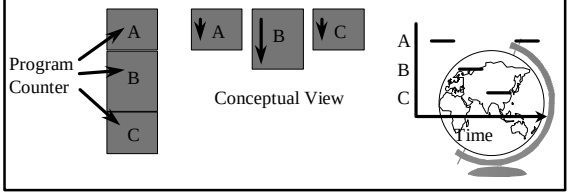
Operating System I

Processes

Processes

- "A program in execution"
- Modern computers allow several at once
  - "pseudoparallelism"



Processes

- "A program in execution"

```
main() {
...
}
A() {
…
}
```

```
main() {
...
}
A() {
…
}
```

Heap

Stack

A

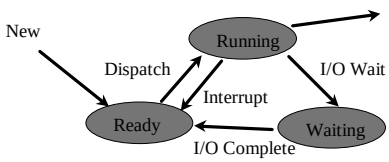
main

  - "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

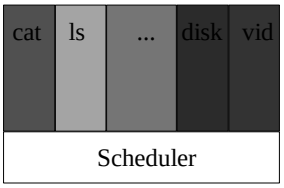
Unix Process Creation

- 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!)

Java Process Creation

- "fork" and "exec" rolled into `exec()`
  `public Process exec(String command)`
  - args separated by whitespace
- Child Process output: status:
  - getOutputStream() - waitFor()
  - getInputStream() - exitValue()
  - getErrorStream() - destroy()
- Depends upon underlying OS proc support

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
- Loads new program counter
  - jump to interrupt service procedure
- Save PCB information
- Set up new stack
- Set "*waiting*" process to "*ready*"
- Re-schedule (probably awakened process)
- If new process, called a *context-switch*

Context Switch

- Pure overhead
- Fast, fast, fast
  - typically 1 to 1000 microseconds
- Sometimes special hardware to speed