Unix Overview

Dr. Alun Moon

Computing Engineering and Information Sciences

6th December 2011

Unix was designed to be:

- portable
- multi-tasking
- multi-user
- time-sharing configuration.

Unix systems are characterized by various concepts:

- the use of plain text for storing data
- a hierarchical file system
- treating devices and certain types of inter-process communication (IPC) as files
- and the use of a large number of software tools,
  - small programs that can be strung together through a command line interpreter using pipes
  - as opposed to using a single monolithic program that includes all of the same functionality.

These concepts are collectively known as the Unix philosophy. Kernighan and Pike summarize this in The Unix Programming Environment as

> *"the idea that the power of a system comes more from the relationships among programs than from the programs themselves."*

Standards

POSIX IEEE based POSIX around the common structure of the major competing variants of the Unix system, publishing the first POSIX standard in 1988

SUS the Common Open Software Environment (COSE) initiative, which eventually became the Single UNIX Specification administered by The Open Group

— Starting in 1998 the Open Group and IEEE started the Austin Group, to provide a common definition of POSIX and the Single UNIX Specification.

The Filesystem Hierarchy Standard was created to provide a reference directory layout for Unix-like operating systems, particularly Linux.

- The Unix system is composed of several components that are normally packaged together.
- By including in addition to the kernel of an operating system the development environment, libraries, documents, and the portable, modifiable source-code for all of these components, Unix was a self-contained software system.
- This was one of the key reasons it emerged as an important teaching and learning tool and has had such a broad influence.

The inclusion of these components did not make the system large the original V7 UNIX distribution, consisting of copies of all of the compiled binaries plus all of the source code and documentation occupied less than 10MB, and arrived on a single 9-track magnetic tape. The printed documentation, typeset from the on-line sources, was contained in two volumes.

Ritchie observes:

> *What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form. We knew from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type programs into a terminal instead of a keypunch, but to encourage close communication.*

The theme of computers being viewed not merely as logic devices but as the nuclei of communities was in the air; 1969 was also the year the ARPANET (the direct ancestor of today's Internet) was invented. The theme of *fellowship* would resonate all through Unix's subsequent history.

- The Unix tradition of lightweight development and informal methods also began at its beginning.
- Where Multics had been a large project with thousands of pages of technical specifications written before the hardware arrived,
- the first running Unix code was brainstormed by three people and implemented by Ken Thompson in two days on an obsolete machine that had been designed to be a graphics terminal for a *real* computer.

Peer pressure and simple pride in workmanship caused gobs of code to be rewritten or discarded as better or more basic ideas emerged. Professional rivalry and protection of turf were practically unknown: so many good things were happening that nobody needed to be proprietary about innovations

–Doug McIlroy

PDP-7 PDP-11

Unix has a couple of unifying ideas or metaphors that shape its APIs and the development style that proceeds from them.

The most important of these are probably the everything is a file model and the pipe metaphor built on top of it.

In general, development style under any given operating system is strongly conditioned by the unifying ideas baked into the system by its designers they percolate upwards into applications programming from the models provided by system tools and APIs.

A fundamental simplifying assumption of Unix was its focus on ASCII text for nearly all file formats.

- There were no binary editors in the original version of Unix
  - the entire system was configured using textual shell command scripts.
- The common denominator in the I/O system was the byte
- The focus on text for representing nearly everything made Unix pipes especially useful,
  - encouraged the development of simple, general tools that could be easily combined to perform more complicated ad hoc tasks.
- The focus on text and bytes made the system far more scalable and portable than other systems.
- Over time, text-based applications have also proven popular in application areas,
  - such as printing languages (PostScript, PDF, ODF),
  - the application layer of the Internet protocols, e.g., FTP, SMTP, HTTP, SOAP and SIP.

Cooperating Processes

> *The pipe was technically trivial, but profound in its effect. However, it would not have been trivial without the fundamental unifying notion of the process as an autonomous unit of computation, with process control being programmable. As in Multics, a shell was just another process; process control did not come from God inscribed in JCL.*
>
> *– Doug McIlroy*

A subtle but important property of pipes and the other classic Unix IPC methods is that they require communication between programs to be held down to a level of simplicity that encourages separation of function.

Internal Boundaries

- Unix has wired into it an assumption that the programmer knows best.
  - It doesn't stop you or request confirmation when you do dangerous things with your own data,
  - like issuing rm -rf *.

Internal Boundaries

- System programs often have their own pseudo-user accounts to confer access to special system files without requiring unlimited (or superuser) access.

Internal Boundaries

Unix has at least three levels of internal boundaries that guard against malicious users or buggy programs.

1. memory management
   - Unix uses its hardware's memory management unit (MMU) to ensure that separate processes are prevented from intruding on the others' memory-address spaces.
2. true privilege groups for multiple users
   - an ordinary (nonroot) user's processes cannot alter or read another user's files without permission.
3. confinement of security-critical functions to the smallest possible pieces of trusted code.
   - Under Unix, even the shell (the system command interpreter) is not a privileged program.

The strength of an operating system's internal boundaries is not merely an abstract issue of design: It has important practical consequences for the security of the system.

What you got/get

The names and filesystem locations of the Unix components have changed substantially across the history of the system.
Nonetheless, the V7 implementation is considered by many to have the canonical early structure:

Kernel

source code in /usr/sys, composed of several sub-components:

- **conf** configuration and machine-dependent parts, including boot code
- **dev** device drivers for control of hardware (and some pseudo-hardware)
- **sys** operating system kernel, handling memory management, process scheduling, system calls, etc.
- **h** header files, defining key structures within the system and important system-specific invariables

Development Environment I

Early versions of Unix contained a development environment sufficient to recreate the entire system from source code:

- **cc** C language compiler (first appeared in V3 Unix)
- **as** machine-language assembler for the machine
- **ld** linker, for combining object files
- **lib** object-code libraries (installed in /lib or /usr/lib). libc, the system library with C run-time support, was the primary library, but there have always been additional libraries for such things as mathematical functions (libm) or database access. V7 Unix introduced the first version of the modern "Standard I/O" library stdio as part of the system library. Later implementations increased the number of libraries significantly.
- **make** build manager (introduced in PWB/UNIX), for effectively automating the build process

Development Environment II

include header files for software development, defining standard interfaces and system invariants

Other languages V7 Unix contained a Fortran-77 compiler, a programmable arbitrary-precision calculator (bc, dc), and the awk scripting language, and later versions and implementations contain many other language compilers and toolsets. Early BSD releases included Pascal tools, and many modern Unix systems also include the GNU Compiler Collection as well as or instead of a proprietary compiler system.

Other tools including an object-code archive manager (ar), symbol-table lister (nm), compiler-development tools (e.g. lex & yacc), and debugging tools.

Commands I

Unix makes little distinction between commands (user-level programs) for system operation and maintenance (e.g. cron), commands of general utility (e.g. grep), and more general-purpose applications such as the text formatting and typesetting package.
Nonetheless, some major categories are:

sh The "shell" programmable command line interpreter, the primary user interface on Unix before window systems appeared, and even afterward (within a "command window").

Utilities the core tool kit of the Unix command set, including cp, ls, grep, find and many others. Subcategories include:

System utilities administrative tools such as mkfs, fsck, and many others.

User utilities environment management tools such as passwd, kill, and others.

Commands II

Document formatting Unix systems were used from the outset for document preparation and typesetting systems, and included many related programs such as nroff, troff, tbl, eqn, refer, and pic. Some modern Unix systems also include packages such as TeX and Ghostscript.

Graphics The plot subsystem provided facilities for producing simple vector plots in a device-independent format, with device-specific interpreters to display such files. Modern Unix systems also generally include X11 as a standard windowing system and GUI, and many support OpenGL.

Communications Early Unix systems contained no inter-system communication, but did include the inter-user communication programs mail and write. V7 introduced the early inter-system communication system UUCP, and systems beginning with BSD release 4.1c included TCP/IP utilities.

Documentation

Unix was the first operating system to include all of its documentation online in machine-readable form. The documentation included:

man manual pages for each command, library component, system call, header file, etc.

doc longer documents detailing major subsystems, such as the C language and troff