Julian Garcia
ID: 900334702

CSE 222 Systems Programming
Homework 3 (100 points)

1. [20 pts] What output would be generated by the following programs (How many times will “This is my output” be printed? Explain why.
   If the output is redirected to a temporary file, is there any difference of the output of each program?

(a)

```
int main(void) {
   printf("This is my output.");
   fork();
   exit(0);
}
```

“This is my output” will be printed twice, this is because output is line buffered and since “This is my output” statement lacks a newline, the buffer is not flushed. Since the child process inherits every part of the parent, along with the unflushed buffer, when that child process exits that buffer it inherits is flushed and so it is printed to the terminal. If we were to pipe this to a temporary file, it would be fully buffered and we’d only get one “This is my output” printed out because the child process wouldn’t be carrying that unflushed copy.

(b)

```
int main(void) {
   printf("This is my output.\n");
   fork();
   exit(0);
}
```

“This is my output” will only be printed once, this is again because we have a line buffered scheme and the “\n” allows for the buffer to be flushed, and so the child process will not inherit the unflushed output. Piping this to a temporary file would result in the same output since the full buffering would still only capture one “This is my output” to print.

Julian Garcia
ID: 900334702

2. [20 pts] The following code segment connects the standard output of a process to its standard input through a pipe. Assume all calls are successful

```
int fd[2];
pipe(fd);
dup2(fd[0], STDIN_FILENO);
dup2(fd[1], STDOUT_FILENO);
close(fd[0]);
close(fd[1]);
```

(a) What happens if the process then executes the following code segment after connecting the standard output to the standard input through a pipe?

```
int i, myint;
for (i = 0; i < 100; i++) {
   write(STDOUT_FILENO, &i, sizeof(i));
   read(STDIN_FILENO, &myint, sizeof(myint));
   fprintf(stderr, "%d\n", myint);
}
```

The numbers 0 to 99 are printed to the terminal as myint reads the i given to the standard output through the standard input.

(b) What happens if the code segment in (a) is replaced by the following code segment?

```
int i, myint;
for (i = 0; i < 100; i++) {
   read(STDIN_FILENO, &myint, sizeof(myint));
   write(STDOUT_FILENO, &i, sizeof(i));
   fprintf(stderr, "%d\n", myint);
}
```

In this case, an infinite loop of 0's would be printed out since the integer i will continually be copying the value of myint, which is never incremented in this code.

3. [10 pts] What is a race condition? Explain and give a simple example.

A race condition is a condition whose output is dependent on which order the processes run. A simple example is when a process is forked and you have two different outputs dependent on if the process is the child or the parent, this is a race condition since the output changes depending on if the parent or child runs first.

4. [10 pts] Give a situation where use of a FIFO would be more desirable than a pipe.

One main situation is where you need to connect multiple readers/writers that are unrelated, or where you need a pipeline without creating a temporary file, since a regular pipe is unable to connect unrelated processes and always creates a temporary file for

Julian Garcia
ID: 900334702

the connection. Two examples of the uses of FIFO are in shell commands and client-server applications.

5. [20 pts] Unless a file contains sensitive or confidential data, allowing other users to read the file causes no harm. But what happens if a malicious process randomly reads a message from a message queue that is being used by a server and several clients? What information does the malicious process need to know to read the message queue?

   To actually read the message queue, it would need the msqid, which would be the queue id you'd retrieve from msget, and it's a unique identifier that allows any process to connect to the message queue. If a malicious process randomly reads a message from the message queue, that message will no longer be available for the server or clients to read themselves.

   Also, if the process is able to read any message from the message queue that is being used by a server and several clients, then that process is theoretically able to run any process on that message queue as the information needed to run those processes is the same for just reading a message.

6. [20 pts] The following figure shows the sequence of semaphore operations at the beginning and at the end of the tasks A, B, C.
   Determine for each of the 4 cases a, b, c, d, given in Table1, whether or in which sequence the tasks are executed, using the initializations of the semaphore variables given in table 1.

Julian Garcia
ID: 900334702

Task A	Task B	Task C
P(SA)	P(SB)	P(SC)
P(SA)	.	P(SC)
P(SA)	.	P(SC)
.	.	.
.		.
.	V(SC)	V(SB)
V(SB)	V(SA)	V(SB)
END	END	END

Table 1 *The initial semaphore values of 4 cases*

Variable	a)	b)	c)	d)
SA	2	3	2	0
SB	0	0	1	0
SC	2	2	1	3

Note:
P() is an atomic operation that waits for semaphore to become positive, then decrements it by 1, i.e. if the semaphore is 0, P() is blocked.
V() is an atomic operation that increments the semaphore by 1, waking up a waiting P() if any.
SA, SB, SC are the three semaphores corresponding to Tasks A, B, and C.

P(SA) = SA - 1
P(0) = Blocked
V(SA) = SA + 1

a: No task will be executed
b: task A, task B, task C, task B
c: task B, task A, task B, task C, task B
d: task C, task B