

CptS 360 (System Programming)

Unit 2: UNIX/Linux Overview

Bob Lewis

School of Engineering and Applied Sciences
Washington State University

Spring, 2022

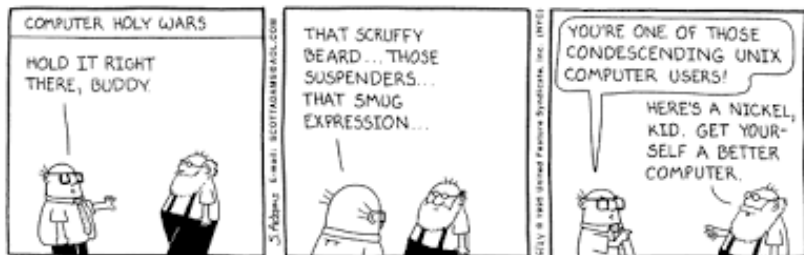
Motivation

- ▶ OSES, languages, and APIs have histories: Learn from them.
- ▶ What has *a/ways* happened when...
 - ▶ a system boots?
 - ▶ you log in?
 - ▶ you log out?
 - ▶ a system shuts down?
- ▶ What general facilities does the OS provide the system programmer?
- ▶ Note in passing: What influenced OS design decisions?

References

- ▶ Stevens & Rago, Ch. 1 & 2
- ▶ <http://www.levenez.com/unix>

One View of Unix(/Linux) Programmers



- ▶ UNIX development started in 1969.
- ▶ It inspired Linux in 1991.
- ▶ It contained many good ideas we still use today (which we'll cover in this lecture) and inspired many more.

Try It Yourself!

From any shell, you can log in to UNIX systems today (as a guest, no charge):

- ▶ A real(?) PDP-11/70 running¹ UNIX v7 (first Bell Labs UNIX wide release, 1979):

```
$ ssh misspiggy@tty.livingcomputers.org
```

- ▶ A simulated MicroVAX 3900 running² BSD 4.3 (classic “BSD”, 1986)

```
$ ssh guildenstern@tty.livingcomputers.org
```

- ▶ A real AT&T 3B2/1000-70 running UNIX SVR3.2.3 (classic “SYS5”, 1987):

```
$ ssh lcm3b2@tty.livingcomputers.org
```

- ▶ A real Sun-3/100 running SunOS 4.1.1 (BSD-based, 1990)

```
$ ssh three@tty.livingcomputers.org
```

If you're familiar with the shell, you'll feel quite comfortable.

¹currently down

²password not provided

One More Famous UNIX Reference

Some people say UNIX is a dinosaur, but in *Jurassic Part* it actually helped people *escape* the dinosaurs:

<https://youtu.be/dFU1AQZB9Ng>

(This is an IRIX system from the former Silicon Graphics.)

Aside: The Living Computer Museum

These logins are courtesy of The Living Computer Museum in Seattle (livingcomputers.org). The museum is closed to live visits now because of COVID, but (most of) their machines are remotely accessible.

You can log in to other non-UNIX machines there via

```
$ ssh menu@tty.livingcomputermuseum.org
```

and there are some photos on their Wiki:

<https://wiki.livingcomputers.org/doku.php>


Logging In and Out

What happens when you log in to a Linux system on a console?
The same thing that happens over an *ssh(1)* session and that happened on a teletype on the first UNIX system.

1. *systemd(1)*³ (formerly, *init(1)*) prompts for your login, passing it to...
2. *login(1)*, which prompts for your password
 - ▶ if unsuccessful, *login(1)* exits and control returns to *systemd(1)*
 - ▶ if successful, *login(1)*...
 - ▶ cd's to your new directory
 - ▶ starts up your shell

It finds the information it needs the *passwd(5)* database.

What happens when you log in on a display?
... and when you log out?

³Remember, "*systemd(1)*" means "the result of '\$ man 1 systemd'" 

The *passwd(5)* Database

The password database (originally in `/etc/passwd`) had these fields in it, one for each user:

- ▶ login name
- ▶ password (encrypted – why?)
- ▶ the user id (UID), a small integer
- ▶ the group id (GID), a small integer
- ▶ comment (usually full name, phone, etc.)
- ▶ home directory
- ▶ login shell

Nowadays, the database is shared over the network and the “password” field is handled differently for security reasons.

UNIX Shells, Past and Present

UNIX users ran other programs using a “shell” or “command line interface” (CLI):

classic shells:

- ▶ *sh(1)*
 - ▶ [Steven] Bourne shell
 - ▶ Bell Labs
- ▶ *cs(1)*
 - ▶ [Bill Joy] C shell
 - ▶ UC Berkeley

more recent shells:

- ▶ *ksh(1)*
 - ▶ [David] Korn shell
 - ▶ AT&T Bell Labs
- ▶ *bash(1)*
 - ▶ “Bourne-again” shell
 - ▶ net-developed
- ▶ *zsh(1)*
 - ▶ “z shell”
 - ▶ lots of features
- ▶ *ash(1)*
 - ▶ [Kenneth] Almquist shell
 - ▶ teensy
 - ▶ used for diagnostics & small systems

UNIX Philosophy: If you don't like an existing tool, build your own!

UNIX Files and Directories

file A named collection of bytes residing in a directory. Under UNIX, it's always byte-, not record-oriented.

directory A collection of files and other directories.

working directory The directory used to interpret relative directory names. (aka “current directory” or “current working directory”)

home directory The working directory to which you log in.

root directory The uppermost directory in the directory tree. It is always named “/”.

UNIX Filenames and Paths

filename a sequence of characters describing a file or directory within a directory.

Filenames may have restrictions on name lengths and permissible characters.

Q: What two characters cannot appear in a filename?
(With one exception.)

path a sequence of one or more filenames joined by "/"s.

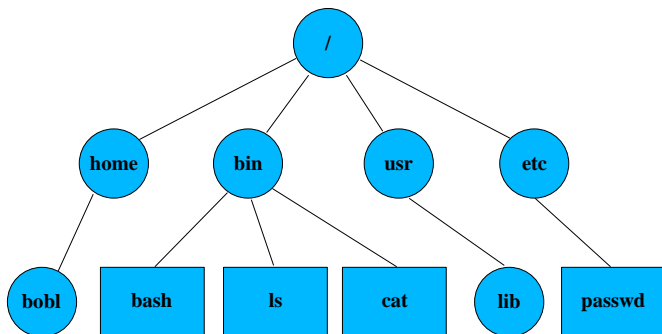
absolute path a path that begins with "/" (which is the name of a directory).

relative path a path that does not begin with "/".

Q: What's "."? Is it absolute or relative?

Q: What's ".."?

Example: A Small Directory Tree



UNIX Philosophy: Filesystems

Everything⁴ has a place in the filesystem.

Originally, these included

- ▶ files
- ▶ directories
- ▶ fixed devices (`/dev`)

Over time, developers added:

- ▶ removeable devices (`/media`, `/cdrom`)
- ▶ FIFOs (named pipes, discussed later)
- ▶ kernel memory (`/dev/mem`)
- ▶ processes (`/proc`)
- ▶ packages (`/snap`)
- ▶ kernel parameters (`/sys`)

⁴With exceptions: message queues, shared memory, semaphores, and network sockets

Standard Input/Output

UNIX supports:

- ▶ standard input ($n = 0$)
(aka *stdin(3)*)
- ▶ standard output ($n = 1$)
(aka *stdout(3)*)
- ▶ standard error ($n = 2$)
(aka *stderr(3)*)

These are all part of the “standard I/O” package, which we’ll study in an upcoming unit.

What do these mean on the shell command line? (n and m are non-negative integers.)

>

>>

<

n >

n >>

n <

|

n > m

n >& m

Programs and Processes

program an executable file

process a running program

pipe a connection between two processes that passes the output of the first to the input of the second

UNIX Innovation: A user-created process can start, control, and stop others.

UNIX Philosophy: Build small programs and make it easy to combine them (e.g. pipes, scripts) as needed.

Oh, Say, Can You C? I

You can't talk about UNIX without mentioning C.

- ▶ A C compiler is a given for all UNIX OSes.
- ▶ Concurrently developed on a *discarded* PDP-7 at Bell Labs.
 - ▶ first version of UNIX was in PDP-7 assembler
 - ▶ 18-bit words
 - ▶ memory from 4K to 64K words
 - ▶ cost \$72K (in 1965, \$630,000 today)
- ▶ Original C was by Brian Kernighan and (mostly) Dennis Ritchie (hence "K & R") at Bell Labs from 1969 to 1973.
- ▶ ANSI (in 1989, hence "C89") and ISO (in 1990, hence "C90") standards are identical. "ANSI C" is common usage.

Oh, Say, Can You C? II

- ▶ ANSI (in 1989, hence “C89”) and ISO (in 1990, hence “C90”) standards are identical. “ANSI C” is common usage.
- ▶ ANSI C New Features:
 - ▶ function prototypes (borrowed from C++)
 - ▶ generic pointers (void *)
 - ▶ international character sets (ISO 8859)
 - ▶ very portable (arguably more so than C++)
- ▶ ISO/IEC 9899:1999 (hence “C99”) added:
 - ▶ inline functions
 - ▶ new types (long long int, complex)
 - ▶ macros with variable numbers of arguments
 - ▶ // comments (borrowed from C++)
 - ▶ looser declaration syntax

and most recently ...

C11

- ▶ alignment control
- ▶ `_Generic()`: selects expressions based on type
- ▶ multi-threading support
- ▶ better Unicode support
- ▶ *gets(3)* goes away (and it's about time!)
- ▶ bounds-checking (deprecated)
- ▶ static assertions that know about types at compile time
- ▶ anonymous structs and unions

There's a C17, but it only clarifies C11. A C2x is expected in 2023. Our (and the Ubuntu) default is C11 with GNU extensions.

So What's C Good For?

- ▶ not just UNIX
- ▶ OS kernels (UNIX, Linux, MacOS, iOS, Android, etc.)
- ▶ device drivers
- ▶ debuggers
- ▶ embedded systems
- ▶ compilers (e.g. Haskell, C++ (originally))
- ▶ interpreters (e.g. Perl, Python)
- ▶ practically anything else, if the design is good

Take a look at

- ▶ <http://www.tiobe.com/tiobe-index>

- ▶ [http:](http://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming)

[//www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming](http://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming)

Other UNIX Entities

- ▶ user/group ID's
 - ▶ positive integers
 - ▶ uniquely identify a user or group
- ▶ signals
 - ▶ notify process that some error has occurred
 - ▶ may have “handlers”
- ▶ time values
 - ▶ time-of-day
 - ▶ CPU usage
 - ▶ elapsed time (time interval)

UNIX vs. UNIX-like OSes

- ▶ Major UNIX implementations:
 - ▶ SVr4
 - ▶ 4.4BSD
 - ▶ FreeBSD (for Intel)
 - ▶ NetBSD (for all platforms)
 - ▶ OpenBSD (secure)
- ▶ Major UNIX-like implementations:
 - ▶ Linux
 - ▶ MacOS X
 - Darwin = FreeBSD running on Mach microkernel (Which is?)
 - ▶ iOS
 - ▶ Android
 - ▶ Solaris

But how many UNIX and UNIX-like OSes do you *think* there are?

UNIX and UNIX-like OSes: A Detailed View

(see chart)