Operating Systems

Processes
(Ch 3.1)

Processes

• “A program in execution”
• Modern computers allow several at once
  – “pseudoparallelism”
Processes

• “A program in execution”

```c
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 its 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 address space
- Shell
  - uses fork() and exec()
  - simple!
- (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”)
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)
- Service interrupt (C and assembly)
- Scheduler (C)
  - Newly awakened process
    + Often called a context-switch
  - Previously running process

Context Switch

- Pure overhead
- So … fast, fast, fast
  - typically 1 to 1000 microseconds
- Sometimes special hardware to speed up
- Real-Time wants worse case
  - RT Linux worse case sub 20 microseconds
- How to decide when to switch contexts to another process is process scheduling
Linux Context Switch Times

Measured with LMBench

Processes in Linux

- PCB is in `struct task_struct`
  - states: RUNNING, INTERRUPTIBLE, UNINTERRUPTIBLE
  - priority: when it runs
  - counter: how long it runs
- Environment inherited from parent
- NR_TASKS max, 2048
  - 1/2 is max per user
Processes in Windows

• States: ready, standby (first in line), running, waiting, transition, terminated
• priority - when it runs
• Processes are composed of *threads*
  – (revisit threads after synchronization)