COMP 2560 FALL 2020

Assignment 4

Due date: Nov. 22, 2020

1. Design a small program to discover what happens within a parent process when a child closes a file descriptor inherited across a fork. In other words, does the file remain open in the parent, or is it closed there?

You need to write a program to investigate the answer to this question. Put in sufficient comment in your code so that the marker knows your design. Submit the code and the usual script files.

2. We discussed q1.c and q2.c (posted online under Week 6 in BB) in class on Jun. 10 briefly. Please study the source code for both programs. Compile and run them. Observe their outputs to see how many processes (parent plus children) each program produces. Explain how you arrive at your answer. Draw a illustrative diagram to assist your explanation if necessary.

3. Write a C program to execute multiple Unix commands in parallel.
   - The number of Unix commands is not fixed.
   - There is no communication among the Unix commands.
   - The Unix commands are given as command line arguments.
   - For simplicity, you can assume that each Unix command has exactly one argument except that the last one can have either no argument or one argument.

   For example,

   ```
   >>>>> miniminishell cat openfile.c ls –t  ps
   ```

   includes three Unix commands: cat with one argument openfile.c, ls with one argument –t, and ps with no argument.

   For each Unix command, use a separate child process to execute it. You need to print out each process id.

   Submit the code and the usual script files.

4. In the simplified shell program we discussed in week 8, the program uses two arrays of chars, as shown below (in the code userin.c)

    static char **inpbuf**[MAXBUF], **tokbuf**[2 * MAXBUF],
    *ptr = inpbuf, *tok = tokbuf;

    As explained in class, the array **inpbuf** is intended to store the user typed command and its arguments. The array **tokbuf** is intended to store each extracted tokens from inpbuf, such that, each element in the array **char *arg[MAXARG + 1]**; (in proc_line.c) is properly assigned using the information in **tokbuf** in order later on to invoke execvp system call in runcommand.c.

    You do not have to use tokbuf.

    Make any necessary changes to make the program still work but you only use inpbuf.

    Comment on your code to explain how you do it.

    Submit the code and the usual script files.