Introduction

This paper contains notes on the section of the UNIX for Broadcast Engineers class that pertains to UNIX history.

Background: What's an Operating System?

An operating system provides a software layer to broker interactions among applications and computer hardware. An operating system saves applications the trouble of having to know how to talk to lots of different hardware by providing a common applications programming interface (API). This also helps applications (and users) by providing a way of controlling the running of applications.

The UNIX operating system is organized into two parts: the kernel and various shells.

Kernel

The kernel manages hardware and applications. The kernel provides an API to do this.

Shells

Shells broker interactions between users, applications, and the kernel, using a command line interface (CLI) a.k.a. command line user interface (CUI). Shells are command interpreters; they execute the commands you type on the command line or supply from the contents of a file.

History

In 1969, engineers (Ken Thompson, Dennis Ritchie, et al) at AT&T Bell Laboratories (a.k.a. Bell Labs) developed the UNIX file system. Bell Labs (Brian Kernighan, Dennis Ritchie, et al) then developed the C programming language (think of it as a portable structured assembly language (what's an assembly language?)) and rewrote UNIX in C (mostly).

AT&T was the only telephone company at that time. There were lots of government regulations/restrictions that applied to AT&T, sort of as the cost of being a government-enabled monopoly. AT&T could not market/sell UNIX. Still, it got more popular (maybe because it was in fact free, kind of like other free software today). AT&T used UNIX internally, and also licensed it to universities very cheaply starting in 1975. Eventually, in 1977, AT&T granted commercial licenses for UNIX.

UNIX Development Paths

Once UNIX got more popular, there became two main flavors of UNIX.

One flavor was developed by AT&T (and later UNIX System Laboratories (USL)), who called its version System III and later System V (Roman numeral 5, not the letter V). Currently, there is SVR4, an acronym of sorts standing for System V Release 4.

The other flavor was developed by the Computer Science Research Group at the University of

California at Berkeley, and was known as the Berkeley Software Distribution (BSD). They not only contributed a lot of code to UNIX, they also integrated code written elsewhere. Their latest release was known as 4.4BSD. (talk about how CSRG licensed UNIX from AT&T). UNIX System Laboratories sued U.C. Berkeley and other users of BSD UNIX in the early 1990's for theft of intellectual property; USL lost their suit in 1994, leaving BSD UNIX essentially unencumbered by legal issues.

Other UNIX-like Operating Systems

UNIX is a trademark of UNIX System Laboratories. Since USL must be paid royalties for the use of the term UNIX, other companies use other names to refer to the UNIX-like operating systems they develop.

There were other computer hardware vendors that developed proprietary versions of UNIX for use on their hardware, including Sun Microsystems (SunOS and Solaris), Hewlett-Packard (HP-UX), and IBM (AIX).

There were also some hobby/research versions of UNIX-like operating systems. Andrew Tanenbaum, a professor, developed MINIX in the early 1980's for educational purposes, and published its source code. In the late 1980's, Linus Torvalds developed what he intended to be a free replacement for UNIX, called Linux, and apparently had as one design goal to go beyond the limitations of MINIX.

FreeBSD, OpenBSD, NetBSD, and other derivatives of BSD UNIX came about in the 1990's, once the USL lawsuit was settled.

GNU/Linux convergence

Richard Stallman founded and built the GNU Project starting in 1983, intended to create a free operating system and utilities that would operate like UNIX. By 1991, the GNU Project had created a lot of software, except they had not built the UNIX-like kernel they needed. Linus Torvald's Linux operating system kernel merged with the GNU operating system and utilities; the two were complementary.

UNIX Philosophy

Each Software Tool does One Thing Well

Resources were extremely limited on early computers. There wasn't room for saving everybody's individual favorite set of tools. People built tools that performed one task well, and shared them.

Text as the Common Data Interchange Format

Each software tool accepted text as input, and generated text as output. There would be some exceptions here, but this is generally the case. The operating system provided mechanisms by which the text generated by one application could be fed to a second application. This enabled people to build tools that reused existing tools whenever possible.

This historical reliance on plain text as a data interchange format will help you understand why a number of Internet protocols are the way they are.

