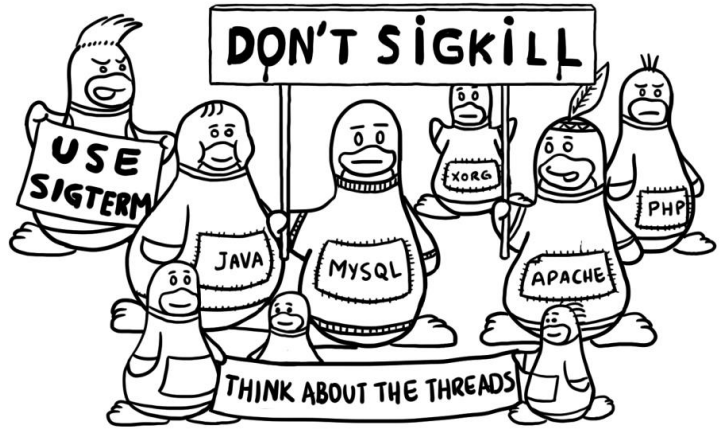
Western

UNIVERSITY · CANADA

Processes and Job Control

Winter 2022



Foreground and Background

- Unix is a multi-user AND a multi-task operating system
- Unix provides you the ability to work with multiple processes in the same session

Foreground and Background

- When you run a process in your session (e.g. ls or vi), the process is running in the foreground
  - It accepts control from your input (keyboard) and controls your output (screen)

Foreground and Background

- However, it is possible to run a process in the background
  - WARNING: If your background process expects input, it will pause until it is "promoted" to the foreground so it can accept input

Foreground and Background

- WARNING: If your background process has output, it will not wait. It will output to the screen (potentially "poisoning" any existing foreground output)
  - (This is by default. There are ways to get around this issue which we will cover later)

Foreground and Background

- We already know how to run a program in the foreground. We do it all the time
  - E.g. <command> <options>
- To force a program to run in the background when we run it, we use the & symbol to make it a background process
  - E.g. <command> <options> &

Managing processes

- When you run a process in the background, the shell will give you:
  - The job number (this is specific to the user)
  - The process number (or the process id) (this is a global number that all users can see)

```
[wbeldman@compute ~]$ cat /dev/zero &
[1] 673979
```

Managing processes

- To see your list of jobs, run the jobs command
- To see your list of processes run the ps command

```
[wbeldman@compute ~]$ jobs
[1]+  Running                 cat /dev/zero &
[wbeldman@compute ~]$ ps
    PID TTY          TIME CMD
 673947 pts/11   00:00:00 bash
 673979 pts/11   00:00:07 cat
 674038 pts/11   00:00:00 ps
```

Managing processes

- Use jobs -l to view job ids and process ids together

```
[wbeldman@compute ~]$ jobs -l
[1]  674243 Running                 cat /dev/zero &
```

Managing processes

- Let's run the command a few times to get a couple of simultaneous jobs

```
[wbeldman@compute ~]$ cat /dev/zero &
[2] 674114
[wbeldman@compute ~]$ cat /dev/zero &
[3] 674115
[wbeldman@compute ~]$ cat /dev/zero &
[4] 674116
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]   Running                 cat /dev/zero &
[3]-  Running                 cat /dev/zero &
[4]+  Running                 cat /dev/zero &
```

Managing processes

- The job ids are counting up one-by-one because these ids belong to the user only
- The processes ids usually do not count up one-by-one because process ids are a global number and other users are getting process ids too

Managing processes

- A user can have one foreground process and multiple background processes at once
  - When using a graphical user interface (e.g. XWindows) instead of SSH, it is possible to have multiple foreground processes
- It is possible to control which jobs are in the foreground/background with the fg/bg commands

Managing processes

- fg <N> - Promote job id N to the foreground
  - The most recent jobs are also labelled as + and - so it is possible to use + or - as a shorthand instead of using a numerical value

Managing processes

- We've seen CTRL+C to "kill" the foreground process
- Use CTRL+Z to simply pause the foreground process and give control back to the shell (also known as "suspending")

```
[wbeldman@compute ~]$ fg 2
cat /dev/zero
^Z
[2]+  Stopped                 cat /dev/zero
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]+  Stopped                 cat /dev/zero
[3]   Running                 cat /dev/zero &
[4]-  Running                 cat /dev/zero &
```

Managing processes

- bg <N> - run job id N in the background
  - If a job is in a stopped state, you can use bg to allow it to resume (as if you ran it with & in the first place)

```
[wbeldman@compute ~]$ bg 2
[2]+ cat /dev/zero &
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]   Running                 cat /dev/zero &
[3]-  Running                 cat /dev/zero &
[4]+  Running                 cat /dev/zero &
```

Managing processes

- To kill a running job, use kill %N where N is your job number

```
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]   Running                 cat /dev/zero &
[3]   Running                 cat /dev/zero &
[4]-  Running                 cat /dev/zero &
[5]+  Running                 cat /dev/zero &
[wbeldman@compute ~]$ kill %5
[5]+  Terminated              cat /dev/zero
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]   Running                 cat /dev/zero &
[3]-  Running                 cat /dev/zero &
[4]+  Running                 cat /dev/zero &
[wbeldman@compute ~]$ _
```

Managing processes

- To kill all your running jobs at once, use disown -a

```
[wbeldman@compute ~]$ jobs
[1]   Running                 cat /dev/zero &
[2]   Running                 cat /dev/zero &
[3]-  Running                 cat /dev/zero &
[4]+  Running                 cat /dev/zero &
[wbeldman@compute ~]$ disown -a
[wbeldman@compute ~]$ jobs
[wbeldman@compute ~]$ _
```

Managing processes

- The kill command sends a signal to a process
  - kill -l – list all the signals
    - SIGTERM – ask the process to terminate itself gracefully if possible
    - SIGKILL – terminate the process

- 请我了断吧.
launcer!

Managing processes

- SIGTERM is the default
  - kill <process id>
- To force the SIGKILL
  - kill -9 <process id>

Maximum values

- The ulimit command tells you your limits (including process counts)
  - ulimit -a – print out your maximum limits
- You can temporarily set a lower limit
  - E.g. ulimit -u 4 – Sets the maximum number of processes to 4

Western

UNIVERSITY · CANADA