Homework: Threads and Context Switching

This assignment requires the files xv6.pdf and xv6_rev0.zip. You may download them from the Assignments page.

Read: setjmp.S and proc.c (focus on the code that switches between processes, specifically `scheduler` and `sched`).

Hand-In Procedure

You are to turn in this homework during lecture. Please write up your answers to the exercises below and hand them in to a 6.828 staff member at the beginning of lecture.

Introduction

In this homework you will investigate how the kernel switches between two processes.

Assignment:

Turn in: Where does the scheduler's stack sit in memory? Is it the same stack that a running process uses when executing in the kernel? (The scheduler is in `proc.c`.)

Turn in: What value does `setjmp` return first to its caller? (`Setjmp` is in `setjmp.S`.)

Turn in: What value does `setjmp` return when it returns the second time to its caller?

Turn in:: If we look at a `setjmp` on a particular jump buffer followed by a `longjmp` to that same buffer (that is, both functions are called once), `setjmp` returns *twice!* and `longjmp` never returns. How is this accomplished by `setjmp.S`? (A sentence or two will suffice.)

```
        longjmp(&p->jmpbuf);
```

In `proc.c`'s `scheduler` there are two lines:

```
      if(setjmp(&cpus[cpu()].jmpbuf) == 0)
```

Replace these with:

```
      cprintf("setjmp called in scheduler\n");
      if(setjmp(&cpus[cpu()].jmpbuf) == 0){
        cprintf("setjmp in scheduler returned 0; longjmp\n");
        longjmp(&p->jmpbuf);
      }else
        cprintf("setjmp in scheduler returned 1\n");
```

Similarly, in `sched`, replace:

```
  if(setjmp(&p->jmpbuf) == 0)
    longjmp(&cpus[cpu()].jmpbuf);
```

with

```
cprintf("setjmp called in sched\n");
if(setjmp(&p->jmpbuf) == 0){
  cprintf("setjmp in sched returned 0; longjmp\n");
  longjmp(&cpus[cpu()].jmpbuf);
}else
  cprintf("setjmp in sched returned 1\n");
```

Rebuild your kernel and boot it on bochs. You will get around 150 printed lines about `setjmp` as the kernel boots.

For now, ignore the first few lines of output. You should see a regular six-line pattern repeated over and over in the printed lines.

Turn in: What is the six-line pattern? (The first line is `setjmp called in scheduler`.)

Turn in: Why are there two `setjmp returned` prints for each `setjmp called` print?

Now look at the first three lines of output about `setjmp`. You will notice that they do not follow the pattern. The first two lines are:

```
setjmp called in scheduler
setjmp in scheduler returned 0; longjmp
```

but then some lines seem to be missing: this first `longjmp` does not behave like the others.

Turn in: In the repeated six-line pattern, where does the `longjmp` in the scheduler return to (which line number in the printout)? Where does the very first `longjmp` in the scheduler return to (which line number in the printout)?

(Feel free to set a breakpoint at the return statement in `longjmp` to answer this question.)

This completes the homework.