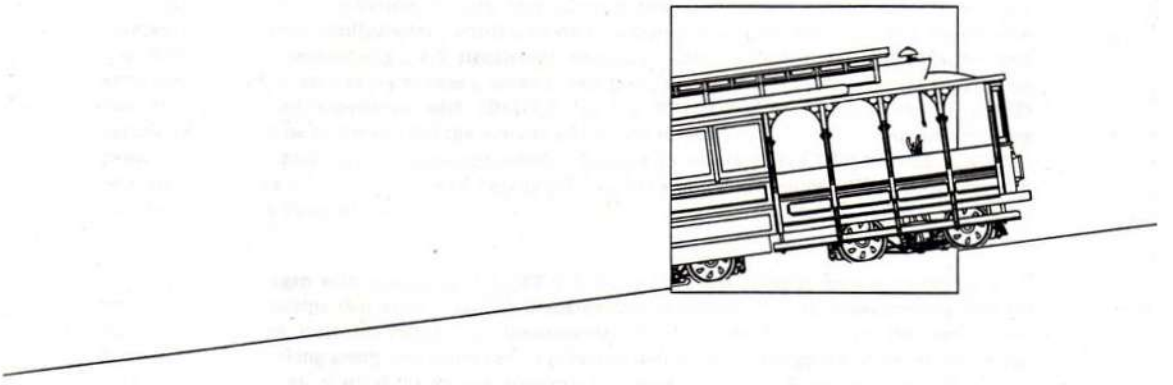
Convergence of AIX and 4.3BSD



Charles H. Sauer
Kathy A. Bohrer
Tom Lang
Conrad Minshall
Gary L. Owens
Kris Solem
Bruce J. Walker
IBM AES
D75/802, 11400 Burnet Road
Austin, TX 78758
(512) 823-3692

ABSTRACT

AIX started with a number of BSD features, e.g., 4.2 signals and concurrent groups[1]. Over time, additional features were added. Based on experience with these features, and experience with IBM/4.3 for the RT, it appeared that fairly strict BSD compatibility could be achieved. This paper describes methodology and decisions made in defining a convergence of BSD 4.3 and AIX.

1. *INTRODUCTION*

AIX is derived from System V, with unique enhancements in virtual memory support, memory mapped files, dynamic configuration, distributed system support and other areas[1]. AIX started with a number of BSD features, e.g., 4.2 signals and concurrent groups. Over time, additional features associated with BSD, such as pty's, select, sockets, sendmail, and symbolic links were added. Based on this experience, and experience with IBM/4.3 for the RT, it appeared that fairly strict BSD compatibility could be achieved, and the authors and others set out to define such compatibility. This paper describes methodology and decisions made in defining a convergence of BSD 4.3 and AIX. This convergence will be reflected in the AIX Family products[4] and the version of AIX to be provided to the Open Software Foundation.

The work began with evaluating AIX/RT 2.1.2 using the test suites prepared for IBM/4.3. A lengthy report describing that evaluation was prepared and reviewed. Though many missing features and problem areas were identified, no fundamental problems were found in the evaluation. Subsequently, a working group was convened to plan and define the convergence. After several week long working meetings, most of the known problems had been addressed. Since then, the specifics of the convergence have been iteratively defined and specified in draft technical reference documentation, which is to become the AIX Operating System Technical Reference and the AIX Operating System Commands Reference[2,3]. Summary matrices, indicating presence or absence of specific commands, system calls and library routines, are available in the AIX Family Definition Overview[4].

2. *GOALS/PRIORITIES*

Inevitably, in convergence work such as this, there will be a number of goals, conflicts amongst those goals and prioritizations which can be used for resolving the conflicts. Following subsections describe the goals of this work, in priority order.

---

*AIX is a trademark of International Business Machines Corporation.

Address correspondence to Charles H. Sauer, IBM Advanced Engineering Systems, 11400 Burnet Road, Austin, TX 78758. email: @cs.utexas.edu:sauer@ibmaus.uucp.

2.1. *POSIX and FIPS Compliance*

During the bulk of the work effort, P1003.1 was still evolving, as evidenced by the continuing revision of draft 12. Several of the participants in the BSD convergence effort were also active in the P1003.1 committee and other POSIX committees, so we were able to try to anticipate where things were stable and where things were changing. In spite of trying to track a moving target, P1003.1 compliance was considered mandatory. In some cases important BSD functions, e.g., job control, were considered to have been superseded by P1003.1 definitions, and only the P1003.1 definitions were provided.

The other P1003 efforts were clearly less well defined than P1003.1. Occasionally we would try to determine how a particular committee, say P1003.2, was dealing with a particular issue, but in general we did not try to anticipate how the other documents would turn out.

Also during the effort, we were anticipating what additional specifications would be included in the planned FIPS corresponding to P1003.1. In some cases, the anticipated specifications were used to resolve conflicts between BSD and System V that were not resolved in P1003.1. For example, we decided to make chown() a privileged operation on this basis.

2.2. *Base SVID Functionality*

Given the above assumptions with respect to P1003.1 and the anticipated FIPS, it seemed possible to preserve the remainder of the functionality in the BA_OS and BA_LIB sections of the SVID without significantly compromising BSD compatibility. Given the importance of this functionality, both in terms of compatibility and licensing issues, we considered preservation of this functionality mandatory. We would also refer to other SVID sections in resolving other issues, but compatibility with those sections was not as high a priority as the other goals listed below, particularly BSD compatibility.

2.3. *BSD 4.3 Compatibility/Completeness*

This was the real justification for the effort. We wanted to make the resulting system feel like a BSD system for both end users and programmers. The assertion was made that the result of the effort would be "99.44% compatible with BSD 4.3," though assertions were also made that the ".56%" was spent more than once! 4.3 compatibility was defined to include:

1. Inclusion, and compatibility with, the commands, system calls and library routines as defined in chapters 1, 2, 3, and 7 of the BSD manuals[5,6]. (See below for discussion of other chapters).

2. Compatibility with important, but undocumented, behavior of BSD functions. These undocumented features were either known from experience or determined by examining code. (In many cases we felt it necessary to diff BSD source code against existing AIX source. Where the source base was radically different, these diffs were of little value, but in a number of cases these diffs were usable to identify where changes/additions should be made.) We have chosen to document these characteristics in the AIX reference manuals.

3. Inclusion, and compatibility with, important functionality added in IBM/4.3.

In many cases the BSD manuals state that functions are obsolete. We usually resisted the temptation to take the manuals literally and included functions that are nominally obsolete. In a few cases we did choose to leave out functions that were considered to really be obsolete. Functions

documented in the BSD manuals, but not supported by IBM/4.3 were normally not included in the convergence.

2.4. *Compatibility with Existing AIX Interfaces*

For this to be a true convergence, we could not afford to significantly break existing AIX functionality; both the "99.44%" and the ".56%" assertions were made in regard to AIX compatibility as well. In many cases, there were no real conflicts, but redundancy was required to provide both BSD compatibility and AIX compatibility. For example, AIX already had a "tn" command which provides the functions of both the BSD telnet and tn3270 commands. We chose to add the telnet and tn3270 command interfaces, but keep the tn command interface as well. In the cases where there were serious conflicts, we had to compromise compatibility with either BSD or AIX, trying to assess importance of compatibility in either direction.

2.5. *Minimizing Redundancy*

In order to achieve the level of compatibility described above, redundancy is inevitable. However, redundancy is undesirable for several reasons:

1. New users will have difficulty deciding which commands and interfaces to use. What is the "preferred" or "strategic" interface?

2. Programmers will similarly have difficulty deciding what facilities to exploit and which ones to avoid. This is exacerbated by questions of portability to/from non-AIX systems.

3. Unnecessary maintenance effort is required, that could be better spent providing enhanced function.

For these and other reasons, we endeavored to minimize redundancy in the convergence. Except in cases where redundancy seemed inescapable, conflicts were resolved to provide a single merged definition of system call, library and command interfaces. Even where redundancy seemed inescapable, we designated one of the redundant functions the "preferred" version, and ensured that it include all of the capabilities of the other versions, though typically with different syntax. Typically, the source code for this preferred version can also be used to present the alternate interfaces, as well. For example, we use the same source code for the tn, telnet and tn3270 commands.

2.6. *Implementation Dependencies/Administrative Facilities*

In a number of areas, the implementation in the 4.3 kernel shows through at the kernel interfaces to an extent we felt was unnecessary, potentially precluding incorporation of new technologies. For example, we were in the midst of developing a new file system implementation that provides integration with virtual memory and recovery characteristics not present in the 4.3 file system; we could not allow convergence to preclude file system implementations other than the one supported by the 4.3 kernel. Though we preserve the system call and library interfaces associated with file access in 4.3, we do not, for example, preserve super block and inode structures specified in chapter 5 of the BSD Programmer's Manual.

In general, we did not feel committed to convergence of system administration facilities in terms of preservation of command interfaces. However, we were committed to providing administrative capabilities present in BSD that were not previously present in AIX. Where nominally administrative

interfaces (i.e., those in Chapter 8 of the BSD System Manager's Manual[7]) were likely to be used by end users on workstations, e.g., the ifconfig command, we provided those interfaces with BSD compatible syntax and semantics.

3. *SYSTEM CALLS AND LIBRARY ROUTINES*

Conflicts/Redundancies. The first fundamental issue we had to deal with was how we were going to handle conflicts and redundancies between BSD programming interfaces and existing AIX interfaces. We considered a variety of relatively complex approaches, but decided on a very simple one primarily to enforce discipline in defining a converged interface; if there was a sophisticated mechanism for retaining conflicting interfaces, then that mechanism would be used too freely. Rather, we wanted a mechanism of last resort, something usable when all else failed, but something not sufficiently attractive to be used frequently or pervasively.

We necessarily used different, but related mechanisms to deal with redundancy in function calls (including system calls), preprocessor definitions and commands. For function calls, we decided to have two auxiliary libraries, libbsd and libusg. In general, the preferred implementation of the function would be provided in libc, and an alternate implementation in either libbsd or libusg.

For example, in existing AIX and System V, the nice() function assumes nonnegative values for its argument, while the (obsolete) BSD nice() function assumes values from -20 to 20. nice() following the System V convention is in libc, and nice() following the BSD convention is in libbsd. In BSD, nice() is superseded by setpriority(), and the BSD definition of setpriority() is provided in the new libc in AIX.

In some cases, 1003.1 had made a clear choice of a preferred function, e.g., getcwd(), and a redundant BSD routine, getwd() in this case, was left in libbsd.

Functions which were not strict implementations of their traditional semantics would also be included in libusg or libbsd. For example, AIX supports a paging oriented fork() call which does not guarantee the parent/child precedence assumptions of the the BSD vfork() system call. A version of vfork() which simply invokes fork() is included in libbsd. Similarly, flock() as in BSD and locking provided by fcntl() in System V differ in regards to release of locks on the last close of a multiply opened file or the first close, respectively. flock() is provided in libbsd, but it is implemented as an invocation of fcntl().

As intended, both libusg and libbsd have very few elements. So far, we have placed no routines in libusg. libbsd includes flock(), fopen(), ftime(), getwd(), nice(), re_comp(), re_exec(), signal(), sleep(), valloc(), vfork(), vhangup(), vlimit() and vtimes().

BSD Environment Variable. As part of our compatibility objectives, we wanted programmers to be able to take programs developed on 4.3BSD, bring the source to an AIX machine and simply type "make" and have the program compile properly. With some routines present only in libbsd or redundantly present in libc and libbsd, we needed a mechanism to give preference to libbsd without having to change makefiles. We chose to modify cc to recognize an environment variable BSD. If cc finds this variable defined, then it will include libbsd in the ld search order prior to libc. Also, if environment variable BSD is defined, cc will define _BSD for cpp. Both of these effects can be overridden by explicit use of -U_BSD with the cc command.

Include Files. Another major area of redundancy and potential conflict is in the /usr/include hierarchy. In AIX 2, BSD compatibility in this area was originally addressed by a /usr/include/bsd

subhierarchy. This approach clearly requires makefile modifications (inclusion of -I/usr/include/bsd) and thus is undesirable. AIX 2.2.1 began merging the BSD include files into the standard hierarchy. Completion of this merging is part of the convergence definition. Where files and subhierarchies did not previously exist and they provide new definitions, it is straightforward to add the BSD files and directories. However, there are several cases requiring careful merging:

1. The same file, e.g., <sys/ioctl.h>, already existed in both AIX and BSD. In the case of ioctl.h, the old AIX ioctl.h was nearly empty and the new file is just the concatenation of the old AIX file and the BSD file.

2. Related, but different items are defined in different files. For example, the "whence" constants SEEK_SET, SEEK_CUR and SEEK_END in old AIX <unistd.h> correspond to L_SET, L_INCR and L_XTND in BSD <sys/file.h>. In this case, there is no conflict, and the definitions can remain in both places. However, to encourage use of a single set of constants (the ones specified in 1003.1), the BSD constants are only defined if the _BSD preprocessor variable is defined.

3. The same item is in different files. For example, NGROUPS is defined in <grp.h> in old AIX, but in BSD it is in <sys/param.h>. There is little else in <grp.h>, so <sys/param.h> includes <grp.h> in the converged definition.

Again, we let POSIX and ANSI X3J11 take precedence over our prior conventions and BSD conventions.

Errnos. We did not feel constrained to provide the same mapping of errno symbols to numeric values as in IBM/4.3. However, where the IBM/4.3 values did not conflict we AIX/RT assignments, we preserved the 4.3 assignments. Most of the difficult questions were resolving use of specific symbols where the two systems used different symbols for the same error. For example, AIX/RT used EAGAIN where 4.3 used EWOULDBLOCK. Given the POSIX usage of EAGAIN, we stayed with EAGAIN in these situations. However, EWOULDBLOCK is defined, to a unique value, so that programs which reference it, including switch statement references, will compile properly.

Signals. By AIX 2.2, all of the (traditionally bounded by 31) signal numbers had been used except for 5 reserved for future implementation of the BSD job control signals. However, there were other BSD signals not yet supported. For the RT implementation of AIX, we couldn't change the numbering without losing object code compatibility. However, for the PS/2 and 370 implementations which were not yet externally available, we could redefine the numbering. Also, the 1003.1 signal definitions allow for more than 31 signals. We agreed upon a new numbering, for implementations where we could change the numbering. For the first 25 numbers, we preserved the traditional BSD numbering. Most of the remaining numbers under 31 were used for signals judged likely to be masked (using pre-1003.1 interfaces), with two numbers left reserved, and the remaining signals were given numbers above 31.

Redundant Routines. There are a few cases where AIX and BSD had routines with identical function, but different names. The classical examples are index()/rindex() and strchr()/strrchr(). Much code designed to be portable from one environment to the other will try to manage these with #define's, but this is not appropriate for symbolic debugging. In some cases we chose to provide the BSD function in libbsd, in others, e.g., the above examples, to provide these as alternate entry points in the libc implementations of the prior AIX function, e.g., index() as an alternate entry point in strchr().

"Orange Book" Security. AIX 2.2.1 on the RT is designed to be certifiable at a C2 level. In some questions, perceived security issues forced the decision. For example, a traditional BSD vs. System V question has been whether chown() is restricted to processes with appropriate privileges. 1003.1 leaves the question open, to be selected by _POSIX_CHOWN_RESTRICTED. We chose to make chown() privileged, as we understood the interim FIPS would also do. We perceived minor security problems with vhangup() and provided a replacement call which could be invoked by the libbsd vhangup().

Other Semantic Issues. There are a number of areas where BSD provides upwardly compatible semantic extensions from AIX. For example, BSD restricts deletion of files in directories where the "sticky bit" is set. We chose to support these semantics. In other areas, there are conflicts of semantics, e.g., whether a created file should take on the group associated with the creating process, as in existing AIX, or the directory, as in BSD. In this particular example, AT&T and Sun had a good convention for resolving the conflict: where a directory has the set-gid bit set, then the BSD semantics are used; otherwise the group id of the process is used[8]. We adopted this convention, also.

4. *USER COMMANDS*

Conflict Resolution. As with function calls, we wanted a simple, last resort, mechanism for resolving conflicts. We chose a path based mechanism. Where there is a single version of a command, that command will typically actually reside in /usr/bin. For those commands that have a single version (e.g., vi), commands which traditionally appeared in /usr/ucb in BSD (e.g., /usr/bin/vi), there are symbolic links from the traditional path to /usr/bin (e.g., ln -s /usr/bin/vi /usr/ucb). For those commands with conflicts that we could not resolve (e.g., install), there is a superset version of the command that provides all of the functions (but not the syntax) of both versions. That superset version goes in /usg (e.g., /usg/install). A strictly BSD compatible version goes in /usr/ucb (e.g., /usr/ucb/install). By default the system is provided with symbolic links from the /usg version to /usr/bin (or wherever is appropriate, e.g., /etc/install). However, there is a script provided to reverse the sense of the links, i.e., make the /usr/ucb version the one that is symbolically linked into the presumed default path, and to remove the links entirely (for evaluating dependencies). On the order of 10 commands have these redundant versions, some of which are described below.

Scope. Our intention was to provide all commands, but we had to define what "all" meant. For the most part we left out the games and user contributed commands except where they had been previously with AIX and/or IBM/4.3, e.g., the Rand MH package and the "notes" package are provided. In some cases we considered a command to be truly obsolete or machine specific and omitted it. In other cases we considered a command not likely to be used by end users and redundant with other commands. In many of these cases we decided to provide a shell script in place of the command that would echo a message suggesting another command. For example, we do not provide the IBM/4.3 "MAKEDEV" command for adding special files, but there is a script version of MAKEDEV which suggests use of the "devices" command.

Name Conflicts. In some cases the same command name is used for entirely different functions. For example, "rsh" was used for "restricted shell" in AIX/RT prior to 2.2.1, while "rsh" is used for "remote shell" in BSD. When the remote shell function was added in AIX 2.2, it was named remsh. Starting with 2.2.1, the restricted shell is named "Rsh" and the remote shell is named "rsh" but also has the link "remsh". Since the restricted shell will primarily be invoked by /etc/passwd entries, it was agreed that it could be reasonably renamed. We encountered a few other cases like this, and were generally able to agree on renaming of one of the commands without perceived problems.

Flag Conflicts. Unfortunately, there were many commands where the flags were in conflict between AIX and BSD. In the majority of these cases, we were able to agree that one flag

interpretation was sufficiently unimportant that we could just change the flag used. One of the primary criteria was to avoid breaking shell scripts (and programs which exec commands) as far as we could judge the impacts of flag conflicts. For example, the –s flag on cat means that cat should be silent about missing files in the old AIX version, and means that cat should squeeze out blank lines in the BSD version. In this case we retained the BSD meaning and provided a –q (for "quiet") for the old AIX function, since it was not present in the BSD command. If –s had not been supported in the BSD sense, some programs, e.g., the man command, would have noticeably different results.

Other Syntactic Conflicts. In the tr command the flags were not in conflict, but the notation used for specifying the strings conflicted in whether ranges of characters were enclosed in square brackets, e.g., [a–z] as in AIX, or not, e.g., a–z in BSD. We decided to provide both /usg/tr and /usr/ucb/tr in this case.

Output Formatting. In some commands, e.g., df, the output formatting was significantly different and we had to agree on converged formatting that would give the user, and possibly shell scripts, output in a familiar and usable format. One pervasive aspect of this was to try to report file system values in 1K byte units where previously there had been mixed usage of 512 byte and 1K unit.

5. *ADMINISTRATIVE FUNCTIONS*

In some cases an administrative function, e.g., BSD's ifconfig, was provided by an alternate AIX function, e.g., netconfig. However it was judged that this function was sufficiently important in some sense that it be provided. In the ifconfig/netconfig case, one program now provides both the old netconfig interfaces and the traditional BSD ifconfig interfaces. Other administrative functions, e.g., vmstat and iostat, were missing and considered to be important; we decided to add these.

Other administrative functions were very specific to the BSD kernel implementation or to the BSD file system implementation. Commands with these dependencies were omitted in the convergence definition.

6. *SUMMARY*

The above discussion just gives examples of some of the interesting questions that had to be addressed. The general definition of the convergence is in the documents previously cited[2–4]. The manual pages in references 2 and 3 include compatibility discussions, indicating issues relative to both BSD and prior versions of AIX. However, it is intended that the above discussion gives a reasonable understanding of both the breadth and depth of issues addressed in the definition. Given the historical difficulties in resolving BSD and System V conflicts, we were surprised at how few fundamentally unresolvable conflicts were found.

Until the products implementing this convergence are generally available, it is impossible to really assess the effectiveness of the converged definition in meeting our original objectives. However, there is much room for optimism. First, there is the collective agreement of the body of people that worked on the problem. Second there is the evidence that even the intermediate level of BSD compatibility provided by AIX/RT 2.2.1 is very effective in providing a BSD-like environment. Test suites originally designed for IBM/4.3 are being used to test 2.2.1. Major programs, e.g., Apollo's Network Computing Kernel, which are configurable for compilation/execution in a both System V and BSD environments, are naturally built on AIX 2.2.1 as if it were really a BSD environment.

REFERENCES:

1. L.K. Loucks and C.H. Sauer, "Advanced Interactive Executive (AIX) Operating System Overview," *IBM Systems Journal 26*, 4 (1987).

2. IBM, "AIX Operating System Commands Reference," to appear.

3. IBM, "AIX Operating System Technical Reference," to appear.

4. IBM, "AIX Family Definition Overview," GC23-2002-0 (July 1988).

5. UC-Berkeley, "Unix User's Manual Reference Guide," Computer Systems Research Group, University of California, Berkeley (April 1986).

6. UC-Berkeley, "Unix Programmer's Reference Manual," Computer Systems Research Group, University of California, Berkeley (April 1986).

7. UC-Berkeley, "Unix System Manager's Manual," Computer Systems Research Group, University of California, Berkeley (April 1986).

8. D.W. Cragun, "Merged BSD and UNIX System V Semantics," *Sun Technology 1*, 1 (Winter 1988).