# A/UX® 1.1 Release Notes

031-0185-A

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, California 95014
(408) 996-1010



Simultaneously published in the United States and Canada.

031-0185-A

**LIMITED WARRANTY ON MEDIA AND REPLACEMENT**

If you discover physical defects in the manuals distributed with an Apple product or in the media on which a software product is distributed, Apple will replace the media or manuals at no charge to you, provided you return the item to be replaced with proof of purchase to Apple or an authorized Apple dealer during the 90-day period after you purchased the software. In addition, Apple will replace damaged software media and manuals for as long as the software product is included in Apple's Media Exchange Program. While not an upgrade or update method, this program offers additional protection for up to two years or more from the date of your original purchase. See your authorized Apple dealer for program coverage and details. In some countries the replacement period may be different; check with your authorized Apple dealer.

**ALL IMPLIED WARRANTIES ON THE MEDIA AND MANUALS, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.**

Even though Apple has tested the software and reviewed the documentation, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO SOFTWARE, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE IS SOLD "AS IS," AND YOU THE PURCHASER ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE OR ITS DOCUMENTATION,** even if advised of the possibility of such damages. In particular, Apple shall have no liability for any programs or data stored in or used with Apple products, including the costs of recovering such programs or data.

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED.** No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

# Section 1

# New Features

## 1. New Features

Welcome to Release 1.1 of A/UX®. These pages summarize the new features and improvements to A/UX, as well as document the late-breaking technical information. After you install A/UX following the instructions in the *A/UX Installation Guide*, carefully review these notes. For future reference, place these notes at the end of *A/UX Local System Administration*, behind the tab labeled "Notes."

A/UX 1.1 has been enhanced with the following new features:

**The Ability to Run More Macintosh® Computer Applications**
Many more 32-bit compatible Macintosh applications run under A/UX 1.1 than under previous releases. If an application adheres to the coding specifications documented in *Inside Macintosh*, specifically the requirements for the Macintosh 32-bit environment, the application should run under A/UX without change.

**The Ability to Run HyperCard®**
You can now run HyperCard under A/UX. HyperCard Version 1.2.2 is required.

**Additional Toolbox Support**
This release of A/UX implements more of the Macintosh Toolbox. The Macintosh Toolbox programs that have been added to the A/UX Toolbox are the Color Manager, Slot Manager, Print Manager, Event Manager, and Macintosh Video Driver.

**Color Monitor Support**
The Macintosh Color Manager supports color monitors with a 256K bitmap screen and lets you select the colors for display by using the Control Panel desk accessory.

**POSIX Conformance**
A/UX 1.1 fully conforms to the Federal Information Processing Standard (FIPS) #151 POSIX, a requirement for Federal purchase of a UNIX system.

**X Window System Support**
The X Window System is a windowing environment that runs on A/UX. With this support, you can run programs written for the X Window System. Multiple screens and color monitors are supported for this product. Note that the X Window System is sold separately.

**Serial Line/Internet Protocol (SL/IP) Support**
SL/IP gives you internet access over serial lines to A/UX services such as Network File System (NFS) and standard BSD networking facilities.

**AppleTalk® Network Printing**
In A/UX Release 1.1, you can share ImageWriter® and LaserWriter® printers with other users of Macintosh computers connected over an AppleTalk network. To print files, you can use the A/UX print utilities such as `lp` and `pr`, as well as the print commands available from Macintosh applications. You need to add a third-party NuBus™ card to use these capabilities.

**Apple Tape Backup 40SC™ Support**
The Apple magnetic tape device lets you install or update software from 1/4-inch magnetic tape. The device also lets you use the A/UX backup utilities `tar`, `cpio`, and `dump.bsd` to back up files onto tape. You can also back up and restore A/UX disk partitions from the Macintosh Operating System using the Tape Backup 40SC software.

**Additional Desk Accessory Support**
You can now set parameters for A/UX applications by using the Chooser and Control Panel desk accessories. All the Macintosh desk accessories run in A/UX.

**File Transfer between Macintosh OS and A/UX**
You can transfer data files and applications between the two operating systems with greater ease using a new utility called *hfx* (Hierarchical File system eXchange).

**Apple CDSC CD-ROM™ Support**
A/UX 1.1 supports A/UX file systems stored on the Apple CDSC CD-ROM device, giving third-party developers an efficient, economical means to distribute large amounts of data to A/UX users. A/UX file systems on a CD-ROM are mounted as any other additional file systems.

**Macintosh Video Driver Support**
This release of A/UX supports compatible video cards and monitors without requiring special drivers for them.

**Increased Speed**
This release of A/UX is up to two-and-a-half times faster during disk input and output than the previous releases. Also, this release quadruples SCSI throughput to 1 MB per second.

**Sixteen-user AT&T UNIX License**
Each binary copy and software update of A/UX now includes a 16-user AT&T UNIX license.

# Section 2

# General Notes

## 1. HyperCard under A/UX

HyperCard, Version 1.2.2, runs under A/UX. HyperCard is provided for free with every new Macintosh computer. If you don't have HyperCard, or you have a version earlier than 1.2.2, you can purchase the updated version from your authorized Apple Dealer. Apple Developers can order HyperCard 1.2.2 from the Apple Programmers and Developers Association (APDA) for a fee of $10, plus the annual APDA membership dues of $20.

### 1.1 Installing HyperCard

While in A/UX, use the `hfx` utility to copy the HyperCard application over to A/UX. You can read all about `hfx` in the chapter A/UX Toolbox Utilities in *Getting Started with AUX*.

### 1.2 Starting up HyperCard

1. Have A/UX running in multi-user mode, as it is prudent to run applications while A/UX is in multi-user rather than single-user mode.
2. Move to the directory where you placed HyperCard, if you aren't already there.
3. Enter

```
launch HyperCard
```

HyperCard is opened on the screen and ready for use. For information on using HyperCard, refer to *HyperCard User's Guide*.

#### 1.2.1 Caveats for HyperCard under A/UX

You use HyperCard under A/UX as you would in the Macintosh OS, with the following exceptions.

- The ToolBox Sound Manager is not supported by A/UX, so HyperCard stacks using sound will be silent but otherwise function properly.
- HyperCard stacks cannot directly access hardware under A/UX. Therefore, HyperCard stacks like HyperScan, which accesses SCSI ports, or that attempt to access Macintosh device files, do not work.
- If HyperCard can't find a file you ask for, HyperCard presents a Get File dialog box. It may be that HyperCard couldn't find the file because you misspelled the name. Remember that in A/UX uppercase and lowercase letters in a filename are important. For example, if you instruct HyperCard to open a stack named Weasels and the name is actually spelled *weasels*, HyperCard won't find the file. HyperCard instead presents the Find File dialog box. Select the correct filename from those listed in the box and HyperCard can then find the file.

  HyperCard retains in memory cache the spelling of the filename as you gave it, rather than storing the name you selected in the Get File dialog box. Therefore, even if you ask for the file again by the correct spelling, HyperCard retains the misspelled name and replies with the same Get File dialog box. To end this loop, you need to quit HyperCard and launch it again.
- As a minor but related problem, HyperCard occasionally mispells a filename displayed in a card. A name is not always spelled on a card as it is spelled on an A/UX disk or as passed to a HyperTalk Go command. This problem is only cosmetic but can cause confusion.

## 2. Protection from Internet worms

In response to the worm programs transmitted over some networks in November 1989, this release contains improvements to the programs `sendmail` and `fingerd` to prevent the spread of this menace.

A worm program, which differs from a virus program, is a parasite program that infests a host and consumes some of that host computer's resources without substantially changing the host. A virus attacks other programs and subverts them to the virus's purposes, which is primarily to propagate itself.

## 3. Public-domain software

To save disk space, the source code for public-domain software is not included in this release. In Release 1.0, public-domain software was included in the directory `/usr/src`.

## 4. The alarm on the Alarm Clock

The Macintosh desk accessories are available while you run A/UX Toolbox programs such as the `term` utility; you have access to these desk accessories from the Apple menu. The Alarm Clock desk accessory keeps time, but the alarm does not function properly as it does in the Macintosh Operating System. Though you may set the alarm, it will not go off at the specified time.

## 5. Security for `ftp`

The file-transfer program, `ftp`, has been improved to prevent users from anonymously destroying files that have write permission only for the root user. This prevention was accomplished by modifying the `ftp` daemon (`/usr/etc/in.ftpd`) to contain a list of the shells that `in.ftpd` can execute. The default list consists of `/bin/sh` and `/bin/csh`. To add other shells, create the file `/etc/shells` and enter the shells you want, including the shells `/bin/sh` and `/bin/ csh`. Only programs found in `/etc/shells` are executed by `in.ftpd`.

## 6. The FDHD (Floppy Disk High Density) drive

A/UX cannot take advantage of the 1.4 MB floppy disk drive included with some Macintosh computers. Under A/UX, this drive functions as an 800 KB floppy drive.

# Section 3

# Notes For Local System Administrators

## 1. Autoconfiguration

In this release of A/UX, the software drivers for slots and the Toolbox are preconfigured into the kernel. Because of this change, A/UX no longer runs `autoconfig` when it first boots as it did in the previous release. Thus, you won't see the system reboot following the check of the root file system the first time you start up A/UX.

When using `autoconfig`(1M), always use the `-I` option. This option ensures all appropriate nodes in the `/dev` directory are built as required.

## 2. `eupdate` and autorecovery

A day or two after running either `autoconfig` or `newunix`, or editing `/etc/HOSTNAME` or `/etc/inittab`, run the `/etc/eupdate` command. The `eupdate` command updates the system files necessary to maintaining correct records for the proper functioning of autorecovery.

## 3. The `dump` command

This release of A/UX includes a command called `dump.bsd`. This command has the same functionality as the BSD command called `dump`, but is not related to the AT&T System V command called `dump`. A/UX names the command `dump.bsd` to distinguish it from the System V command, and because it is a Berkeley utility. Note that in A/UX Release 1.0, the documentation referred to the command `dumpfs` although that command was not in the software.

## 4. Running `fsck` on multiple file systems

Release 1.1 changes the file `/etc/bcheckrc` to include the command `/etc/fsck -DB -q -p2`. The `p2` option is new in this release. The `p2` option specifies that when entering multi-user mode, `fsck` is to check only file systems with pass numbers of 2 or greater. You enter a file system's pass number in the last field of its entry in `/etc/fstab`. See *A/UX Local System Administration*, "Multiple File Systems and `fsck`" for more information.

## 5. Slice 31 and the Macintosh disk driver

Apple Computer highly recommends not making an A/UX file system using slice 31, which is the slice number representing the entire disk. A file system on slice 31 is not visible from the Stand-Alone Shell (`sash`), as evidenced by the inability to move to the disk with the `cd` or `chroot` commands.

Another reason not to use slice 31 is that it deletes the Macintosh disk driver partition from the disk. The absence of a Macintosh driver partition is a problem because some information remains on the disk that points to the driver that is now missing. If you use HD SC Setup to partition a disk, do not delete the partition labeled Mac Driver.

If you insist on creating an A/UX file system on slice 31, you can avoid trouble by understanding the following information summarized from *Inside Macintosh*, Volume V, "The SCSI Manager":

Physical block 0 (which `mkfs` does not touch) contains important information about the device, including the location of the Macintosh disk driver. Unless you invalidate the information in physical block 0, you will be unable to boot from this disk. To use slice 31 effectively (without a Macintosh driver) you must change the first two bytes of physical block 0 to zero. If you do not know how to do this, do not use slice 31.

# Section 4

# For Network Administrators

## 1. AppleTalk Network Support

The AppleTalk network is supported in this release with the following limitations.

### 1.1 Servers

It is possible for an A/UX node (such as the host) to run as many as ten local software servers using the `at_server` command. However, exceeding four servers per node severely affects system performance.

You should invoke AppleTalk software servers only while the system is in multi-user mode. Doing this prevents the `at_server`(1) command from receiving the hangup signal (SIGHUP) while the system is going from single-user to multi-user mode.

### 1.2 Using the latest printer drivers with AppleTalk

A/UX 1.1 is shipped with version 6.0.2 printer driver files. If later versions of printer driver files are installed in the Macintosh OS on a node of your network, then you need to also put those files in A/UX. Otherwise, printer driver battles can result when you print over the network from a Macintosh application.

Put updated printer driver files in the directory `/usr/lib/mac`. In Release 1.1, that directory contains the latest printer driver files, which are `LaserPrep, LaserWriter, AppleTalk ImageWriter`, and `LQ AppleTalk ImageWriter`.

Use the `hfx` utility to copy updated printer driver files over from the Macintosh OS. The default filenames that `hfx` gives to the files are the names expected by the AppleTalk A/UX software, so do not rename them. The names are `Laser Prep, LaserWriter, AppleTalk Imag`, and `LQ AppleTalk I`.

### 1.3 AppleTalk demands on streams buffers

AppleTalk makes strong demands on the resources of streams buffers. When AppleTalk is installed on a system and `autoconfig` is run,

these buffers are increased automatically to accommodate AppleTalk. However, if another streams application is also run, the buffers will need further increases. For this reason, if you plan on using AppleTalk and another streams application simultaneously, you need to change the kernel to increase the number of streams buffers. Increasing the number of streams buffers increases the portion of physical memory allocated to the kernel; as a result, you may find you need to add memory to your system.

Changing kernel parameters is described in Chapter 2 of *A/UX Local System Administration*. In encapsulated form, here's the process you need to follow to double the size of `NBLK`, the kernel parameter that allocates streams buffers.

While the system is in single-user mode, enter the following commands. These commands change the value of the NBLK buffer, build a new kernel, and boot the new kernel.

```
kconfig -n /newunix
NBLK=x
Control-d
autoconfig -IuS /etc/startup
sync
sync
sync
reboot
```

where *x* is two times the current value of the NBLK buffer.

## 1.4 Third-party NuBus cards for serial I/O

Third-party NuBus cards that provide serial I/O support (for example, the AST-ICP card) may use streams-based software drivers that differ from the drivers used by the two serial ports (the modem port and the printer port) on the main logic board. If you are using such a third-party board, your system has the following limitations.

- To produce the expected behavior, application programs that use the standard line discipline provided with `termio`(7) may need to push the line discipline module (`line`) on serial I/O ports using streams. See `streams`(7), `line_push`(3), and `line_sane`(1) for more information.

- The `getty`(1m) program, which establishes login sessions, is already modified to support streams drivers. Other programs, such as `cu`(1c), `tip`(1c), and `uucp`(1c), have not been modified to support streams.
- Serial Line/Internet Protocol (SL/IP) is not supported by streams.

### 1.4.1 AST-ICP card in serial mode

The cable delivered with the AST-ICP card does not support modem control. Rather, it has a two-headed DB-9 connector for external connections. Do not turn on modem control if this cable is connected for serial I/O. Instead, order a four-port DB-25 cable from AST or another supplier. The AST part for this cable is part #220130-002.

If you are using an AST-ICP serial card to control a modem, you need to use the `stty` command to turn on modem controls for the tty I/O drivers. The command you need to enter is

`stty -n /dev/tty`*n* `modem`

where *n* is the number of the tty port you're using. You need to use this command to establish modem control for each tty connected to a modem.

On an AST-ICP card, the serial ports on `/dev/tty2-5` may not be used as the system console.

Do not use the `getty` command to establish a terminal on an unconnected serial port on an AST-ICP card; if you do, the `getty` cycles, attempting to determine the port baud rate, and consumes CPU cycles. A `getty` is usually invoked from `/etc/inittab` and is off by default.

### 1.4.2 AST-ICP card in LocalTalk mode

A/UX 1.1 supports both serial mode and LocalTalk mode on the AST-ICP card, but not both modes at the same time.

Only one head of the two-headed cable provided with the AST ICP-2 card can be connected to the LocalTalk network. The LocalTalk DB-9 connector must be connected to the J-1 head of the AST cable.

## 1.5 ALAP types

The AppleTalk Link Access Protocol (ALAP) includes a type field for specifying the type of protocol of the receiving client. Under A/UX,

AppleTalk allows registration of 16 ALAP types in this field:

- Type 0 is reserved for control.
- Types 1 and 2 are registered by Datagram Delivery Protocol (`DDP`).
- Types 3 to 15 are available. Developers can call `lap_open`, followed by `lap_bind` to register new ALAP protocol handlers corresponding to these types. The `lap` interface for types 3-15 is experimental and as such may change in the future.

## 1.6 The `fwd_lkup` command

The `fwd_lkup` command does not work.

# Section 5

# Notes For Developers

## 1. Header files

A range of C header files is available. Two C header files, `/usr/include/net/un.h` and `/usr/include/net/unpcb.h`, have been moved since Release 1.0 and now reside in the directory `/usr/include/sys`.

## 2. POSIX conformance

This release of A/UX conforms to the Federal Information Processing Standard (FIPS) #15 POSIX. Changes made to Release 1.0 to acheive POSIX conformance may affect some applications. Those applications need to be modified or recompiled to run under A/UX 1.1. The applications that may be affected are ones that

- access the proc structure (defined in `/usr/include/sys/proc.h`) or the user structure (defined in `/usr/include/sys/user.h`)
- call `setcompat`(2) to set the process compatibility flag called `COMPAT_BSDPROT`
- call the ioctl(2) command `TIOCSCOMPAT`
- use the terminal interface provided by `termio`(7)
- set the save-on-stack flag to a value other than one

The following subsections explain these changes in detail.

### 2.1 User and proc structure changes

The size of the proc structure has been increased to allow the addition of the `p_sgid` field. This implements the saved set-group-ID feature defined in POSIX, Version 3.

The `p_ttyp` field has been moved from the user structure to the proc structure. This allows querying of other process tty group IDs.

## 2.2 Changes to `setcompat`(2)

In Release 1.0, the `COMPAT_BSDPROT` flag controlled both the function of multiple groups and the function of `setuid`(2) and `setgid`(2). Since POSIX conformance requires BSD-style groups and System V-style behavior for these functions, it was necessary to divide this functionality. The bit that previously controlled both functions now controls the function of multiple groups only (the more visible feature); old binaries that performed `setcompat(COMPAT_BSDPROT)` only get this feature when run under the new release.

In this release, `COMPAT_BSDPROT` represents both `COMPAT_BSDGROUPS` and `COMPAT_BSDSETGUID`. The value of the default process compatibility flag still includes `COMPAT_BSDPROT`.

## 2.3 TIOCSCOMPAT

`TIOCSCOMPAT` is an A/UX-specific ioctl(2) that supports the `TOSTOP` flag for job control. The value of the TOSTOP flag has been changed. The ioctl(2) interface remains unchanged, but the new kernel does not recognize the old value of the `TOSTOP` flag.

POSIX specifies that this flag may be set in the `c_lflag` of the `termios` structure that is implemented on top of the System V `termio` structure. The value of this flag had to be changed so it would not conflict with existing flags in `c_lflag`.

## 2.4 Terminal Interface with `termio`(7)

In Release 1.0, if the `CLOCAL` parameter of `termio` was enabled and `CLOCAL` was set, modem control was enabled. Under the same circumstances in Release 1.1, modem control is ignored.

In Release 1.0, if the `CREAD` parameter of `termio`(7) was not set, the terminal would still receive characters. Under the same circumstances in Release 1.1, the terminal will not receive characters. In Release 1.1, `CREAD` is set by default.

If an application modifies `termio` settings, the application should inherit the current termio settings using the ioctl command `TCGETA`.

### 2.5 `sigvec` Save-on-stack Flag

In Release 1.0, A/UX supported a 4.2BSD-style `sigvec` structure with a "save-on-stack" field. As required by POSIX, Release 1.1 supports a 4.3BSD-style `sigvec` structure that supports additional values in the save-on-stack field. *A/UX Programmer's Reference* specifies that this field should be set to 1 to enable the save-on-stack feature. Applications that incorrectly set the flag to a value other than 1 do not receive the expected save-on-stack behavior with this release.

## 3. Bourne shell function parameters

The Bourne shell was designed to prevent function parameters used within a shell script from colliding with the positional parameters of the script itself. This behavior is described in `sh`(1) in *A/UX Command Reference (M-Z and Games)*; however, Bourne shell functions do not currently behave this way.

Instead, a Bourne shell function call obliterates the current positional parameters. Two workarounds are available:

1. Run the script using `ksh`(1) instead of `sh`(1). The command `ksh`(1) is upwardly compatible with `sh`(1) and correctly handles function arguments with respect to the functional parameters.
2. Add code to the script to save the positional parameters where such saving is necessary.

## 4. Changes to the A/UX Toolbox

The A/UX Toolbox has been greatly enhanced since the last release. Most enhancements are documented in *A/UX Toolbox: Macintosh ROM Interface*; however, other enhancements have been added since the book went to press. This section describes those enhancements. It also mentions the change to the handling of toolbox routine names, which is described in detail in the manual. It is called to your attention here because it requires a slight modification to applications written for previous releases of A/UX.

## 4.1 Changes to toolbox names

Toolbox routine names are handled differently in the current releases of A/UX and MPW than in their previous releases. A/UX Release 1.1 makes the toolbox compatible with other Macintosh C compilers, including MPW 3.0.

In MPW 2.0 and A/UX Release 1.0, strings were passed in C-style and points were passed by reference, never by value. In MPW 3.0 and A/UX Release 1.1, strings are passed in Pascal-style and points are passed by reference, unless they are marked as VAR parameters as described in *Inside Macintosh*.

For complete information on the two versions of passing strings and points, see Appendix D of *A/UX Toolbox: Macintosh ROM Interface*.

You can retain the semantics of toolbox calls used in A/UX Release 1.0. To do so, you call the routine using lowercase rather than uppercase letters.

If you have a program that was written for Release 1.0, you must go through your code and change either the spelling of the toolbox calls or the format of the parameters. You can locate all instances of calls to relevant routines by using the script `/usr/toolboxbin/searchString` *filename*. For further information on the script `searchString`, see Chapter 3 of *A/UX Toolbox: Macintosh ROM Interface*.

## 4.2 `PBOpen`

If an application uses `PBOpen` to open a file for which a user does not have read permission, then the user also does not have write permission for the file. In this situation, a write-permission error occurs when the user attempts to write the file. Unless the application opens the file as read-only, the A/UX and Macintosh file permissions differ. Attempts to write the file then produce error conditions other than a write permission error.

## 4.3 `PBGetFPos`

The `ioResult` field returned from a `PBGetFPos` call contains an incorrect value if the call completes successfully. Specifically, `ioResult` contains a small positive integer rather than zero as expected. If the call completes unsuccessfully, `ioResult` contains the appropriate error code.

### 4.4 Size resources

In this release of A/UX, the toolbox uses size resources for Macintosh applications. Consequently, Macintosh applications that ran on the previous releases of A/UX using the `launch` command might fail to run under A/UX 1.1 due to an out-of-memory error. To avoid this error, ensure that the size resource for the application is set properly by doing the following:

1. Save a copy of the application.
2. Transfer the application to a floppy disk using the `hfx`(1) utility. You can learn how to use `hfx` by reading the appropriate section of *Getting Started with A/UX*.
3. Reboot the system to run the Macintosh OS and the Finder.
4. Put the floppy disk containing the application into the disk drive.
5. In the Finder, select the application and then choose Get Info from the File menu.
6. In the Info window that appears, set the value of Application Memory Size to 2000.
7. Close the window and eject the floppy disk.
8. Boot A/UX and use `hfx` to transfer the application back to the A/UX file system.
9. Launch the application with the `launch` command.

### 4.5 The `fcnvt` program

The `fcnvt` program, as documented in section 1 of *A/UX Command Reference* and *A/UX Toolbox: Macintosh ROM Interface*, is not part of this release.

## 5. The `uvar`(2) system call

The `uvar`(2) system call uses a variable structure defined in `/usr/include/sys/var.h`. The size of this variable structure has changed since the previous release, requiring you to recompile programs that use this structure.

# Section 6

# Replacement Pages

Some corrections to manuals are best made by simply replacing the page. Please replace pages 2-25 and 2-26 in *Local System Administration* with the pages in this section.

have `autoconfig` build them into a new kernel:

```
kconfig -n /newunix
NBUF=1000
NINODE=100
NFILE=100
NPROC=100
Control-d
autoconfig -IuS /etc/startup
sync
sync
sync
reboot
```

NBUF should not exceed 1000 unless you have 8 MB of main memory, in which case you can increase NBUF to 2500. It is safe to double other parameters (NINODE, NPROC, and NFILE), each time you run into the kernel-configured limits that initially caused you to increase these parameters.

You must reboot to put the new parameters into effect because `kconfig` changes the kernel file and not the currently running kernel. See `kconfig(1)` for an explanation of these parameters.

## 4.7 Initial processes: `/etc/inittab`

The system's topmost program is `init` ("initial process"). It is the first process to run after you boot the system. The `init` program makes shells and spawns processes. It gets all its process information from the file `/etc/inittab`.

The first command `init` executes is the `/etc/sysinitrc` shell program, which performs such basic functions as setting the system's clock and running `/etc/fsck` before you see the single-user mode shell prompt.

The second line in `/etc/inittab` specifies the default initial run level:

```
is:s:initdefault:       #First Init State
```

The run level in the `/etc/inittab` file is specified in the second field of each entry, in this case `s` for "single-user." Once the initial run level is determined, `init` processes only those `/etc/inittab`

entries whose run-level field is the same as the run level currently in effect. When you enter multi-user mode by typing

```
init 2
```

you invoke the `init` process with a run level of 2. In this case, `init` reads `/etc/inittab` and invokes every command that specifies a run level of 2.

If you check the contents of `/etc/inittab` on your system, you will see the processes invoked by `init`. For example, the following processes are in `/etc/inittab`. (However, the process names are preceded by an *id* and a *run-level*, as discussed in the next section.)

`/etc/bcheckrc`
: A startup script that runs `fsck` on the remaining file systems.

`/etc/brc`
: A startup script that sets the permissions on pseudo-ttys.

`/etc/rc`
: A startup script that mounts the file systems (if applicable) and performs some general housekeeping.

`/etc/getty`
: A process that enables logins on serial ports. See Chapter 3, "User and Group Administration," and Chapter 7, "Managing Other Peripheral Devices."

### 4.7.1 `/etc/inittab` entry format

Each line in `/etc/inittab` is an entry containing four fields separated by colons and followed by an optional number sign (#) and an optional comment. The format of each entry is

*id*:*run-level*:*action*:*command*

where

| | |
|---|---|
| *id* | unique entry identification |
| *run-level* | run level where the entry is processed |
| *action* | action to be taken with next field |
| *command* | command to be executed |

The following two lines are in the `/etc/inittab` file distributed with the standard A/UX system: