A Collection of NCUBE UNIX Utilities

Dave Tolle
Shell Development Company
3737 Bellaire Blvd.
Houston, TX 77025

1 The Problem

The AXIS operating system on the host processor of the NCUBE hypercube computer, as originally delivered by NCUBE, was similar to UNIX[1], but was lacking many of the most useful utility programs that UNIX users are accustomed to having.

We have therefore gone to some trouble to move a number of UNIX utility programs to the host processor of the NCUBE hypercube computer.

2 The Approach

When we started this work, there was no C compiler available on the NCUBE. (Now there is one, from Caine, Farber, and Gordon, Inc., and it works well, but it has some restrictions on addressing, because of the underlying Intel 80286 processor, that make it unsuitable for porting certain of the UNIX utilities.) Thus we sought out a C cross-compiler, and found one from AT&T.

We obtained the necessary UNIX source licenses and the sources for AT&T's System V/iAPX286 UNIX cross-development system.

That system is intended to run on a VAX running System V UNIX, but our VAX runs Ultrix, so we ported the cross-development system to Ultrix. This involved, among other things, making hybrid versions of certain UNIX utilities, hybrids with System V ancestry but adapted to run on a Berkeley UNIX file system. It was necessary to do this for **lex**, **yacc**, **m4**, **cut**, **sh**, and **make** before we were able to build the C cross-compiler and cross-linker, because, for each of these utilities, there was some small but crucial difference in behavior between the Ultrix (Berkeley UNIX) version and the System V version.

Once these hybrid utilities were in place, we were able to make the cross-linker and cross-compiler (which generates object code for an Intel 80286 processor that is assumed to be running System V UNIX).

Before the cross-compiler and linker were of much use, we had to have a library of UNIX system calls, such as access(), alarm(), chdir(), close(), creat(), execve(), exit(), fork(), ioctl(), lseek(), open(), pipe(), read(), sbrk(), stat(), and write(). We wrote these and others in C, in terms of NCUBE system calls. But, of course, the NCUBE system calls were not available through C, so we had to implement on the VAX a library of C-callable NCUBE system calls in System V/286 assembler.

It was then necessary to write certain C environment routines: the C startup routine (which gets linked with every main() routine), a command-line parser, and an error-handler.

With all of these tools at hand, we could compile and link a complete executable module, with appropriate code for accessing the NCUBE system services. Unfortunately, such a module is in a format appropriate not for AXIS but for System V UNIX.

Thus we wrote a translator to convert UNIX executable modules to NCUBE executable modules. This was a fairly straightforward, if tedious, task that required identifying the text and data sections of the object module and arranging them in a form suitable to the AXIS system.

Once all that was accomplished (after several months of effort) , we suddenly found ourselves able to port some powerful UNIX utilities to the NCUBE; in a day or so, more than a dozen were ported. As we built up the library of UNIX system calls for AXIS in the following months, we were able to port another dozen.

Through our UNIX sublicensing agreement with AT&T, we have since made these UNIX utilities available to NCUBE, which now includes them in its software distribution.

[1] UNIX is a trademark of AT&T

The Programs Ported

he UNIX utilities ported include

- **awk** — pattern scanning and processing language
- **banner** — display big text
- **cal**— print calendar
- **cmp** — compare two files
- **comm** — select or reject lines common to two sorted files
- **cut** — cut out selected fields of each line of a file
- **diff** — differential file comparator
- **egrep** — search a file for a pattern
- **factor** — factor an integer
- **fgrep** — search a file for a pattern
- **file** — determine file type
- **grep** — search a file for a pattern
- **make** — maintain, update, and regenerate groups of programs
- **m4** — macro processor
- **od** — octal dump
- **paste** — side-by-side catenation of files
- **sed** — stream editor
- **sort** — sort and/or merge files
- **split** — split a file into pieces
- **touch** — update access and modification times of a file
- **tr** — translate characters
- **tsort** — topological sort
- **uniq** — report repeated lines in a file
- **wc** — count lines, words, and characters in a file

4 Some Lessons

Even with a working C cross-compiler and a cross-linker, it was not simple to port non-trivial UNIX utilities to a non-UNIX system. Why was it difficult? Here are some of the reasons:

- AXIS and UNIX are different enough in their system services that not all of the UNIX system calls could be emulated with AXIS system calls. Some that could be emulated took considerable effort.
- When we started, AXIS had nothing resembling fork() or exec(). At our request, NCUBE provided similar services. Even then, we had to surround AXIS's nfexec() with hundreds of lines of C code to make a rough equivalent of UNIX's execve().
- AXIS file types are different from UNIX file types.
- AXIS file protection is different from UNIX file protection.
- AXIS signals are different from UNIX signals.
- AXIS user ids are different from UNIX uids; AXIS doesn't have the concept of group ids.
- AXIS error reporting is different from UNIX error reporting.
- AXIS memory allocation services are different enough from UNIX's sbrk() that it took a few hundred lines of C code to emulate sbrk().
- AXIS knows nothing about the ".." and "." directory notation.
- AXIS directories are not files; UNIX directories are, so UNIX directory-searching code does not work in AXIS. Porting **make** therefore required extra effort.
- The AXIS shell does not parse the command line in the way that the UNIX shell does; it strips off quotes before the C startup routine gets to see the command line, so there is no way to pass quoted strings containing blanks into **grep**, for instance.
- The AXIS terminal driver does not handle echoing, backspacing, and so on, so under AXIS that either has to be done by read() or not at all. (We don't do it at all.)

- The AXIS password file is different from UNIX's.
- The NCUBE host processor is an Intel 80286, basically a 16-bit machine; some of the UNIX utilities (notably **awk**) assume that integers are 32 bits.

Thus, although most of the two dozen UNIX utilities that were ported needed no change whatsoever to their source code, a few had to have their functionality slightly restricted, and a few required major changes to their source code. Many other utilities simply have not been ported, because of the considerable time and effort it would take to do so.

One interesting sidelight is that the attempt to port UNIX utilities revealed a few subtle bugs—some benign and some not—in the UNIX sources that probably had lain there unnoticed for years. On a non-protected-memory computer these might never be noticed, but the NCUBE host (an Intel 80286) enforces strict memory protection, refusing to let programs access data outside of their allocated regions.

5 Summary

We have succeeded in porting some useful UNIX utilities to a non-UNIX computer, with considerable effort. We think, though, that the improved computing environment has been worth the effort and has helped make an excellent parallel computer somewhat friendlier to the software developer.