CSCI 460 Operating Systems

The Process (Part I)

Professor Travis Peters
Fall 2019

Some slides adapted from Stallings resources.

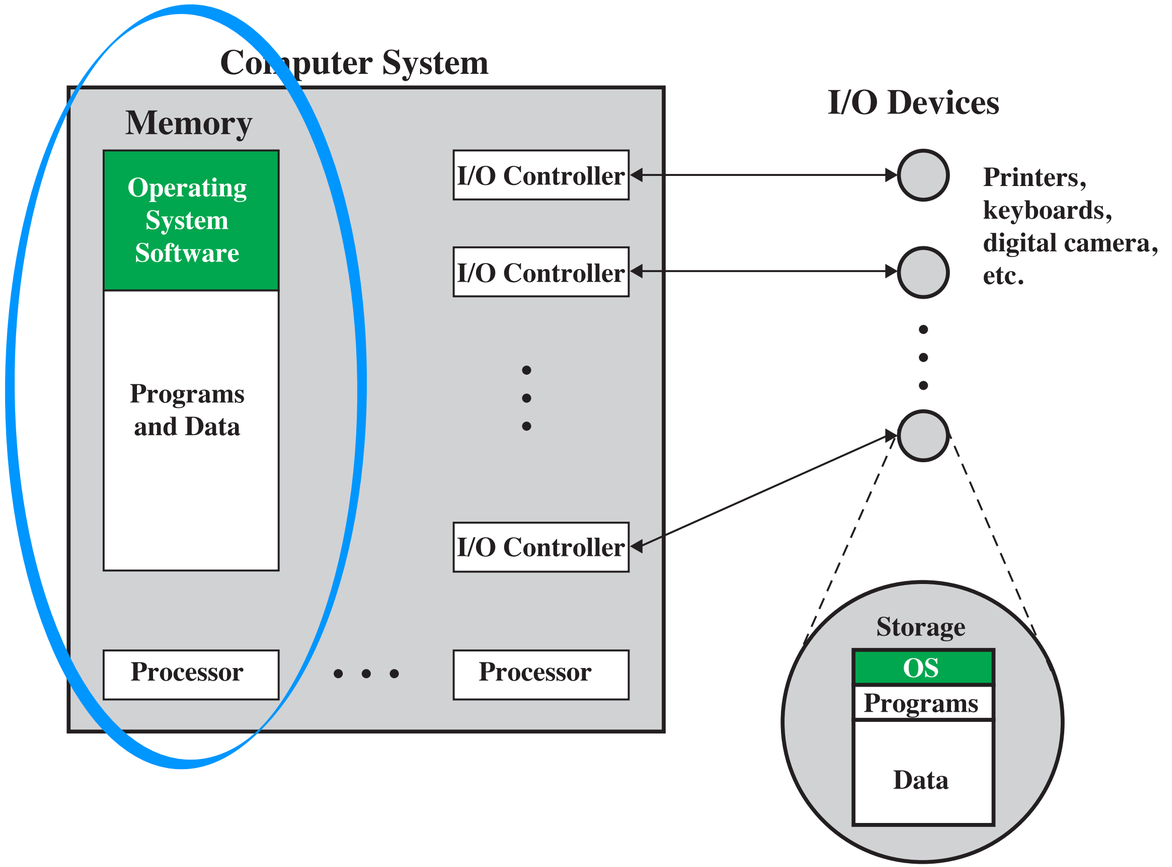
Some slides adapted from Adam Bates's F'18 CS423 course @ UIUC
https://courses.engr.illinois.edu/cs423/sp2018/schedule.html

Goals for Today

- **Learning Objectives**
  - Understand basic concept of process (control info, creation, termination, states, etc.)
  - Review some important UNIX syscalls and concepts for sys. programming
- **Announcements**
  - Grading scheme updated in D2L
    - *still subject to change at any time…*
  - 8 people missing submission for HW1…
    - *6? 2 unnamed papers turned in…*
  - *Coming Soon…*
    - Rough schedule through the 1st exam to be published soon.
    - 1st programming assignment
    - Details about project

Today in Context

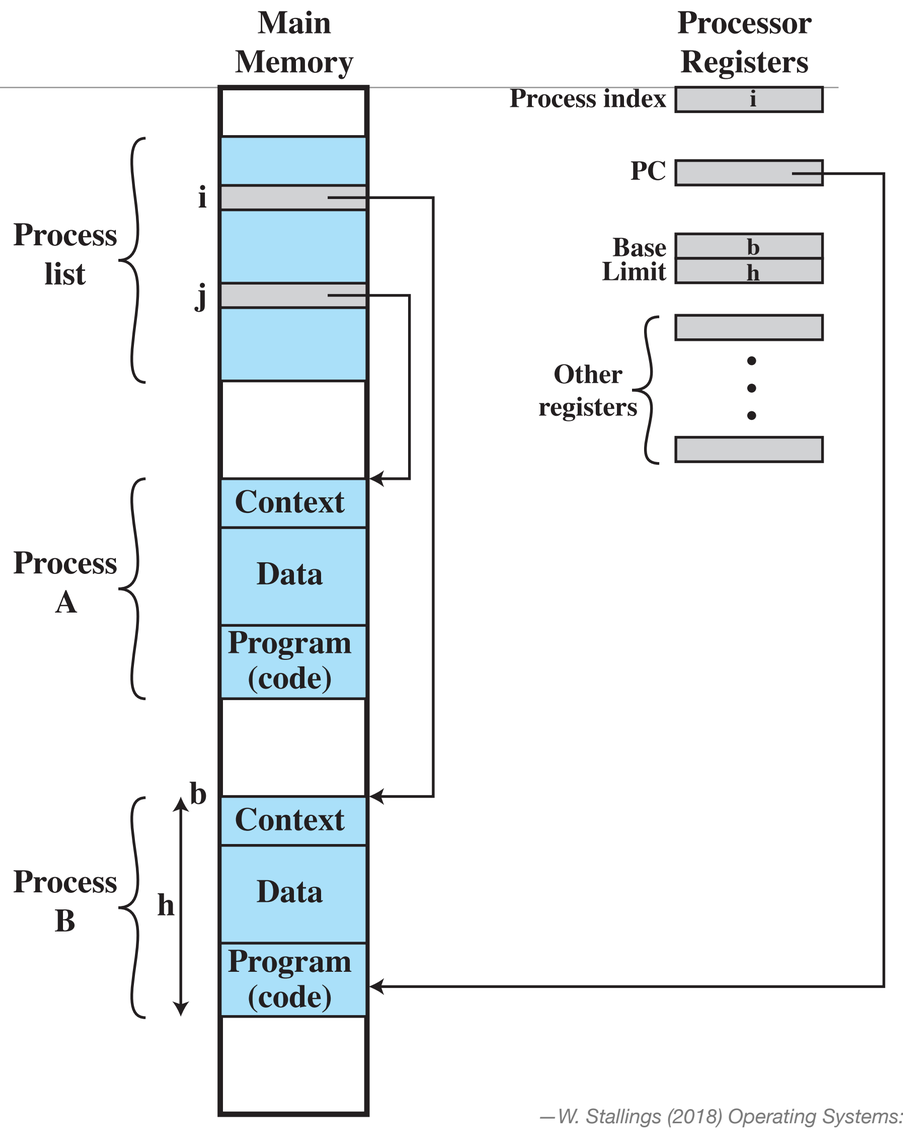
- Focus on ***Processes*** and how…
  - they work
  - they are represented
  - they are managed
  - etc.



—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

The Process

- The notion of a ***Process***…
  - many definitions…e.g.,
    an instance of a program running on a computer;
  - consists of two (three) critical things:
    - (1) an executable program (**code**),
    - (2) assocated **data**,
    - (3) **execution context** (info the OS needs to manage the process)
  - is realized as nothing more than a data structure!



—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

Why Processes?

- **Q:** Why is the notion of a process useful?

Why Processes?

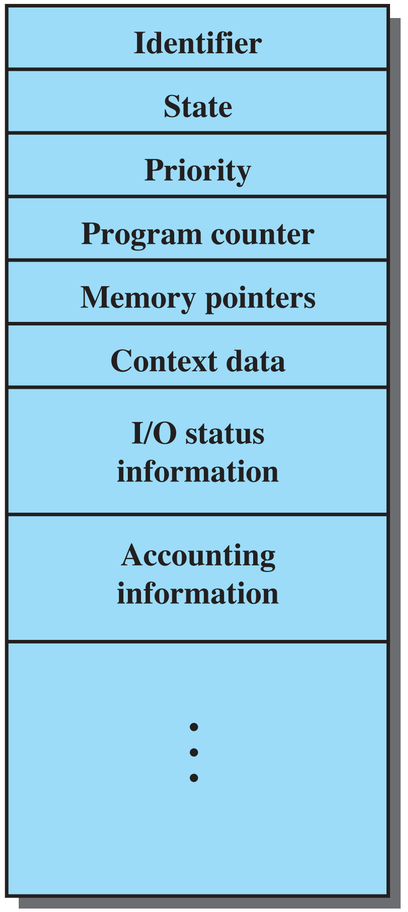
- **Q:** Why is the notion of a process useful?
  - A standard approach for defining exeutable code with associated data
  - An OS abstraction for executing a program w/ limited privileges
  - …

The Process Control Block (PCB) & Getting Process Info

- A “snapshot” that contains all necessary data to restart a process where it left off

- While the program is executing, this process can be uniquely characterized by a number of elements →

- In practice, how to get info about a process?
  - `getpid() //get the proc. ID`
  - `getppid() //get the parent’s proc. ID`
  - `getpriority() //get the proc. priority`
  - etc…



Simplified Process Control Block

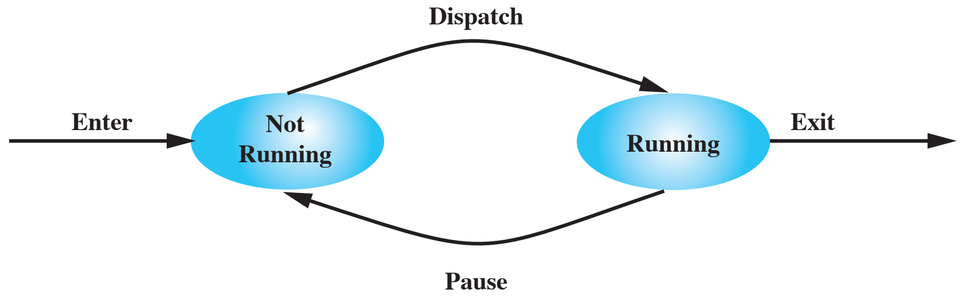
—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

Process States (Simple: 2-States)

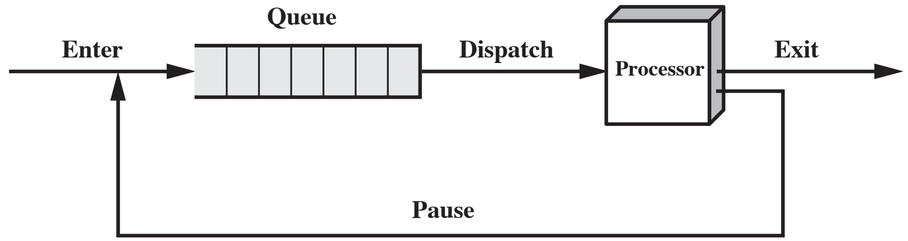
- Simple model:
  - *proc. is either **running** or **not running***
  - *running = on the processor*
  - *not running = not on the processor*
- ***Q:** In a single processor machine, how many processors can be running?*

  1!
- ***Q:** What do we do with the other processes?*

  Hold in a queue; later, interrupt running proc and dispatch next proc.



(a) State transition diagram

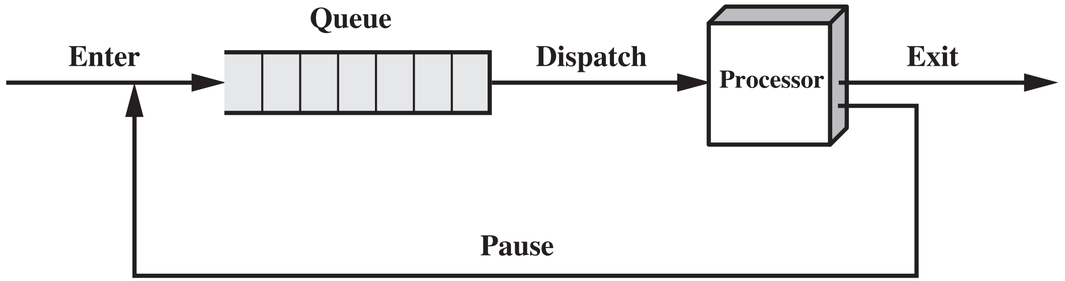


(b) Queuing diagram

—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

Process States

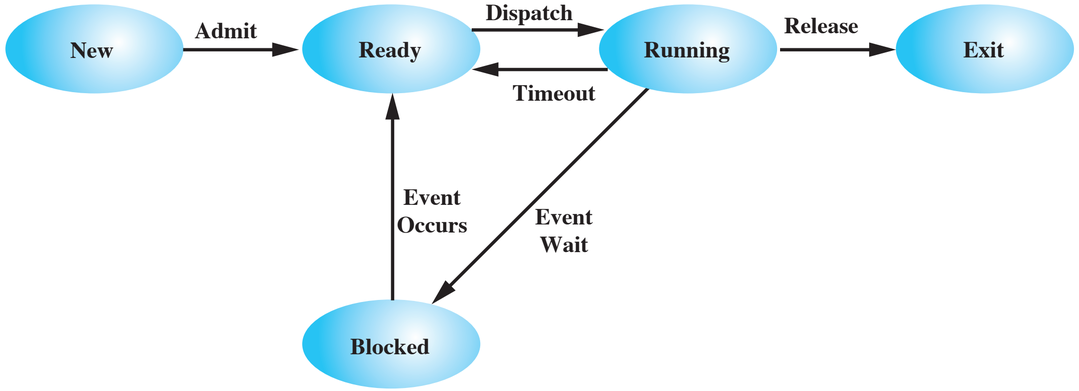
- Not all processes are ready to execute…
  Q: *What problems arise with using a single queue?*

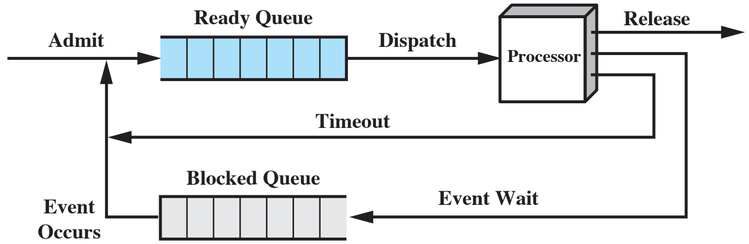


—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

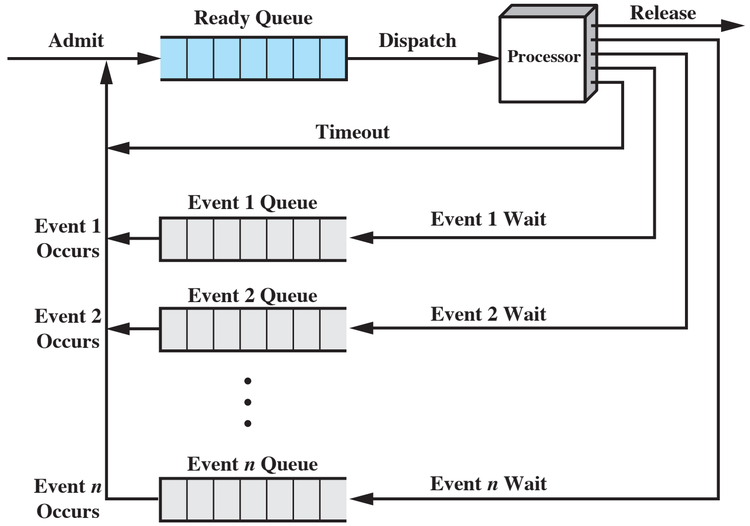
Process States (5 States / Multiple ‘Blocked’ Queues)

- Not all processes are ready to execute…
  Q: *What problems arise with using a single queue?*
- Additional States:
  - **New**—the process is being created
  - **Ready**—the process is ready to run; waiting to be assigned to a processor
  - **Running**—the instructions of the process are being executed
  - **Blocked**—the process is waiting for an event to occur (e.g., I/O)
  - **Exit**—the process has finished execution





(a) Single blocked queue

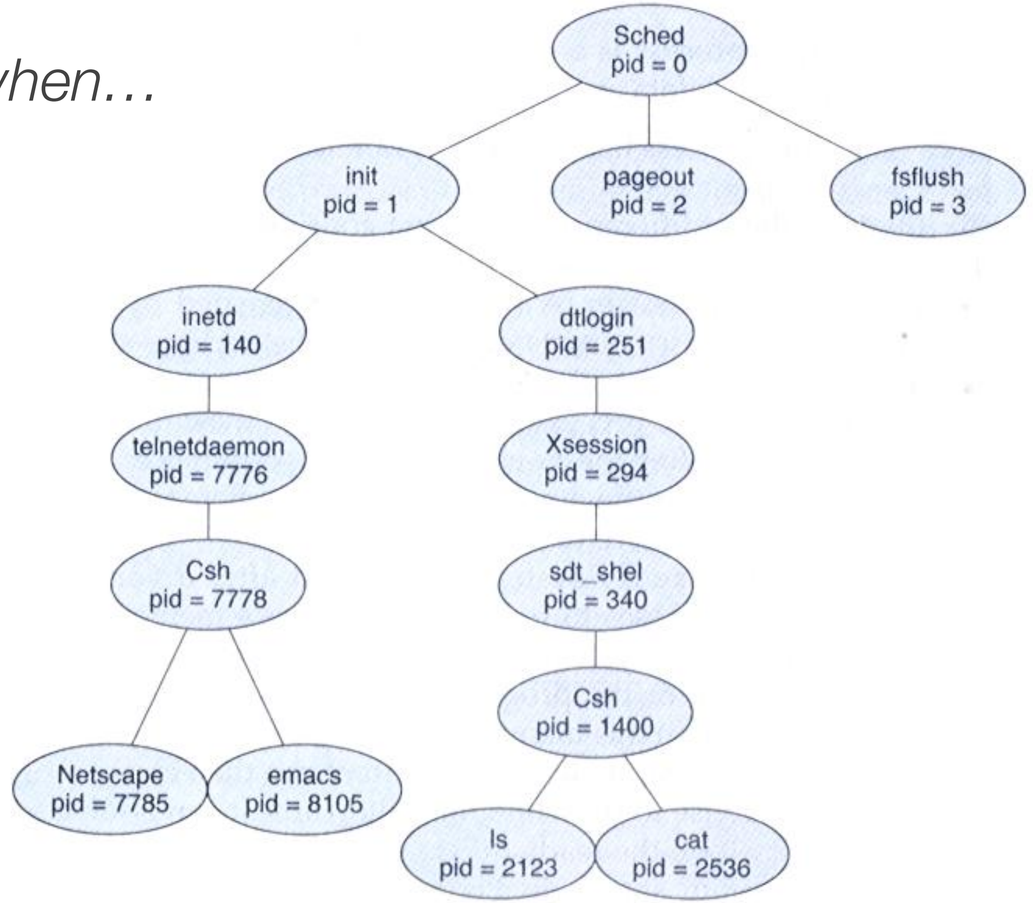


(b) Multiple blocked queues

—W. Stallings (2018) Operating Systems: Internals and Design Principles (9th Edition).

Creating a Process

- **Q:** How do UNIX processes get created?
- *Some discussion in the text—processes created when…*
  - *…system boots,*
  - *…user opens an app,*
  - *…an existing process spawns a child process*
  - *etc.*



Creating a Process

```
vagrant@osbox:~/os$ ps axjf
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
    0     2     0     0 ?           -1 S        0   0:00 [kthreadd]
    2     3     0     0 ?           -1 S        0   0:00  \_ [ksoftirqd/0]
    2     4     0     0 ?           -1 S        0   0:00  \_ [kworker/0:0]
    2     5     0     0 ?           -1 S<       0   0:00  \_ [kworker/0:0H]
    2     7     0     0 ?           -1 S        0   0:01  \_ [rcu_sched]
```

DEMO

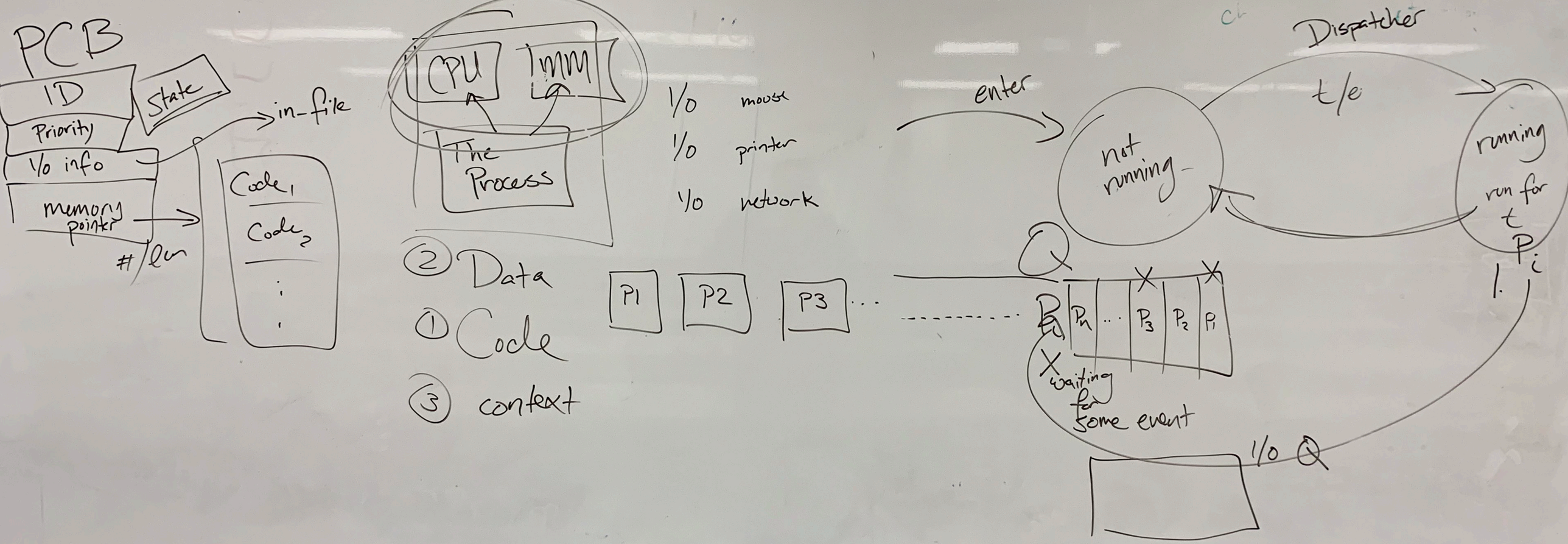
take a look at a process tree (already-created processes)

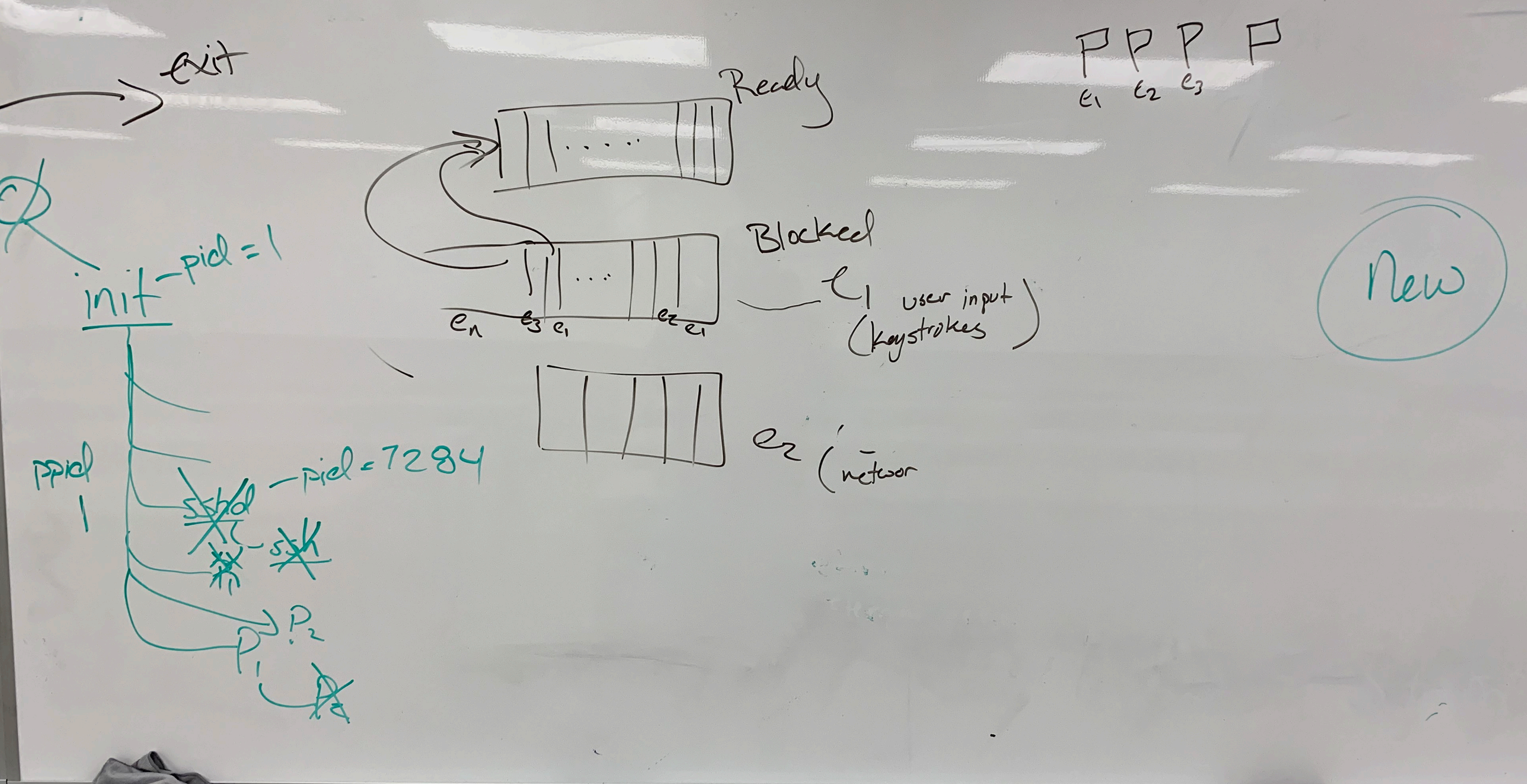
`ps axjf`

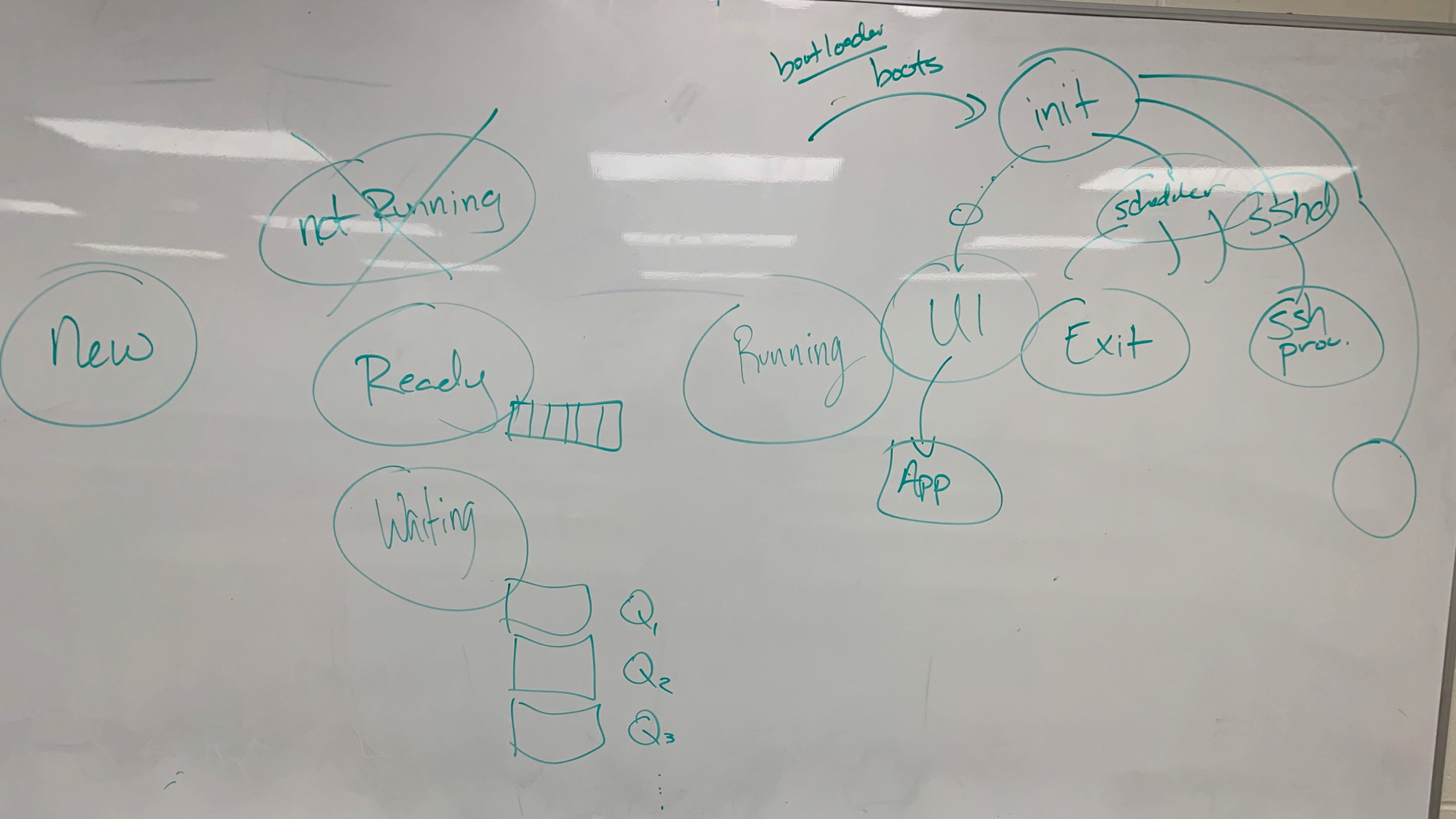
```
    0     1     1     1 ?           -1 Ss       0   0:00 /sbin/init
    1   524   524   524 ?           -1 Ss       0   0:00 dhclient -1 -v -pf /run/dhclient.eth0.pid -lf /va
    1   621   621   621 ?           -1 Ss       0   0:00 rpcbind
    1   663   663   663 ?           -1 Ss     107   0:00 rpc.statd -L
    1   741   741   741 ?           -1 Ss     102   0:02 dbus-daemon --system --fork
    1   806   806   806 ?           -1 Ss       0   0:00 rpc.idmapd
    1   826   826   826 ?           -1 Ss       0   0:00 /lib/systemd/systemd-logind
    1   832   832   832 ?           -1 Ssl    101   0:00 rsyslogd
    1   937   937   937 tty4       937 Ss+      0   0:00 /sbin/getty -8 38400 tty4
    1   940   940   940 tty5       940 Ss+      0   0:00 /sbin/getty -8 38400 tty5
    1   944   944   944 tty2       944 Ss+      0   0:00 /sbin/getty -8 38400 tty2
    1   945   945   945 tty3       945 Ss+      0   0:00 /sbin/getty -8 38400 tty3
    1   947   947   947 tty6       947 Ss+      0   0:00 /sbin/getty -8 38400 tty6
    1   987   987   987 ?           -1 Ss       0   0:00 /usr/sbin/sshd -D
  987  8560  8560  8560 ?           -1 Ss       0   0:00  \_ sshd: vagrant [priv]
 8560  8637  8560  8560 ?           -1 S     1000   0:00      \_ sshd: vagrant@pts/0
 8637  8638  8638  8638 pts/0    14242 Ss    1000   0:00          \_ -bash
 8638 14242 14242  8638 pts/0    14242 R+    1000   0:00              \_ ps axjf
    1   989   989   989 ?           -1 Ss       0   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.socke
    1   990   990   990 ?           -1 Ss       0   0:00 cron
    1   991   991   991 ?           -1 Ss       1   0:00 atd
    1  1183  1183  1183 ?           -1 Ssl      0   0:01 /usr/bin/ruby /usr/bin/puppet agent
    1  1199  1196  1196 ?           -1 Sl       0   0:12 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-
    1  1322  1319  1319 ?           -1 Sl       0   0:00 ruby /usr/bin/chef-client -d -P /var/run/chef/cli
    1  1349  1349  1349 tty1      1349 Ss+      0   0:00 /sbin/getty -8 38400 tty1
    1  2176  2175  2175 ?           -1 S        0   0:00 upstart-file-bridge --daemon
    1  2179  2178  2178 ?           -1 S        0   0:00 upstart-socket-bridge --daemon
    1  3800  3799  3799 ?           -1 S        0   0:00 upstart-udev-bridge --daemon
    1  3803  3803  3803 ?           -1 Ss       0   0:00 /lib/systemd/systemd-udevd --daemon
```

Some things to note:
names *(e.g., init),*
relationships *(parents, children, grandchildren),*
IDs *(PID, PPID)*

Photos of whiteboard after class


PCB
ID
State
Priority
I/O info
memory pointer
#/len
in-file
Code1
Code2
CPU
mm
The Process
I/O mouse
I/O printer
I/O network
② Data
① Code
③ context
P1
P2
P3
enter
not running
Dispatcher
t/e
running
run for t
Pi
I.
Q
Pn
P3
P2
P1
X
waiting for some event
I/O Q


exit
Ready
Blocked
e1 user input
(keystrokes
en e3 e1 e2 e1
e2 (networ
P P P P
e1 e2 e3
New
init - pid = 1
ppid
1
- pid = 7284
P2
P1


bootloader
boots
init
not Running
New
Ready
Running
UI
Exit
scheduler
sshd
ssh
proc.
App
Waiting
Q_1
Q_2
Q_3