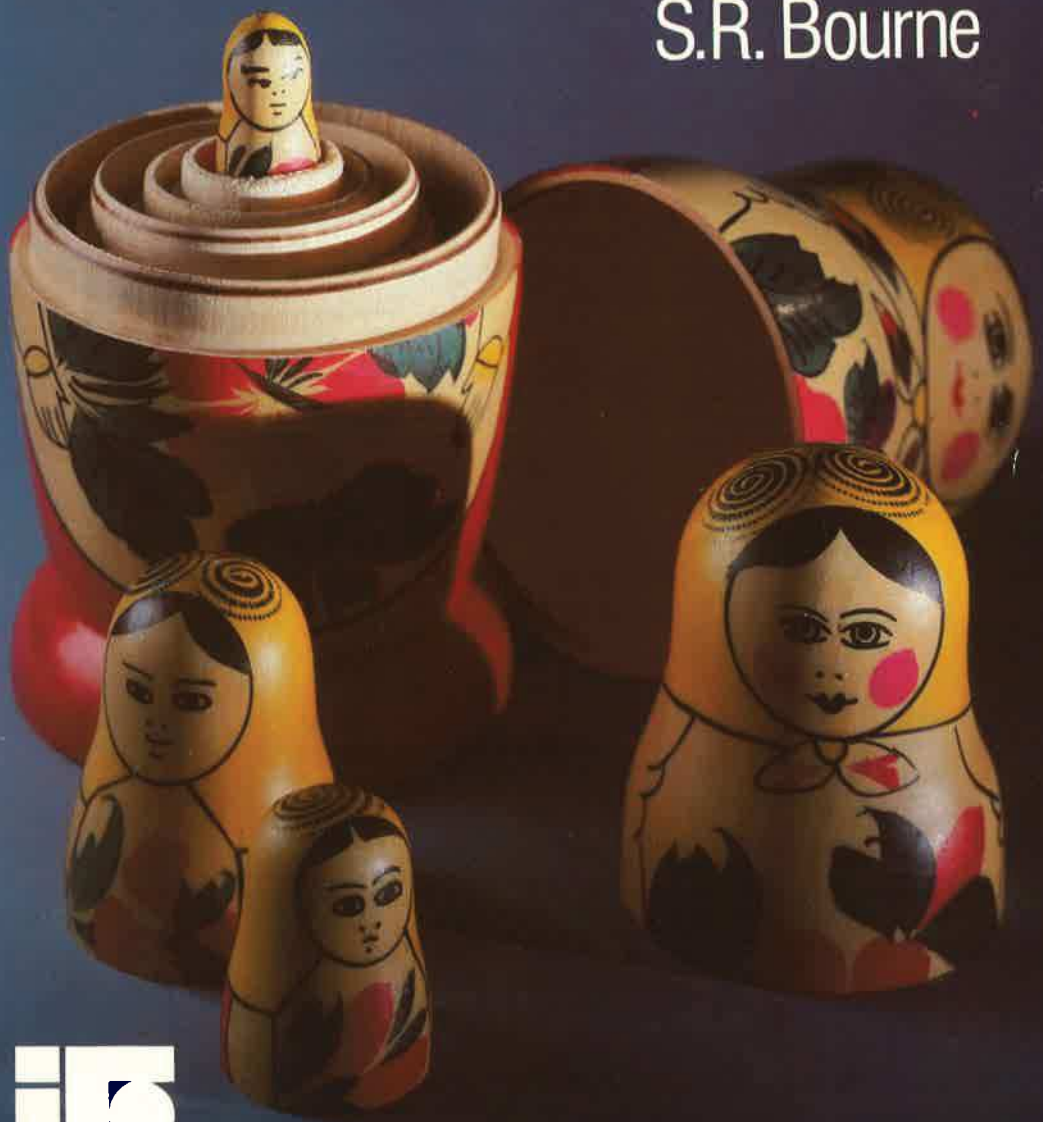
The UNIX System

S.R. Bourne



INTERNATIONAL COMPUTER SCIENCE SERIES

Chapter 1
Introduction

UNIX describes a family of computer operating systems developed at Bell Laboratories. The UNIX system includes both the operating system and its associated commands. The operating system manages the resources of the computing environment by providing a hierarchical file system, process management and other housekeeping functions. The commands provided include basic file and data management, editors, assemblers, compilers and text formatters. A powerful command interpreter is available and this allows individual users or projects to tailor the environment to suit their own style by defining their own commands.

The background leading up to the first UNIX system is worth exploring. During the sixties the major issues being addressed by the computing science community included programming language and operating system design. In the former area such languages as PL/I, APL, SIMULA 67, ALGOL 68 and COBOL were designed and were the subject of debate, often fierce, over their relative merits. In the United Kingdom the Combined Programming Language project (CPL) was undertaken jointly by London and Cambridge Universities but failed to produce any direct results. However, it did form the basis for BCPL (Basic CPL), an ingredient of the story.

The operating systems of this period were designed for medium and large scale computers as a means of sharing the resources among users in a cost effective way. Time-sharing and interactive (as opposed to batch) use was introduced. Such questions as paging strategies, protection, activity scheduling and file system design were explored. Systems like CTSS (Crisman, 1965), Multics (Feiertag, 1969) and, in Europe, the Cambridge Multiple Access System (Hartley, 1968) were being designed and provided many of the key ideas found in the UNIX system. For example, file systems and device independent input-output, processes and command languages were all available in one form or another in these systems.

1.1 History

The story begins with Ken Thompson in 1968. Thompson had recently returned from Berkeley where Butler Lampson was working on the SDS930 operating system (Deutsch and Lampson, 1965). Dennis Ritchie joined Bell Laboratories in 1967 from Harvard where his interest was applied mathematics.

Thompson shared space with a talented group many of whom had recently abandoned Multics; a joint project between Bell Laboratories, General

Electric, and The Massachussets Institute of Technology. Following the withdrawal of Bell Laboratories from Multics and the removal of the GE 645 system in March 1969, the computer science research group began looking for a replacement computing environment. Proposals for new equipment were submitted and rejected as too expensive. Also, operating system development was not a popular research direction after the Multics debacle.

Thompson's own interests were to build a file system rather than an operating system. During discussions between Rudd Canaday, Thompson and Ritchie the design was sketched out and Thompson wrote simulations of early versions of this file system on the GECOS system.

Another thread of the story is the 'space travel' program written on the GECOS machine by Thompson and Ritchie. This program performed poorly on the time-shared computer and better response was needed. A cast-off PDP 7 with a 340 display was available but the PDP 7 provided only an assembler and loader. One user at a time could use the computer, each user having exclusive use of the machine. This environment was crude and parts of a single user UNIX system were soon forthcoming. The space travel program was rewritten for the PDP 7 and an assembler and rudimentary operating system kernel were written and cross assembled for the PDP 7 on the GECOS system. This early system did not provide time-sharing. Indeed, much like the modern personal computers, the PDP 7 hardware was simple and provided no support for such activities. An assembler and a command interpreter were soon available. This file system provided a name structure that was a directed graph. A single directory was used for all sub-directories and links made through this directory.

Cross assembling meant using two computer systems and carrying paper tapes of programs from one to the other each time a change was made. The system was soon bootstrapped onto the PDP 7. The process creation primitive, fork, and process images were added to the system during this rewrite. Essential utilities, such as file copy, edit, remove, and print were soon available. This system supported two people working at the same time and the term UNIX was coined by Brian Kernighan in 1970.

The computing research group still had no computer of its own. Following a series of unsuccessful attempts a proposal was made by Joe Ossanna to purchase a PDP 11/20 for a text preparation project. In late 1970 the PDP 11 arrived and work started to transfer the UNIX system to this more powerful machine.

The text processing project was successful and the Patent department became the first user of UNIX, sharing the facility with the research group. This First Edition system was documented in a manual authored by Thompson and Ritchie dated November 1971. All the important ideas found in modern UNIX systems except pipes, but including the file system, process management, system interface, and major command utilities, were provided with this edition.

The Second Edition appeared in June 1972 incorporating pipes at Doug McIlroy's urging. The system and utilities were still written in assembler. Thompson had also been working on the language B and the assembler for

this system was written in B. B was a direct descendant of BCPL, but programs were compiled in one pass to produce interpretive code.

Both B and BCPL were typeless languages, providing a single data object called the machine word making access to the PDP 11 byte handling instructions difficult. Types were therefore added to B to produce NB and an attempt to rewrite the system in NB was unsuccessful. Ritchie started work on a code generator for NB to make execution of programs faster. This language was called C although there were no structures or global variables. The language was sufficiently attractive, however, that new utilities were being written directly in C.

The year 1973 saw major progress. The system was still written in assembler but following the addition of structures to C the UNIX system was successfully rewritten in C. Thompson wrote the process management and Ritchie the input-output system.

The Sixth Edition UNIX system that became the first widely available version was issued in May 1975, and was distributed for a nominal fee.

Work continued to improve the system. A new file system allowing for larger files was written and the shell was modernized to provide better support for the many programs written in this language. The last major project undertaken by Thompson and Ritchie was to rewrite the system so that it could be transported from one computer to another. The pilot project used an Interdata 8/32, a 32-bit computer similar to the IBM 370 series, that was sufficiently different from the PDP 11 to unearth most machine dependencies. This project also generated some additions to the C language, including unions, casts, and type definitions. This work resulted in the production of the Seventh Edition UNIX system released for general use in 1979. Although the Sixth edition is still in use in some installations it has been generally superseded by the Seventh Edition system.

The UNIX system is now regarded as a standard operating system and has been implemented on many different computers ranging from micros to mainframes. The Seventh Edition system was made available for the PDP 11 16-bit computers. The first VAX 11/780 system, UNIX 32V, was bootstrapped by John Reiser and Tom London, also at Bell Laboratories. This system was further developed, and is now distributed by the University of California at Berkeley. Bell Laboratories has also continued to develop the UNIX system; UNIX System V is the version currently available for license from Western Electric Co.

Some differences exist between these versions both in the operating system and in the commands although these should cause the reader little difficulty. This text is applicable to each of these systems and features found in only one system have been avoided. The programs in this book have been compiled and run on UNIX System V from Bell Laboratories, and on the University of California, Berkeley release 4.1.

Many commands were initially written for the PDP 11 where address space was limited to 64K bytes. This constraint had a generally beneficial effect on the software. Systems are designed as a set of loosely coupled commands. Lack of address space did prevent such languages as LISP from ef-

fective implementation until the arrival of 32 bit machines.

The UNIX system is well engineered and has set a standard of simplicity for time-shared operating systems. It was one of the first operating systems to be widely available on a mini-computer, namely the PDP 11. This combination was affordable by university departments and a generation of computer scientists has been educated on UNIX systems.

The initial interface has aged well and is essentially unchanged since its original design. This stability provided the basis for the development of the user level commands. The UNIX documentation has a conciseness that is appealing although some consider it to be too brief.

The UNIX system is very successful. At the time of writing there are over 3000 UNIX systems in active use throughout the world. These can be found in universities, government laboratories, commercial organizations and many other areas of industry. At Bell Laboratories it is used by staff members both for interactive program development and as a communications and word processing system. The system is portable, easily grasped by both users and maintainers, and provides facilities not available in other, sometimes larger, systems.

1.2 The programming environment

The UNIX system is simple and elegant and provides an attractive programming environment. The facilities available include the following:

- a C compiler and debugger;
- a variety of other language processors, including APL, Basic, Fortran 77, Pascal, and Snobol;
- the text editors ed, vi, and emacs;
- text processing facilities and document preparation aids including mathematical typesetting tbl, eqn, troff, and nroff;
- compiler construction aids yacc, and lex;
- communication among users mail, and write;
- graphics and plotting; and
- applications such as circuit design packages.

These tools are made available to users via a command language that provides the interface between users and the UNIX operating system. This program is called the *shell* and programs written in this language are sometimes referred to as shell scripts.

The shell provides notation for directing input and output from commands and control flow mechanisms typical of algorithmic languages.

Techniques for effective program development have emerged in this environment and include the following:

- Arrange each program to perform a single function.
- Avoid cluttering the output of a program unnecessarily. Assume that the output from any program will be the input to another.
- Use or modify an existing tool if possible rather than rewrite a new

tool from scratch.

- Get something small working as soon as possible and then modify it incrementally until it is finished. This requires that the framework of the design should be in place before significant quantities of program are written.

1.3 UNIX system concepts

1.3.1 The file system

A file system allows users to store information by name. Protection from hardware failures can be provided and security from unauthorized access is also available. The UNIX file system is simple; there are no control blocks, devices are hidden, and there is a uniform interface for all input-output. Within the file system three types of file are distinguished.

- An ordinary file contains characters of a document or program. Executable programs (binary files) are also stored as ordinary files. No record structure is imposed on files; a file consists of a sequence of characters. A newline character may delimit records as required by applications.
- A directory holds the names of other files or directories. A user may create sub-directories to group files related to a project. Consequently, the file system is a hierarchy. A directory can be read, but not written, as if it were an ordinary file.
- Special files correspond to input or output devices. The same interface as ordinary files is available; however, information is not kept in the file system, it is provided directly by the device. The same access protection is available for special and ordinary files.

1.3.2 Processes

All user work in the UNIX system is carried out by processes. A process is a single sequence of events and consists of some computer memory and files being accessed. A process is created by a copy of the process being made. The two processes are only distinguished by the parent being able to wait for the child to finish. A process may replace itself by another program to be executed. This mechanism is both elegant and effective.

1.3.3 The shell

The shell is a command language that provides a user interface to the UNIX operating system. The shell executes commands that are read either from a terminal or from a file. Files containing commands may be created, allowing users to build their own commands. These newly defined commands have the same status as 'system' commands. In this way a new environment can be established reflecting the requirements or style of an individual or a group.

Pipes allow processes to be linked together so that the output from one process is the input to the next. The shell provides a notation enabling pipes to be used with a minimum of effort.