LE/EECS 3221 – Operating System Fundamentals

Summer 2025—Section E

Programming Assignment 1

Submission Deadline: June 15, 2024 before 23:59

Objectives

In this assignment we will try to practice the concept of parent-child process, inter-process communication, shell commands, and some related system calls.

General Assignment Notes

When writing and submitting your assignments follow these requirements:

- Name your source code file as: your YorkU student number, an underscore, 'a' (for 'assignment', then the assignment number in two digits. For example, if the user 100131001 submits Assignment 1, the name should be: `100131001_a01.c.txt`. No other file name format will be accepted. We require the .txt extension in the end because eClass does not allow .c extension.
- Use the same naming scheme for the assignment title when submitting the assignment to the eClass.
- For this assignment you must use the C language syntax and version that works in the Lassonde labs. **Hence, before submitting make sure your code works perfectly fine on the Lassonde servers**. **Your code must compile using make without errors and warnings.** You are provided with a makefile and instructions on how to use it. The makefile that we have provided doesn't let your program compile if there are warnings; hence, make sure **no warnings or errors** at all.
- **Test your program thoroughly with the `gcc` compiler in a Linux environment or preferably on Lassonde servers.**
- If your code does not compile, **then you will get zero**. Therefore, make sure that you have removed all syntax errors from your code.

Marks will be deducted from any question(s) where these requirements are not met.

WARNING

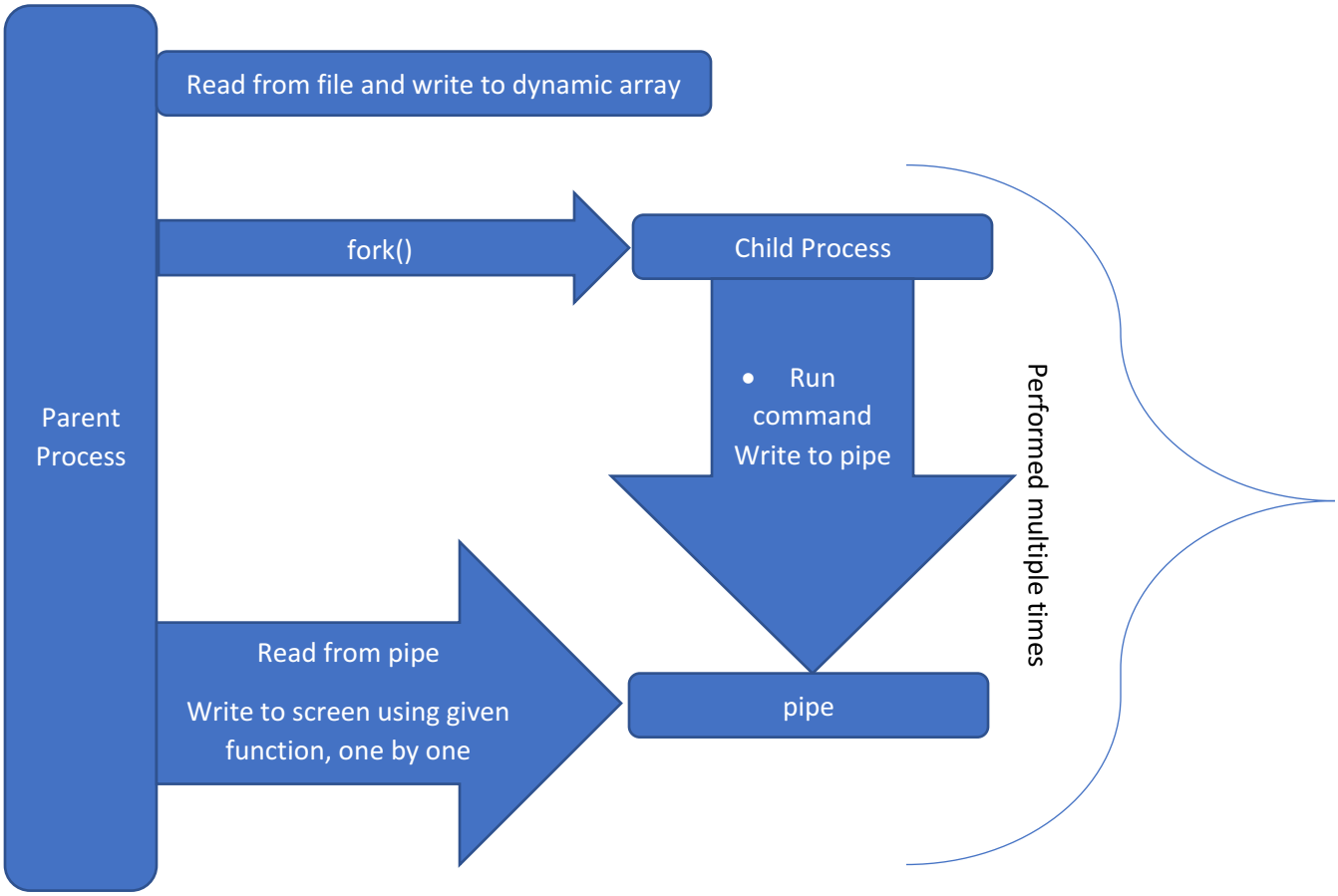
Follow the assignment instructions to the letter in terms of the file names and function names, as this assignment will be auto graded. If anything is not as per description, the auto grading will fail, and your assignment will be given a mark of 0.

You program must terminate normally on the provided test case; if it does not terminate normally then auto-grader will get empty output and it will be awarded zero.

Synopsis

In this assignment, you will have a parent process that will create children processes to perform tasks and will collect the output from these children processes. There are three tasks that should be performed:

- Read the input file that contains Linux shell commands. Parent process will read that.
- Execute the Linux shell commands read from the input file and execute them one by one. A child process will be created to execute these commands and the output will be returned by the child process in the form of string **using anonymous pipe**.
- The parent process will write the output of the command(s) to the screen using the given function.
- The following flow chart describes the flow of the program.



Description

Write a C/C++ program that includes the code for following tasks. Write all the code in single file:

1. Write the parent program as per the description in the "Synopsis". The parent program must use `fork` system call to create children process(es) when required.
2. Read the sample input file, sample_in.txt. **The name of the file must be given to the main/parent process through command line arguments.**
    a) This file contains one shell command per line.

b) Since the file is created in Windows environment, line terminator is '\r\n'. Make sure your code works for both kind of files, Linux and Windows.

3. Store the commands in a dynamically allocated array.
4. Execute the commands one by one with the help of a child process:
    a) Use a suitable version of `exec` system call to execute shell commands.
    b) The child process must write the output of the command to an anonymous pipe but doesn't display it to the screen. The parent process will read the command output from the pipe and will display it to the screen.
    c) You may use only one child process by doing fork once to execute all the commands or fork again and again.
    d) The parent process **must use** the provided output function, `writeOutput()`, to write the output to the screen. Invoke the output function iteratively, once for each command. **Warning: If you will not use this function to display the output then your outcome will not match with our outcome, and you will be awarded zero. You must copy this function into your C code, the submission file.**
5. **The other implementation details are on your discretion, and you are free to explore**.
6. In order to test your code, use the `sample_in.txt` and compare the output of your program for this input file with the content of `sample_out.txt`. However, keep in mind that due to the nature of the commands, the output is specific to the content of the directory in which you run your program. Hence, verify the output of commands by running them directly on Linux shell.
7. You **must not use mkfifo** or **system** in your code; otherwise, you will be given 0.

Grading

The grading will be done by an auto-grading script. Hence, it is extremely important to precisely follow the instructions. The grading usually has two components: test case output matching and use of required functions/systems calls (you need to figure out from description). The test cases for grading may or may not be different than the test case provided with the assignment.

Common Issues

These are the few of the common issues that may happen and will affect your grade. Therefore, follow the instructions properly:

- Your code has warnings, and you didn't remove them. **Your code must compile using make without errors and warnings.** You are provided with a makefile and instructions on how to use it. The makefile that we have provided doesn't let your program compile if there are warnings; hence, make sure **no warnings or errors** at all.
- You didn't take the name of the input file as command line argument, or you forgot to remove the filename that you hardcoded during the testing. Check # 2 under "Description".
- You didn't use the correct line terminator. Check # 2 under "Description".
- You forgot to remove **printf** that you have added for testing.