**1 Assignment task**

This assignment will focus on the use of calls such as pipe(), fork(), execve(), dup2(), open(), close(), read(), and write() in C programming to facilitate inter-process communication. The task is with the creation of two programs: a child program (the reverser program) and a controller program that will oversee multiple child programs. The communication between the processes will simulate more complex tasks – although you could achieve similar results to this assignment with different algorithms, you are specifically going to be looked by your ability to facilitate this inter-process communication.

**2 Reverser Program**

create a ‘reverser program’ that forms a simple processing unit for text. This program will read from standard input and write to standard output. The task of this program is to:

* Continuously read from standard input UNTIL it encounters EOF
* Store all bytes received from standard input
* Once EOF is encountered, repeat all received bytes in reverse order.

ensure that errors are handled appropriately.

This program has no knowledge about pipes.

**3 Controller Program**

create a ‘controller program’ that will make use of the above reverser program. This program will first receive instructions from a ‘.usp’ file, create the required number of child programs, then send information to those child programs.

**3.1 The ‘.usp’ File Format**

The contents of the .usp file will guide how the controller program will operate. The file will contain:

* One or more lines of text
* The text may be alphabetic, numeric, or symbolic data

For example:

!elif

PSU

a si

sihT

**3.2 Controller program task**

The controller program will be responsible for reading the ‘.usp’ file, creating and maintaining pipes, and creating child processes to process data. For **each line** of text in the ‘.usp’ file, the controller program will:

* Create a pipe
* Create a child process
* Replace the child process image with the ‘reverser program’ image outlined above (i.e. the child process will be running the reverser program)
* Write the line to the pipe for that child

The controller program will then:

* Read from pipes in reverse order
* Write each line read to a text file

For example, the sample ‘.usp’ file from above should result in a text file containing:

This

is a

USP

file!

For this sample text file, the controller program will create and maintain four pipes and four child processes. One with ten lines will have ten pipes/processes, twenty lines will have twenty pipes/processes, and so on.

**Note**

To perform the task that has been outlined here. You must need to ‘connect’ the two programs. Additionally, you should be freeing resources appropriately and accounting for any common errors that might happen when dealing with file I/O, creating pipes, etc.

**4 Restrictions**

This assignment is looking for the allocation specifically for the use of low-level system calls such as open(), close(), read(), write(), and execve(). For this assignment task, **any related library calls are NOT allowed. This includes functions such as fopen(), fclose(), fprintf(), fscanf(), fgetc(), fputc(), fgets(), fputs(), and so on. If any of these are used the task cannot be properly shown.**

printf() is allowed *only* for debugging or informational purposes (such as informing the using how to use the program correctly).

Library functions not related to reading, writing, or executing processes are allowed.

**5 what the task must be able to do**

* Reverser Program behaviour
  + Reads for standard input until EOF is encountered
  + Reverses text and outputs to standard output
* Controller Program behaviour
  + Processes the ‘.usp’ file
  + Creates pipes correctly
  + Creates child processes correctly
  + Handles file descriptors correctly
  + Replaces child process image correctly
  + Pipe I/O is handled correctly
  + Overall reversing effect is functional

## 6 What I need to receive and in what format

all necessary source code, make file. Include everything in a single .zip file You do not need to include any compiled code.

**(however, a demonstration will be needed)**

Keep in mind that other will use their with our own “.usp” files