Jacob Stone

CIS 452

Lab 3

1. **(1 pt) what does the program print, and in what order? Be sure to identify which text is printed by which function.**

It prints “waiting…” until a signal is found from the keyboard. Then it prints “Inside signal handler.” and “outta handler.” on new lines. It then continues to print “Oops…: Interrupted system call”, “waiting…” 5 more times. Lastly it prints “Out of main loop”

1. **(2 pts) describe *exactly* what is happening to produce the answer observed for the above question**

The System is waiting for signal so it is printing the first output until i pressed Ctrl + C, Which gave it a quit signal, the signal handler was opened and exited from this command. The program then tries to go back into a loop but violates a statement since the boolean value is false. It continues like this until the loop counter goes to 0, and then prints out of main loop.

1. **(1 pt) What is the meaning of EINTR error code? Whose error code is checked by the if statement?**

Interrupted System Call. The signal system code is checked by the if statement

1. **(1 pt) Uncomment the two lines following the end of the loop. Read the man page of pause(). Recompile and rerun the program. What do you have to do on your terminal to see the output from the last printf() in main?**

Insert another signal.

1. **(1 pt) What happened to the parent process? What happened to the child process? Relate your observation to what we discussed in class**

The Process was terminated and only the child PID was printed. The child became a orphan.

1. **(1 pt) What happened to the parent process? What happened to the child process? Relate your observation to what we discussed in class**

The child process was terminated and only the parent PID was printed. The parent became a zombie

1. **Send a USR1 signal to the parent process and observe the output. Describe in sufficient details what happened to each process.**

During this experiment, both kept running until the signal was received where it was a user defined signal 1. And then the child was the only one to run.

1. **(1 pt) Send a USR1 signal to the child process and observe the output. Describe in sufficient details what happened to each process.**

Both processes began to run together, but as soon as the kill command was executed, the child was terminated and the signal process was from the child. The parent was the only one to run after.

1. **(1 pt) Suppose a process has used dup2() to make its standard output point to a file named temp. The process then issues a fork() system call to spawn a child. The child issues an exec() system call to execute a different program. Where does the standard output of the child process go? Explain.**

The child outputs to the file named temp instead of the output window, since the dup2 call process happened before the fork, the child and the parent still had its standard output going to the temp file.

1. **(1 pt) Suppose a process issues a fork() system call to spawn a child, then the parent uses dup2() to make its standard output point to a file named temp. The child issues an exec() system call to execute a different program. Where does the standard output of the child process go? Explain.**

As long as the child does not issue a dup2() call before calling the exec(), it will still output to the output window, Since nothing has changed/no command for the child was made it wont change its standard output.

1. **(4 pts) what *exactly* does the program do (i.e. describe its functionality)?**

The program waits for input from the user, then prints to a output from a process/

* 1. Which process (parent or child) reads data from the keyboard?

parent

* 1. Which process (parent or child) writes data to the pipe?

* 1. Which process (parent or child) reads data from the pipe?
  2. Which process (parent or child) prints to console?

1. (3 pts) create a diagram that visually describes the input/output structure of the executing program. how processes and handles as in the pipe example diagrammed in class; show the file descriptor table as presented above in the File I/O section:  
   1. at point A (after the pipe is created)
   2. at point B (after the parent forks)
   3. at point C (after the parent and child have duplicated descriptors)
   4. at point D (after parent and child close descriptors)