Name: **Justin Weigle**

This must be submitted to Gradescope AND you must use this as a template – you MAY NOT change the pagination. Question 11 starts at the top of page 5, yours must or you will get a zero.

1. Describe the steps followed by the CPU and the kernel to handle a context switch.

   - receives an interrupt or system call
   - save the state of the current process into PCB
   - restore the state of a different process from the PCB

2. When should you choose to use ordinary vs. named pipes, and vice versa?

   Ordinary pipes should be used if you want a unidirectional child - parent relationship and communication.
   Named pipes should be used if you need bidirectional communication with no parent - child relationship.

3. Using the program in following figure, identify the values of pid at lines A, B, C,and D. (Assume that the actual pids of the parent and child are 5500 and 5510, respectively.)

A:	**0**	C:	**5500**
B:	**5500**	D:	**5499**

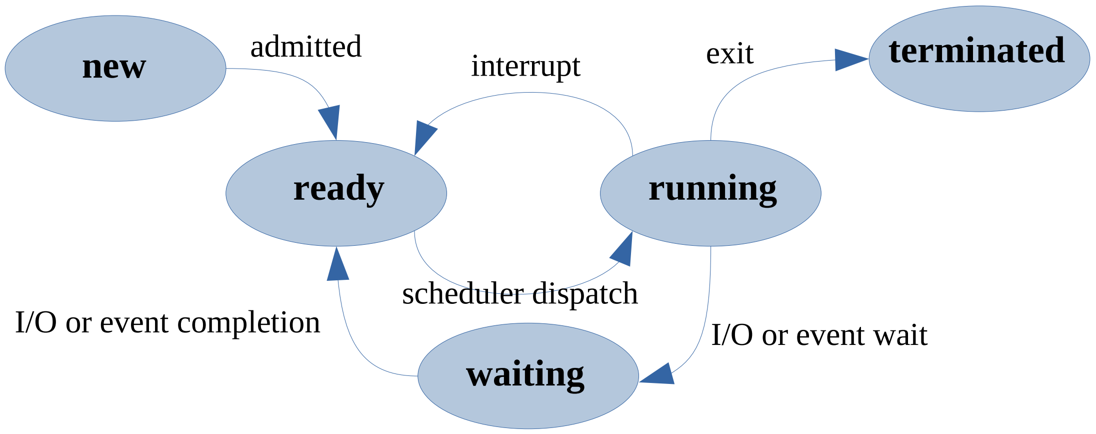
```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

   /* fork a child process */
   pid = fork();

   if (pid < 0) { /* error occurred */
     fprintf(stderr, "Fork Failed");
     return 1;
   }
   else if (pid == 0) { /* child process */
     pid1 = getpid();
     printf("child: pid = %d",pid); /* A */
     printf("child: pid1 = %d",pid1); /* B */
   }
   else { /* parent process */
     pid1 = getpid();
     printf("parent: pid = %d",pid); /* C */
     printf("parent: pid1 = %d",pid1); /* D */
     wait(NULL);
   }

   return 0;
}
```

Hint: remember that data with a value *before* the fork is copied, values *after* the fork are not shared between the parent and the child

4. Your process is using memory and you try to access a virtual address that hasn't been given to your process. Your friend, Bob, says that "its OK, the operating system will catch it," but you know that Bob is being really sloppy with his language. Whats wrong with his statement and how would you correct it.

When a user program tries to execute an illegal instruction or access memory that is not in the user's address space, the hardware traps to the operating system. The trap transfers control through the interrupt vector to the OS. The OS then terminates the program abnormally.

5. What are the five stages of the process state? Draw the chart.



A "legal" state transition would be one of the edges in your chart (above). List an "illegal" state and explain why its illegal.

A state transition from running to new would be illegal, because if a process has been running, obviously it isn't new.

6. What is the difference between a *program* and a *process*?

   A process is a program in execution

7. What is the operating system scheduler?

   The process scheduler selects available processes for execution on a core.

8. What is a CPU bound process?

   A process that uses more of its time doing computations than I/O

9. What is an I/O bound process?

   A process that uses more of its time doing I/O than computations.

10. How could you use the UNIX “time” command to tell the difference?

    A process that is I/O bound will show higher amounts of inputs and outputs, while a process that is CPU bound would show higher CPU usage.
    Also the CPU time should be lower than real time for I/O bound processes, and should be nearly the same for CPU bound processes.

11. What is a zombie process?

    A process that has terminated, but whose parent has not yet called wait(). These processes can also become orphans and must be inherited by another process such as init or systemd to invoke wait() to terminate the orphan.

Coding Questions – These are Individual Exercises!!!

Do not submit your code for these here in this homework file, rather submit your code to the Grad-eScope auto-grader. These must be in C / C++ and you need to include a Makefile so that I can do "make" and build your programs. Zip all of these up and upload the file to Grade scope.

1. Write a "menu" program that will display a menu with 5 different commands, and prompt the user to enter the number of the command that they should run. Use fork() and exec() to run those commands. After the program is done, you should show the user the menu. Make sure that choice "0" exits your menu. For example:

   Alices Cool Menu – Enter the # of the command
   1 – list all the files in your home directory
   2 – ssh into sloop
   3 – rsync your files from sloop to your home directory
   4 – backup all of your files to a thumb drive
   5 – play dungeon!
   0 – quit

Again – your list of commands will be different and unique to you. Take a look at the examples that I've handed out for how to use the fork(), exec(), and some other important system calls.

2. Make the worst facebook "wall" ever, no, actually, facebook beat you to it, so 2nd worse wall ever. When the user runs the program, you will either create or reuse an existing shared memory segment. This segment needs to be large enough to hold a message, then display the contents of the shared memory to the user (e.g. printf), and then ask the user if they want to change the message. Each time you run the command, you should be able to see your old message, and then possibly change it.

3. Write a prime number checker using two processes and an ordinary pipe between them. Your main process will create a pipe, and then fork. Your child will integers from its side of the pipe and if the number is prime it will print it out. If the child reads a negative number, then it will close its pipe and exit. Your parent will write at least 50 numbers to the write side of its pipe, and it will send a negative number when its done. It will have to prevent zombies.