AT&T Bell Laboratories Technical Journal
Vol. 63, No.8, October 1984
Printed in U.S.A.

The UNIX *System:*

The Evolution of the *UNIX* Time-sharing System

By D. M. RITCHIE*

This paper presents a brief history of the early development of the *UNIX'"* operating system. It concentrates on the evolution of the file system, the process-control mechanism, and the idea of pipelined commands. Some attention is paid to social conditions during the development of the system. This paper is reprinted from *Lecture Notes on Computer Science, No.* 79, *Language Design and Programming Methodology,* Springer-Verlag, 1980.

I. INTRODUCTION

During the past few years, the *UNIX* operating system has come into wide use, so wide that its very name has become a trademark of Bell Laboratories. Its important characteristics have become known to many people. It has suffered much rewriting and tinkering since the first publication describing it in 1974/ but few fundamental changes. However, *UNIX* was born in 1969 not 1974, and the account of its development makes a little-known and perhaps instructive story. This paper presents a technical and social history of the evolution of the system.

II. ORIGINS

For computer science at Bell Laboratories, the period 1968-1969 was somewhat unsettled. The main reason for this was the slow, though clearly inevitable, withdrawal of the Labs from the Multics project. To the Labs computing community as a whole, the problem was the increasing obviousness of the failure of Multics to deliver promptly any sort of usable system, let alone the panacea envisioned earlier. For much of this time, the Murray Hill Computer Center was

* AT&T Bell Laboratories.

extricating themselves not only from an operating system development effort that had failed, but from running the local Computation Center. Thus it may have seemed that buying a machine such as we suggested might lead on the one hand to yet another Multics, or on the other, if we produced something useful, to yet another Comp Center for them to be responsible for.

Besides the financial agitations that took place in 1969, there was technical work also. Thompson, R. H. Canaday, and Ritchie developed, on blackboards and scribbled notes, the basic design of a file system that was later to become the heart of *UNIX*. Most of the design was Thompson's, as was the impulse to think about file systems at all, but I believe I contributed the idea of device files. Thompson's itch for creation of an operating system took several forms during this period; he also wrote (on Multics) a fairly detailed simulation of the performance of the proposed file system design and of paging behavior of programs. In addition, he started work on a new operating system for the GE 645, going as far as writing an assembler for the machine and a rudimentary operating system kernel whose greatest achievement, so far as I remember, was to type a greeting message. The complexity of the machine was such that a mere message was already a fairly notable accomplishment, but when it became clear that the lifetime of the 645 at the Labs was measured in months, the work was dropped.

Also during 1969, Thompson developed the game of 'Space Travel.' First written on Multics, then transliterated into Fortran for GECOS (the operating system for the GE, later Honeywell, 635), it was nothing less than a simulation of the movement of the major bodies of the Solar System, with the player guiding a ship here and there, observing the scenery, and attempting to land on the various planets and moons. The GECOS version was unsatisfactory in two important respects: first, the display of the state of the game was jerky and hard to control because one had to type commands at it, and second, a game cost about $75 for CPU time on the big computer. It did not take long, therefore, for Thompson to find a little-used PDP-7 computer with an excellent display processor; the whole system was used as a Graphical terminal. He and I rewrote Space Travel to run on this machine. The undertaking was more ambitious than it might seem; because we disdained all existing software, we had to write a floating-point arithmetic package, the pointwise specification of the graphic characters for the display, and a debugging subsystem that continuously displayed the contents of typed-in locations in a corner of the screen. All this was written in assembly language for a cross-assembler that ran under GECOS and produced paper tapes to be carried to the PDP-7. Space Travel, though it made a very attractive game, served mainly as an introduction to the clumsy technology of preparing programs for

of two programs? What is the appropriate notation for invoking a program with two parallel output streams?

I mentioned above in the section on 10 redirection that Multics provided a mechanism by which 10 streams could be directed through processing modules on the way to (or from) the device or file serving as source or sink. Thus it might seem that stream-splicing in Multics was the direct precursor of *UNIX* pipes, as Multics 10 redirection certainly was for its *UNIX* version. In fact I do not think this is true, or is true only in a weak sense. Not only were coroutines well-known already, but their embodiment as Multics spliceable 10 modules required that the modules be specially coded in such a way that they could be used for no other purpose. The genius of the *UNIX* pipeline is precisely that it is constructed from the very same commands used constantly in simplex fashion. The mental leap needed to see this possibility and to invent the notation is large indeed.

IX. HIGH-LEVEL LANGUAGES

Every program for the original PDP-7 *UNIX* was written in assembly language, and bare assembly language it was-for example, there were no macros. Morever, there was no loader or link-editor, so every program had to be complete in itself. The first interesting language to appear was a version of McClure's TMG[11] that was implemented by McIlroy. Soon after TMG became available, Thompson decided that we could not pretend to offer a real computing service without Fortran, so he sat down to write a Fortran in TMG. As I recall, the intent to handle Fortran lasted about a week. What he produced instead was a definition of and a compiler for the new language B.[12] B was much influenced by the BCPL language;" other influences were Thompson's taste for spartan syntax, and the very small space into which the compiler had to fit. The compiler produced simple interpretive code; although it and the programs it produced were rather slow, it made life much more pleasant. Once interfaces to the regular system calls were made available, we began once again to enjoy the benefits of using a reasonable language to write what are usually called 'systems programs': compilers, assemblers, and the like. (Although some might consider the PL/I we used under Multics unreasonable, it was much better than assembly language.) Among other programs, the PDP-7 B cross-compiler for the PDP-ll was written in B, and in the course of time, the B compiler for the PDP-7 itself was transliterated from TMG into B.

When the PDP-ll arrived, B was moved to it almost immediately. In fact, a version of the multi-precision 'desk calculator' program *de* was one of the earliest programs to run on the PDP-ll, well before