Pipe Simulator

Unix Systems Programming (COMP2002)
Semester 1, 2021

Due: Sunday 16 May, 23:59
Weight: 20% of the unit mark

This assignment focuses on the use of system calls in C Programming to simulate a pipe in Unix.

1. Overall Task

A pipe in Unix is used to facilitate inter-process communication. In this assignment, you will be creating a simulation of this structure through the use of systems calls and linked lists. The first step will be the implementation of a linked list through the system calls. The next step will be the use of this data structure in executing processes in sequence and communicating the output of one process to be the input of the next.

In Unix/Linux you might use a pipe such as: "cat test.txt | grep -E bash". The output of cat is used as input to grep. Your program will simulate this, albeit it through an input file that lists the sequence programs to execute.

Note: Do be careful about what is and is not a system call. If in doubt, do ask!

Linked List Implementation

You will have created a linked list in your practical sessions. For this assignment, the functionality will be similar, but will be achieved through the use of system calls. The use of strace is recommended to determine what calls will need to be made to achieve similar functionality.

An overview of the requirements is as follows:
Your linked list must be generic.
Only system calls (or the glibc wrappers for system calls) and functions that you create yourself may be used.

The variant of linked list to be implemented (i.e. single-ended, double-ended, etc.) is up to your own design.
A test harness must be provided demonstrating the use of your linked list.
The test harness is not required to use system calls.

Pipe Simulation Program

The program will utilise the above linked list to simulate inter-process communication.

Usage

Requirements for the program usage are as follows:
The program should be used as: '`./pipesim inputfile [outputfile]`'
`pipesim` is the program name
`inputfile` will be a plain text file with a list of programs to be executed, including their parameters. Each line should correspond to a new program. e.g.:
'`cat test.txt`'
'`grep -E bash`'
`outputfile` is an optional parameter where the output of the final process will be written to. In the absence of this parameter, output should be printed to the screen.

Behaviour

The linked list will be used to store information about the processes to be run. These names and arguments of these processes are stored in the `inputfile` described in usage. The number of these processes should not be constrained.

Once the linked list has been populated with process information, the processes should be executed in sequence.

The program will iterate through the linked list; for each node it should:

- ➔ Create a new process via clone()
- ➔ Use the child process to execute the process indicated by the current node
- ➔ Store the output from the child process
- ➔ Use the output from the child process as input to the next node

System calls are also a necessity here.

Note: This means that you should have two processes executing at any one time. You should be careful here, as asynchronous programming can be tricky. I strongly advise that you design your program in such a way to minimise how much asynchronous processing you are actually doing.

2. Marking Criteria

Your work will be marked out of 45 marks, as follows:

Linked List Implementation

2 marks — Generic

3 marks — Appropriate memory management

5 marks — Test harness

Usage

1 marks — Usage as specified

2 marks — Robust against missing parameters

2 marks — Optional output file functioning

Program Behaviour

2 marks — Appropriate structs for processes

4 marks — Appropriate use of parent process

4 marks — Correct creation of child processes

Input/Output

5 marks — Output from child process

5 marks — Input to next process

Code Quality

5 marks – Program readability/commenting

5 marks – Memory management

System Calls

The above criteria make the assumption that you are using system calls where appropriate. If your program does not use them, you may expect penalisation proportional to the use of non-system calls.

Does the program run correctly?

This isn't strictly one of the criteria, but rather a cross-cutting concern that affects all of them. The marker will attempt to compile and run your code. If it does not work correctly, this will have implications for how we assess the other criteria.

3. Submission

Submit all of the above, plus a signed declaration of originality, to the appropriate area on Blackboard. Include everything in a single .zip/.tar.gz file (please avoid .rar, .zipx or .7z). You do not need to include any compiled code.
You must verify that your submission is correct and not corrupted. Once you have submitted, please download your own submission and thoroughly check that it is intact. You may make multiple submissions. Only your last one will be marked.

4. Academic Integrity

Please see the Coding and Academic Integrity Guidelines on Blackboard.
In summary, this is an assessable task. If you use someone else's work or assistance to help complete part of the assignment, where it's intended that you complete it yourself, you will have compromised the assessment. You will not receive marks for any parts of your submission that are not your own original work. Further, if you do not reference any external sources that you use, you are committing plagiarism and/or collusion, and penalties for academic misconduct may apply.
Curtin also provides general advice on academic integrity at academic integrity.curtin.edu.au
The unit coordinator may require you to provide an oral justification of, or to answer questions about, any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an academic misconduct inquiry.vv