Edward Tischler
COP4600
0647
10/29/15
Term Project Part I

Affirmation that the work submitted is my own and that the Honor Code was neither bent nor broken:

Edward Tischler .

Summary of learning experience:

This assignment taught me a lot. The assignment had many clear objectives that were relevant to the operating systems material. In addition however, I learned a lot about the C programming language and how to find resources to complete a task. By this I mean that it was clear that we were meant to understand inter-process communication. We needed to learn how to fork processes and then control them once forked. Furthermore, we need to learn how to conceptually manage where forks were occurring. In this assignment in particular we needed to understand that when the receiver's PID is greater than zero than you are clearly in main and that you must fork here to have only a three process solution. Furthermore, we needed to understand that for each process we much close the pipe. At first I did not understand why we needed to close the pipe for the main process since there was no communication between those processes. However in the process of achieving this main goal of understanding how inter-process communication works, I also learned some other valuable skills. One of example of a learned skill was my ability to use "man" pages to understand how to read and write to the buffer. This I would say was the hardest part about this assignment. Conceptually this assignment wasn't so difficult but reading the "man" pages and then being able to understand what was in these pages and apply it was a considerable challenge for me. For example, I had no idea what a constand void * buffer was when I began this assignment but learned through trial and error. I also learned C, another valuable skill. This also caused some heart ache. It took me forever to understand that when I thought my loops weren't occurring they were instead actually occurring. However, when I tried printing multiple times in these loops the prints were being stored in a buffer. This I would also considered one of the most difficult parts of the assignment. The easiest parts of the assignment was to conceptually understand what was happening in the assignment and what our overall goal was. Had I had more practice I believe the execution of this assignment would have gone much quicker and smoother. This was a valuable learning experience.

Edward Tischler
COP4600
10/29/15
Term Project Part I

Does the program compile without errors?

- Yes, the program compiles without errors.

Does the program compile without warnings?

- Yes, the program compiles without warnings.

Does the program run without crashing?

- Yes, the program runs without crashing.

Describe how you tested the program.

- I tested my program by making sure my output matched the sample output given. Furthermore, I made sure that only one letter was written to the pipe, read from the pipe, and then written to standard output at a time.

Describe the ways in which the program does not meet assignment specifications.

- None

Describe all known and suspected bugs.

- No bugs known or suspected.

Does the program run correctly?

- Yes

OUTPUT.TXT

```
ZYXWVUTSRQPONMLKJIHGFEDCBA
Your pal, Process_4924 says, "Auf Wiedersehen."
Your pal, Process_7784 says, "Auf Wiedersehen."
Your pal, Process_8914 says, "Auf Wiedersehen."
```

pipetest.c

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
int main(){
 //get ready
 pid_t receiver;
 pid_t sender;
 pid_t pidnumber;

 // char buffer[26];
 int turn = 0;
 // int z = "z";
 int senderterminate = 0;
 int receiverterminate = 0;
char phrase[] = "ZYXWVUTSRQPONMLKJIHGFEDCBA\n";
 char buffer[50];
 int fd[2];
 int count = 0;
 int othercount = 0;
 char z = 'Z';
 const void * newbuf;
  pipe(fd);

  receiver = fork();
  //you are in main
```

```
if(receiver > 0){

  sender = fork();
}
//main process
if(receiver > 0 && sender > 0){
  close(fd[0]);
  close(fd[1]);
  sleep(30);
  pidnumber = getpid();
  printf("Your pal, Process_");
  printf("%i",pidnumber);
  printf(" says, \"Auf Wiedersehen.\"\n");
}
if(sender == 0 && senderterminate == 0){

  close(fd[0]);

   while(count < 26){

    write(fd[1],phrase+count,sizeof(char));

    count += 1;
    sleep(1);

   }
   close(fd[1]);
   senderterminate = 1;
    pidnumber = getpid();
  printf("Your pal, Process_");
```

```
    printf("%i",pidnumber);
    printf(" says, \"Auf Wiedersehen.\"\n");

  }

 if(receiver == 0 && receiverterminate == 0){

 close(fd[1]);
  while(othercount<26){



 read(fd[0], buffer+othercount, sizeof(char));
 write(1,buffer+othercount,sizeof(char));

  othercount+=1;
    }

   printf("\n");
  close(fd[0]);
   pidnumber = getpid();
    printf("Your pal, Process_");
    printf("%i",pidnumber);
    printf(" says, \"Auf Wiedersehen.\"\n");
  receiverterminate = 1;

 }
 return 0;

}
```