Recitation 8

Scheduling, sched-demo, and skeletons

Agenda

- Ucontext Review
- Sched-demo.c
- Scheduling
- Kernel Stuff

Ucontext Review

- User level threads
- Each has its own separate stack and execution context
- Very primitive and easy to segfault
- Man page https://man7.org/linux/man-pages/man3/getcontext.3.html

Sched-demo.c

- Schedules 4 threads total, one running “cat” and 3 which just increment a value
- Uses a timer and a signal handler to switch between different contexts
- Need to use VALGRIND_STACK_REGISTER to avoid valgrind errors when switching between stacks
- Ctrl-D to exit from the program
- Can compile with just **clang sched-demo.c**

Sched-demo.c and helloContext.c

- Live Demo

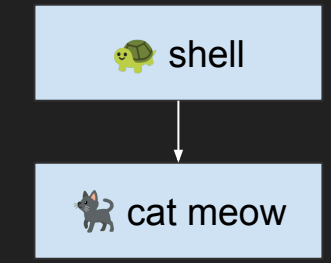
Scheduling

- Every clock tick, switch to scheduler
- Processes have priority 1, 0, -1, with -1 scheduled the most
- -1 is scheduled 1.5x 0
- 0 is scheduled 1.5x 1
- Processes are 0 by default, but shell is -1
- Still schedule a process even if

For Milestone 1

- Should be able to schedule multiple processes
- Already have what a process is
- Be able to switch between processes
- There should be priority enforcement

Kernel stuff 🌽
+ skeleton code 💀



User Programs

```
void shell() {
  p_spawn(cat, {"meow"}, ...);
  p_waitpid(pid, ...);
}

// shell built-in
void cat() {
  char buf[];
  p_read(buf);
  p_write(buf);
}
```

User Level Functions

```
void p_spawn() {
  k_process_create();
}

void p_read() {
  if (stdin) {
    read();
  } else {
    f_read();
  }
}
```

Kernel Level Functions

```
void k_process_create() {
  makecontext();
  // modify PCB
}

void f_read() {
  // read relevant file
  // from PennFAT
  // binary file
}
```

extern + header guards

- Global state across files
- #include is preprocessor that literally includes content into source code
- Header guards → prevent including code multiple times in same file

global_state.h

```c
#ifndef GLOBAL_STATE_H
#define GLOBAL_STATE_H

typedef struct GlobalState {
  int id;
} GlobalState;

extern GlobalState gs;

#endif // GLOBAL_STATE_H
```

main.c

```c
#include "global_state.h"
#include "helper.h"

GlobalState gs;

int main() {
  gs.id = 0;
  ...
}
```

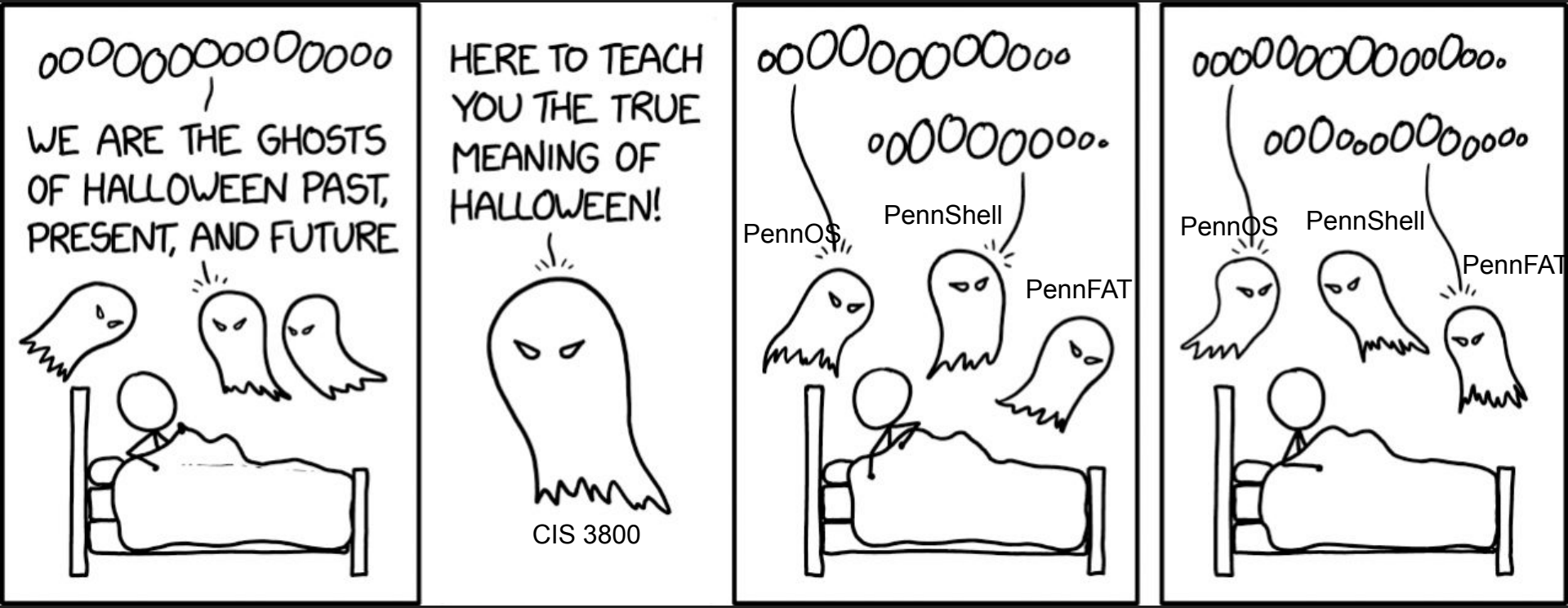
helper.c

```c
#include "global_state.h"

void helper_func() {
  gs.id++;
  printf("%d\n", gs.id);
}
```

<stdarg.h>

```c
#include <stdio.h>
#include <stdarg.h>

void foo(char *fmt, ...) {
  va_list ap;
  va_start(ap, fmt);
  while (*fmt) {
    switch (*fmt++) {
      case 's':   // string
        printf("%s ", va_arg(ap, char*));
        break;
      case 'd':   // int
        printf("%d ", va_arg(ap, int));
        break;
      case 'c':   // char
        // need a cast bc va_arg only takes fully promoted types
        printf("%c ", (char) va_arg(ap, int));
        break;
    }
  }
  va_end(ap);
}
```

```c
foo("dsc", 4871, "memory leaks", '!');
// Prints “4871 memory leaks !”
```


OOOOOOOOOOOOOO
WE ARE THE GHOSTS OF HALLOWEEN PAST, PRESENT, AND FUTURE
HERE TO TEACH YOU THE TRUE MEANING OF HALLOWEEN!
CIS 3800
OOOOOOOOOO
OOOOOOOO.
PennOS
PennShell
PennFAT
OOOOOOOOOOOO.
OOOoOOOOOOO
PennOS
PennShell
PennFAT