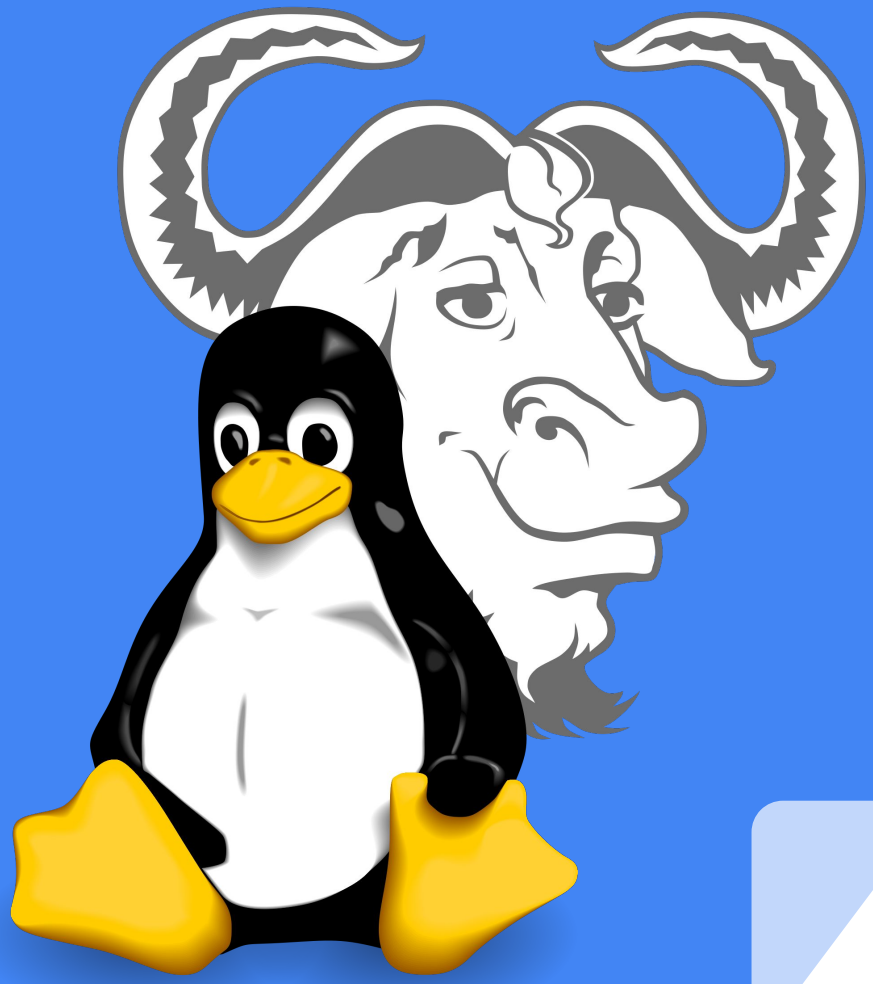
GNU/Linux

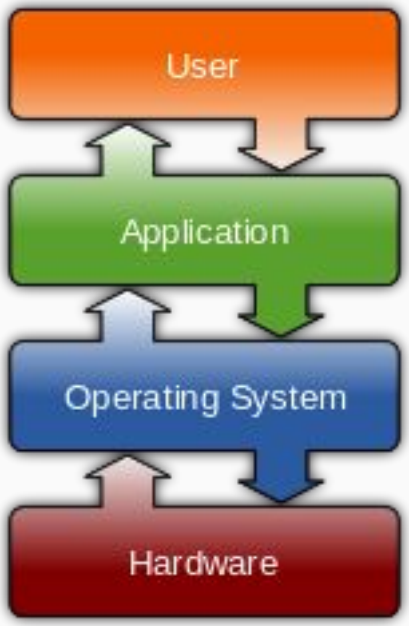
CPP CSS Fall 2016



Agenda

- What is an OS?
- What is Linux?
- The command line
- Daemons & systemd

What is an OS?



- The layer between applications and hardware
- Manages computer hardware and software resources
- Provides an interface for applications to interact with the hardware

OS Components

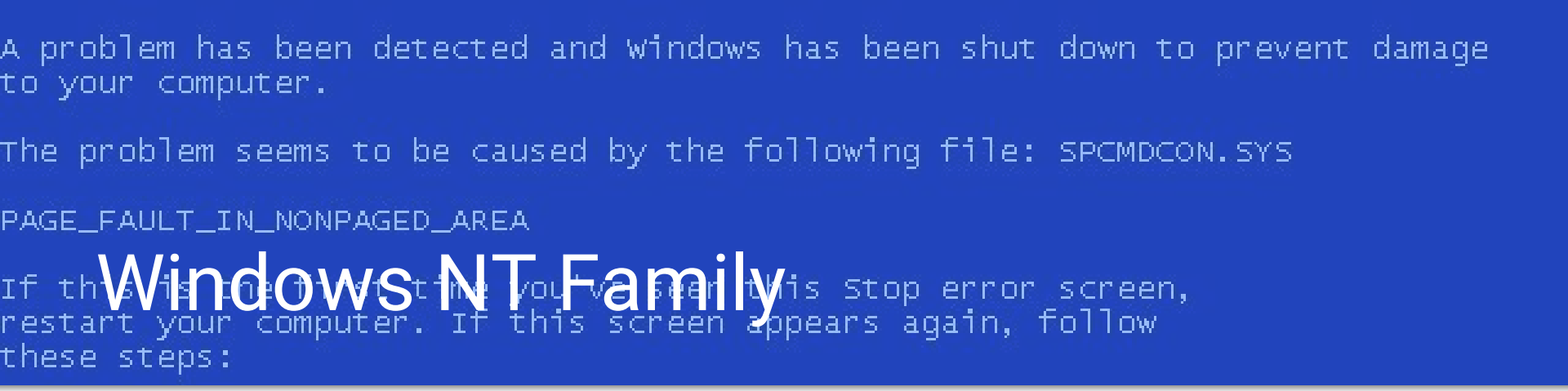
- Kernel
  - Execution modes
    - User space vs kernel space
  - Memory management
    - Protected memory
    - Virtual memory
  - Multitasking / process scheduling
- Networking
- User interface
  - GUI or CLI

Types of OS

- Single- and multi-tasking
- Single- and multi-user
- Distributed
- Embedded
- Realtime

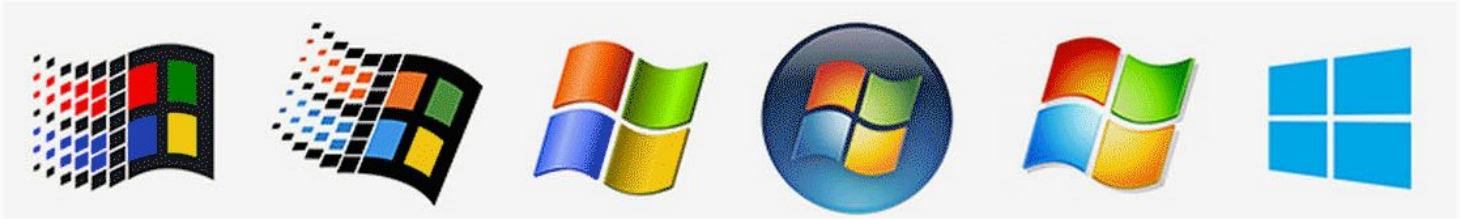
Families of OS

- Windows NT based
- POSIX/Unix-like



Windows NT Family

- Microsoft
  - Windows 2000 through Windows 10
- Closed source
- >80% of the desktop market



POSIX

- Portable Operating System Interface
- Family of standards specified by the IEEE Computer Society
- Uniform application programming interface (and command line shell) for UNIX-Like operating systems
- Generally very easy to port applications from one Unix-like OS to another Unix-like OS
- Unix-like systems are generally open source
- Once you learn one Unix-like system, you will feel at home in most other Unix-like systems

Examples of Unix-Like OS

- Unix (BSD, Solaris, etc...)
- Darwin (OS X and iOS)
- Linux

Motivation

- Wasn’t Unix developed in the 70’s? Why learn Unix/Linux?
- Most open source development is done in Unix-like environments
- Most major software companies are powered by Linux
  - Google, Facebook, Amazon, etc...
- Linux/Unix is everywhere

Linux/UNIX is everywhere!

- Linux
  - Embedded devices
    - Routers, cable boxes, smart TVs, IOT devices, IP cameras, Tesla Model S/X Infotainment, Raspberry Pi, etc...
  - Android and Chrome OS
  - >66% of web servers on the public internet
  - Supercomputers
  - ~2% of desktop users
- UNIX
  - macOS and iOS
  - PS3, PS4, PSP, PS Vita
- The only market dominated by Windows is the desktop

https://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Linux

- Started by Linus Torvalds in 1991
- Named after himself
- Free and open source
- Unix-like

Linux is not an OS

- Linux itself is just a kernel -- only one component of a full OS
- People create software collections based around the linux kernel that comprise a full OS
- These collections are called distributions or distros

Linux Distributions

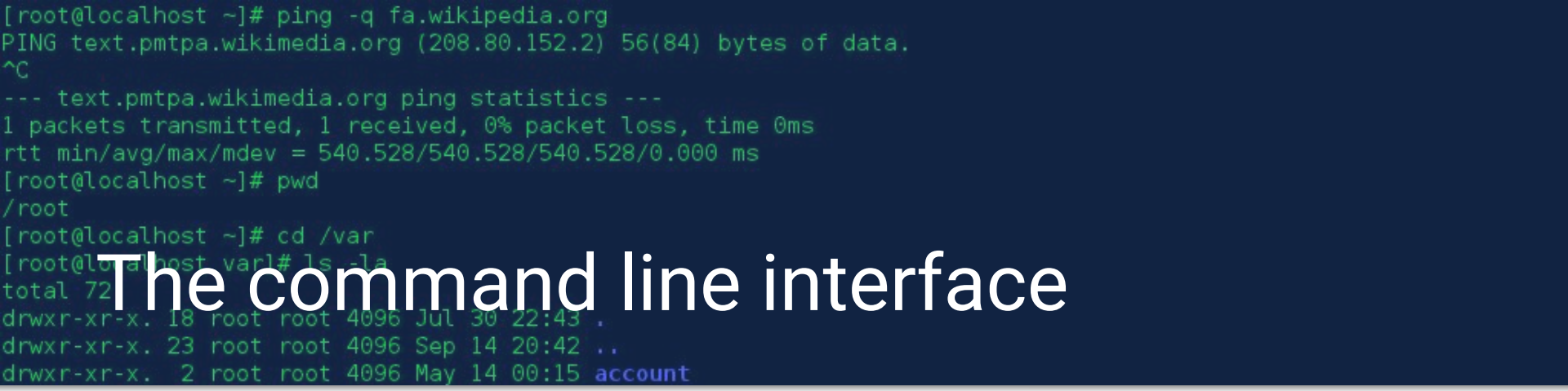
- A typical distro includes: the kernel, GNU tools and libraries, window system, window manager, desktop environment, and additional user software
- Examples:
  - Arch, Debian, Ubuntu, Red Hat, CentOS, etc...


GNU/Linux vs NSA/Windows

- Linux uses forward slashes for directories
  - e.g. /var/log/apache2/access.log
- Filenames are case sensitive
- “Everything is a file”
  - All resources are simple streams of bytes accessed through the filesystem namespace
  - Everything on the system lives under “/” (the root directory)
- Strong focus on stability and security
- Requires fewer resources to run
- All open source. Everything can be tweaked, changed, or replaced

Linux Directory Structure

- Some variations from distro to distro
- / - the root directory
  - /bin - user binaries
  - /sbin - system binaries
  - /lib - system libraries
  - /etc - configurations files
  - /var - variable files
  - /home - user home directories
  - /tmp - temporary files
  - /dev - device files
  - /usr - user programs



The command line interface

- More powerful than a GUI
- Some tasks are much faster through a CLI
- Some tasks simply can not be completed using a GUI
- Many developer tools are command line only
- Difficult to do serious development without the CLI

The command line interface

- Forward slashes for directories
- / is the system's root directory
- ~ is your home directory
  - ~ is evaluated to /home/<your username>
- .. is shorthand for the parent of the current directory
- . is shorthand for the current directory
- * is wildcard

Demo #1

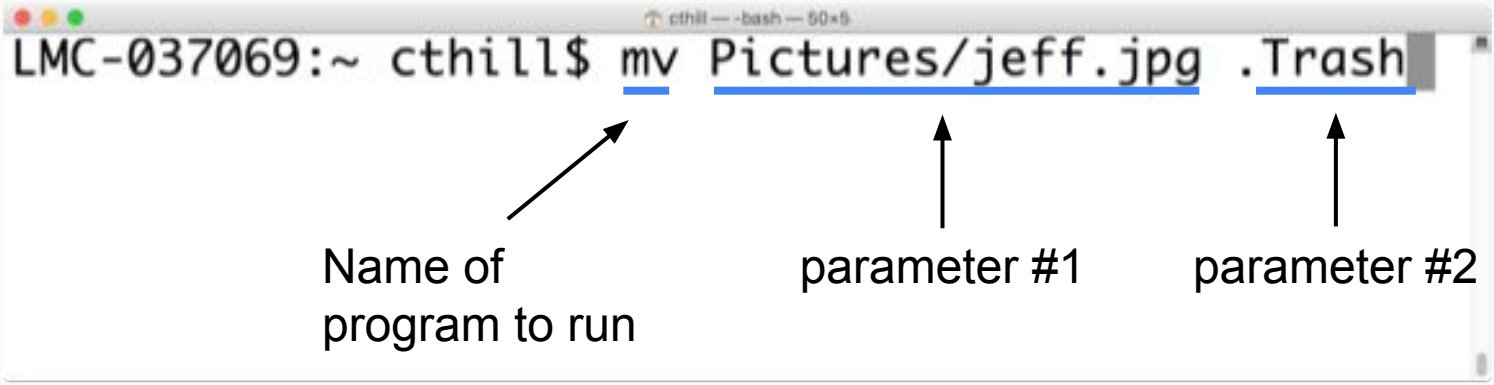
Anatomy of a command

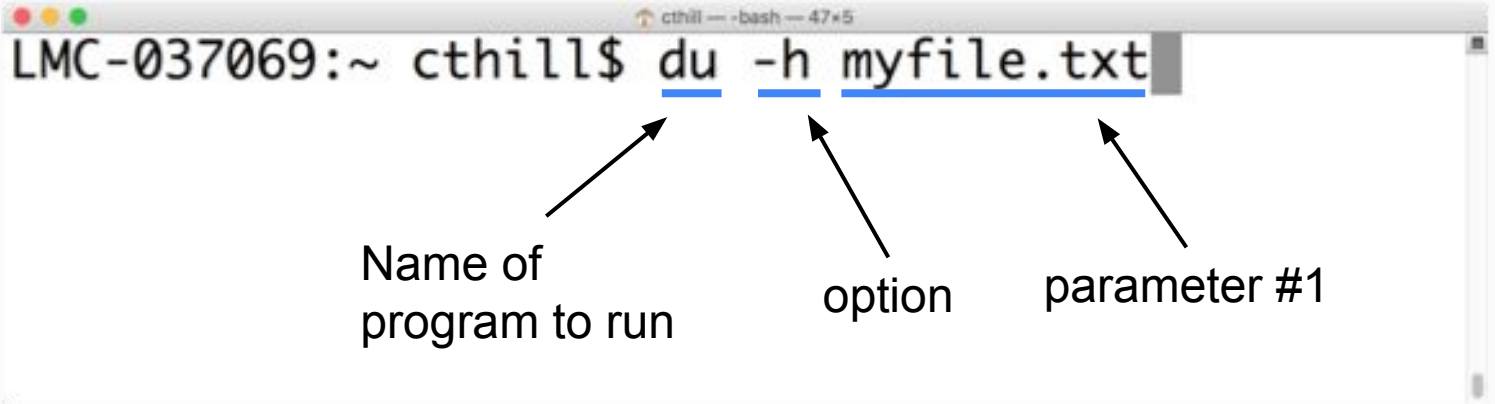
Three parts:

- Name of program to run
- Parameters (also call arguments)
- Options (also called flags or switches)
  - Prefixed by - or --

- All components of a command are separated by spaces
- Parameters and options are not required

Anatomy of a command





Basic commands

- ls
- cd
- mkdir
- pwd
- touch
- mv
- cp
- rm
- echo
- cat
- man
- ping
- curl
- less
- du
- file
- ps
- grep

Demo #2

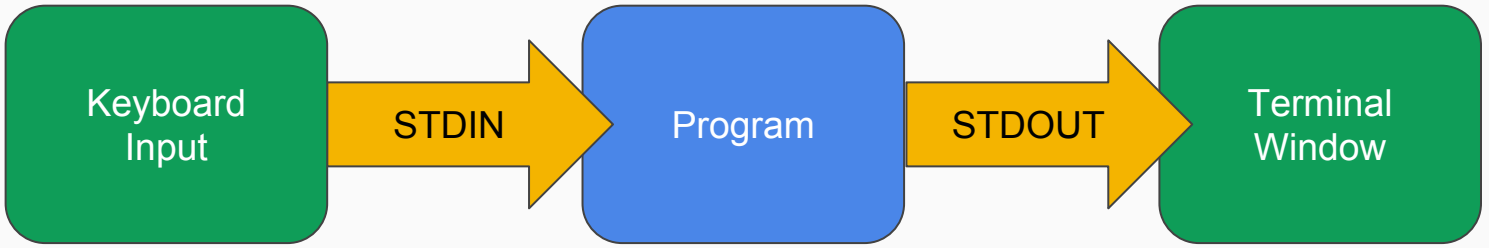
Long running commands

- Not all commands immediately exit
- Some take time to process and exit when they are done
- Others stay running until you stop them
- To exit, use:
  - Ctrl-d (tells the terminal to send EOF on stdin)
  - Ctrl-c (sends a SIGINT to the foreground process)

Demo #3

Standard Input and Output

- Every program has standard in (stdin) and standard out (stdout)
  - Also stderr
- By default, keyboard input is sent to stdin and stdout output is printed to the terminal

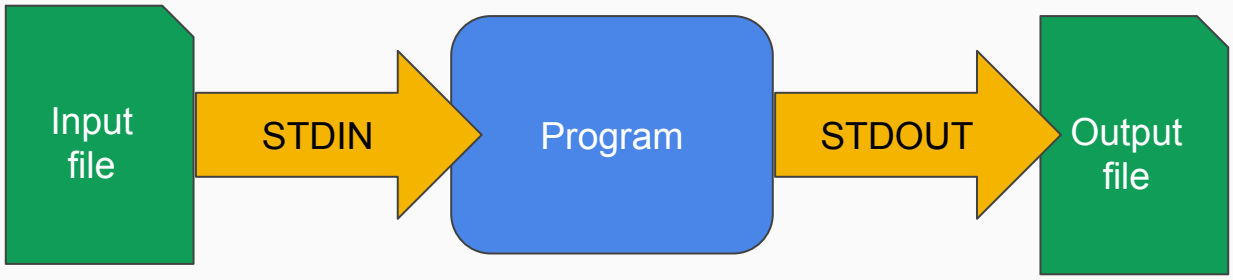


Demo #4

File Redirection

- What if we don’t want to input data with the keyboard?
- Or if we want to save the output in a file?
- We can use file redirection to accomplish this
- The > symbol will write stdout to a file
  - Be careful, > will overwrite files without warning or confirmation
  - Use >> to append instead of overwriting
- The < symbol will write a file to stdin

File Redirection Syntax

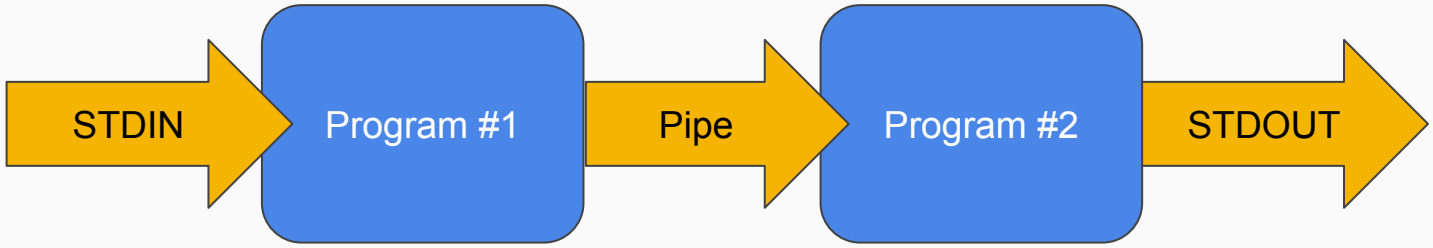


- ps -ax > processes.txt
- grep ssh < processes.txt
- grep ssh < processes.txt > matches.txt

Demo #5

Pipes

- Unix Pipes are used to connect the stdout of one program to the stdin of another
- Use the | symbol



Pipes Demo Video

AT&T Archives: The UNIX Operating System (1982)

Brian Kernighan @ 5m31s on Unix Pipes

https://www.youtube.com/watch?v=tc4ROCJYbm0&t=5m31s

Daemons

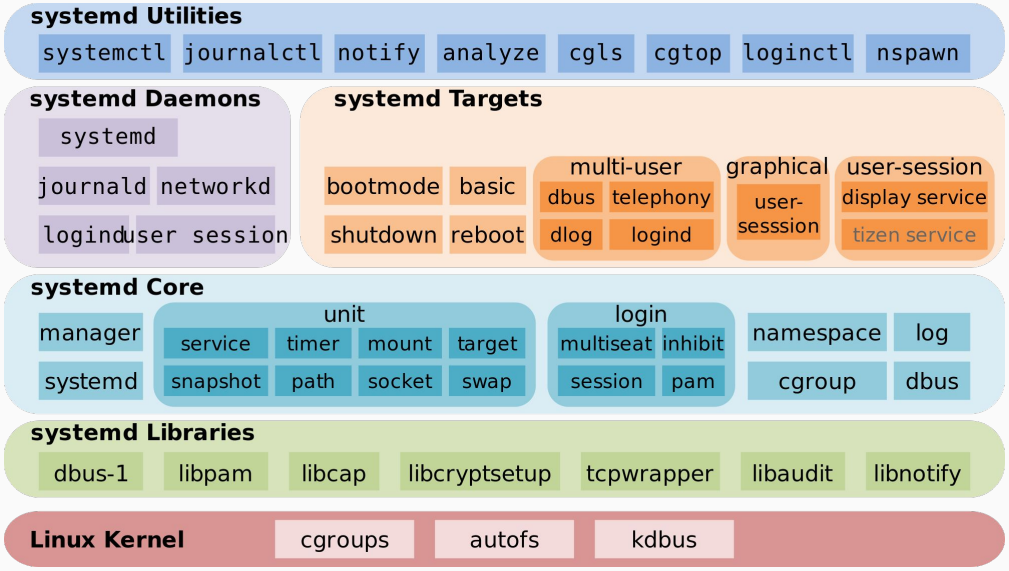
- Daemon are background processes
  - No gui, no user interaction
  - A Unix daemon is similar to a Windows service
- Names of daemons end in d
  - e.g. sshd, crond, ftpd, etc..
- Most often child processes of the init system

Init system

- Started by the kernel during boot process
  - Has process id of 1
- The init process is responsible for setting up userland
  - Ancestor of all other processes
  - Starts all necessary services in correct order
- Continues running until system shutdown

systemd

- Default init system in most linux distributions
- More than just a daemon



Using systemd to create a daemon

- Two main concepts: service units and targets units
- A systemd service unit is a configuration file that describes the process you would like to run
- A target unit is grouping mechanism that systemd uses to start up groups of processes at the same time

Sample systemd unit

```
1  [Unit]
2  Description=MyAwesomeApp
3
4  [Service]
5  WorkingDirectory=/home/christian/www/
6  ExecStart=/usr/bin/python -m SimpleHTTPServer 8888
7
8  [Install]
9  WantedBy=multi-user.target
```

Using systemd to create a daemon

- Systemd unit files go in /etc/systemd/system
- Most daemons are grouped under multi-user.target
  - multi-user.target group will start when the whole system is up but before the gui starts
  - Use graphical.target if you want your daemon to run after the gui starts
- Once you create your service file, run:
  - sudo systemctl enable /etc/systemd/system/<myservice.service>
  - sudo systemctl start myservice

Demo #6

Thanks!