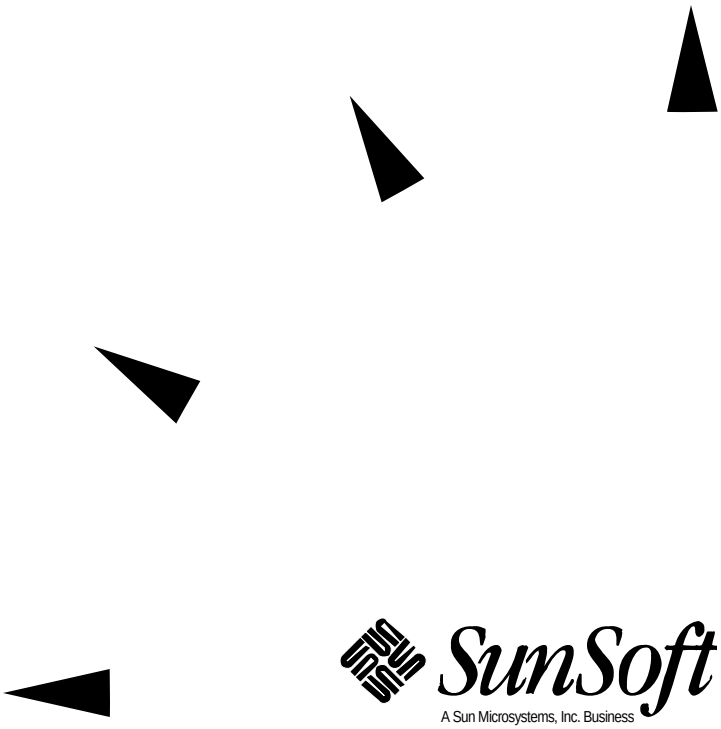
Solaris Binary Compatibility Guide



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

Contents

Preface

The SunOS 5.x operating system is based on AT&T's *System V Release 4.0* (SVR4), and therefore is not binary compatible with SunOS 4.x releases. This means that SunOS 4.x programs and applications based on those releases will not execute correctly when run directly on this release.

The *SunOS* and *OpenWindows Binary Compatibility Package*s allow SunOS 4.x based applications to run on the Solaris 2.x release, making them available for use before they are fully ported to this new release. Using these packages, well-behaved application binaries based on the SunOS 4.x release will run on this release without modifications or recompilation. These packages are provided as an *aid* in the transition to the Solaris 2.x release, and not as a substitute for porting applications to this operating system.

Scope of this Manual

In the context of this guide, "SunOS 4.x"is a universal term that includes these releases:

- SunOS 4.1
- SunOS 4.1.1
- SunOS 4.1.2
- SunOS 4.1.3

Who Should Use This Book

This guide is intended for application writers who want to ensure that their SunOS 4.x applications will execute easily on the Solaris 2.x release. It describes the binary compatibility package, what it does and does not handle, and how to install and use it. It also discusses specific areas to consider in developing an application or in evaluating how easily an existing SunOS 4.x application will execute on this release. Most importantly, this guide describes restrictions on this package; that is, areas where binary compatibility is not available.

A complete discussion of general compatibility issues can be found in the *Solaris 1.x to Solaris 2.x Transition Guide*.

How This Book Is Organized

This book is organized into three areas:

Chapter 1, “Introducing the Binary Compatibility Packages,” describes how to install these packages, and how to use them.

Chapter 2, “Binary Compatibility,” explains what the SunOS Binary Compatibility Package provides for your applications at the system interface level. This chapter also explains areas where binary compatibility is not available.

Chapter 3, “Window System Compatibility,” provides details on window system compatibility. This chapter discusses the binary compatibility available for the various window managers and toolkits available in the SunOS 4.x release.

Introducing the Binary Compatibility Packages 1

What the Binary Compatibility Packages Are

The Solaris 2.x and OpenWindows Binary Compatibility Packages are optional packages available with the Solaris 2.x release. They allow existing SunOS 4.x applications, including OpenWindows™ applications, to run on the Solaris 2.x release without modification or recompilation. It handles most binary interface discrepancies between the two releases transparently, providing an operating environment where SunOS 4.x applications can run properly.

Why the Binary Compatibility Packages Are Provided

These packages are only transition tools. They are intended for *end-user* environments, and allow existing SunOS 4.x binaries to run on the Solaris 2.x release. These packages are not provided as a *development* environment. All Solaris 2.x user application development should be done using the base Solaris 2.x environment.

Installing the Binary Compatibility Packages

These packages are optionally installable and require that the Source Compatibility Package also be installed. The SunOS Binary Compatibility Package is called `SUNWbcp` and the Open Windows Binary Compatibility Package is called `SUNWowbcp`.

1

See the *SPARC: Installing Solaris Software* or *x86: Installing Solaris Software* for details on installing optional packages and the *Solaris Source Compatibility Guide* for details on the Source Compatibility Package.

Note – If you customize your software installation, and want to use the binary compatibility packages, you must make sure you install both compatibility packages.

Using the Binary Compatibility Package

The binary compatibility packages are invoked automatically and transparently when running a SunOS 4.x application. You do not need to alter your application.

Some of the many differences between the SunOS 4.x and Solaris 2.x releases involve the location of commands and other interfaces. Because of these differences, the definition of your PATH environment variable affects how your applications run with the binary compatibility packages. The following section explains this in detail.

Resolving Path Names

There are two problems in resolving path names. First, an object may reside in a different place in this release than it did in the SunOS 4.x release. For example, a command found in `/usr/bin` in the SunOS 4.x release may be in `/usr/ucb` in the Solaris 2.x release. The Source Compatibility Package resolves this difference by setting up symbolic links whenever possible so that the applications, using the old path names, can access the files.

Second, many commands in this release are not compatible with their SunOS 4.x counterparts. As part of the Source Compatibility Package, a set of SunOS 4.x–compatible command binaries is also installed. If applications specify only the command name to be executed (and not a path name), the Binary Compatibility Package reads the PATH environment variable to find the correct command. For this reason the PATH variable should be set correctly before running the application.

For example, to get the default SunOS 4.x behavior, specify `/usr/ucb` before `/usr/bin` in PATH; otherwise, the default Solaris 2.x behavior is provided.

When a full path name is given, the Binary Compatibility Package interprets the path name as given. When the path begins with `/usr/5bin`, the default Solaris 2.x version of the command is executed; otherwise, the Source Compatibility Package version is executed. A relative path name is interpreted as given.

Performance

Running SunOS 4.x binaries using the binary compatibility packages requires more memory than running the same binaries under SunOS 4.x, or than running the same application after it has been ported to Solaris 2.x. Similarly, it requires more disk storage space and disk accesses. Using the binary compatibility packages will reduce the performance of the application and of the overall system.

Well-Behaved Applications

The binary compatibility packages work with *well-behaved* user applications. Applications that are not well-behaved can produce unpredictable results, and it is not recommended that the binary compatibility packages be used with these applications. Well-behaved applications are either statically or dynamically linked and meet the following requirements:

- Do not make any traps directly to the kernel
- Do not write directly to any system files
- Do not use `/dev/kmem`, `/dev/mem`, or `libkvm`
- Do not use unpublished SunOS interfaces
- Do not rely on customer-supplied drivers
- Do not rely on knowledge of the SPARC architecture

`/dev/kmem`, `/dev/mem` *and* `libkvm`

The contents of `/dev/kmem` and `/dev/mem` are release specific. Programs using `/dev/kmem` and `/dev/mem` are tied to a specific version of the operating system. Maintaining compatibility is not possible. Using `/dev/kmem`, `/dev/mem`, or `libkvm` routines in an application makes it non-portable.

1

Installing SunOS 4.x Applications

SunOS 4.x applications must be available on a Solaris 2.x system to use the binary compatibility packages. SunOS 4.x applications can be installed or mounted onto the Solaris 2.x system.

To install these applications, copy the executables in their original `a.out` format to the corresponding locations on the Solaris 2.x system. Applications can be copied using `cp`(1), `rcp`(1), `tar`(1), `cpio`(1), and so on. See the *SunOS Reference Manual* for information on these commands.

In most cases, you should be able to install a 4.x application from the original distribution medium. Install scripts may require that `/usr/ucb` precede `/usr/bin` in the `PATH` variable to perform correctly.

It is also possible to mount SunOS 4.x directories on a Solaris 2.x system and execute binaries contained in them using the Binary Compatibility Package.

If the corresponding location does not already exist, or if there is a conflict with Solaris 2.x files, you may have to create directories, rename files, or resolve name conflicts individually. The *Software Developer Kit Introduction* provides a description of the Solaris 2.x directory structure.

Shared Object Files

All `a.out` shared object libraries required on the SunOS 4.x release, except for the SunOS and the Open Windows libraries, must be moved to Solaris 2.x or mounted. These libraries can be copied to the `/usr/4lib` directory. These libraries *must* retain the same major and minor version numbers they had in the SunOS 4.x release.

The run-time linker searches the following paths (specified as colon-separated lists of directories) for the dependent libraries:

- Environment variable `LD_LIBRARY_PATH`
- Directories specified at link-time with the `-L` *dir* option to `ld`(1)
- Default directories: `/usr/4lib:/usr/lib:/usr/local/lib`

The `/usr/4lib` directory did not exist in SunOS 4.x. It is provided as a location for SunOS 4.x `a.out` libraries. This directory has been added to the default search rules so that compatibility package users can store their SunOS 4.x libraries there and be assured that the run-time linker will find them.

Binary Compatibility 2

This chapter describes the binary compatibility provided for system interfaces by the SunOS Binary Compatibility Package. The features and limitations described in this chapter are relevant to both window and non-window based applications. Chapter 3, “Window System Compatibility,” provides compatibility information specific to the OpenWindows Binary Compatibility Package.

What the Package Does

The specific incompatibilities that the SunOS Binary Compatibility Package resolves are listed below, and are then described in further detail in the rest of this chapter. Because some of these incompatibilities are resolved by the Source Compatibility Package, that package must be installed with the Binary Compatibility Package.

Together these packages do the following:

- Ensure that SunOS 4.x executables are properly linked and loaded at run time, using the correct set of libraries.
- Enable SunOS 4.x executable files in `a.out` object file format to execute on the Solaris 2.x release (where the default object file format is ELF).
- Provide SunOS 4.x behavior for all library routines and system calls, including those with new interfaces or different behaviors: even if the interface to or behavior of the calls differs, this mapping ensures that the SunOS 4.x binaries will get the expected behavior.

- Provide the SunOS 4.x signal handling behavior, and correctly map between SunOS 4.x and Solaris 2.x signals.
- Correctly map the set of supported SunOS 4.x `ioctl`'s to the corresponding Solaris 2.x `ioctl`'s, and ensure that the `ioctl` parameters are properly mapped.
- Ensure that commands and utilities are at the expected locations, or transparently perform the necessary mapping of the path names.
- Whenever possible, provide SunOS 4.x–compatible versions of system files when the file formats are different or the SunOS 4.x files have no counterpart in the Solaris 2.x release.
- Ensure that proper links are set up when system file names or paths are different.

What the Package Does Not Handle

This section describes areas in which incompatibilities remain when you use the Binary Compatibility Package. Applications using any of the items identified as incompatible can do any of simply fail, produce different results than when run under SunOS 4.x, or produce results inconsistent with those expected in SunOS 5.x.

`get`*XX*`by`*YY()* *Functions*

SunOS 4.x has two sets of functions to look up information about users, hosts, groups, and so forth. One set of functions is described in the Reference Manual pages `gethostent`(3N), `getnetent`(3N), `getnetgrent`(3N), `getprotoent`(3N), `getpublickey`(3R), `getrpcent`(3N), `getsecretkey`(3R), `getservent`(3N), and `yp_get_default_domain`(3N). These functions first search the appropriate NIS map. They search the corresponding file in `/etc` only if NIS is not running. The second set of functions is described in the Reference Manual pages `getpwent`(3V) and `getgrent`(). These functions first search a file in `/etc`. If the data is not found and NIS is running, these functions then query NIS. For statically linked applications, each of these functions behaves in SunOS 5.x exactly as it does running under SunOS 4.x.

For SunOS 5.0, and subsequent releases, all of these functions are changed to perform their functions through the Name Service Switch. At run time, the contents of the configuration file `/etc/nsswitch.conf` determine the sources and the sequence of look ups.

When an application has been statically linked on SunOS 4.x and is run on SunOS 5.x with the Binary Compatibility Package, these functions behave exactly as they do on SunOS 4.x. If the Name Service Switch of the machine running SunOS 5.x is not configured to behave as described in the first paragraph of this section, these functions in statically linked 4.x applications can behave differently than they do in native 5.x applications. These functions in dynamically linked 4.x applications running on SunOS 5.x behave the same as they do in native 5.x applications.

Object and `core` *File Formats*

The SunOS 4.x release uses the `a.out` object file format. The Solaris 2.x release uses the new Extensible Linking Format (`ELF`). Solaris 2.x programming tools (such as `cc`, `ld`, and `ar`) generate `ELF` files, and the default `exec` file format is `ELF`.

The Solaris 2.x `exec` identifies the executable file format (`a.out` or `ELF`) and uses the proper loading scheme.

The default `core` file format in SunOS 4.x is `a.out` and in the Solaris 2.x release it is `ELF`. When executing an `a.out` file in this release using the Binary Compatibility Package, the `core` files generated will be in `a.out` format. The compatibility mechanism is triggered only when loading and executing files with `a.out` binary format.

SunOS 4.x programs expecting an `a.out` executable file or `core` format will not work on files with other formats. For example, an application (such as the SunOS 4.x `nm(1)` command) expecting to access a file with the `a.out` format will not work when the file is in `ELF` format.

Programming tools and utilities in the Solaris 2.x release will not work as expected on binaries and `core` files in `a.out` format.

2

System Calls

There are major discrepancies between SunOS 4.x and Solaris 2.x system calls, such as different system call numbers and different interfaces or behaviors. Most of the differences between system calls in the two versions are handled by the Binary Compatibility Package through mappings and the use of alternate routines providing the expected interface and behavior.

The system call mappings can cause the output of the `truss(1)` program run under the Binary Compatibility Package to differ from your expectations. A system call name might differ, or a system call might not occur when expected. The arguments to the call or the returned value might be different. For example, in cases requiring path name resolution (see “Resolving Path Names” on page 2), an `open`() call to one file can actually open a different file.

Applications should invoke the system calls through the wrapper routines provided in `libc`. Do *not* directly invoke system calls through the trap mechanism, as it is then impossible to call the appropriate routine and map the data and system call number properly.

Incompatible Interfaces

Some system call incompatibilities are not addressed by this package because the calls have become obsolete, because they should not be used by user applications, or because they are included in other independent packages. The following is a detailed list of all such calls:

`acct`

The Solaris 2.x version of `acct` is compatible with the Application Binary Interface (ABI) and the *System V Interface Definition* (SVID); the SunOS 4.x version is not. The differences involve turning accounting off and the format of the accounting record. In the Solaris 2.x release, with accounting already on, a call to `acct` with a given path name turns accounting off. In the SunOS 4.x release in the same situation, the call will switch only to another accounting file, and will not turn accounting off.

audit, auditon, auditsvc, getauid, setauid, setaudit, setuseraudit

Auditing is not supported by the Binary Compatibility Package or by the base Solaris 2.x environment. Auditing is handled by an independent package in the Solaris 2.x release. The auditing facility provided by that package will not be binary compatible with SunOS 4.x applications.

getdirentries

This call has become obsolete. It is not supported in the Solaris 2.x release or the Binary Compatibility Package.

flock

A version of `flock` is implemented on top of `fcntl(2)` locking. It does not provide complete binary compatibility, and the following differences exist between this version and the SunOS 4.x version:

- `flock` requires the file to be opened for writing before requesting an exclusive lock.
- Read permission is required on the file to obtain a shared lock.
- Locking a segment that is already locked by the calling process causes the old lock type to be removed and the new lock type to take effect.
- Locks are not inherited by a child process in a `fork` function.
- Locks obtained through the `flock` mechanism under SunOS 4.x were known only within the system on which they were placed. This is no longer true.

The SunOS 4.x behavior is not available in the default Solaris 2.x release or with this package.

ioctl's

All `ioctl`'s related to `filio`, `sockio`, `streamio`, `termio`, `termios`, `mtio`, and `dkio`, as well as `ioctl`'s supported by the older version 7 and 4BSD terminal drivers are supported. Otherwise, only the `ioctl`'s pertaining to standard devices of Solaris 2.x platforms are provided. Discrepancies between the `ioctl` numbers (for the `ioctl`'s supported) in the two versions are handled transparently. The `ioctl` parameters are mapped whenever necessary.

2

The following SunOS 4.x ioctl's are incompatible with the Solaris 2.x release:

`DKIOCGCONF`

This `ioctl` is not available in this release, but it is supported by the Binary Compatibility Package. This `ioctl` is replaced by `DKIOCINFO`, which now includes the combined information of the SunOS 4.x `DKIOCGCONF` and `DKIOCINFO` structures.

`DKIOCGLOG`

This `ioctl` is not supported in Solaris 2.x. With the Binary Compatibility Package it returns `EINVAL`.

`DKIOCWCHK`

In SunOS 4.x this `ioctl` toggles the write check on the floppy device. With the Binary Compatibility Package, this `ioctl` does *not* toggle the write check on the floppy device, but it returns success.

`DKIOCSCMD`

This `ioctl` is available only for the `xd`(7), `xy`(7), and `ipi`(7) drives. This `ioctl` will fail for SCSI devices. Use the USCSI `ioctl` for these devices.

`_O_TIOCCONS`

This `ioctl` is obsolete and is not supported by the Solaris 2.x release or this package.

`_O_TIOCGSIZE`

This `ioctl` is obsolete and is not supported by the Solaris 2.x release or this package.

`_O_TIOCSSIZE`

This `ioctl` is obsolete and is not supported by the Solaris 2.x release by the Solaris 2.x release or this package.

`TIOCMODG`

This `ioctl` is obsolete and is not supported by the Solaris 2.x release or this package.

`TIOCMODS`

This `ioctl` is obsolete and is not supported by the Solaris 2.x release or this package.

kill

This Solaris 2.x system call behaves differently from SunOS 4.x. When a `-1` is supplied as the first argument, the calling process is also killed; this was not the case in SunOS 4.x.

pipe

The `pipe` system call in SunOS 4.x opens one descriptor for reading and one for writing. In the Solaris 2.x release it opens both descriptors for reading and writing. Because this difference should affect few if any binaries, no compatibility version has been provided.

ptrace

The optional *addr2* argument to the SunOS 4.x `ptrace` system call is not supported. In addition, the following requests are not supported in the Solaris 2.x release:

```
PTRACE_ATTACH       /* 10, attach to an existing process */
PTRACE_DETACH       /* 11, detach from a process */
PTRACE_GETREGS      /* 12, get all registers */
PTRACE_SETREGS      /* 13, set all registers */
PTRACE_GETFPREGS    /* 14, get all floating point regs */
PTRACE_SETFPREGS    /* 15, set all floating point regs */
PTRACE_READDATA     /* 16, read data segment */
PTRACE_WRITEDATA    /* 17, write data segment */
PTRACE_READTEXT     /* 18, read text segment */
PTRACE_WRITETEXT    /* 19, write text segment */
PTRACE_GETFPAREGS   /* 20, get all fpa regs */
PTRACE_SETFPAREGS   /* 21, set all fpa regs */
PTRACE_GETWINDOW    /* 22, get register window n */
PTRACE_SETWINDOW    /* 23, set register window n */
PTRACE_22           /* 22, filler */
PTRACE_23           /* 23, filler */
PTRACE_SYSCALL      /* 24, trap next sys call */
```

```
PTRACE_DUMPCORE         /* 25, dump process core */
PTRACE_SETWRBKPT        /* 26, set write breakpoint */
PTRACE_SETACBKPT        /* 27, set access breakpoint */
PTRACE_CLRDR7           /* 28, clear debug register 7 */
PTRACE_26               /* 26, filler */
PTRACE_27               /* 27, filler */
PTRACE_28               /* 28, filler */
PTRACE_GETUCODE         /* 29, get u.u_code */
```

swapon

This system call does not exist in the Solaris 2.x release or the Binary Compatibility Package and should not be needed by user applications.

vadvise

This system call does not exist in the default Solaris 2.x release or the Binary Compatibility Package.

Library Routines

The Binary Compatibility package provides a set of libraries compatible with the SunOS 4.x libraries. Therefore, even if the interface or behavior of a routine has changed in the Solaris 2.x release, user applications should not be affected when the Binary Compatibility Package is used.

xtab

Library routines that deal with the `/etc/xtab` file will not work correctly with the Binary Compatibility Package. See the `xtab`(5) and `exportent`(3) manual pages in the *SunOS Reference Manual.*

setlocale

The order of locales in the string returned by `setlocale`() differs between the SunOS 4.x and Solaris 2.x releases. This string is normally used by a subsequent call to `setlocale`(), and the order should not matter. Applications that rely on a specific order of locales may not be binary compatible.

Errors

All data mapping takes place in the binary compatibility library. If bad addresses are passed as an argument, the expected EFAULT error is not always returned. The data mapping library routines attempt to catch bad addresses, but since this is not possible at all times, accessing bad addresses may result in a Bus Error / Segmentation Violation.

Signals

The Binary Compatibility Package provides a signal handling mechanism compatible with SunOS 4.x. It also resolves the differences in signal numbers.

The SIGLOST signal is not supported.

System Files

Many system files have been renamed or moved in the Solaris 2.x release. Some do not exist in this release and others have a different format. Whenever possible, the Binary Compatibility Package addresses this problem by creating symbolic links, installing new files, or mapping the data.

Note – A portable program should avoid using system-dependent files. If a program must access such files, then it must do so using the standard interface routines provided. For example, rather than opening and reading the `/etc/mtab` file directly, the program should use the `getmnt()` family of routines to access the contents of the `mtab` file.

Accounting Files

Accounting files in the Solaris 2.x release have different formats than those in SunOS 4.x. An application accessing an accounting file will not be binary compatible.

/etc/exports
/etc/xtab

Exporting system files in the Solaris 2.x release is handled very differently than it was in SunOS 4.x. Binary compatibility cannot be provided for these files. This should not be a problem since user applications are not expected to access these files.

/etc/fstab
/etc/mtab

The name and format of these files have changed. The Binary Compatibility Package performs the mappings needed to give applications *read-only* access to these files. Applications requiring *write* access to these files are not binary compatible. Applications are not expected to write to these files.

Note – Symbolic links to these files are no longer supported. This is, if an application creates a link to `/etc/fstab` or `/etc/mtab`, and attempts to open the symbolic link, the `open` call fails.

/etc/gettytab

This file has no direct equivalent in the Solaris 2.x release and is not be provided as part of this package. Applications using this file will not be binary compatible.

/etc/passwd

In the Solaris 2.x release, the actual passwords are kept in a shadow file. Applications accessing the `passwd` file to obtain a password will not be binary compatible. All applications should access this file through the `getpw()` interface routines.

/etc/printcap

The `printcap` of SunOS 4.x is replaced by a directory tree in SunOS 5.x. The Binary Compatibility Package provides applications *read-only* access to the equivalent data for the host on which the application is executed. W*rite* access to `printcap` is not supported.

/etc/ttys

This file was obsoleted in SunOS 4.x. Because the `utmp` file format and access mechanism are very different, programs depending on the relationship between `/etc/ttys` and the `utmp` file will not work on the Solaris 2.x release. Use `ttyslot` for dynamically linked applications (will not work for statically linked applications).

/etc/ttytab

This file in SunOS 4.x replaced `/etc/ttys`. Obsolete in SunOS 5.x. No compatibility provided.

/etc/utmp
/var/adm/wtmp

These files hold user and accounting information for commands such as `who`, `write`, and `login`. In Solaris 2.x, each of these files has been replaced with a pair of files: `/etc/utmp` has been replaced with `/var/adm/utmp` and `/var/adm/utmpx`, and `/var/adm/wtmp` with `/var/adm/wtmp` and `/var/adm/wtmpx`. When an application opens one of these files, it receives the concatenation of its two new counterparts.

Note – Symbolic links to these files are no longer supported. This is, if an application creates a link to `/etc/utmp` or `/var/adm/wtmp`, and attempts to open the symbolic link, the `open` call fails.

Localized Applications

Localized 4.x applications will not run on a domestic SunOS 5.x regardless of static or dynamic BCP. See the appropriate localization documentation to determine if a particular 4.x application will run on the appropriate localized Solaris 2.3.

2

Warnings and Side Effects

File Descriptor Limit

The default limit on the number of file descriptors that a process can open is 64. The limit can be increased through the `limit` command of `csh`(1), the `ulimit` command of `sh`(1) and `ksh`(1), and the `setrlimit`(2) system call. If the limit on the number of file descriptors is increased to more than 256 a process running under either static or dynamic BCP can fail. The failure can happen even though fewer than 256 files are open. Most 4.x applications can handle only 256 file descriptors.

Device Numbers

In SunOS 4.x, the largest major device number that the system supported was 255. SunOS 5.x has a much larger limit on the major device number. A 4.x application running under either static or dynamic BCP will not be able to recognize a device number greater than 255, even though the device number is legitimate.

Window System Compatibility 3

The default window system for the Solaris 2.x release is Open Windows Version 3.3. SunOS 4.x applications are usually based on an earlier version of OpenWindows. This chapter addresses the binary compatibility available for window systems. Binary compatibility limitations discussed in Chapter 2, “Binary Compatibility” also apply to window-based applications.

OpenWindows Binary Compatibility

Binary Compatibility in the OpenWindows environment varies depending on the protocol or toolkit used. The following sections are standard guidelines for binary compatible OpenWindows applications, and describe the level of binary compatibility available for the X11 protocols and the various toolkits.

OpenWindows Version 2.0 (V2) and OpenWindows Version 3.0 (V3) were available with the SunOS 4.x release. Solaris 2.x contains OpenWindows V3.3 (V3.3) as the default window system. In general, OpenWindows V2 applications that ran on OpenWindows V3 should also run on OpenWindows V3.3. All limitations and problems noted in the following sections apply to OpenWindows V2 applications that are running on OpenWindows V3.3.

Linking OpenWindows Applications

When an application links libraries, all libraries must be dynamically linked.

For example, if `libxview` is dynamically linked but `libolgx` is statically linked against OpenWindows V2 libraries, this application will not run on V3. The user will see `ld.so` error messages similar to:

```
ld.so: call to undefined procedure _olgx_xxx from 0xf77906ec
```

If all libraries except `libc` are dynamically linked against OpenWindows V2, error messages such as the following will be seen:

```
ld.so: call to undefined procedure _strdup from 0xf778ea30
```

If an application used and modified OpenWindows V2 XView source and *dynamically* linked the libraries when the application was built, the application will not run with V3 XView. The modified XView source must be removed.

X11

In OpenWindows V3, the X11 protocol and XLib (the client library) were maintained with 100% backwards compatibility with OpenWindows V2. Both releases support Revision 4 of the X11 MIT protocol. The OpenWindows V3.3 release does contain a large number of bug fixes, however, that make it more protocol reliant than OpenWindows V2 was. If an application relies on an incompatibility with the protocol, it may no longer work.

Unsupported Methods

NeWS

The NeWS protocol is not supported in OpenWindows V3.3.

SunView

The Sunview protocol is not supported in OpenWindows V3.3.

Pixrects

Pixrects are not supported in OpenWindows V3.3.

Toolkits

XView

XView binary compatibility (taking any OpenWindows V2 application dynamically linked against OpenWindows V2 XView, pointing the application towards V3 XView libraries and expecting the application to run) is supported with two possible exceptions:

- Drag-and-Drop
- XView PostScript (XVPS)

XView 2.0 contained an interim API to shield clients from changes in the underlying Drag-and-Drop protocol. Applications that used this interface to "receive drops" will be fine when run against OpenWindows V3 libraries. However, if an application attempted to handle the protocol exchange itself, it will need source level changes.

XView applications that use the XVPS are not supported in Solaris 2.x.

OLIT

An OLIT application written to the OpenWindows V2 libraries will run without problems using the OpenWindows V3.3 libraries with one exception: Drag-and-Drop support. An OpenWindows V2 application that attempts to use the Drag-and-Drop facilities in the OpenWindows V3.3 libraries will not succeed: the drop will fail. Cut and paste will work without problems. Due to the changes in the underlying Drag-and-Drop protocol, an application needs source level changes to continue supporting Drag-and-Drop.

TNT and Lite

Applications written using any version of TNT or Lite toolkits are not supported in Solaris 2.x.

3

DeskSet

The DeskSet applications available with OpenWindows V2 and OpenWindows V3 are still available with OpenWindows V3.3. Running OpenWindows V2 or V3 Deskset applications on Solaris 2.x is not necessary or supported.

Index