CIS241

System-Level Programming and Utilities

Linux Processes

Erik Fredericks, frederer@gvsu.edu

Fall 2025

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**

What is a process?

- **Execution of a program/command**
  - Or, something running in memory!
- **User process**
  - Process begun by a user on a terminal
- **Daemon process**
  - System process
  - Not associated with a terminal

Viewing processes

- `ps` - list active processes
- `ps -f` - more information
- `ps -u` - what processes am *I* running?
- `top` - interactive
    - If installed, `htop` even more interactive

Processes and Terms

Process ID (PID)

- Unique identifier assigned to a process

Child process

- Process started by another process (parent process)

Parent process

- Process that has started other processes (child processes)

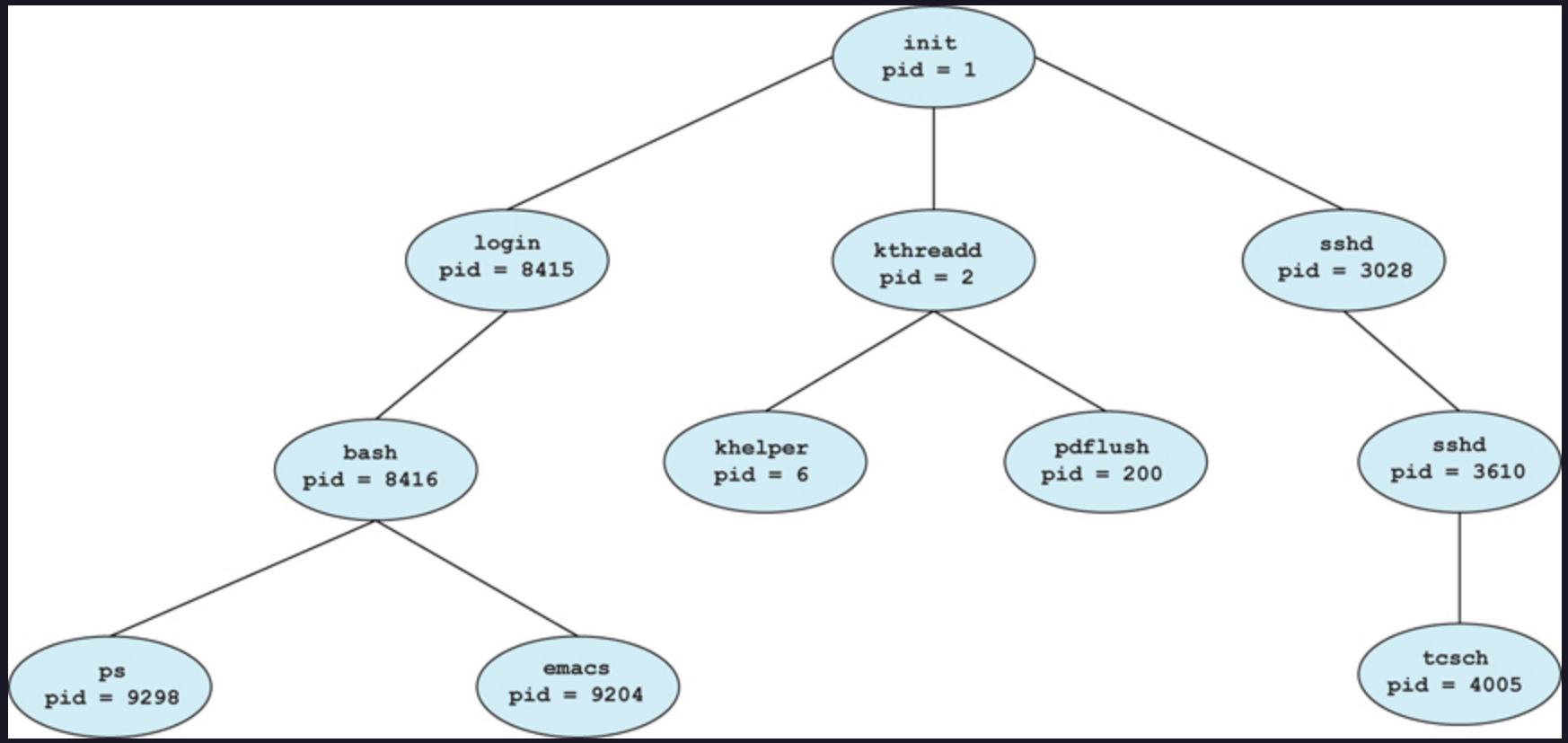
Parent Process ID (PPID)

- PID of the parent process

The init daemon has a PID of 1 and a PPID of 0

0 refers to the kernel itself

Process tree



`pstree`

Processes

`init` daemon

- Starts most other daemons during the system initialization process
- Including those that allow for user logins

The login program starts a bash shell

- bash shell then interprets user commands and starts all user processes
- Each process on the Linux system can be traced back to the init daemon by examining PPIDs

Processes

Process state (`ps aux` or `ps -l`)

- Current processor state of process
- Generally sleeping (`S`) or running (`R`)
- This column is the most valuable to system administrators

Zombie process

- Process finished, but parent has not released child process's PID
- Defunct process
- Process state is `Z`

Foreground and background

Processes typically run in the foreground when you launch them

- Takes over the shell until completed!
- Kind of annoying if you want to do other things...
- **To suspend a running process:** `Ctrl + Z`
- **To resume suspended process in foreground:** `fg`
- **To resume suspended process in background:** `bg`
- `jobs` **to show foreground, background, and suspended processes!**
- **Can also run in background immediately:** `command &`

Stopping a process

In the foreground (running)

- `Ctrl + c`

In the background

- `kill pid` - stop gracefully -- lookup with `ps`
  - What is *gracefully*?
- `kill -9 pid` - force kill - stop immediately!
- `killall processname` - kills ALL processes with `processname` (typically requires `sudo`)

Stopping a process

64 different types of kill signals!

- Why?

Name	Number	Description
SIGHUP	1	Also known as the hang-up signal, it stops a process, then restarts it with the same PID. If you edit the configuration file used by a running daemon, that daemon might be sent a SIGHUP to restart the process; when the daemon starts again, it reads the new configuration file.
SIGINT	2	This signal sends an interrupt signal to a process. Although this signal is one of the weakest kill signals, it works most of the time. When you use the Ctrl+c key combination to kill a currently running process, a SIGINT is actually being sent to the process.
SIGQUIT	3	Also known as a core dump, the quit signal terminates a process by taking the process information in memory and saving it to a file called core on the hard disk in the current working directory. You can use the Ctrl+\ key combination to send a SIGQUIT to a process that is currently running.
SIGTERM	15	The software termination signal is the most common kill signal used by programs to kill other processes. It is the default kill signal used by the `kill` command.
SIGKILL	9	Also known as the absolute kill signal, it forces the Linux kernel to stop executing the process by sending the process's resources to a special device file called /dev/null.

Table 9-2: Common administrative kill signals

Killing processes

Trap: the ability of a process to ignore a kill signal

- The `SIGKILL` signal cannot be trapped by any process
- Use only as last resort because it prevents a process from closing temporary files and resources properly

When a parent process receives a kill signal

- Terminates all child processes before terminating self
- To kill several related processes send signal to parent process
- Kill parent process in order to kill zombie processes