6ELEN013W Operating Systems and Drivers

Coursework 2 – Programming Assignment

This assessment is weighted at 25% for the module.

Submission via Blackboard by 13:00 Thursday 1st April 2021

Written feedback and marks will be given 15 working days after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.

The work must be entirely your own. Any material from other authors must be correctly referenced in accordance with University regulations. Referencing entire paragraphs or paraphrasing them (i.e. rewriting them in your own words) is not acceptable. Use of references should support your argument, not be your argument. Any work found to be in breach of this will be dealt with officially.

WARNING : If your assignment consists mainly of a collection of referenced material or excessive collaboration with others, you will get 0% for the assignment which is likely to result in you having to do extra work over the summer and could result in you being capped for the module. If material is not correctly referenced your assignment will be considered as plagiarized and dealt with officially.

Specification

As explained during the lectures and tutorials, you are to write a program in C, based on the tutorial exercises, that performs the following:

Parent Process

- The process creates one child
- Map STDIN to the reading end of the pipe
- A file “STATUS.TXT” is created to contain information about the running of the program which should include:
    - The time the program was run
    - Process and Child process information (DON’T do this from the child processes)
    - You are also to write to it various pieces of information associated with the inode of STATUS.TXT.
    - When processes terminate
    - Use printf to write to the file (this will require a redirect)
- Map STDOUT to the status file
- It waits until the child completes
    - Detect this using a suitable signal
- Once the child is detected as terminated

- The random characters and numbers created by the child are then to be written to a database are via a python driver/connector.
- From the parent a python-based command is to be run to print out the contents of the database
  - Only the random characters/numbers and not the auto-number field

Child Process

- Map STDOUT to the writing end of the pipe
- It is to run until CTRL-C is pressed
- Every time CTRL+Z is pressed, a random number between 10 to 50 is generated and written to a pipe to the parent
  - Using printf
- Use a single signal handler

Python Notes

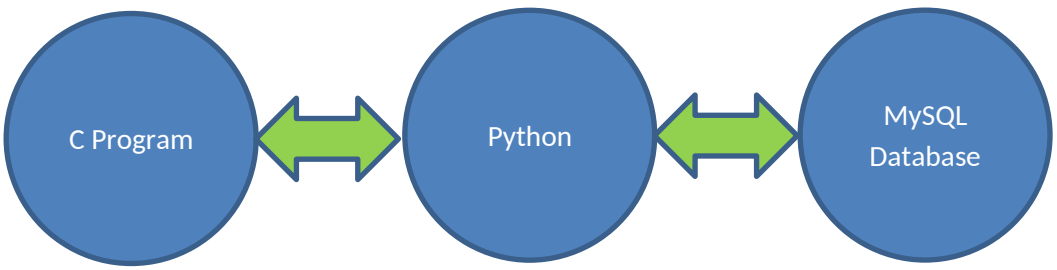
- Use python 3.9.2
- The python code is to be a connector/driver to the database
- Use python calls within your C code - it requires python.h
- The python code is to include the SQL statements to insert data into the database and retrieve information from it.

Database Notes

- Use MySQL
- Should only contain two fields
  - Auto number – unique id for record
  - The randomly generated number from the C program
- You can pre-create the database separately; it is not necessary to create it from your C or Python code. You just need to ensure that it is emptied when the results are retrieved from the child process

General Specification

- There should be a single signal handler for the parent process and a separate single one for the child.
- Checks should be done on relevant function calls to ensure no errors and that the operation was successful. There are a number of functions that can return -1.
- Proper housekeeping should be done when processes terminate (e.g. closing/releasing resources)
- Ensure that printf calls are written to files or the pipe and not to the console.



- There will be at least 3 signals in the program.
- A single signal handler for each process should be written to manage the signals
- Be aware that each process should not react to the signal that the other process uses.
- Marks will be awarded for presentation (includes comments).
- Only the source code is to be submitted.

Coursework 2 Mark Allocation Scheme

Criteria	Marks
Using signals, appropriate implementation of:	
• Handle CTRL-C	4
• Handle CTRL-Z	4
• Ignoring the relevant signals in the appropriate process(es)	4
• Detecting Child process completion	6
• Signal handlers	6
Main structure – Appropriate implementation and use of:	
• Pipe	4
• Fork to create child process	4
• Redirection of input and output streams	6
• Generation of random letters and characters	4
File Operations	
• Correct creating and handling of output file with suitable messages for program run	
○ Parent information	4
○ Child information	4
○ Inode information	4
○ General information from the running of the program	4
Python	
• C to Python Interfacing	6
• Writing data to database through Python	8
• Displaying database results through call to Python	8
General	
• Ensuring operation checks are performed to ensure that they succeeded (e.g. fork, dup2, signal, fopen, etc)	8
• Good layout of code – appropriate indentation	6
• Technical commenting of parts of code	6