1477

SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 9, 701-709 (1979)

Experiences with the UNIX Time-sharing System

JOHN LIONS

Department of Computer Science, University of New South Wales, Kensington 2033, Australia

SUMMARY

The UNIX* Time-sharing System has been in use at the University of New South Wales since 1975, and favourable experience has led to its widespread adoption on campus for both teaching and research. It has proved very adaptable to the university's needs, and very usable in a situation where staffing levels are critically low. One important application area, the teaching of Computer Science, is now firmly based upon the use of the UNIX system.

KEY WORDS UNIX Time-sharing Computer science

INTRODUCTION

There is a commonly held view, consistent with this author's own early experiences, that software obtained for free is likely to be worth what one has paid. Thus when the University of New South Wales negotiated a license for UNIX software from the Western Electric Company of New York at the end of 1974, there was no particular expectation on our part as to what might result. Certainly we did not anticipate the major changes that have since occurred in our local computing practice. This paper is a personal account, by a member of the Department of Computer Science, of some of the recent history of computing at the University of New South Wales. On our campus the UNIX system has proved to be not only an effective software tool, but an agent of technical and social change within the University.

The UNIX time-sharing system, for those who have not yet read Thompson and Ritchie's paper,[1] or better, seen the recent issue of the *Bell System Technical Journal*[2] devoted to the UNIX system, was written at Bell Laboratories to run on the larger models of the PDP11 class of computers. This time-sharing system was designed very much for the convenience of its first users, who happened also to be its designers and implementers. In practice it has proved to be convenient and effective for all its users, be they novice or expert. Running on a large PDP11/70 configuration, the UNIX system can serve up to 40 users before the response time becomes unacceptable (and it may be noted that most users of UNIX systems are intolerant of response times that users of some other systems would consider normal). The UNIX system may also be used effectively on much smaller computers. In fact, it can be run on a small PDP11/34 in single user mode to make a very attractive personal computer.

Western Electric distributes UNIX software without warranty or any after-sales support. There is no publicity and new releases outside the Bell System are made only very irregularly. (More than 3 years after the release of the sixth edition of the UNIX system, the seventh edition had still not appeared.) That a software package can achieve fame and success in the face of a policy of benign neglect argues for its being something special.

* UNIX is a trademark of Bell Laboratories.

0038-0644/79/0909-0701$01.00 Received 26 March 1979

Because it is available for educational purposes for only a nominal charge, its use is spreading rapidly within the academic community. UNIX software is spreading less rapidly among commercial and public institutions because of the substantial license fee ($20,000 U.S. or more for the first CPU), but the rate is increasing. Many computer users in the world at large seem to have been so influenced by their experiences with other software suppliers that they find it difficult to believe that a major software system could be delivered in a highly reliable form, where a single competent systems programmer might be able to provide all locally needed system support, without further assistance from the supplier. But the UNIX system proves it can be so.

We have found the UNIX system delightful and efficient to use, amenable and resilient to change and generally very easy to live with. The fact that its suppliers treat its users with an apparent neglect has not proved a disadvantage for us—in fact, we have come to see that as an actual advantage. UNIX software has an immediate appeal to people who like to build programs. It is like APL in that it possesses an extensive selection of 'building blocks' that may be readily strung together to create powerful program complexes. The important difference for the user is that, whereas with APL he manipulates arrays, with the UNIX system he manipulates text strings and the contents of files. Some of the features of the UNIX system that caught our immediate attention were:

1. The creation of small files and the initiation and termination of processes (program executions) are subject to acceptably low overheads, so that the user may use these freely in a wide variety of ways, without undue penalty. Files are referenced via an effective, multi-level system of directories. The details of how files are stored on disc are completely concealed from the user, so that he has no choice but to remain totally unconcerned.
2. The I/O buffering techniques are well adapted to smoothing the data flow to and from interactive terminals, and to sharing limited disc storage resources among many different users.
3. There is no job control language as such, but there is a free-form, no-nonsense command statement syntax. The command line interpreter, called the *shell*, interprets files of command lines that may come directly from a user's terminal or from a disc file; in general it treats the first word of each command line as the name of an executable file, that it endeavours to locate and invoke.
4. The shell will search several directories (including the user's current directory) in order to recognize a given command name. Because the contents of these directories may change dynamically, quite drastic surgery may be performed on the system during regular operation.

This last point was most important to us, because it allowed the development and testing of major software items on a machine that was already dedicated to production. Without this capability, our story would have been greatly different.

Our discovery of the UNIX system is of course not unique. It is already in use on many other campuses throughout the world, and has been in use for student instruction in places such as Harvard and U.C. Berkeley for several years. However, we believe that in the majority of cases, as in ours, its arrival has not been the result of any clear policy or managerial decision, but of the interest of one or two individuals. Once established, it has prospered and spread, even in the face of determined opposition from the computing establishment. We feel sure that the UNIX system is a computing phenomenon whose full influence has not yet been experienced.

A BRIEF HISTORY OF COMPUTING AT THE UNIVERSITY OF NEW SOUTH WALES

The University of New South Wales (UNSW) is currently (by a short head) the largest university in Australia with more than 18,000 students. The University has a strong technological bias with a large engineering faculty. The School of Electrical Engineering, which has an academic staff of more than fifty, includes the Department of Computer Science as one of five departments.

In 1974 the UNSW Computing Services Unit (CSU) replaced its 1965-vintage IBM 360/50 computer with a CDC Cyber 72-26 running under KRONOS. The Cyber is used to serve about 100 teletype-like (300 baud) interactive terminals and twelve PDP11/40 computers, that are distributed among the principal buildings of the campus. The latter, that serve primarily as remote-batch job-entry stations, communicate with the Cyber via 4800-baud synchronous lines, using a baroque communications protocol that imitates CDC's User 200 terminal. Each batch station is under the day-to-day management of an academic department in the same building.

While seven of the PDP11/40's were purchased in a minimum configuration (16 Kbytes of memory), five systems were configured with 128 Kbytes of core memory, a DJ11 terminal multiplexor and three RK05 disc units each, so that they would be able to provide local processing as well as to serve as remote-batch job-entry stations. The Department of Computer Science, that had been unsuccessful at that time in obtaining a computer for its own use, was given control of the 'large' batch station in the School of Electrical Engineering. This was the first to go on-line to the Cyber, using a U-200 emulator that runs under the RSX-11D operating system and that was written locally by Digital Equipment Australia. This emulator did not work particularly well at first, but it did at least work (no mean feat, as anyone who has tried to understand the official U-200 documentation will know). Three problems were soon painfully apparent: the batch station needed constant operator attention; throughput was only half the anticipated value; and the excess computer power that should have been available was firmly locked up by the RSX-11D operating system and was inaccessible (at least so it seemed) until such time as we purchased a second operator's terminal.

The stage was thus set for the arrival of our copy of the UNIX software. Just two evenings of experimentation were sufficient to show that we had stumbled on something rather interesting and that the UNIX system would have substantial advantages over RSX-11D in meeting the requirements of the Department of Computer Science for its own facilities. Our problem became 'How to arrange to run UNIX during the day as well as the evening when the remote-batch service was not required?' The answer obviously was to write a U-200 emulator that would run under the UNIX system. A group of four enthusiasts led by Ian Johnstone accepted the challenge and were able to develop a prototype system in 2 weeks. Four weeks later RSX-11D was displaced from the Electrical Engineering batch station, never to return.

While the initial version of the new U-200 emulator worked about as well as the one it had displaced, many improvements were clearly possible and were soon in hand: spooling of jobs relieved many of the operational headaches relating to job input; various operator type-ins required by real U-200 operators were automated, and the shared (never dedicated) operator's terminal became increasingly available for other use; the peripheral device drivers were improved to drive the card reader, the line printer and the link to the Cyber at their full rated speeds. (The latter was a significant achievement: the protocol followed by the U-200 is so complex that even the writers of the manufacturer's manuals do not seem to have fully understood it.) Ian Johnstone discovered that, without insight and careful software design, much of the time on the Cyber link could be wasted by redundant and unnecessary message traffic. A period of experimentation and observation eventually resulted in very substantial improvements in the performance of the communication link to the Cyber, until it ceased to be a major bottleneck.

How should the U-200 emulator project be assessed ? How is the credit to be apportioned between the programmers and the tools provided in the UNIX software that they employed ? Our feeling is that the project was a difficult one and that the UNIX software was a major contributor to its success. It is noteworthy that while the UNIX system was conceived and implemented purely as an interactive time-sharing system (and as such provides absolutely no support for card reader or card punch peripherals), very little difficulty was encountered in providing software extensions to support the card reader and a 'local batch processing' subsystem.

Meanwhile, the Computing Services Unit (CSU) had gone on to install the RSX–11D-based system in the other larger batch stations around the campus. These had then proceeded to rediscover what we at the Department of Computer Science already knew, and soon evinced an interest in the UNIX alternative. (It may be added that certain elements of gamesmanship became manifest and certain rivalries developed.) Before very long, it became clear to all but the blind that the UNIX-based remote-batch processing system was outperforming the corresponding RSX–11D-based system and that the UNIX system was allowing a significant amount of local processing as well. The CSU management responded by pressuring its programmers to improve the RSX–11D-based system. Although the CSU had a PDP11 that could be devoted almost entirely to software development, whereas our system developers had to share a machine operating with a full work load (including processing several hundred remote-batch jobs per day) and to scrounge after-hour's use of other campus machines, the performance gap was not closed by the CSU for over a year, by which time both U-200 emulators were capable of driving their peripherals at their full rated speeds. But by the time the barndoor was closed, the animals had bolted! By the time the RSX–11D-based system was running smoothly, there were no longer any users of RSX–11D for the CSU to support. Further, the actual support for the other batch stations was being provided not by the CSU staff but by volunteer labour from the Computer Science department. This was clearly an untenable situation for the CSU management. The time for the great software debate had arrived.

The CSU management maintained that the university could not afford to support more than one PDP11 operating system, and that RSX–11D was still the preferred choice. They based their argument mainly on the grounds that whereas RSX–11D had the full support of the software development staff of one of the world's foremost computer manufacturers, the UNIX system was an unknown upstart (it had not by then featured in advertisements in *Datamation*), which had apparently been abandoned by its progenitors without support, and which was quite likely soon to disappear from the face of the earth. Eventually, after considerable reverberations, the decision to adopt UNIX was made and the CSU manager resigned! The epilogue to this story is, of course, that while support for UNIX software by its user community at large has continued to grow rapidly, RSX–11D has now been superseded.

During 1975, the Fifth Edition UNIX system running on the Electrical Engineering PDP11/40 was able to service three or four interactive terminals satisfactorily while the remote-batch system was operating. By the end of 1975, the user community had grown to about thirty, mainly staff members and post-graduate students, but also including a few enterprising undergraduate students. A large number of undergraduates had sampled the system via local batch processing of assembly language programming exercises.

Early in 1976, the batch station was greatly enhanced by the addition of 80 Kbytes of core memory (bringing the total to 208 Kbytes) and by the installation of the Sixth Edition of UNIX software (received at the end of 1975). Further, Ian Johnstone took the opportunity to rewrite completely the batch processing subsystem. (This included the addition of a 'shared data segments' feature into the UNIX operating system.) By the end of 1976, the password file boasted almost two hundred registered users, which were more than enough in view of the limited amount of disc storage available. (The user community actually survived quite well with only three RK05 disc drives—2·4 megabytes each—and no tape drive.) Senior undergraduates in Computer Science were able to sample the facilities offered, even though the bulk of their computing still had to be performed using the Cyber. With the additional memory, the Sixth Edition UNIX software, and improvements to the software disc drivers, the capacity of the 11/40 had increased substantially during 1976. When, one day late in the year, the system suddenly hung because the swap area on disc had become exhausted, it was discovered that no less than 14 users had been logged in, and had been actively and successfully using the system until then.

By mid-1976, the Cyber had become supersaturated and students using that system were reporting waiting as long as three-quarters of an hour for a single program compilation initiated from an 'interactive' terminal. The need for further extension to the University's computing facilities was clearly indicated, and this time there was general agreement—especially from the CSU—that separate facilities should be provided for the Department of Computer Science, which was generating approximately 20 per cent of the Cyber load. During the second half of 1977, the University was able to undertake another major expansion of its computing facilities. Purchases were made of a second Cyber (171) for the CSU and a PDP11/70 for the School of Electrical Engineering.

The arrival of the PDP11/70 prepared the way for the workhorse PDP11/40 in the School of Electrical Engineering to be put out to pasture early in 1978. The latter has become a research and development machine for the Department of Computer Science. Its place has been assumed by the PDP11/70 which has 640 Kbytes of core memory, a tape drive, two Ampex 9100 disc drives (84 megabytes each) and 48 interactive terminal ports (DZ-11). The remote-batch load absorbs about 5 per cent of the 11/70 and its residual capacity is used to serve nearly 50 interactive terminals. (Most of these are CRTs running at 2400 baud, that have been built in-house to our own design; each incorporates a high-quality CRT display and keyboard, and supports the full ASCII 96 character set, and a graphics mode.) The 11/70 normally runs continuously and unattended for over 16 hours per day, and is used mostly for undergraduate teaching with 600 registered users. The principal language processor is a Pascal compiler/interpreter written at U.C. Berkeley.[3]

As noted above, several other sites on campus that use PDP11 computers have also switched to UNIX, for purposes such as: teaching the elements of programming to all students in first year mathematics; teaching BASIC programming to students in the Faculty of Commerce; on-line gaming for students in Industrial Engineering; data reduction from hydraulic models in the Water Research Laboratory; interactive graphics and design in the Faculty of Architecture; cross-assembly and down-line loading of microprocessors in the Digital Systems Laboratory in Electrical Engineering. The Department of Power Engineering is using an 11/40 running under the UNIX system to control an 11/10 that monitors the behaviour of an analogue simulator for electrical power networks. The recently formed Australian Graduate School of Management uses a PDP11/70 to provide on-line teaching demonstrations, to analyse a number of large financial data bases, to execute simulation models and to prepare research papers for publication. Features of the UNIX system which have appealed to the general user community have included the user-oriented documentation, an effective text editor, easy-to-learn shell commands, full-duplex terminal operation and relative freedom to create new files and directories as required. Several individuals with jobs requiring lots of CPU cycles but not too much core memory have discovered that progress can be significantly faster by running in the background of their neighbourhood PDP11 than by taking their turn at the central Cyber computer. Whichever way one measures it, it is clear that a significant percentage of the University's computing is being performed using the UNIX system.

LOCAL DEVELOPMENTS

UNIX software is delivered in both source and object code forms. Most of the source code is written in a higher-level language called C[4] and is relatively easy to understand and to change. Without the threat of frequent new releases of the system to enforce conformity, we have been free to modify and adapt the system to suit our own purposes. Several members of the Department of Computer Science have developed a deep knowledge of the system, so that there has been little difficulty in understanding and correcting the occasional program errors we have encountered. On more than one occasion, we have found it has been quicker to correct a newly discovered program error than to document its existence. We feel we are in a relatively advantageous position compared with users of other brands of software.

Under the UNIX system, a user may control the access of other users to his files and directories. Provided the access permissions are properly set, each user can protect his files quite adequately from all other users except those who have super-user status. We have found the UNIX approach to file security to be sound in principle. It can also be sound in practice provided proper administrative attention is paid to protecting the super-user password, to setting the access permissions properly on all sensitive files, and to checking regularly for evidence of security breakdowns. The standard UNIX system is distributed with a skeleton file system in which access permissions have *not* been set with security as a principal consideration. As a result there are a number of loopholes by way of which an unscrupulous user can achieve 'super-user' status. For this reason, in some quarters, the UNIX system has earned a reputation for being insecure. We consider this reputation to be quite undeserved, though we do admit that the effort to set the access permissions correctly is not trivial.

Facilities for sharing files among a restricted group of users (but not the whole user community) are not well developed. The standard system contains an embryonic implementation of the concept of user groups that we have discarded in the change from 8-bit to 16-bit user identification codes. The simple group concept is not adequate for our purposes because users typically belong to several different groups (i.e. academic subjects).

A long list of the changes that have been made locally is most probably out of place here. Changes have been made to improve the efficiency of the system: the I/O buffering has been extensively modified; changes have been made to various aspects of the swapping algorithms; and frequently used table searching algorithms have been modified. Changes have been made to the security arrangements and error recovery. Limits have been placed on the amounts of resources that can be claimed by a single individual (number of files, amount of file space, number of processes, amounts of line-printer output, etc.). Changes to the scheduling algorithms to emulate appropriate aspects of the Cambridge 'share' system (Larmouth[5]) are in hand. (It may be noted that these changes are largely orthogonal to the recent developments at Bell Laboratories, where the principal directions have been towards improving portability to other systems, providing networking capabilities, handling very large files and file systems and providing a greatly-enhanced command language interpreter.)

Due to the late delivery of some of the hardware, many of the planned PDP11/70 software enhancements were not ready when our academic year commenced in March 1978. Some major software changes (e.g. to raise the number of registered users beyond the standard limit of 256) had to be delayed and then hurriedly implemented in the week after Easter. Full-scale operation with 500 impatient users commenced immediately afterwards. We feel it is a tribute to the robustness of the UNIX system that we could do this and not suffer the direst consequences. By the standards of most installations, we have been getting away with 'murder' in the casual way we have managed (or failed to manage) software development. As already noted, late delivery of some hardware for the 11/70 made a mockery of our original development schedule for software changes to allow 16-bit user identification numbers and resource control. During the first few weeks of full operation of the 11/70, we experienced quite frequent system crashes, about half of which were due to hardware or power supply problems. The remaining problems were mostly associated with some local patches introduced in the search for improved system performance . . . and we did find one esoteric bug dating back to the original UNIX software distribution. Because the 11/70 work day is invariably 16 hours or more, there is little time available for the more desirable forms of system testing. More than once, a tentative version of the system was tested in full production mode. Most people would consider this a recipe for disaster. Fortunately our students, while occasionally vocal, are generally long-suffering. The average time to recover from a crash was about 15 minutes, most of which was spent checking and, occasionally, patching the file system. Even after the worst crash, only a few files were lost. In this respect the UNIX file system compares very favourably with that of KRONOS on the Cyber, where more than once after a system crash it has been necessary to spend 3 hours restoring the entire file system to its state of 2 or 3 days earlier. With our modified version of UNIX stabilized, the reliability of our system has again become very much a function of hardware reliability.

SUPPORT STAFF

The question of support staff for the PDP11/70 is one dark cloud presently on the horizon; at a time when the University is under heavy pressure to contain its recurrent expenses, we have not been able to obtain additional staff, even though at least one more full-time member of staff is clearly desirable, if not absolutely essential. The current situation is that the entire staff officially concerned with the management, operation and software support for the two machines running 16 hours per day in the School of Electrical Engineering, consists of a *single* person. For the time being, the shortfall is being overcome through the voluntary assistance of students and academic staff, but this is clearly no more than a short-term solution. While a staff of one is not satisfactory, it is likely that a staff of two would be satisfactory in the longer term once the initial development period is over, because the system normally runs without supervision or operator interference, often for very extended periods. Obviously, these permanent staff need to be more than usually skilled in both hardware and software, but we anticipate, should the need arise, little difficulty in recruiting from the ranks of our own graduates. By comparison, the CSU has a staff of over 35, which by our standards is extremely luxurious. (If the Department of Computer Science represents 20 per cent of the University's computing activity, then on a comparative basis we should be entitled to a staff of seven or more!)

Our staffing situation can hardly be unique. Although we now operate a machine that is more powerful than the IBM 360/50 that once occupied the same site, we cannot expect to have as many staff now as once served the latter. With UNIX we can survive with less, and because UNIX can be mastered, almost casually, by a significant number of our students, we should always be able to supplement our minimal permanent staff with voluntary, unpaid labour when required. As computer managers generally become more aware of the desirability of choosing computer systems on the basis of the support staff needed, the popularity of systems such as UNIX can only grow even greater.

Obviously our staffing levels have not provided much scope for more than a minimal amount of system documentation. However, again we have been rescued by UNIX, whose document formatting tools have allowed us to maintain the necessary minimum of documentation without undue effort and to reissue current versions of the UNIX software manuals once or twice per year.

COMPUTER SCIENCE TEACHING

Needless to say, the UNIX system now plays an important role in the Computer Science program at the University of New South Wales, and it is the first time-sharing system that the students learn to use. By exposing the students *ab initio* to software of a high standard (they also learn Pascal as their first programming language), we expect that later in their careers they will not be complacent about many of the obsolescent software tools that are still in use. If, one day, they are required to use FORTRAN or certain of the well-known operating systems, then at least they should be able to decide for themselves whether they are working at 'the leading edge of technology' or not. Our undergraduate students not only use the UNIX system to complete their ordinary programming exercises, but it is also discussed as part of the subject on Operating Systems.[6] Other subjects teach details of PDP11 assembler and hardware, so that by the time of graduation, most of our students have a good knowledge, at several different levels, of one complete system of hardware and software.

CONCLUSION

Our enthusiasm for the UNIX system has been somewhat infectious. Not only has it migrated around our campus, but it has been transmitted to other universities in Australia as well.[7] In the process, we have made contact with many people whom we would not normally encounter. Because there is no other way, users of UNIX software need to band together for mutual support. Approximately half the universities in Australia together with two major federal government organizations are now UNIX software licensees, and twice-yearly meetings of our local user group in Australia have created a channel for informal communication that did not exist previously.

It is likely to be many years before the impact of the UNIX system on the computing milieu can be properly assessed. Like so many good ideas its importance is only slowly filtering into the consciousness of the world at large. For the University of New South Wales the benefits are already real and positive. The Department of Computer Science now has available to it modern software and hardware facilities commensurate with its needs, and obtained at a reasonable cost.

ACKNOWLEDGEMENTS

Our local computing scene has been enlivened by many cheerful spirits but the following deserve special mention here for their contributions to the developments described above: Ian Hayes, Peter Ivanov, Greg James, Ian Johnstone, Piers Lauder, Chris Maltby, Greg Rose and John Wainwright. The assistance of Ted Dolotta, Dick Haight, David Hunt and Berkley Tague in reviewing drafts of this paper is gratefully acknowledged.

REFERENCES

1. D. M. Ritchie and K. Thompson, 'The UNIX time-sharing system', *Comm. ACM*, **17**, No. 7, 365–375 (1974).
2. *The Bell System Technical Journal*, **57**, No. 6, Part 2 (July–August 1978).
3. W. N. Joy, S. L. Graham and C. B. Haley, *UNIX Pascal User's Manual*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1977.
4. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J., 1978.
5. J. Larmouth, 'Scheduling for a share of the machine', *Software—Practice and Experience*, **5**, 29–49 (1974).
6. J. Lions, 'An operating system case study', *Operating Systems Review*, **12**, No. 3, 46–53 (1978).
7. R. Miller, 'UNIX—a portable operating system', *Operating Sytems Review*, **12**, No. 3, 32–37 (1978).