Background Processes

Introduction

- Up until now, process execution assumed foreground tasks
  - i.e. the shell fully waits until the command finishes executing
- Background tasks are when the shell continues executing while the command executes
- In terms of syntax, the difference is the inclusion of the '&' at the end of the command
  - *ls*
  - *ls &*

Waiting?

- Before you used waitpid to signal to the shell to block
  - Blocking is when a process temporarily stops executing to prevent consuming resources
- This is fine for foreground processes because the shell isn't doing anything anyway
- But you can't use wait the same way for background processes
  - But you still need to wait to capture when the command finishes
  - Otherwise your child process will become a zombie

WNOHANG

- *waitpid(pid, &status, WNOHANG);*
  - Returns pid of changed process, 0 if still running
- This causes waitpid to return immediately
- But it captures whether the process (pid) has finished or not
  - It only tells you when you check
  - This means you will have to check repeatedly

Exit

- What happens if you have background processes and exit is issued?
- You will need to fully wait on each of them before quiting
  - Otherwise they will become orphans
- This requires keeping a counter or array of executing processes

Project Specifics

- Keep a queue of running background processes
  - Position in queue (queue number)
  - Process ID (pid)
  - Command (cmd)
- When process starts execution, print out
  - *[queue number] [pid]*
  - [0] [5532]
- When process finishes execution, print out
  - *[queue number]+ [cmd]*
  - [0]+ [ls -la]