CIS 5480
Recitation 3

Thursday, February 20, 2025

Penn Engineering
University *of* Pennsylvania

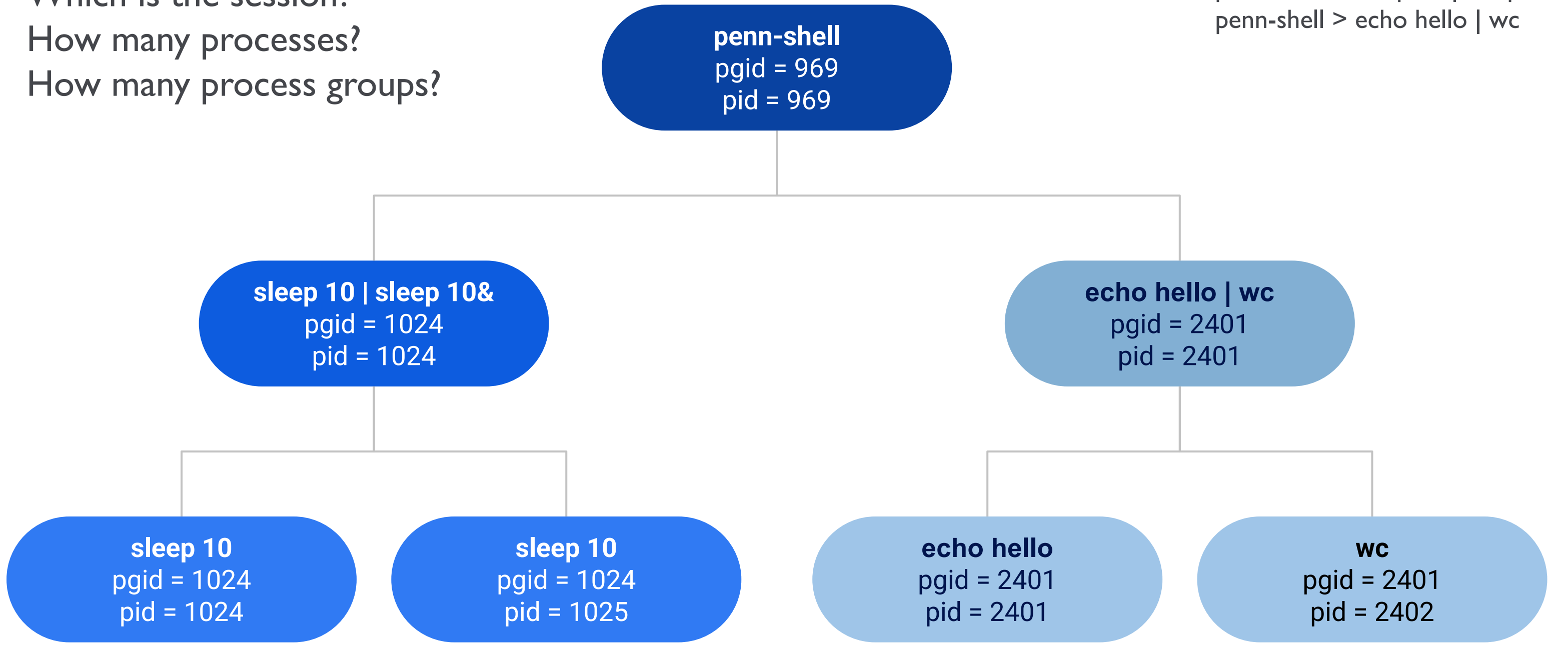
Agenda

- Process Groups
  - waitpid()
  - Signal handling
- Terminal Control

Process Groups

1. Which is the session?
2. How many processes?
3. How many process groups?

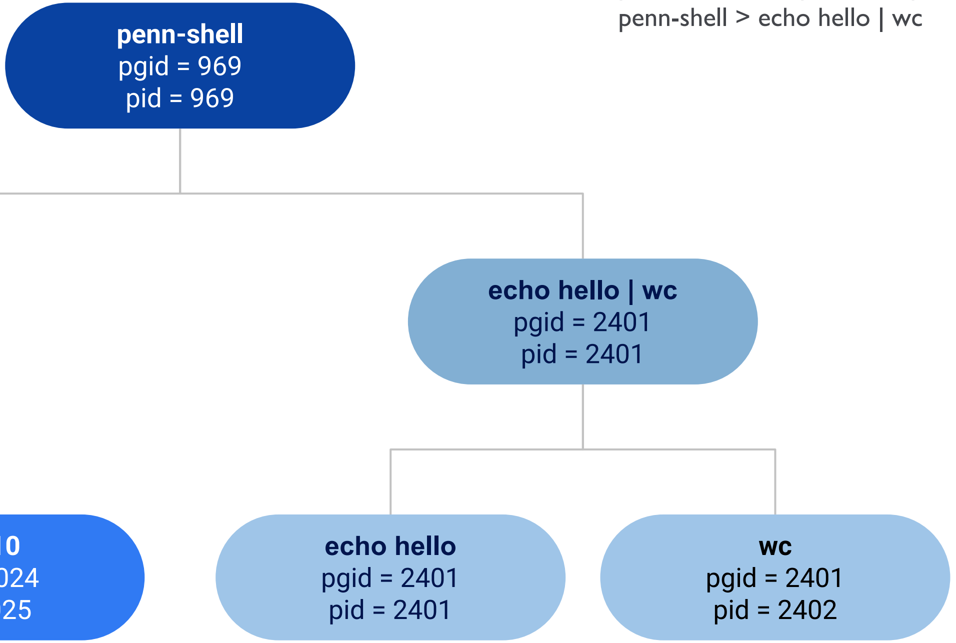
```
penn-shell > sleep 10 | sleep 10 &
penn-shell > echo hello | wc
```



Process Groups

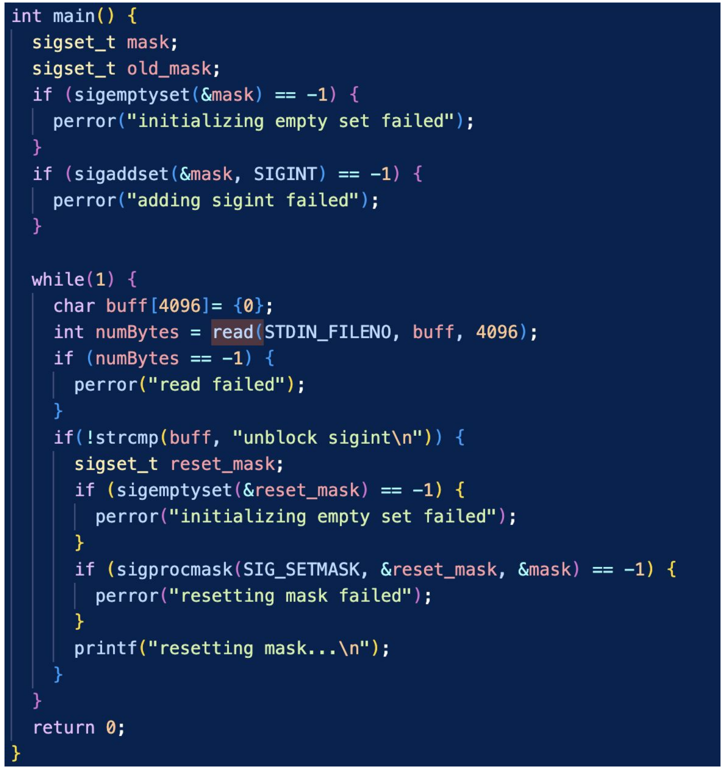
1. Which is the session? **penn-shell**
2. How many processes? **7**
3. How many process groups? **3**

```
penn-shell > sleep 10 | sleep 10 &
penn-shell > echo hello | wc
```



penn-shell pgid = 969 pid = 969
sleep 10 | sleep 10& pgid = 1024 pid = 1024
echo hello | wc pgid = 2401 pid = 2401
sleep 10 pgid = 1024 pid = 1024
sleep 10 pgid = 1024 pid = 1025
echo hello pgid = 2401 pid = 2401
wc pgid = 2401 pid = 2402

Signal Handling



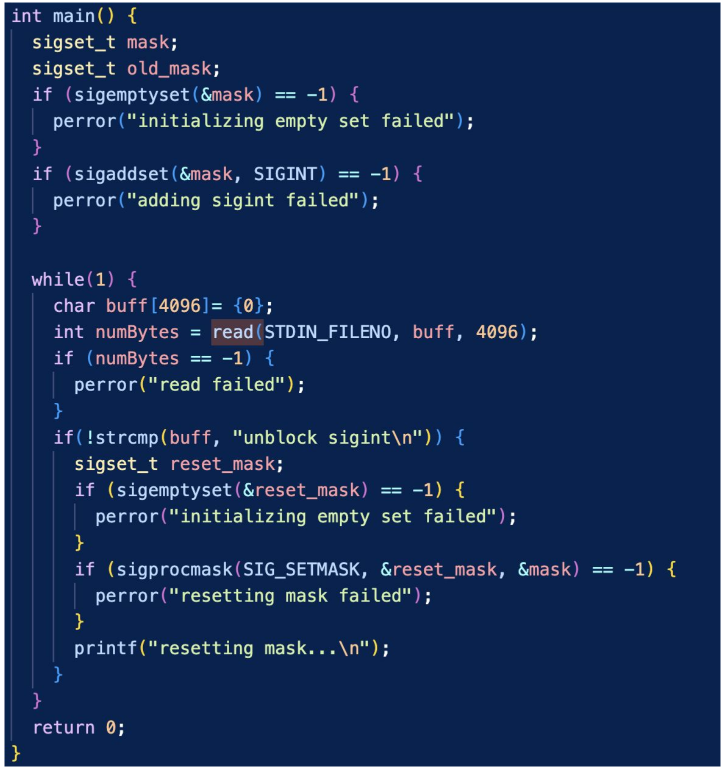
```c
int main() {
  sigset_t mask;
  sigset_t old_mask;
  if (sigemptyset(&mask) == -1) {
    perror("initializing empty set failed");
  }
  if (sigaddset(&mask, SIGINT) == -1) {
    perror("adding sigint failed");
  }

  while(1) {
    char buff[4096]= {0};
    int numBytes = read(STDIN_FILENO, buff, 4096);
    if (numBytes == -1) {
      perror("read failed");
    }
    if(!strcmp(buff, "unblock sigint\n")) {
      sigset_t reset_mask;
      if (sigemptyset(&reset_mask) == -1) {
        perror("initializing empty set failed");
      }
      if (sigprocmask(SIG_SETMASK, &reset_mask, &mask) == -1) {
        perror("resetting mask failed");
      }
      printf("resetting mask...\n");
    }
  }
  return 0;
}
```

What does this code do?

Assume you have all the proper #include statements

Penn Engineering

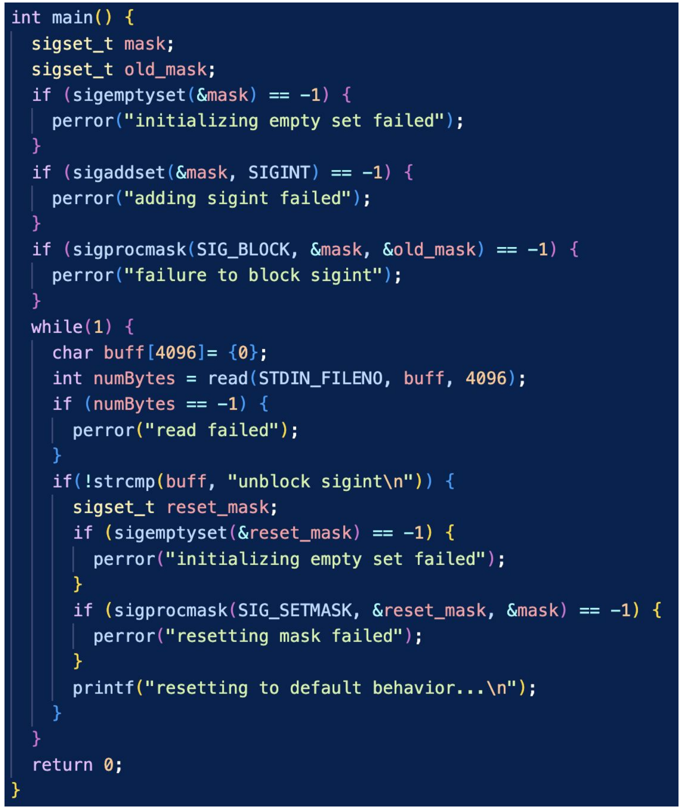
Signal Handling



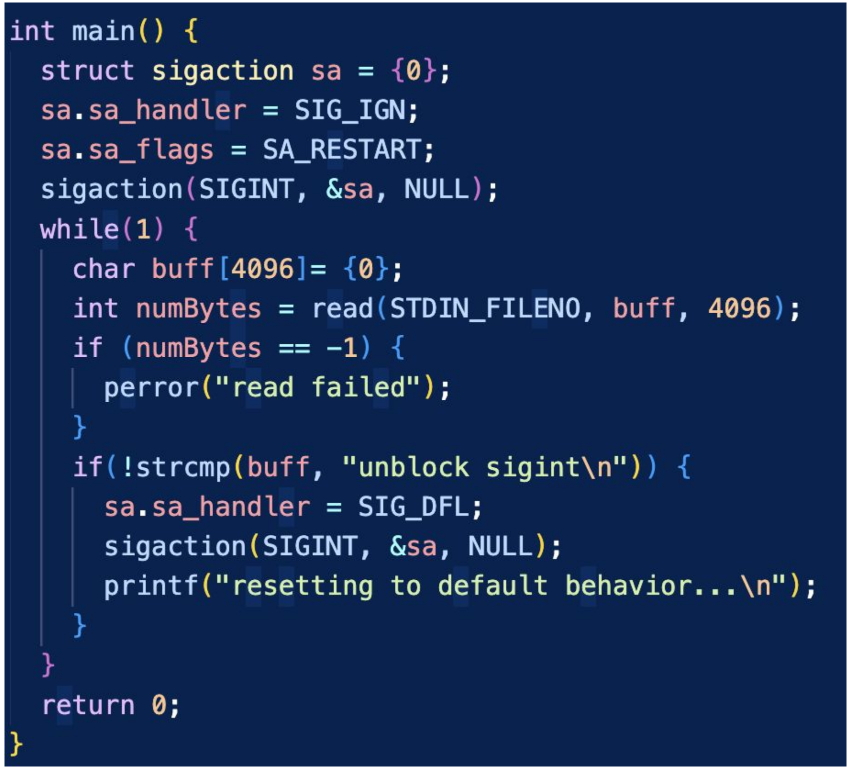
```c
int main() {
  sigset_t mask;
  sigset_t old_mask;
  if (sigemptyset(&mask) == -1) {
    perror("initializing empty set failed");
  }
  if (sigaddset(&mask, SIGINT) == -1) {
    perror("adding sigint failed");
  }

  while(1) {
    char buff[4096]= {0};
    int numBytes = read(STDIN_FILENO, buff, 4096);
    if (numBytes == -1) {
      perror("read failed");
    }
    if(!strcmp(buff, "unblock sigint\n")) {
      sigset_t reset_mask;
      if (sigemptyset(&reset_mask) == -1) {
        perror("initializing empty set failed");
      }
      if (sigprocmask(SIG_SETMASK, &reset_mask, &mask) == -1) {
        perror("resetting mask failed");
      }
      printf("resetting mask...\n");
    }
  }
  return 0;
}
```

How would you fix the code?

What’s the difference between these two?



```c
int main() {
  sigset_t mask;
  sigset_t old_mask;
  if (sigemptyset(&mask) == -1) {
    perror("initializing empty set failed");
  }
  if (sigaddset(&mask, SIGINT) == -1) {
    perror("adding sigint failed");
  }
  if (sigprocmask(SIG_BLOCK, &mask, &old_mask) == -1) {
    perror("failure to block sigint");
  }
  while(1) {
    char buff[4096]= {0};
    int numBytes = read(STDIN_FILENO, buff, 4096);
    if (numBytes == -1) {
      perror("read failed");
    }
    if(!strcmp(buff, "unblock sigint\n")) {
      sigset_t reset_mask;
      if (sigemptyset(&reset_mask) == -1) {
        perror("initializing empty set failed");
      }
      if (sigprocmask(SIG_SETMASK, &reset_mask, &mask) == -1) {
        perror("resetting mask failed");
      }
      printf("resetting to default behavior...\n");
    }
  }
  return 0;
}
```



```c
int main() {
  struct sigaction sa = {0};
  sa.sa_handler = SIG_IGN;
  sa.sa_flags = SA_RESTART;
  sigaction(SIGINT, &sa, NULL);
  while(1) {
    char buff[4096]= {0};
    int numBytes = read(STDIN_FILENO, buff, 4096);
    if (numBytes == -1) {
      perror("read failed");
    }
    if(!strcmp(buff, "unblock sigint\n")) {
      sa.sa_handler = SIG_DFL;
      sigaction(SIGINT, &sa, NULL);
      printf("resetting to default behavior...\n");
    }
  }
  return 0;
}
```

Now: we mix signals and process groups

True or False?

- Only foreground processes can read from the terminal
- Only foreground processes can write to the terminal
- Multiple processes can exist in the foreground
- Multiple process groups can exist in the foreground
- Multiple process groups can exist in the background
- Signals sent from the terminal is sent to all processes
- Signals can only be sent to the foreground job
- Signal blocking behavior is always different from signal ignoring behavior

Situation #1

A parent creates its own signal handler to ignore SIGINT. It forks a child. You hit Ctrl+C. Which process terminates?

A. The parent only
B. The child only
C. Both parent and child
D. Neither parent nor child

Situation #2

A parent creates its own signal handler to ignore SIGINT. It forks a child. The child execs the command "sleep 100" You hit Ctrl+C. Which process terminates?

A. The parent only
B. The child only
C. Both parent and child
D. Neither parent nor child

Situation #3

A parent creates its own signal handler to ignore SIGINT. It forks a child. The child execs the command "sleep 100 &" You hit Ctrl+C. Which process terminates?

A. The parent only
B. The child only
C. Both parent and child
D. Neither parent nor child

Situation #4

A parent with no signal handlers forks a child. Both processes run an infinite loop. You hit Ctrl+C. Which process terminates?

A. The parent only
B. The child only
C. Both parent and child
D. Neither parent nor child

Terminal Control

- Only one process group should have terminal control at a time
- Only the controlling group can read from and write to terminal, and receive terminal signals (`SIGINT` from Ctrl-C & `SIGTSTP` from Ctrl-Z)
  - foreground job in penn-shell

Terminal Control

- Which process group has terminal control at this point?
  - `pid_t tcgetpgrp(int fd);`
- What happens when another group (e.g., in the background) tries to access the terminal?
  - OS sends a `SIGTTIN` signal
  - Default action: stop the program

Terminal Control

- How can another process group take over terminal control?
  - `int tcsetpgrp(int fd, pid_t pgrp);`
- What if we need to write to stdout from the background?
  - Call `tcsetpgrp()` and receive a `SIGTTOU` signal (default: stop the program)
  - We can configure it to block or ignore this signal

./tc_demo

- Questions? :)

Penn Engineering