Department of Physics,
Computer Science & Engineering

CPSC 410 – Operating Systems I

Virtualization: The CPU

Keith Perkins

Adapted from "CS 537 Introduction to Operating Systems"
Arpaci-Dusseau

Process Creation

Two ways to create a process

- Build a new empty process from scratch
- Copy an existing process and change it appropriately

Option 1: New process from scratch

- Steps
  - Load specified code and data into memory;
    Create empty call stack
  - Create and initialize PCB (make look like context-switch)
  - Put process on ready list
- Advantages: No wasted work
- Disadvantages: Difficult to setup process correctly and to express all possible options
  - Process permissions, where to write I/O, environment variables
  - Example: WindowsNT has call with 10 arguments

Process Creation

Option 2: Clone existing process and change

- Example: Unix fork() and exec()
  - Fork(): Clones calling process
  - Exec(char *file): Overlays file image on calling process
- Fork()
  - Stop current process and save its state
  - Make copy of code, data, stack, and PCB
  - Add new PCB to ready list
  - Any changes needed to child process?
- Exec(char *file)
  - Replace current data and code segments with those in specified file
- Advantages: Flexible, clean, simple
- Disadvantages: Wasteful to perform copy and then overwrite of memory

Unix Process Creation

How are Unix shells implemented?

```
While (1) {
    Char *cmd = getcmd();
    Int retval = fork();
    If (retval == 0) {
            // This is the child process
            // Setup the child’s process environment here
            // E.g., where is standard I/O, how to handle signals?
            exec(cmd);
            // exec does not return if it succeeds
            printf(“ERROR: Could not execute %s\n”, cmd);
            exit(1);
    } else {
            // This is the parent process; Wait for child to finish
            int pid = retval;
            wait(pid);
    }
}
```