**EECS 338**

***Today...***

1. **HW #1 discussion**
2. **GPL is evil? Torvalds speaks up!**
3. **Concurrency using fork()**

**Announcements**

* How was the recitation?
* HW #2 posted! More on this later today.

**HW #1: status column**

<https://idea.popcount.org/2012-12-11-linux-process-states/>

R = running or runnable (on run queue)

D = uninterruptible sleep (usually IO)

S = interruptible sleep (waiting for an event to complete)

Z = defunct/zombie, terminated but not reaped by its parent

T = stopped, either by a job control signal or because

it is being traced

**HW #1: understanding %CPU**

Alternative (my 4-core computer):

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

2993 chris 20 0 4388 616 548 R 85.0 0.0 9:24.88 while

3022 chris 20 0 4388 624 556 R 82.1 0.0 9:03.21 while

3029 chris 20 0 4388 660 592 R 82.1 0.0 8:58.21 while

3039 chris 20 0 4388 712 644 R 77.1 0.0 8:32.02 while

3029 chris 20 0 4388 660 592 R 70.8 0.0 8:55.74 while

CPU info:

<http://www.binarytides.com/linux-cpu-information/>

**HW #1: killing processes**

* “pkill *name*” vs. “kill *PID*”
* <http://www.linfo.org/kill.html>
* <http://man7.org/linux/man-pages/man7/signal.7.html>
* Kill all processes and close shell: kill -9 -1

**HW #1: traced, killed, and overflowed**

*What happens in “input.c” when the input exceeds the limit of int?*

* 231-1 = 2147483647 is the maximum for a 32-bit signed integer
* What is wrong here? <https://en.wikipedia.org/wiki/C_data_types>
* Two’s complement
  + <https://en.wikipedia.org/wiki/Two%27s_complement>
  + Convert from positive to two’s complement: invert and add one.

|  |  |  |
| --- | --- | --- |
| **Input** | **eecslab-1** | **HPCC** |
| 2147483647 | 2147483647 | 2147483647 |
| -2147483648 |  |  |
| 2147483648 |  |  |
| 2147483649 |  |  |
| -2147483649 |  |  |
| 4294967296 |  |  |

**GPL is evil?**

Linus Torvalds on GPL:

<https://www.youtube.com/watch?v=PaKIZ7gJlRU>

Extras:

<https://www.gnu.org/licenses/gpl-3.0.en.html>

<https://en.wikipedia.org/wiki/Free_Software_Foundation>

<https://en.wikipedia.org/wiki/Tivoization>

<https://www.eff.org/>

**Ch 3: concurrency using fork()**

*See separate textbook slides (posted with these notes)*

**process**: <http://www.linfo.org/process.html>

A process is an executing (i.e., running) instance of a program. Processes are also frequently referred to as tasks.

Each process consists of (1) system resources that are allocated to it, (2) a section of memory, (3) security attributes (such as its owner and its set of permissions) and (4) the processor state.

**scheduler:** <https://en.wikipedia.org/wiki/Scheduling_(computing)>

Scheduling is the method by which work specified by some means is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources.

Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality of service.

**context switch:** <https://en.wikipedia.org/wiki/Context_switch>

(Also: <http://www.linfo.org/context_switch.html>)

The process of storing and restoring the state (the execution context) of a process or thread so that execution can be resumed from the same point at a later time. This enables multiple processes to share a single CPU.

Context switches are usually computationally intensive. In the Linux kernel, context switching involves switching registers, stack pointer, and program counter, but is independent of address space switching, though in a process switch an address space switch also happens.

**Ch 3: concurrency using fork()**

* fork1.c (Figure 3.9)
  + fork()
    - <http://man7.org/linux/man-pages/man2/fork.2.html>
    - Some may need #include <sys/wait.h>
  + pid < 0
    - <http://man7.org/linux/man-pages/man2/fork.2.html#ERRORS>
  + execlp: <http://linux.die.net/man/3/execlp>
  + wait(NULL)
* fork-question-1.c: Exercise 3.1 (Figure 3.30)
* fork-question-2.c: Exercise 3.2 (Figure 3.31) (add fflush?)

**Ch 3: concurrency using fork()**

HW #2