Intro to IPC, and Signals

44-550: Operating Systems

Processes

- Processes (even forked processes) do NOT share memory with each other
  - Threads within a process do
- Threading has limitations
  - Limited to one computer's worth of hardware
  - Ryzen Threadripper 3990X: 64 cores, $3.5k
- Communication between processes enables processes to share state and data
  - Removes the limitation on the maximum number of resources

Interprocess Communication (IPC)

- A mechanism by which different processes communicate data (and other information)
  - Signal
    - Mostly used for processes to control other processes, not send data
  - Files (memory-mapped or not...)
    - Record stored on disk (or in RAM if memory-mapped) while accessed by multiple processes
  - Socket
    - Data stream sent over a network interface (could be locally)
  - Pipe
    - One way data stream between two processes, sends information over stdin and stdout
    - Named pipes go through files
  - Message passing (used by MPI)
  - Shared Memory (explicitly shared)
  - Semaphores (not supported by sem.h)

Signals

- Used as control mechanisms, usually
- Interrupt the process to do some task
- You've used these before
  - ctrl + c sends `SIGINT` to the running process
  - kill sends whatever signal you specify to a given PID
    - defaults to `SIGTERM`
    - `kill -9` sends `SIGKILL` (which is signal 9)
  - `SIGSEGV`: enough said
- For a full list of what signals are available, you can check out `man signal`
- You can also have your program handle signals specially (both sent and recieved)
  - This includes overriding the default behavior of signals

Catching and Handling Signals

- `#include <signal.h>`

Intercepting Signals

`signal(int signum, void (*handler) (int));`

Function pointer: takes an int (the signal the process got) and returns nothing.

Example:

```
void sigterm_handler(int sig)
{
    // do thing
}
// in main
signal(SIGTERM, sigterm_handler);
```

Example: Catching SIGTERM

- `examples/ipc/magicword.c`

Sending Signals

- `#include <signal.h>`

Sending Signals

```
kill(pid_t pid, int signum);
```

Example:

```
int cpid = fork();
// send SIGINT to the child in the parent
kill(cpid, SIGINT);
```

Example: Baseball!

- `examples/ipc/baseball.c`

Practice

Modify the `baseball.c` file to, instead of playing baseball, playing ping pong (send signals back and forth between the parent and child process)

- `examples/ipc/pingpong.c`