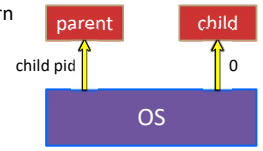
Creating New Processes

- `pid_t fork();`
  - Creates a new process (the "child") that is an *exact clone** of the current process (the "parent")
    - *Everything is cloned except threads. Sockets, file descriptors, virtual address space, variables, etc.
  - The new process has a separate virtual address space from the parent

`fork()`

- `fork()` has peculiar semantics
  - The parent invokes `fork()`
  - The OS clones the parent
  - *Both* the parent and the child return from fork
    - Parent receives child's pid
    - Child receives a 0



- See `fork_example.cc`

CSE 351 review:
When a child process exits, it is a zombie until its parent reaps it.

Poll Everywhere

pollev.com/cse33320su

- What will happen when one of the grandchildren processes finishes?

A. Zombie until grandparent exits
B. Zombie until grandparent reaps
C. Zombie until init reaps
D. ZOMBIE FOREVER!!!
E. We're lost…

Why Concurrent Processes?

- Advantages:
  - No shared memory between processes
  - No need for language support; OS provides "fork"
  - Concurrent execution leads to better CPU, network utilization
- Disadvantages:
  - Processes are heavyweight
    - Relatively slow to fork
    - Context switching latency is high
  - Communication between processes is complicated