Fork control flow

- fork()
  - returns child's PID
  - getpid() returns child PID
  - getpid() returns parent's PID

- fork() returns 0

- hello world!
  - fork()
  - returns 0

- hello from parent!
  - return 0

- hello from child!
  - return 0
Scheduling

Managed by OS scheduler

CPU

\[ \text{running set} \]

\[ \text{ready queue} \]

\[ \text{blocked set} \]

Important:
- Ready queue is not a simple FIFO queue
- Don’t make assumptions about how long you get the CPU for

Print letters output: abcdd...?
Busy waiting = keeping yourself busy waiting for something to happen

waitpid(), wait for child to exit
- If child already exited, returns immediately
- Only for children
- Once you call it on a process, can't call it again

fork() =>
Like malloc: creating processes

Like malloc: creating processes

program is "running"

Ready => Running => Terminated

Zombie process

"reap" zombie by calling waitpid

Like free()
Job control

- Send SIGSTOP signal to pause a process
- Send SIGCONT signal to resume a process

On command line:

```
kill -SIGSTOP pid
```

Programmatically:

```
kill (pid, SIGSTOP/CONT)
```