Homework 1

CS5460: Operating Systems, Spring 2018 **Jake Pitkin**
University of Utah

2.13 **Describe at least two general methods for passing parameters to the operating system.**

1. Syscalls use registers to pass arguments to the operating system. The number of the syscall to be executed is stored in `%rax` with a few other registers used to store parameters.
2. Large parameters to syscalls, such as buffers, are passed to the kernel by passing a pointer to the block of memory.

2.14 **Describe how you could obtain a statistical profile of the amount of time spent by a program executing different sections of its code. Discuss the importance of obtaining such a profile.**

First, I would identify sections of my code that are making sycalls. I would keep a few statistics about each syscall: the average time each syscall takes, the average time for each category of syscalls (talking to devices, I/O, etc), and the total time spent making syscalls. Additionally I would time code that doesn't require syscalls.

This would allow me to see a ratio of how much of my program runtime is spent making syscalls. I could also see if specific syscalls or categories of calls dominate the runtime of my program. This would be important as operations such as I/O can dominate the runtime of a program and could be important areas for program optimization. The program being profiled would be ran a large number of times to get good average performance. This is important as OS scheduling is complicated and using a small dataset could lead to misleading results.

3.9 **Describe the actions taken by a kernel to context-switch between processes.**

1. P_1 experiences an interrupt or trap and enters the handler.
2. The scheduler has selected P_2 as the next process to run.
3. The CPU state of P_1 is stored in its process control block (PCB).
4. The CPU is loaded with the state of P_2 from its PCB.
5. The handler that P_1 entered is returned from with P_2 being the active process.

3.12 **Including the initial parent process, how many processes are created by the provided program?**

The program will create a total of **eight** processes. Consider the following table which shows the processes created at each `fork` and call its parent (process creation order is non-deterministic and the `pids` do not reflect a creation order).

pid	ppid	fork
p1	-	-
p2	p1	fork 1
p3	p1	fork 2
p4	p1	fork 3
p5	p2	fork 2
p6	p2	fork 3
p7	p3	fork 3
p8	p5	fork 3

3.17 **Using the provided program, explain what the output will be at lines X and Y.**

There are two processes in this program, a parent and a child. The child will be first to print to `stdout` and complete as the parent calls `wait()` above their for-loop that outputs. The `wait()` syscall causes a process to wait on the given line until its children processes are done executing. The other point of interest is both the parent and child have their own copy of the `nums` array. As such, when the child process is modifying their copy of `nums`, this won't be reflected in the parent's copy. As such, the child will output the negative squares of each value in `nums` followed by the parent process just outputting each value in `nums`.

Output: CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16 PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4