

Assignments

- Homework 2 is up. Meaning you send me your Overleaf email address in Slack and I share the homework document with you.
 - Goal – you compile some code, run it on different clusters, answer some questions and make a couple of plots in Overleaf Latex.
 - Along the way I know that you
 - 1) Can compile C code with a makefile (you don't have to write anything)
 - 2) You can setup a conda environment.
 - 3) You can open a jupyter notebook and enter data into a plot
 - 4) Upload that to overleaf so Nick and I can see your progress.

This will be the pattern for all homeworks and projects in this class.

Install Matplotlib with Conda

```
mfricke@hopper:~ $ module load miniconda3
```

```
mfricke@hopper:~ $ conda create --name plotting matplotlib ipykernel
```

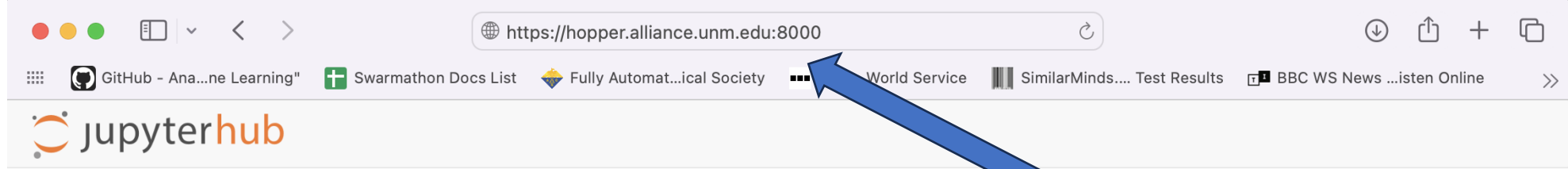
Clone Homework Repository

```
mfricke@hopper:~ $ git clone https://github.com/gmfricke/SIMD_HW
```

Make addition benchmark code

```
mfricke@hopper:~ $ cd SIMD_HW
```

```
mfricke@hopper:~ $ make
```



<https://hopper.alliance.unm.edu:8000>

Sign in

Username:

Password:

Sign in



Logout

Control Panel

Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾

↻

☐ 0 ▾	📁 / SIMD_HW	Name ▾	Last Modified	File size
	📁 ..		seconds ago	
☐	📄 result_plots.ipynb		7 hours ago	121 kB
☐	📄 add.h		7 hours ago	424 B
☐	📄 add_benchmarks.c		7 hours ago	2.37 kB
☐	📄 add_main.c		7 hours ago	2.16 kB
☐	📄 Makefile		7 hours ago	1.06 kB
☐	📄 Readme.txt		7 hours ago	143 B
☐	📄 regular_add.c		7 hours ago	448 B
☐	📄 sse_add.c		7 hours ago	1.08 kB

Not Trusted

Python [conda env:.conda-plotting] ○

File

Edit

View

Insert

Cell

Kernel

Widgets

Help



Run



Interrupt



Restart



Restart & Clear Output

Restart & Run All

Reconnect

Shutdown

Change kernel ▶

Julia 1.5.2

Julia 1.8.5

Python 3

Python [conda env:.conda-MSML]

Python [conda env:.conda-MSML_TF_GPU]

Python [conda env:.conda-MSML_tensorflow_decision_forest]

Python [conda env:.conda-chu122_Final_Macrophage]

Python [conda env:.conda-plotting]

Python [conda env:.conda-polymer_evolution]

R-Seurat

To use this jupyter notebook, you need to install it into that environment.

You can do this with the following steps:

- 1) login to Hopper with your account
- 2) module load miniconda3
- 3) conda create --name plotting matplotlib
- 4) Once the install is complete login to <https://hopper.sdsc.edu> and change the kernels to "Python [conda-env:.conda-plotting]"

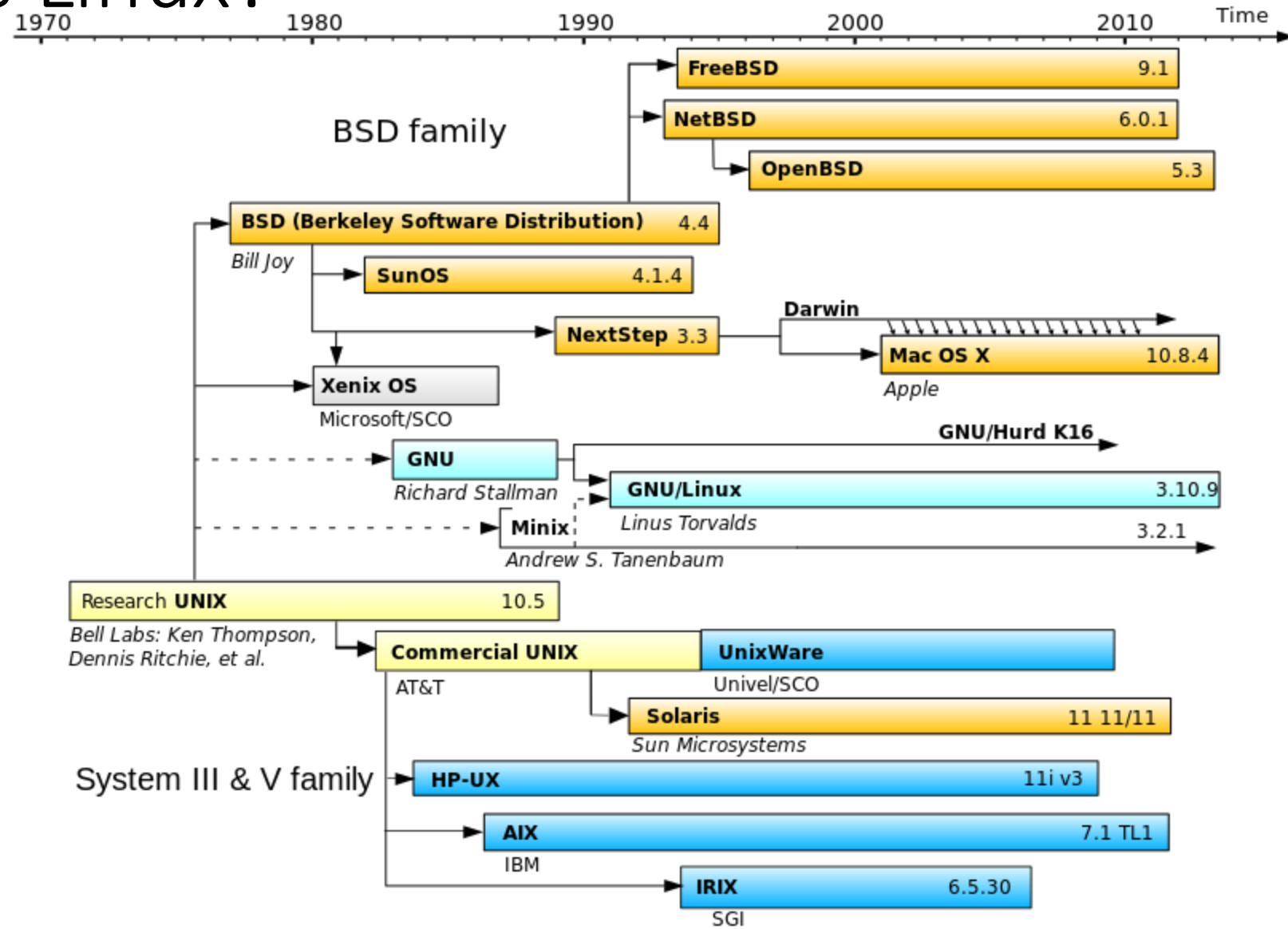
Problem 2

```
In [9]: 1 import matplotlib.pyplot as plt
        2
        3 fig, ax = plt.subplots()
```

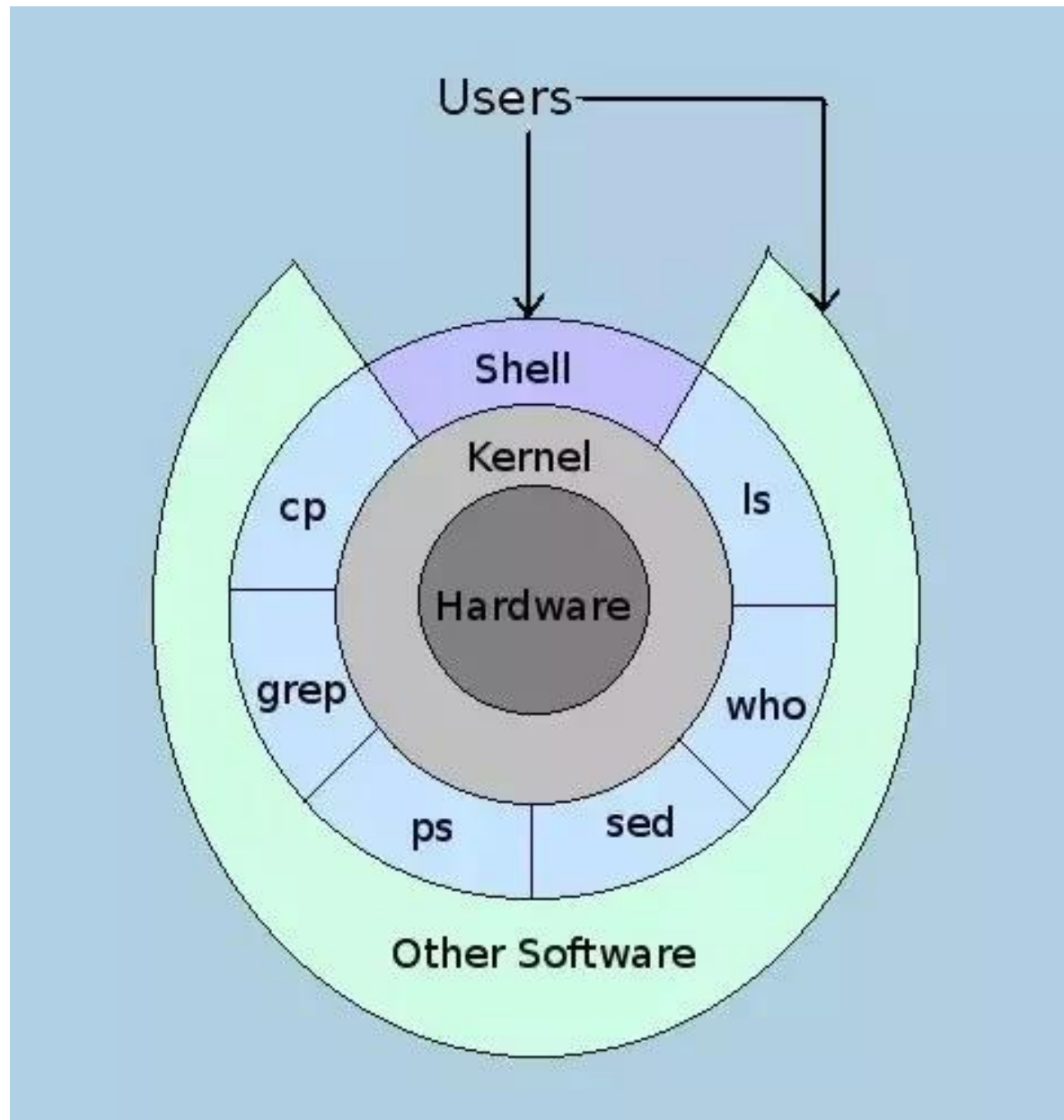
Lecture 4: Devices

High Performance Computers

What is Linux?



The Kernel



What is Linux?

Linux + GNU Utilities = Free Unix



TUX: Torvold's Linux



- Linux is an O/S core written by Linus Torvalds and others

(Bitten by a Penguin)

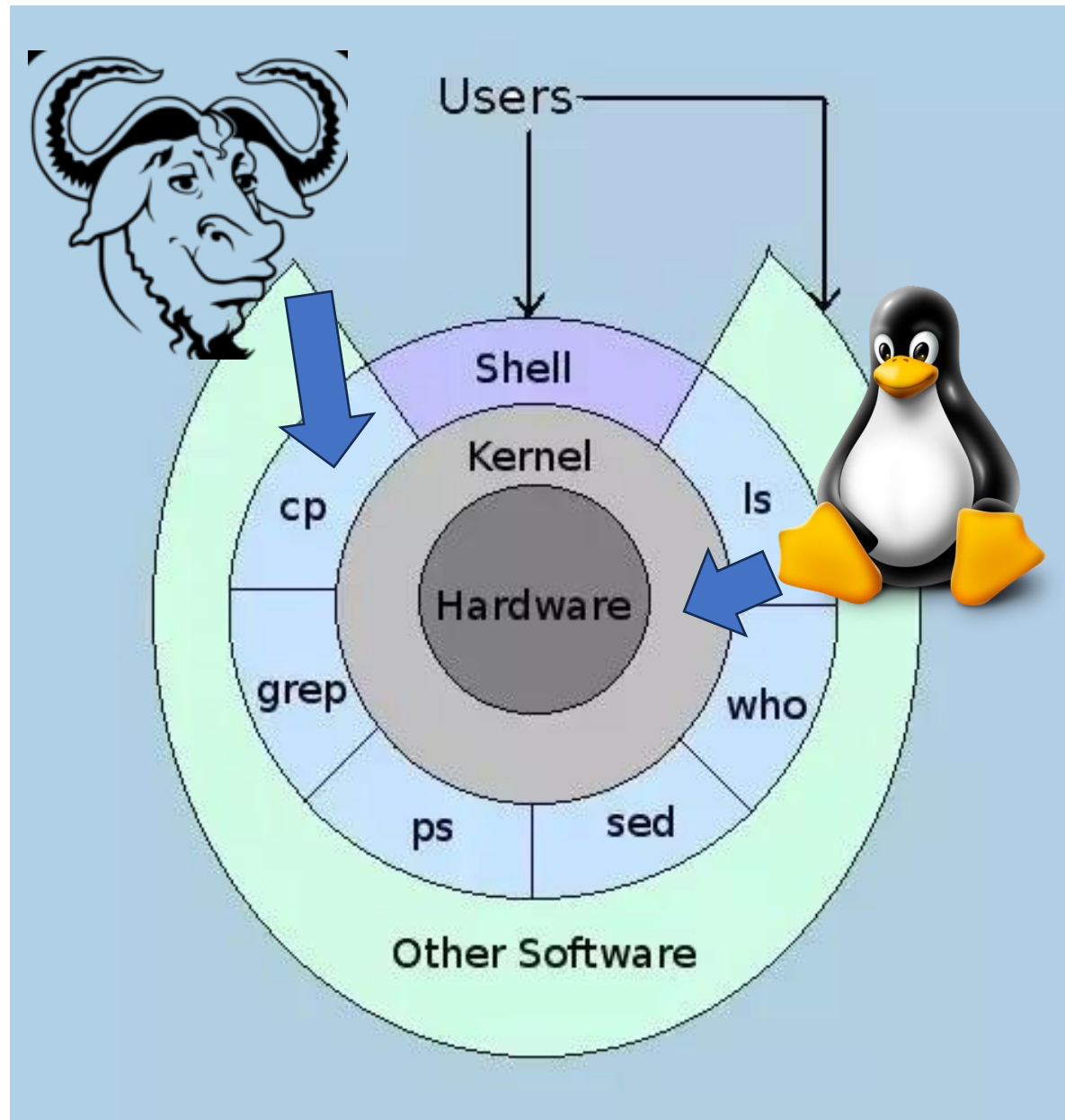


- a set of programs written by Richard Stallman and others. They are the GNU utilities.

<http://www.gnu.org/>

GNU stands for GNU is Not Unix

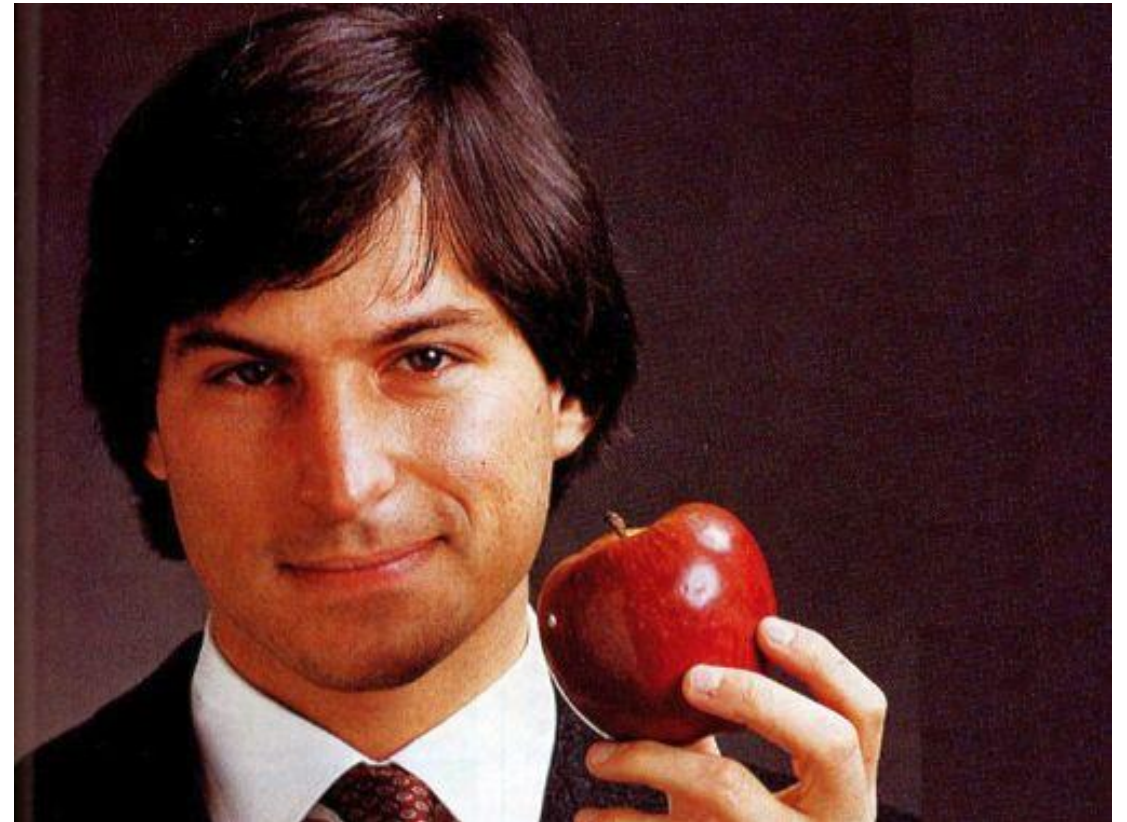
GNU/Linux



Mac OS X



- Darwin, the system on which Apple's Mac OS X is built, is a derivative of 4.4BSD-Lite2 and FreeBSD. In other words, the Mac is a Unix system!
- For X11 (graphics), see XQuartz
(<http://xquartz.macosforge.org/landing/>)



Udevd

- The udevd service monitors the kernel for hardware events:

Enter `systemctl status systemd-udev`

And creates device files based on those events.

Device Files

- Most everything in Linux is a file.
- A file in Linux is anything that supports simple input and output including many devices.
- Interacting with a file is really just a way to ask the kernel to get or put data somewhere
- Device files are stored in the /dev directory
- Many device files point to a program that generates or consumes data

Enter `ls /dev`

Enter `head -n 1 /dev/urandom`

Device Files

- Device files are stored in the /dev directory

Enter `ls /dev`

- Many device files point to a program that generates or consumes data

Enter `head -n 1 /dev/urandom`

Provides a stream of random output (e.g. for creating encryption keys)

Enter `echo blah blah > /dev/null`

Sometimes a process will generate output we don't care about. We can send it to /dev/null where it is consumed and lost.

E.g. when some programs are run for days at a time and they sent informational output to standard out they can consume all available memory. Instead, we throw away the output.

Device Files - Types

Devices have permissions just like other files:

Enter `ls -l /dev`

- The type of device is given by the characters before the permission triplet:
- “b”: block – random access device
- “c”: character – data streams (e.g. /dev/null, /dev/urandom)
- “p”: pipe – a data stream from one process to another (unidirectional)
- “s”: socket – a data stream between two processes (bidirectional)

Devices without Files

- OK, not everything is a file. Network cards don't have a device file.
- Device files have read and write functions that take bytes, but networks work on packets so their read and write functions are more complicated.
- Solaris systems did have device files for network cards btw

SysFS

- /dev provides a simple interface through the kernel to devices
- Downsides are that it is a bit too simple and the devices are named in the order that the kernel found them
- If you remove a device then the names the kernel gives everything else might change
- A more recent approach is to use sysfs to get information about hardware
- Devices files are kept in /sys/devices

Enter `ls -l /sys/devices`

SysFS

- /dev provides a simple interface through the kernel to devices
- Devices files are kept in /sys/devices
 - Enter `ls -l /sys/devices`
- Sysfs is used for managing and getting info about devices
- /dev is for reading and writing to them
- To find the sysfs file that corresponds to a /dev/ device file use this command:
- Enter `udevadm info --query all /dev/sda`

Is sda on Wheeler a spinning disk or SSD?

How about on your workstation?

Udev

- To monitor udev detected events live:

Enter `udevadm monitor`

After running `udevadm monitor` insert a usb device into your computer. You should see hardware events.

- Login to Hopper: Udev allows us to set rules for how hardware is handled:

Enter `ls /lib/udev/rules.d/`

dd command: Data Duplicator

- dd's function is to read and write block or stream data to device files.
- This makes it a powerful tool*

Enter `dd if=/dev/random of=/dev/stdout bs=128 count=1`

What do you see. What do you think /dev/stdout is?

*Nicknamed the disk destroyer

/dev/stdout

Let's take a closer look at /dev/proc/stdout

```
ls -l /dev/stdout
```

```
lrwxrwxrwx 1 root root 15 Aug 23 09:08 /dev/stdout -> /proc/self/fd/1
```

/proc contains a directory for every running process
(many listed by process ID)

/proc/self refers to the process reading the filesystem (your terminal)

Every process has a file descriptor (fd) so it can communicate.

fd 1 is the standard character stream to the screen (it's a socket)

fd 2 is standard error

When you write `cout` in C++ or `print()` in python you are sending a character stream to `/proc/self/fd/1`

dd command: Data Duplicator

You can also write block data.

```
dd if=/dev/urandom of=/dev/sdX bs=1M
```

Will write the stream of random characters from /dev/urandom to SCSI drive X in 1 MB chunks.

This will overwrite everything on drive X.

Hard Disk Device Files /dev/sd*

- SCSI: Small Computer System Interface
- Linux uses SCSI protocols to talk to disks even SATA disks

You can list SCSI devices on the Wheeler and Hopper head nodes:

Enter `ls SCSI`

Enter `cat /proc/scsi/scsi`

(Again “everything in Linux is a file”)

Virtual Disk Device Files /dev/xvd* /dev/vd*

- Some disks are optimized for virtual environments such as Amazon Web Services or VirtualBox.
- Basically, a Virtual Disk Device allows virtual machines to communicate with the real disk faster.

Non-Volatile Memory Device Files

/dev/nvme*

- NonVolatile Memory **Express** (NVME) is a protocol for talking to solid state drives (SSDs) and flash drives.
- SATA supports up to 6 Gbit/s, which was a lot faster than hard drives used to be (even older SSDs).
- SCSI devices support a queue of up to 2^8 commands at a time. SAS drives up to 2^8 so they can handle multiuser systems with lots of requests. SATA is intended for single user workstations and has a typical queue depth of 2^5 commands.

Enter `ls SCSI -l` to see the queue depth

- PCI **Express** sockets are much faster than SATA connections with some capable of 32 Gbit/s since recent SSDs are capable of 25 Gbit/s speeds and support 2^{16} commands in the queue.

Device Mapper Device Files (LVM) /dev/dm* or /dev/mapper*

- A challenge with disk devices has always been that they are hard to change.
- When you partition a disk into filesystems (more later) it is hard to change the size of the partition when it fills up.
- Logical Volume Management (LVMs) solve that problem by abstracting the disk and allowing you to change the virtual disks size even while the OS is running.
- If you run out of space, you can just add another physical drive to the LVM and expand the partition.
- Enter `ls /dev/mapper*` and `df -h | grep mapper` on wheeler to see that we use LVMs.

Terminal Device Files /dev/tty* /dev/pts*

- Terminals are devices that take character streams from processes and send them to and from Input/Output devices like this VT100 from 1978.
- Over the years physical terminals were replaced with terminal emulators but the protocols remained basically the same. It is always easier to reuse old code.
- The VT100 protocol is still used because it supports ANSI colors.
- You are typing and reading text in a virtual terminal whenever you use BASH.



Terminal Device Files `/dev/tty*` `/dev/pts*`

- Terminals that read directly from a local keyboard and write to a computer monitor are named `/dev/tty*`
- Terminals that connect to a remote program are called pseudoterminals and they are named `/dev/pts*`

Terminals that run over SSH are pseudoterminals.

Enter `tty` to get the path to the terminal to which you are connected. E.g. `/dev/pts/3`. Try it on hopper.

Now try writing to the terminal

Enter `echo Hello > /dev/pts/3`



Terminal Device Files `/dev/tty*` `/dev/pts*`

Let's read from your terminal:

Enter `read data < /dev/pts/3`

This stores whatever character stream you enter next into the variable "data".

We can display the contents of that variable:

Enter `echo $data`

TTY stands for tele-typewriter



Terminal Device Files

`/dev/tty*` `/dev/pts*`

Open a new terminal and ssh into hopper
(so you have two ssh sessions on Hopper).

Enter `tty` to get the device file path of
your new SSH pseudoterminal.

From your first terminal

Enter `echo Hello > /dev/pts/#`

Where `#` is the device file for your second
terminal.



Terminal Device Files `/dev/tty*` `/dev/pts*`

- Virtual terminals read and write to hardware connected directly to the computer (`/dev/tty*`)
- Many terminals have a `getty` process that waits for input on a terminal. Gettys guard access by asking for a login name and password.
- You can switch between virtual terminals with `Ctrl-Alt-F#` (F here is the function keys on your keyboard and # is the number of the device file).
- So to switch to Terminal 1 on your Linux machine enter `Ctrl-Alt-F1`.
- This is used all the time for troubleshooting. If the graphical terminal that comes up breaks and I want to login to Linux to fix it, I hit `Ctrl-Alt-F1` to get a local text-based terminal so I can login.
- Try it on your Linux workstation.

Terminal Device Files /dev/tty* /dev/pts*

- Terminals can be text or graphical.
- If you installed a GUI with Linux then a graphical Greeter will be waiting for input from one of the TTY device files.
- Keep pressing Alt-F# incrementing # until you get to the graphical terminal.



Udev Overview

- Udev is responsible for creating device files, and does the following:
 1. The kernel sends a **uevent** message to the systemd-udevd service.
 2. Udev parses the uvent message
 3. Udev applies all the rules in `/etc/udev/rules.d` and `/lib/udev/rules.d` to decide what to do about the hardware uevent message.

Take a look at the rules for handling mice by entering

```
cat /lib/udev/rules.d/60-nvidia.rules
```

The **mknod** command makes a device files named `/dev/nvidia*`

Enter **grep -R tty /lib/udev/rules.d/**

To see all the rules that create virtual terminals