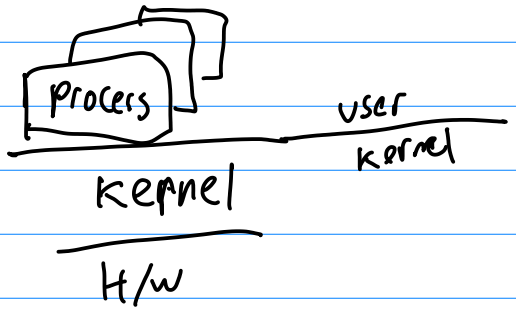
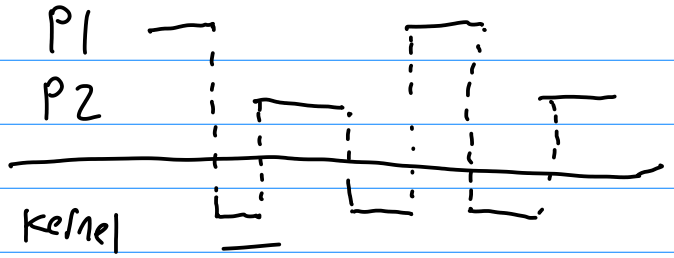
CS 3214 lecture #4 more on processes!

{UTF} - unicode transformation format

Ex 0

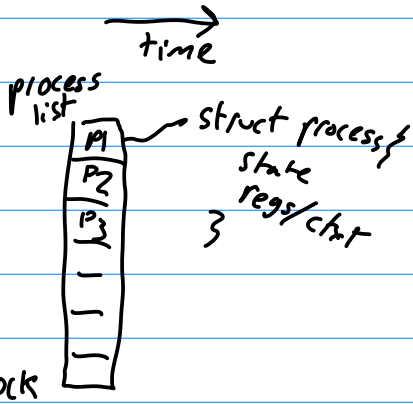
My office hours Zoom T/Th 1-2



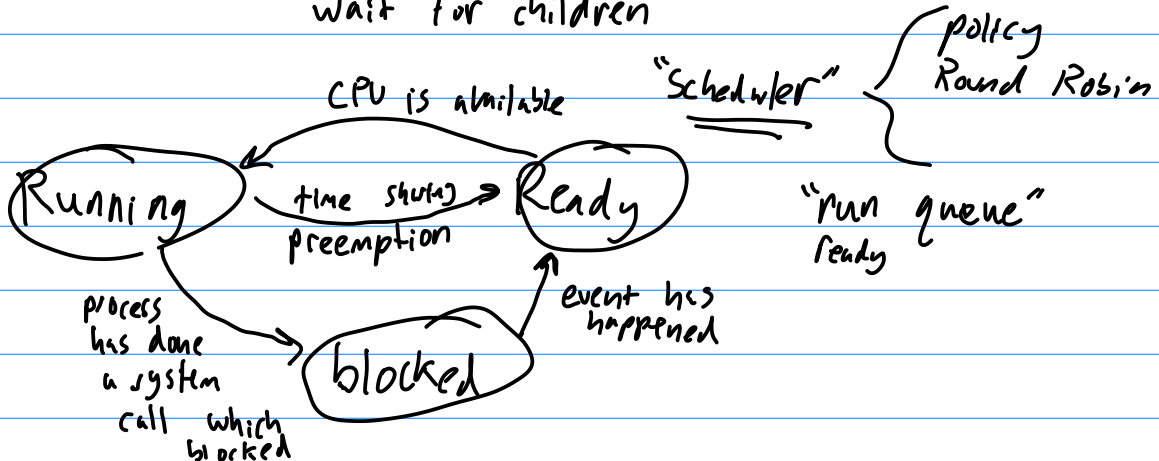


How does the kernel manage processes?

process states



time sharing { Running — on CPU [fetch/decode/execute]
"Ready" ← could be on CPU
Blocked ← user input / I/O devices, net, lock
data from a pipe
cat foo | head | wc
sleep
wait for children

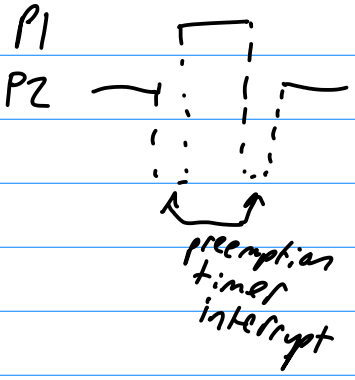


what if:

- nCPU + n ready processes ← all will run on CPU
- nCPU + 0 ready processes ← "idle" process "hlt"
- nCPU + $k \leq n$ ready

- nCPU + 2n ready ← get to run half as often
  my job takes twice as long
- nCPU m >> n ready ← laggy

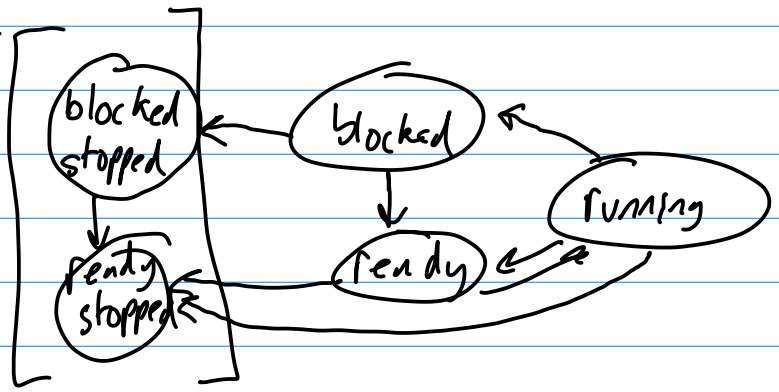
typical laptop: 150-500 blocked
0-2 running



Real States in Linux

B { S - sleep "interruptible"
  { D - uninterruptible sleep
running/ready { R - running/ready "runnable"
  I - "idle kernel thread"
  T - job control stop
  t - debugger
  ~~X~~ ~~X - dead~~
  Z - defunct "zombie" process (terminated but parent has not waited for it)

Ctrl-z



fg/bg

- poll ← wastes CPU cycles
- block

measure CPU load
- load average
- time

Generally prefer blocking*
↳ use the facilities your kernel gives you

Transitions are unpredictable

Next time: process management!
creation / termination / etc.