

# What's Missing?

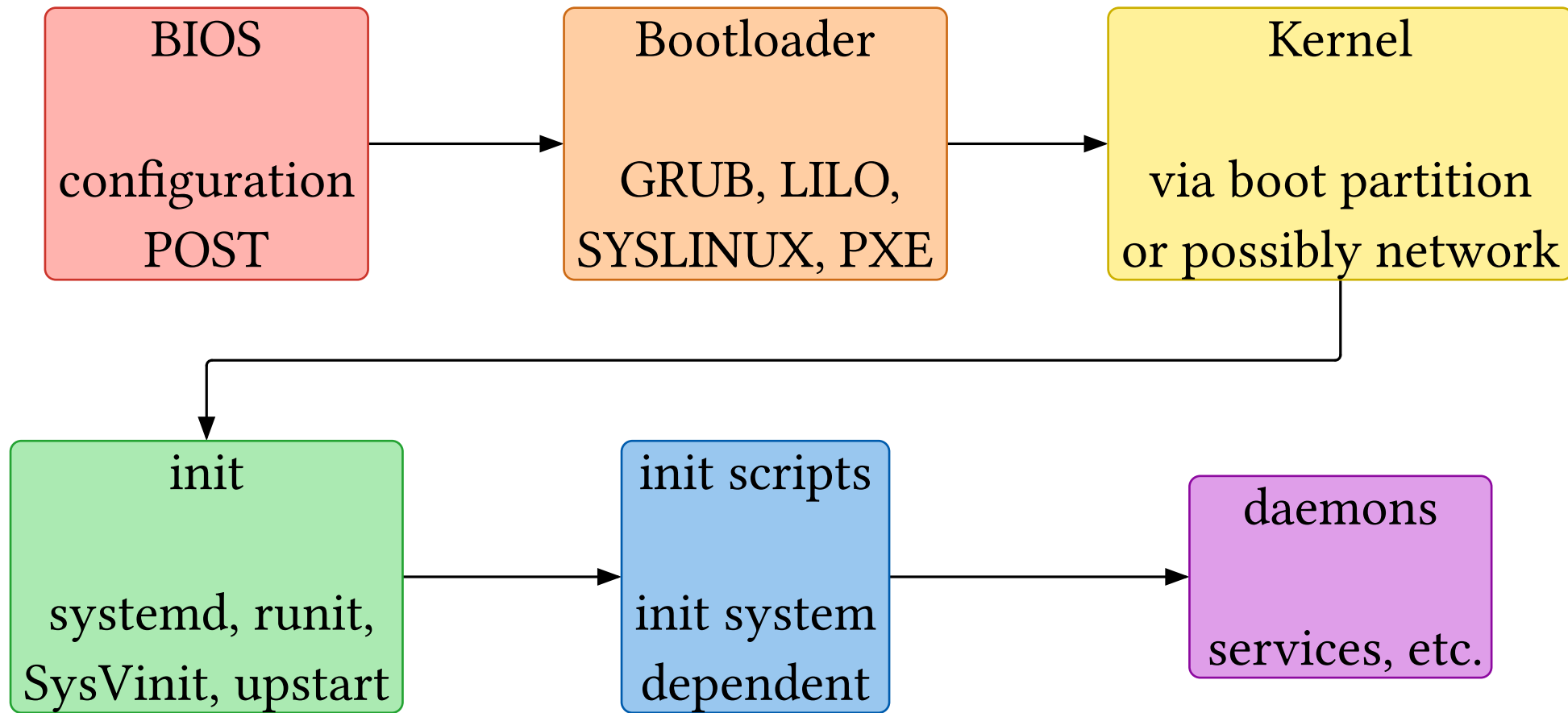
Ryan Tolboom

New Jersey Institute of Technology



We've been learning in a virtualized environment. What haven't we had a chance to work with?

# The Boot Process



# Init System



“Foam Finger Fun!” by [Matthew Peck](#) is licensed under [CC BY-NC-ND 2.0](#)

- PID 1
- SysV was the old way of doing it
- Most modern systems run systemd (it's contentious)
- The init system brings up and monitors daemon processes

# Basic systemctl Commands

- `systemctl list-units --type=service`
- `systemctl start <servicename>`
- `systemctl stop <servicename>`
- `systemctl restart <servicename>`
- `systemctl enable <servicename>`



[“Full Color Logo”](#) by [Tobias Bernard](#) is licensed under [CC BY-SA 4.0](#)

# Processes

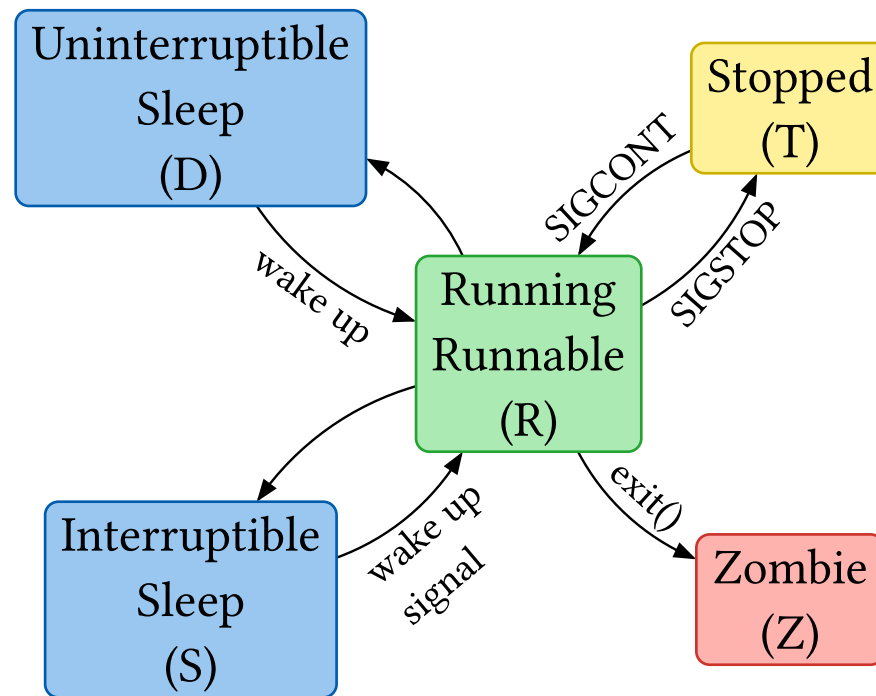
```
guest-mm2x7k@support21: ~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	185352	5984	?	Ss	08:39	0:01	[systemd]
root	2	0.0	0.0	0	0	?	S	08:39	0:00	[kthreadd]
root	4	0.0	0.0	0	0	?	I<	08:39	0:00	[kworker/0:0H]
root	5	0.0	0.0	0	0	?	I	08:39	0:00	[kworker/u8:0]
root	6	0.0	0.0	0	0	?	I<	08:39	0:00	[mm_percpu_wq]
root	7	0.0	0.0	0	0	?	S	08:39	0:00	[ksoftirqd/0]
root	8	0.1	0.0	0	0	?	I	08:39	0:02	[rcu_sched]
root	9	0.0	0.0	0	0	?	I	08:39	0:00	[rcu_bh]
root	10	0.0	0.0	0	0	?	S	08:39	0:00	[migration/0]
root	11	0.0	0.0	0	0	?	S	08:39	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	08:39	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	08:39	0:00	[cpuhp/1]
root	14	0.0	0.0	0	0	?	S	08:39	0:00	[watchdog/1]
root	15	0.0	0.0	0	0	?	S	08:39	0:00	[migration/1]
root	16	0.0	0.0	0	0	?	S	08:39	0:00	[ksoftirqd/1]
root	18	0.0	0.0	0	0	?	I<	08:39	0:00	[kworker/1:0H]
root	19	0.0	0.0	0	0	?	S	08:39	0:00	[cpuhp/2]
root	20	0.0	0.0	0	0	?	S	08:39	0:00	[watchdog/2]
root	21	0.0	0.0	0	0	?	S	08:39	0:00	[migration/2]
root	22	0.0	0.0	0	0	?	S	08:39	0:00	[ksoftirqd/2]
root	24	0.0	0.0	0	0	?	I<	08:39	0:00	[kworker/2:0H]
root	25	0.0	0.0	0	0	?	S	08:39	0:00	[cpuhp/3]

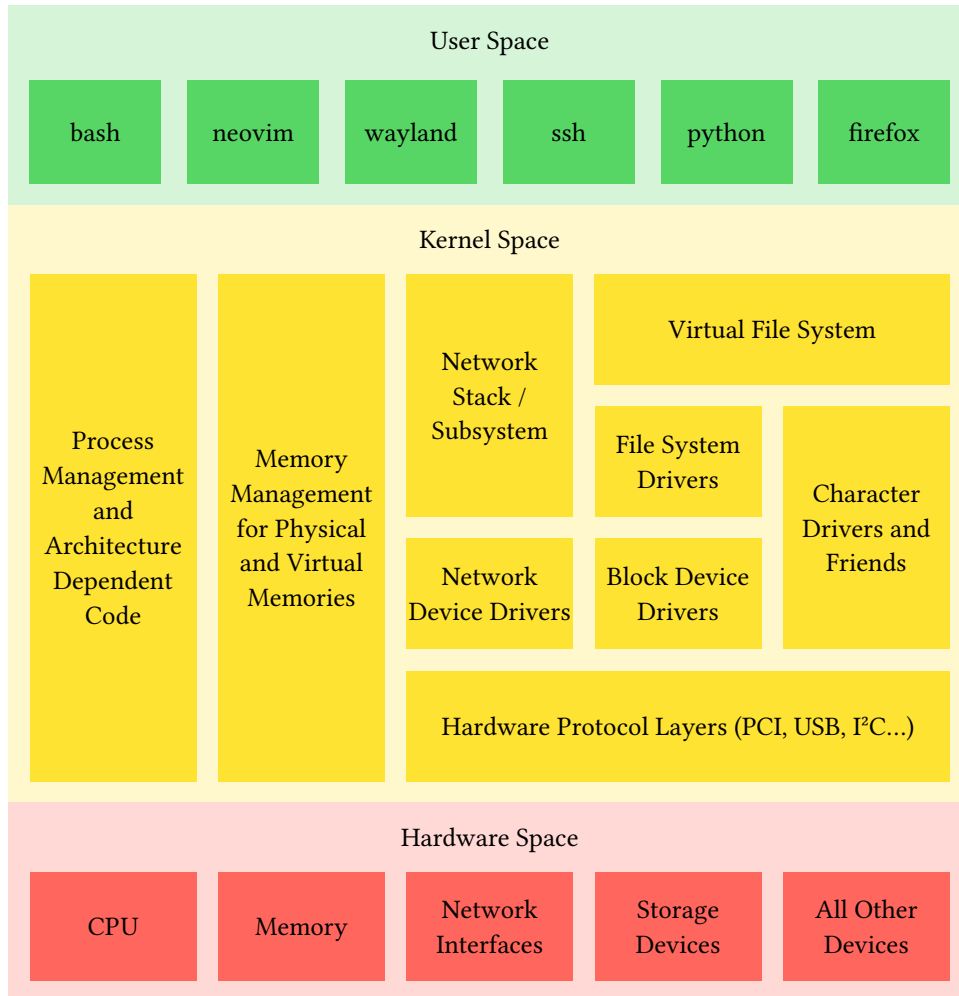
- OS kernel allows for multiple things to run at once
- A process is one of those things
- The kernel scheduler splits time between them
- [This can be adjusted!](#)

# Tuning the Process Scheduler

- `/proc/sched_debug` shows all tunable variables
- `sysctl` (not `systemctl`!) can be used to adjust them
- `chrt` shows the real-time attributes of a running process
- If you make changes, don't forget to make them permanent! (`/etc/sysctl.conf`)



# Devices



- Real systems have real devices
- I/O is a typical bottleneck in production systems
- [Sysfs allows for tuning of I/O devices](#)
- I/O also has schedulers



- A list of rules that determines what to do/create when a device is added
- Devices can have persistent names through devfs (can be very useful for USB)
- Initialization can take place automatically

[/lib/udev/rules.d/80-usb.rules](#)

```
KERNEL=="sd*", SUBSYSTEMS=="scsi", ATTRS{model}=="USB 2.0 Storage Device", SYMLINK+="usbhd%n"
```

# General Advice for Tuning Linux

- Determine your metric in advance!
- Take slow steps and monitor changes
- Be prepared to walk-back changes



[Advice Icon](#) is licensed under [the uxwing license](#)