CSSE 332 -- Operating Systems

Rose-Hulman Institute of Technology

Exam 1

Name: _____

Section: _____

Question	Points	Score
Problem 1	15	
Problem 2	5	
Problem 3	20	
Problem 4	10	
Problem 5	25	
Total:	75	

Problem 1. Assume we have two sibling processes $\texttt{P}_1$ and $\texttt{P}_2$ that share the same parent process `P`.

(a) (5 points) Describe a way that would allow $\texttt{P}_1$ and $\texttt{P}_2$ to establish a two-way communication channel, i.e., we would like both $\texttt{P}_1$ and $\texttt{P}_2$ to be able to read from and write to each other.

(b) (10 points) Write a simple pseudocode that illustrates how the approach can be implemented. It doesn't matter what gets read or written. What you need to show are the order of system calls (`fork`, `pipe`, `close`, etc.) and the file descriptors from which each `read` or `write` call should operate on.

Problem 2. (5 points) When writing a program, the developer realized that they don't care about alarm signals and they would like to completely disregard such signals. Describe a way, using specific syntax, that the developer can achieve that.

Hint: You can assume that the developer has access to the `setsighandler` function defined in class, but you need to describe the correct arguments for it if you need to use it.

Problem 3. Circle the correct answer for each of the following questions.

(a) (5 points) All signals can be caught and overridden by the programmer.

A. **True**.

B. **False**.

(b) (5 points) Multiple processes can use the same pipe as long as (1) they have a shared ancestor that created the pipe and (2) each process assumes the role of either a writer or that of a reader.

A. **True**.

B. **False**.

(c) (5 points) Assume that a parent process `P` has created two child processes, P_1 and P_2. Assume that P_1 has crashed a long time ago.

While `P` is running and has not called `wait` yet, P_1 can be described as a _____ process.

(d) (5 points) This question continues part(c). Assume now that P_2 also crashes (long after P_1 has crashed). Having realized that, `P` calls the `wait` system call.

It is guaranteed that `wait` will return after having consumed (or waited for) P_2.

A. **True.**

B. **False.**

Problem 4. (10 points) Describe what happens when you call `waitpid(pid, 0, 0);` where `pid` is the process ID of a child process that is never-ending (e.g., it runs into an infinite loop).

Note: The question below is a **design** question, there is no one right solution. Use your creative thinking along with the tools we learned in this class to design a possible solution. Should you need to make any assumptions, please state them clearly in the answer box below.

Problem 5. Assume that a process `P` and its child process $\texttt{P}_1$ want to agree to a certain value of a variable `x`, read from a user input. In other words, each process wants to make sure the other one has the same value for `x`.

(a) (10 points) Describe in simple terms how the two processes can make sure that they both have the same value for `x` after they have read it form the user.

(b) (15 points) In the boxes below, write a pseudocode for each process that can implement your design above. You do not have to worry about the code that performs the `fork`, only focus on implementing the agreement design.

```
// code for P_1
x := get_value_from_user();

// write pseudocode that checks
// if both P_1 and P_2 have the
// same value of x
```

```
// code for P_2
x := get_value_from_user();

// write pseudocode that checks
// if both P_1 and P_2 have the
// same value of x
```