A/UX® 2.0.1 Release Notes With MacX® 1.1 Release Notes

Final Draft

nnn-nnnn

nnn-nnnn

2/9/90 A/UX Warranty
(follows copyright page)

Table of Contents

About A/UX 2.0.1 / 1

Installing and updating A/UX / 3

Using A/UX 2.0.1 with your Macintosh IIsi / 3

New capabilities / 4

- MacX / 5
- Mouse control settings / 5
    - Adjusting the tracking speed of the mouse / 5
    - Adjusting the double-click rate of the mouse / 6
- Multiple monitor configurations / 7
    - Designating a main monitor / 8
    - Changing the location of the Login dialog box and startup screens / 9

Feature enhancements / 10

- New version of the awk programming language / 10
- New version of the Korn shell / 10
- ANSI C header files / 11
- Asynchronous serial support / 11
- CommandShell function keys / 11
- Mounting CDs / 11
- Macintosh System Software version 6.0.7 / 12
- POSIX 1003.1-1990 / 12
- Version 5.65 of sendmail / 12
- Known problems / 12
    - Shared libraries / 12
    - Device drivers / 13
    - Printing / 13

About the MacX 1.1 Release Notes / 15
About MacX 1.1 / 16
New MacX 1.1 features / 16
Copying images to the Macintosh Clipboard / 16
Compiling directories of BDF files / 18
Displaying nonrectangular windows / 23
MacX 1.1 hints / 24
Low memory warnings / 24
Displaying client applications on color screens / 25
Optimizing client application performance on multiple monitors / 26
Running foreign window managers / 27
Client applications can't connect to MacX / 27
Displaying motion-sensitive client applications / 28
MacX 1.1 limitations / 28
Dimmed connection tools in the Host dialog box / 28
Conflicts between the X11 protocol and Macintosh keyboard characters / 29
Macro software incompatibility / 29
Conflicts between xclipboard and the Macintosh Clipboard / 30
Closing MacX while client applications are active / 30
Limits to color-table animation / 31
Client applications appear "grainy" on color screens / 31
Problems displaying sample text in Font Director window / 31

Reference manual update / 33
Errata / 34

About A/UX 2.0.1

A/UX® version 2.0.1 is the latest version of the A/UX operating system that was specifically designed to run on the new Macintosh® IIsi, as well as all other members of the Macintosh II family and the Macintosh SE/30. A/UX 2.0.1 also includes several performance enhancements and even a few new features such as MacX, an X window system for A/UX.

These release notes are an adjunct to the existing A/UX 2.0 documentation and were designed to update the documentation to reflect the current state of the software. You may place them after the A/UX 2.0 release notes at the end of the *A/UX Installation Guide.*

The *MacX 1.1 Release Notes* are bundled with these release notes. You may want to store them in the *Apple MacX* Binder for easy reference. For additional information about MacX and the X Window System, see the *MacX User's Guide*, and *MacX for A/UX Supplement* included with this kit.

For additional information on any of the topics covered in these release notes, see the reference manual pages at the end of this document, or consult the on-line version of the manual page by typing
`man` *command-name.*

The A/UX 2.0.1 release notes give you information on

- installing and updating A/UX
- using A/UX 2.0.1 with your Macintosh IIsi
- inclusion of MacX™ in A/UX
- adjusting the mouse control settings and configuring a multiple monitor setup
- new versions of the `awk` programming language and Korn shell
- enhancements to ANSI C header files

- asynchronous serial support in A/UX
- using CommandShell function keys
- using CDs with A/UX
- The latest versions of Macintosh System Software, POSIX, and `sendmail`
- MacX 1.1 Release Notes
- Reference manual update

Installing and updating A/UX

There are two separate releases of A/UX 2.0.1: one for users who already own A/UX 2.0 and are updating to version 2.0.1, and another for new customers who are purchasing A/UX for the first time.

If you are updating from A/UX 2.0, you received the Incremental Update Kit which includes the *A/UX 2.0.1 Incremental Update Guide.* Use this guide to install A/UX 2.0.1 on your existing system.

◆ *Note:* To install A/UX 2.0.1 on a Macintosh IIsi you must follow special installation instructions. See *Running A/UX 2.0.1 on a Macintosh IIsi* in the Incremental Update Guide for more information.

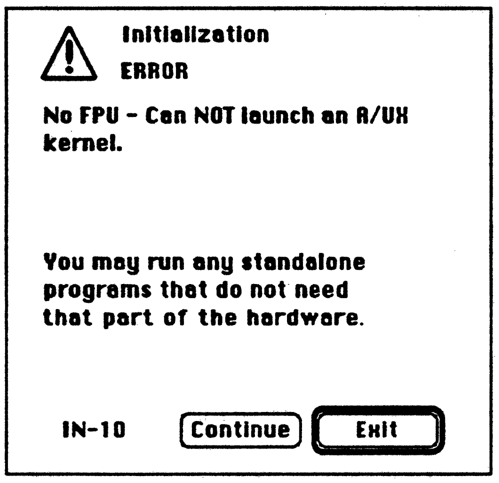
If you are installing A/UX for the first time–that is, you are not updating from version 2.0– follow the installation instructions in the *A/UX Installation Guide* which came with the kit.

◆ *Note:* The HDSC Setup utility is no longer found on the System Software disk. It is now located on the Utilities 1 floppy disk.

Using A/UX 2.0.1 with your Macintosh IIsi

The Macintosh IIsi is the newest member of the Macintosh II family to run A/UX 2.0.1. However, you must have a floating-point unit (FPU) installed on your Macintosh IIsi in order to run A/UX 2.0.1. An FPU is built onto both the NuBus and 030 Direct Slot adapter cards. If you do not have an FPU installed, A/UX will display the dialog box shown in Figure 1.

■ **Figure 1** No FPU warning dialog box



Initialization
ERROR

No FPU - Can NOT launch an A/UX kernel.

You may run any standalone programs that do not need that part of the hardware.

IN-10

See your authorized Apple™ dealer for information on installing a NuBus or 030 Direct Slot adapter card.

In addition to the capacity to run A/UX 2.0.1, the Macintosh IIsi offers several other features, such as built-in video support, a single expansion slot that allows you to attach a NuBus or an 030 Direct Slot expansion card, and keyboard-based interrupt and reset switches that replace the old programmer's switch.

◆ *Note:* The Macintosh IIsi also has a built-in sound input feature; however, this feature is not currently supported in A/UX 2.0.1.

New capabilities

In addition to Macintosh IIsi support, A/UX 2.0.1 offers numerous enhanced features and new capabilities. The following section describes the new capabilities of A/UX.

MacX

One of the most significant additions to A/UX 2.0.1 is the inclusion of MacX. MacX is the Apple version of the X Window System, which allows users of the UNIX operating system to display multiple X applications in windows. It also allows you to integrate different types of computers transparently on a network, and to run applications on other computers while displaying them on your computer screen. MacX is located in the `/mac/macx` directory.

Mouse control settings

You can use the Control Panel in A/UX to change the setting of the mouse, just as you can in the Macintosh Operating System. The following section describes how to adjust the mouse tracking speed and the double-click rate.

Adjusting the tracking speed of the mouse

When you start up your Macintosh for the first time, the mouse tracking speed is set to slow. This means that the mouse and pointer move equal distances. For example, you must move the mouse eight inches to move the pointer eight inches on the screen. Some users may find this speed cumbersome. To adjust the tracking speed of the mouse, follow these steps:

1. **Log in to A/UX.**

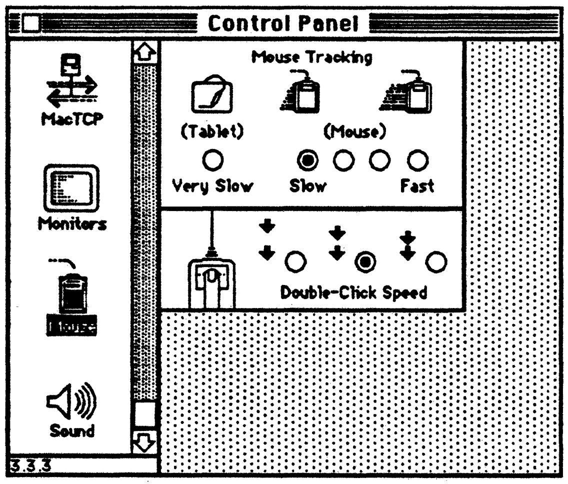
   The integration of the Macintosh environment into A/UX is so seamless that from time to time it may be difficult to determine which operating system you are working in. When you make any changes using the Control Panel, it is important that you are running A/UX when you make them, otherwise you may accidentally make changes in the Macintosh environment. This isn't harmful, but may cause some confusion.

△ **Tip** A simple way to verify if you are indeed running A/UX is to pull down the Apple menu at the far left of the menu bar and check for a menu command named CommandShell. The CommandShell menu item is unique to A/UX and you will only see it when you are in the A/UX environment. △

2. **Choose Control Panel from the Apple menu.**
3. **Click the Mouse icon from the scrolling list on the left.**

   Use the scroll bar if necessary to bring the Mouse icon into view. You see the dialog box shown in Figure 2.

■ **Figure 2** Mouse Control Panel settings



4. **Click any of the five options to set the mouse tracking speed.**
5. **Click the close box.**

Adjusting the double-click rate of the mouse

The Mouse control settings in the Control Panel also allow you to set the speed at which a double click is executed. To adjust the double-click rate of the mouse, follow these steps:

1. **Log in to A/UX.**
2. **Choose Control Panel from the Apple menu.**
3. **Click the Mouse icon from the scrolling list on the left.**
   Use the scroll bar if necessary to bring the Mouse icon into view.

4. **Click one of the three buttons.**

   You will see three options. The first requires two slow, discernable clicks to execute a double click. The second option requires less time between clicks to execute, and the third option requires two rapid clicks to execute. Select one of the options to set the double-click rate.

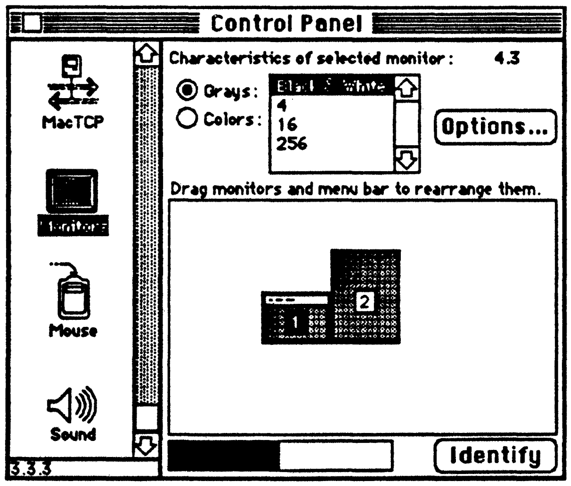
5. **Click the close box.**

   You can test the double-click rate by double-clicking an item on the desktop. If you wish, you may return to the Control Panel and readjust the double-click rate.

Multiple monitor configurations

You may connect more than one monitor to your Macintosh computer running A/UX 2.0.1. Once your system is properly configured, you can use these monitors as one screen so that the mouse pointer moves freely between their boundaries. Figure 3 shows the Monitors Control Panel for more than one monitor.

▪ **Figure 3** Monitors Control Panel



To set a multiple monitor configuration, follow these steps:

1. **Choose Control Panel from the Apple menu.**
2. **Click the Monitors icon from the scrolling list on the left.**
   Use the scroll bar if necessary to bring the Monitors icon into view.
3. **Position each monitor icon according to the actual location of the monitors in your work area.**

   For example, if monitor 2 sits left of monitor 1 in your work area, drag its icon so that it is touching the left border of the icon for monitor 1. Continue to arrange the icons until you've reproduced your entire monitor setup.

   ◆ *Note:* The number of each monitor is determined by the location of the video card or built-in video socket to which the monitor is connected. The monitor numbers remain the same unless you change the location of their video connections. If an icon for each connected monitor does not appear in the Monitors Control Panel, check to see that each monitor is plugged in, that the power is on, and that all cables are securely connected.
4. **Click the close box.**
5. **Choose Logout from the Special menu to implement the new monitor positions.**

Designating a main monitor

You may designate any of the monitors in your arrangement as the main monitor–that is, the monitor that displays the menu bar. To designate the main monitor, complete the following steps:

1. **Choose Control Panel from the Apple menu.**
2. **Click the Monitors icon from the scrolling list on the left.**
   Use the scroll bar if necessary to bring the Monitors icon into view.
3. **Drag the white menu bar from the top of one monitor icon to the icon of the monitor you want to be the main one.**

4. **Choose Logout from the Special menu to implement the changes.**

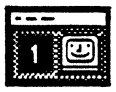
   Your menu bar will now appear on the newly designated monitor, however your Login dialog box, Mouse pointer, and status screens will remain in the original monitor's window until you are fully logged in.

Changing the location of the Login dialog box and startup screens

If you want the Login dialog box and A/UX startup screens to appear in a different monitor, do the following:

1. **Choose Control Panel from the Apple menu.**
2. **Click the Monitors icon from the scrolling list on the left.**
   Use the scroll bar if necessary to bring the Monitors icon into view.
3. **Hold down the Option key.**
   A small happy Macintosh icon appears next to the monitor number as shown in Figure 4.

▪ **Figure 4** The happy Macintosh icon



4. **While holding down the Option key, drag the happy Macintosh icon onto the icon of the Monitor you wish to display your Login screens.**

   The login dialog box and A/UX startup screens will be displayed in the new monitor the next time you log in. For more information on multiple monitor configurations, see the *Macintosh Reference* guide that came with your computer.

Feature enhancements

Several of the features introduced with A/UX version 2.0 have been enhanced with this release. the following section describes these improvements.

New version of the `awk` programming language

A/UX 2.0.1 offers the latest version of the `awk` programming language. There are many new features, the most significant being the addition of user-defined functions. Other enhancements include:

- dynamic regular expressions such as added text substitution and pattern-matching capabilities
- error message improvement
- multiple file input capacity
- command-line argument access

For more detailed information on the new version of `awk`, see the `awk` reference manual page.

New version of the Korn shell

A/UX 2.0.1 has integrated the latest version of the Korn shell. Some of the new features include

- new pattern-matching capabilities
- a new compound version of the `test` command
- new arithmetic operators
- six new variables: `ERRNO`, `FPATH`, `LINENO`, `OPTARG`, `OPTIND`, and `PS4`
- new options added to built-in commands

For more information about the new version of the Korn shell, see the `ksh` reference manual page.

ANSI C header files

A/UX 2.0.1 includes changes to the A/UX header files that provide function prototypes and ANSI symbol definitions for third-party ANSI C compilers.

Asynchronous serial support

The A/UX 2.0.1 Serial Manager supports asynchronous read and writes using the Macintosh toolbox calls `PBRead` and `PBWrite`. See *Inside Macintosh Volume II*, page 185 for more information.

CommandShell function keys

A/UX 2.0.1 supports the use of function keys found on the Macintosh Extended Keyboard in both the CommandShell environment and console mode. In addition, A/UX 2.0.1 supports VT100 application keypad mode in the CommandShell environment. It is not, however, supported in console emulator mode. See the `console (7)` reference manual page for more information.

Mounting CDs

While you are logged in, CDs containing an HFS volume will now appear on the desktop when inserted into a CD drive.

▲ **Warning** In order for the CD drive to operate correctly, you should never launch A/UX with a CD in the CD drive. First switch on the CD drive without a CD inserted, then launch A/UX. ▲

Macintosh System Software version 6.0.7

A/UX 2.0.1 includes Macintosh System Software version 6.0.7. This new version of the Macintosh Operating System is required for running A/UX 2.0.1 on the Macintosh IIsi.

POSIX 1003.1-1990

A/UX 2.0.1 supports the latest version of the IEEE POSIX 1003.1 standard. For more information see *A/UX Guide to POSIX, Version 2.0.*

Version 5.65 of `sendmail`

A/UX 2.0.1 includes version 5.65 of the `sendmail` utility. For more information, see the `sendmail` reference manual page.

Known problems

The following section describes the known problems of A/UX 2.0.1.

Shared libraries

There are two new shared libraries: `libc` and `libmac`. If you create binaries on A/UX 2.0.1 and subsequently try to run them on A/UX 2.0, they will not work because the binaries don't exist. If you are going run binaries on A/UX 2.0.1, do not use shared libraries.

Device drivers

If you are developing device drivers using A/UX, you should be aware that the u-dot has decreased. If your driver software references the field `u_phys` or any field located after `u_phys`, you will need to recompile your software with the new header files. See `<sys/user.h>`.

Printing

By default, the commands `psroff` and `enscript` direct output to the Berkeley print spooler `lpr`. You may select the System V print spooler by setting the environment variable `SPOOLER` to `lp`.

About the MacX 1.1 Release Notes

MacX™ 1.1 is a full X Window System, Version 11, Release 4 server implementation with dramatically increased performance. MacX 1.1 comes with A/UX version 2.0.1; you do not need to install MacX separately.

This document presents late-breaking information about MacX 1.1. Read this document before you use MacX 1.1 on your Macintosh® computer running A/UX. It provides information about MacX 1.1, including information about

- new features
- user hints
- limitations

For more information about MacX, see *MacX for A/UX Supplement* and *MacX User's Guide*. You can store these release notes in the *Apple MacX* Binder.

About MacX 1.1

MacX 1.1 is an X Window System display server that allows you to use X client applications on your A/UX Finder™ desktop. MacX 1.1 is based on X Window System, Version 11, Release 4, and includes

- the display server
- the `xcalc` client application

After you install A/UX 2.0.1 on your system, look for MacX 1.1 in the `mac/macx` directory. To start MacX, just double-click the MacX application icon.

For more information about starting MacX and opening X client applications, see the following documentation that comes in the A/UX 2.0.1 kit:

- *MacX for A/UX Supplement*
- *MacX User's Guide*

New MacX 1.1 features

This section presents information about using these new MacX 1.1 features:

- copying and pasting images on your desktop to the Macintosh Clipboard
- compiling directories of font files
- displaying nonrectangular windows

Copying images to the Macintosh Clipboard

With MacX 1.1 you can copy images appearing on your desktop to the Macintosh Clipboard. You can copy an image of a client application, select a specific area on your screen to copy, or copy the entire desktop. You can then paste the image that you copied into other Macintosh and X applications.

Follow these steps to copy an image from your screen to the Clipboard while running MacX.

1. **Open MacX by double-clicking its icon.**
2. **Open a client application.**

   To open a client application, create a remote command or enter a command at a command line in a terminal emulator window.

3. **Choose Copy Screen to Clipboard from the MacX Window menu, or press COMMAND-B.**

   The pointer changes from an arrow to a crosshair pointer. You can perform any of the three tasks listed below to copy an image on your screen to the Clipboard.

   ◆ *Note:* Remember that the Macintosh Clipboard holds only one image at a time. If you copy one image to the Clipboard and then copy a second image, the second image replaces the first.

- **To copy an image of a client application, position the crosshair pointer in a client application's window, and press the mouse button.**

  This copies the contents of the client application window to the Macintosh Clipboard. When you copy an image of a client application to the Clipboard using this method, the image of the client application does not include the Macintosh window borders.

- **To copy an image of the desktop, position the crosshair pointer in the menu bar or on the desktop, and press the mouse button.**

  This copies an image of your entire desktop, including the menu bar, windows, and icons, to the Macintosh Clipboard.

- **To copy a specific area on your screen, position the crosshair pointer on the desktop, press the mouse button, drag a "box" around the area you want to copy, and release the mouse button.**

  This copies an image of the selected area to the Macintosh Clipboard.

You can paste an image from the Macintosh Clipboard into another Macintosh application by using the Paste command or by pressing COMMAND-V. Depending on whether a particular X application offers a method for pasting the `PRIMARY` or `CLIPBOARD` selection, you can also paste an image into an X application.

Compiling directories of BDF files

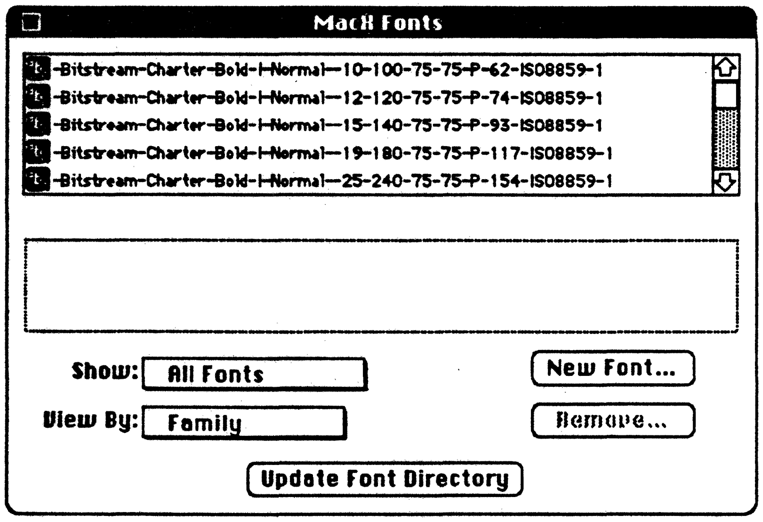
In MacX 1.1 you can compile an entire directory of bitmap distribution format (BDF) files. Using the Font Director window, you choose a folder of font files to compile instead of compiling each font file individually.

After you have installed MacX 1.1, follow these steps to compile a directory of BDF files.

1. **Open MacX by double-clicking its icon.**
2. **Choose Fonts from the Edit menu, or press COMMAND-F.**

   The window shown in Figure 5 appears on your desktop. This is the Font Director window.

■ **Figure 5** The Font Director window



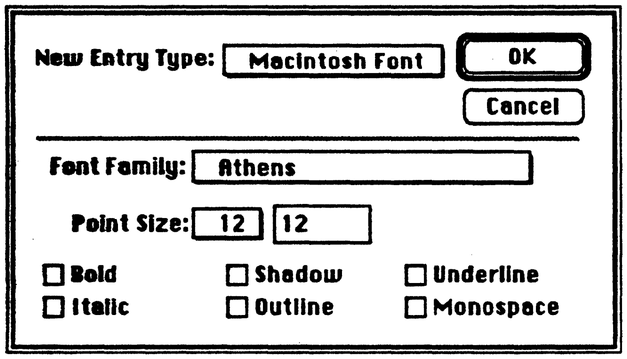
-Bitstream-Charter-Bold-I-Normal--10-100-75-75-P-62-ISO8859-1
-Bitstream-Charter-Bold-I-Normal--12-120-75-75-P-74-ISO8859-1
-Bitstream-Charter-Bold-I-Normal--15-140-75-75-P-93-ISO8859-1
-Bitstream-Charter-Bold-I-Normal--19-180-75-75-P-117-ISO8859-1
-Bitstream-Charter-Bold-I-Normal--25-240-75-75-P-154-ISO8859-1

Show: All Fonts
View By: Family

3. **Click the New Font button in the Font Director window.**

   The dialog box shown in Figure 6 appears.

■ **Figure 6** The New Font dialog box

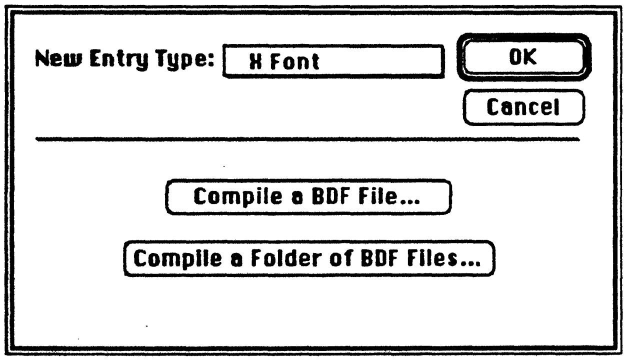


4. Choose X Font from the New Entry Type pop-up menu.

Position the pointer over the New Entry Type box, and press and hold down the mouse button. The New Entry Type menu "pops up," allowing you to choose X Font.

The New Entry dialog box appears, as shown in Figure 7.

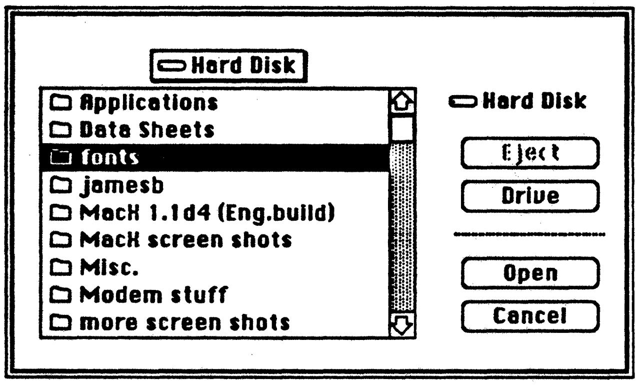
■ **Figure 7** The New Entry dialog box



5. Click Compile a Folder of BDF Files.

The dialog box shown in Figure 8 appears.

- **Figure 8** Finding the folder of BDF files



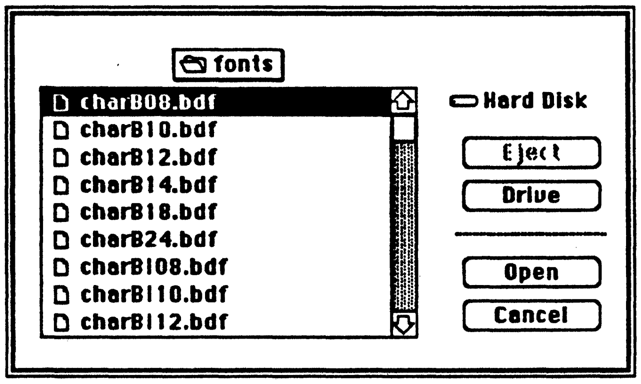
6. **Locate the folder containing the BDF files you want to compile.**

   In Figure 8, the BDF files to be compiled are located in the "fonts" folder on Hard Disk.

7. **Open the folder containing the BDF files by double-clicking.**

   You can also open the folder by selecting it and clicking Open. The dialog box in Figure 9 appears, showing the names of the font files inside the folder.

■ **Figure 9** Selecting a font file



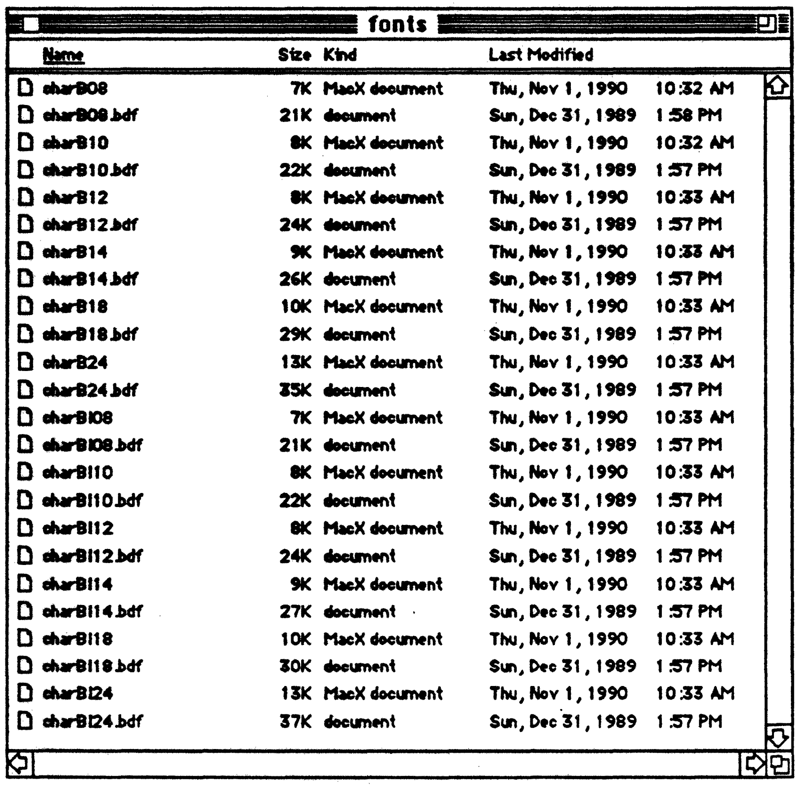
8. **Select any of the font files in the dialog box and click Open.**

   By selecting one of the font files, you are actually specifying that the entire folder of font files be compiled. You see a box that contains a message about compiling the BDF source files you selected. The box disappears after the fonts are compiled.

9. **Click OK in the New Entry dialog box.**

   The New Entry dialog box disappears. The newly compiled fonts appear in their original folder without the suffix `.bdf`. For example, the `charB10.bdf` font files appears in the "fonts" folder as `charB10`. The type of file also changes from "document" to "MacX document." Figure 10 shows the new MacX font names in the "fonts" folder.

■ **Figure 10** MacX font names in the "fonts" folder



Name	Size	Kind	Last Modified	
charB08	7K	MacX document	Thu, Nov 1, 1990	10:32 AM
charB08.bdf	21K	document	Sun, Dec 31, 1989	1:58 PM
charB10	8K	MacX document	Thu, Nov 1, 1990	10:32 AM
charB10.bdf	22K	document	Sun, Dec 31, 1989	1:57 PM
charB12	8K	MacX document	Thu, Nov 1, 1990	10:33 AM
charB12.bdf	24K	document	Sun, Dec 31, 1989	1:57 PM
charB14	9K	MacX document	Thu, Nov 1, 1990	10:33 AM
charB14.bdf	26K	document	Sun, Dec 31, 1989	1:57 PM
charB18	10K	MacX document	Thu, Nov 1, 1990	10:33 AM
charB18.bdf	29K	document	Sun, Dec 31, 1989	1:57 PM
charB24	13K	MacX document	Thu, Nov 1, 1990	10:33 AM
charB24.bdf	35K	document	Sun, Dec 31, 1989	1:57 PM
charBI08	7K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI08.bdf	21K	document	Sun, Dec 31, 1989	1:57 PM
charBI10	8K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI10.bdf	22K	document	Sun, Dec 31, 1989	1:57 PM
charBI12	8K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI12.bdf	24K	document	Sun, Dec 31, 1989	1:57 PM
charBI14	9K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI14.bdf	27K	document	Sun, Dec 31, 1989	1:57 PM
charBI18	10K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI18.bdf	30K	document	Sun, Dec 31, 1989	1:57 PM
charBI24	13K	MacX document	Thu, Nov 1, 1990	10:33 AM
charBI24.bdf	37K	document	Sun, Dec 31, 1989	1:57 PM

10. On the desktop, drag the newly created font files into the MacX Fonts folder on your hard disk.

The MacX Fonts folder is located in the MacX 1.1 folder that you created on your hard disk.

You can also drag the entire folder of new font files into the MacX Fonts folder; the new font files just need to be located somewhere in the directory hierarchy of the MacX Fonts folder.

11. Click the Update Font Directory button in the Font Director window.

MacX updates the Font Directory file. The new MacX font files appear in the Font Director window. You can also quit and restart MacX to update the Font Directory file.

12. **If you're done compiling fonts, close the Font Director window by clicking its close box.**

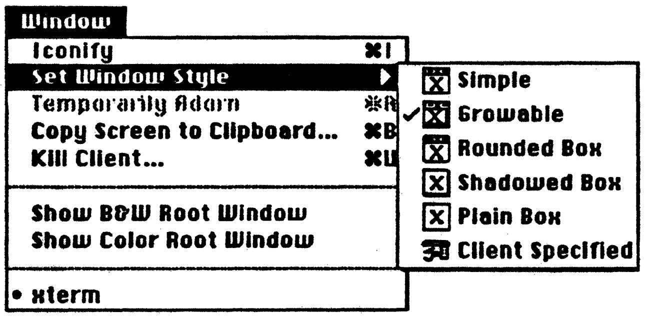
Displaying nonrectangular windows

MacX 1.1 provides support for X Window System, Version 11, Release 4 shaped window extension, allowing client applications to display nonrectangular windows. To implement this feature, MacX 1.1 includes the Client Specified menu item in the Set Window Style pop-up menu. This pop-up menu is a submenu of the Window menu. The Client Specified menu item allows you to specify that a client application determine its own style of window, such as the circular window used for the `oclock` client application.

Figure 11 shows the Window menu and the Set Window Style pop-up menu, which displays the Client Specified option. You can apply the Client Specified option on a per-window basis using the Set Window Style pop-up menu. For example, you display the `oclock` client application on your screen. You click the title bar in the `oclock` window to make it active, and choose Client Specified from the Set Window Style menu. The `oclock` window changes to its own specific style. If you open subsequent client applications, they will *not* appear in their own style of windows unless you select Client Specified as the default window style in the Preferences window.

For more information about setting window styles using the Set Window Style pop-up menu and the Preferences window, see Chapter 4, "Handling Windows," in *MacX User's Guide*.

■ **Figure 11** The Set Window Style pop-up menu



△ **Important** If you are running MacX on A/UX version 2.0 or earlier, the nonrectangular support is not available. Only users of A/UX version 2.0.1 or later are able to take advantage of this feature. △

MacX 1.1 hints

This section presents hints for using MacX 1.1.

Low memory warnings

When performing some operations in MacX, your computer may run low on memory. You can attempt to prevent this problem by increasing MacX's application memory size using the Get Info window. (See Chapter 6, "Troubleshooting," in *MacX Users' Guide.*)

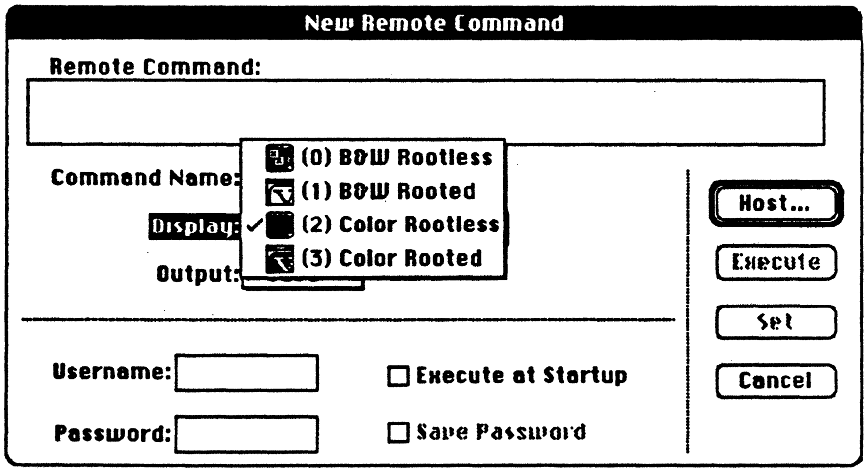
It is not always possible to predict memory requirements before performing some types of operations, so it's important that you not allow memory space to become critically low. If you receive the low-memory warning, indicated by a flashing "caution" symbol in your menu bar, you should take steps to free up memory, as described in Chapter 6, "Troubleshooting," in *MacX User's Guide.* If MacX runs out of application memory, you see an alert box appears. You should save in a MacX settings document any modifications you may have made and restart your computer as soon as possible to avoid a possible system crash or corruption of data.

Displaying client applications on color screens

If you're using MacX 1.1 on a color monitor, you can display client applications in color. Depending on how you open the client application, you can specify its window appear in color using one of two methods.

- If you are using the New Remote Command window, as shown in the Figure 12, you choose (2) Color Rootless or (3) Color Rooted from the Display pop-up menu. For more information about displaying client applications in color, see "Special Uses of the `-display` Option" in Chapter 3 of *MacX User's Guide.* That guide also contains descriptions of MacX rooted and rootless modes.

■ **Figure 12** Choosing a display screen in the New Remote Command window



- If you want to open a client application from the command line in a terminal emulator window, you need to specify the display number at the command line. For example, enter

  `xclock -display` *server*`:0.2`

  to display the `xclock` client application in color rootless mode [(2) Color Rootless]. Type the name of your computer in place of *server* in order to display the client's window on your computer screen.

  To display the `xclock` client application in color rooted mode [(3) Color Rootless], enter

  `xclock -display` *server*`:0.3`

  Type the name of your computer in place of *server* in order to display the client's window on your computer screen.

Optimizing client application performance on multiple monitors

MacX 1.1 allows you to display client applications on multiple monitors connected to your computer. You can display a client application on one monitor and drag its window onto another monitor. A client window can even "straddle" monitor boundaries, appearing on more than one monitor at the same time.

Suppose you have two monitors attached to your Macintosh computer. One monitor is set to a depth of 1 bit per pixel and the other monitor is set to a depth of 8 bits per pixel. You open MacX and create a remote command for the `maze` client application. You specify `maze` to appear on the 8-bit color monitor, using either the color rooted screen or the color rootless screen. You run the command and `maze` appears in a window on your 8-bit color monitor.

Since you set `maze` to appear on a color monitor at 8 bits per pixel, MacX draws `maze` fastest on the 8-bit color monitor. If `maze` had been set to display on a 1 bit per pixel monochrome monitor, the optimal speed would have been achieved on the 1-bit monitor.

To optimize the speed with which MacX draws windows in a multiple monitor setup, position the window on a screen of the same type (color or monochrome) and depth as the screen on which the window was originally displayed.

Running foreign window managers

MacX includes a built-in window manager for rootless screens designated screen number 0 and screen number 2. If you start a foreign window manager, such as `twm`, and specify that the window manager should manage either of these screens, you see the following messages in the Command Output window or in your terminal emulator window:

```
twm: another window manager is already running on screen 0?
twm: another window manager is already running on screen 2?
```

You may also see these messages if the window manager you are attempting to start is set by default to manage all screens. As long as the window manager is running in the location (screen) that you want, you may safely ignore these messages.

You can start some foreign window managers so that they only manage the screen that you specify. The `twm` window manager allows you to do this by providing the `-s` flag option. For example, enter

```
twm -s -display server:0.1
```

to start `twm` only on the black-and-white rooted screen [(1) B&W Rooted]. For more information, see `twm`(1) in *X11 Command Reference for A/UX.*

◆ *Note:* For more information about how MacX deals with screens, see the *MacX User's Guide.* That guide contains a complete description of the four screens (black-and-white rootless, black-and-white rooted, color rootless, and color rooted) available in MacX.

Client applications can't connect to MacX

If the client applications running on your host computer are having a problem connecting to MacX, check the Communications Folder in your System Folder. Make sure you have only one connection tool for each network protocol (TCP/IP, ADSP, and so on.). If you have more than one connection tool for a particular network protocol, remove one of the tools from the Communications Folder and restart MacX.

Displaying motion-sensitive client applications

Some client applications are designed to "follow" the movements of the pointer on the desktop. For example, the `xeyes` client application displays a pair of eyes that move as you move the pointer on the desktop. Depending on the applications appearing on your desktop and the type of screen on which you opened the applications, the motion-sensitive applications may not follow the pointer.

When you open a client application in MacX, you choose the type of screen on which its window appears: black-and-white rootless, black-and-white rooted, color rootless, or color rooted. MacX treats these screen types as four distinct physical screens, even though you may have any arrangement of monitors attached to your Macintosh computer. If you're displaying a motion-sensitive client window on one type of screen and click a client window (making it active) that is displayed on a different type of screen, the motion-sensitive window does not respond to the movements of the pointer. To make the motion-sensitive client follow the pointer, you must click the motion-sensitive client window to make the application active. (When you make a client application active, any operations you subsequently perform, such as typing at the keyboard or choosing a command from a menu, are performed on that active client application.)

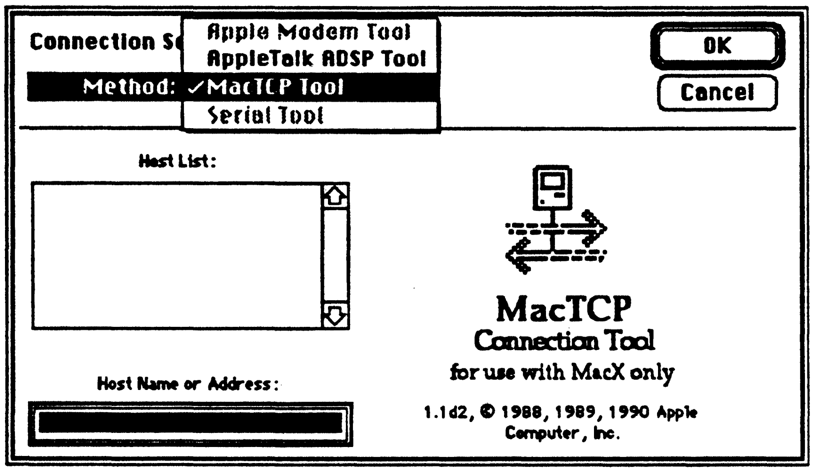
MacX 1.1 limitations

This section provides information about certain limitations of MacX 1.1.

Dimmed connection tools in the Host dialog box

To create a remote command, you choose a connection tool from the Method pop-up menu in the Connection Settings box. Some connection tools do not fully support the remote command facility in MacX. The names of these connection tools appear dimmed in the Method pop-up menu, as shown in Figure 13.

■ **Figure 13** The Method pop-up menu in the Connection Settings box



Conflicts between the X11 protocol and Macintosh keyboard characters

In order to conform to the X11 protocol, MacX translates keypresses into a predefined set of X11 "keysyms." This set contains the X11 character set (ISO Latin-1) and a number of special-purpose keys, such as the RETURN key. If you type a character on a Macintosh keyboard that MacX cannot translate into the X11 keysym set, MacX sounds the system beep to inform you that the character is unavailable in the MacX environment and has been ignored.

Macro software incompatibility

Do not use MacroMaker to simulate keypresses in MacX. Other keyboard macro software may also cause problems in MacX. These types of keyboard macro programs simulate keypresses but do not include key-release simulations. When MacX receives a keypress simulation and then does not receive a subsequent key-release simulation, it "thinks" that the key is being pressed continuously.

Conflicts between `xclipboard` and the Macintosh Clipboard

MacX uses the built-in Macintosh Clipboard when you cut and paste text. If you run the `xclipboard` client application with MacX, it conflicts with the Clipboard. To prevent this problem, *do not* run `xclipboard` with MacX. (In other words, do not open the `xclipboard` client application.)

Closing MacX while client applications are active

MacX won't allow you to open any new client applications if you close a MacX session, disconnect all active client applications simultaneously, and then open a new MacX session.

To prevent this problem, close each client application individually using one of several methods, such as clicking the close box of a client application's window, or by selecting a window and pressing COMMAND-W. After closing the last active client application, wait a couple of minutes before you close the current MacX session and open a new one.

MacX does not implement certain X requests

Certain incompatibilities exist between the Macintosh and X environments. For example, certain X applications attempt to change keyboard and bell sound settings, interfering with the functions of the Macintosh Operating System. MacX ignores the following requests:

- ChangeHost
- SetAccessControl
- SetScreenSaver
- ForceScreenSaver
- SetFontPath
- ChangeKeyboardControl

For rootless screens, MacX ignores these requests:

- CirculateWindow
- GrabPointer
- GrabButton

MacX ignores any attempts by the ConfigureWindow request to change the stacking order of top-level rootless windows. MacX recognizes all other features of ConfigureWindow.

Limits to color-table animation

In MacX 1.1, color-table animation works only on monitors of 8 bits per pixel that are based on a color look-up table (CLUT).

Client applications appear "grainy" on color screens

MacX loads standard color maps for the color rooted and color rootless screens. These standard color maps use a certain number of color map entries, and leave the remaining entries free. Some client applications, such as `xgif`, attempt to allocate more than the number of free colormap entries. If a client application attempts to do this, it will appear "grainy" on your screen. The client application also may display an error message.

Client applications will have this problem if they are written to make use of all (or nearly all) of the colormap entires in a standard default colormap. If you are an advanced X user or X programmer, you can modify client applications so that each one creates and sets the contents of its own color map. For information about displaying client applications on color screens, see "Displaying Client Applications on Color Screens" earlier in this document.

Problems displaying sample text in Font Director window

When you examine fonts in the Font Director window, you see a sample of the font. For example, if you select Geneva, you see the phrase "The quick brown fox" displayed in the Geneva font. Don't worry if samples for some fonts contain blank spaces instead of characters.

The Font Director attempts to create a sample of a particular font whether or not a character used in the sample actually exists in the font that has been selected. For example, if you select the Cursor font, the Font Director attempts to display the phrase "The quick brown fox jumps over the lazy dog" in that font. If the character *T* does not exist in the Cursor font, a blank space appears in its place.

Reference manual update

This section is of interest to those who have purchased any of the A/UX 2.0 reference manuals. This section contains both new and replacement manual pages for the reference manuals. The affected manual pages are listed below under the manual in which they reside.

A/UX Command Reference, Section 1(A-L)

`df`(1)	Replace with `deroff`.1(1) page 2 through `df`.1(1) page 3.
`f77`(1)	Replace with `expr`.1(1) page 3 through `f77`.1(1) page 3.
`ksh`(1)	Replace with `ksh`.1(1) page 1 through `last`.1(1) page 1.

A/UX System Administrator's Reference, Sections 1M, 7, and 8

`newconfig`(1m)	Replace with `newconfig`.1m(1m) page 1 through `newfs`.1m(1m) page 1.
`sendmail`(1m)	Replace with `sendmail`.1m(1m) page 1 through `setport`.1m(1m) page 1.
`setfile`(1)	Replace `sed`.1(1) page 5 through `sh`.1(1) page 1.
`startmac`(1)	In the A/UX 2.0 Release Notes this manual page was mistaken inserted in section 1M.A replacement page is provided with `sprayd`(1M) on the front and `StartMonitor`(1) on the back to repair section 1M. To insert `startmac`(1) properly in Section 1, replace the original `ssp`(1) with the new `ssp`(1) backed by `startmac`(1) on through to `split`(1) page 1.

Errata

The command `settc` has been changed to `setfile`. All occurances of `settc` should be corrected to read `setfile`. Please mark the following pages in these books:

- In *A/UX Toolbox: Macintosh ROM Interface*, pages 1-3, and 6-15 change all occurances of `settc` to `setfile`.
- In *A/UX Command Reference Section 1(A-L)*, page three of the `fcnvt(1)` reference manual page, change all occurances of `settc` to `setfile`.

Contents of Rolling Review #2
November 21, 1990

X WINDOW REFERENCE FOR A/UX

Entry (Pass Number) **Author: Description of Changes**

No recently-changed man pages were detected.

A/UX PROGRAMMER'S REFERENCE

Entry (Pass Number) **Author: Description of Changes**

No recently-changed man pages were detected.

A/UX COMMAND REFERENCE

Entry (Pass Number)	**Author: Description of Changes**
`awk.1` (Pass 2)	BY mikee: Incorporated changes received from Vicki Brown. She expressed concern over length. The old man page was 4 pages, and this one is now 11 pages. If the added material is useful, I am inclined to leave it in. BY mikee: General editorial cleanup.
`df.1` (Pass 2)	BY mikee: Made changes requested since beta review, mostly involving what options are mutually exclusive of one another (-f and -t versus -P and -B). BY mikee: Through various sources, we have learned about this program reporting different free spaces for root and normal users. I tried to document this as a feature by adding a paragraph this checkout cycle.
`f77.1` (Pass 2)	BY eric: added the -A flag option, technical BY kathyw: technical: Correct the spacing between A and factor in the SYNOPSIS to match the explanation below. BY kathyw: Cosmetic: Regularize SEE ALSO for automatic checking. BY kathyw: Cosmetic: Regularize SEE ALSO for automatic checking. "f77 Reference" should be "f77 Command Syntax" in SEE ALSO section. BY mikee: added coverage of the -N flag option; updated SYNTAX description too. BY mikee: Cosmetic changes from editorial. Added coverage of the -R flag, which had been missing previously.
`ksh.1` (Pass 2)	BY mikee: Turned in changes made by Tom Berry to track latest version of ksh.

	BY mikee: Added section on ksh overview. Commented out section on process substitution; this is not supported in A/UX 2.0. Added in editorial corrections. Clarified use of . command in CAVEATS.
`settc.1` (Pass 2)	BY mikee: Added coverage for all flag options, including desktop location, and attributes. Rewrote SYNOPSIS accordingly.

A/UX SYSTEM ADMINISTRATOR'S REFERENCE

Entry (Pass Number)	**Author: Description of Changes**
`newconfig.1m` (Pass 2)	BY mikee: Added the -k flag option as well as references to kconfig(1M). Changes received from John Sovereign. BY mikee: Made several cosmetic changes after beta review by engineering and DTP editorial staff.
`sendmail.1m` (Pass 2)	BY jeff: added description of -w option BY mikee: Thorough editiorial cleanup of certain gotchas, particularly setting second-person-imperative verb forms. aliases.pag is listed under FILES without any explanation. (A feature or a problem?)

SEE ALSO

eqn(1), nroff(1), tbl(1), troff(1).

BUGS

deroff is not a complete troff interpreter, so it can be confused by subtle constructs. Most such errors result in too much rather than too little output.
The -ml flag option does not handle nested lists correctly.

NAME

`df` — report the amount of unused storage capacity for a disk

SYNOPSIS

`df -t` [`-f`] [`-T` *file-system-type*] [*mount-point*]...
`df -P` [`-i`] [`-T` *file-system-type*] [*mount-point*]...
`df -B` [`-i`] [`-T` *file-system-type*] [*mount-point*]...

DESCRIPTION

`df` prints out the number of free blocks and free inodes available for mounted file systems. File systems may be specified either by device name (for example, `/dev/dsk/c0s0d0`) or by filenames (for example, `/usr`). If the *mount-point* argument is a regular file or directory, `df` reports on the amount of free space for the file system on which that file resides. If no argument is specified, the amount of free space on all of the mounted file systems is printed.

For Berkeley file systems (BSD), the remaining free space is calculated differently depending on whether you are using the root account or not. A root user is allowed to access more blocks, even if all free space are consumed. BSD file systems cannot be used effectively without a residual amount of free disk space. That is why normal users are only allowed to access less blocks than would be required to fill the root file system. The amount by which free blocks are reduced for normal users is controlled through a `tunefs`(1M) parameter. However, you should not reduce this parameter because of the danger of increased fragmentation resulting in impaired performance.

The following flag options are supported:

`-B` list free disk blocks in the BSD style of output format.

`-f` compute the number of blocks in the free list (for SVFS file systems only).

`-i` list the number of free inodes along with everything else reported. Available with `-P` and `-B` options only.

`-P` list free disk blocks in the POSIX style of output format (POSIX User Portability Extension).

`-t` report the total allocated block and inode figures along with everything else. Not available when the `-P` or `-B` flag options are used.

`-T` *file-system-type*
establish the type of file system as *file-system-type*. The ac-

cepted types are: `4.2`, and `5.2`. See `fstab`(4) for more detailed information regarding file system types.

FILES

`/bin/df`	
`/dev/dsk/*`	disk partitioning
`/etc/mtab`	list of currently mounted file systems

SEE ALSO

`mount`(1M), `fs`(4), `fstab`(4), `mtab`(4).

BUGS

Since inodes are file system dependent, the number of inodes reported on remotely mounted file systems is always zero.

This page intentionally blank

BUGS

After argument processing by the shell, `expr` cannot tell the difference between an operator and an operand except by the value. If `$a` is an =, the command:

```
expr  $a  =  '='
```

looks like:

```
expr  =  =  =
```

as the arguments are passed to `expr` (and they will all be taken as the = operator). The following works:

```
expr  X$a  =  X=
```

NAME

f77 — Fortran 77 compiler

SYNOPSIS

f77 [-1] [-66] [-A *factor*] [-c] [-C] [-E] [-f] [-F] [-g] [-I[24s]] [-m] [-N*table entries*]... [-o*output*] [-O] [-onetrip] [-p] [-R] [-S] [-u] [-U] [-w] *file*...

DESCRIPTION

f77 is the Fortran 77 compiler; it accepts several types of *file* arguments:

Arguments whose names end with .f are taken to be Fortran 77 source programs; they are compiled and each object program is left in the current directory in a file whose name is that of the source, with .o substituted for .f. However, if a single Fortran source program is compiled and loaded all at one time, the .o file is deleted. By default the process produces an executable file, named a.out, in the current directory

Arguments whose names end with .r or .e are taken to be EFL source programs; these are first transformed by the EFL preprocessor, then compiled by f77, producing .o files.

In the same way, arguments whose names end with .c or .s are taken to be C or assembly source programs and are compiled or assembled, producing .o files.

Flag Options

The following flag options have the same meaning as in cc(1) (see ld(1) for link editor flag options):

-A *factor*	Expand the default symbol table allocations for the assembler and link editor. The default allocation is multiplied by the factor given.
-c	Suppress link editing and produce .o files for each source file.
-f	In systems without floating-point hardware, use a version of f77 that handles floating-point constants and links the object program with the floating-point interpreter.
-g	Generate additional information needed for the use of sdb(1)
-o*output*	Name the final output file *output*, instead of a.out.

-O	Invoke an object code optimizer.
-p	Prepare object files for profiling (see prof(1)).
-R	Remove the dynamically created assembler input file upon completion.
-S	Compile the named programs and leave the assembler language output in corresponding files whose names are suffixed with .s. (No .o files are created.)

The following flag options are specific to f77:

-onetrip	Perform all DO loops at least once. (Fortran 77 DO loops are not performed at all if the upper limit is smaller than the lower limit.)
-1	Same as -onetrip.
-66	Compile as a Fortran 66 program.
-C	Generate code for run-time subscript range-checking.
-E	The remaining characters in the argument are used as an EFL flag argument whenever processing a .e file.
-F	Apply EFL preprocessor to relevant files and put the result in files whose names have their suffix changed to .of. (No .o files are created.)
-I[24s]	Change the default size of integer variables (only valid on machines where the normal integer size is not equal to the size of a single precision real). -I2 causes all integers to be 2-byte quantities, -I4 (default) causes all integers to be 4-byte quantities, and -Is changes the default size of subscript expressions (only) from the size of an integer to 2 bytes.
-m	Apply the M4 preprocessor to each EFL source file before transforming with the efl(1) processor.

-N*table entries*

Set the maximum number of table entries to the number *entries*. Replace *table* with one of the following letter designations corresponding to a compiler table:

q	equivalence table
x	external names table
s	statement number table

	c control block table n identifier table To allow up to 1000 statement numbers, use -Ns1000 as the flag option and argument.
−U	Do not "fold" cases. F77 is normally a no-case language (i.e., a is equal to A). The −U flag option causes f77 to treat upper and lower cases separately.
−u	Make the default type of a variable *undefined*, rather than using the default Fortran rules.
−w	Suppress all warning messages. If the flag option is −w66, only Fortran 66 compatibility warnings are suppressed.

Other arguments are taken to be link editor flag option arguments, f77-compatible object programs (typically produced by an earlier run), or libraries of f77-compatible routines. These programs, together with the results of any compilations specified, are linked (in the order given) to produce an executable program with the default name a.out.

FILES

/usr/bin/f77	
file.[fresc]	input file
file.o	object file
a.out	linked output
./fort[*pid*].?	temporary
/usr/lib/f77comp	compiler
/lib/c2	optional optimizer
/usr/lib/libF77.a	intrinsic function library
/usr/lib/libI77.a	Fortran I/O library
/lib/libc.a	C library; see Section 3 in *A/UX Programmer's Reference.*

SEE ALSO

asa(1), cc(1), efl(1), fpr(1), fsplit(1), ld(1), m4(1), prof(1), sdb(1)
"f77 Command Syntax," in *A/UX Programming Languages and Tools, Volume 1*

DIAGNOSTICS

The diagnostics produced by f77 itself are self-explanatory. The linke editor, ld(1), may occasionally write messages upon the output stream(s). ld(1).

NAME

`ksh` — interpret input as commands, in a way that is compatible with the Bourne shell

SYNOPSIS

`ksh [-a] [-e] [-f] [-h] [-i] [-k] [-m] [-n] [-o] [-p] [-r] [-s] [-t] [-u] [-v] [-x] [-o` *option*`]... [-c` *string*`] [`*arg*`...]`

DESCRIPTION

`ksh` is a command programming language that executes commands read from a terminal or a file. See "Invocation" later in this section for the meaning of arguments to the shell.

Structure

A *command* is an instruction to the shell requesting that the operating system perform some task. Commands can be combined to perform highly complex tasks. `ksh` allows you to combine commands without the inconvenience (such as compilation) that other languages require. The shell reads each command and carries out the desired action either directly or by invoking separate utilities (thus creating a separate process). A *special command* is a command that is carried out by the shell without creating a separate process (see "Special Commands" later in this section). Except for documented side effects, most special commands can be implemented as separate utilities.

The structure of commands is fairly simple. Commands are sequences of *words*. In this context, a *word* is a sequence of characters separated by one or more nonquoted *metacharacters*. A *metacharacter* is one of the following characters:

`; & ( ) | < >` *newline space tab*

These have special meanings to the shell, described below.

The most common command form is the *simple-command*, a series of blank-separated words (*blanks* are spaces or tabs). The first word is the actual command, the remainder are parameters that effect what the command does, or define the file(s) on which a command will operate. For notational ease, the entire sequence of command word and parameters is called a *command*. Another simple command form is the single-line command where newlines serve to separate commands. This looks like

```
command1
command2
. . .
```

This is the way commands are normally input at terminals, but this form can also be put into a file and read from there. When `ksh` takes its input from a file, it reads all the commands provided then executes them sequentially.

Semicolons also serve to separate commands, so the above could be written

```
command1 [;command2]...
```

This shows that there can be several commands on a line. Some commands, known as *keywords*, are required to be at the beginning of a line, that is, preceded by a newline or a semicolon (see "Commands" later in this section).

Commands can also be much more complex, involving multiple lines of commands using flow control constructs. Complex commands will commonly use *identifiers* to manipulate quantities and strings. An *identifier* is a sequence of letters, digits, or underscores starting with a letter or underscore excluding metacharacters. Identifiers are used as names for *aliases*, *functions*, and *variables*.

Commands

A *simple-command* is a sequence of blank-separated words which may be preceded by a parameter assignment list (see "Environment" later in this section). The first word specifies the command name to be executed. With exceptions described later, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see `exec`(2)). The *value* of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see `signal`(3) for a list of status values).

A *pipeline* is a sequence of one or more *commands* separated by a vertical bar (`|`). The standard output of all but the last command is connected by a `pipe`(2) to the standard input of the next command. Each command is run as a separate process; the shell waits for the final command to terminate. The exit status of a pipeline is the exit status of the last command.

A *list* is a sequence of one or more pipelines separated by `;`, `&`, `&&`, or `||`, and optionally terminated by `;`, `&`, or `|&`. Of these five symbols, `;`, `&`, and `|&` have equal precedence, which is lower than that of `&&` and `||`. The symbols `&&` and `||` also have equal precedence. A semicolon (`;`) causes sequential execution of the preceding pipeline; an ampersand (`&`) causes asynchronous execution of the preceding pipeline (that is, the shell does *not* wait for that pipeline to finish). The symbol `|&` causes asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell. The standard input and output of the spawned command can be written to and read from by the parent shell using the `-p` flag option of the special commands, `read` and `print` described later. Only one such command can be active at any given time. The symbol `&&` (`||`) causes the *list* following it to be executed only if the preceding pipeline returns a zero (nonzero) value. An arbitrary number of newlines may appear in a *list*, instead of semicolons, to delimit commands.

Unless stated otherwise, the value returned is that of the last simple-command executed in the command. A *command* is either a simple-command or one of the following:

`for` *identifier* `[in` *word*...`]` `;do` *list* `;done`
: Each time a `for` command is executed, *identifier* is set to the next *word* taken from the set of words in *word.* This effectively loops through the commands in *list* once for each occurance of a word in *word.* The commands in *list* may contain references to `$`*identifier*, a variable set to the *n*th word, corresponding to the *n*th iteration of the loop. If `in` *word*... is omitted, then the `for` command executes the `do` *list* once for each positional parameter that is set (see "Parameter Substitution" later). Execution ends when there are no more words in the *word* list.

The notation

```
for identifier [in word...] ;do list ;done
```

is precisely equivalent to

```
for identifier [in word...]
       do
              list
       done
```

`select` *identifier* `[in` *word*... `]` `;do` *list* `;done`
: A `select` command prints on standard error (file descriptor 2), the set of *words*, each preceded by a number. If `in` *word* ... is omitted, then the positional parameters are used instead (see "Parameter Substitution" later). The `PS3` prompt is printed and a line is read from the standard input. If this line

consists of the number of one of the listed `words`, then the value of the parameter *identifier* is set to the *word* corresponding to this number. If this line is empty, the selection list is printed again. Otherwise the value of the parameter *identifier* is set to `null`. The contents of the line read from standard input is saved in the parameter `REPLY`. The *list* is executed for each selection until a break or end-of-file is encountered.

`case` *word* `in` [*pattern* [| *pattern*] ...) *list* `;;`] ... `esac`
A `case` command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for filename generation (see "Filename Generation" later).

`if` *list* `;then` *list* [`elif` *list* `;then` *list*] ... [`;else` *list*] `;fi`
The *list* following `if` is executed and, if a zero exit status is returned, the *list* following the first `then` is executed. Otherwise, the *list* following `elif` is executed and, if its value is zero, the *list* following the next `then` is executed. Failing that, the `else` *list* is executed. If no `else` *list* or `then` *list* is executed, then the `if` command returns a zero exit status.

`while` *list* `;do` *list* `;done`
`until` *list* `;do` *list* `;done`
A `while` command repeatedly executes the `while` *list* and, if the exit status of the last command in the list is zero, executes the `do` *list*; otherwise the loop terminates. If no commands in the `do` *list* are executed, then the `while` command returns a zero exit status; `until` may be used in place of `while` to negate the loop termination test.

(*list*)
Executes *list* in a separate environment. Note that if two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation as described later. A parenthesized list used as a command argument, denoting "process substitution," is also described.

{*list;*}
list is simply executed. Note that { is a *keyword* and must occur at the beginning of a line or after a ; in order to be recognized.

[[*expression*]]
Evaluates *expression* and returns a zero exit status when *ex-*

pression is true. This replaces the `test` command of previous shell versions (see "Conditional Expressions" later in this section).

`function` *identifier* `{`*list*`;` `}`
identifier `()` `{`*list*`;` `}`
: Defines a function which is referenced by *identifier*. The body of the function is the *list* of commands between { and }. (See "Functions" later in this section).

`time` *pipeline*
: The *pipeline* is executed and the elapsed time, as well as the user and system time, are printed on standard error.

The following keywords are recognized only when occurring at the beginning of a line or after a *;* and when not quoted.

```
     if then else elif fi case esac for do
while until done { } function select time [[]]
```

Comments

A # causes the word it precedes and all the following characters prior to a newline to be ignored.

Aliasing

The first word of each command is replaced by the text of an `alias` if an `alias` for this word has been defined. The first character of an `alias` name can be any nonspecial printable character, but the rest of the characters must be the same as for a valid identifier. The replacement string can contain any valid shell script, including the metacharacters listed earlier. The first word of each command of the replaced text will not be tested for additional aliases. If the last character of the alias value is a blank, the word following the alias will also be checked for alias substitution. Aliases can be used to redefine special built-in commands but cannot be used to redefine the keywords listed earlier. Aliases can be created, listed, and exported with the `alias` command and can be removed with the `unalias` command. Exported aliases remain in effect for subshells but must be reinitialized for separate invocations of the shell (see "Invocation" later in this section).

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an alias to take effect, the `alias` command has to be executed before the command that references the alias is read.

Aliases are frequently used as a type of shorthand for full pathnames. A flag option to the aliasing facility allows the value of the alias to be automatically set to the full pathname of the corresponding command. These aliases are called "tracked" aliases. The value of a tracked alias is defined the first time the corresponding command is looked up and becomes undefined each time the PATH variable is reset. These aliases remain tracked so that the next subsequent reference will redefine the value. Several tracked aliases are compiled into the shell. The -h flag option of the set command makes each command name that is a valid alias name into a tracked alias.

The following "exported aliases" are compiled into the shell but can be unset or redefined:

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -'
true=':'
type='whence -v'
```

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted tilde (~). If it does, the word prior to a / is checked to see if it matches a user name in the /etc/passwd file. If a match is found, the ~ and the matched login name are replaced by the login directory of the matched user. This is called a tilde substitution. If no match is found, the original text is left unchanged. A ~ by itself, or in front of a /, is replaced by the value of the HOME parameter. A ~ followed by a + or – is replaced by the value of the parameter PWD or OLDPWD, respectively.

In addition, the value of each keyword parameter is checked to see if it begins with a ~ or if a ~ appears after a :. In either of these cases, a tilde substitution is attempted.

Command Substitution

The standard output from a command enclosed in parentheses preceded by a dollar sign `$(`*command*`)` or a pair of grave accents `` ` ``*command*`` ` `` may be used as part or all of a word; trailing newlines are removed. In the second (archaic) form, the string between the quotes is processed for special quoting characters before the command is executed. (See "Quoting" later.) The command substitution `$(cat` *file*`)` can be replaced by the equivalent but faster `$(<`*file*`)`. Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process.

An arithmetic expression enclosed in double parentheses preceded by a dollar sign `$((`*arithmetic-expression*`))` is replaced by the value of the arithmetic expression within the double parentheses.

Parameter Substitution

A parameter is an identifier , one or more digits, or any of the characters `*`, `@`, `#`, `?`, `-`, `$`, and `!`. A variable (a parameter denoted by an identifier) has a value and has zero or more attributes. By using the `typeset` special command, variables can be assigned values and attributes. The attributes supported by the shell are described later with the `typeset` special command. Exported variables pass values and attributes to subshells but values only to the environment.

The character `$` is used to access the value of a variable or parameter. Thus, the expression `echo $ENV` yields the current value of the variable named ENV.

The shell supports a one-dimensional array facility. An element of an array variable is referenced by a subscript. A subscript is denoted by a `[`, followed by an arithmetic expression (see "Arithmetic Evaluation" later in this section), followed by a `]`. To assign values to an array, use `set -A` *name value* .... The value of all subscripts must be in the range of 0 through 1024. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the first element.

The value of a variable may also be assigned by writing:

name=*value* [*name*=*value*] ...

If the integer attribute, `-i`, is set for *name* the value is subject to arithmetic evaluation as described later.

Positional parameters, parameters denoted by a number, are assigned values with the `set` special command. Parameter `$0` is set from argument zero when the shell is invoked.

`${`*parameter*`}`
: The shell reads all the characters in *parameter* as part of the same word even if it contains braces or metacharacters. The value, if any, of the parameter is substituted. The braces are required when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name or when a variable is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is `*` or `@`, all the positional parameters, starting with `$1`, are substituted (separated by a field separator character). If an array identifier with subscript `*` or `@` is used, the value for each of the elements is substituted (separated by a field separator character).

`${#`*parameter*`}`
: If *parameter* is `*` or `@`, the number of positional parameters is substituted. Otherwise, the length of the value of *parameter* is substituted.

`${#`*identifier*`[*]}`
: The number of elements in the array *identifier* is substituted.

`${`*parameter*`:-`*word*`}`
: If *parameter* is set and is not null, substitute its value; otherwise substitute *word.*

`${`*parameter*`:=`*word*`}`
: If *parameter* is not set or is null, set it to *word*; the value of the parameter is then substituted. Positional parameters can not be assigned in this way.

`${`*parameter*`:?`*word*`}`
: If *parameter* is set and is not null, substitute its value; otherwise print *word* and exit from the shell. If *word* is omitted print a standard message.

`${`*parameter*`:+`*word*`}`
: If *parameter* is set and is not null, substitute *word*; otherwise substitute nothing.

$ {*parameter#pattern*}
$ {*parameter##pattern*}
: If the shell *pattern* matches the beginning of the value of *parameter*, the value of this substitution is the value of the *parameter* with the matched portion deleted; otherwise substitute the value of this *parameter*. In the first form, the smallest matching pattern is deleted and in the latter form, the largest matching pattern is deleted.

$ {*parameter%pattern*}
$ {*parameter%%pattern*}
: If the shell *pattern* matches the end of the value of *parameter*, the value of this substitution is the value of *parameter* with the matched part deleted; otherwise substitute the value of *parameter*. In the first form, the smallest matching pattern is deleted and in the latter form, the largest matching pattern is deleted.

In the preceding, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, pwd is executed only if d is not set or is null.

```
echo ${d:- $(pwd)}
```

If the colon (:) is omitted from the above expression, the shell checks only whether *parameter* is set.

The following variables are automatically set by the shell. If you unset some of these variables, ksh removes their special meaning even if you subsequently set them.

#	The number of positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the set command.
?	The decimal value returned by the last executed command.
$	The process number of this shell.
_	The last argument of the previous command. This parameter is not set for asynchronous commands. This parameter is also used to hold the name of the matching MAIL file when checking for mail. Finally, the value of this parameter is set to the full pathname of each program the shell invokes and is passed in the environment.

	Unsetting this variable removes its special meaning.
!	The process number of the last background command invoked.
A_z	This variable is used to store information on exported variables that have special meaning or have been made readonly.
ERRNO	The value of *errno* as set by the most recently failed system call. This value is system dependent and is intended for debugging purposes. Unsetting this variable removes its special meaning.
LINENO	The line number of the current line within the script or function being executed. Unsetting this variable removes its special meaning.
OLDPWD	The previous working directory set by the `cd` command.
OPTARG	The value of the last option argument processed by the `getopts` special command. Unsetting this variable removes its special meaning.
OPTIND	The index of the last option argument processed by the `getopts` special command. Unsetting this variable removes its special meaning.
PPID	The process number of the parent of the shell.
PWD	The present working directory set by the `cd` command.
RANDOM	Each time this variable is referenced, a random integer is generated. The sequence of random numbers can be initialized by assigning a numeric value to RANDOM. Unsetting this variable removes its special meaning.
REPLY	This variable is set by the `select` statement and by the `read` special command when no arguments are supplied.
SECONDS	Each time this variable is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, the

value returned upon reference will be the value that was assigned plus the number of seconds since the assignment. Unsetting this variable removes its special meaning.

The following variables are used by the shell. In A/UX the default values shown may have been set (via a `.profile` or file) to different values.

`CDPATH`	The search path for the `cd` command.
`COLUMNS`	If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing `select` lists.
`EDITOR`	If the value of this variable ends in `emacs`, `gmacs`, or `vi` and the `VISUAL` variable is not set, the corresponding flag option (see "Special Commands" later in this section) will be turned on.
`ENV`	If this variable is set, then parameter substitution is performed on the value to generate the pathname of the script that will be executed when the shell is invoked (see "Invocation" later in this section). This file is typically used for `alias` and `function` definitions.
`FCEDIT`	The default editor name for the `fc` command.
`IFS`	Internal field separators, normally space, tab, and newline that are used to separate command words that result from command or parameter substitution and for separating words with the special command `read`. The first character of the `IFS` parameter is used to separate arguments for the `$*` substitution (see "Quoting" later in this section).
`HISTFILE`	If this variable is set when the shell is invoked, the value is the pathname of the file that will be used to store the command history (see "Command Reentry" later in this section).
`HISTSIZE`	If this variable is set when the shell is invoked, the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

	Setting this to a outrageous value, such as 10000, may result in slow startup for a new invocation of `ksh`.
`HOME`	The default argument (home directory) for the `cd` command.
`LINES`	If this variable is set, the value is used to determine the column length for printing `select` lists. Select lists will print vertically until about two-thirds of the `LINES` lines are filled.
`MAIL`	If this variable is set to the name of a mail file *and* the `MAILPATH` variable is not set, the shell informs the user of the arrival of mail in the specified file.
`MAILCHECK`	This variable specifies how often (in seconds) the shell will check for changes in the modification time of any of the files specified by the `MAILPATH` or `MAIL` variables. The default value is 600 seconds. When the time has elapsed, the shell will check before issuing the next prompt. Unsetting this variable removes its special meaning.
`MAILPATH`	A list of filenames separated by colons (`:`). If this variable is set, the shell informs the user of any modifications to the specified files that have occurred within the last `MAILCHECK` seconds. Each filename can be followed by a `?` and a message that will be printed. The message will undergo parameter and command substitution with the parameter `$_` defined as the name of the file that has changed. The default message is `you have mail in $_`.
`PATH`	The search path for commands (see "Execution" later in this section).
`PS1`	The value of this variable is expanded for parameter substitution to define the primary prompt string, which by default is `$`. The character `!` in the primary prompt string is replaced by the command number (see "Command Reentry" later in this section). Two successive oc-

	currences of ! will produce a single ! when the prompt string is printed.
PS2	Secondary prompt string, by default >.
PS3	Selection prompt string used within a select loop, by default #?.
PS4	The value of this variable is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is +.
SHELL	The pathname of the shell is kept in the environment. At invocation, if the value of this variable contains an r in the basename, the shell becomes restricted.
TMOUT	If set to a value greater than zero, the shell will terminate if a command is not entered within the prescribed number of seconds after issuing the PS1 prompt. (Note that the shell can be compiled with a maximum bound for this value that cannot be exceeded.) Unsetting this variable removes its special meaning.
VISUAL	If the value of this variable ends in emacs, gmacs, or vi, the corresponding option (see "Special Commands") will be turned on.

The shell gives default values to PATH, PS1, PS2, MAILCHECK, TMOUT, and IFS. HOME , SHELL, ENV, and MAIL are not set by the shell (although HOME, MAIL, and SHELL are set by login(1)).

Blank Interpretation

After parameter and command substitution, the results of substitutions are scanned for the field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments "" or '' are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Filename Generation

Following substitution, each command word is scanned for the characters *, ?, and [unless the -f option has been set. If one of these characters appears, then the word is regarded as a pattern. The word is replaced with alphabetically sorted filenames that match the pattern. If no filename is found that matches the pattern, the word is left unchanged. When a pattern is used for filename generation, the character `.` at the start of a filename or immediately following a `/`, as well as the `/` itself, must be matched explicitly. In other instances of pattern matching the `/` and `.` receive no special treatment.

`*`
: Matches any string, including the null string.

`?`
: Matches any single character.

`[[!]`*char1 char2 ... charN*`]`
: Matches any one of the enclosed characters. A pair of characters separated by `-` matches any character lexically between the pair, inclusive. If the first character following the opening `[` is a `!` any character not enclosed is matched. A `-` can be included in the character set by putting it as the first or last character.

A *pattern-list* is a list of one or more patterns separated from each other with a `|`. Composite patterns can be formed with one or more of the following:

`?(`*pattern-list*`)`
: Optionally matches any one of the given patterns.

`*(`*pattern-list*`)`
: Matches zero or more occurrences of the given patterns.

`+(`*pattern-list*`)`
: Matches one or more occurrences of the given patterns.

`@(`*pattern-list*`)`
: Matches exactly one of the given patterns.

`!(`*pattern-list*`)`
: Matches anything, except one of the given patterns.

Quoting

Each of the metacharacters listed above (see "Definitions" earlier) has a special meaning to the shell and causes termination of a word unless protected. A character may be protected form interpretation as a metacharacter by preceding it with a backslash (\). This is referred to as "quoting" or "escaping" a character. The pair \newline is ignored. All characters enclosed between a pair of single quote marks `''` are quoted. A single quote cannot appear within single quotes. Inside double quote marks `""` parameter and command substitution occurs and \ quotes the characters \, \`, ", and $. The meaning of `$*` and `$@` is identical when not quoted or when used as a parameter assignment value or as a filename. However, when used as a command argument, `$*` is equivalent to `$1`*d* `$2`*d* and so on, where *d* is the first character of the `IFS` parameter, whereas `$@` is equivalent to `$1 $2` and so on. Inside grave accent marks (\`\`), \ escapes the characters \, \`, and $. If the grave accents occur within double quotes, then \ also escapes the character ".

The special meaning of keywords or aliases can be removed by quoting any character of the keyword. The recognition of function names or special command names listed later in this section cannot be altered by quoting them.

Arithmetic Evaluation

An ability to perform integer arithmetic is provided with the special command `let`. Evaluations are performed using *long* arithmetic. Constants are of the form [*base*#]*n* where *base* is a decimal number between 2 and 36 representing the arithmetic base and *n* is a number in that base. If *base* is omitted, then base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression as the C language. All the integral operators, other than ++, −−, ?:, and , are supported. Variables can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a variable is referenced, its value is evaluated as an arithmetic expression.

To define a variable as an integer (thus speeding up execution) use `typeset -i` *variable*. When this attribute is selected, the first assignment to the parameter determines the arithmetic base to be used when parameter substitution occurs (see "Special Commands" later in this section).

Since many of the arithmetic operators require quoting, an alternative form of the `let` command is provided. For any command that begins with a `((`, all the characters until a matching `))` are treated as a quoted expression. More precisely, `((`*arithmetic-expression*`))` is equivalent to `let` "arithmetic-expression".

Prompting

When used interactively, the shell prompts with the value of `PS1` before reading a command. If at any time a newline is typed and further input is needed to complete a command, the secondary prompt (that is, the value of `PS2`) is issued.

Conditional Expressions

A conditional expression is used with the `[[` compound command to test attributes of files and to compare strings. Word splitting and file name generation are not performed on the words between `[[` and `]]`. Each expression can be constructed from one or more of the following unary or binary expressions:

`-a` *file*	True if *file* exists.
`-b` *file*	True if *file* exists and is a block special file.
`-c` *file*	True if *file* exists and is a character special file.
`-d` *file*	True if *file* exists and is a directory.
`-f` *file*	True if *file* exists and is an ordinary file.
`-g` *file*	True if *file* exists and has its setgid bit set.
`-k` *file*	True if *file* exists and has its sticky bit set.
`-n` *string*	True if length of *string* is non-zero.
`-o` *option*	True if the named *option* is on.
`-p` *file*	True if *file* exists and is a fifo special file or a pipe.
`-r` *file*	True if *file* exists and is readable by current process.
`-s` *file*	True if *file* exists and its size is greater than zero.
`-t` *n*	True if file descriptor number *n* is open and associated with a terminal device.
`-u` *file*	True if *file* exists and has its setuid bit set.
`-w` *file*	True if *file* exists and is writable by current process.
`-x` *file*	True if *file* exists and is executable by current process. If *file* exists and is a directory, the current process has permission to search in the directory.
`-z` *string*	True if length of *string* is zero.
`-L` *file*	True if *file* exists and is a symbolic link.
`-O` *file*	True if *file* exists and is owned by the effective user id of this process.

`-G` *file* — True if *file* exists and its group matches the effective group id of this process.

`-S` *file* — True if *file* exists and is a socket.

file1 `-nt` *file2*
: True if *file1* exists and is newer than *file2*.

file1 `-ot` *file2*
: True if *file1* exists and is older than *file2*.

file1 `-ef` *file2*
: True if *file1* and *file2* exist and refer to the same file.

string `=` *pattern*
: True if *string* matches *pattern*.

string `!=` *pattern*
: True if *string* does not match *pattern*.

string1 `<` *string2*
: True if *string1* comes before *string2* based on ASCII value of their characters.

string1 `>` *string2*
: True if *string1* comes after *string2* based on ASCII value of their characters.

exp1 `-eq` *exp2*
: True if *exp1* is equal to *exp2*.

exp1 `-ne` *exp2*
: True if *exp1* is not equal to *exp2*.

exp1 `-lt` *exp2*
: True if *exp1* is less than *exp2*.

exp1 `-gt` *exp2*
: True if *exp1* is greater than *exp2*.

exp1 `-le` *exp2*
: True if *exp1* is less than or equal to *exp2*.

exp1 `-ge` *exp2*
: True if *exp1* is greater than or equal to *exp2*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

`(`*expression*`)`
: True if *expression* is true. Used to group expressions.

! *expression*
True if *expression* is false.

expression1 && *expression2*
True if *expression1* and *expression2* are both true.

expression1 || *expression2*
True if either *expression1* or *expression2* is true.

Input/Output

Before a command is executed, its input and output may be redirected by using a special notation interpreted by the shell. The following may appear anywhere in a simple command or may precede or follow a command and are *not* passed on to the invoked command. Command and parameter substitution occurs before *file*, *word*, or *digit* is used except as noted. Filename generation occurs only if the pattern matches a single file and blank interpretation is not performed.

<*file*	Use *file* as standard input (file descriptor 0).
>*word*	Use *file* as standard output (file descriptor 1). If *file* does not exist, it is created. If *file* exists, and the `noclobber` option is on, this causes an error; otherwise, it is truncated to zero length.
>|*file*	Sames as >, except that it overrides the `noclobber` option.
>>*file*	Use *file* as standard output. If *file* exists, output is appended to it (by first seeking to the end-of-file); otherwise, *file* is created.
<>*file*	Open *file* for reading and writing as standard input.
<<[-]*word*	The shell input is read up to a line that is the same as *word*, or to an end-of-file. No parameter substitution, command substitution, or filename generation is performed on *word*. The resulting document, called a "here-document," becomes the standard input. If any character of *word* is quoted, no interpretation is placed on the characters of the document; otherwise, parameter and command substitution occurs, and \ must be used to quote the characters \, $, and `. If - is appended to <<, all leading tabs are stripped from *word* and from the document.

`<&`*digit*	The standard input is duplicated from the file descriptor *digit* (see dup(2)). Similarly for the standard output using `>&` *digit*.
`<&-`	The standard input is closed. Similarly for the standard output using `>&-`.
`<&p`	The input from the co-process is moved to standard input.
`>&p`	The output to the co-process is moved to standard output.

If one of the above is preceded by a digit, then the file descriptor referenced is that specified by the digit (instead of the default 0 or 1). For example,

```
...2>&1
```

means the file referenced by descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*descriptor-to-file*) association at the time of evaluation. For example,

`...1>`*fname*`2>&1`

first associates file descriptor 1 with file *fname*. It then associates file descriptor 2 with the file associated with file descriptor 1 (that is, *fname*). In the example

`...1<`*fname*`2<&1`

file descriptor 2 is associated with the terminal (assuming previous association by file descriptor 1), and then file descriptor 1 is then associated with file *fname*.

If a command is followed by `&` and job control is not active, the default standard input for the command is the empty file `/dev/null`. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The environment (see `environ`(7)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, gives it the corresponding value, and marks it `export`. Executed commands inherit the environment. If the user modifies the values of these parameters or creates new ones, using the `export` or `typeset -x` commands, they become part of the environment. The environment seen by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values may be modified by the current shell, plus any additions, which must be noted in `export` or `typeset -x` commands.

The environment for any simple command or function may be augmented by prefixing it with one or more parameter assignments. A parameter assignment argument is a word of the form *identifier=value*. Thus, the following two commands are equivalent (as far as the above execution of *cmd* is concerned).

```
TERM=450 cmd args
(export TERM; TERM=450; cmd args)
```

If the `-k` flag is set, *all* parameter assignment arguments are placed in the environment, even if they occur after the command name. The command that follows first prints `a=b c` and then `c`.

```
echo a=b c
set -k
echo a=b c
```

This feature is intended for use with scripts written for early versions of the shell and its use in new scripts is strongly discouraged. It is likely to disappear someday.

Functions

The `function` keyword, described in "Commands," earlier in this section, is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters. (See "Execution" later in this section.)

Functions execute in the same process as the caller and share all files, traps (other than `EXIT` and `ERR`), and present working directories with the caller. A trap set on `EXIT` inside a function is executed after the function completes. Ordinarily variables are shared between the calling program and the function. However, the `typeset` special command used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command `return` is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the `-f` or `+f` option of the `typeset` special command. The text of functions will also be listed with `-f`. Functions can be undefined with the `-f` option of the `unset` special command.

Ordinarily, functions are unset when the shell executes a shell script. The `-xf` option of the `typeset` command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be placed in the ENV file.

Jobs

If the `monitor` option of the `set` command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with `&`, the shell prints a line that looks like

```
[1] 1234
```

This line indicates that the job that was started asynchronously was job number 1 and had one (top-level) process, whose process ID was 1234.

If you are running a job and wish to do something else you can press CONTROL-Z, which sends a `STOP` signal to the current job. The shell will then normally indicate that the job has been stopped, and print another prompt. You can then manipulate the state of this job, putting it in the background with the `bg` command, or run some other commands and then eventually bring the job back into the foreground with the foreground command `fg`. A CONTROL-Z takes effect immediately and is similar to an interrupt in that pending output and unread input are discarded when it is typed.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command `stty tostop`. If you set this option, background jobs will stop when they try to produce output as they do when they try to read input.

There are several ways to refer to jobs in the shell. The character `%` introduces a job name. If you wish to refer to job number 1, you can use the name `%1`. Jobs can also be referred to with prefixes of the string typed in to start them. Thus, `fg %ed` would normally restart a suspended `ed(1)` job, provided a suspended job whose name began with the string `ed` was present. Similarly, the notation `%?` *string* refers to any job whose command line contains *string*.

The shell maintains a notion of current and previous jobs. In output pertaining to jobs, the current job is marked with a + and the previous job with a –. The abbreviation `%+` refers to the current job and `%–` refers to the previous job. `%%` is also a synonym for the current job.

The shell knows immediately when the state of a process changes and will generally inform you when a job becomes blocked and no further progress is possible, This information is offered just before the shell prints a prompt, so as not to be interrupted.

When the monitor mode is on, each background job that completes triggers any trap set for `CHLD`.

When you try to leave the shell while jobs are running or stopped, you will be warned that "`You have stopped (running) jobs`". You may use the `jobs` command to see what they are. If you use this command or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

Signals

The `INT` and `QUIT` signals for an invoked command are ignored if the command is followed by an `&` and the job `monitor` option is not active. Otherwise, signals have the values inherited by the shell from its parent (see also the `trap` command later in this section).

Execution

Each time a command is executed, the substitutions described above are carried out. If the command name matches one of the "Special Commands" listed later, it is executed within the current shell process. Next, the command name is checked to see if it matches one of the user defined functions. If it does, the positional parameters are saved and then reset to the arguments of the *function* call. When the *function* completes or issues a RETURN, the positional parameter list is restored and any trap set on `EXIT` within the function is executed. The value of a *function* is the value of the last command executed. A function is also executed in the current shell process. If a command name is not a special command or a user defined *function*, a process is created and an attempt is made to execute the command via `exec`(2).

The shell parameter `PATH` defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path in A/UX 2.0 is `/bin:/usr/bin:/usr/ucb:/mac/bin:` (specifying `/bin`, `/usr/bin`, `/usr/ucb`, `/mac/bin`, and the current directory in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list. If the command name contains a `/`, the search path is not used. Otherwise each directory in the path is searched for an executable file. If the file has execute permission but is not a directory or an `a.out` file, it is assumed to be a file containing shell commands and a subshell is spawned to read it. All nonexported aliases, functions, and variables are unavailable in this case. If the shell command file doesn't have read permission, or if the `setuid` and/or `setgid` bits are set on the file, the shell executes an agent whose job it is to set up the permissions and execute the shell with the shell command file passed down as an open file. A parenthesized command is also executed in a subshell having access to both exported and nonexported quantities.

Command Reentry

The text of the last `HISTSIZE` (default 128) commands entered from a terminal device is saved in a `history` file. The file `$HOME/.sh_history` is used if the `HISTFILE` variable is not set or is not writable. A shell can access the commands of all interactive shells that use the same named `HISTFILE`. The special command `fc` is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by typing the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to `fc`, the value of the parameter `FCEDIT` is used. If `FCEDIT` is not defined, `/bin/ed` is used. The edited command(s) is printed and reexecuted upon leaving the editor. The editor name - is used to skip the editing phase and to reexecute the command. In this case a substitution parameter of the form *old=new* can be used to modify the command before execution. For example, if `r` is aliased to `'fc -e -'` then typing `r bad=good c` will reexecute the most recent command that starts with the letter c, replacing the first occurrence of the string bad with the string good.

Inline Editing Options

Normally each command line entered from a terminal device is simply typed and followed by a newline (RETURN or LINEFEED). If the vi, emacs, or gmacs option is active, the user can edit the command line. To be in any of these edit modes, set the corresponding option. An editing option is automatically selected each time the VISUAL or EDITOR variable is assigned a value ending in either of these option names.

The editing features require that the user's terminal accept RETURN as a carriage return without a linefeed and that a space ( ) must overwrite the current character on the screen.

Note: ADM terminal users should set the "space-advance" switch to "space". Hewlett-Packard series 2621 terminal users should set the straps to "bcGHxZ etX".

The editing modes implement a concept where the user is looking through a window at the current line. The window width is the value of COLUMNS if it is defined, otherwise 80. If the line is longer than the window width minus two, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries, the window will be centered around the cursor. The mark is a > (<, *) if the line extends on the right (left, both) side(s) of the window.

The emacs Editing Mode

This mode is entered by enabling either the emacs or gmacs option. The only difference between these two modes is the way they handle CONTROL-T. To edit, the user moves the cursor to the point needing correction and then inserts or deletes characters or words as needed. All the editing commands are control characters or escape sequences. The notation for control characters is caret (^) followed by the character. For example, ^F is the notation for CONTROL-F. The SHIFT key is *not* depressed. (The notation ^? indicates the DELETE.)

The notation for escape sequences is M- followed by a character. For example, M-f (pronounced Meta f) is entered by depressing ESCAPE (ASCII 033) followed by f. (M-F would be the notation for ESCAPE followed by SHIFT (uppercase) F.)

All edit commands operate from any place on the line (not just at the beginning). Neither RETURN nor LINEFEED is entered after edit commands except when noted.

`^F`	Move cursor forward (right) one character.
`M-f`	Move cursor forward one word. (The editor's idea of a word is a string of characters consisting of only letters, digits, and underscores.)
`^B`	Move cursor backward (left) one character.
`M-b`	Move cursor backward one word.
`^A`	Move cursor to start of line.
`^E`	Move cursor to end of line.
`^]`*char*	Move cursor to character *char* on current line.
`^X^X`	Interchange the cursor and mark.
erase	Delete previous character. (User-defined erase character as defined by the `stty` command, usually CONTROL-H or `#`).
`^D`	Delete current character.
`M-d`	Delete current word.
`M-^H`	Delete previous word. (Meta-backspace)
`M-h`	Delete previous word.
`M-^?`	Delete previous word (Meta-delete) If your interrupt character is `^?` (DELETE, the default), this command will not work.
`^T`	Transpose current character with next character in `emacs` mode. Transpose two previous characters in `gmacs` mode.
`^C`	Capitalize current character.
`M-c`	Capitalize current word.
`M-l`	Change the current word to lowercase.
`^K`	Kill from the cursor to the end of the line. If given a parameter of zero, kill from the start of the line to the cursor.
`^W`	Kill from the cursor to the mark.
`M-p`	Push the region from the cursor to the mark on the stack.
kill	Kill the entire current line. (User-defined kill character as defined by the `stty` command, usually CONTROL-G or `@`.) If two kill characters are entered in succession, all kill characters following will cause a linefeed (useful when using paper terminals).

`^Y`	Restore last item removed from the line. (Yank item back to the line.)
`^L`	Line feed and print current line.
`^@`	Set mark. (Null character)
`M-`	Set mark. (Meta space)
`^J`	Execute the current line. (Newline)
`^M`	Execute the current line. (RETURN)
eof	End-of-file character, normally CONTROL-D, will terminate the shell if the current line is null.
`^P`	Fetch previous command. Each time CONTROL-P is entered, the previous command, backward in time, is accessed.
`M-<`	Fetch the least recent (oldest) history line.
`M->`	Fetch the most recent (youngest) history line.
`^N`	Fetch next command. Each time CONTROL-n is entered, the next command, forward in time, is accessed.
`^R`*string*	Reverse search history for a previous command line containing *string*. If a parameter of zero is given, the search is forward. The string is terminated by a RETURN or newline character. If *string* is preceded by a `^`, the matched line must begin with *string*. If *string* is omitted, the next command line containing the most recent string is accessed. In this case, a parameter of zero reverses the direction of the search.
`^O`	Operate–execute the current line and fetch the next line relative to the current line from the history file.
`M-`*digits*	Define numeric parameter, the digits are taken as a parameter to the next command. (ESCAPE) The commands that accept a parameter are `.`, `^F`, `^B`, *erase*, `^D`, `^K`, `^R`, `^P`, `^N`, `M-.`, `M-_`, `M-b`, `M-c`, `M-d`, `M-f`, `M-h`, and `M-^H`.
`M-`*letter*	Soft-key searches your alias list for an alias by the name *_letter* and if an alias of this name is defined, its value will be inserted on the input queue. The letter must not be one of the above named metafunctions.
`M-.`	The last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert other than the last word.

M-_	Same as M-..
M-*	Attempts filename generation on the current word. An asterisk is appended if the word doesn't contain any special pattern characters.
M-ESCAPE	File name completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.
M-=	Lists all files matching current word pattern if an asterisk is appended.
^U	Multiplies parameter of next command by 4.
\	Escape next character. Editing characters, the user's erase, kill, and interrupt (normally ^?) characters, may be entered in a command line or in a search string if preceded by a \. The \ removes the next character's editing features (if any).
^V	Display version of the shell.

The vi Editing Mode

There are two typing modes. Initially, when you enter a command you are in the input mode. To edit, the user enters control mode by typing ESCAPE (033), moves the cursor to the point needing correction, and then inserts or deletes characters or words as needed. Most control commands accept an optional repeat count prior to the command.

When in vi mode on most systems, canonical processing is initially enabled and the command will be echoed again if the speed is 1200 baud or greater, if it contains any control characters, or if less than one second has elapsed since the prompt was printed. The escape character terminates canonical processing for the remainder of the command and the user can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode.

If the option viraw is also set, the terminal will always have canonical processing disabled. This mode is implicit for systems that do not support two alternate end-of-line delimiters, and may be helpful for certain terminals.

The notation for control characters is caret (^) followed by the character. For example, ^F is the notation for control F. This is entered by depressing 'f' while holding down the 'CTRL' (control) key. The 'SHIFT' key is *not* depressed.

Input Edit Commands

By default the editor is in input mode.

Erase	Delete previous character. (User-defined erase character as defined by the `stty` command, usually ^H (CONTROL-H) or #.)
^W	Delete the previous blank separated word.
^D	Terminate the shell.
^V	Escape next character. Editing characters, the user's erase or kill characters, may be entered in a command line or in a search string if preceded by a CONTROL-V. The CONTROL-V removes the next character's editing features (if any).
\	Escape the next erase or kill character.

Motion Edit Commands

These commands will move the cursor.

[*count*]`l`	Cursor forward (right) one character.
[*count*]`w`	Cursor forward one alphanumeric word.
[*count*]`W`	Cursor to the beginning of the next word that follows a blank.
[*count*]`e`	Cursor to the end of the current word.
[*count*]`E`	Cursor to the end of the current blank-delimited word.
[*count*]`h`	Cursor backward (left) one character.
[*count*]`b`	Cursor backward one word.
[*count*]`B`	Cursor to the preceding blank-separated word.
[*count*]`f`*c*	Find the next character *c* in the current line.
[*count*]`F`*c*	Find the previous character *c* in the current line.
[*count*]`t`*c*	Equivalent to `f` followed by `h`.
[*count*]`T`*c*	Equivalent to `F` followed by `l`.
[*count*]`;`	Repeat the last single character find command, `f`, `F`, `t`, or `T`.
[*count*]`,`	Reverses the last single character find command.
`0`	Cursor to the start of the line.
`^`	Cursor to the first nonblank character in the line.
`$`	Cursor to the end of the line.
`%`	Move to balancing `(`, `)`, `{`, `}`, `[`, or `]`. If cursor is not on one of the above characters, the

remainder of the line is searched for the first occurrence of one of the above characters first.

Search Edit Commands

These commands access your command history.

[*count*]k	Fetch previous command. Each time k is entered, the previous command in time is accessed.
[*count*]-	Equivalent to k.
[*count*]j	Fetch next command. Each time j is entered, the next command in time is accessed.
[*count*]+	Equivalent to j.
[*count*]G	Fetch the command number *count*. The default is the least recent history command.
/*string*	Search backward through history for a previous command containing *string*. The string is terminated by a RETURN or newline. If *string* is null the previous string will be used.
?*string*	Same as / except that the search will be in the forward direction.
n	Search for the next match of the last pattern to / or ? commands.
N	Search for the next match of the last pattern to / or ?, but in reverse direction. Search history for the *string* insert by the previous / command.

Text Modification Edit Commands

These commands will modify the line.

a	Enter input mode and enter text after the current character.
A	Append text to the end of the line. Equivalent to $a.
[*count*]c*motion* c[*count*]*motion*	Delete from the current character through the character preceding the cursor that was moved by *motion*. If *motion* is c, the entire line will be deleted and the input mode entered.
C	Delete from the current character through the end of the line and enter input mode. Equivalent to c$.
S	Equivalent to cc.
D	Delete from the current character through the end of the line. Equivalent to d$.

[*count*]d*motion* d[*count*]*motion*	Delete from the current character through the character that *motion* would move to. If *motion* is d, the entire line will be deleted.
`i`	Enter input mode and insert text before the current character.
`I`	Insert text before the beginning of the line. Equivalent to the two character sequence `^i`.
[*count*]`P`	Place the previous text modification before the cursor.
[*count*]`p`	Place the previous text modification after the cursor.
`R`	Enter input mode and replace the characters on the screen with characters you type in overlay fashion.
`r`*c*	Replace the current character with *c*.
[*count*]`x`	Delete current character.
[*count*]`X`	Delete preceding character.
[*count*]`.`	Repeat the previous text modification command.
`~`	Invert the case of the current character and advance the cursor.
[*count*]`_`	Appends the *count* word of the previous command and enter input mode. The last word is used if *count* is omitted.
`*`	Append an `*` to the current word and attempt filename generation. If no match is found, it rings the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.
	Filename completion. Replace the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.

Other Edit Commands

[*count*]y*motion*
y[*count*]*motion*

[*count*]y*motion* / y[*count*]*motion*	Yank from the current character through the character that *motion* would move the cursor to and puts them into the delete buffer. The text and cursor are unchanged.
`Y`	Yank from the current position to the end of the line. Equivalent to `y$`.
`u`	Undo the last text modification command.
`U`	Undo all the text modification commands performed on the line.
[*count*]`v`	Returns the command `fc -e ${VISUAL:-${EDITOR:-vi}}` *count* in the input buffer. If *count* is omitted, use the current line.
`^L`	Line feed and print current line. Has effect only in the control mode.
`^J`	Execute the current line, regardless of mode. (Newline)
`^M`	Execute the current line, regardless of mode. (RETURN)
#	Insert a # before the line and after each newline prior to sending it. Useful for causing the current line to be inserted in the history without being executed.
=	List the filenames that match the current word as if an asterisk were appended to it.
@*letter*	Search your `alias` list for an `alias` by the name _*letter* and if an alias of this name is defined, its value will be inserted on the input queue for processing.

Special Commands

The following simple commands are executed in the shell process. Input/output redirection is permitted. Unless otherwise indicated, the output is written on file descriptor 1. Commands that are preceded by one or two † are treated specially in the following ways:

- Parameter assignment lists preceding the command remain in effect when the command completes.
- Commands are executed in a separate process when used within command substitution.

- I/O redirections are processed after parameter assignments.
- Errors cause the script that contains them to abort.
- Words, following a command preceded by a †† that are in the format of a parameter assignment, are expanded with the same rules as a parameter assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

† `:` [*arg* ...]
    Expand only parameters. A zero exit code is returned.

†† `.` *file* [*arg* ...]
    Read and execute commands from *file* and return. The commands are executed in the current shell environment. The search path specified by `PATH` is used to find the directory containing *file*. If any arguments *arg* are given, they become the positional parameters. Otherwise the positional parameters are unchanged.

`alias` [`-tx`] [*name* [=*value*] ...]
    `alias` with no arguments prints the list of aliases in the form *name=value* on standard output. An `alias` is defined for each name whose value is given. A trailing space in *value* causes the next word to be checked for alias substitution. The `-t` flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given name. The value becomes undefined when the value of `PATH` is reset but the aliases continue to be tracked. Without the `-t` flag, assigned to each name in the argument list for which no value is given, the name and value of the alias is printed. The `-x` flag is used to set or print exported aliases. An exported alias is defined across subshell environments. Alias returns true unless a name is given for which no alias has been defined.

`bg` [`%`*job*]
    Put the specified job into the background. The current job is put in the background if *job* is not specified.

† `break` [*n*]
    Exit from the enclosing `for`, `while`, `until`, or `select` loop, if any. If *n* is specified, break *n* levels.

† `continue` [*n*]
    Resume the next iteration of the enclosing `for`, `while`, un-

`til`, or `select` loop. If *n* is specified, resume at the *n*th enclosing loop.

`cd` [*arg*]
`cd` *old new*
This command can be in either of two forms. In the first form it changes the current directory to *arg*. If *arg* is – the directory is changed to the previous directory. The shell parameter `HOME` is the default *arg*. The parameter `PWD` is set to the current directory. The shell parameter `CDPATH` defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (`:`). The default path is `<null>` (specifying the current directory). Note that the current directory is specified by a null pathname, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a `/`, the search path is not used. Otherwise, each directory in the path is searched for *arg*.

The second form of `cd` substitutes the string *new* for the string *old* in `PWD`, the current directory name, and tries to change to this new directory.

`echo` [*arg...*]
The built-in `echo` command writes its arguments (separated by blanks and terminated by a RETURN) on the standard output. See `echo(1)` for additional information. `echo` is useful for producing diagnostics in shell programs and for writing constant data on pipes. To send diagnostics to the standard error file, use

```
echo ... 1>&2
```

† `eval` [*arg* ...]
Read the arguments as input to the shell and execute the resulting command(s).

† `exec` [*arg* ...]
If *arg* is given, the command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and affect the current process. If no arguments are given, the command modifies file descriptors as prescribed by the input/output redirection list. In this case, any file descriptors that have numbers greater than 2 and are opened with this mechanism,

are closed when invoking another program.

† `exit` [*n*]
: Cause the shell to exit with the exit status specified by ***n***. If ***n*** is omitted, the exit status is that of the last command executed. An end-of-file will also cause the shell to exit unless it has the `ignoreeof` option (see `set` later in this section) turned on.

†† `export` [*name* . . .]
: Mark the given *names* for automatic export to the environment of subsequently executed commands.

†† `fc` [`-e` *ename*] [`-nlr`] [*first*] [*last*]
†† `fc -e -` [*old=new*] [*command*]
: In the first form, select a range of commands from *first* to *last* from the last `HISTSIZE` commands that were typed at the terminal. The arguments *first* and *last* may be specified as numbers or as strings. A string is used to locate the most recent command that begins with the given string. A negative number is used as an offset to the current command number. If the flag `-l`, is selected, the commands are listed on standard output. Otherwise, the editor program *ename* is invoked on a file containing these keyboard commands. If *ename* is not supplied, the value of the parameter `FCEDIT` (default `/bin/ed`) is used as the editor. When editing is complete, the edited command(s) is executed. If *last* is not specified only *first* will be selected. If *first* is not specified, the default will be the previous command for editing and −16 for listing. The flag `-r` reverses the order of the commands and the flag `-n`, when listing, suppresses the command numbers. In the second form *command* is re-executed after the substitution *old=new* is performed.

`fg` [`%`*job*]
: If *job* is specified, bring it to the foreground. Otherwise, bring the current job into the foreground.

`getopts` *opt-string name* [*arg* . . .]
: Check *arg* for legal options (see **getopt**(1)). If *arg* is omitted, the positional parameters are used. An option argument begins with a + or a −. An option not beginning with + or − or the argument −− ends the options. The variable *opt-string* contains the letters that *getopts* will recognize in *arg*. If a letter is followed by a `:`, that option is expected to have an

argument. The options can be separated from the argument by blanks.

`getopts` places the next option letter it finds inside variable *name* each time it is invoked with a + prepended when *arg* begins with a +. The index of the next argument is stored in OPTIND. The option argument, if any, gets stored in OPTARG.

A leading `:` in *opt-string* causes `getopts` to store the letter of an invalid option in OPTARG, and to set *name* to `?` for an unknown option and to `:` when a required option is missing. Otherwise, `getopts` prints an error message. The exit status is nonzero when there are no more options.

`jobs` [`-l`]
List the active jobs; when given the `-l` option, list process IDs in addition to the normal information.

`kill` [-*sig*] *process* ...
Send either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are given either by number or by name (as given in `/usr/include/signal.h`, stripped of the prefix SIG). The signal numbers and names are listed by `kill -l`. If the signal being sent is TERM (terminate) or HUP (hangup), the job or process will be sent a CONT (continue) signal if it is stopped. The argument *process* can be either a process ID or a job. See also `kill(1)`.

`let` *arg* ...
Each *arg* is an arithmetic expression to be evaluated. All calculations are executed with long integers and no check for overflow is performed. Expressions consist of constants, variables, and operators. The following set of operators, listed in order of decreasing precedence, have been implemented:

`-`	unary minus
`!`	logical negation
`* / %`	multiplication, division, remainder
`+ -`	addition, subtraction
`<= >= < >`	comparison
`== !=`	equality inequality
`=`	arithmetic replacement

Subexpressions in parentheses `()` are evaluated first and can be used to override the precedence rules listed. The evaluation within a precedence group is from right to left for the = operator and from left to right for the others.

A parameter name must be a valid identifier. When a parameter is encountered, the value associated with the parameter name is substituted and expression evaluation resumes. Up to 9 levels of recursion are permitted.

The return code is 0 if the value of the last expression is nonzero, and 1 if otherwise.

† `newgrp` [*arg* ...]
: Equivalent to `exec /bin/newgrp` *arg* ...

`print` [`-Rnprsu`[*n*]] [*arg* ...]
: The shell output mechanism. With no flags or with the flag `-`, print the arguments on standard output as described by `echo`(1). In raw mode, `-R` or `-r`, ignore the escape conventions of `echo`. The `-R` option will print all subsequent arguments and options other than `-n`. The `-p` option causes the arguments to be written onto the pipe of the process spawned with `|&` instead of standard output. The `-s` option causes the arguments to be written onto the history file rather than standard output. The `-u` flag option can be used to specify the one digit file descriptor unit number `n` on which the output will be placed. The default is 1. If the flag option `-n` is used, no newline is added to the output.

`pwd`
: Equivalent to `print -r - $PWD`

`read` [`-prsu` [*n*]] [*variable?prompt*] [*variable* ...]
: The shell input mechanism. Read one line and break it up into words using the characters in `IFS` as separators. In raw mode, `-r`, a \ at the end of a line does not signify line continuation. The first word is assigned to the first *variable*, the second word to the second *variable*, and so on, with leftover words assigned to the last *variable*. The `-p` option causes the input line to be taken from the input pipe of a process spawned by the shell using `|&`. If the `-s` flag is present, the input will be saved as a command in the history file. The flag `-u` can be used to specify a one-digit file descriptor unit to read from. The file descriptor can be opened with the `exec` special command. The default value of *n* is 0. If *variable* is

omitted then `REPLY` is used as the default variable. The return code is 0 unless an end-of-file is encountered. An end-of-file with the `-p` option causes cleanup for this process so another can be spawned. If the first argument contains a `?`, the remainder of this word is used as a prompt when the shell is interactive. If the given file descriptor is open for writing and is a terminal device, the prompt is placed on this unit. Otherwise the prompt is issued on file descriptor 2. The return code is 0 unless an end-of-file is encountered.

†† `readonly` [*variable=value*] [*variable* ...]
Mark the given *variables* "read only." These variables cannot be changed by subsequent assignment.

† `return` [*n*]
Cause a shell function to return to the invoking script with the return status specified by *n*. If *n* is omitted, the return status is that of the last command executed. If `return` is invoked while not in a *function* or a `.` script, then it is the same as an `exit`.

`set` [`1laefhkmnostuvx`] [`-o` *option* ...] [`-A` *name*] [*arg* ...]
The flags for this command have the following meanings:

`-A`	Array assignment. Unset the parameter *name* and assign values sequentially from the list *arg*. If `+A` is used, the parameter *name* is not unset first.
`-a`	Automatically export all subsequent parameters defined.
`-e`	If the shell is noninteractive and if a command fails, execute the `ERR` trap, if set, and exit immediately. This mode is disabled while reading profiles.
`-f`	Disable filename generation.
`-h`	Cause each command whose name is an identifier to become a tracked alias when first encountered.
`-k`	Place all parameter assignment arguments in the environment for a command, not just those that precede the command name.
`-m`	Cause background jobs to run in a separate process group and print a line upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this

flag is turned on automatically for interactive shells.

-n Read commands and check them for syntax errors, but do not execute them. Ignored for interactive shells.

-o Give the following argument one of the following names:

allexport	Same as -a.
errexit	Same as -e.
bgnice	Run all background jobs at a lower priority.
emacs	Enter into an emacs style in-line editor for command entry.
gmacs	Enter into a gmacs style in-line editor for command entry.
ignoreeof	The shell will not exit on end-of-file. The command exit must be used.
keyword	Same as -k.
markdirs	Append a trailing / to all directory names resulting from filename generation.
monitor	Same as -m.
noclobber	Prevent redirection > from truncating existing files. Require >| to truncate a file when turned on.
noexec	Same as -n.
noglob	Same as -f.
nolog	Do not save function definitions in history file.
nounset	Same as -u.
privileged	Same as -p.
verbose	Same as -v.
trackall	Same as -h.

`vi`	Enter into insert mode of a `vi` style inline editor until the escape character 033 is used. This enables the move mode. A RETURN sends the line.
`viraw`	Cause each character to be processed as it is typed in `vi` mode.
`xtrace`	Same as `-x`.

If no flag option name is supplied, the current settings are printed.

`-p`	Reset the `PATH` variable to the default value, disable processing of the `$HOME/.profile` file, and use the file `/etc/suid_profile` instead of the ENV file. This mode is automatically enabled whenever the effective user ID (or group ID) is not equal to the real user ID (or group ID).
`-s`	Sort the positional parameters.
`-t`	Exit after reading and executing one command.
`-u`	Treat unset parameters as an error when substitution is necessary.
`-v`	Print shell input lines as they are read.
`-x`	Print commands and their arguments as they are executed.
`-`	Turn off `-x` and `-v` flags and stop examining arguments for flags.
`--`	Do not change any of the flags; useful in setting `$1` to a value beginning with `-`. If no arguments follow this flag, the positional parameters are unset.

Using + rather than - causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in `$-`. Unless `-A` is specified, the remaining arguments are positional parameters and are assigned, in order, to `$1 $2` and so on. If no arguments are given then the values of all names are printed on the standard output.

† `shift` [*n*]
: Rename the positional parameters from `$`*n*`+1` `$1` default *n* is 1. The parameter *n* can be any arithmetic expression that evaluates to a non-negative number less than or equal to `$#`.

`test` [*expr*]
: This archaic command, used to evaluate conditional expressions, has been replaced by the `[[ ]]` compound command. See "Commands" and "Conditional Expressions" earlier in this section.

† `times`
: Print the accumulated user and system times for the shell and for processes run from the shell.

† `trap` [*arg*] [*sig*] ...
: Cause the command *arg* to be read and executed when the shell receives signal(s) *sig*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Each *sig* can be given as a number or as the name of the signal. Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. If *arg* is omitted or is –, all trap(s) *sig* are reset to their original values. If *arg* is the null string, this signal is ignored by the shell and by the commands it invokes. If *sig* is ERR, *arg* will be executed whenever a command has a nonzero exit code. This trap is not inherited by functions. If *sig* is 0 or `EXIT` and the `trap` statement is executed inside the body of a function, the command *arg* is executed after the function completes. If *sig* is 0 or `EXIT` for a `trap` set outside any function, the command *arg* is executed on exit from the shell. The `trap` command with no arguments prints a list of commands associated with each signal number.

†† `typeset` [`1lHLRZfilprtux` [*n*] [*name* [=*value*]] ...]
: Set attributes and values for shell variables and functions. When invoked inside a function, a new instance of the variable *name* is created. The variable value and type are restored when the function completes. The following list of attributes may be specified:

`-H` This flag provides A/UX to host *namefile* mapping on non-UNIX® machines.

- `-L` Justify left and remove leading blanks from *value*. If *n* is nonzero, it defines the width of the field, otherwise the width is determined by the width of the value of first assignment. When a value is assigned to the variable, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the `-Z` flag is also set. The `-R` flag is turned off.
- `-R` Justify right and fill with leading blanks. If *n* is nonzero, it defines the width of the field, otherwise the width is determined by the width of the value of first assignment. The field is filled with blanks or truncated from the end if the variable is reassigned. The `L` flag is turned off.
- `-Z` Justify right and fill with leading zeros if the first non-blank character is a digit and the `-L` flag has not been set. If the `-L` flag has been set, the field is left adjusted and any leading zeros are removed. If *n* is nonzero, it defines the width of the field; otherwise the width is determined by the width of the value of first assignment.
- `-f` The names refer to function names rather than variable names. No assignments can be made and the only other valid flags are `-t`, `-u`, and `-x`. The flag `-t` turns on execution-tracing for this function. The flag `-u` causes this function to be marked undefined. The FPATH variable will be searched to find the function definition when the function is referenced. The flag `-x` allows the function to remain in effect across shell procedures executed in the same process environment.
- `-i` Variable is an integer. This makes arithmetic faster. If *n* is nonzero, it defines the output arithmetic base, otherwise the first assignment determines the output base.
- `-l` Convert all uppercase characters to lowercase. Turn off the uppercase flag, `-u`.
- `-r` Mark the given *names* read only. These names cannot be changed by subsequent assignment.
- `-t` Tag the variables. Tags are user-definable and have no special meaning to the shell.
- `-u` Convert all lowercase characters to uppercase. Turn off the lowercase flag, `-l`.

- `-x` Mark the given *names* for automatic export to the environment of subsequently executed commands.

Using + rather than – causes these flags to be turned off. If no *name* arguments are given but flags are specified, a list of names (and optionally the values) of the variables which have these flags set is printed. (Using + rather than – keeps the values to be printed.) If no *names* or flags are given, the names and attributes of all variables are printed.

`ulimit` [`-HSacdfmnpstv`] [*limit*]

Set or display a resource limit. The available resources limits are listed below. Many systems do not contain one or more of these limits. The limit for a specified resource is set when *limit* is specified. The value of *limit* can be a number in the unit specified below with each resource, or the value `unlimited`. The `H` and `S` flags specify whether the hard limit or the soft limit for the given resource is set. A hard limit cannot be increased once it is set. A soft limit can be increased up to the value of the hard limit. If neither the `H` or `S` options is specified, the limit applies to both. The current resource limit is printed when *limit* is omitted. In this case the soft limit is printed unless `H` is specified. When more than one resource is specified, the limit name and unit are printed before the value.

- `-a` List all of the current resource limits.
- `-c` Print the number of 512-byte blocks on the size of core dumps.
- `-d` Print the number of kilobytes on the size of the data area.
- `-f` Print the number of 512-byte blocks on files written by child processes (files of any size may be read).
- `-m` Print the number of kilobytes on the size of physical memory.
- `-n` Print the number of file descriptors plus 1.
- `-p` Print the number of 512-byte blocks for pipe buffering.
- `-s` Print the number of kilobytes on the size of the stack area.
- `-t` Print the number of seconds to be used by each process.
- `-v` Print the number of kilobytes for virtual memory.

If no option is given, `-f` is assumed.

`umask` [*three-character-sequence*]

Set the user file-creation mask to *three-character-sequence* (see umask(2)). The three-character-sequence can either be an octal number or a symbolic value as described in chmod(1). If a symbolic value is given, the new umask value is the complement of the result of applying *three-character-sequence* to the complement of the previous umask value. If *three-character-sequence* is omitted, the current value of the mask is printed.

unalias *name* ...
: Remove the aliases given by the list of *names* from the alias list.

unset [−f] *name* ...
: Unassign the variables given by the list of *names*. That is, erase their values and attributes. Readonly variables cannot be unset. If the flag, −f, is set, the names refer to function names. Unsetting ERRNO, LINENO, MAILCHECK, OPTARG, OPTIND, RANDOM, SECONDS, TMOUT, and _ removes their special meaning even if values are subsequently assigned them.

wait [*process*]
: Wait for the specified child process and report its termination status. If *process* is not given, all currently active child processes are waited for. The return code from this command is that of the process waited for. See "Jobs," earlier in this section, for a description of the format of *n*.

whence [−pv] *name* ...
: For each *name*, indicate how it would be interpreted if used as a command name.

: The flag, −v, produces a more verbose report. The −p flag does a path search for *name* even if name is an alias, a function, or a reserved word.

Invocation

If the shell is invoked by exec(2), and the first character of argument zero ($0) is −, the shell is assumed to be a login shell and commands are read from /etc/profile and then from either .profile in the current directory or $HOME/.profile, if either file exists. Next, commands are read from the file named by performing parameter substitution on the value of the environment variable ENV if the file exists (it is usually named .kshrc). If the ENV variable is not set, no files are read for commands. If the −s flag is not present and *arg* is, a path search is performed on the first *arg* to determine the name of the script to execute. The script *arg* must have read permission and any `setuid` and `setgid` settings will be ignored. Commands are then read as described later; the following flags are interpreted by the shell when it is invoked.

`-c` *string* If the `-c` flag is present read commands from *string*.

`-s` If the `-s` flag is present or if no arguments remain, read commands from the standard input. Shell output, except for the output of the "Special Commands" listed earlier, is written to file descriptor 2.

`-i` If the `-i` flag is present or if the shell input and output are attached to a terminal (as told by `ioctl`(2)), this shell is interactive. In this case `TERM` is ignored (so that `kill 0` does not kill an interactive shell), and `INTR` is caught and ignored (so that `wait` is interruptible). In all cases, `QUIT` is ignored by the shell.

`-r` If the `-r` flag is present, the shell is a restricted shell and is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of the restricted shell are identical to those of `ksh`, except that changing the directory; setting the value of `SHELL`, `ENV`, or `PATH`; specifying path or command names containing `/`; and redirecting output (`>` and `>>`) are disallowed.

The restrictions above are enforced after `.profile` and the `ENV` files are interpreted.

When a command to be executed is found to be a shell procedure, the restricted shell invokes `ksh` to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the `.profile`, by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory), has complete control over user actions.

The system administrator often sets up a directory of commands (for example, `/usr/rbin`) that can be safely invoked by the restricted shell. Some systems also provide a restricted editor (**red**).

The remaining flags and arguments are described under the `set` command in "Special Commands" earlier in this section.

EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. Otherwise, the shell returns the exit status of the last command executed (see also the `exit` command earlier). If the shell is being used noninteractively, execution of the shell file is abandoned. Run-time errors detected by the shell are reported by printing the command or function name and the error condition. If the number of the line on which the error occurred is greater than one, the line number is also printed in square brackets (`[]`) after the command or function name.

FILES

```
/bin/ksh
/etc/passwd
/etc/profile
/etc/suid_profile
$HOME/.profile
/tmp/ksh*
/dev/null
```

SEE ALSO

`cat`(1), `csh`(1), `echo`(1), `ed`(1), `env`(1), `newgrp`(1), `sh`(1), `vi`(1), `dup`(2), `exec`(2), `fork`(2), `ioctl`(2), `lseek`(2), `pipe`(2), `umask`(2), `ulimit`(2), `wait`(2), `rand`(3C), `signal`(3), `a.out`(4), `profile`(4), `environ`(5).
"Korn Shell Reference" in *A/UX User Interface*
Bolsky, Morris and David Korn. *The KornShell Command and Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

CAVEATS

If a command that is a tracked alias is executed, and then a command with the same name is installed in a directory in the search path prior to the directory where the original command was found, the shell will continue to execute the original command. Use the `-t` option of the `alias` command to correct this situation.

Some very old shell scripts contain a ^ as a synonym for the pipe character `|`. This synonym is not supported in A/UX 2.0.

If a command is piped into a shell command, all variables set in the shell command are lost when the command completes.

Using the `fc` built-in command within a compound command will cause the whole command to disappear from the history file.

The built-in command `.` (usage: `.` *file* [*arg* ...]) reads the whole *file* before any commands are executed. Therefore, `alias` and `unalias` commands in the file will not apply to any functions defined in the file.

Traps are not processed while a job is waiting for a foreground process. Thus a trap on `CHLD` won't be executed until the foreground job terminates.

Unsetting some special variables removes their special meaning, even if they are subsequently set.

A/UX® 2.0.1 Incremental Update Guide

030-1732

030-1732

Contents

Updating to A/UX 2.0.1 1
What you'll need for the update 1
Updating to A/UX 2.0.1 for a Macintosh IIsi 2
Copying A/UX Startup Utilities to the MacPartition disk 3
  What you need now 3
  Copying A/UX Startup to the MacPartition disk 3
Starting the installer program 6
Using the System Checker 8
  Your current system and the new release 12
  Examining the syschk.report file 13
  Preserving essential A/UX distribution files 13
  Preserving user files 15
  Files to be removed 16
Using bakchk to back up files 17
Loading the A/UX 2.0.1 software 19
Installing the System Folder 21
  What you need now 21
  Using the Installer 22

Figures

Figure 1 Updating to A/UX 2.0.1 for a Macintosh IIsi 2
Figure 2 Macintosh disk dialog box 3
Figure 3 The A/UX Startup window 4
Figure 4 The A/UX Startup Utilities window 5
Figure 5 Updating your A/UX 2.0 system to 2.0.1 10
Figure 6 Using the System Checker and bakchk 11
Figure 7 Installer welcome screen 22

Figure 8	Installer primary dialog box	23
Figure 9	Customize dialog box selected	24

Updating to A/UX 2.0.1

This document describes how to install the A/UX 2.0.1 Incremental Update software. The following procedures assume that you already have A/UX 2.0 installed and set up on your system as described in the *A/UX Installation Guide*. Here's an overview of what this document covers:

- Requirements for performing the incremental update
- Preparing for the update, including backing up existing files
- Copying A/UX Startup Utilities to the MacPartition
- Loading A/UX 2.0.1 software
- Restoring pre-existing files

△ **Important** The A/UX 2.0.1 Incremental Update procedures are significantly different from previous A/UX installations. Make sure you carefully follow the instructions described in this guide. △

What you'll need for the update

In order to complete the Incremental Update procedure you'll need about 90 minutes with a Macintosh system that already has A/.UX 2.0 installed. You'll also use the *A/UX Installation Guide* for A/UX 2.0, and these floppy disks, which are included in your Incremental Update Kit:

- *A/UX 2.0.1 Startup*
- *A/UX 2.0.1 Startup Utilities*
- *A/UX 2.0.1 Installation Tools*
- Approximately 15 *A/UX 2.0.1 Incremental Update disks*, labeled *Disk 1*, *Disk* 2, and so on

x030-1732

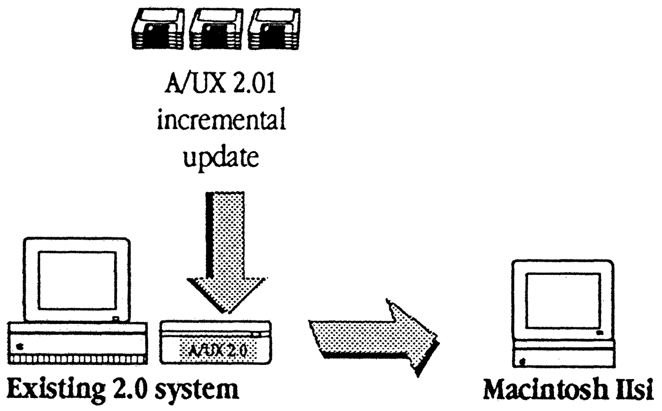
- Three System 6.0.7 disks: *System Tools*, *Utilities 1* and *Utilities 2*,

△ **Important** Before you begin the update procedure, make sure that you have backed up all your files, including any DAs or Fonts you have installed in your personal system folder. Then follow the instructions in the following sections. △

Updating to A/UX 2.0.1 for a Macintosh IIsi

If you're using the A/UX 2.0.Incremental Update kit to run A/UX 2.0.1 on a Macintosh IIsi, you must first update the external hard disk of another system running A/UX 2.0, then transfer the hard disk to your IIsi after the update is complete.

- **Figure 1** Updating to A/UX 2.0.1 for a Macintosh IIsi



△ **Important** In order to run A/UX on a Macintosh IIsi, you must have a math coprocessor, the Motorola MC68882 floating-point unit (FPU). For more information, see your authorized Apple dealer or representative. △

Copying A/UX Startup Utilities to the MacPartition disk

To begin the update procedure, you need to copy a few programs from the *A/UX 2.0.1 Startup* and *A/UX 2.0.1 Startup Utilities* disks. to the hard disk partition labeled MacPartition.

What you need now

Make sure you have the floppy disks *A/UX 2.0.1 Startup* and *A/UX 2.0.1 Startup Utilities* available. These disks contain the stand-alone A/UX shell and stand-alone versions of some key A/UX utilities.

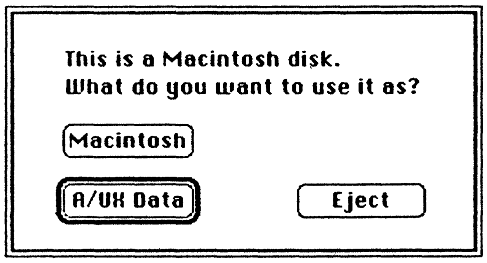
Copying *A/UX Startup* to the MacPartition disk

Follow these steps to copy the contents of *A/UX 2.0.1 Startup* and *A/UX 2.0.1 Startup Utilities* disks to your MacPartition disk:

1. **Insert the disk labeled *A/UX 2.0.1 Startup* into the disk drive.**

   A dialog box (shown in Figure 2) may appear, prompting you to specify whether you want to use the disk as a Macintosh disk or for A/UX data.

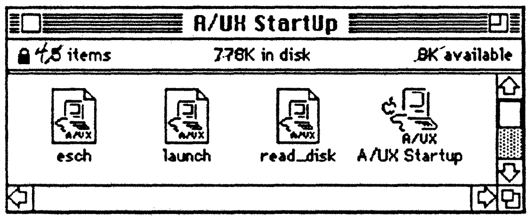
■ **Figure 2** Macintosh disk dialog box


This is a Macintosh disk.
What do you want to use it as?

2. **If the Macintosh disk dialog box appears, click Macintosh.**

   You should see a window showing the contents of the disk as shown in Figure 3. If you do not see this window, double-click the disk icon labeled A/UX Startup.

■ **Figure 3** The A/UX Startup window <<<*to be replaced with current screen shot*>>>



3. **Select everything in the A/UX Startup window.**

   Use Select All from the Edit menu. Another way to select all items in the window is to position the pointer so that it does not touch any of the icons, then press and hold down the mouse button while dragging the pointer across the icons you want to copy. Notice that this draws a dotted line box around the icons. When the box encloses all icons shown in Figure 3, release the mouse button, and the icons are highlighted.

4. **Drag the selected icons to the MacPartition icon.**

   Point to any of the highlighted icons. Press and hold down the mouse button while you drag the icons onto the MacPartition hard disk icon. Release the mouse button when the MacPartition icon is highlighted. Click OK when the system asks you whether to replace files with the same name.

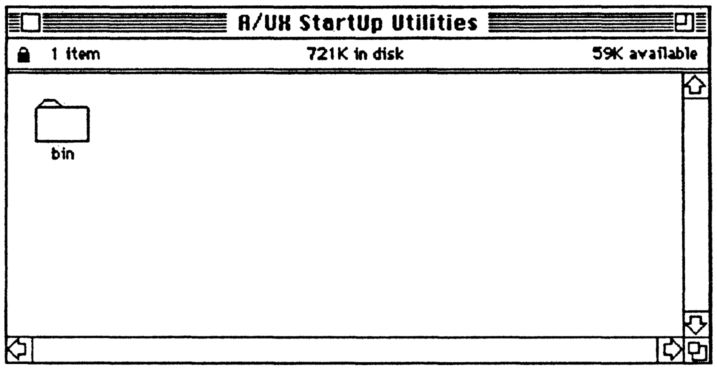
5. **Eject the *A/UX Startup* disk.**

Drag the disk icon to the trash can or choose Eject from the File menu. Remove the *A/UX Startup* disk and put it in a safe place.

6. **Insert the disk labeled *A/UX Startup Utilities* into the floppy-disk drive.**

   A window reveals the contents of the disk, as shown in Figure 4. If you do not see this window, double-click the disk icon labeled A/UX Startup Utilities.

■ **Figure 4** The A/UX Startup Utilities window <<<*new screen shot to be included*>>>



7. **Drag the `bin` folder to the MacPartition icon.**

   Point to the `bin` folder. Press and hold down the mouse button while you drag the icon onto the MacPartition icon. Release the mouse button when the icon is highlighted. Click OK when the system asks you whether to replace files with the same name.

8. **Eject the *A/UX 2.0.1 Startup Utilities* disk.**

   Drag the disk icon to the trash or choose Eject from the File menu. Remove the *A/UX 2.0.1 Startup Utilities* disk and put it in a safe place.

9. **Continue on to the steps in the next section.**

   At this point, your system is prepared for you to start the `installer` program. Continue the 2.0.1 update by following the procedure in the next section.

Starting the `installer` program

The next phase of the update process is to use the `installer` program to perform system checks. Follow these steps:

1. **Open the root hard disk icon.**

   Double-click on the icon labeled /.

2. **Insert the *A/UX Installation Tools* disk into the drive, and open the disk icon if it doesn't open automatically.**

3. **Drag the Install folder to the `/tmp` icon.**

4. **Eject the *A/UX Installation Tools* disk.**

   Drag the disk icon to the trash or choose Eject from the File menu. Remove the *A/UX Installation Tools* disk and put it in a safe place.

5. **Log in to the root account..**

   If you are not already logged in to the root account (in other words, if you're not the super user), log in as root now.

6. **Shut the system down to single user mode.**

   Enter this command in a Command Shell window:

   ```
   /etc/shutdown
   ```

△ **Important** Make sure you type the `shutdown` command. DO NOT use the Shut Down command from the pull down menu. △

7. **If the system prompts you to enter a run level, enter `s`.**

   After you enter the `shutdown` command, messages similar to these appear:

   ```
   SHUTDOWN PROGRAM
   Fri Nov 9 16:44:57 1990
   ```

```
Do you wish to enter your own delay (y or n):
Do you wish to enter your own message? (y or n):
Do you want to continue? (y or n):
```

8. **Respond to the delay and message prompts as desired, then enter `y` when "`Do you wish to continue?`" appears.**

9. **If the system you are updating uses a separate filesystem for `/usr`, `/tmp`, or both, mount them now.**

   These are the commands you use to mount `/usr` and `/tmp`:

   ```
   mount /usr
   mount /tmp
   ```

. **Start the A/UX `installer` program.**

   Enter this command at the single user console prompt (usually this prompt is the number sign, #):

   ```
   /tmp/Install/installer
   ```

   You see these messages:

   ```
   Welcome to the A/UX Incremental Update Procedure.

   If you have two floppy disk drives, be sure to insert all
   disks only into drive 0 when prompted.

   It is not recommended that you interrupt this procedure.
   However, if you must do so, you may type q at any time while
   the program is waiting for a floppy disk.
   Press RETURN to continue
   ```

10. **Press RETURN to continue with the installation.**

    The system checker runs, displaying these messages

    ```
    Checking to see if there's enough room...
    Checking for files modified since Release 2.0...
    Checking for files added since Release 2.0...
    Checking for files that are no longer a part of A/UX...
    ```

11. **If necessary, create additional free space.**

Updating to A/UX 2.01 requires approximately 16,000 free blocks (8 Mb). If the system displays a message indicating that more room is needed for the A/UX 2.0.1 update, remove unnecessary files to increase the available space.

You may be able to increase the available space with the `tunefs` command. To determine if there is additional space available through `tunefs` , use `tunefs -p` on the root filesystem. For example:

```
tunefs -p /dev/dsk/c?/d0s0
```

If the minimum free space that displays is more than 5%, you can reduce it to create more free space. To do this, enter `reboot` at the `A/UX Startup` prompt, and during the reboot process press CTRL -. to stop the boot process in Startup mode, then enter this command:

```
tunefs -m 5 /dev/default
```

After entering the `tunefs` command, repeat the previous steps 6 through 10.

12. **Review the system checker report file and preserve, remove, and back up files as necessary.**

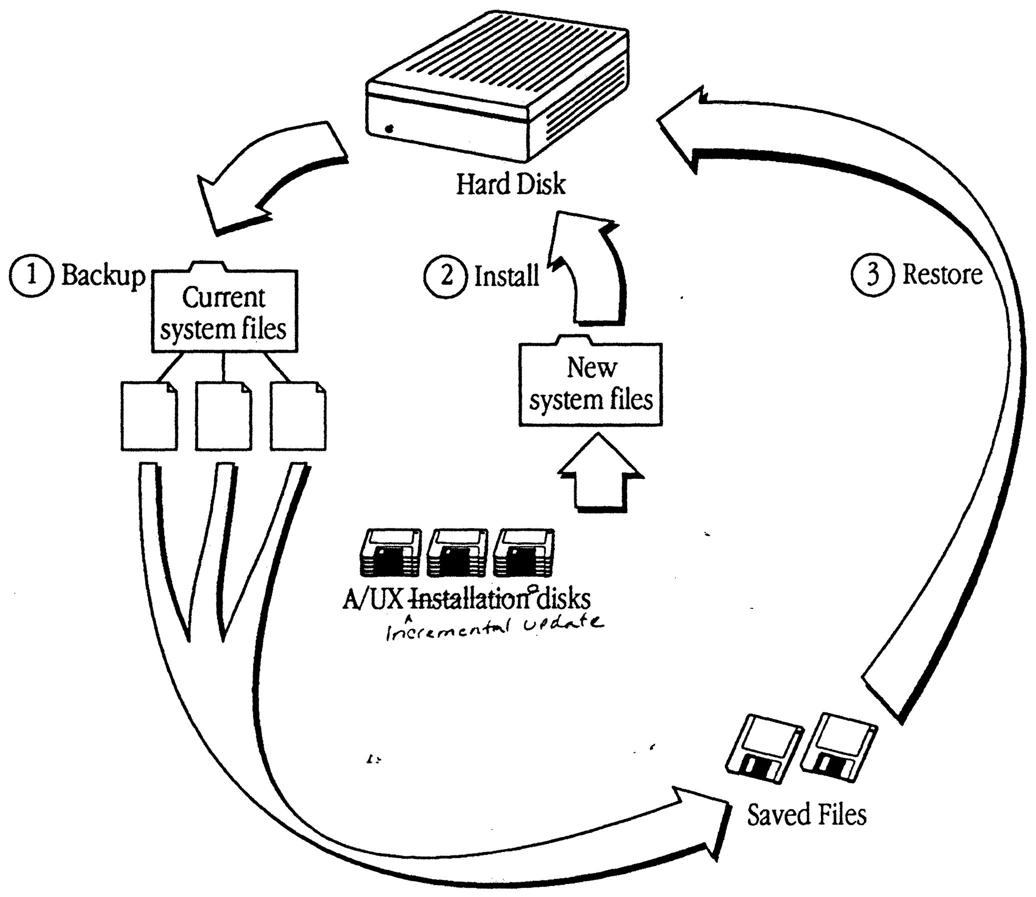
    Examine `/tmp/Install/AUX/syschk.report`, as described in the next section, and save, remove and back up any files as necessary. If there are no A/UX files that you are interested in preserving, skip to the section "Loading the A./UX 2.0.1 software."

Using the System Checker

The programs described in this section, the system checker and `bakchk`, are provided to assist you in the update process. You run these programs from your current A/UX operating system. You also back up your files from A/UX, with floppy disks formatted in the Macintosh Operating System. If there are no existing A/UX files that you wish to save, skip to the section "Loading A/UX 2.0.1 software."As shown in Figure 5, updating your current A/UX 2.0 system to A/UX 2.0.1 consists of these parts:

1. Renaming any user files that have the same name as new A/UX distribution files, then backing up these renamed files and any essential A/UX files that exist on your current system. You can run the System Checker to help you with this process.
2. Loading A/UX 2.0.1 files as described in later sections of this guide.
3. Restoring user files and certain system setup files that you backed up. Instead of restoring the system setup files, you can edit the new A/UX setup files to restore your original system setups. By editing the new A/UX setup files, you ensure that your system retains all your original setups while getting the benefits of the new A/UX settings. "Restoring Saved Files" in Chapter 5 of *A/UX Installation Guide* describes how to restore the files you saved.

■ **Figure 5** Updating your A/UX 2.0 system to 2.0.1

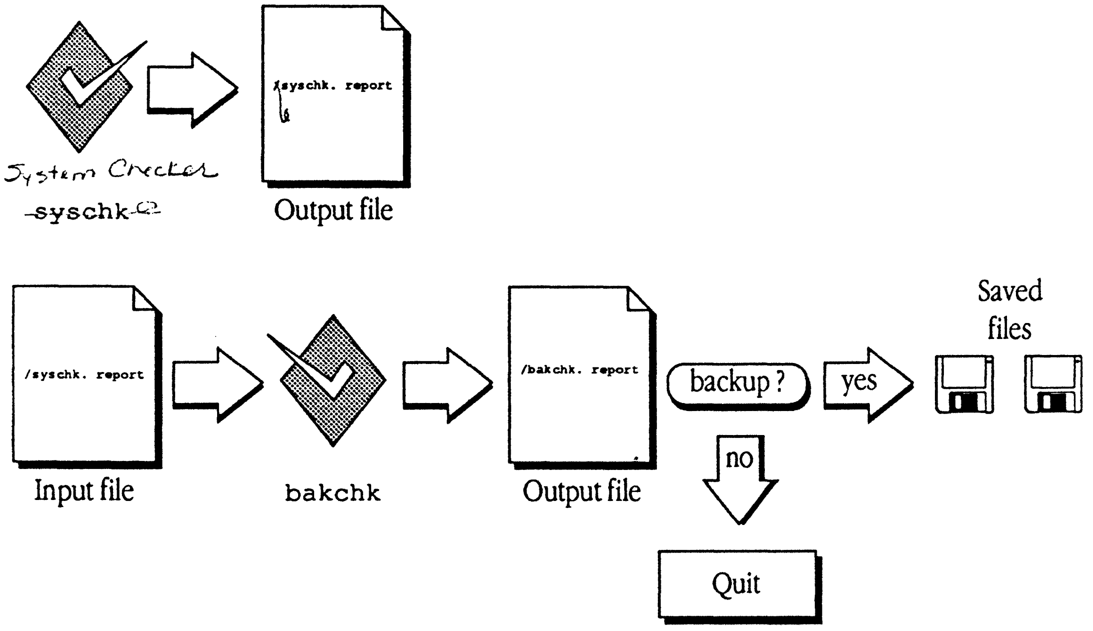


The System Checker ran automatically when you entered the `installer` command during steps in the previous section. The System Checker reveals which of your current A/UX distribution files differ from the new A/UX release.

The System Checker program returns information to you in the report file `syschk.report` in the root directory. The `syschk.report` file lists system files in your current A/UX system that will be replaced in the new release, system files in your current A/UX system that do not exist in the new release, and any user files that have the same name as new A/UX distribution files.

After running System Checker, you can use `/tmp/install/bakchk` to make a floppy diskbackup of files reported by `syschk`. Figure 6 illustrates how the System Checker and `bakchk` are used in the update process.

■ **Figure 6** Using the System Checker and `bakchk`



Your current system and the new release

By running the System Checker, you can identify these categories of files:

- Current A/UX distribution files that will be replaced with new files of the same name. For example, `/etc/inittab` on your current system may be replaced by the `/etc/inittab` file in the new release.

  Certain files that will be replaced have essential information that you may wish to preserve; for instance, your password file and parts of your `/etc/inittab` file. You can use the System Checker to identify A/UX distribution files that have been modified since their installation. The System Checker program lists these files in a report file called `syschk.report`. You can modify that list and use `bakchk` to copy listed files onto floppy disks.

- User-named files that happen to have the same name as one of the new A/UX distribution files and thus are overwritten. You should rename these user files to avoid conflicts with the new files. As a general rule you should not create new files in directories that contain the A/UX distribution, such as `/etc` and `/bin`.

  The System Checker lists in `syschk.report` any existing user files that have the same name as A/UX distribution files in the new installation. You can use this information to rename those files or make backups of them while you are backing up the essential A/UX distribution files. You can later restore your own files from these copies, renaming them or placing them in an appropriate directory to avoid collision with A/UX distribution file names.

Examining the `syschk.report` file

The System Checker helps you identify the files on your 2.0 A/UX system that differ from A/UX 2.0.1. Note that the files listed in `syschk.report` include all the files that have had their modification date changed, not just those that have actually changed. In other words, if you used the command `touch /files`, this file would appear in the list even though you did not actually change the contents of the file.

The `syschk.report` file produced by the System Checker in `/tmp/Install/AUX`, can be used as a control file for a backup utility such as `cpio`. You can edit this file and then use `bakchk` to make backup copies. Or you can edit the file and use it as input on a `cpio` command line. See "Using `bakchk` to Back Up Files" for information on the `bakchk` program.

The `syschk.report` file is divided into three parts:

- Part 1 shows A/UX distribution files that appear to have been modified since your original system was shipped. These files will be replaced by the corresponding A/UX distribution files in the new release.
- Part 2 shows user files whose names conflict with those of files contained in the new release.
- Part 3 shows files that are no longer part of the A/UX release.

Use a text editor to examine your `syschk.report` file. The following sections explain each part of the `syschk.report` file and discuss how you can use the contents of this file to preserve essential A/UX files and back up user files.

Preserving essential A/UX distribution files

Part 1 of the `syschk.report` file shows the A/UX distribution files that appear to have been modified since your original system was shipped. Part 1 of a sample `syschk.report` file is listed here:

```
# These files have been modified, and may need backup:
```

```
# =====================================================
etc/inittab
etc/motd
etc/passwd
etc/server
etc/startup
# unix
...
...
```

These files will be replaced by the corresponding A/UX distribution file in the new release. You can use `bakchk` to back up these files as explained in "Using `bakchk` to Back Up Files," in Chapter 5 of *A/UX Installation Guide.*

When you upgrade to A/UX 2.0.1, your system will contain the new A/UX distribution files. You can then restore your old A/UX files or modify the new A/UX distribution files to include your customized information. (For example, you can back up `/etc/inittab`, and after installing the new release, incorporate the changes from your old `/etc/inittab` file into the new `/etc/inittab` file.) "Restoring Saved Files" in Chapter 5 of *A/UX Installation Guide* helps you customize your new A/UX system.

Follow these steps to decide which files to back up:

1. **Carefully read the list of modified files in the `syschk.report` file.**

   A general rule is to back up filenames that you recognize and files that you may have modified. Most files in `/etc` should be backed up. You probably do not need to back up such files as old log files.

2. **Edit `syschk.report`.**

   Edit the `syschk.report` file by putting a number sign (#) in the first column of each line containing a filename that you do not want to back up (or by deleting the line from the file). In the example, you will notice that `unix` has a # in the first column, because this kernel will be obsolete once you install the new release. Files such as old log files (for example `/usr/spool/lp/oldlog`) and binary files probably do not require backup.

Preserving user files

Part 2 of the `syschk.report` file lists the user files that happen to have the same name as one of the A/UX distribution files contained in the new release. You can rename these files so that when you restore them on the A/UX system they don't overwrite the new A/UX files. Or when you restore these files, you can put them in a directory where they won't conflict with the A/UX file of the same name.

Part 2 of a sample `syschk.report` file is listed here:

```
# These files conflict with files added in the new release:
# ========================================================
/mac/MacX
```

Follow these steps to decide which user files to back up:

1. **Read the list of conflicting filenames in the `syschk.report` file.**
2. **Edit `syschk.report`.**

   Edit the `syschk.report` file by putting a number sign (#) in the first column of each line containing a filename that you do not want to back up (or by deleting the line from the file).

   If you later restore any other files listed here, be sure to restore them to a new directory to avoid overwriting the new A/UX distribution files with the same name.

   If you want to add any user data files to this list, add them after the list of modified files (Part 1) but before the list of filename conflicts (Part 2). The installation process will not overwrite any user data files that have names unlike the names of A/UX distribution files. (However, you should always make a full backup of your system before reconfiguring your system.)

   Files are listed by full pathname in `syschk.report`, and `bakchk` removes the leading / before backing up.

Files to be removed

Part 3 of the `syschk.report` file contains a list of all A/UX distribution files on your current A/UX system that are no longer part of the A/UX release.

Part 3 of a sample `syschk.report` file is listed here:.

```
# These files are no longer in A/UX, and should be removed:
# ========================================================
README
/usr/include/local/dbug.h
/usr/include/local/rmt.h
/usr/spool/lp/FIFO
.
.
.
# Old autoconfiguration files (pre-2.0) WILL NOT work with
# A/UX release 2.0. These files must be removed. Following
# installation of A/UX, re-install third-party drivers ONLY
# after consulting your vendor regarding usability with 2.0.
# ========================================================
/etc/boot.d
/etc/boot.d/BNET
/etc/boot.d/ae6
/etc/boot.d/at_sig
/etc/boot.d/debugger
/etc/boot.d/nfs
/etc/boot.d/slots
/etc/boot.d/toolbox
/etc/init.d
/etc/master.d
/etc/master.d/BNET
/etc/master.d/ae6
/etc/master.d/at_sig
/etc/master.d/debugger
/etc/master.d/nfs
/etc/master.d/slots
/etc/master.d/toolbox
/etc/startup.d
/etc/startup.d/BNET
/etc/startup.d/ae6
```

Using `bakchk` to back up files

The `bakchk` program is designed to help you back up your files. The `bakchk` program uses the list of files in `syschk.report` to produce a file called `bakchk.report` in the root directory. The `/bakchk.report` file can be used as an input file to `cpio`.

If you save files yourself, without using the `bakchk` program, use relative pathnames without a leading slash (/); `bakchk` handles pathnames properly. Always label the disks to identify the files on them and store them in a safe place.

Follow these steps to back up your files:

1. **Edit the `syschk.report` file.**

   Edit the contents of the `syschk.report` file to contain the filenames you want to back up as described in "Preserving Essential A/UX Distribution Files" and "Preserving User Files."

2. **Enter `/tmp/Install/bakchk` to check files in `syschk.report`.**

   The `bakchk` program is in the same directory as the installer program. Enter this command:

   ```
   /tmp/Install/bakchk
   ```

   The `bakchk` program checks the `syschk.report` file for inconsistencies and displays this message

   ```
   Trying dummy cpio...
   ```

   The `bakchk` program inspects the `syschk.report` file to determine whether it contains acceptable input for a `cpio` command. If `bakchk` finds problems with the `syschk.report` file (for example, if `bakchk` could not locate a file), you see a message like

   ```
   could not locate the following files:
             < reports >
             < documents >
   ```

   and `bakchk` quits. If you see this message, examine the contents of the `syschk.report` file and correct any errors reported by `bakchk`. Then run `bakchk` again to make sure you have corrected any problems.

3. **When prompted, enter `y` to back up files listed in `syschk.report` to a floppy disk.**

   If the `syschk.report` file contains filename entries that are acceptable as input to `cpio`, `bakchk` asks if you want to back up the files at this time. Then `bakchk` informs you of the number of floppy disks required to make the backup.

   You see a message like

   ```
   The cpio archive will require 1153 Kbytes.  Do you wish to
   make a cpio archive at this time?  (You must have 2 800K
   disks available to proceed.)  [y/n, default n]:
   ```

   Write down the number of Kbytes that the `cpio` archive will require. You will need this number later when you are restoring files.

   If you decide not to make a backup at this time, enter `n`, which causes the system to quit the `bakchk` program. The `bakchk` program creates a report in the file `/tmp/Install/AUX/bakchk.report`. You can examine this file and use it later as an input file to `cpio` if you wish.

Enter `y` if you want to make a backup at this time and if you have the required number of floppy disks available.

4. **Insert a floppy disk into drive 0 when prompted.**

   The `bakchk` program displays the message

   ```
   Waiting for floppy
   ```

   and waits for you to insert a floppy disk into drive 0. Label and number each disk sequentially, starting with 1. *It is very important that you correctly label each disk.*

   Continue removing and inserting disks as prompted by `bakchk`.

5. **Reinsert the backup disks for verification.**

   After the backup has been made, you see the messages

   ```
   xxxx blocks
   The archive has been written.  Please re-insert the disk(s)
   for verification of readability.
   ```

   Reinsert the floppy disks one by one, starting with disk 1. The `bakchk` program concludes when you see the message

   ```
   xxxx blocks
   ```

   Remove the last floppy disk and store your backup disks in a safe place.

   The system doesn't remove files in `/tmp/Install/AUX`, so you can refer to `bakchk.report` later.

Continue the incremental update by going on to the next section, which describes how to load A/UX 2.0.1 software.

Loading the A/UX 2.0.1 software

To load the A/UX 2.0.1 software onto your 2.0 system you once again use the new A/UX `installer` program. This program allows you to load the 15 or so floppy disks that contain the A/UX 2.0.1 files, prompting you for each disk. This part of the procedure takes about 60 minutes. Follow these steps:

1. **Run the installer again.**

   Enter this command:

   ```
   /tmp/Install/installer
   ```

   The system displays these messages:

   ```
   Welcome to the A/UX Incremental Update Procedure.

   If you have two floppy disk drives, be sure to insert all
   disks only into drive 0 when prompted.

   It is not recommended that you interrupt this procedure.
   However, if you must do so, you may type q at any time while
   the program is waiting for a floppy disk.
   Press RETURN to continue
   ```

12. **Press RETURN to continue with the update.**

    You see these messages:

    ```
    Checking to see if there is enough room to install A/UX...
    Please insert A/UX Incremental Update Disks as prompted
    Waiting for floppy. Insert disk 1
    ```

13. **Insert *A/UX Incremental Update Disk 1* into disk drive 0.**

△ **Important** Always use disk drive 0 (usually the drive on the right if your computer has more than one). △

The software loads the files from *A/UX Incremental Update Disk 1* onto your computer and displays this message:

```
Processing floppy disk 1.
```

When all files on *A/UX Incremental Update Disk 1* are loaded, you see these messages:

```
Waiting for floppy. Insert disk 2
```

14. **Load the remaining A/UX Incremental Update disks.**

The A/UX `installer` program continues to prompt you to insert the rest of the disks. When it finishes loading files from the final disk, it automatically runs the `postinstall` program, which in turn runs `autoconfig` to put device drivers back in your kernel. During this phase, the system prompts you with questions about Yellow Pages and the like. Be sure to give the same answers as you did when originally installing the system. If you're not sure of the correct answer, see your network administrator or answer "No."

15. Restart the system

Use this command:

```
sync; reboot
```

Go on to the next section for instructions on installing the System Folder.

Installing the System Folder

This section shows you how to set up the System Folder on your A/UX disk by using the Installer found on the *System Tools* disk. The System Folder contains resources, such as those necessary for printing, along with desk accessories and other system resources. The resources in the System Folder on your A/UX disk are accessible from the Macintosh Operating System and when you are running A/UX Startup.

What you need now

You use the Installer to install Macintosh system resources on your hard disk labeled MacPartition. Have ready the 6.0.7 System disks, *System Tools*, *Utilities 1* and *Utilities 2*, which are in your A/UX Incremental Update Kit.

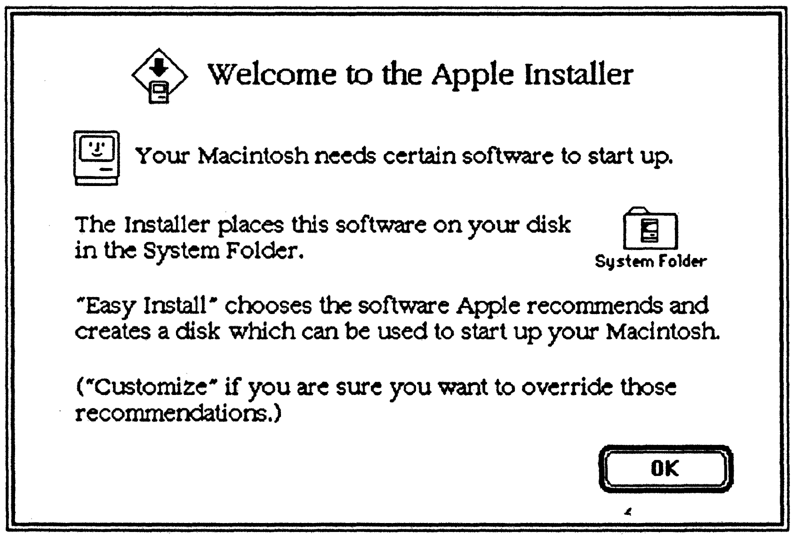
Using the Installer

Follow these steps to install the Macintosh System Folder:

1. **Insert the System Tools disk.**
2. **Open the Installer.**

   Double-click to open the Installer icon. The Installer welcome screen appears as shown in Figure 7.

■ **Figure 7** Installer welcome screen



Welcome to the Apple Installer

Your Macintosh needs certain software to start up.

The Installer places this software on your disk in the System Folder.

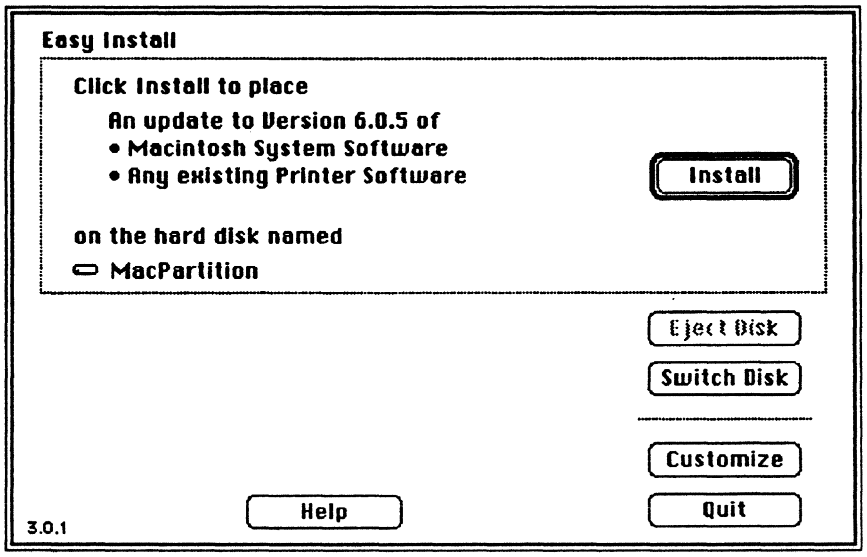
"Easy Install" chooses the software Apple recommends and creates a disk which can be used to start up your Macintosh.

("Customize" if you are sure you want to override those recommendations.)

3. **Click OK to clear the welcome screen.**

   The welcome screen disappears and the Installer's primary dialog box (Figure 8) appears.

■ **Figure 8** Installer primary dialog box


Easy Install

Click Install to place
An update to Version 6.0.5 of
• Macintosh System Software
• Any existing Printer Software

on the hard disk named
MacPartition

3.0.1

4. **Verify that the disk indicated on the screen is the one for installation.**

   Unless you named the disk something else, the hard disk name should be MacPartition. If the wrong disk is indicated, click the Switch Disk button until the correct drive's name appears.

5. **Click Customize.**

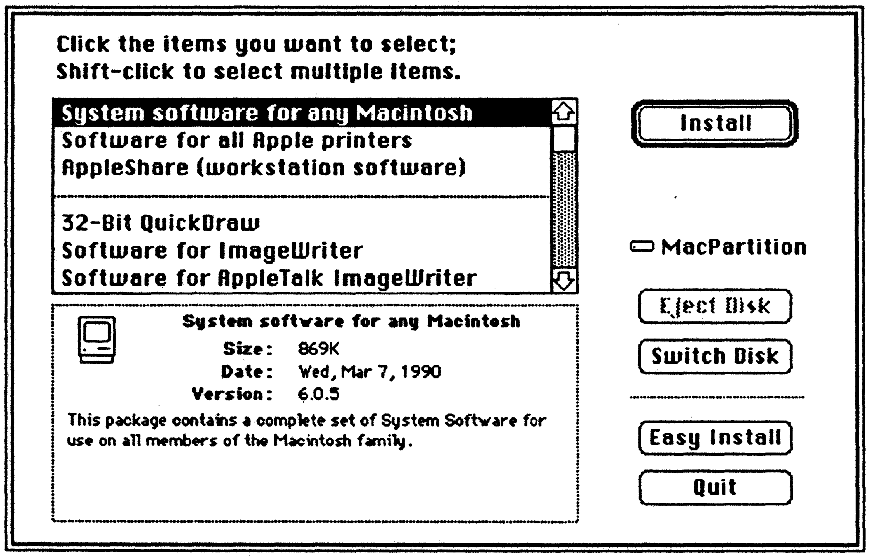
△ **Important** Be sure to click Customize (*not* Install) at this point. If you click Install, the system will display error messages indicating that there is not enough space. If this occurs, repeat the procedure using the Customize option. △

The Customize dialog box appears.

6. **Select "System software for any Macintosh."**

   The Customize dialog box appears as shown in Figure 9.

■ **Figure 9** Customize dialog box selected



Click the items you want to select;
Shift-click to select multiple items.

System software for any Macintosh
Software for all Apple printers
AppleShare (workstation software)
32-Bit QuickDraw
Software for ImageWriter
Software for AppleTalk ImageWriter

System software for any Macintosh
Size: 869K
Date: Wed, Mar 7, 1990
Version: 6.0.5
This package contains a complete set of System Software for use on all members of the Macintosh family.

7. **Click Install.**
8. **When the System Tools disk is ejected and you see a message to insert a different disk, insert the specified disk. Repeat this process as necessary.**

   Depending on the software needed for your system, you may be asked to switch disks more than once. If the system prompts you to insert the *Printing Tools* disk, Cancel and repeat from step 1 of this procedure.
9. **When you see a message reporting that installation was successful, click Quit.**

   The Installer window closes. (If for some reason the installation was not successful, a message on the screen reports this, and you can begin the installation again.)
10. **Choose Restart from the Special menu.**

When you choose Restart, the system ejects the floppy disk from the disk drive; you can now remove the disk. Your computer restarts from a hard drive that contains a System Folder. You should now see the Finder.

11. **If desired, update any personal system folders.**

    The Incremental Update procedure doesn't update any personal System Folders. If you have any personal System Folders, you may update them by entering this command:<<<*Need to be any specific place to enter this command? Necessary to name the system folder*>>>

    ```
    systemfolder -f
    ```

At this point the A/UX 2.0.1 update procedure is complete. See Chapter 5 of *A/UX Installation Guide* for steps to restore any files that you saved. For information on the features and enhancements of this version of A/UX, see the accompanying *A/UX 2.0.1 Release Notes.*

Stopping and restarting the update procedure

To interrupt the A/UX incremental update procedure, enter `q` when the program prompts you to insert an A/UX Incremental Update disk. When you interrupt the update procedure by entering `q`, you see this message:

```
Do you wish to continue the A/UX 2.0.1 Incremental Update [y/n,
default =y, ? for help]>>>
```

To continue the installation, enter `y` and continue with step 13.

If you enter `n`, you see this message

```
Quitting A/UX 2.0.1 Incremental Update procedure.
```

and the A/UX `installer` program returns a command prompt. You can either resume the incremental update immediately by restarting the A/UX `installer` program, starting with step 3 of the previous section, or perform the A/UX 2.0.1 update at a later date.

△ **Important** Don't attempt to interrupt the A/UX 2.0.1 update at any point other than when the Installer program prompts you for a disk. △

If you interrupt the A/UX update process, the A/UX `installer` program is designed to restart at the correct point the next time you run it. For example, if you've successfully copied *A/UX Incremental Update Disk 1* through Disk 5 and then interrupt the update process before inserting *A/UX Incremental Update Disk 6*, the A/UX `installer` program will keep track of which disk was the last to be loaded. When you restart the update, the A/UX Installer program will automatically prompt you for *A/UX Incremental Update Disk 6*.

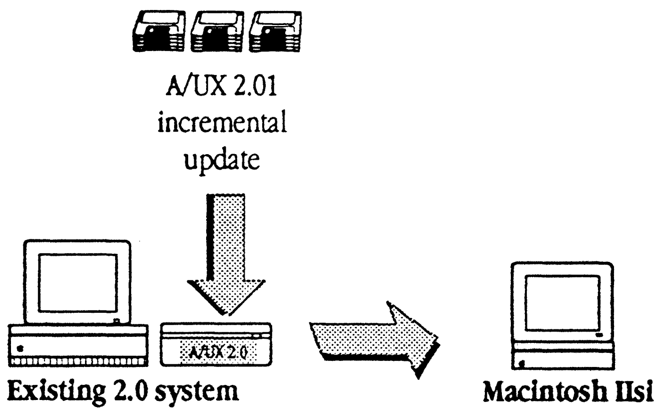
- Three System 6.0.7 disks: *System Tools*, *Utilities 1* and *Utilities 2*,

△ **Important** Before you begin the update procedure, make sure that you have backed up all your files, including any DAs or Fonts you have installed in your personal system folder. Then follow the instructions in the following sections. △

Updating to A/UX 2.0.1 for a Macintosh IIsi

If you're using the A/UX 2.0.Incremental Update kit to run A/UX 2.0.1 on a Macintosh IIsi, you must first update the external hard disk of another system running A/UX 2.0, then transfer the hard disk to your IIsi after the update is complete.

- **Figure 1** Updating to A/UX 2.0.1 for a Macintosh IIsi

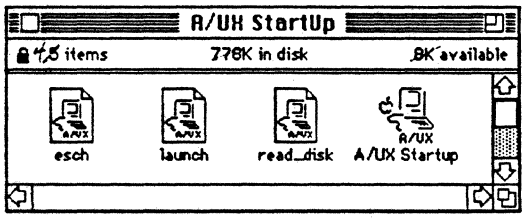


△ **Important** In order to run A/UX on a Macintosh IIsi, you must have a math coprocessor, the Motorola MC68882 floating-point unit (FPU). For more information, see your authorized Apple dealer or representative. △

2. **If the Macintosh disk dialog box appears, click Macintosh.**

   You should see a window showing the contents of the disk as shown in Figure 3. If you do not see this window, double-click the disk icon labeled A/UX Startup.

■ **Figure 3** The A/UX Startup window <<<*to be replaced with current screen shot*>>>



3. **Select everything in the A/UX Startup window.**

   Use Select All from the Edit menu. Another way to select all items in the window is to position the pointer so that it does not touch any of the icons, then press and hold down the mouse button while dragging the pointer across the icons you want to copy. Notice that this draws a dotted line box around the icons. When the box encloses all icons shown in Figure 3, release the mouse button, and the icons are highlighted.

4. **Drag the selected icons to the MacPartition icon.**

   Point to any of the highlighted icons. Press and hold down the mouse button while you drag the icons onto the MacPartition hard disk icon. Release the mouse button when the MacPartition icon is highlighted. Click OK when the system asks you whether to replace files with the same name.

5. **Eject the *A/UX Startup* disk.**

Starting the `installer` program

The next phase of the update process is to use the `installer` program to perform system checks. Follow these steps:

1. **Open the root hard disk icon.**

   Double-click on the icon labeled /.

2. **Insert the *A/UX Installation Tools* disk into the drive, and open the disk icon if it doesn't open automatically.**

3. **Drag the Install folder to the `/tmp` icon.**

4. **Eject the *A/UX Installation Tools* disk.**

   Drag the disk icon to the trash or choose Eject from the File menu. Remove the *A/UX Installation Tools* disk and put it in a safe place.

5. **Log in to the root account..**

   If you are not already logged in to the root account (in other words, if you're not the super user), log in as root now.

6. **Shut the system down to single user mode.**

   Enter this command in a Command Shell window:

   ```
   /etc/shutdown
   ```

△ **Important** Make sure you type the `shutdown` command. DO NOT use the Shut Down command from the pull down menu. △

7. **If the system prompts you to enter a run level, enter `s`.**

   After you enter the `shutdown` command, messages similar to these appear:

   ```
   SHUTDOWN PROGRAM
   Fri Nov 9 16:44:57 1990
   ```

Updating to A/UX 2.01 requires approximately 16,000 free blocks (8 Mb). If the system displays a message indicating that more room is needed for the A/UX 2.0.1 update, remove unnecessary files to increase the available space.

You may be able to increase the available space with the `tunefs` command. To determine if there is additional space available through `tunefs` , use `tunefs -p` on the root filesystem. For example:

```
tunefs -p /dev/dsk/c?/d0s0
```

If the minimum free space that displays is more than 5%, you can reduce it to create more free space. To do this, enter `reboot` at the `A/UX Startup` prompt, and during the reboot process press CTRL -. to stop the boot process in Startup mode, then enter this command:

```
tunefs -m 5 /dev/default
```

After entering the `tunefs` command, repeat the previous steps 6 through 10.

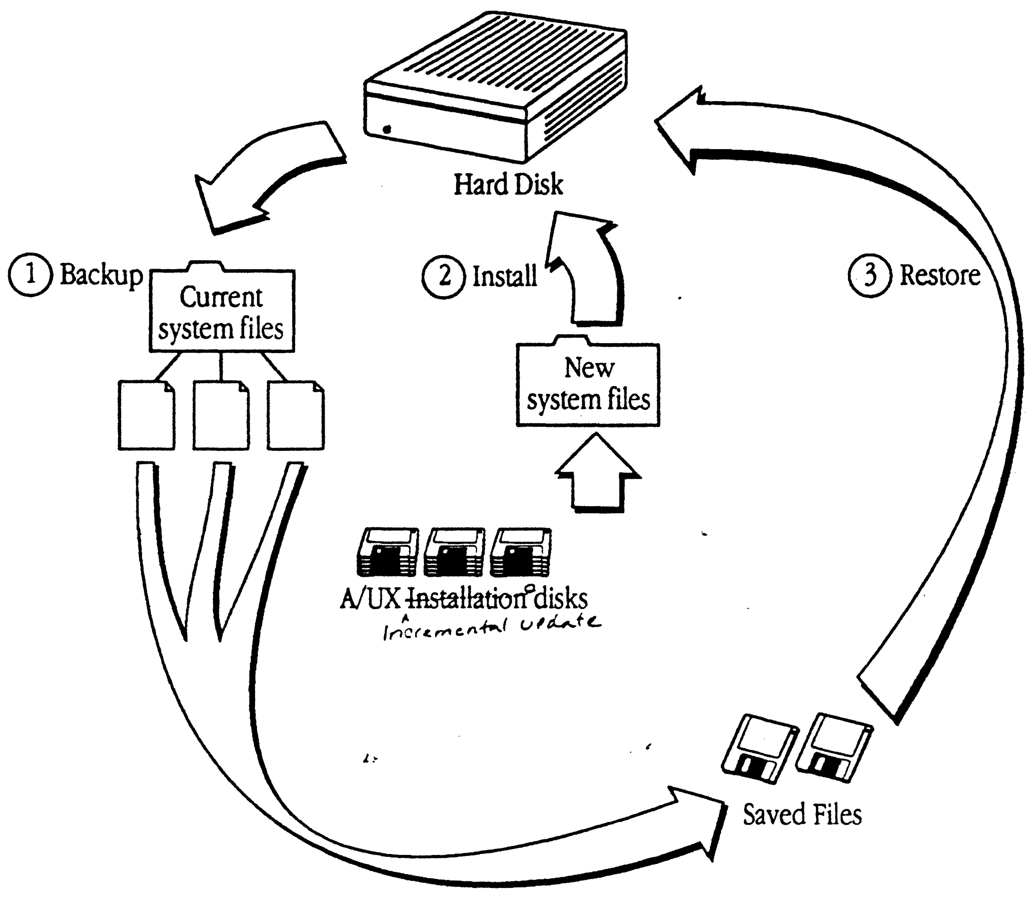
12. **Review the system checker report file and preserve, remove, and back up files as necessary.**

    Examine `/tmp/Install/AUX/syschk.report`, as described in the next section, and save, remove and back up any files as necessary. If there are no A/UX files that you are interested in preserving, skip to the section "Loading the A./UX 2.0.1 software."

Using the System Checker

The programs described in this section, the system checker and `bakchk`, are provided to assist you in the update process. You run these programs from your current A/UX operating system. You also back up your files from A/UX, with floppy disks formatted in the Macintosh Operating System. If there are no existing A/UX files that you wish to save, skip to the section "Loading A/UX 2.0.1 software."As shown in Figure 5, updating your current A/UX 2.0 system to A/UX 2.0.1 consists of these parts:

■ **Figure 5** Updating your A/UX 2.0 system to 2.0.1



Your current system and the new release

By running the System Checker, you can identify these categories of files:

- Current A/UX distribution files that will be replaced with new files of the same name. For example, `/etc/inittab` on your current system may be replaced by the `/etc/inittab` file in the new release.

  Certain files that will be replaced have essential information that you may wish to preserve; for instance, your password file and parts of your `/etc/inittab` file. You can use the System Checker to identify A/UX distribution files that have been modified since their installation. The System Checker program lists these files in a report file called `syschk.report`. You can modify that list and use `bakchk` to copy listed files onto floppy disks.

- User-named files that happen to have the same name as one of the new A/UX distribution files and thus are overwritten. You should rename these user files to avoid conflicts with the new files. As a general rule you should not create new files in directories that contain the A/UX distribution, such as `/etc` and `/bin`.

  The System Checker lists in `syschk.report` any existing user files that have the same name as A/UX distribution files in the new installation. You can use this information to rename those files or make backups of them while you are backing up the essential A/UX distribution files. You can later restore your own files from these copies, renaming them or placing them in an appropriate directory to avoid collision with A/UX distribution file names.

```
# ==================================================
etc/inittab
etc/motd
etc/passwd
etc/server
etc/startup
# unix
...
...
```

These files will be replaced by the corresponding A/UX distribution file in the new release. You can use `bakchk` to back up these files as explained in "Using `bakchk` to Back Up Files," in Chapter 5 of *A/UX Installation Guide.*

When you upgrade to A/UX 2.0.1, your system will contain the new A/UX distribution files. You can then restore your old A/UX files or modify the new A/UX distribution files to include your customized information. (For example, you can back up `/etc/inittab`, and after installing the new release, incorporate the changes from your old `/etc/inittab` file into the new `/etc/inittab` file.) "Restoring Saved Files" in Chapter 5 of *A/UX Installation Guide* helps you customize your new A/UX system.

Follow these steps to decide which files to back up:

1. **Carefully read the list of modified files in the `syschk.report` file.**

   A general rule is to back up filenames that you recognize and files that you may have modified. Most files in `/etc` should be backed up. You probably do not need to back up such files as old log files.

2. **Edit `syschk.report`.**

   Edit the `syschk.report` file by putting a number sign (#) in the first column of each line containing a filename that you do not want to back up (or by deleting the line from the file). In the example, you will notice that `unix` has a # in the first column, because this kernel will be obsolete once you install the new release. Files such as old log files (for example `/usr/spool/lp/oldlog`) and binary files probably do not require backup.

Files to be removed

Part 3 of the `syschk.report` file contains a list of all A/UX distribution files on your current A/UX system that are no longer part of the A/UX release.

Part 3 of a sample `syschk.report` file is listed here:.

```
# These files are no longer in A/UX, and should be removed:
# ==========================================================
README
/usr/include/local/dbug.h
/usr/include/local/rmt.h
/usr/spool/lp/FIFO
.
.
.
# Old autoconfiguration files (pre-2.0) WILL NOT work with
# A/UX release 2.0. These files must be removed. Following
# installation of A/UX, re-install third-party drivers ONLY
# after consulting your vendor regarding usability with 2.0.
# ==========================================================
```

2. **Enter `/tmp/Install/bakchk` to check files in `syschk.report`.**

   The `bakchk` program is in the same directory as the installer program. Enter this command:

   ```
   /tmp/Install/bakchk
   ```

   The `bakchk` program checks the `syschk.report` file for inconsistencies and displays this message

   ```
   Trying dummy cpio...
   ```

   The `bakchk` program inspects the `syschk.report` file to determine whether it contains acceptable input for a `cpio` command. If `bakchk` finds problems with the `syschk.report` file (for example, if `bakchk` could not locate a file), you see a message like

   ```
   could not locate the following files:
            < reports >
            < documents >
   ```

   and `bakchk` quits. If you see this message, examine the contents of the `syschk.report` file and correct any errors reported by `bakchk`. Then run `bakchk` again to make sure you have corrected any problems.

3. **When prompted, enter `y` to back up files listed in `syschk.report` to a floppy disk.**

   If the `syschk.report` file contains filename entries that are acceptable as input to `cpio`, `bakchk` asks if you want to back up the files at this time. Then `bakchk` informs you of the number of floppy disks required to make the backup.

   You see a message like

   ```
   The cpio archive will require 1153 Kbytes.  Do you wish to
   make a cpio archive at this time?  (You must have 2 800K
   disks available to proceed.)  [y/n, default n]:
   ```

   Write down the number of Kbytes that the `cpio` archive will require. You will need this number later when you are restoring files.

   If you decide not to make a backup at this time, enter `n`, which causes the system to quit the `bakchk` program. The `bakchk` program creates a report in the file `/tmp/Install/AUX/bakchk.report`. You can examine this file and use it later as an input file to `cpio` if you wish.

1. Run the installer again.

Enter this command:

```
/tmp/Install/installer
```

The system displays these messages:

```
Welcome to the A/UX Incremental Update Procedure.

If you have two floppy disk drives, be sure to insert all
disks only into drive 0 when prompted.

It is not recommended that you interrupt this procedure.
However, if you must do so, you may type q at any time while
the program is waiting for a floppy disk.
Press RETURN to continue
```

12. Press RETURN to continue with the update.

You see these messages:

```
Checking to see if there is enough room to install A/UX...
Please insert A/UX Incremental Update Disks as prompted
Waiting for floppy. Insert disk 1
```

13. Insert *A/UX Incremental Update Disk 1* into disk drive 0.

△ **Important** Always use disk drive 0 (usually the drive on the right if your computer has more than one). △

The software loads the files from *A/UX Incremental Update Disk 1* onto your computer and displays this message:

```
Processing floppy disk 1.
```

When all files on *A/UX Incremental Update Disk 1* are loaded, you see these messages:

```
Waiting for floppy. Insert disk 2
```

14. Load the remaining A/UX Incremental Update disks.

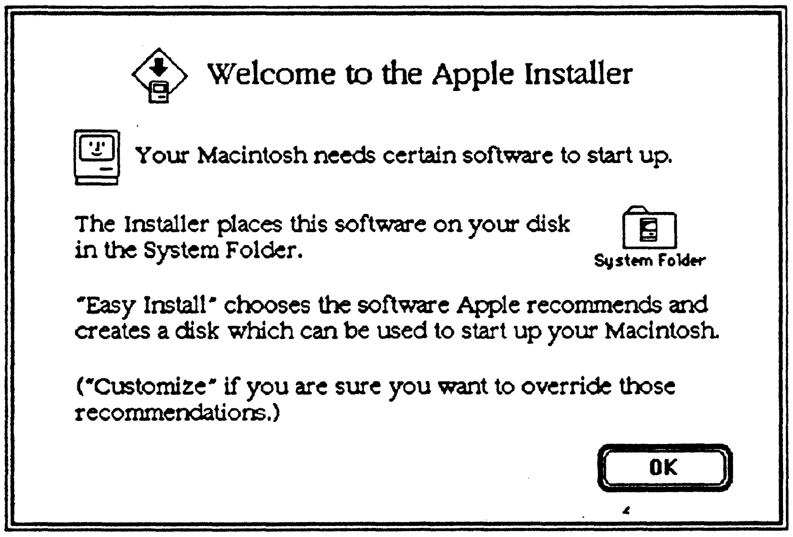
Using the Installer

Follow these steps to install the Macintosh System Folder:

1. **Insert the System Tools disk.**
2. **Open the Installer.**

   Double-click to open the Installer icon. The Installer welcome screen appears as shown in Figure 7.

■ **Figure 7** Installer welcome screen


Welcome to the Apple Installer

Your Macintosh needs certain software to start up.

The Installer places this software on your disk in the System Folder.

"Easy Install" chooses the software Apple recommends and creates a disk which can be used to start up your Macintosh.

("Customize" if you are sure you want to override those recommendations.)

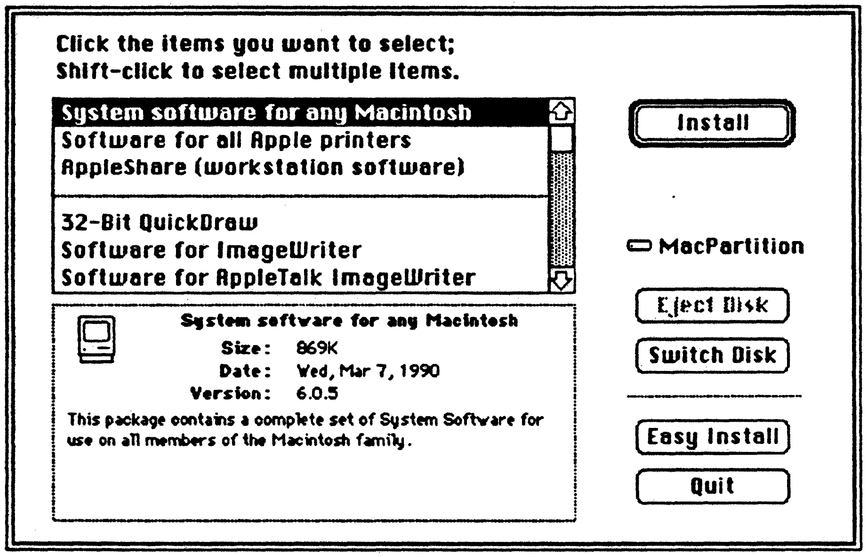
3. **Click OK to clear the welcome screen.**

   The welcome screen disappears and the Installer's primary dialog box (Figure 8) appears.

6. **Select "System software for any Macintosh."**

   The Customize dialog box appears as shown in Figure 9.

■ **Figure 9** Customize dialog box selected



Click the items you want to select;
Shift-click to select multiple items.

System software for any Macintosh
Software for all Apple printers
AppleShare (workstation software)

32-Bit QuickDraw
Software for ImageWriter
Software for AppleTalk ImageWriter

System software for any Macintosh
Size: 869K
Date: Wed, Mar 7, 1990
Version: 6.0.5
This package contains a complete set of System Software for use on all members of the Macintosh family.

7. **Click Install.**
8. **When the System Tools disk is ejected and you see a message to insert a different disk, insert the specified disk. Repeat this process as necessary.**

   Depending on the software needed for your system, you may be asked to switch disks more than once. If the system prompts you to insert the *Printing Tools* disk, Cancel and repeat from step 1 of this procedure.
9. **When you see a message reporting that installation was successful, click Quit.**

   The Installer window closes. (If for some reason the installation was not successful, a message on the screen reports this, and you can begin the installation again.)
10. **Choose Restart from the Special menu.**

If you interrupt the A/UX update process, the A/UX `installer` program is designed to restart at the correct point the next time you run it. For example, if you've successfully copied *A/UX Incremental Update Disk 1* through Disk 5 and then interrupt the update process before inserting *A/UX Incremental Update Disk 6*, the A/UX `installer` program will keep track of which disk was the last to be loaded. When you restart the update, the A/UX Installer program will automatically prompt you for *A/UX Incremental Update Disk 6*.

030-1732