

# Linux Workshop

## Spring 2021

Jack Leightcap<sup>12</sup>

<sup>1</sup>IEEE – nuieeeofficers@gmail.com

<sup>2</sup>Wireless Club – nuwirelessclub@gmail.com

January 25, 2021

# What is Linux?

- Linux is a *kernel* – one of the first program that launches in the boot process, which manages interactions between hardware and software
- Software usually bundled with Linux is commonly referred to as part of Linux as an umbrella term
- Linux, with this additional software, is an *Operating System (OS)*

## What makes Linux unique?

- Free and open source – <https://github.com/torvalds/linux>
- Source code makes debugging and extending (comparatively!) easy
- Modular components: small low power systems to large complex systems
- 10,000+ contributors, regular release cycle

# The UNIX Philosophy

Peter H. Salus, *A Quarter-Century of Unix*:

- Write programs that do one thing and do it well.
- Write programs that work together.
- Write programs to handle text streams, because that is the universal interface.

Opposed to *Zawinski's Law*:

- Every program attempts to expand until it can read [e]mail. Those programs which cannot so expand are replaced by ones which can.

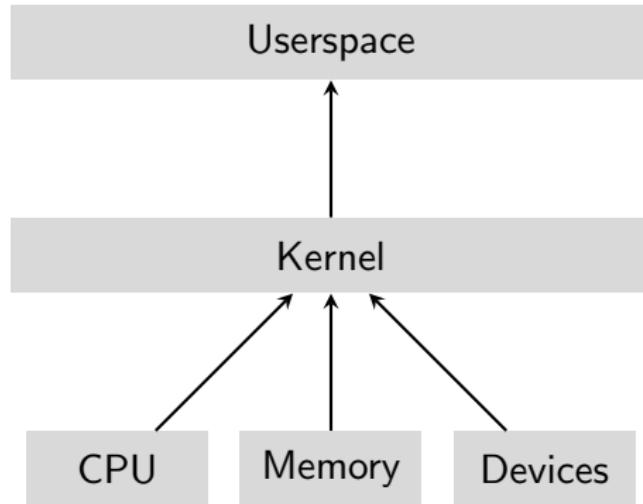


Figure: General System Hierarchy

Linux/x86 5.8.8 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ---). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.  
Legend: [\*] built-in [ ] excluded <M> module < > module capable

```
|| General setup --->
[*] 64-bit kernel (NEW)
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
    Firmware Drivers --->
[*] Virtualization (NEW) --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
--> Enable the block layer --->
    IO Schedulers --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
```

v(+)

<Select> < Exit > < Help > < Save > < Load >

Figure: menuconfig – Linux configuration

# What is Linux Used For?

From 'big' to 'small':

- Servers ('The Cloud')
- Supercomputers
- Personal computers (Linux Distros, OS X is a 'cousin' of Linux)
- Android kernel
- Embedded systems and IoT Devices (routers, video game consoles, smart TVs, probably some fancy toasters)

And many things in between

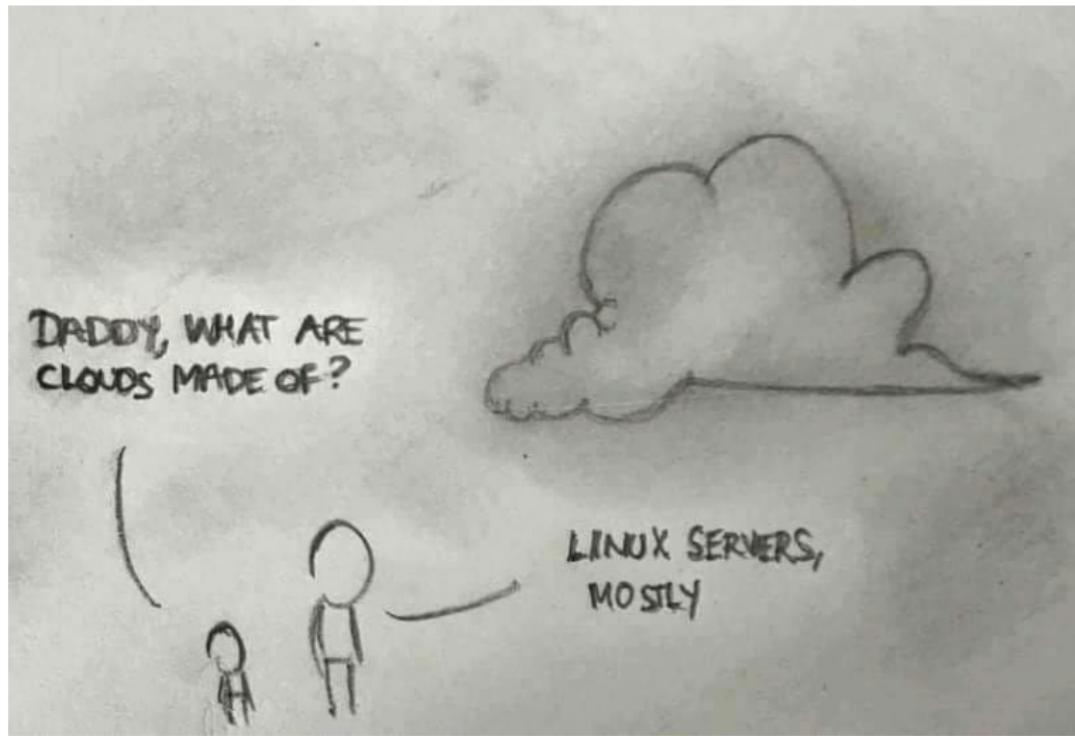


Figure: Source



Figure: Source

```

*1 2 3 4 5 6 7 8 www []= uxterm
dum-4.7

checking whether gcc accepts -g... (cached) yes
checking for gcc option to accept ISO C89... (cache >
d) none needed
checking dependency style of gcc... (cached) gcc3
checking if RANDR is defined... yes
checking if RENDER is defined... yes
checking if XV is defined... yes
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for XORG... yes
checking for ANSI C header files... (cached) yes
./configure: line 21341: XORG_MANPAGE_SECTIONS: command not found
./configure: line 21342: XORG_RELEASE_VERSION: command not found
configure: creating ./config.status
/bin/sh ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating man/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
cd . && /bin/sh /root/xf86-video-cube/missing --run
autoheader
rm -f stamp-h1
touch config.h.in
cd . && /bin/sh ./config.status config.h
config.status: creating config.h
make all-recursive
make[1]: Entering directory `/root/xf86-video-cube'
cd . && /bin/sh /root/xf86-video-cube/missing --run
automake-1.10 --foreign Makefile

```

jleightcap@wii:~\$ piifetch  
jleightcap@wii:~\$

os	Debian GNU/Linux
host	wii
kernel	2.6.32-isobel-w
uptime	0d 4h 36m
packages	482
memory	20660 kB / 7620

jleightcap@wii:~\$ scrot

Figure: Source

# Booting Linux in Web Browser

Ideally in another window,

`http://copy.sh/v86/?profile=linux26`

# Interacting with Linux: the shell

- The *shell* is a text-based interface to the kernel
- a *REPL* (Read → Eval → Print → Loop) of a complete programming language; loops, variables, etc.

After booted:

**/root%** `pwd`      `pwd` – “Print Working Directory”

**/root**

**/root%** `cd /`      `cd` – “Change Directory”

**/%** `ls`

`ls` – “List”

...

# Filesystem Organization

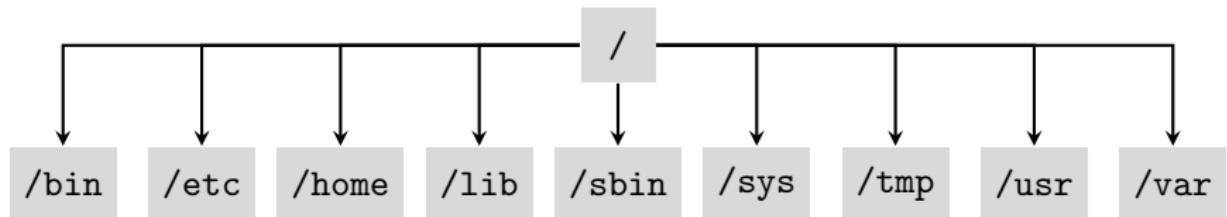


Figure: Common Linux Directory Structure

shorthand	meaning
.	current directory
..	parent (upper) directory
~	home directory

## QUESTION: Parent Directory of /

```
/% pwd
```

```
/
```

```
/% cd ..
```

where are we?

# Special Directories: /dev, /proc, and /tmp

“[text streams are] the universal interface“ – everything is a file!

- /dev: devices
  - Random numbers: /dev/urandom
  - Zero: /dev/zero (surprisingly useful)
  - Physical storage: /dev/hda, /dev/sda, ...
  - RAM: /dev/ram
  - dd if=/dev/urandom of=test count=4 bs=1024
- /proc: processes
  - CPU info: cat /proc/cpuinfo
  - Linux version info: cat /proc/version
  - Processes: ps PIDs compared to ls /proc
- /tmp: temporary, cleared on reboot

## /bin, /sbin, /usr/bin, and /usr/sbin: Programs

- These directories contain programs that can be run anywhere
- ls /bin... see ls there?
- How does the system know where to look for these? A *shell variable!*
  - FOO=hello, get contents of variable with echo \$FOO
  - echo \$PATH
  - Have programs in /root? PATH=\$PATH:/root
- sbin - programs requiring administrator (*superuser*) privileges to run
- /usr/ programs not 'required' for system maintenance
- Exact layout has standards loosely followed, in the end up to user
  - See File Hierarchy Standard – but not hard rules
  - *symlinks* – “Symbolic Links“ have one file or directory ‘point’ to another
  - /usr/local/bin, programs will change default \$PATH, plenty of others...

# Shell Extras

The shell follows the UNIX Philosophy: compose simple programs together to achieve a complex result.

The tool that allow this is redirection:

- [COMMAND] > file – write output of [COMMAND] to file
- [COMMAND]<sub>1</sub> | [COMMAND]<sub>2</sub> | ... – pipe output of one command to the input of the next
- [COMMAND] & – run [COMMAND] in the background.
- foo() { [COMMAND] } define a function foo which does [COMMAND].

Piping examples:

- ls /bin | sort | head -n 1 – first program in bin, alphabetically
- ls /bin | sort | tail -n 1 – last program in bin, alphabetically

## QUESTION: What Do These Do?

```
/% fork() { fork | fork & }
```

```
/% fork
```

```
/% rm -r /
```

# Power and Responsibility...

*Linux gives you the power to shoot yourself in the foot.*

General precautions:

- Don't copy paste a shell command without understanding what it's doing.
- Don't run a script from an untrusted source without verifying it.
- Easy mistakes that can permanently delete files:
  - Moving file over another
  - Writing over a file
  - Over-reliance on tab completion
  - There is no recycling bin in the shell!