many of the more popular terminals.

ESCape
Typing an ESCape has the same effect as does typing CANcel.

DC2 If the screen becomes "dirty" for some reason, you can force bsh to clear it and redisplay the current contents by transmitting an ASCII "DC2." This is Control-r on most of the currently popular terminals.

q Typing a "q", "Q" or "Quit" all have the same effect:

bsh is terminated. If bsh is your login shell, "quit" also results in your being logged out.

Environment

BSHINIT

The BSHINIT environment variable contains the initial value of the default mode ("fast" or "slow"). If this variable does not exist in the environment, bsh assumes "fast" mode. BSHINIT should be set by inserting the line BSHINIT="fast" or BSHINIT="slow" into your .profile file.

Note that even if bsh is designated as the "login shell" in /etc/passwd, your .profile file will be interpreted correctly. (See login(1) and sh(1).) In particular, any overriding definitions you may have for the kill and erase characters will be correctly interpreted by bsh.

FILES

~/.profile	contains commands to be executed during login(1)
/etc/menusys.bin	default menu system used by bsh
/etc/passwd	used to define a user's login name, password, home directory, shell, etc.
/etc/termcap	contains terminal attribute descriptions
/usr/lib/bsh.messages	system warning and error messages

SEE ALSO

digest(1M), login(1), menus(5), sh(1), termcap(5)

DIAGNOSTICS

The diagnostics produced by bsh are intended to be self-explanatory.

BUGS

Bsh probably should never allow itself to be run in the background.

Bsh should detect the fact that the current terminal is not defined in /etc/termcap and abort gracefully.

NAME

digest -- create menu system(s) for the Business Shell

SYNOPSIS

digest [options] menufile ...

DESCRIPTION

Digest is used to create a menu system for use by the Business Shell (bsh(1)). This program is also used to modify an existing menu system.

One or more menu systems may be created under control of the options described below:

-h Display an informative summary of the available options and defaults. -a is the same as -h.

-l number
Check for menus longer than number lines in length. The default value is 25 if none is specified. This is the correct maximum number for a conventional 24-line crt screen. In general, number should be one larger than the length of the screen area (as defined by "li" in termcap) for the terminal to be used. The user is responsible for ensuring that the width of a menu will fit onto the terminal(s) he uses. Bsh(1) will truncate lines which are too wide (without issuing a warning message).

-m Multiple menu systems: For each menu file (which must be a directory), produce a separate menu system. The names for each menu system are created by suffixing ".bin" to the menu file name.

-s menu
The starting menu for the generated menu system is the one specified. This option doesn't make much sense if used with the -m option. If no starting menu is specified, the alphabetically first menu name is used for each menu system.

-v Verbose: echo menu names as they are processed.

-o file
The digested output is sent to the named file. By convention, a digested menu system file name should end with a ".bin" suffix.

A menu file may contain one or more menus or directories containing menus. Digest will recursively process all menus within a directory structure.

Note that the -m and -o options are mutually exclusive. The -m option indicates that each menu is to produce a separate ".bin" file: -o indicates that a single output file is to be produced with the name given.

The default output file is "menu1.bin" if none is specified via the -o option, where "menu1" is the first menu file name.

The recommended way to create a menu system is to create a tree of directories containing the various portions of the system. Each subtree contains all the menus related to a given subject. Thus, a primary menu (directory) is created for, say, system management functions and subsidiary menus are placed beneath (within) the directory for each of the individual system management functions or function areas. Help menus may be placed wherever appropriate in the structure.

SEE ALSO

bsh(1), menus(5), termcap(5)

DIAGNOSTICS

The diagnostics produced by digest are intended to be self-explanatory.

BUGS

No outstanding bugs are known.

Digest might check each menu for validity and each menu system for consistency.

NAME

fcopy - copy a floppy diskette

SYNOPSIS

fcopy

DESCRIPTION

Fcopy is used to make duplicate copies of either a single or double density floppy diskette. Fcopy is menu driven and will ask whether you wish to copy a single or double density disk or quit. After one copy has been made, it will ask if you desire to make more copies of the same diskette. All disks must have been previously formatted.

OUTPUT

for double density disks:

```
 13+0      records in
 13+0      records out
900+0      records in
900+0      records out
```

for single density disks:

```
500+0      records in
500+0      records out
```

BUGS

Since the routine was written for a single floppy disk system it reads the entire disk off and then back on, requiring either 913 or 500 blocks of space on the hard disk.

FILES

./junk.?????? Temporary working file, created and subsequently removed by fcopy.

NAME

flagbad - report and fix up physical bad sectors in a file system

SYNOPSIS

flagbad /dev/filesystem

DESCRIPTION

Flagbad reads all the blocks in a file system and if any of the data blocks cannot be read, flagbad chains them on the first i-node, which is otherwise unused.

Flagbad reports all bad blocks that it cannot read, even though it cannot do anything with bad blocks it finds in the i-nodes.

If flagbad is run and finds any bad sectors, fsck(1) must be run to fix up references and free-list counts. Flagbad does not pay attention to where the bad data blocks are found, so it is best to run flagbad on an empty file system.

FILES

Default file systems may vary with installation, but the standard file system for the Altos XENIX distribution is /dev/root.

SEE ALSO

fsck(1), filsys(5)

DIAGNOSTICS

First data block is out of range.

This means that the number of i-node blocks is greater than the size of the file system. (Probably means the device is not a file system or is messed up.)

flagbad: bad i-node block, pn=xxxx

A bad block was found in the i-nodes, flagbad reports it, but is not equipped to do anything about it.

BUGS

Flagbad does not handle errors in the block it wants to use as an indirect block very well.

Flagbad is obsolete for XENIX version 2.2a and later. LAYOUT(1) and MAP(1) supercede it.

NAME

format - format a floppy disk while running XENIX

SYNOPSIS

format

DESCRIPTION

Format is a menu-driven program for formatting floppy disks. Floppy disks can be formatted in either standard IBM single-density format, Altos standard double-density format for XENIX and for diagnostics, or finally double-sided double-density, if that drive is available.

The user enters the name format and is prompted by the menu to select the desired format and insert the diskette.

All double-density diskettes are a combination of two formats, (see the Appendix on floppy diskettes in the Altos XENIX supplement). The first two cylinders are single-density and the rest of the floppy is double-density.

NAME

fsck - file system consistency check and interactive repair

SYNOPSIS

fsck [option] ... [filesystem] ...

DESCRIPTION

Fsck audits and interactively repairs inconsistent conditions for the named filesystems. If a file system is consistent then the number of files, number of blocks used, and number of blocks free are reported. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. Most corrections lose data; all losses are reported. The default action for each correction is to wait for the operator to respond 'yes' or 'no'. Without write permission fsck defaults to -n action.

These options are recognized:

-y Assume a yes response to all questions

-n Assume a no response to all questions

-sX Ignore the actual free list and (unconditionally) construct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done, or extreme care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-core copy of the superblock will not continue to be used, or written on the file system.

The free list is created with optimal interleaving according to the specification X:

-sc:s space free blocks s blocks apart in cylinders of c blocks each.

If X is not given, the values used when the filesystem was created are used. If these values were not specified, then c = 400, s = 9 is assumed.

-SX Conditionally reconstruct the free list. This option is like -sX except that the free list is rebuilt only if there were no discrepancies discovered in the file system. It is useful for forcing free list reorganization on uncontaminated file systems. -S forces -n.

-t If fsck cannot obtain enough memory to keep its tables, it uses a scratch file. If the -t is specified, the file named in the next argument is used as the scratch file. Without the -t option, fsck prompts if it needs a scratch file. The file should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when fsck completes.

If no filesystems are given to fsck then a default list of file systems is read from the file /etc/checklist.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one i-node or the free list.

2. Blocks claimed by an i-node or the free list outside the range of the file system.

3. Incorrect link counts.

4. Size checks:

   Incorrect number of blocks in file.
   Directory size not a multiple of 16 bytes.

5. Bad i-node format.

6. Blocks not accounted for anywhere.

7. Directory checks:

   File pointing to unallocated i-node.
   I-node number out of range.

8. Super Block checks:

   More than 65536 i-nodes.
   More blocks for i-nodes than there are in the file system.

9. Bad free block list format.

10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the "lost+found" directory. The name assigned is the i-node number. The only restriction is that the directory "lost+found" must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making "lost+found", copying a number of files to the directory, and then removing them (before fsck is executed).

Checking the raw device is almost always faster.

FILES

/etc/checklist contains default list of file systems to check.

SEE ALSO

dcheck(1), icheck(1), checklist(5), fs(5), crash(8)

BUGS

I-node numbers for . and .. in each directory should be checked for validity.
The -b option of icheck(1) should be available.

NAME

layout - configure a hard disk

SYNOPSIS

layout layout-device [20 40 add20 add40]

DESCRIPTION

Layout creates a table defining a number of "logical devices" associated with each physical disk in the XENIX system. Layout records this table on cylinder zero of each disk. Each entry in the table is in the following format:

```
struct layout {
        daddr_t l_blkoff; /*Block offset to area */
        daddr_t l_nblocks; /*Number of blocks in area */
};
```

Layout defines ten "logical devices" on the hard disk:

0 The whole disk, with the alternate sector mechanism disabled.

1 The swap area.

2 The root file system.

3-8 Unused.

9 Alternate sector area into which bad disk sectors are automatically mapped by the XENIX kernel.

The logical device numbers above correspond to device numbers in the hard disk driver.

Other device numbers are pre-defined in the XENIX kernel as follows:

10 Future expansion.

11 All of track0.

12 Boot program area.

13 Portion of cylinder zero used for fsck temporary file.

14 Layout information created by this utility.

15 Alternate sector map (see map(1)).

The options to layout are used to create some very common

layouts. The options 20 and 40 create default layouts for either a 20 or 40 megabyte hard disk system.

The options add20 and add40 are used to create a layout for an add-on 20 or 40 megabyte hard disk. The second drive is set up to contain one file system. Its logical devices are:

16 all of hard disk (without sector mapping).

17 additional file system.

25 spare blocks for alternate sector mapping.

27 all of track 0.

29 rest of cylinder 0. (Consists of cyclinder 0 except for track 0.)

30 layout information.

31 mapping information for alternate sectors. (See map (1)).

USAGE

```
layout /dev/hd0.layout 20
layout /dev/hd0.layout 40
```

Configures a 20 or 40 megabyte disk with a layout pre-defined to support XENIX version 2.2a or later.

```
layout /dev/hd0.layout add20
layout /dev/hd0.layout add40
```

Configures an add-on 20 or 40 megabyte drive with a pre-defined layout suitable for a second file system under XENIX version 2.2a or later.

NOTE

Use of the layout utility or the map utility will essentially erase a disk file system created under XENIX version 2.0a through and including version 2.0e. If you are upgrading your system to version 2.2a or later, you must follow the instructions for "Re-Installing XENIX" given in Section 2 of this supplement.

SEE ALSO

map(1), sizefs(1)

NAME

map - create an alternate sector map for a hard disk drive

SYNOPSIS

map layout mapfile drive

DESCRIPTION

Map creates a bad sector map, on mapfile, using the layout information, in layout, created by layout(1). The last argument is the logical device name which references the whole drive.

The standard invocation is:

```
map /dev/hd0.layout  /dev/hd0.secmap  /dev/hd0
```

The structure used for the bad sector to alternate sector mapping is as follows:

```
struct mapsec {
     int   bad_cyl;  /* Cylinder number of bad sector */
     char  bad_hed;  /* Head number of bad sector */
     char  bad_sec;  /* Sector number of bad sector */
     int   bad_good; /* Offset into alternate sector
                        area */
};
```

This structure provides a way for the XENIX hard disk driver to recover from bad sectors it encounters when reading the hard disk. If a bad sector is read, a search of a table of the above structures is made. If an exact match of cylinder, head and sector is found, the corresponding offset is used as an index into the area reserved on the disk for alternate sectors.

NOTE

Use of the layout utility or the map utility will essentially erase a disk file system created under XENIX version 2.0a through and including version 2.0e. If you are upgrading your system to version 2.2a or later, you must follow the instructions for "Re-Installing XENIX" given in Section 2 of this supplement.

SEE ALSO

layout(1), sizefs(1)

NAME

multiuser - bring the system up multiuser

SYNOPSIS

multiuser

DESCRIPTION

Multiuser prompts the user to set the current system date and time, and then brings the system up multiuser.

First, multiuser displays the current system date and time and asks the user to confirm or change the date and then the time. Confirmation is done by entering Return. The format for entering the date is "yymmdd." Time is entered as a 24-hour clock in the form "hhmm."

SEE ALSO

date(1)

NAME

sizefs - determine the size of a logical device from the layout information associated with a hard disk.

SYNOPSIS

sizefs layout-file logical-device-number

DESCRIPTION

Sizefs prints on the standard output the size in blocks of the specified area on the disk. It gets its information out of the structure created by layout(1). Its most common use is in shell scripts for creating a file system on the hard disk, where its output is used as an argument to mkfs(1).

SEE ALSO

layout(1), map(1), mkfs(1)

NAME

tar - tape or floppy archiver

SYNOPSIS

tar [key] [name ...]

DESCRIPTION

Tar saves and restores files on magtape or floppy. Its actions are controlled by the key argument. The key is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying which files are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Note that XENIX, version 2.2b and later, contains a new version of tar, which permits a file to extend across media boundaries. For compatability considerations with the previous version of tar, refer to the BUGS section below.

The function portion of the key is specified by one of the following letters:

r The named files are written on the end of the tape. The c function implies this.

x The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, this directory is (recursively) extracted. The owner and mode are restored (if possible). If no file argument is given, the entire content of the tape or floppy is extracted. Note that if multiple entries specifying the same file are on the tape, the last version will overwrite all preceeding versions.

t The names of the specified files are listed each time they occur on the tape. If no file argument is given, all of the names on the tape are listed.

u The named files are added to the tape if either they are not already there or have been modified since last put on the tape.

c Create a new tape; writing begins on the beginning of the tape instead of after the last file. This command implies r.

The following characters may be used in addition to the letter which selects the function desired.

v Normally tar does its work silently. The v (verbose) option causes it to type the name of each file it treats preceded by the function letter. With the t function, v gives more information about the tape entries than just the name.

w Causes tar to print the action to be taken followed by file name, then wait for user confirmation. If a word beginning with 'y' is given, the action is performed. Any other input means don't do it.

f Causes tar to use the next argument as the name of the archive instead of /dev/tar. If the name of the file is '-', tar writes to standard output or reads from standard input, whichever is appropriate. Thus, tar can also be used to move hierarchies with the command

```
cd fromdir; tar cf - . | (cd todir; tar xf -)
```

b Causes tar to use the next argument as the blocking factor for tape records. The default is 1, the maximum is 8. This option should only be used with raw magnetic tape archives (see f above). Altos recommends a blocking factor of 8 when using the cartridge tape.

l Tells tar to complain if it cannot resolve all the links to the files dumped. If this is not specified, no error messages are printed.

s Obsolete. No longer supported. (Was size parameter, used when files did not cross diskette boundaries.)

FILES

/dev/tar default input/output device
/tmp/tar*

DIAGNOSTICS

Complains about bad key characters and tape read/write errors.
Complains if not enough memory is available to hold the link tables.
Tar will tell you to change volumes when the current volume (floppy or tape) becomes full. It expects you to type one or more characters and then return.

BUGS

This version of tar can read old style tar disks, and the old tar program can read new style tar disks, as long as they do not extend over multiple floppies.
Note that the old version of tar (XENIX 2.2a and earlier) cannot be used to read multiple volume archives created by the new version of tar.
There is no way to ask for the n-th occurrence of a file.
Tape errors are handled ungracefully.
The u option can be slow.

The b option should not be used with archives that are going to be updated. If the archive is on a disk file, the b option should not be used at all, as updating an archive stored in this manner can destroy it.
The current limit on file name length is 100 characters.

EXAMPLES

To dump the directory /usr/john to diskette(s), enter the command

```
tar cvf /dev/fd0.dd /usr/john
```

To dump the same directory to cartridge tape, enter the command

```
tar cvbf 8 /dev/ct0 /usr/john
```

Note that if the device /dev/tar has been configured to reference the floppy disk drive or the cartridge tape, as desired, the above commands can be abbreviated to:

```
tar cv /usr/john
tar cvb 8 /usr/john
```

respectively.

NAME

ua -- user administration

SYNOPSIS

ua [-h]

DESCRIPTION

Ua is used for the addition, deletion and modification of users and groups. It provides an effective means for maintaining the system password (/etc/passwd) and system group (/etc/group) files.

The command is implemented using the termcap and curses facilities from UC Berkeley. It must be run interactively from a terminal which is defined within /etc/termcap.

This command should only be run by Super User -- improper results may occur if it is run by a normal user.

The following option is interpreted by ua:

-h Displays the program's current version and copyright notice as well as a short description of the program's functions.

Ua displays its legal commands at the top of the screen. The "Command?" prompt at the bottom of the screen indicates that ua is awaiting input. The command language syntax is:

[add|delete|show|change] [user <name> | group <name>]

show [Users | Groups]

[help | ?]

! [<shell command(s)>]

quit

The system responds as soon as the first letter of a command is typed. Full command words are not acceptable as input. The case of each word is significant: "group" is not the same as "Group."

Typing an interrupt (usually RUBOUT or DEL) will cause ua to immediately return to the top-level command interpreter.

Add allows the addition of a new user or group. After user/group is specified and a new name provided, the system immediately enters the change command so as to allow modification of the new entry. At the conclusion of the change command the addition is made. If a directory already

exists for a new user, it is not removed. All files under /etc/newuser are copied to the new directory during the user installation process. Typically /etc/newuser will contain the standard versions of the following files: .cshrc, .login, .logout, .profile. The initial value given to a new user ID is one more than the maximum user ID currently in use. The same is true for a new group ID.

Delete allows the deletion of an existing user or group. Deleting a user optionally also deletes his directory and all files contained within it. Deleting a user will not cause all files throughout the system owned by the user to be deleted -- only those beneath his directory. Thus, some files may have an "unknown" owner after a user is deleted. And, if a user is later added with the same user ID as the deleted user, these files will suddenly belong to the new user. The same problem may arise with the deletion and later addition of a group.

Show will show an individual user or group or all users or groups. The word "show" may be omitted if desired.

Change allows the modification of any existing user or group. A special display mode is entered with a menu of fields for selection of the item to be modified. Typing RETURN or LINE FEED in response to a field change request will empty the field. Changes to a user or group change the corresponding entries in the /etc/passwd and /etc/group files. Changing a user's directory entry will not cause a renaming of the actual directory. It is the user's responsibility to ensure that the /etc/passwd and /etc/group files remain consistent.

Help displays a short informative text on the screen. "?" is equivalent to help. The message is the same one as obtained by invoking ua with the "-h" option.

! escapes to the shell (see sh(1)). If no arguments are given, a shell is invoked which will continue until it receives an end-of-file. Then ua resumes. If arguments are present, a shell is invoked with the "-c" option and the arguments are passed along. Ua resumes immediately thereafter. If csh(1) is desired rather than sh(1), the command !csh should be used.

Quit immediately terminates ua and returns to the system.

Any command which is not understood by ua causes an appropriate message to be displayed. As a side-effect, the working portion of the screen is cleared.

Ua does not distinguish between RETURN and LINE FEED. They may be used interchangably.

If the screen becomes "dirty" for some reason, you can force

ua to clear it and redisplay the current contents by transmitting an ASCII "DC2." This is Control-r on most of the currently popular terminals.

Ua understands the Backspace function (as obtained from /etc/termcap). In addition, any time a word is partially formed, the ESCape key will cause the partial word to be discarded and input restarted.

Ua interprets the CANcel key to mean "terminate the current operation." The CANcel key is Control-x on many of the more popular terminals. The CANcel key is more powerful than ESCape, but not so powerful as "interrupt."

Ua will immediately return to the top-level command interpreter upon receipt of an interrupt signal. Such a signal is usually generated via the DEL, RUBOUT or BREAK key.

Ua creates a special user named "standard" in /etc/passwd if one is not already present. This entry is used as the template for installing new users. Thus, if it is desired to have all new users defaulted to the standard UNIX shell (/bin/sh) for the Shell field, it is only necessary to update the Shell field in the "standard" user.

Before adding a new user with a new group, the new group should be added. Otherwise, ua has no way to properly create the new entry in /etc/passwd since it contains group numbers rather than group names.

During program initialization ua copies /etc/passwd and /etc/group to /etc/opasswd and /etc/ogroup, respectively. Thus, if a mistake or disaster occurs during the use of this program, the user may recover the prior state of either or both files.

FILES

/etc/passwd	used for login name to user ID conversions
/etc/group	used for group name to group ID conversions
/etc/opasswd	this file is a copy of /etc/passwd before any modifications are made
/etc/ogroup	this file is a copy of /etc/group before any modifications are made
/etc/newuser	directory containing files which will be installed in a new user's account
/etc/termcap	contains terminal attribute descriptions
/tmp/passwd	temporary file
/tmp/group	temporary file
/etc/ua.lock	lock file

SEE ALSO

group(5), passwd(5)

DIAGNOSTICS

The diagnostics produced by ua are intended to be self-explanatory.

BUGS

Ua should allow specification of alternate passwd and group files.

Complete consistency checking between the /etc/passwd and /etc/group files is not done. In particular, it is possible to install a user with an unknown group in the passwd file and it is possible to install a group with an unknown user in the group file. The shells and directories specified in the /etc/passwd file are not checked for existence or accessibility.

Ua does not check for duplicated user IDs or duplicated group IDs.

Impossible user IDs and group IDs are permitted.

Impossible or non-existent names may be specified for a user's Directory and Shell fields.

The System 3 commands pwck(1M) and grpck(1m) should be incorporated into ua.

NOTE:

DO NOT USE ua TO SET A USER'S PASSWORD. The password would be incorrectly encrypted, and the user would NOT be able to log in successfully. Passwords may only be set with the passwd command, explained in PASSWD(1). The password field displayed by ua is the encrypted version contained in /etc/passwd.

NAME

locking - lock or unlock a record of a file

SYNOPSIS

locking(fildes, ltype, nbytes); int fildes, ltype, nbytes;

DESCRIPTION

locking performs a locking action **ltype** on a record of the open file specified by **fildes**. The record starts at the current file position and has a length of **nbytes**. If the value of nbytes is **0**, the entire file is locked. Nbytes may extend beyond the end of the file, in which case only the process issuing the lock call may access or add information to the file within the boundary defined by nbytes. Thus, lock defines a range in the file controlled by the locking process, and this control may extend to records that have yet to be added to the end of the file. The available values for ltype are:

UNLOCK	0	Unlock the record.
LOCK	1	Lock the given record; the calling process will sleep if any part of the record has been locked by a different process.
NBLOCK	2	Lock the given record; if any part of the record is already locked by a different process, return the error EACCESS.
RLOCK	3	Same as LOCK except that reading is allowed by other processes.
NBRLOCK	4	Same as NBLOCK except that reading of the record is allowed by other processes.

Any process that attempts a read or write on a locked record will sleep until the record is unlocked. If the record is locked with an RLOCK then reading is permissible. When a process terminates, all locked records are unlocked.

SEE ALSO

read (2), write (2), open (2)

DIAGNOSTICS

If an error occurs, -1 is returned. The error code EACCESS is returned if any portion of the record has been locked by another process for the LOCK & RLOCK actions. The error code EDEADLOCK is returned if locking the record would cause a deadlock. This error code is also returned if there are no more free internal locks.

FILES

/usr/include/user.h contains definitions for EACCESS and EDEADLOCK.

/usr/include/sys/locking.h contains definitions for UNLOCK, LOCK, NBLOCK, RLOCK, NBRLOCK.

NAME

rdchk - check if there is data to be read

SYNOPSIS

```
rdchk(fdes);
int fdes;
```

DESCRIPTION

Rdchk checks to see if a process will block if it attempts to read the file designated by fdes. Rdchk returns 1 if there is data to be read or if it is the end of the file (EOF). In this context, the proper sequence of calls using rdchk is:

```
if (rdchk(fildes) > 0) read(fildes, buffer, nbytes);
```

SEE ALSO

read(2)

DIAGNOSTICS

Rdchk returns -1 if an error occurs (e.g., EBADF), 0 if the process will block if it issues a read and 1 if it is okay to read. EBADF is returned if a rdchk is done on a semaphore file or if the file specified doesn't exist.

MENUS

menus -- format of a Business Shell menu system

DESCRIPTION

A menu system is a collection of menus which has been processed (digested) by digest(1M). The Business Shell, bsh(1), requires a menu system upon which to operate: it contains all the menus which are normally displayed to accomplish some set of functions. As distributed, the Business Shell includes the default menu system (/usr/lib/menusys and /etc/menusys.bin).

A menu source file may contain one or more individual menus. However, in the interest of maintainability, it is recommended that each menu source file contain only a single menu, or only very closely related menus. It is also recommended that the name of the menu source file and the menuidentifier be the same.

A source menu system may be a single menu file (containing one or more menus) or a directory structure containing menu files and subordinate directories.

Each menu file is an ASCII file consisting of two logical parts: the body and the actions. A (digested) menu system contains an additional part, the index. The index appears prior to the body. It specifies the byte-offset locations of each of the body and action sections as well as the associated menuidentifiers. Users should never attempt to construct an index by hand -- that is the function of digest(1M). Moreover, users should never attempt to edit a digested menu system; rather, the source menu files should be edited and then the menu system recreated using digest(1M).

The precise format of a menu source file is described below:

&Menuidentifier
The substance of the menu represented essentially as it is to be displayed. Within this area there usually will be one or more occurrences of:

~prompt strings

as well as other special commands as described below.

&Actions size
Zero or more occurrences of:

~prompt
The sequence of actions to be taken

for this prompt. These are bsh(1) commands and/or sh(1) commands.

An example menu for Electronic Mail Services is:

```
&Mail

!date                          \ELECTRONIC~MAIL~SERVICES

          ~a - Receive~mail
          ~b - Send~mail
          ~c - Return~to~starting~menu

&Actions
~a        0
          mail

~b        -1
          echo -n "To whom do you wish to send mail?"
          read x
          echo "Now type the message."
          echo "Terminate it by typing a control -d."
          mail $x
~c
          Start
```

Body

&Menuidentifier must appear beginning in column one. Menuidentifier is any string having relevance to the user. A short descriptive string is usually best. The string may not contain any blanks or punctuation and it must begin with a capital letter. If the string ends with a question mark ("?"), the menu is called a "help menu." It will be invoked automatically when bsh is displaying the base menu and the user types a "?" command. Thus, the &Admin? menu is invoked when &Admin is the current menu and "?" is typed. The remainder of the &Menuidentifier line should be empty.

The body of each menu is composed of text which will be reproduced on the screen exactly as it appears (with exceptions as described below).

~prompt may occur one or more times within the body. This indicates a prompt for which there will be an associated action within the &Actions portion of the menu. Usually there will be a short phrase or sentence describing the action just to the right of the ~prompt. A prompt may be any letter, numeral or string of characters not containing punctuation. Usually shorter (1-2 character) prompts are preferred. A prompt must be separated from its surroundings by one

or more spaces or tabs. If a menu name and a prompt both have the same spelling, the prompt is given preference in all cases.

!date inserts the current date and time, left-justified on the "!." The date/time format is "Tue Jul 13 17:10 1982." !date may appear more than once if desired.

!user inserts the current user id, left-justified on the "!." !user may appear more than once if desired.

!pwd inserts the current directory, left-justified on the "!." The full path name is displayed, e.g. /usr/jones/admin/currwork. !pwd may appear more than once if desired.

!@ indicates where the cursor is to be placed on the screen. Usually this should be just slightly to the right of the current prompt. If !@ is omitted, the cursor will be placed at the bottom left corner of the screen. At most, one occurrence of !@ should appear in each menu.

With the exception of !@, the "!" may appear as a suffix in which case the string will be right-justified instead of left-justified.

The "!" is an "escape characater", and may not be used for any other purpose within a menu.

\string denotes a string which is to be "highlighted" using the terminal's highlighting capabilities (usually reverse video). The "\" character must be on the left of the string. It is converted into the appropriate highlighting information during display. The string may be of any length up to the width of the display screen.

`string denotes a string which is to be "underlined" using the terminal's underlining capabilities (usually true underline). The "`" character must be on the left of the string. It is converted into the appropriate underlining information during display. The string may be of any length up to the width of the display screen.

The backslash "\" and backquote "`" as the initial letter of a string are "escape characters" and will always have the interpretations given above.

In order to create a highlighted or underlined string containing spaces, "significant spaces" may be represented as tildes ("~") within a string. Thus, \~hi~there~ will create a highlighted ten-character string.

The tilde "~" is an "escape character," and may not be used for any other pur-pose within a menu.

Each of the special escape sequences described above must be separated from surrounding text by one or more spaces or tabs.

It is important to know the number of lines and columns of the terminal(s) to be used with a menu system and to be certain not to create menus longer or wider than these values. Their values are specified within the termcap(5) file for each terminal upon which the Business Shell may be run.

Actions

&Actions must appear beginning in column one. &Actions must appear, even if there are no actions. Size is optional. It specifies the length of the window to be used during execution of the actions. If omitted, the default value is 5, and a window 5 rows by column columns will be reserved at the bottom of the screen for output. Column is the terminal column width as obtained from termcap(5). A size of 0 will reserve the entire screen. In this case the screen is blanked prior to execution of the actions; and a prompt requesting a return or line feed is issued after execution. A negative size will reserve the entire screen similarly to the zero case, but after execution, the Business Shell is immediately resumed without waiting for a return or line feed. It is the user's responsibility to ensure that the action window is large enough.

Each prompt in the actions section must be reproduced exactly as it appears in the body of the menu. It is the user's responsibility to ensure that the spelling of prompts in the body and actions sections match. The case of characters is significant; so "A" is not the same as "a."

The actions may be composed of bsh(1) commands or commands which are executed by the standard shell, sh(1). The actions should all be indented one tab stop from the left side of the file.

A bsh(1) command is the instruction to transfer immediately to a particular menu. This is specified by writing the name of the destination menu in the semantics field. Bsh(1) commands must be typed one-per-line.

Sh(1) commands follow the usual rules as described in Section 1 of this manual.

Since a menu file may contain one or more menus or directories containing menus, the recommended way to create a menu system is to create a tree of directories containing the various portions of the system. Each subtree contains all the menus related to a given subject. Thus, a primary menu (directory) is created for, say, system management functions: and subsidiary menus are placed beneath (within) the directory for each of the individual system management functions or function areas. Help menus may be placed wherever appropriate in the structure.

FILES

/usr/lib/menusys	source directory for /etc/menusys.bin
/etc/menusys.bin	digested default menu system for bsh(1)

SEE ALSO

bsh(1), digest (1M), sh(1), termcap(5)

NAME

ttytype - data base defining terminal type; used for associating terminals with serial ports

DESCRIPTION

The ttytype data base is used to associate a manufacturer's terminal with the different serial ports on the system. Each line contains the name of a terminal, a tab character, and then the XENIX device entry for the serial ports associated with that terminal. The terminal name must correspond to an entry in /etc/termcap.

Making an entry in the ttytype file for your terminals allows the system to make maximum use of terminal features for certain system facilities that use full screen capabilities. Among these programs are vi(1), bsh(1), and ua(1).

FILES

/etc/ttytype

SEE ALSO

vi(1), bsh(1), ua(1), termcap(5)

USAGE

A typical line in the ttytype file might look like "dumb /dev/tty3" or "wyse /dev/tty5." The first says that serial port 3 is connected to a terminal described in /etc/termcap as having no special characteristics such as cursor movement. The second entry tells XENIX that serial port 5 is connected to a terminal manufactured by Wyse Technology that is described in termcap under the name "wyse." The terminal name is the name found between the first and second vertical bars of the appropriate entry in /etc/termcap.

Appendix A

HARD DISK ORGANIZATION

CONFIGURATION

The built-in hard disk on ACS 8600 twenty-megabyte (-12), and forty-megabyte (-14), systems is configured as follows:

Twenty Megabytes (-12)

Cylinder	Use
0	Bootstrap program
1-35	Swap area
36-37	Alternate-sector area
38-end	XENIX file system

Forty Megabytes (-14)

Cylinder	Use
0	Bootstrap program
1-17	Swap Area
18-19	Alternate-sector area
20-end	XENIX file system

LOGICAL DEVICES

The following logical devices are defined in the ALTOS configuration of XENIX.

Logical Devices - Integral Hard Disk

	Logical Device	Description
0	hd0	all of hard disk (without sector mapping).
1	hd0a, swap	swap area.
2	hd0b, root	root file system.
3-8		unused.
9	hd0.spares	spare blocks for alternate sector mapping.
10		future expansion.
11	hd0.track0	all of track 0.
12	hd0.boot	primary bootstrap on track 0.
13	hd0.roc0	rest of cylinder 0. (Consists of cylinder 0 except for track 0.) Used for fsck temporary file.
14	hd0.layout	layout information. (See layout (1)).
15	hd0.secmap	mapping information for alternate sectors (See map (1)).

Logical Devices - Additional Hard Disk

	Logical Device	Description
16	hd1	all of hard disk (without sector mapping).
17	hd1a	additional file system.
25	hd1.spares	spare blocks for alternate sector mapping.
27	hd1.track0	all of track 0.

Logical Devices - Additional Hard Disk (cont.)

	Logical Device	Description
30	hd1.layout	layout information. (See layout (1)).
31	hd1.secmap	mapping information for alternate sectors. (See map (1)).

Appendix B

FLOPPY DISKETTE ORGANIZATION

CONFIGURATION

The floppy disk organization for a bootable XENIX File System is as follows:

Track	Use
0	Bootstrap program
1	Unused
2	Swap area
3-end	Xenix file system

LOGICAL DEVICES

The following logical devices are defined in the ALTOS configuration of XENIX:

Logical Device	Track	Definition
fd0.dd	2-end	"Pseudo-tape" (single-sided, double density, see below)
fd0.swap	2	swap area (double density)
fd0.root	3-end	File System (double density)
fd0.boot	0	Bootstrap (single density)
fd0.sd	0-end	Standard IBM format (single density)

FURTHER INFORMATION

Double-Sided Drive

If you have an ACS-8600 with a double-sided double-density diskette drive, you may refer to it as follows:

Device Name	Device Numbers Major/Minor	Definition
fd0.2s	1/7	Cylinders 2-77, double-sided, double-density 512-byte sectors.

Booting From Floppy Diskette

Because the swap area on the floppy disk is very small, XENIX should not be run in multi-user mode after booting from the floppy. Of course, it is fine to access a file system on the floppy after booting off hard disk since the swap area is then on the hard disk.

Diskettes as Pseudo-Tape

The floppy disk may be used sequentially as a "pseudo-tape", for example, by the tar utility. The command:

tar c file1 file2<CR>

archives file1 and file2 to the device /dev/tar, which is usually equivalent to the floppy disk device /dev/fd0.dd

These files may be recovered from that diskette with the command:

tar x<CR>

For information on using this utility, see the earlier section, "Saving and Restoring Files Using tar".

Diskettes as Random-Access Files

Floppy diskettes may also be used as mountable/demountable random access file systems. To use a diskette in this mode, you must first initialize it with an empty file system as follows:

a. If necessary, format the diskette (double-density), using the format utility. (Described in "UNIX Manual Changes and Additions", in Section 3 of this Supplement.

b. Load the formatted diskette.

Diskettes as Random-Access Files (cont.)

c. Enter

/etc/mkfs /dev/fd0.root 888<CR>

Although the newly created file system is physically loaded, you must "mount" the file system before you can use it. This links the diskette's file system with the root XENIX file system on your hard disk.

Similarly, you must "dismount" (or unlink) the file system before physically unloading it.

To "mount" a file system:

a. Load the diskette, if necessary.

b. Enter

/etc/mount /dev/fd0.root /maint<CR>

You may now access the diskette's file system through the root directory /maint. Note that diskettes must be "mounted" with the mount command each time they are loaded in the drive.

Before removing a file system diskette, dismount it:

a. Enter

/etc/umount /dev/fd0.root<CR>

b. Unload the diskette from the drive.

Appendix C

DISKETTE HANDLING

The following information is useful for doing the installation procedures, for regular handling of diskettes, and for arranging proper storage of diskettes.

DISKETTE INSERTION

To insert an 8" floppy diskette in the drive, open the drive door cover and insert the diskette with the diskette manufacturer's label up. The door cover can be opened by pressing the small rectangular latch beneath the cover. Seat the diskette gently and close the door. (Note: inserting it upside down does no harm; the system just will not respond).

DISKETTE WRITE PROTECTION

Eight-inch floppy diskettes are furnished in two ways; either write-enabled or write-protected. Write-enabled diskettes have the write notch closed and write-protected diskettes have the write notch open.

Hold the diskette so that the label faces you and is the top of the diskette. The write notch, if there is one, will be along the bottom edge about 1-3/4 inches in from the bottom right hand corner.

If the there is no notch, the diskette is write-enabled; the diskette can be written to, or erased from.

If there is a notch in the diskette at the position specified, the diskette is write-protected; it cannot be written to nor erased from.

If the diskette is write-protected, it can be write-enabled by covering the write notch with a special silver-tape piece provided by the diskette manufacturer. This tape is best for the purpose, because the covering material must block infrared light. Scotch tape and certain other tapes will not work.

After you have prepared a diskette and wish to write-protect it, you may remove the silver tape. Thereafter you may neither write to nor erase from the diskette unless you replace the tape.

LABELLING DISKETTES

Use only felt-tip pens when writing on the diskette jacket or label. A pencil or pen can cause surface indentations which will result in errors when the diskette is used. It is best to prepare and write diskette labels before placing them on diskettes.

STORING DISKETTES

In daily use of your computer system, make backup copies of new files and changed files. Store backup diskettes in a safe place separate from your working computer system area. Store all diskettes in their jackets and inside boxes when they are not in use. A metal box is best.

Dust on a diskette surface damages the disk drive as well as the diskette. Heat damages diskettes. Avoid temperatures above 115 degrees Farenheit and below 40 degrees. Avoid humidity above 80%.

Magnetic fields can damage the data integrity of the diskette. Magnetic fields are more common than usually realized. Electrical equipment generates a magnetic field. The ringing of a telephone near a diskette can damage it. Screwdrivers, even if not magnetized originally, can be magnetized by use. Avoid placing screwdrivers or other small metal tools on or near a diskette.

Elevators and elevator shafts have electrical cabling which can put out momentary surges of magnetic fields. Do not store diskettes or other magnetic media near a wall behind which is an elevator.

Appendix D

SERIAL LINE PRINTER AND SPOOLER

In the Altos distribution of XENIX, serial port 8 is configured for a serial printer operating at 9600 baud. The logical device name /dev/lp may be used to refer to this port, and the lpr utility references this device automatically for printing and spooling.

The printer should be configured to use the X-ON, X-OFF protocol, because XENIX uses this protocol to control the flow of data to the serial printer.

The lpr utility assumes that only one printer, /dev/lp, is attached to the system. The printer may be a serial or a parallel printer. See the next appendix for information on using a parallel printer. It is possible to attach both a serial and a parallel printer to the system, but the lpr utility will make use of only one of them. The second printer would have to be explicitly referred to with an appropriate device name when used for output.

If you wish to support eight terminals (with no printer), you should:

1. Remove the lp entry in /dev. This is done in super-user mode by entering **rm /dev/lp<CR>**

2. Enable the login and shell on port 8 by editing the line referencing tty8 in the file /etc/ttys from "02tty8" to "12tty8", before going multi-user.

If you wish to run the serial printer at other than 9600 baud, the simplest method is to treat the printer as a terminal. Enable the login and shell on port 8, as in step 2, above, and set the baud rate as necessary using the information in TTYS(5) and GETTY(8) of the **UNIX Programmer's Manual**, volume 1. In this case the spooled output will contain one initial "login" prompt,

but will otherwise be as before.

Appendix E

PARALLEL LINE PRINTER AND SPOOLER

In the Altos distribution of XENIX, serial port 8 is configured for a serial printer operating at 9600 baud. The logical device name /dev/lp may be used to refer to this port, and the lpr utility references this device automatically for printing and spooling.

The logical device /dev/plp can be used to reference the parallel port with a Centronics-type interface. It is possible to have both a serial and a parallel printer attached to the system, but the lpr utility will make use of only one printer, /dev/lp, for printing and spooling.

To use the lpr utility with a parallel port:

1. Remove the lp entry in /dev. This is done in super-user mode by entering

   rm /dev/lp<CR>

2. Move the plp entry to /dev/lp by executing the command

   mv /dev/plp /dev/lp<CR>

Appendix F

CARTRIDGE TAPE ORGANIZATION

The add-on cartridge tape drive for ACS 8600 systems has the following characteristics:

MEDIA

Scotch(r) DC 300XL data cartridge or equivalent

SPECIFICATIONS

Four tracks; capacity is approximately three megabytes per track, twelve megabytes per tape.

The approximate capacity is calculated assuming a four-kilobyte block size.

CONFIGURATION

There are no pre-defined allocations or special uses of any portion of the cartridge tape.

All four tracks are treated as contiguous; that is, the tape appears to be a single track, twelve megabytes in length.

A single logical device is defined for cartridge tape access:

	Device Number	
Logical Device	**Major**	**Minor**
ct0	5	0

Appendix G

LIST OF TERMINAL CAPABILITIES

The basic XENIX system works with nearly all the generally available terminals, by making use of a standard "lowest common denominator" of terminal capabilities. However, some of the XENIX utilities, including many especially useful utilities, can make use of special terminal capabilites. A major example is the Business Shell.

For this reason, the /etc/termcap data base has been developed to describe terminal capabilities. The following pages give essential information extracted from /etc/termcap, in a form more easily understood that when the file itself is viewed. The information given describes all terminals currently supported for special use by the Altos release of the XENIX operating system.

Customizing Your Altos XENIX System

The following information explains how to inform XENIX of the special capabilities of the terminals you are using with the system. The operation should be done by an experienced programmer.

The XENIX utilities, such as the Business Shell, that make use of special terminal capabilities access the file /etc/ttytype, which defines the type of the terminal attached to each serial port. It may be necessary to edit this file to provide the correct terminal type for each port. Each line in /etc/ttytype has two fields; the terminal type, and the associated port number.

The "terminal type" used in /etc/ttytype is the second field of the appropriate terminal entry in the /etc/termcap data base; that is, it is between the first two vertical bars, "|", in the entry. On the following list of "Terminals Supported by the

XENIX Operating System," the entry called "name of terminal" is the proper reference. When editing /etc/ttytype, use that name as the "terminal type." Any editor that is convenient can be used. Change the terminal type, if necessary, for each active serial port that your system uses. (See TTYTYPE(5) in the Utility Reference Section for more information about /etc/ttytype.)

For example, if you are using a TeleVideo terminal, current model 920, when you consult the following list you will find a group of entries for Televideo. The appropriate entry is "920b." Editing /etc/ttytype, you find that all ports are associated with "wyse." Change the port assignments you are using, or all port assignments, to "920b" and update the file.

If you do not find a listing for the terminal you are using, consult your dealer or Altos Customer Support.

Terminals Supported by the XENIX Operating System

The material below is derived from the system file /etc/termcap. This file describes terminal capabilities and characteristics. The use of this file is to support screen-oriented programs, such as vi. The /etc/termcap file is composed of a description entry for each supported terminal (and sometimes more than one, if the terminal has options, or is part of a family of products).

This document cites the name by which a particular terminal is known to the system, and contains a short description of the terminal, including the manufacturer's name, and other useful information. Included are comments relevant to the use of the terminal.

an example:

```
wyse                            WYSE WY-100
```

this entry indicates that the WYSE WY-100 terminal is supported by the system and that its name is 'wyse' (case is significant).

The 'name' of a terminal is specified to several system programs. Among them are:

the shell (sh):

```
% TERM = name; export TERM
```

the C shell (csh):

```
# setenv TERM name
```

or, for the default definition of a port.

a typical line in /etc/ttytype:

```
name    tty5
```

Please reference the appropriate documentation for an expanded explanation of the capabilties and uses of the above programs and structures.

Terminal naming conventions:

Terminal names look like:
<manufacturer> <model> - <modes/options>

Certain abbreviations (e.g. c100 for concept100) are also allowed for upward compatibility. The part to the left of the dash, if a dash is present, describes the particular hardware of the terminal. The part to the right is used for flags indicating special ROM's, extra memory, particular terminal modes, or user preferences. Names are always in lower case, for consistency in typing.

The following are conventionally used flags:

- rv Terminal in reverse video mode (black on white)
- 2p Has two pages of memory. Likewise 4p, 8p, etc.
- w Wide - in 132 column mode.
- pp Has a printer port which is used.
- na No arrow keys - termcap ignores arrow keys which are actually there on the terminal, so the user can use the arrow keys locally.

Special manufacturer codes:

- A: hardcopy daisy wheel terminals
- M: Misc. (with only a few terminals)
- q: Homemade
- s: special (dialup, etc.)

the terminals:

name of terminal	description
wyse	WYSE WY-100
du	dialup
hp	Hewlett-Packard
2621-nl	HP 2621 with no labels
2621	HP 2621
2621-wl	HP 2621 w/labels
hl9-u	Heathkit with underscore cursor
hl9-us	Heathkit w/keypad shifted/underscore cursor
hl9-bs	Heathkit w/keypad shifted
hl9	Heathkit hl9
c100-rvs	slow reverse Concept 100
c100-s	slow Concept 100
c100-rvna	c100 with no arrows
c100-rvpp	c100 with printer port
c100	Concept 100
c100-rv	c100 rev video
adm3a	LSI ADM3A
adm3	LSI ADM3
mime	Microterm Mimel
bg	BBN BitGraph terminal
vt52	DEC VT52
gigi	DEC GIGI

A: DAISY WHEEL PRINTERS

The A manufacturer represents Diablo, DTC, Xerox, Qume and other Daisy wheel terminals.

1620	Diablo 1620
1620-m8	Diablo 1620 w/8 column left margin
dtc	DTC 382 with VDU
dtc300s	DTC 300s
gsi	
aj830	Anderson Jacobson
5520	NEC Spinwriter 5520
qume5	Qume Sprint 5

x1720	Xerox 1720 same as the Diablo 1620.

C: CONTROL DATA

cdc456	CDC
cdc456tst	CDC

D: DATAMEDIA

dm1520	Datamedia 1520
dm2500	Datamedia 2500
dm3025	Datamedia 3025a
3045	Datamedia 3045a
dt80	Datamedia dt80/1
dt80w	Datamedia dt80/1 in 132 char mode

H: HAZELTINE

h1000	Hazeltine 1000
h1552	Hazeltine 1552
	(be sure auto lf/cr switch is set to cr)
h1552rv	Hazeltine 1552 reverse video
h1420	Hazeltine 1420
h1500	Hazeltine 1500
h1510	Hazeltine 1510
h1520	Hazeltine 1520
h2000	Hazeltine 2000

I: IBM, INTERACTIVE SYSTEMS, and INTECOLOR

8001	ISC8001 Compucolor/Intecolor
compucolor2	CompucolorII
intext	Interactive Systems Corporation
	(modified PE Owl 1200)
ibm	IBM 3101-10

M: MISCELLANEOUS TERMINALS

tab132	TAB 132/15
tab132w	TAB 132/15
tab132rv	TAB 132/15
tab132wrv	TAB 132/15
mw2	Multiwriter 2
trs80	TRS-80 Radio Shack Model I
d800	Direct 800/A
vc404	Volker-Craig 404
vc404s	Volker-Craig 404 w/standout mode
vc404na	Volker-Craig 404 w/no arrow keys
vc404sna	Volker-Craig 404 w/standout mode and no arrow keys
vc303a	Volker-Craig 303A

```
vc303           Volker-Craig 303

ampex           Ampex Dialogue 80
d132            Datagraphix 132a
soroc           Soroc 120

tec400          TEC scope
tec500          TEC 500
tec

teletec         Teletec Datascreen
digilog         Digilog 333
ep48            Execuport 4080
ep40            Execuport 4000
terminet1200    GE Terminet 1200
aed512          AED 512
datapoint       Datapoint 3360
falco           Falco TS-1
dg              Data General 6053
cdi             CDI 1203
        (^S is an arrow key. not recommended for use)
sol
x183            Cybernex XL-83
omron           Omron 8025AG
plasma          Plasma Panel
swtp            Southwest Technical Products CT82
terak           Terak emulating Datamedia 1520
virtual         CB unix virtual terminal
delta           Delta Data 5000
md1110          Cybernex MDL-110
zen30           Zentec 30


N: ANN ARBOR

aa              Ann Arbor 4080
aaa-18          Ann Arbor Ambassador/18 lines
aaa-20          Ann Arbor Ambassador/20 lines
aaa-22          Ann Arbor Ambassador/22 lines
aaa-24          Ann Arbor Ambassador/24 lines
aaa-26          Ann Arbor Ambassador/26 lines
aaa-28          Ann Arbor Ambassador/28 lines
aaa             Ann Arbor Ambassador/30 lines
aaa-36          Ann Arbor Ambassador/36 lines
aaa-40          Ann Arbor Ambassador/40 lines
aaa-48          Ann Arbor Ambassador/48 lines
aaa-60          Ann Arbor Ambassador/60 lines
aaa             Ann Arbor Ambassador
aaa-db          Ann Arbor Ambassador 30
        (destructive backspace)


T: TELETYPE

33              Model 33 Teletype
43              Model 43 Teletype
```

37 Model 37 Teletype

V: VISUAL

vi200	Visual 200 with function keys
vi200-rvic	Visual 200 reverse video using insert char
vi200-f	Visual 200 no function keys
vi200-rv	Visual 200 reverse video
vi200-ic	Visual 200 using insert char

X: TEKTRONIX

tek	Tektronix 4012
tek4013	Tektronix 4013
tek4014	Tektronix 4014
tek4015	Tektronix 4015
tek4014-sm	Tektronix 4014 in small font
tek4015-sm	Tektronix 4015 in small font
tek4023	Tektronix 4023
4025	Tektronix 4024/4025/4027
4025-17	Tektronix 4025 17 line window
4025-17ws	Tektronix 4025 17 line window in workspace
4025ex	Tektronix 4025 w/!

a: ADDS

Regent: lowest common denominator, works on all regents.

regent	ADDS Regent series - works on all of series
regent100	ADDS Regent 100
regent20	ADDS Regent 20
regent25	ADDS Regent 25
regent40	ADDS Regent 40
regent60	ADDS Regent 60
regent60na	ADDS Regent 60 w/no arrow keys
a980	ADDS Consul 980

Note: if return acts strangely on a980, check internal switch 2 on the top chip on the CONTROL PC board.

viewpoint	ADDS Viewpoint

b: BEEHIVE

sb2	fixed Super Bee
bh3m	BeehiveIIIm
superbeeic	Super Bee with insert char
microb	Micro Bee series
sb1	Beehive Super Bee - f1=escape, f2=^C.

c: CONCEPT (HUMAN DESIGNED SYSTEMS)

There seem to be a number of different versions of the C108 PROMS. The first one that we had would

lock out the keyboard of you sent lots of short lines (like /usr/dict/words) at 9600 baud. Try that on your C108 and see if it sends a ^S when you type it. If so, you have an old version of the PROMs. The old one also messed up running vi with a 132-character line-length. You should configure the C108 to send ^S/^Q before running this.

```
c108            Concept 108 w/8 pages and ^S/^Q
c108            Concept 108 w/4 pages and ^S/^Q
```

Concepts have only window relative cursor addressing, not screen relative. To get it to work, a one page window scheme is used for screen style programs.

d: DEC (DIGITAL EQUIPMENT CORPORATION)

It is assumed that you have smooth scroll off or are at a slow enough baud rate that it doesn't matter (1200? or less). Also this assumes that you set auto-nl to on; if you set it off, use "vt100-nam".

The xon/xoff switch should be on.

```
vt100           DEC VT100
vt100           VT100 w/no am
gt42            DEC GT42
vt132           VT132
gt40            DEC GT40
vt50            DEC VT50
dw1             Decwriter I
vt50h           DEC VT50h
ovt100          old DEC VT100
vt100-s         DEC VT100 132 cols 14 lines
        (w/o advanced video option)
vt100-w         DEC VT100 132 cols (w/advanced video)
dw2             Decwriter II
dw4             Decwriter IV
```

h: HEWLETT PACKARD

```
2621-ba         2621 w/new rom, strap A set
2621k45         HP 2621 with 45 keyboard
        (should be used at 4800 baud or less)
hp2645          HP 264x series
hp2626          HP 2626
        (should be used at 4800 baud or less)
hp2648          HP 2648a graphics terminal
2640            HP 2640a
2640b           HP 264x series
2621-48         HP 48 line 2621
2621-nt         HP 2621 w/no tabs
        (2621 with no labels ever)
```

i: INFOTON (GENERAL TERMINAL)

```
i100            General Terminal 100A
        (formerly Infoton 100)
i400            Infoton 400
addrinfo
infotonKAS
```

k: HEATHKIT (ZENITH)

```
h19-a           Heathkit H19 ANSI mode
```

l: LEAR SIEGLER (ADM)

If the adm31 gives you trouble with standout mode, check the DIP switch in position 6, bank @c11, 25% from back end of pc. Should be OFF. If there is no such switch, you have an old adm31 and must use oadm31.

```
adm31           LSI adm31
adm2            LSI adm2
adm42           LSI adm42
adm5            LSI adm5
adm3a+          ADM3A PLUS
oadm31          old ADM31
```

m: MICROTERM

These mimel entries refer to the Microterm Mime I or Mime II. The default mime is assumed to be in enhanced act iv mode.

```
mime3a          Mimel emulating 3a
microterm        Microterm Act iv
microterm5       Microterm Act v
act5s           skinny act5 -  Act V in split screen mode
mime-fb         full bright Mimel
mime-hb         half bright Mimel
mime2a-s        Microterm Mime2a
        (emulating an enhanced Soroc IQ120)
        (but ^X can't be used as a kill character)
mime2a          Microterm Mime2a
        (emulating an enhanced vt52)
mime-3ax        Mimel emulating enhanced 3a
```

p: PERKIN ELMER

```
pe550           Perkin-Elmer 550
fox             Perkin-Elmer 1100
owl             Perkin-Elmer 1200
```

s: SPECIALS

Special "terminals". These are used to label tty lines when you don't know what kind of terminal is on it. The characteristics of an unknown terminal are the lowest common denominator - they look about like a TI 700.

arpanet	network
bussiplexer	
ethernet	network
lpr	line printer
dumb	unknown
switch	intelligent switch

t: TEXAS INSTRUMENTS

ti	TI Silent 700
ti745	TI Silent 745
ti800	TI Omni 800

v: TELEVIDEO

note: the 912 has a <funct> key that's like shift: <funct>8 xmits "^A8\r". The 920 has this plus real function keys that xmit different things. Termcap makes you use the funct key on the 912 but the real keys on the 920.

tvi912	TVI920 old TeleVideo
912b	TVI new TeleVideo 912
920b	TVI new TeleVideo 920
tvi912-2p	TeleVideo w/2 pages set to page 1 when entering ex or vi. reset to page 0 when exiting ex or vi.
tvi950-ap	TVI 950 w/alt pages
tvi950-b	bare TVI 950 no is
tvi950-ns	TVI 950 w/no standout

note:
The following TVI descriptions are for all 950's.
It sets the following attributes:

full duplex
write protect off
conversation mode
graphics mode off
white on black
auto page flip off
turn off status line
clear status line
normal video
monitor mode off
edit mode

```
  load blank char to space
  line edit mode
  enable buffer control
  protect mode off
  local edit keys
  program unshifted send key to send line all
  program shifted send key to send line unprotected

  set the following to nulls:
     field delimiter
     line delimiter
     start-protected field delimiter
     end-protected field delimiter

  set end of text delimiter to carriage return/null
  clear all column tabs


tvi950          TeleVideo 950

note:
  tvi950 sets duplex (send) edit keys (\El) when entering vi
         sets local (no send) edit keys (\Ek) when exiting vi


tvi950-2p       TeleVideo 950 w/2 pages

note:
  tvi950-2p is for 950 with two pages adds the following:
     set 48 line page
     place cursor at page 0, line 24, column 1
     when entering ex, set 24 line page
     when exiting ex, reset 48 line page
              place cursor at 0,24,1

tvi950-4p       TeleVideo 950 w/4 pages

note:
  tvi950-4p is for 950 with four pages adds the following:
     set 96 line page
     place cursor at page 0, line 24, column 1
     when entering ex or vi, set 24 line page
     when exiting ex or vi, reset 96 line page,
     place cursor at 0,24,1

tvi950-rv       TeleVideo 950 rev video
tvi950-rv2p     TeleVideo 950 rev video w/2 pages
tvi950-rv4p     TeleVideo 950 rev video w/4 pages


y: TELERAY

t3700           dumb Teleray 3700
t3800           Teleray 3800 series
tl061           Teleray 1061
tl061f          Teleray 1061 with fast PROMs
```

Appendix H

NUMERIC FORMATS, C AND FORTRAN 77

The following information is for reference only. This information on the internal formats used for numeric representation is not necessary for general use of the C language or Fortran 77. It can be useful when examining actual memory contents or doing other specialized system programming work.

The same formats are used by both languages.

INTEGER FORMATS

Integers and "short integers" are 16 bits in length. "Long integers" are 32 bits. For both sizes, the leftmost bit is a sign bit and the other 15 or 31 bits are magnitude. The sign is zero for positive, one for negative. Negative numbers are in twos-complement form.

The range of values is as follows:

Signand 15 bits	-32,768 to 32,767
Sign and 31 bits	-2,147,483,648 to 2,147,483,647

FLOATING-POINT FORMATS

Single precision floating point is 32 bits in length, double is 64. The leftmost eight bits consist of an exponent in excess 80 notation. "Excess 80" means that the hexadecimal values from 80 to FF are positive exponents, corresponding to 0 through 7F. Values less than 80 are negative exponents; 7F through 0 correspond to -1 through -7E.

The remaining 24 or 56 bits consist of a leading sign bit

and magnitude values. Magnitudes are normalized. "Normalized" means that the representation of magnitude and exponent is adjusted so that each magnitude value can be thought of as starting with .1nnn...

For example, the value of 101, decimal 5, would be .101 with an exponent of 3. The leftmost digit of magnitude does not need to be represented, because it is always 1 except for the special case of a value of zero. Therefore, the leftmost magnitude bit is not stored but is implied. It is referred to as the "hidden bit."

Example:

The value 15.25, decimal. In binary, this is 1111.01

(In binary, .1 = .5 decimal; .01 = .25, etc. Moving to the right of the point halves the value at each move, just as moving to the left of the point doubles the base 2 value.)

So, 1111.01 represents 15.25 decimal. Normalizing our binary value, we have .111101 with an exponent of 4. The exponent becomes 84 in excess 80 notation, or 1000 0100 in binary. The sign bit is zero (positive), and the magnitude is 11101000... with as many trailing zeros as needed. Notice that the leading ".1" has disappeared. It is the unnecessary "hidden bit." The binary and hexadecimal values are shown below.

```
1000 0100 0111 0100 0000 0000 0000 0000
  8    4  s 7    4    0    0    0    0
```

The example is single-precision. Double precision, in this case, would be the same with eight bytes (32 bits) of trailing zeros.

Other examples:

The fractional decimal value .625. In binary, this is .101; that is, .5 plus .125. The value is normalized as it is, the exponent is 0, the sign is positive, 0.

```
1000 0000 0010 0000 ...
  8    0  s 2    0  ...
```

Negative 5. In binary, 5 is 101. Before taking the twos-complement, we supply a leading zero which will become the negative sign bit: 0101. The twos-complement is 1011. Removing the sign bit, 011. Normalizing, .1100 with the exponent -2. In excess 80, -2 is 7E. Result:

```
0111 1110 1100 0000 ...
  7    E  s C    0  ...
```

Zero, the exception. This is an all zero value.

```
0000 0000 0000 0000 ...
  0    0    0    0  ...
```

All zeros can be thought of as zero by convention. Otherwise, it would represent the smallest positive number possible in the scheme.

VALUES IN MEMORY

As with other values in 8086 memory, floating point values are stored "back-words." The least signficant 16-bit word is stored first, then the next, and so forth. If the single-precision value 84740000 is stored at location x, it will show as follows when displaying memory contents:

```
x       0000
x + 2   8474
```

However, long integers are stored in order. The long integer with a hexadecimal value of 128A34BF will show as:

```
x       128A
x + 2   34BF
```

Appendix I

LIST OF XENIX UTILITIES

The following pages present a list of the typical utilities in a XENIX system. You can obtain a list of your system's utilities by entering:

```
cd /<CR>
ls -FCR | lpr<CR>
```

LIST OF XENIX UTILITIES

```
bin/            etc/            lost+found/  pribootfd   xenix*
boot*           lib/            maint/       tmp/        xenix.fd*
dev/            load.hd         priboot      usr/

./bin:
adb*            dd*             kill*        passwd*     sum*
ar*             deroff*         l*           pr*         sync*
as*             df*             ld*          ps*         tar*
at*             diff*           lint*        pwd*        test*
basename*       du*             ln*          ranlib*     time*
calendar*       echo*           login*       rm*         true*
cat*            ed*             lpr*         rmail*      tsort*
cc*             edit*           ls*          rmdir       uucp*
chgrp*          egrep*          mail*        sed*        uulog*
chmod*          ex*             make*        sh*         uux*
chown*          expr*           mesg*        size*       vi*
cmp*            false*          mkdir*       sleep*      wc*
comm*           file*           mv*          sort*       who*
cp*             find*           nice*        split*      write*
csh*            flagbad*        nm*          strip*
cu*             fsck*           nroff*       stty*
date            grep*           od*          su*

./dev:
console         hd0.boot        mem          rhd0.layout swap
cua0            hd0.layout      null         rhd0.roc0   tar
cul0            hd0.roc0        plp          rhd0.secmap tty
f80             hd0.secmap      rfd0.boot    rhd0.spares tty2
fd0.boot        hd0.spares      rfd0.dd      rhd0.track0 tty3
fd0.dd          hd0.track0      rfd0.root    rhd0a       tty4
fd0.root        hd0a            rfd0.sd      rhd0b       tty5
fd0.sd          hd0b            rfd0.swap    root        tty6
fd0.swap        kmem            rhd0         rroot       tty7
hd0             lp              rhd0.boot    rswap       tty8

./etc:
asktime*        group           motd         systemid    utmp
checklist       haltsys*        mount*       termcap     wall*
cron*           inir*           mtab         ttys
dial_login*     init*           passwd       ttytype
dmesg*          mkfs*           rc*          umount*
getty*          mknod*          shutdown*    update*

./lib:
c0*             c2*             crt0.o       libcurses.a libmp.a
c1*             cpp*            libc.a       libm.a      libtermlib.a

./lost+found:

./maint:
```

LIST OF XENIX UTILITIES (Continued)

```
./tmp:

./usr:
adm/            bin/            lib/            tmp/
altos/          include/        spool/

./usr/adm:
messages        msgbuf          wtmp

./usr/altos:
.login          qa.test*

./usr/bin:
Mail*           msgs*           printenv*
more*           nohup*          reset*

./usr/include:
a.out.h         errno.h         olda.out.h      stddef.h        utmp.h
ar.h            execargs.h      olddump.h       stdio.h         varargs.h
assert.h        grp.h           pack.h          symbol.h        whoami.h
core.h          ident.h         psout.h         sys/            xout86.h
ctype.h         local/          pwd./h          sys.s
curses.h        math.h          setjmp.h        sysexits.h
dk.h            mp.h            sgtty.h         time.h
dumprestor.h    mtab.h          signal.h        tp_defs.h

./usr/include/local:
layout.h        sspare.h        uparm.h

./usr/include/sys:
acct.h          locking.h       pk.p            systm.h         user.h
charg.h         mpx.h           proc.h          timeb.h
dir.h           mx.h            reg.h           times.h
filsys.h        param.h         sites.h         tty.h
inode.h         pk.h            stat.h          types.h

./usr/lib:
Mail.help       crontab         lintl*          llib-port       uucp/
Mail.help.~     ex2.13preserve* lint2*          lpd*
atrun*          ex2.13recover*  llib-lc         more.help
calendar*       ex2.13strings   llib-lm         tmac/

./usr/lib/tmac:
tmac.an         tmac.help       tmac.s          tmac.sdisp      tmac.srefs
tmac.e          tmac.r          tmac.scover     tmac.skeep

./usr/lib/uucp:
L-devices       L.sys           uucico*         uuxqt*
L-dialcodes     USERFILE        uuclean*
```

LIST OF XENIX UTILITIES (Continued)

```
./usr/spool:
at/          lpd/          mail/       uucp/

./usr/spool/at:
lasttimedone past/

./usr/spool/at/past:

./usr/spool/lpd:

./usr/spool/mail:

./usr/spool/msgs:

./usr/spool/uucp:
.XQTDIR/     LOGFILE       SYSLOG

./usr/spool/uucp/.XQTDIR:

./usr/tmp:
```

Appendix J

8086 ASSEMBLY LANGUAGE REFERENCE MANUAL

The following pages represent an 8086 Assembly Language Reference Manual extracted with permission from a Microsoft, Inc. publication. The section and page numbers of this excerpt reflect the enumeration and pagination of the original publication.

2.5 AS: The XENIX Assembler

This document describes the usage and input syntax of the XENIX 8086 assembler as. As is an assembler that produces an output file containing relocation information and a complete symbol table. The output is acceptable to the XENIX loader ld, which may be used to combine the outputs of several assembler runs and to obtain object programs from libraries. The output format has been designed so that if a program contains no unresolved references to external symbols, it is executable without further processing.

2.5.1 Usage

As is invoked as follows:

as [-l] [-o output] file

If the optional `-l' argument is given, an assembly listing is produced which includes the source, the assembled (binary) code, and any assembly errors.

The output of the assembler is by default placed on the file a86.out in the current directory; The `-o' flag causes the output to be placed on the named file.

2.5.2 Lexical conventions

Assembler tokens include identifiers (alternatively, ``symbols'' or ``names''), constants, and operators.

2.5.2.1 Identifiers An identifier consists of a sequence of alphanumeric characters (including period ``.''and underscore ``_'' as alphanumeric) of which the first may not be numeric. Only the first eight characters are significant. The case of alphabetics in identifiers is significant.

2.5.2.2 Constants A hex constant consists of a sequence of digits and the letters `a', `b', `c', `d', `e', and `f' (any of which may be capitalized), preceeded by the character `/'. The letters are interpreted with the following values:

HEX	DECIMAL
A	10
B	11
C	12
D	13
E	14
F	15

An octal constant consists of a series of digits, preceded by the tilde character `` ''. The digits must be in the range from 0 to 7.

A decimal constant consists simply of a sequence of digits. The magnitude of the constant should be representable in 15 bits; i.e., be less than 32,768.

2.5.2.3 Blanks Blank and tab characters may be freely interspersed between tokens, but may not be used within tokens (except in character constants). A blank or tab is required to separate adjacent identifiers or constants not otherwise separated.

2.5.2.4 Comments The character ``|'' introduces a comment, which extends through the end of the line on which it appears. Comments are ignored by the assembler.

2.5.3 Segments

Assembled code and data fall into three segments: the text segment, the data segment, and the bss segment. The text segment is the one in which the assembly begins, and it is the one into which instructions are typically placed. The XENIX system will, if desired, enforce the purity of the text segment of programs by trapping write operations into it. Object programs produced by the assembler must be processed by the link-editor ld (using its `-i' flag) if the text segment is to be write-protected. A single copy of the text segment is shared among all processes executing such a program.

The data segment is available for placing data or instructions which will be modified during execution. Anything which may go in the text segment may be put into the data segment. In programs with write-protected, sharable text segments, the data segment contains the initialized but variable parts of a program. If the text segment is not pure, the data segment begins immediately after the text segment. If the text segment is pure, the

data segment is in an address space of its own, starting at location zero (0).

The bss segment may not contain any explicitly initialized code or data. The length of the bss segment (like that of text or data) is determined by the high-water mark of the location counter within it. The bss segment is actually an extension of the data segment and begins immediately after it. At the start of execution of a program, the bss segment is set to 0. The advantage in using the bss segment for storage that starts off empty is that the initialization information need not be stored in the output file. See also location counter and assignment statements below.

2.5.4 The location counter

The special symbol, ``.'', is the location counter. Its value at any time is the offset within the appropriate segment from the start of the statement in which it appears. The location counter may be assigned to, with the restriction that the current segment may not change; furthermore, the value of ``.'' may not decrease. If the effect of the assignment is to increase the value of ``.'', the required number of null bytes are generated (but see Segments above).

2.5.5 Statements

A source program is composed of a sequence of statements. Statements are separated by new-lines. There are four kinds of statements: null statements, expression statements, assignment statements, and keyword statements.

The format for most 8086 assembly language source statements is:

```
[<label field>]
op-code [<operand field>] [<comment>]
```

Any kind of statement may be preceded by one or more labels.

2.5.5.1 Labels There are two kinds of labels: name labels and numeric labels. A name label consists of a identifier followed by a colon (:). The effect of a name label is to assign the current value and type of the location counter ``.'' to the name. An error is indicated in pass 1 if the name is already defined; an error is indicated in pass 2 if the ``.'' value assigned changes the definition of the

label.

A numeric label consists of a string of digits 0 to 9 and dollar-sign ($) followed by a colon (:). Such a label serves to define local symbols of the form ``n$'', where n is the digit of the label. The scope of the numeric label is the labelled block in which it appears. As an example, the label 9$ is defined only between the lables foobar and foo:

```
foobar:
9$:     .byte 0
        .
        .
        .
foo:    .word a
```

As in the case of name labels, a numeric label assigns the current value and type of ``.'' to the symbol.

2.5.5.2 Null statements A null statement is an empty statement (which may, however, have labels and a comment). A null statement is ignored by the assembler. Common examples of null statements are empty lines or lines containing only a label.

2.5.5.3 Expression statements An expression statement consists of an arithmetic expression not beginning with a keyword. The assembler computes its value and places it in the output stream, together with the appropriate relocation bits.

2.5.5.4 Assignment statements An assignment statement consists of an identifier, an equal sign (=), and an expression. The value and type of the expression are assigned to the identifier. It is not required that the type or value be the same in pass 2 as in pass 1, nor is it an error to redefine any symbol by assignment.

Any external attribute of the expression is lost across an assignment. This means that it is not possible to declare a global symbol by assigning to it, and that it is impossible to define a symbol to be offset from a non-locally defined global symbol.

As mentioned, it is permissible to assign to the location counter ``.''. It is required, however, that the type of the expression assigned be of the same type as ``.'', and it is forbidden to decrease the value of ``.''. In practice,

the most common assignment to ``.'' has the form ``.=.+n'' for some number n; this has the effect of generating n null bytes.

2.5.5.5 Keyword statements Keyword statements are numerically the most common type, since most machine instructions are of this sort. A keyword statement begins with one of the many predefined keywords of the assembler; the syntax of the remainder depends on the keyword. All the keywords are listed below with the syntax they require.

2.5.6 Expressions

An expression is a sequence of symbols representing a value. Its constituents are identifiers, constants, and operators. Each expression has a type.

Arithmetic is two's complement. All operators have equal precedence, and expressions are evaluated strictly left to right.

2.5.6.1 Expression operators The operators are:

Operator	Description
(blank)	same as +
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Logical OR
&	Logical AND
!	Logical NOT
>	Right Shift
<	Left Shift

2.5.6.2 Types The assembler deals with expressions, each of which may be of a different type. Most types are attached to the keywords and are used to select the routine which treats that keyword. The types likely to be met explicitly are:

undefined
: Upon first encounter, each symbol is undefined. It may become undefined if it is assigned an undefined expression.

undefined external

A symbol which is declared .globl but not defined in the current assembly is an undefined external. If such a symbol is declared, the link editor ld must be used to load the assembler's output with another routine that defines the undefined reference.

absolute

An absolute symbol is defined ultimately from a constant. Its value is unaffected by any possible future applications of the link-editor to the output file.

text

The value of a text symbol is measured with respect to the beginning of the text segment of the program. If the assembler output is link-edited, its text symbols may change in value since the program need not be the first in the link editor's output. Most text symbols are defined by appearing as labels. At the start of an assembly, the value of ``.'' is text 0.

data

The value of a data symbol is measured with respect to the origin of the data segment of a program. Like text symbols, the value of a data symbol may change during a subsequent link-editor run since previously loaded programs may have data segments. After the first .data statement, the value of ``.'' is data 0.

bss

The value of a bss symbol is measured from the beginning of the bss segment of a program. Like text and data symbols, the value of a bss symbol may change during a subsequent link-editor run, since previously loaded programs may have bss segments. After the first .bss statement, the value of ``.'' is bss 0.

external absolute, text, data, or bss

Symbols declared .globl but defined within an assembly as absolute, text, data, or bss symbols may be used exactly as if they were not declared .globl; however, their value and type are available to the link editor so that the program may be loaded with others that reference these symbols.

other types
: Each keyword known to the assembler has a type which is used to select the routine which processes the associated keyword statement. The behavior of such symbols when not used as keywords is the same as if they were absolute.

2.5.6.3 Type propagation in expressions When operands are combined by expression operators, the result has a type which depends on the types of the operands and on the operator. The rules involved are complex to state but were intended to be sensible and predictable. For purposes of expression evaluation the important types are

undefined
absolute
text
data
bss
undefined external
other

The combination rules are then: If one of the operands is undefined, the result is undefined. If both operands are absolute, the result is absolute. If an absolute is combined with one of the `other types'mentioned above, the result has the other type. If two operands of `other type' are combined, the result has the numerically larger type. An `other type' combined with an explicitly discussed type other than absolute acts like an absolute.

Further rules applying to particular operators are:

\+ If one operand is text-, data-, or bss-segment relocatable, or is an undefined external, the result has the postulated type and the other operand must be absolute.

\- If the first operand is a relocatable text-, data-, or bss-segment symbol, the second operand may be absolute (in which case the result has the type of the first operand); or the second operand may have the same type as the first (in which case the result is absolute). If the first operand is external undefined, the second must be absolute. All other combinations are illegal.

others
: It is illegal to apply these operators to any but absolute symbols.

2.5.7 Pseudo-operations

The keywords listed below introduce statements that influence the later operations of the assembler. The metanotation

[stuff] ...

means that 0 or more instances of the given stuff may appear. Also, boldface tokens are literals, italic words are substitutable.

2.5.7.1 .even If the location counter ``.'' is odd, it is advanced by one so the next statement will be assembled at a word boundary. This is useful for forcing storage allocation to be on a word boundary after a .byte or .ascii directive.

2.5.7.2 .float, .double

.float 31459E4

The .float psuedo operation accepts as its operand an optional string of tabs and spaces, then an optional sign, then a string of digits optionally containing a decimal point, them an optional `e' or `E', followed by an optionally signed integer. The string is interpreted as a floating point number. The difference between .float and .double is in the number of bytes for the result; .float sets aside four bytes, while .double sets aside eight bytes.

2.5.7.3 .B .globl

.globl *name* [, *name*] ...

This statement makes the *names* external. If they are otherwise defined (by assignment or appearance as a label) they act within the assembly exactly as if the .globl statement were not given; however, the link editor *ld* may be used to combine this routine with other routines that refer to these symbols.

Conversely, if the given symbols are not defined within the current assembly, the link editor can combine the output of this assembly with that of others which define the symbols. It is possible to force the assembler to make all otherwise undefined symbols external.

2.5.7.4 .text, .data, .bss These three pseudo-operations cause the assembler to begin assembling into the text, data, or bss segment respectively. Assembly starts in the text segment. It is forbidden to assemble any code or data into the bss segment, but symbols may be defined and ``.'' moved about by assignment.

2.5.7.5 .comm The format of the .comm is:

.comm ARRAY

Provided the name is not defined elsewhere, this statement is equivalent to .globl. That is, the type of name is ``undefined external'', and its size is expression. In fact the name behaves in the current assembly just like an undefined external. However, the link-editor ld has been special-cased so that all external symbols which are not otherwise defined, and which have a non-zero value, are defined to lie in the bss segment, and enough space is left after the symbol to hold expression bytes. All symbols which become defined in this way are located before all the explicitly defined bss-segment locations.

2.5.7.6 .insrt The format of a .insrt is:

.insrt "filename"

where filename is any valid XENIX filename. Note that the filename must be enclosed within double quotes.

The assembler will attempt to open this file for input. If it succeeds, source lines will be read from it until the end of file is reached. If as was unable to open the file, a Cannot open insert file error message will be generated.

This statement is useful for including a standard set of comments or symbol assignments at the beginning of a program. It is also useful for breaking up a large source program into easily managable pieces.

A maximum depth of 10 (ten) files may be .insrted at any one time.

System call names are not predefined. They may be found in the file /usr/include/sys.s.

2.5.7.7 .ascii, .asciz The .ascii directive translates character strings into their 7-bit ascii (represented as 8-bit bytes) equivalents for use in the source program. The format of the .ascii directive is as follows:

```
.ascii      /character string/
```

where

character string contains any character valid in a character constant. Obviously, a <newline> must not appear within the character string. (It can be represented by the escape sequence \en).

/ and / are delimiter characters, which may be any character not appearing in character string

Several examples follow:

```
   Hex Code Generated:                    Statement:
 22 68 65  6C 6C 6F 20 74    .ascii  /"hello there"/
 68 65 72 65 22
 77 61 72 6E 69 6E 67 20     .ascii "Warning-\007\007 \n"
 2D 07 07 20 0A
 61 62 63 64 65 66 67        .ascii *abcdefg*
```

The .asciz directive is equivalent to the .ascii directive with a zero (null) byte automatically inserted as the final character of the string. Thus, when a list or text string is to be printed, a search for the null character can terminate the string. Null terminated strings are used as arguments to some XENIX system calls.

2.5.7.8 .list, .nlist These pseudo-directives control the assembler output listing. These, in effect, temporarily override the `-l' switch to the assembler. This is useful when certain portions of the assembly output is not necessarily desired on a printed listing.

```
.list     turns the listing on
.nlist    turns the listing off
```

2.5.7.9 .blkb, .blkw The .blkb and .blkw directives are used to reserve blocks of storage: .blkb reserves bytes, .blkw reserves words.

The format is:

```
.blkb       [expression]
.blkw       [expression]
```

where expression is the number of bytes or words to reserve. If no argument is given a value of 1 is assumed. The expression must be absolute, and defined during pass 1.

This is equivalent to the statement ``.=.+expression'', but has a much more transparent meaning.

2.5.7.10 .byte, .word The .byte and .word directives are used to reserve bytes and words and to initialize them with certain values.

The format is:

```
.byte       [expression]
.word       [expression]
```

The .byte directive reserves one byte for each expression in the operand field and initializes the value of the byte to be the low-order byte of the corresponding expression.

For example,

```
.byte 0
                reserves an byte, with a value
                of zero.
state: .byte 0
                reserves a byte with a zero
                value called state.
```

The semantics for .word are identical, except that 16-bit words are reserved and initialized.

2.5.7.11 .end The .end directive indicates the physical end of the source program. The format is:

```
.end        [expression]
```

where expression is an optional argument which, if present, indicates the entry point of the program, i.e. the starting point for execution. If the entry point of a program is not specified during assembly, it defaults to zero.

Every source program must be terminated with a .end statement. Inserted files which contain a .end statement will terminate assembly of the entire program, not just the inserted portion.

2.5.8 Machine

The 8086 instructions treat different types of operands uniformly. Nearly every instruction can operate on either byte or word data. In the table that follows, with some notable exceptions, an instruction that operates on a byte operand will have a b suffix on the opcode.

The 8086 instruction mnemonics which follow are implemented by the Microsoft 8086 assembler described in this document. Some of the opcodes are not found in any other 8086 manual.

For example, this document describes branch instructions not found in any 8086 manual. The branch instructions expand into a jump on the inverse of the condition specified, followed by an an unconditional intra-segment jump. The operand field format for the branch opcodes is the same as the operand field for the jump long opcodes. The opcodes which are implemented only in this assembler will be annotated by an asterisk, and will be fully defined and described in this document.

8086 Assembler Opcodes

Opcode	Description
aaa	ascii adjust for addition
aad	ascii adjust for division
aam	ascii adjust for multiply
aas	ascii adjust for subtraction
adc	add with carry
adcb	add with carry
add	add
addb	add
and	logical AND
andb	logical AND
*beq	long branch equal
*bge	long branch grt or equal
*bgt	long branch grt
*bhi	long branch on high
*bhis	long branch high or same
*ble	long branch les or equal
*blo	long branch on low
*blos	long branch low or same
*blt	long branch less than
*bne	long branch not equal
*br	long branch
call	intra segment call
calli	inter segment call
cbw	convert byte to word
clc	clear carry flag
cld	clear direction flag
cli	clear interrupt flag
cmc	complement carry flag
cmp	compare
cmpb	compare
cmps	compare string
cmpsb	compare string
cwd	covert word to double word
daa	decimal adjust for addition
das	decimal adjust for subtraction
dec	decrement by one
decb	decrement by one
div	divison unsigned
divb	divison unsigned
hlt	halt
idiv	integer division
idivb	integer division
imul	integer multiplication
imulb	integer multiplication
in	input byte
inc	increment by one
incb	increment by one
int	interrupt

into	interrupt if overflow
inw	input word
iret	interrupt return
j	short jump
ja	short jump if above
jae	short jump if above or equal
jb	short jump if below
jbe	short jump if below or equal
jcxz	short jump if CX is zero
je	short jump on equal
jg	short jump on greater than
jge	short jump greater than or equal
jl	short jump on less than
jle	short jump on less than or equal
jmp	jump
jmpi	inter segment jump
jna	short jump not above
jnae	short jump not above or equal
jnb	short jump not below
jnbe	short jump not below or equal
jne	short jump not equal
jng	short jump not greater
jnge	short jump not greater or equal
jnl	short jump not less
jnle	short jump not less or equal
jno	short jump not overflow
jnp	short jump not parity
jns	short jump not sign
jnz	short jump not zero
jo	short jump on overflow
jp	short jump if parity
jpe	short jump if parity even
jpo	short jump if parity odd
js	short jump if signed
jz	short jump if zero
lahf	load AH from flags
lds	load pointer using DS
lea	load effective address
les	load pointer using ES
lock	lock bus
lodb	load string byte
lodw	load string word
loop	loop short label
loope	loop if equal
loopne	loop if not equal
loopnz	loop is not zero
loopz	loop if zero
mov	move
movb	move byte
movs	move string
movsb	move string byte

mul	multipication unsigned
mulb	multipication unsigned
neg	negate
negb	negate
nop	no op
not	logical NOT
notb	logical NOT
or	Iogical OR
orb	logical OR
out	output byte
outw	output word
pop	pop from stack
popf	pop flag from stack
push	push onto stack
pushf	push flags onto stack
rcl	rotate left through carry
rclb	rotate left through carry
rcr	rotate right throuch carry
rcrb	rotate right throuch carry
rep	repeat string operation
repnz	repeat string operation not zero
repz	repeat string operation while zero
ret	return from procedure
reti	return from intersegment procedure
rol	rotate left
rolb	rotate left
ror	rotate right
rorb	rotate right
sahf	store AH into flagsno operands
sal	shift arithmetic left
salb	shift arithmetic left
sar	shift arithmetic right
sarb	shift arithmetic right
sbb	subtract with borrow
sbbb	subtract with borrow
scab	scan string
shl	shift logical left
shlb	shift logical left
shr	shidr logical right
shrb	shidr logical right
stc	set carry flag
std	set direction flag
sti	set interrupt enable flag
stob	store byte string
stow	store word string
sub	subtraction
subb	subtraction
test	test
testb	test
wait	wait while TEST pin
xchg	exchange

```
xchgb     exchange
xlat      translate
xor       xclusive OR
xorb      xclusive OR
```

2.5.9 Addressing Modes

The 8086 assembler provides many different ways to access instruction operands. Operands may be contained in registers, within the instruction itself, in memory, or in I/O ports. In addition, the addresses of memory and I/O port operands can be calculated in several different ways.

2.5.9.1 Register Operands Instructions that specify only register operands are generally the most compact and fastest executing of all the instruction forms. This is because the register `addresses' are encoded in the instructions with just a few bits, and because these operations are performed entirely within the CPU. Registers may serve as source operands, destination operands, or both.

EXAMPLES OF REGISTER ADDRESSING

```
sub       cx,di
mv        ax,/3*4
mv        /3*4/,ax
mov       ax,*1
```

2.5.9.2 Immediate Operands Immediate operands are constant data contained in an instruction. The data may be either 8 or 16 bits in length. Immediate operands can be accessed quickly because they are available directly from the instruction queue; it is possible that no bus cycles will be needed to obtain an immediate operand. An immediate operand is always a constant value and can only be used as a source operand.

The assembler can accept both 8 and 16 bit operands. It does not perform any checking on the operand size, but determines the size of the operand by the following symbols:

```
*expr     an 8 bit immediate
#expr     a 16 bit immediate
```

EXAMPLES OF IMMEDIATE ADDRESSING

```
mov     cx,*PAGSIZ/2
mov     cx,#PAGSIZ/2
mov     map,#PAGSIZ/2
mov     map,*PAGSIZ/2
```

2.5.10 Memory Addressing Modes

When reading or writing a memory operand, a value called the offset is required. This offset value, also called the effective address is the operand's distance in bytes from the beginning of the segment in which it resides.

2.5.10.1 Direct Addressing Direct addressing is the simplest memory addressing mode since no registers are involved. The effective address is taken directly from the displacement field of the instruction. It is typically used to access simple (scalar) variables.

EXAMPLES OF DIRECT ADDRESSING

```
push    *6(bp)
mov     cx,#256
add     si,*4
```

2.5.10.2 Register Indirect Addressing The effective address of a memory operand may be taken from a base or index register. One instruction can operate on many different memory locations if the value in the base or index register is updated appropriately. Indirect addressing is denoted by an ampersand @ preceding the operand.

EXAMPLES OF INDIRECT ADDRESSING

```
popl    rr0,@r15
calli   @moncall
```

2.5.10.3 Based Addressing In based addressing, the effective address is the sum of a displacement value and the content of register bx or bp. Based addressing also provides a straightforward way to address structures which may be located in different places in memory. A base register can be pointed at the base of the structure and elements of the structure addressed by their displacements from the base. Different copies of the same structure can be accessed by simply changing the base register.

EXAMPLE OF BASED ADDRESSING

```
mov         *2(si),#/1000
```

2.5.10.4 Indexed Addressing In indexed addressing, the effective address is calculated from the sum of a displacement plus the content of an index register. Indexed addressing often is used to access elements in an array. The displacement locates the beginnning of the array, and the value of the index register selects one element. Since all array elements are the same length, simple arithmetic on the index register will select any element.

EXAMPLE OF INDEXED ADDRESSING

```
mov         #_cat,(bx)
```

2.5.10.5 Based Indexed Addressing Based indexed addressing generates an effective address that is the sum of a base register, an index register, and a displacement. Based indexed addressing is a very flexible mode because two address components can be varied at execution time.

Based indexed addressing provides a convenient way for a procedure to address an array allocated on a stack. Register bp can contain the offset of a reference point on the stack, typically the top of the stack after the procedure has saved registers and allocated local storage. The offset of the beginning of the array from the reference point can be expressed by a displacement value, and an index register can be used to access individual array elements.

EXAMPLES OF BASED INDEXED ADDRESSING

```
mov         (bx)(dx),_sym
mov         *2(bx)(dx),_sym
mov         #2(bx)(dx),_sym
```

2.5.11 Diagnostics

When syntactic errors occur, the line number and the file in which they occur is displayed. Errors in pass 1 cause cancellation of pass 2.

```
***ERROR*** syntax error, line xx
file: vv errors
```

where xx represents the line number(s) in error, and yy represents the total number of errors.