Porting 4.3 BSD Programs to AIX Version 3

ABSTRACT

The intent of this document is to note the differences between 4.3 BSD or similar UNIX* operating systems and AIX** Version 3 that are relevant to the programmer porting 4.3 BSD applications to AIX Version 3 on the RISC System/6000**.

*UNIX is a registered trademark of AT&T.

**AIX and RISC System/6000 are trademarks of IBM.

© Copyright International Business Machines Corporation, 1990.

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice is preserved on all copies.

1. Introduction

AIX Version 3 (AIXv3) is a merge of many different industry-standard operating system functions and interfaces. When conflicts arose in merging these standards, conformance to official standards (IEEE POSIX 1003.1, ANSI C, FIPS) was considered essential and had the highest priority. Interface definitions supported by conformance test suites from non-standard groups (X/Open Issue 3) were given the next highest priority. Finally, functions in widespread use were considered; that is, System V Interface Definition (SVID) and 4.3 BSD. The goal was to minimize dual functionality and to merge overlapping functions into a single function that satisfied all the requirements. In some instances this was not possible or practical, and two similar but separate functions were provided. Many of the differences between AIXv3 and 4.3 BSD are a result of the conformance of AIXv3 to these new evolving standards.

Many features from 4.4 BSD have been included in AIX Version 3.2, but only differences between 4.3 BSD and AIXv3 are discussed in this document.

Where it was possible, merged system calls, libraries or commands were created to compensate for differences between standards implementations and interfaces such as System V and 4.3 BSD. Where a merge was not possible, one version was designated the "preferred" version, and it includes all of the capabilities of the other versions, though with different syntax.

In general, the designers of the merged system did not feel committed to convergence of system administration facilities in terms of preservation of command interfaces. However, they were committed to providing the administrative capabilities present in 4.3 BSD that were not previously present in AIX. See the */usr/lpp/bos/bsdadm* document from the *Base Development Libraries and Include Files* LPP.

2. Document Conventions

Italics are used for refering to commands by name (*nroff*), files and directories in the operating system (*/usr/man/cat?*), environment variables (*PATH*), and special facilities (*InfoExplorer*). **Bold** is used for section headings, user-typed commands (**catman -w**), options (**-S**), C source code, and stanzas. The character, ^, is the "caret" character and usually denotes a control sequence.

3. On-Line Documentation

AIXv3 contains a new method of accessing on-line documentation as well as retaining the *nroff*-formatted manual pages in */usr/man/**. Though *man*, *apropos*, *catman* and *whatis* are supported, the information is stored in a hypertext-structured database and the tool used to access this information is called *InfoExplorer*.

3.1. InfoExplorer

InfoExplorer is a graphical tool for a hypertext-structured database that contains all AIXv3 documentation. *InfoExplorer* can be accessed either through an application under AIXWindows or through a curses-based version. By default, *InfoExplorer* determines whether you are running AIXWindows, and starts up in the appropriate mode automatically. This can be overridden with command line options (**info -a** or **info -g**). To access the *InfoExplorer* tool under AIXWindows, the *AIXWindows Runtime* LPP (Licensed Program Product) must be installed so the */usr/lpp/info/bin/info_gr* file is available.

New manual pages can be added in flat text or *nroff*-formatted versions to be accessed by *man*. However, there are currently no facilities for adding documentation to the *InfoExplorer* database, or viewing non-database information with the *InfoExplorer* tool.

3.1.1. *InfoExplorer* Fonts

InfoExplorer adds */usr/lpp/info/X11fonts* to the X Window server's font path when invoked under AIXWindows or with **-g**. This is where additional fonts needed by the *InfoExplorer* program are shipped with AIXv3.

3.2. *man*

The *man* command accesses information from the *InfoExplorer* database. First it looks for flat text pages in */usr/man/cat?/*. Failing that, it looks for *nroff*-formatted man pages in */usr/man/man?/*. If a man page is found there, the *Text Formatting Services* LPP must be installed so that the *nroff* command is available for displaying the page. Since flat text and *nroff*-formatted pages are not shipped (with a couple of exceptions), the man pages you see will be from the *InfoExplorer* database. The *man* command accesses that subset of the *InfoExplorer* database that corresponds to traditional section 1-8 man pages.

There are some attributes of the AIXv3 *man* command that are different from the 4.3 BSD *man* command. The search mechanism of *man* is different in that the user will potentially get multiple hits on a single name (e.g. **man open**). There are also some formatting problems translating from the hypertext-structured database to flat text. This will show up when the *man* command is run and displays data from the *InfoExplorer* database.

The *man* command provides automatic caching of *nroff* man pages in */usr/man/cat?/*, but not *InfoExplorer* pages. The system administrator must write a shell script over *man* if it is desired to save the *InfoExplorer* database pages in flat text form. An example of such a shell script can be found in the paper titled *AIXv3 for 4.3 BSD System Administrators* in */usr/lpp/bos/bsdadm*.

The */usr/man/whatis* database is not shipped; **catman -w** must be run to construct this for use by *whatis* and *apropos*. The *InfoExplorer* database is structured into 3 sections: *commands*, *subroutines* and *files*. *man* and *catman* translate section numbers 1-8 into sections 1 (*commands*), 3 (*subroutines*), and 4 (*files*).

4. C Compiler

The default behavior of the AIXv3 C compiler is specified by the command options in a compiler configuration file, */etc/xlc.cfg*. There are several stanzas in the file; each one describes how the compiler behaves according to the command name used to invoke the compiler. The current valid compiler names are: *xlc*, *cc*, and *c89*. Other names may also be

made available, as noted below.

In all cases, the AIXv3 C compiler defines the following two tokens: *_AIX*, *_IBMR2*. To write code portable to the 6000 and other IBM AIX platforms, something like the following should be included in the source code:

```
#ifdef _AIX
#ifdef (_IBMR2)
      ...RISC System/6000 specific code...
#elif defined (_I386)
      ...AIX PS/2 specific code...
#elif defined (_U370)
      ...AIX/370 specific code...
#endif
#endif /* _AIX */
```

This can be seen by using the verbose flag: **-v**.

4.1. xlc and c89

When the C compiler is invoked by the command *xlc* or *c89*, compilation will be done in strict ANSI mode. Nothing outside of the ANSI C standard should be assumed to be available when they are invoked with no flags. If anything external to the standard is required - POSIX, X/Open, 4.3 BSD - you must use some combination of the following defines: *_BSD, _POSIX_SOURCE, _XOPEN_SOURCE,* and *_ALL_SOURCE.*

See the *bsdcc* stanza below on one possible method of doing this.

Note: if you invoke the compiler as *xlc* or *c89*, the preprocessor macro *_ANSI_C_SOURCE* is automatically defined and a language level of **ansi** is used.

4.2. cc

When the compiler is invoked as *cc*, the rules are relaxed somewhat to aid programmers that are porting existing code. In this mode, most K&R syntax, as implemented in the System V.2 *pcc*, is allowed. The stanza in */etc/xlc.cfg* does not define any preprocessor macros. By default, *cc* doesn't define any test feature macros to restrict the programming environment (*_ANSI_C_SOURCE, _POSIX_SOURCE, _XOPEN_SOURCE*), thus the header files are arranged to define *_ALL_SOURCE* which is the default AIXv3 programming environment (See the file */usr/include/standards.h*).

4.3. Setting up a 4.3 BSD Compilation Environment

The *RISC System/6000 Technology Book* refers to a *BSD* environment variable which can be set to cause 4.3 BSD behavior by the compilation environment. However, this is not currently supported by the AIXv3 C compiler. To get the compiler to behave as friendly as possible by default with code that has been written for a 4.3 BSD system, you can create your own stanza in the */etc/xlc.cfg* file. Below is a *bsdcc* stanza that works quite well:

```
bsdcc:  use         = DEFLT
        crt         = /lib/crt0.o
        mcrt        = /lib/mcrt0.o
        gcrt        = /lib/gcrt0.o
        libraries           = -lbsd, -lc
        proflibs    = -L/lib/profiled,-L/usr/lib/profiled
        options     = -H512,-T512, -qlanglvl=extended, -qnoro,
                      -D_BSD, -D_NONSTD_TYPES, -D_NO_PROTO,
                      -bnodelcsect, -U__STR__, -U__MATH__
```

Note: The **options** line must be a single line. It is split up above *only* for easier reading.

To use the compiler as *bsdcc*, link */bin/xlc* to */bin/bsdcc*. If you want to be able to execute *cc*, and have it behave like *bsdcc*, put a shell script somewhere in your *PATH* ahead of */bin*, call it *cc*, and have it invoke *bsdcc*.

When *bsdcc* is invoked, it will behave like *cc*, with the following exceptions:

1) *Libbsd.a* will be linked in ahead of *libc.a*. *Libbsd.a* has some functions in it that replace the identically named functions in *libc.a*. The *libbsd.a* functions behave more like the 4.3 BSD counterpart functions would whenever the 4.3 BSD behavior is different from AIXv3. Also, there are some obsolete functions in *libbsd.a* that no longer appear in *libc.a*. *Libbsd.a* is explained more fully in a later section of this document.

2) The preprocessor tokens *_BSD, _NONSTD_TYPES*, and *_NO_PROTO* will automatically be defined. The default preprocessor tokens *__STR__* and *__MATH__* will be undefined. This will help 4.3 BSD programs avoid the strcpy() and strcmp() subroutine macros being defined.

- The *_BSD* token is used within the *_ALL_SOURCE* environment to provide 4.3 BSD function that conflicts with standards.
- *_NONSTD_TYPES* is used because header files have changed due to standards compliance. This compliance can cause problems with non-compliant programs.
- *_NO_PROTO* is used to avoid ANSI C function prototyping which most older programs don't use.
- -U*__STR__*, -U*__MATH__* turns off inlining of string and math functions. These are inlined by default. See the header files *math.h* and *string.h* for a list of which functions are inlined by these defines. Because this avoids inlining of math and string functions, performance penalties may occur. If possible, the C source should be changed to avoid this.

3) **-bnodelcsect** turns off deletion of duplicate code section symbols. This option should eventually become the default.

4) The compiler's built-in preprocessor will not be invoked. Instead, the */lib/cpp* C preprocessor with a language level of extended will be used. See the section below on the AIXv3 C preprocessor.

4.4. -S Flag

The **-S** flag, which causes many existing C compilers to generate a re-assemblable machine language source file from a C program, is not supported under AIXv3. The XL C compiler does not go through any convenient intermediate step from which to generate such a file. There is the **-qlist** option which lists the assembly language along with a lot of other information. To see exactly what flags the compiler recognizes, invoke it with no arguments, e.g. type **cc**, or use **man cc**.

5. Fortran Compiler

The AIXv3 Fortran compiler, *xlf*, follows a similar code-generating path as the C compiler. It contains many options for turning on or off code optimizations. The 4.3 BSD Fortran compiler, *f77*, may be approximated through editing of the */etc/xlf.cfg* configuration file similar to the *bsdcc* stanza noted above.

Run-time error messages may be turned off by setting the environment **xrf_messages=no**.

6. C Preprocessor

There are three versions of the C Preprocessor on AIXv3 that may be used depending on your needs. There is no stanza specifically supporting the preprocessor in */etc/xlc.cfg*. A manner to specify which preprocessor to use is described below.

6.1. Built-In Preprocessor

By default when you compile a program on AIXv3, the AIXv3 C compiler uses its built-in preprocessor to preprocess the source file. The semantics of the built-in preprocessor does not support old System V.2 *cpp* rules that conflict with ANSI C.

6.2. */lib/cpp*

/lib/cpp is not the Kernighan & Ritchie *cpp*. Instead, it is similar to the AIXv3 C compiler preprocessor, but externalized, and uses a default language level of *extended*. For example, */lib/cpp* does not have to start preprocessor macros in the first column (mandated by ANSI C), but any text after an *#endif* is ignored (not allowed by ANSI C). Other differences between */lib/cpp* and *xlc -E -qlanglvl=extended* include:

- */lib/cpp* version has less error-checking because */lib/cpp* can be used to preprocess files other than C language source files.
- *xlc* version does line splicing always when **** is encountered. However, */lib/cpp* does line splicing only in argument text and replacement text in a macro expansion.

The flags **-tp -B/**path**/** can be used as compiler arguments to cause */*path*/cpp* to be used for preprocessing the source file (see below). For example, to compile a single C source program, *file.c*, using */lib/cpp* instead of the C compiler's built-in preprocessor, use:

cc -tp -B/lib/ file.c

See the *InfoExplorer* documentation in *Appendix A* of the *XL C Reference Manual* for more information about language levels.

6.3. Kernighan & Ritchie *cpp*

Because many programs, specifically X11 programs, rely on the behavior of the K&R C preprocessor, this has been provided in */usr/lpp/X11/Xamples/util/cpp/cpp*. To use this preprocessor during compilation, you need to have the *X11 Development* LPP installed and invoke it directly, use the flags:

-tp -B/usr/lpp/X11/Xamples/util/cpp/

with the compiler, or modify the */etc/xlc.cfg* stanza as mentioned above.

7. Linker (*/bin/ld*)

AIXv3's *ld* command operates differently from the 4.3 BSD linker. It supports an eXtended COFF object module format (XCOFF) that allows rebinding of object modules and resolution of exported references at load (*exec*()) time.

7.1. Replacing Bound Objects

Without changing argument conventions or other linkage elements, a new to create a new executable (*file.new*) by the following:

cc -o file.new file1.o file.old

7.2. Exporting and Importing Symbols

By default, the AIXv3 *ld* discards all functions within object files that are not referenced and not exported, even though other functions are referenced in that same object file. This could cause a problem for object oriented applications that may not have all the functions referenced at link time, and that cannot identify all these in an export list at link time.

The method to get around this is to give the linker the number of the object files that are about to be linked into the executable. For example: say you want to link *a.o, b.o, c.o,* and *d.o* together. Only *a.o* and *d.o* are referenced at link time, then

ld -e__start a.o b.o c.o d.o /lib/crt0.o -b gcbypass:4 -lc

will cause garbage collection on the first four .o's to be skipped (the **__start** has two underscores preceeding **start**). The */lib/crt0.o* module must be at the end to make sure it will have its unreferenced contents discarded. *ld* may need to be invoked separately, or the stanza in */etc/xlc.cfg* may need to be modified.

7.3. Symbol Resolution

The AIXv3 linker reads all objects into virtual memory and builds a hash index of all symbols. This eliminates the need for sorting of libraries using the traditional *lorder*, *tsort* and *ranlib*, though these are provided for compatibility. Each object specified (including libraries) is read into virtual memory; thus redundant libraries will cause redundant virtual memory copies which can lead to extremely large virtual memory images, and can exhaust virtual memory. Also, symbols are resolved in a first-come rule, so redundant library invocations cause unnecessary warnings about duplicate symbols.

See the */usr/lpp/xlc/bin/README.xlc* file for more information about the linker.

8. Debuggers

AIXv3 supports the *adb* and *dbx* debugging commands. There is also an X Window front-end to *dbx*, *xde*.

The AIXv3 compilers support the **-g** option for generating source level debugging information. The **-g** and **-O** options may be used concurrently, but it is not recommended as this will throw off *dbx*'s source level debugging information.

9. Porting C Code

This section describes the changes that have to be made to C code because of the nature of the AIXv3 compiler and Operating System.

9.1. Derefencing NULL Pointers

The word at memory location zero contains a zero. That means that code which has been *incorrectly* written to use a NULL pointer to represent a null string will work, but will not be portable to all UNIX operating systems, and may not work on future AIX systems.

9.2. Failing to Pass Parameters

Code that fails to pass parameters to functions that expect and use these parameters will probably not work on AIXv3. While it is a programming error to expect unpassed parameters to a function to be zero, this does work on some UNIX operating systems. On AIXv3, the value that gets `passed' is often **0xdeadbeef**. This is a result of security enhancements.

9.3. Alloca

The use of the function *alloca*() is discouraged since it is non-portable, but it is available. In order to reference *alloca*() in C code, you need to put a **#pragma alloca** at the beginning of the C program source. Though some compilers give a compile-time option of **-ma** for recognition of *alloca*(), this is not currently supported by the AIXv3 C compiler.

9.4. ANSI & POSIX Standard Gotcha's

ANSI C and POSIX 1003.1 define changes to the C language definition that may break programs that compile on older UNIX operating systems. Such differences include, but are not limited to:

- *sprintf*() returns *int* instead of *char* *,

- *dev_t*, *gid_t*, *ino_t*, *mode_t*, *mid_t*, *nlink_t*, *pid_t*, and *uid_t* are used for device, group, inode, mode, module, link, process and user types and id's instead of the shorts, ints and longs in 4.3 BSD (see *sys/types.h* for type definitions),
- signal handlers return *void* instead of *int*, (see *signal.h* for information),
- function prototyping [e.g. *fclose*(*FILE *stream*) instead of *fclose*()].

See *Appendix A, XLC Compiler Language Levels* in the *XLC Reference Manual* in *InfoExplorer* for more information.

10. Header Files

The header files in */usr/include* were modified to conform to X/Open Portability Guide, Issue 3, POSIX 1003.1 and ANSI C standards and merged to include both System V and 4.3 BSD function.

10.1. Standards Compliant Header Files

In order to be as compliant as possible with the POSIX 1003.1 and ANSI standards as well as X/Open, System V and 4.3 BSD interfaces, some of the header files under */usr/include* are different from many existing UNIX systems' header files. This may cause problems when porting existing programs to AIXv3. To help make this task easier, we've provided *#ifdef*'s throughout the header files which will make them look more BSD-ish when *_BSD*, *_NO_PROTO*, and *_NONSTD_TYPES* are defined. However, there are still incompatibilities with 4.3 BSD header files, and the rest of this section describes those differences.

10.1.1. Header Files with Minor Differences

The following header files have minor differences:

math.h	Many compilation errors on AIXv3 are due to old 4.3 BSD programs declaring functions explicitly while also including AIXv3 header files which again declare the functions. This can cause redeclaration errors. To fix, remove the explicit function declarations from the C source code.
net/if.h	*IFNET_SLOWHZ* has been replaced by *DOG_TICKS* in AIXv3.
netinet/tcp_debug.h	*tanames*[] and *tcp_debx* variables are not in AIXv3.
netinet/tcp_fsm.h	*tcpstates*[] and *tcp_acounts*[] are not in AIXv3.
netinet/tcp_timer.h	*tcptimers*[] is not in AIXv3.
pwd.h	*pw_comment* and *pw_quota* are not in AIXv3.
stdio.h	*sprintf*() returns *char** in 4.3 BSD, *int* in AIXv3 for ANSI compliance. Many of the problems mentioned for *math.h* also occur.
sys/ioctl.h	The *TIOCCONS* and *TOICSETT* ioctls aren't suuported in AIXv3.
sys/mount.h	The *m_qinod* element of the *mount* structure is not in AIXv3. See also information regarding *fstab.h*.
sys/times.h	The *times*() subroutine returns *clock_t* in AIXv3 with a *tms* structure of *clock_t* elements; it returns *int* in 4.3 BSD with a *tms* structure of *time_t* elements.

10.1.2. Header Files with Major Differences

Since AIXv3 implements a new Journalling File System (JFS), the filesystem header files used by the system are in */usr/include/jfs* and are different from 4.3 BSD filesystem header files.

a.out.h	AIXv3 uses an eXtended COFF (XCOFF) format which is different from the 4.3 BSD file format.
ar.h	AIXv3 has implemented a new archive structure to improve performance.
curses.h	Most pseudo-functions no longer exist as macros in *curses.h*. These pseudo-functions are now found in the library, *libcurses.a*.
fstab.h	AIXv3 does not support the BSD-style filesystem structure. Thus, any references to the *FSTAB* macro or the *fs* structure will fail. This has been superceeded by the the AIXv3 filesystem structure. See the *Filesystem Differences* section below.
nlist.h	AIXv3 uses an eXtended COFF (XCOFF) format which is different from the 4.3 BSD symbol table format.
regexp.h	AIXv3 has implemented the XPG3- and POSIX 1003.2-defined regular expression implementation which is different from the 4.3 BSD regular expression implementation.

The following header files exist on 4.3 BSD and AIXv3, but due to software or hardware specifics, there are too many differences to list individually:

sys/buf.h, sys/conf.h, sys/inode.h, sys/malloc.h, sys/mbuf.h, sys/param.h, sys/proc.h, sys/reboot.h, sys/seg.h, sys/systm.h, sys/tty.h, sys/user.h, sys/vnode.h

10.1.3. Header Files that do not Exist in AIXv3

The following 4.3 BSD header files do not exist in AIXv3:

disktab.h, lastlog.h, mtab.h, pcc.h, ranlib.h, stab.h, struct.h, sys/bk.h, sys/callout.h, sys/clist.h, sys/cmap.h, sys/dk.h, sys/dmap.h, sys/exec.h, sys/fs.h, sys/gprof.h, sys/map.h, sys/mtio.h, sys/namei.h, sys/nfs_defines.h, sys/quota.h, sys/syscall.h, sys/text.h, sys/trace.h, sys/vadvise.h, sys/vcmd.h, tzfile.h, and *vfont.h*.

11. Libraries

Since AIXv3 is compliant with POSIX 1003.1, ANSI C, and other standards, as well as System V and 4.3 BSD interfaces wherever possible, the behavior of some of the library functions in *libc.a* will not be consistent with the behavior of the same routines on a 4.3 BSD system.

AIXv3 supports both 4.3 BSD and System V compatible math libraries.

11.1. Standards Compliant Libraries

Because *libc.a* is not 4.3 BSD compatible, *libbsd.a* is provided. To obtain functions closer to 4.3 BSD behavior, link in *libbsd.a* before *libc.a*. Invoking the compiler as *bsdcc* (see the stanza above) will do this for you.

11.1.1. libbsd.a

Many of the functions in *libbsd.a* do not exist in *libc.a* because they are originally from the 4.3 BSD Multiple Precision Integer Arithmetic Library (*libmp.a*) which is merged with *libbsd.a* in AIXv3. These are:

fatal, fmin, fmout, gcd, invert, itom, m_add, m_div, m_dsb, m_in, m_mult, m_out, m_sub, m_trq, madd, mcan, mcmp, mdiv, mfree, min, mout, move, msqrt, msub, mtox, mult, omin, omout, pow, rpow, s_div, sdiv, tradd, and *xtom.*

Other functions in *libbsd.a* were not put in *libc.a* on AIXv3 because they are only partially supported or are considered obsolete. They are:

flock, ftime, timezone, valloc, vfork, and *vtimes.*

Finally, the following functions exist in both *libbsd.a* and *libc.a* because the behavior of the 4.3 BSD function was different in such a way that could not be resolved with merged functionality.

fcntl, ioctl, mktemp, nice, nlist, re_comp, re_exec, setpgrp, signal, sigvec, and *sleep.*

Here are some details on differences between the *libbsd.a* functions and the functions that are found on a 4.3 BSD system; and explanations of obsolescence:

fcntl — The 4.3 BSD semantics of *fcntl*() are fully supported but performed using multiple AIXv3 system calls. When you use *SETFL* to change the *FIONBIO* or *FIFOASYNC* attributes of a tty on a 4.3 BSD system, the attributes are affected on all file descriptors that are open on that device. In the AIXv3 kernel, the change only affects file descriptors that originated from the same *open*() call. To fix this, the *fcntl* function in *libbsd.a* performs the necessary *ioctl* calls to change the relevant attributes of all of the file descriptors that are open or will be opened on the tty device.

flock — On a 4.3 BSD system, whether a file is opened for reading or writing does not matter. All that is required is an open file descriptor, and a shared (*F_RDLCK*) or a exclusive (*F_WRLCK*) can be made. On AIXv3, in order to obtain a shared lock, the file must be opened for reading, and to obtain an exclusive lock, the file must be opened for writing. This implies that the file must be opened for reading and writing if a lock is to be upgraded from shared to exclusive. Also, on AIXv3, unlike 4.3 BSD, locks are not inherited. If a process holding locks calls *fork*(), the child process will not have any locks.

ftime — This function is provided for portability, but is considered obsolete. Use *gettimeofday*() instead.

ioctl — *FIOCLEX*, *FIONCLEX*, *FIONBIO*, *FIOASYNC* are fully supported, but are emulated with *fcntl*().

nlist — On 4.3 BSD systems, *nlist*() returns the number of "unfound namelist entries". On AIXv3 the libc.a returns a 0 on successful completion or a -1 if the file does not contain a valid namelist. The version of *nlist*() in *libbsd.a* behaves like the 4.3 BSD *nlist*() function.

pow — The Multiple Precision Integer Arithmetic library for 4.3 BSD is *libmp.a*. This library does not exist in AIXv3, but the MP library functionality has been merged with *libbsd.a*. Because the function *pow*() exists in both *libbsd.a* and *libm.a*, care must be taken to link with *libm.a* before *libbsd.a* to get the floating-point *pow*() function when both libraries are used.

re_comp, re_exec — These regular expression functions are provided for enhanced portability of old programs, however they are considered obsolete and are not enhanced with National Language Support (NLS), or POSIX and X/Open changes to regular expression syntax.

setpgrp — *setpgrp* may return errno values of *EINVAL* if the process group is less than 0 or is invalid, and *EACCESS* if the child has successfully exec'd already. It is otherwise compatible with *setpgrp*() in 4.3 BSD.

signal — This function differs from the 4.3 BSD *signal*() function in that it always make the signal restartable, rather than preserving the *SV_INTERRUPT* flag. The mask and other 4.3 BSD equivalent flags are preserved. For portability, the POSIX 1003.1 *sigaction*() from *libc.a* should be used.

sigvec — This function is provided for portability, but is considered obsolete. The POSIX 1003.1 *sigaction*() should be used instead. On AIXv3, the version of *sigvec*() in *libc.a* always assumes that the *SV_INTERRUPT* bit is set;

system calls are never restarted. The version of *sigvec*() in *libbsd.a* interprets the *SV_INTERRUPT* so that it behaves the same as the 4.3 BSD *sigvec*().

sleep — In *libbsd.a* on AIXv3, *sleep*() is a library function and can't operate the same way as 4.3 BSD. The *sleep*() function in *libbsd.a* approximates the 4.3 BSD *sleep*() function by, in effect, making *sleep*(sleepsec) become (*alarm*(sleepsec), *sigsuspend*(mask);).

timezone — This function is provided for portability, but is not in *libc.a* because it has not been adopted for standardization. POSIX 1003.1 and ANSI C provide *ctime*(), *mktime*(), *tzset*(), and the *tzname*[] and *timezone* variables. In *libc.a*, *timezone* is an external variable containing the difference in seconds between GMT and local standard time. In *libbsd.a* however, *timezone*() is a function that behaves like *timezone* on 4.3 BSD, except that if the environment variable *TZ* is set, the name of the timezone contained in *TZ* is returned.

valloc — This function is provided for portability, but is considered obsolete. The ANSI C *calloc*() can be used instead.

vfork — On 4.3 BSD systems, *vfork*() is like *fork*(), except that the parent process is suspended until the child process either exits, or execs another program. Also, any changes made to variables by the child that are not on the stack by the child are seen by the parent after it is restarted. On AIXv3, *vfork*() simply calls *fork*(). The inherent inefficiencies associated with *fork*() in the past are not a problem on AIXv3 since the child's memory is allocated via a lazy copy on read.

vtimes — The function is provided for portability, but is considered obsolete. *getrusage*() can be used instead.

11.1.2. libc.a

Some functions in AIXv3 are different than 4.3 BSD, and some functions from 4.3 BSD are missing from AIXv3. Specifically, quota subroutines are not currently supported in AIXv3, though the *EDQUOT* errno is supplied for both compatibility with 4.3 BSD and possible future extensions.

11.1.2.1. Functions that are different in AIX

setruid, setreuid, setrgid, setregid — These functions have additional restrictions where a process cannot reset its real (effective) UID (GID) unless the argument is the current real or saved UID (GID). See the *setruid* and *setrgid* man pages for details.

getpgrp — On 4.3 BSD, *getpgrp*() returns the process group ID of the process ID passed to *getpgrp*(). In AIXv3, *getpgrp*() returns the process group ID of the calling process with no argument available, as defined by POSIX.

malloc, free — *malloc*() and *free*() on AIXv3 are modeled on the algorithms from 4.3 BSD. AIXv3 does late allocation of pages so that attempts to allocate large sparse memory spaces will succeed. When AIXv3 detects that it is running low on paging space, it begins sending out SIGDANGER signals to processes warning them that page space is low. AIXv3 does not refuse new requests for page space at this point, but may end up sending SIGKILL signals to processes until it is below the low page space threshold. This is discussed in more detail by the *Paging Space Overview* article in *InfoExplorer*.

11.1.2.2. Functions in 4.3 BSD that are not in AIX

11.1.2.2.1. *getmntent*(), etc.

The NFS/BSD routines for manipulating */etc/fstab* and */etc/mtab*: *setmntent*(), *getmntent*(), *addmntent*(), *hasmntopt*(), and *endmntent*() do not exist on AIXv3. See the section below on *Filesystem Differences* for more information. In order to query the AIXv3 kernel about what is currently mounted, the following C functions could be used to emulate *getmntent*(), *setmntent*(), and *endmntent*() using the AIXv3 *mntctl*() function. They are not meant to be perfect replacements for the original functions; they provide an example of using *mntctl*(). In order to get this code to compile on AIXv3, at least part of the NFS/4.3 BSD *mntent.h* file will have to be copied.

```
#include <sys/mntctl.h>

static char *mntbuf = NULL;
static struct vmount *mnt = NULL;
static int entries_left;

FILE *setmntent()
{
        ulong size;

        if (mntbuf) {
                free(mntbuf);
        }

        mntctl(MCTL_QUERY,sizeof(size),&size);
        mntbuf=(char*)malloc(size);
        if (!mntbuf) {
                fprintf(stderr,"can't malloc %d bytes0,size);
                exit(1);
        }

        entries_left=mntctl(MCTL_QUERY,size,mntbuf);
        if (!entries_left) {
                fprintf(stderr,"error calling mntctl()0);
                exit(1);
        }

        mnt=(struct vmount*)mntbuf;
        return(1);
}

struct mntent *getmntent()
{
        static struct mntent mntstruct;
        static char *ufs = MNTTYPE_UFS;
        static char *nfs = MNTTYPE_NFS;
        static char remote_dev[MAXHOSTNAMELEN+MAXPATHLEN+1];

        if (!entries_left) {
                return((struct mntent*)0);
        }

        mntstruct.mnt_dir=vmt2dataptr(mnt,VMT_STUB);
```

```
        switch ((ulong)(struct vmount*)mnt->vmt_gfstype) {
            case MNT_JFS:
                mntstruct.mnt_fsname=
                    vmt2dataptr(mnt,VMT_OBJECT);
                mntstruct.mnt_type=MNTTYPE_UFS;
                break;
            case MNT_NFS:
                sprintf(remote_dev,"%s:%s",
                    vmt2dataptr(mnt,VMT_HOSTNAME),
                    vmt2dataptr(mnt,VMT_OBJECT));
                mntstruct.mnt_fsname=remote_dev;
                mntstruct.mnt_type=MNTTYPE_NFS;
                break;
            default:
                fprintf(stderr,
                    "warning: unknown fs type for %s0,
                    mntstruct.mnt_dir);
                break;
        }
        mnt=(struct vmount*)((mnt->vmt_length)+(ulong)mnt);
        entries_left--;
        return(&mntstruct);
}

int endmntent()
{
        mnt=NULL;
        if (mntbuf) {
            free(mntbuf);
            mntbuf=(char*)0;
        }
        return(0);
}
```

11.1.2.2.2. Other missing functions

The following functions are found on standard 4.3 BSD systems, but are not on AIXv3:

endusershell, getdiskbyname, getusershell, quota, setquota, setusershell, and *x3wind*.

The following have replacement functionality:

infnan	*infnan*() is not required because IEEE 754 is fully supported by the RISC System/6000 hardware and AIXv3 math and C libraries.
vhangup	*vhangup*() should be achieved through the *revoke*() subroutine.

11.2. Math Libraries

There are two math libraries in AIXv3. They are *libm.a* and *libmsaa.a*.

11.2.1. libm.a

The standard math library, *libm.a*, is compatible with the 4.3 BSD math library.

11.2.2. libmsaa.a

The *libmsaa.a* math library contains similar functionality to *libm.a*, but is written using the System V *exception* structure and *TLOSS*, *PLOSS*, *DOMAIN*, etc. return values. *libmsaa.a* is fully System V compliant and SAA compliant with minor exceptions. The exceptions are noted in the *Deviations of langlvl=saal2 from the SAA Level 2 C Standard* section of *Appendix A. Compiler Language Levels* in the *XL C Reference Manual*.

11.3. Shared Libraries

Many of the libraries in */lib* and */usr/lib* contain a mix of shared and unshared objects within each library archive. Because many people wish to use their own object modules/libraries, below is a discussion on how this may be accomplished on AIXv3.

11.3.1. Creating Shared Objects/Libraries

To create a shared object, you need to create an import/export list and then link the objects together using *ld*. One method for this is the following:

Compile object files

cc -c object1.c object2.c ... objectN.c

Create Import/Export list

/usr/ucb/nm -g object*.o | awk '$(NF-1) == "D" {print $NF}' > shobj.exp

Edit Import/Export list

Add the line:

#! shared_obj.o

to the file *shobj.exp*. As an import file, the first line specifies the pathname of the object which will be exporting the symbols. As an export file, the first line is just a comment.

Create shared object

ld -o shared_obj.o object*.o -H512 -T512 -bglink:/lib/glink.o -bexport:shobj.exp -bM:SRE -lc

- **-bM:SRE** specifies the resultant object is shared and re-entrant.
- The **-bglink** option defines the standard linkage routine for out-of-module references.

Note: Using the **-r** option during the *ld* phase will appear to work, but will return an error at run-time. Also, any other libraries should be linked in with the object to create an object without unresolved references.

11.3.2. Replacing Shared Objects/Libraries

The user may create their own functions with the same name as the library object modules, e.g. *malloc*() or *printf*(), and use them instead. This can be accomplished by binding the user version into a private copy of *libc.a*. The linker, */bin/ld*, has options for this. It is discussed more fully in the *ld* command in *InfoExplorer*.

A safe method to replace a system library (e.g. *libbsd.a*) is the following:

cd /lib
cp libbsd.a libbsd.a.system
cp /path**/libbsd.a libbsd.a.new**
sync
mv libbsd.a.new libbsd.a

Because *mv* is atomic, you'll either get one libbsd.a or the other.

11.3.3. Using User-defined Libraries

The environment variable, *LIBPATH*, is read by the */bin/ld* command and the *exec*() and *load*() subroutines. This environment variable acts as the search path for system libraries and is the suggested way to try out new libraries before replacing system libraries. It allows you to recover gracefully in the case of a corrupted library by unsetting the *LIBPATH* environment variable. One limitation is that any program with higher privilege (i.e. setuid/setgid programs) will ignore the *LIBPATH* environment and use the default library path.

Another method to replace/test a library on AIXv3 to mount your library over the system library. If your library then crashes the system, the mount will be lost and the system library will take over on reboot.

12. Profiling and Tracing Differences

Kernel profiling is not supported. Profiling is supported by various system libraries (as noted below) and is supported by compiling your code with **-p** or **-pg** options. See the *List of Process Management Services* in *InfoExplorer* for more information.

12.1. Profiling Subroutines and Commands

AIXv3 supports the 4.3 BSD *monitor*(), *monstartup*(), *moncontrol*() and *profil*() subroutines and *prof* and *gprof* commands. The *kgmon* command is not supported.

12.2. Profiled Libraries

The libraries compiled with profiling turned on (**cc -p**) are *libbsd.a*, *libc.a*, *libm.a* and *libmsaa.a*. These libraries are found in */lib/profiled*. The profiled libraries path searched is defined by the **proflibs** option in */etc/xlc.cfg*.

12.3. Trace Facility

The *trace* facility is also available for monitoring system events through various commands and subroutines. See the *Trace Facility Overview* in *InfoExplorer* for more information.

12.4. Crash Command

crash is an interactive command for examining various system/core image attributes.

13. Filesystem Differences

AIXv3 is based on a new filesystem named the Journaled File System. This uses database journalling techniques to maintain consistency, especially during system crashes. AIXv3 keeps its file system information in */etc/filesystems* instead of */etc/fstab*, and the list of currently mounted filesystems is not kept in any file; the kernel keeps track of that information. The information in */etc/filesystems* can still be queried using *getfsent*(), *setfsent*(), and *endfsent*(). See the *File Systems Overview* in *InfoExplorer* for more information.

14. Device Drivers

The procedure for writing a device driver to run under AIXv3 is different from other UNIX systems in order to support loadable, preemptable, and pagable device drivers. Under *InfoExplorer*, search on the string "*writing a device driver*" for more information.

15. TTY Differences

AIXv3 provides both a merged POSIX/SystemV.4/4.3 BSD line discipline and a strict 4.3 BSD line discipline. The merged line discipline is the default, and is needed for AIXv3 commands and subsystems to operate correctly. The merged discipline correctly handles all

4.3 BSD ioctls except the following: *TIOCEXCL*, *TIOCNXCL*, and *TIOCSTI*. The *TANDEM* bit in *sg_flags* and the *LTILDE*, *LMDMBUF*, *LLITOUT*, and *LETXACK* bit in the local mode words are not supported. If the 4.3 BSD line discipline must be used, care must be taken to replace the original line discipline upon completion of the activity and correctly catch job control signals to switch appropriately.

AIXv3 uses *terminfo* files instead of *termcap* entries. There is a compatibility library, */lib/libtermlib.a* and the file */etc/termcap* on 4.3 BSD systems can be found in */lib/libtermcap/termcap.src*.

16. Signal Differences

The number of signals has been increased to 63 to accommodate all the 4.3 BSD and AIXv2 signals. Traditional 4.3 BSD numbering is used for the first 25 numbers. This is also traditional numbering of the first 15 System V signals. The remaining numbers under 31 were used for signals judged likely to be masked by existing 4.3 BSD applications.

The RISC System/6000 currently doesn't generate an interrupt for floating point exceptions. Therefore, catching *SIGFPE* and calling a trap-handler is not supported.

17. Command Differences

In deciding how to merge commands, a small amount of syntactic change was considered better than having many duplicate commands. Where the same command name was used for entirely different functions in 4.3 BSD and AIXv2, both were given new names, and the one with the most impact to general users is linked to the 4.3 BSD name. In resolving flag conflicts, one of the primary criteria was to avoid breaking shell scripts. Determining which flag function was most likely to be used in a shell script was a subjective decision by a task force of both System V and 4.3 BSD users. The options least likely to be used in a shell script was typically reassigned to a new flag value. Where output of commands differed, merged output formats are provided, along with flags for reordering the output to match 4.3 BSD or AIXv2.

For those commands with conflicts that could not be resolved, the System V version is in the */usr/usg* directory and the 4.3 BSD version is in the */usr/ucb* directory. The preferred version is linked into the */usr/bin* directory. Non-conflicting 4.3 BSD commands are placed in the same directory that they would be in on a 4.3 BSD system. 4.3 BSD users can move */usr/ucb* in front of */bin* and */usr/bin* in their *PATH* environment to ensure 4.3 BSD command usage.

Also, AIXv3 does not currently support quotas which is why many quota commands are missing.

17.1. Commands not supported in AIXv3

The following commands are missing from AIXv3:

arff, bad144, badsect, dcheck, diskpart, drtest, error, edquota, gcore, kgmon, lpc, mset, quot, quotacheck, quotaoff, quotaon, repquota, systat, trsp, and *XNSrouted.*

The following 4.3 BSD commands have functionality similar to AIXv3 commands:

dmesg	*dmesg* is similar to the AIXv3 *syslogd* message log daemon.
dcheck, icheck	*dcheck* and *icheck* are not supported because their function has has been incorporated into *fsck* and because of the AIXv3 Journalling File System (JFS) type.
makedev	*makedev* is similar to the AIXv3 command *mkdev*.
newfs	*newfs* may be approximated by the AIXv3 command *crfs*.
rxformat	*rxformat* is similar to the AIXv3 *format* and *fdformat* commands.
tunefs	*tunefs* can be approximated through the *chfs* command and other filesystem utilities.

17.2. Commands with some Differences

The following commands have slightly different options and/or output due to the merged functionality and differences in the operating system:

crash, dumpfs, iostat, lint, lpd, lpq, make, mkhosts, mkpasswd, mkproto, pac, plot, ps, pstat, and *swapon.*

The following commands have small differences due to the fact that they have been merged from different environments:

cron, *fsck*, *ping*, *routed*,

find — The "fast-find" functionality, where *find file* looks for occurrences of *file* in a database of files, is not supported under AIXv3. There is no replacement command at this time. However see the *4.3 BSD System Administration for AIXv3* document in */usr/lpp/bos/bsdadm* for a replacement shell script.

lprm — *lprm* invoked without any arguments in order to delete the user's currently active print job is not allowed on AIXv3.

ps — A hyphen-prefixed flag will invoke the System V version. A non-hyphen-prefixed flag will invoke the 4.3 BSD version. Even so, certain 4.3 BSD options do not work the same under AIXv3.

- **g** - Prints global process listing; not all the user's processes.
- **k** - Not supported.
- **u** - Doesn't show niced processes.
- **x** - Format ordering is different.

size — The *size* command output has more fields than 4.3 BSD because of the XCOFF object module format and its new sections for dynamic linking and rebinding. Use the **-f** option to find out what the fields are. See the *xcoff.h* header file for more information.

stty — The *stty* command on AIXv3 requires different options from the 4.3 BSD *stty* command. For example, the **all** and **everything** options are not valid on AIXv3; instead, you can use the **-a** flag to get similar behavior. There is also the flag *iexten* which should be set so that non-POSIX behaviors are available, such as <control-V> <control-?>. Otherwise, you may set up an alias so that *stty* is */usr/lib/tty/stty-bsd*. That version of the *stty* command behaves more like 4.3 BSD.

tn3270 — *tn3270* on AIXv3 is a hard link to the *telnet* and *tn* commands. *tn3270* queries the */etc/tn3270.keys* file for 3270 key-binding information.

tset — Uses the */usr/lib/terminfo/?/** files instead of */etc/termcap*.

vmstat — The *vmstat* command does not list the following 4.3 BSD output fields: *w*, *at*, *de*.

17.3. Commands not Merged

The following commands' functions were not merged from different environments, thus there are two versions: one in */usr/usg*, and one in */usr/ucb*. 4.3 BSD behavior is received from the command in */usr/ucb*. There are links in one of the common binary directories (*/usr/bin*, */bin*) to the preferred version, so put */usr/ucb* in the first part of the *PATH* environment to get 4.3 BSD behavior from the following commands:

tr, *install*, and *nm*.

17.4. Different Files

map3270 — AIXv3 contains files which account for 3270 emulation. *3270.keys* is the key-binding definition for limited function terminals 3270 emulation. *3270keys.hft* contains the 3270 key bindings and color initialization for the IBM HFT (High

Function Terminal). *tn3270.keys* contains the key bindings for use with the *tn3270* command. The default 3270 key-bindings may be customized by putting a *.3270keys* file in the home directory, i.e. *$HOME/.3270keys*. An editable version may be copied from */usr/lpp/tcpip/samples/3270keys.hft*. Cntrl-T is the default escape sequence for AIXv3. To change this, the *TNESC* environment needs to be modifed. 4.3 BSD default is (Control-[right-bracket]):

setenv TNESC]

17.5. Shells and */bin/sh*

AIXv3 supports the Bourne Shell (*/bin/bsh*), C Shell (*/bin/csh*), and Korn Shell (*/bin/ksh*). The file */bin/sh* is a hard link to */bin/bsh*. If necessary, you may change */bin/sh* to point to the shell of your choice. Be warned, however, that although AIXv3 shell scripts specify the actual name of the shell command, many external scripts rely on */bin/sh* as the Bourne Shell.

17.5.1. C Shell

/bin/csh does not come with the base operating system. It is in the *Base Operating System Extensions #1* LPP. If */bin/csh* does not exist on the RISC System/6000, and you import a password file (as mentioned in the *4.3 BSD System Administration for AIXv3* document) and attempt to log onto an account which uses the C shell, you will either get */etc/motd* printed out and:

"Connection closed."

or you may get:

"Cannot set process environment : Permission denied"

Install the above mentioned LPP, and the logon should proceed.

18. X11

It has been noted by some users that a possible method of improving portability of X11 source to AIXWindows is to compile the code assuming System V environment, but also compile with *_BSD* defined and link with *libbsd.a*.

18.1. Compiling and Installing the X11 Clients in the *X11 Development* LPP

Some of the X11 clients and libraries normally found in the MIT distribution are different or do not exist as executable programs on AIXv3. Many of these clients are distributed in source form in the *X11 Development* LPP. These clients and libraries include:

aixwm, appres, bitmap, dogs, hellomotif, ico, listres, motifanim, motifburger, motifburint, motifgif, motifshell, mre, muncher, oclock, plaid, periodic, puzzle, uilsymdump, widgetView, xauth, xbiff, xcalc, xclipboard, xclock, xdm, xdpyinfo, xedit, xev, xeyes, xfd, xfontsel, xgc, xkill, xload, xlogo, xlsatoms, xlsclients, xlsdisp, xlsfonts, xlswins, xmag, xman, xmaze, xmh, xmsamplers, xprop, xrefresh, xscope, xset, xsetroot, xstdcmap, xwininfo, xwud, libmbX.a, libXaw.a, and *libXmu.a*.

To build these extra programs and libraries, install the *X11 Development* LPP, and follow the instructions in the *README* file in */usr/lpp/X11/Xamples*. *imake* is shipped in */usr/lpp/X11/Xamples/util/imake* and the Kernighan & Ritchie *cpp* is shipped in */usr/lpp/X11/Xamples/util/cpp*. See the discussion above regarding the *C Preprocessor* for more information.

18.2. X11 Font Differences

The directory where the X11 fonts are kept on AIXv3 is different from where MIT's X11 server expects to find them. MIT's X11 server looks in */usr/lib/fonts*; AIXv3's X11 server looks in */usr/lpp/fonts*.

18.3. X11 Client and Library Differences

18.3.1. xterm

Xterm is different from the MIT distributed *xterm* in the following ways:

Default Name
: When AIXv3 xterm looks in the *.Xdefaults* file, it looks for default listings titled *aixterm*, not *xterm*. You can change your entries in your *.Xdefaults* file, or you can specify the arguments: **-name xterm** to *xterm* on startup. Example: "**xterm -name xterm**".

Default Values
: There are other *xterm* attributes which have different default values such as the default font, where the scrollbar shows up, the pointer cursor, etc. Run "**xterm -help**" to get a list of the command line options which can override the default behavior.

TERM type
: *Xterm*, by default, starts up with *TERM* set to *aixterm*. To change this back to *xterm*, use the **-tn** flag.

Scrollbar
: The scrollbar on *xterm* shows up on the right side of the window when it is selected, and operates differently from the MIT *xterm* scrollbar. The flag to move the scrollbar over to the left is **-leftscroll**, but the scrollbar's operation is still different.

Tektronix Emulation
: AIXv3 xterm has a Tektronix emulation mode which is available through the undocumented **-t** option. This option is not supported nor documented and may not be available in future releases.

18.3.2. xinit

On AIXv3, *xinit* is a shell script which starts up the X11 server, the Motif Window Manager (*mwm*), and a couple of other clients. The shell script has been written to start up the X server, using the lowest unused *DISPLAY* number, and then start up the clients on that *DISPLAY*. Users that are used to the MIT *xinit* program should be aware of this since the AIXv3 *xinit* program will not necessarily accept the same command line flags.

18.3.3. uwm

Uwm, a window manager shipped with MIT's X11 distribution is not shipped with AIXv3. AIXv3 comes with *mwm*, and the source to *aixwm*. If you have the source to *uwm*, it is a simple matter to port it to AIXv3.

Table of Contents

Introduction 3
Document Conventions 3
On-Line Documentation 4
InfoExplorer 4
InfoExplorer Fonts 4
man 4
C Compiler 4
xlc and c89 5
cc 5
Setting up a 4.3 BSD Compilation Environment 5
-S Flag 6
Fortran Compiler 6
C Preprocessor 6
Built-In Preprocessor 7
/lib/cpp 7
Kernighan & Ritchie *cpp* 7
Linker (*/bin/ld*) 7
Replacing Bound Objects 7
Exporting and Importing Symbols 7
Symbol Resolution 8
Debuggers 8
Porting C Code 8
Derefencing NULL Pointers 8
Failing to Pass Parameters 8
Alloca 8
ANSI & POSIX Standard Gotcha's 8
Header Files 9
Standards Compliant Header Files 9
Header Files with Minor Differences 9
Header Files with Major Differences 9
Header Files that do not Exist in AIXv3 10
Libraries 10
Standards Compliant Libraries 10
libbsd.a 10
libc.a 12
Functions that are different in AIX 12
Functions in 4.3 BSD that are not in AIX 13
getmntent(), etc. 13
Other missing functions 14
Math Libraries 14
libm.a 14
libmsaa.a 15
Shared Libraries 15
Creating Shared Objects/Libraries 15
Replacing Shared Objects/Libraries 15

Using User-defined Libraries 16
Profiling and Tracing Differences 16
Profiling Subroutines and Commands 16
Profiled Libraries 16
Trace Facility 16
Crash Command 16
Filesystem Differences 16
Device Drivers 16
TTY Differences 16
Signal Differences 17
Command Differences 17
Commands not supported in AIXv3 17
Commands with some Differences 18
Commands not Merged 18
Different Files 18
Shells and */bin/sh* 19
C Shell 19
X11 19
Compiling and Installing the X11 Clients in the *X11 Development* LPP 19
X11 Font Differences 19
X11 Client and Library Differences 20
xterm 20
xinit 20
uwm 20