Linux/bash process control

- Crucial to be able to monitor and control processes and their resource use
- Relevant process info includes user, cpu time, memory use, priority, start time, % of cpu dedicated to the process, etc
- Every running process has an integer id associated with it, many of the relevant commands provide or make use of ids
- Parent processes can create child processes

Foreground and background

- Processes can be active or paused, and can be running in the foreground or background
- In the command shell, when you type a command and must wait for it to complete it is running in the foreground
- If you run it in the background instead, while it runs you can still run other commands in the foreground (put & at the end of the command to run in background)

Controlling processes

- ^C (control-c) kills the current foreground process
- ^Z (control-z) pauses the current foreground process
- The command bg resumes a paused process, but in the background
- The command fg moves a background process to the foreground
- The command ps lists your active processes, showing the owning username, the process id, the % of cpu being used, the % of total memory being used, the start time, the cumulative cpu time used, etc

Signals to processes

- You can send signals to processes using the kill command if you know the process id, e.g. “kill -9 213” sends the signal 9 (which terminates a process) to process id 213
- The kill command can send many other signal types
- If you have a background process you’re trying to terminate, use ps to lookup its process id, then the kill -9 to terminate it

Process priority

- Process priorities range from 0 to 19, and determines how much cpu time a process gets in competition with the other running processes (priority 0 is “most important”, gets most cpu time, 19 is least important)
- You can deliberately run a process at lower priority using the “nice” command, e.g. “nice g++ mybigprog.cpp”

Monitoring processes

- “ps” lists all your active processes
- “top” displays a constantly-updating list of everyone’s active processes, sorted by which ones are using the most cpu right now
- “w” gives similar information, but as a single snapshot
- “time *yourcommand*” runs your command then tells you how much cpu time, system time, and real time it took to complete (e.g. 1:32 0:10 4:25 would mean it took 4 minutes, 25 seconds of real time, of which it had control of the cpu for 1:32, and of that it spent 10 seconds running system commands)