COSC 3360/6310 - Operating Systems Spring 2022

Programming Assignment 1 - Pipes and Process Synchronization

Due Date: Friday, March 4, 2022, 11:59pm CST

Objectives

This assignment will help you understand several basic UNIX/Linux primitives, implemented at the process management layer of an operating system, such as fork(), pipe(), and execv() as well as simple process synchronization and dataflow systems.

In this assignment, you will implement a program to synchronize, with Unix/Linux pipes, concurrent processes which perform word sorting and duplicate-removal operations. You will learn concepts from several computer science disciplines, including data flow architecture, parallel computation, natural language processing (NLP), and, of course, operating systems. The objective is to use concurrent processes communicating with pipes to collaboratively and alphabetically sort English words, removing duplicates while indicating the number of each repeated word.

Adjacency Matrix for Representing Process Precedence Graph

The input (passed as the first argument of your main program) to your program is a process precedence/dataflow graph specified in a square $k \times k$ adjacency matrix M where rows and columns are both labeled from 0 to k-1 such as the following three examples with $k = 2, 3, 7$, respectively:

```
0 1
0 0

0 0 1
0 0 1
0 0 0

0 0 0 1 0 0 0
0 0 0 1 1 0 0
0 0 0 0 1 0 1
0 0 0 0 0 1 0
0 0 0 0 0 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

The element $M_{i,j}$ is one (1) when there is a directed edge from vertex i to vertex j, and zero (0) when there is no edge. Such a matrix for representing a directed graph is typically introduced in an undergraduate data sturctures course (a prerequisite to this OS course).

Vertices are Unix Processes; Edges are Unix Pipes

Each vertex is represented by a Unix/Linux process you create. Each edge is represented by a Unix/Linux pipe you create. If a vertex has no incoming edges, then it is an input vertex. If a vertex has no outgoing edges, then it is an output vertex. Vertices which are neither input nor output are called internal vertices.

The first matrix above corresponds to the precedence (dependency) graph for the write-reader with pipe problem discussed in class (Lecture 5 Slides). The second matrix above corresponds to the precedence graph for the concurrent sorting example discussed in class (Lecture 4 Slides). The third matrix above corresponds to the following precedence graph:

```
v0 -> v3 -> v5
v1 -> v3
v1 -> v4 -> v5
v2 -> v4 -> v6
v2 -> v6
```

Word Processing

Each input vertex (process) reads an English word (a string of characters) stored in a second input file or from standard input (stdin). For example, in the third matrix above, there are 3 input vertices (processes) and therefore, there are 3 words to be read respectively by these input vertices (processes): v0, v1, and v2. The example input file here has the following content:

```
apple
orange
pear
```

To keep the input simple, the words to be read will be ordered for input from the lowest to the highest-indexed input vertices (processes). In this example, the first word in the input file is for v0, the second word is for v1, and the third word is for v3. In this example, 'apple' is the input to v0, 'orange' is the input to v1, and 'pear' is the input to v2. There are at most 10 vertices (processes) in the precedence graph corresponding to matrix M.

If a non-input vertex (process) has one incoming edge (pipe) and when it receives (reads) the incoming words, this process sends (writes) this word on each of its outgoing edges (pipes). If this vertex has two or more incoming edges (pipes), then it alphabetically sorts the incoming words, separating them by commas (,), and sends (writes) the resulting string to every outgoing pipe. If there are repeated (duplicated) words, then this vertex indicates the number of occurrences of the same word by appending this number in English before (with one blank space in between) the word and adds an 's' to this word to indicate that it is plural (do not worry about irregular plurals). For example, vertex v5 has two incoming pipes with the strings 'apple, orange' and 'orange, pear'. Therefore, the resulting string is: 'apple, two oranges, pear'. If there is word appearing two or more times with counting adjectives (not in this example, such as 'two oranges' and 'three oranges'), combine them into one phrase by adding the numbers indicated by the counting adjectives to derive a new counting adjective. In this case, the resulting string is 'five oranges'. As another example, 'orange' and 'three oranges' become 'four oranges'.

If there are two or more output vertices (processes) as in this example (v5 and v6), a new process P must be created with pipes to obtain the strings from these output vertices and perform the above word processing operations (sorting and duplicate-removal). The output of your program consists of a printout of the sorted words or phrases separated by commas and ended with a period (.) from P or the output vertex if there is only one such vertex. For this example, since the string at v5 is 'apple, two oranges, pear' and the string at v6 is 'orange, two pears', the new process P combine these strings to yield the final output string for printing: 'apple, three oranges, three pears.'

Submitting the program:

For submission guidelines, please visit Blackboard/MS Teams/TAs' instructions.

Notes

Before you start your assignment, familiarize yourself with the way C/C++ programs handle arguments that are passed to them by execv() and with the UNIX functions fork(), pipe() and dup().

The programs you turn in should be well documented. Each C/C++ function should start by a brief description of its purpose and its parameters. Each variable declaration should contain a brief description of the variable(s) being declared.

Code-level plagiarism or collaboration is forbidden. If you copy code(s) from any source, including those from students who previously took the course and any online source, please explicitly cite it. Even with explicit citations, the logical similarity to any codes available online or submitted either this year or previous years should be less than 75%. Otherwise, you may end up receiving as a minimum penalty a '0' for this assignment and a maximum penalty of an 'F' for the course plus additional disciplinary actions.