CSCI 2132
Software Development

Lecture 11:

Processes and Job Control

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Previous Lecture

- More about scanf
- Fractions program example
- Shells and Computing Environment
- Shells
  - functionality
  - popular shells
- Bash shell
  - commands
  - variables
- Processes and programs
  - process memory composition

Thread of Execution

- is a sequential execution of a program by processor (CPU)
- Contains CPU position in code
  - but also registers, and some other information when not executing
- Traditionally, one process had one thread
- Modern OS: one process can have multiple threads
- Process: heavier-weight object including resources information
- There could be more than one process (and thread) with the same code in memory

Process Control Block (PCB)

- Created by system when a process has started
- PCB includes:
  - Process Identification: PID (process identifier, unique nonnegative integer)
  - The present position in the thread of execution
  - Resources allocated to the process (e.g., memory, open files)
  - Process ownership (user and group)
  - Process state (running, sleeping, pre-empted, created, zombie)

Process Creation

- Processes are created by other processes
- All processes descend from the process `init` (PID=1)
- A user typically creates new processes using shell
- Steps how a parent process creates a child process:
  - Operation fork (system call): same code base
  - Separating child from parent: fork return code
  - Operation exec (system call)
- **Job Control** is a name for shell functionality for managing processes

Job Control

- Shell facility for:
  - Starting processes in the background
  - Changing processes between background and foreground mode
  - Suspending and resuming processes
  - Terminating processes
  - Displaying a list of current processes

Foreground and Background Processes

- Foreground process: controls the terminal
- Background process: cannot read from keyboard but can print to terminal
- To create a process as a background process: use `&`
- Actually `&` makes a whole pipeline to run in background
- Shell refers to a pipeline as a job
- If a command generates a lot of output and errors, we may want to redirect them; for example:

  ```
  find / -name gcc > result 2> error &
  ```
- Another way to send process to background:
  Ctrl-z and `bg`

Job and Process Control

- Print jobs: `jobs`
- Print processes: `ps`
- Running job in backgroud: use ‘`&`’
- Running job in background, another way:
  - run, Ctrl-z, `bg` or `bg %jobID`
- Bringing job to foreground: `fg` or `fg %jobID`
- Terminating a job or process:
  `kill`, `kill %jobID`, or `kill PID`