Review

Questions

- ✦ What are two functions of an OS?
- ✦ What “layer” is above the OS?
- ✦ What “layer” is below the OS?

Questions

- ✦ When is it appropriate for OS to “waste” resources?
- ✦ How might the growth in networks influence OS design?

True or False

- ✦ Unix is a “simple structure” OS
- ✦ Micro Kernels are faster than other OS structures
- ✦ Virtual Machines are faster than other OS structures

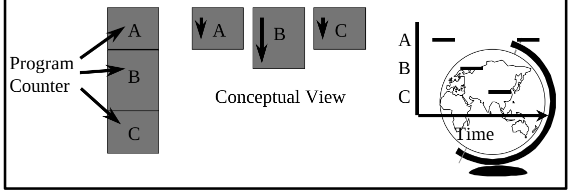
Operating Systems

Processes

(Ch 4.1)

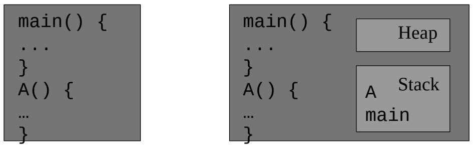
Processes

- ✦ “A program in execution”
- ✦ Modern computers allow several at once
  - – “pseudoparallelism”



Processes

- ✦ "A program in execution"



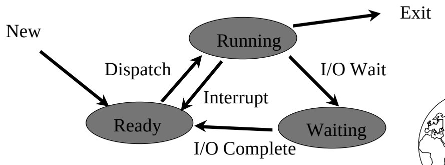
```
main() {
...
}
A() {
…
}
```

```
main() {
...
}
A() {
…
}
```

  - "more" than a program: `ls, tcsh`
  - "less" than a program: `gcc blah.c` (`cpp, cc1, cc2, ln …`)
- ✦ "A sequential stream of execution in it's own address space"

Process States

- ✦ Consider:
  `cat /etc/passwd | grep claypool`



(Hey, you, show states in `top`!)

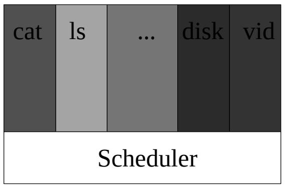
Design Technique: State Machines

- ✦ Process states
- ✦ Move from state to state based on events
  - – *Reactive* system
- ✦ Can be mechanically converted into a program
- ✦ Other example:
  - – string parsing, pre-processor

Unix Process Creation

- ✦ System call: `fork()`
  - – creates (nearly) identical copy of process
  - – return value different for child/parent
- ✦ System call: `exec()`
  - – over-write with new process memory
- ✦ (Hey, you, show demos!)

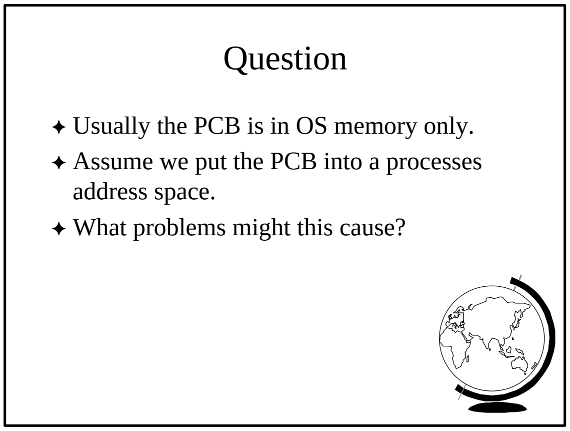
Process Scheduler



- ✦ All services are processes
- ✦ Small scheduler handles interrupts, stopping and starting processes

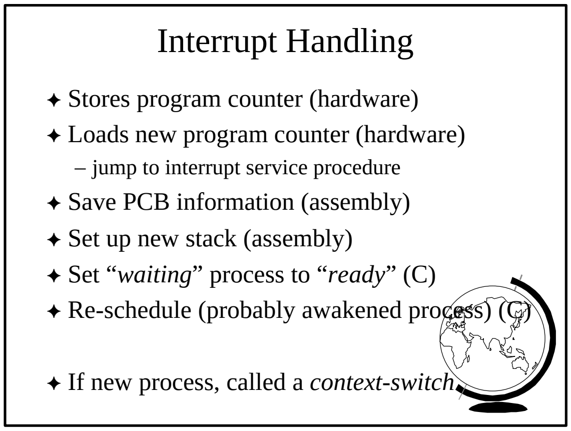
Process Control Block

- ✦ Each process has a PCB
  - – state
  - – program counter
  - – registers
  - – memory management
  - – …
- ✦ OS keeps a table of PCB's, one per process
- ✦ (Hey! Simple Operating System, "`system.h`")



Question

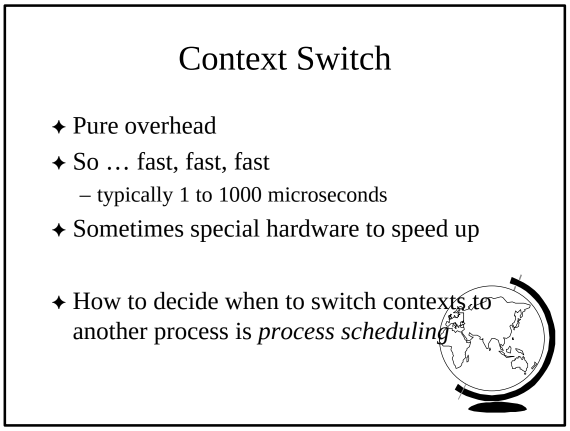
- ✦ Usually the PCB is in OS memory only.
- ✦ Assume we put the PCB into a processes address space.
- ✦ What problems might this cause?



Interrupt Handling

- ✦ Stores program counter (hardware)
- ✦ Loads new program counter (hardware)
  - – jump to interrupt service procedure
- ✦ Save PCB information (assembly)
- ✦ Set up new stack (assembly)
- ✦ Set "*waiting*" process to "*ready*" (C)
- ✦ Re-schedule (probably awakened process) (C)

- ✦ If new process, called a *context-switch*



Context Switch

- ✦ Pure overhead
- ✦ So … fast, fast, fast
  - – typically 1 to 1000 microseconds
- ✦ Sometimes special hardware to speed up

- ✦ How to decide when to switch contexts to another process is *process scheduling*