Overview and Processes

9/29/15
Announcement/Reminder: Tutoring options

(Also, we still have an exam instead of class on Friday...)
Role of the operating system

• From Wikipedia: “An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs.”
Role of the operating system

• From Wikipedia: “An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs.”

  – OS is a program
  – gets control from system calls and interrupts
Role of the operating system

• From Wikipedia: “An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs.”

  – program counter
  – RAM and disk space
  – access to network
  – locks
Role of the operating system

- From Wikipedia: “An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs.”

- file system (files, directories)
- I/O buffering
- scheduling
Processes
Process

• State of program being executed
• Contains code + everything needed to resume execution:
  – ????
Process

- State of program being executed
- Contains code + everything needed to resume execution:
  - register values (including program counter)
  - address space (aka memory, including stack)
  - system resources (open files, network connections, etc)
Process states

• Process is always in one of 3 states:
  – running: Process actively running on the processor
  – ready: Could run, but not currently doing so
  – blocked: Not ready to run; waiting for something

(Some explanations include states for process creation or completion)
Which of the following will send a running process P into the ready state?

A. OS scheduler giving the processor to another process
B. P attempting I/O from the user
C. A different user logging into the system (in addition to P’s user)
D. More than one of the above
E. None of the above
Which of the following will send a running process P into the ready state?

A. OS scheduler giving the processor to another process
B. P attempting I/O from the user
C. A different user logging into the system (in addition to P’s user)
D. More than one of the above
E. None of the above
Process creation

- Processes are created
  - At system initialization
  - When one process creates another
    - User asks for one (e.g. via shell)
    - Beginning of batch (scheduled) job
Creating a process in Unix (et al)

• use fork system call to create child process
  – both parent and child continue executing program
  – state is same (copied) except for return value they see from fork

• use call from exec family to turn child into something else

• parent uses wait to block until child finishes
Sample code framework

```c
pid_t childID = fork();
if(childID < 0) { /* fork failed; give error, exit*/ } // I'm the parent
if(childID == 0) { // I'm the child
    execlp(file, arg0, arg1, ..., NULL);
} else { // I'm the parent
    wait(NULL);
}
```
Why have fork and exec to create a process rather than creating one with a new program in one call?

A. Creating a copy (fork w/o exec) is common enough for the API to be designed supporting it
B. There is hardware support for fork that doesn’t apply if a separate program is started
C. To maintain backwards compatibility
D. Processes are somewhat vain and they want to be surrounded by copies of themselves
E. More than one of the above
Why have fork and exec to create a process rather than creating one with a new program in one call?

A. Creating a copy (fork w/o exec) is common enough for the API to be designed supporting it
B. There is hardware support for fork that doesn’t apply if a separate program is started
C. To maintain backwards compatibility
D. Processes are somewhat vain and they want to be surrounded by copies of themselves
E. More than one of the above
Ending a process

• Process terminates itself with exit()
  (sometimes implicit in main returning)

• Process terminated by uncaught signal
  – error: SIGSEGV (seg fault), SIGBUS (bus error)
  – Control-Z: SIGTSTP
  – killing a process: kill -9 pid
  – Signals also used to communicate with the process
Batch scheduling

- Jobs arriving in system over time, waiting for access to processor

- Each job has
  - release time: when system learns about it and earliest it can run
  - processing time: how long job must run to complete

(Sometimes also deadlines, weights, etc)
Common algorithms

• First-Come First-Served (FCFS)
  – When processor is idle, start earliest-arriving job

• Shortest Processing Time (SPT)
  – When processor is idle, start shortest job

• Shortest Remaining Processing Time (SRPT)
  – Always switch jobs to run job with least remaining processing time
Metrics for schedule quality

- Main one: total response time
  - Response for job $j = \text{completion}_j - \text{arrival}_j$
    - Captures time user spent waiting
    - Sum this over all jobs

- Alternatives:
  - Sum of completion times
  - Max response time
  - Max completion time
  - Max or Sum Stretch: divide resp. time by processing time
What is the total response time of the jobs below if scheduled by FCFS?

A. 12
B. 16
C. 20
D. 24
E. 28

<table>
<thead>
<tr>
<th>Job</th>
<th>arrival</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
What is the total response time of the jobs below if scheduled by FCFS?

A. 12  
B. 16  
C. 20  
D. 24  
E. 28

<table>
<thead>
<tr>
<th>Job</th>
<th>arrival</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>