

CS 370 – Project #1, Part C

Purpose: Become familiar with the **xv6** Teaching Operating System, shell organization, and system calls
Points: 100

Introduction:

The **xv6** OS is an educational OS designed by MIT for use as a teaching tool to give students hands-on experience with topics such as shell organization, virtual memory, CPU scheduling, and file systems.

Resources:

The following resources provide more in depth information regarding **xv6**. They include the **xv6** reference book, tools for installing, and guidance to better understand **xv6**.

1. **xv6** Reference Book: <https://pdos.csail.mit.edu/6.828/2020/xv6/book-riscv-rev1.pdf>
2. Lab Tools Guide: <https://pdos.csail.mit.edu/6.828/2020/tools.html>
3. Lab Guidance: <https://pdos.csail.mit.edu/6.828/2020/labs/guidance.html>

Project:

Complete the following steps.

- Complete Project #1, Part A and Part B
 - Become familiar with the **xv6** Teaching Operating System.
 - Implement a new system call.
- Implement basic system call
 - Implement a *sps()* system call to display a simplified version of the Linux process status system service.
- Implement a simple C program, **ps.c**, to call the new display process status system service.

Submission

When complete, submit:

- A copy of the **zipped xv6 folder** (not qemu) via the class web page (assignment submission link) by class time. After committing (**git commit**) your final changes, clean and create a zip of the entire project:

```
xv6 $ git commit
xv6 $ make clean
xv6 $ cd ..
$ zip project1.zip -r xv6
```

- Submit the **project1.zip** via Canvas.
- Note, **the submission must include Part A and Part B code.**

Submissions received after the due date/time will not be accepted.

Six Stages of Debugging

1. That can't happen.
2. That doesn't happen on my machine.
3. That shouldn't happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?

Test Program

Implement a user level program to call the new *sps()* system service. This will likely have only one line (excluding the exit).

System Call

Implement a new system, *sps()*. The system call should display the status information for any active processes in the system. The implementation of *sps()*, int *sps(void)*, will be in the *proc.c* file.

The implementation will involve scanning the xv6 process table looking for valid entries (see for loop in *procdump()*). Before the loop, a header should be printed (see example). In the loop, the first action should be to acquire a lock (*acquire(&p->lock);*). Then, if the process state (*p->state*), is either RUNNING, RUNNABLE, or SLEEPING, the process name (*p->name*), pid (*p->pid*), state, and memory (*p->sz*) should be displayed tab (*/t*) separated. Since the process state is an enumeration (see *proc.h*) you will need to convert that to an appropriate string. The last action in the loop should be to release the lock (*release(&p->lock);*). See the example output for formatting.

Files

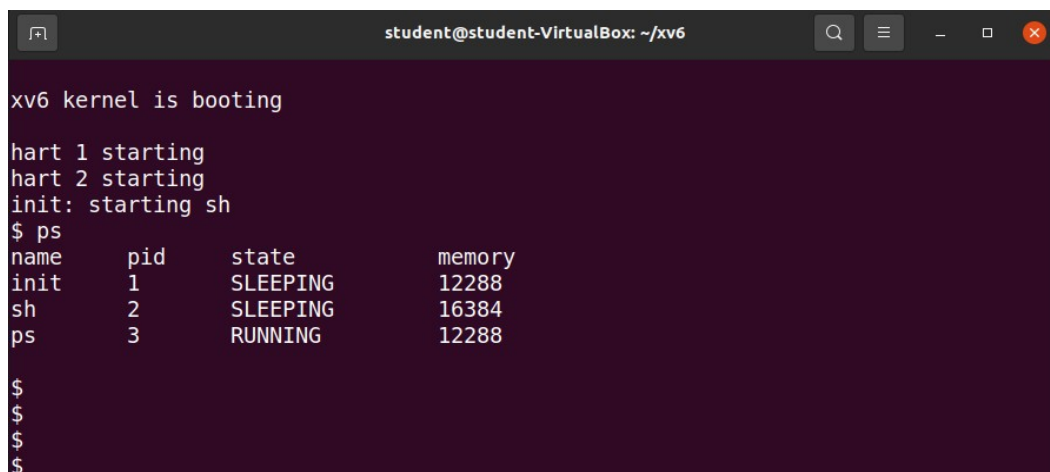
As described in Part B, in order to define your own system call in *xv6*, you need to make changes to files:

<i>xv6/Makefile</i>	// add ps test program
<i>xv6/user/user.h</i>	// add system call
<i>xv6/user/usys.pl</i>	// add entry
<i>xv6/kernel/syscall.h</i>	// assign number
<i>xv6/kernel/syscall.c</i>	// declarations (2)
<i>xv6/kernel/sysproc.c</i>	// call to <i>sps()</i> function
<i>xv6/kernel/defs.h</i>	// <i>sps()</i> declation
<i>xv6/kernel/proc.c</i>	// <i>sps()</i> implementation

As noted above, you will create the file *xv6/user/ps.c*. to call the new *sps()* system service.

Example:

Below is an example of the output of the *ps* test program.



```
student@student-VirtualBox: ~/xv6
xv6 kernel is booting
hart 1 starting
hart 2 starting
init: starting sh
$ ps
name    pid    state    memory
init     1    SLEEPING 12288
sh       2    SLEEPING 16384
ps       3    RUNNING  12288
$
$
$
$
```