CIS 5480 *penn-shell* Mandatory Additional Credit

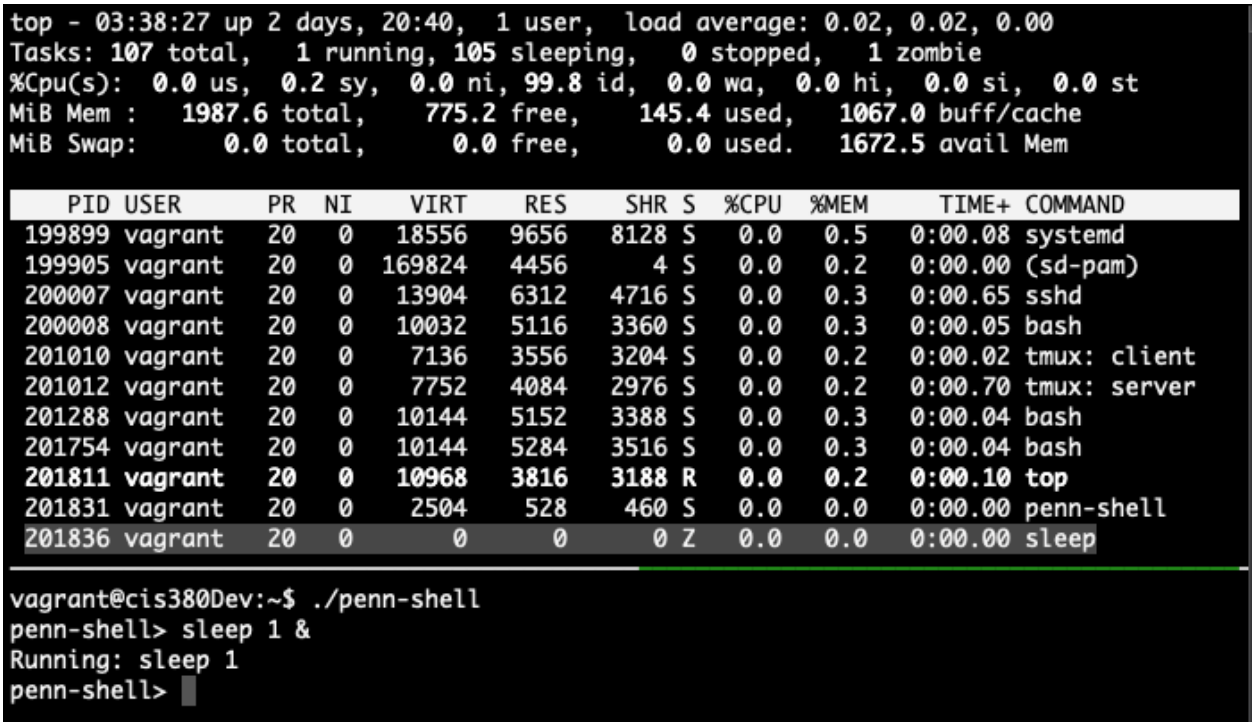
Spring 2023

1 Reap background zombies asynchronously (10 points)

Don't attempt this unless regular credit can clear the autograder. You must submit exactly the same compressed tar file for both regular/mandatory additional credits. The autograder will add the `--async` option when grading the additional credit. The due date is the same as the regular credit for Penn Shell.

The `SIGCHLD` signal is ignored by default. In the regular credit, *penn-shell* polls the background jobs and reaps zombies if any. This may leave some zombies around indefinitely.

See the following example.



Here, we ran `sleep 1` in the background. When it completed, `sleep` became a zombie, as indicated by the `Z` in the `S` column. It will stay in the zombie state[1] until *penn-shell* gets

[1] mmmm ... more brains!

around to reaping it when polling background processes the next time.

For this extra credit, you are to implement asynchronous `SIGCHLD` handling (with detailed documentation on how you implemented it). You must be able to handle signaling child processes on demand. You are not allowed to poll for child state changes. Instead, you will register a `SIGCHLD` handler. Remember, `SIGCHLD` is generated whenever a child changes state (*e.g.*, running to stopped, running to exited, etc.), and this will invoke the signal handler.

You must call `waitpid(2)` as follows:

```
int status;
int cpid = waitpid(-1, &status, WNOHANG | WUNTRACED);
```

In other words, you cannot wait for a particular process or process group.

You are required to add a command-line option `--async` to *penn-shell*. When run with this option, *penn-shell* will remove zombies asynchronously.

Before you submit *penn-shell* to the autograder, make sure you can pass the following basic test:

```
# penn-shell --async
penn-shell> sleep 5 &
Running: sleep 5
penn-shell> cat  # within 5 seconds
```

Note that sleep 5 is not printed while cat is running. You are expected to print the completed job **after** the current running process is finished. While `cat` is waiting for input, run `ps j` in another terminal, and you should see no zombies:

```
# ps j
   PPID     PID    PGID     SID TTY        TPGID STAT   UID   TIME COMMAND
   1004    1005    1005    1005 pts/0      12101 Ss    1000   0:00 -bash
   1005   12095   12095    1005 pts/0      12101 S     1000   0:00 penn-shell --
  12095   12101   12101    1005 pts/0      12101 S+    1000   0:00 cat
  12288   12289   12289   12289 pts/1      12347 Ss    1000   0:00 -bash
  12289   12347   12347   12289 pts/1      12347 R+    1000   0:00 ps j
#
```

For this extra credit, you can also use the following system calls and library functions:

- `sigprocmask(2), sigsuspend(2)`
- `sigemptyset(3), sigaddset(3)`