Macintosh Technical Notes

Developer Technical Support

#229: A/UX 2.0 Compatibility Guidelines

Revised by: Kent Sandvik & B. Winston Hendrickson February 1991
Revised by: B. Winston Hendrickson & Dave Radcliffe June 1990
Written by: Dave Radcliffe April 1989

This Technical Note describes details of the A/UX 2.0 implementation of which developers should be aware, so that their Macintosh applications also work properly under A/UX.

Changes since April 1989: This Note formerly described A/UX 1.1 Toolbox Bugs, but has been completely rewritten to cover A/UX 2.0 compatibility.

Changes since June 1990: Changes due to A/UX 2.0.1, also added some new important issues.

Introduction

A/UX 2.0 and 2.0.1 significantly improves support for Macintosh applications from version 1.1. The major components of the 2.0 Toolbox environment include:

- System Software derived from 6.0.5
- A/UX MultiFinder version 6.9
- 32-Bit QuickDraw
- File Manager access to UNIX, HFS, and AppleShare volumes
- Sound Manager
- Serial Manager

- Notification Manager
- Slot Manager

Most MultiFinder-compatible Macintosh applications and utilities should work unmodified under A/UX 2.0. Complete details concerning the A/UX Toolbox may be found in the *A/UX Toolbox: Macintosh ROM Interface* manual which is part of the A/UX documentation set; however, there are some additional subtle aspects of the A/UX environment of which developers should be aware.

General Compatibility Issues

The following items do not form a definitive list. Rather, they are indicative of common problems Apple encountered with applications while testing them under A/UX 2.0.

Do Not Access Hardware Directly

UNIX is a protected environment where the kernel arbitrates all hardware access. Applications should use the appropriate Macintosh Toolbox manager when it is available. Hardware developers may need to write a custom A/UX device driver to provide access to special hardware. Details on A/UX device drivers, including sample driver source code, is contained in the A/UX Device Driver Kit available from APDA.

Do Not Depend Upon How Things Work

Some applications rely on internal operating system details. This reliance has always been a bad idea, but applications are particularly likely to be bitten under A/UX because the internals are different. Following are some specific cases of which to be aware:

Event Manager

Some applications depend upon being able to access the event queue directly and patch into the `_PostEvent` mechanism. Under A/UX, the event queue is in the kernel and is not accessible to applications. Use if possible existing supported trap calls, instead of trying to access the event queue.

Memory Manager

Some applications work because the Macintosh OS uses a roving allocation scheme. Such applications free memory then use that memory in subsequent traps. A/UX is much more likely to reuse that memory immediately after it is freed (i.e., before the application can use it) than the Macintosh OS.

A/UX-Specific System Files

Some system files are A/UX specific and cannot be interchanged with the Macintosh environment. These files include the System file, Finder, MultiFinder, AppleShare, MacTCP and Patch files.

Privileged Instructions

Since privileged microprocessor instructions must be emulated in software, they execute much slower under A/UX than the Macintosh OS; therefore, using privileged instructions degrades application performance. Avoid a privileged instruction where a non-privileged instruction does just as well. For example, the non-privileged instruction `MOVE <ea>,CCR` can be used in almost all cases instead of the privileged instruction `MOVE <ea>,SR`.

Personal System Folders

Under A/UX, a user has the option to use a public System Folder ("`/mac/sys/System Folder`") which may be shared and changed by all users or employ a private System Folder to allow custom configurations. When the user logs in, A/UX makes two checks for the existence of a private System Folder. First, the environment variable `TBSYSTEM` may be set to the full

UNIX path of a directory to use as the System Folder. Second, if `TBSYSTEM` is not set, the user's home directory is checked for a subdirectory named "`System Folder`". If neither of these conditions are met, the public System Folder is used. Once a particular directory is selected, it becomes the "blessed" folder as long as the user remains logged in.

If an application has an installer utility, it should not place required files in the active System Folder; rather, it should use the System Folder only for configuration data. Required files should be kept with the application.

File Manager

The A/UX File Manager provides access to: UNIX® volumes, HFS volumes, MFS volumes, and AppleShare volumes. UNIX volumes may be a Berkeley 4.2 File System (UFS), a System V File System (SVFS) or a Network File System (NFS™) volume. In addition, since HFS is used as the foundation of the 2.0 File Manager, other external file systems following the Macintosh compatibility guidelines should work correctly under the A/UX environment. In fact, access to UNIX file systems is provided through a Macintosh external file system.

HFS Volumes

A/UX 2.0 allows only one HFS partition per volume; volumes with more than one such partition are not usable. In addition, HFS SCSI devices are only accessible to A/UX if they were accessible to the Macintosh OS when the system booted, and they have a valid partition map. Note that in the case of HFS CD-ROM discs, only the drive needs to be accessible, no volume needs to be in the drive.

UNIX Volumes (/)

Different physical UNIX volumes (including NFS volumes) appear to Macintosh applications as a **single** Macintosh volume named `/`. Moving files within the single logical volume `/` using `_PBCatMove` can fail with a `badMoveErr` if the move would cross physical volumes. It also means the free space reported on `/` may not be an accurate reflection of the space available for a file operation. The `_AUXDispatch` trap, documented in the *A/UX Toolbox: Macintosh ROM Interface* manual, with the `AUX_FS_FREE_SPACE` selector, can be used to accurately determine the free space on the underlying physical volume.

The `/` volume always appears as the boot volume and **always** contains the "blessed" folder.

Filenames

UNIX file systems, unlike Macintosh file systems, are **case sensitive**. Applications should be consistent in the use of case to avoid confusion. For example, the filenames "`Foobar Preferences`" and "`foobar preferences`" are two distinct filenames to UNIX. The A/UX 2.0 File Manager mediates this a bit. It first attempts a case sensitive filename lookup, and if that fails, it then tries a case insensitive lookup. The disadvantage is that not only is this slower, but also the File Manager may fail to correctly identify a single file from a group of files whose names differ only in case. Note that the `_Create` trap is an exception and only performs a **case sensitive** lookup when it checks for the existence of a file, thus an application may create "`Foo`" in the same directory which already contains "`foo`".

Not all UNIX file systems support 31-character filenames. The UFS file system on which A/UX 2.0 is based, supports 255-character filenames, but the older System V file systems used on A/UX 1.x systems only support 14-character filenames. Since these older systems are supported under A/UX 2.0, users may mount them as part of `/`. While the File Manager continues to allow use of 31-character filenames, remember that the underlying file system may truncate the name. For example, on an SVFS volume, the name `ThisNameWillBeTruncated` becomes `ThisNameWillBe`. This also means names like `ThisNameWillBeTruncated` and `ThisNameWillBeTruncatedAlso` may not be unique. You can avoid problems by ensuring that static filenames used by application code are unique in the first 14 characters.

UNIX Volume Pathnames

The slash (/) character is used by UNIX as a pathname separator. Its use is similar, but not identical, to that of the colon (:) in Macintosh file systems. This similarity can lead to ambiguity; for example, "`Font/DA Mover`" might be interpreted as "`:Font:DA Mover`". Again, the File Manager attempts to arbitrate. It accepts **both** UNIX and HFS style pathnames, and if it is trying to parse a pathname according to UNIX semantics and fails to locate a particular path component in the UNIX file system, it translates all slashes to underlines (_) and treats the string as a single name (e.g., "`Font/DA Mover`" becomes "`Font_DA Mover`").

If you must generate a full or relative pathname, it is best to use a colon as a pathname delimiter because this reduces the chance of ambiguity.

End-Of-Line Treatment

UNIX and Macintosh use different end-of-line characters (newline and carriage return, respectively). To allow both UNIX and Macintosh tools to manipulate files of type "TEXT," the A/UX File Manager transparently translates these characters when dealing with files residing on the / volume. Carriage returns are translated into newlines upon `_PBWrite` and newlines are translated into carriage returns upon `_PBRead`. Hence, Toolbox applications always see "`TEXT`" data in Macintosh format and UNIX applications always see "`TEXT`" data in UNIX format. To facilitate this transparency, it is important that applications set the file type immediately after a call to `_Create` (using either `_SetFileInfo` or `_SetCatInfo`).

File Permissions

Unlike AppleShare, UNIX file systems associate access permissions with files as well as directories. Directory permissions are mapped onto corresponding AppleShare file permission values and returned by `_GetDirAccess`. For unreadable files, the File Manager returns a special type and creator pair, namely creator "A/UX" and type "FILE" for data files, and type "EXEC" for executable files.

HFS-Specific File Information for UNIX Volumes

UNIX file systems lack certain HFS file attributes such as directory IDs and Finder information, which means files created outside of the Toolbox environment do not intrinsically have such attributes. When necessary, such information is provided by the File Manager through the use of a database in the blessed folder. When building its database, the File Manager synthesizes type and creator information for these UNIX files from the file's contents.

Directory IDs may, under some circumstances, change. For instance, if a particular UNIX volume is sometimes mounted in different areas of the / volume, the IDs for its directories change according to where it is mounted.

The File Manager tries to keep all files of type "`TEXT`" in "pure" UNIX format; consequently, if an application creates a file of type "`TEXT`" that has no resource information, then the type and creator association is maintained strictly through the database.

Miscellaneous UNIX Volume Details

The following caveats apply only to file accesses to the / volume:

- _Allocate and _AllocContig do nothing; applications should use _SetEOF instead.

- Do not directly access VCBs and FCBs. The UNIX external file system does not use these structures internally. Applications can use the File Manager calls instead (e.g. _GetFCBInfo).

- UNIX activity outside of the Toolbox environment may modify the / volume. The A/UX File Manager regularly changes the reported modification time of the / volume (as returned by _GetVolInfo) to stimulate applications to check if a directory they are displaying needs to be updated. Be sure to check for change the modification time of the particular directory before updating an application's display.

- Almost all File Manager traps intended for / may move or purge memory. The following list contains traps that could, now or in the future involve the Memory Manager to move memory (note that _Read and _Write do not cause any Memory Manager activity):

```
_MountVol                                                    _OpenWD
_SetDir
_DirCreate
_GetCatInfo
_SetCatInfo
_CatMove
_GetDirAccess
_GetFileInfo
_SetFileInfo
_Open
_OpenRF
_Create
_Delete
_ReName
_SetVol      (if HFS-bit is set, as this implicitly involves a _SetDir )
_SetFilLock
_RstFilLock
_InitFS
```

- *Inside Macintosh*, Volume V-380, The File Manager, states that an _Open (and _OpenRF) call using fsWrPerm and fsRdWrPerm is retried as fsRdPerm in the case of read-only folders. Under A/UX 2.0, such calls return fLckdErr if the user does not have write permission for the particular file. Also, if that file resides on a UNIX file system which has been mounted as read-only, a vLckdErr is returned instead of fLckdErr. Both these problems are corrected in A/UX 2.0.1.

Virtual Memory

The A/UX 2.0 Toolbox is a virtual memory environment and is currently limited to 16 MB. The size of the environment defaults to the size of physical memory, but can be set at login time using the `TBMEMORY` environment variable. For example, a value for `TBMEMORY` of 10m results in a 10MB toolbox virtual memory environment.

24-Bit Environment

A/UX 2.0 provides a 24-bit, single application environment which may allow some applications which are not 32-bit clean to run under A/UX. Applications are still subject to limitations regarding direct hardware access and use of privileged instructions, but this environment may allow users to continue using older versions of software. Developers must not depend upon this environment for future development. In this 24-bit environment, virtual memory is limited to 8 MB.

Slot Manager

When A/UX 2.0 boots it checks each NuBus card. If the card contains a `PrimaryInit` (and a `SecondaryInit`) record, A/UX tries to execute the code. If this code does not follow the general A/UX guidelines (not 32-bit clean, it calls traps not supported…) then the kernel gets a panic message and the whole system hangs.

If the PrimaryInit and the SecondaryInit are not present, A/UX assumes that the card initializes itself or does not require initialization. Test for A/UX using `Gestalt` if the `PrimaryInit` contains code that will not work under A/UX.

Video Driver

A/UX 2.0 only supports well-behaved video driver control codes. The supported calls are documented in `/usr/include/mac/video.h`. The driver does a sanity check of the passed control calls. If the call is not supported or accepted, the control call is never sent to the video card.

The only Apple MacOS video driver control code not supported by the A/UX video driver is the `GetGamma, csCode 4`. `SetGamma`, `csCode 8`, is supported from A/UX 2.0.1 forward.

Developers interested to apply for control calls that are accepted under A/UX should contact MACDTS. The request should provide information about what csCode number the developer would like to have assigned, what functionality the control code contains, and what possible side effects this video control code has on the system.

Sound Manager

MacOS 6.0.7 supports the new `_SoundDispatch` trap starting from 6.0.7 forward. This trap is not implemented under A/UX 2.0.1 - even if the basic System release is 6.0.7. This means that programs which only test for System release level for `_SoundDispatch` availability will break with A/UX 2.0.1. Always test for the availability of the `_SoundDispatch` trap itself.

SCSI Manager

A/UX 2.0(.1) does not support the SCSI Manager. Thus the `_SCSIDispatch` trap is not implemented.

Script Manager

Starting with A/UX 2.0.1 all the Script Manager functionality and traps are supported.

Vertical Retrace Manager

`_AttachVBL and _DoVBLTask` traps are not implemented. These traps have to do with managing independent VBL queues for each video device. The A/UX kernel isolates the application from these devices - it handles the interrupts directly, and provides virtual vertical retrace interrupts via Unix signals.

`_AttachVBL` and `_DoVBLTask` are mostly used by system level software and interrupt handlers. Also the low memory global `jDoVBLTask` is not initialized.

A/UX Startup process and MacOS Systems

A/UX is booted from a special A/UX Startup application from MacOS. Some may have problems to actually boot A/UX from the Macintosh environment. The problem has to do with the A/UX Startup program and the Startup tools. These tools (launch,fsck, etc) consist of position dependent code because they are compiled with the A/UX C compiler and moved to the A/UX Startup environment.

Because the base address of these tools are `0x9FC00` (about 639k) it means that any INITs or other patches which increase the system heap above this limit will break the A/UX Startup shell and the tools. It can't either be changed out in the field. The only realistic workaround for the moment is to use the A/UX installed MacOS System when booting the computer for A/UX use later. This problem is subject to change in future.

Further Reference:

- *A/UX Toolbox: Macintosh ROM Interface*
- *Inside Macintosh*, Volume V, Compatibility Guidelines
- *Inside Macintosh*, Volume V, The File Manager
- Technical Note #117, Compatibility: Why & How
- *d e v e l o p,* January 1990, “Compatibility: Rules for the Road”
- *UNIX Review*, June 1990, “UNIX as a Platform for Macintosh Applications”
- A/UX Device Driver Kit (APDA)
- *Designing Cards and Drivers for Macintosh II and Macintosh SE* , Second Edition