Interprocess Communication Exercises

These exercises are grouped into three parts. Part I contains routine exercises to help you understand the ideas directly presented in lectures. The exercises in Part II are designed to grow and deepen your understanding of principles that underpin interprocess communication. These exercises extend the lecture material and invite you to think about 'why' questions and about optimisations to what was lectured as well as alternatives to what was lectured. Part III are open-ended, stretching questions that you could tackle in your mini research project.

I hope you enjoy working through these questions, puzzles and research questions!

John Fawcett, July 2023

Part I

1. What other concepts might you represent using file handles? What about virtual memory?
2. Write a program in any program language you are familiar with, e.g. Python or C, that declares 4 variables called a, b, c, and d, then saves them to a file on disk, loads their values back into 4 other variables called e, f, g, h, before finally confirming that the a=e, b=f, c=g and d=h.
3. Write two programs as follows:
    a. The first program should ask the user to type in two numbers, which it should write into a file called "data.txt". Then it should wait until a file called "total.txt" has appeared. When it does, the program should read a number from "total.txt" and print it out.
    b. The second program should wait until a file called "data.txt" appears. When it does, it should read two numbers from the file, add them together, and write the total into a file called "total.txt".
4. Write a single program that asks the user for two numbers then creates a pipe, writes the two numbers into it, then reads them back from the pipe and prints them out.
5. Modify your program in Q4 to match the following pseudo-C program:
```
[f1R, f1W] = pipe();
[f2R, f2W] = pipe();
int f = fork();
if (f == 0) {
      fclose(f1W); fclose(f2R);
      int x = fread(f1R), y = fread(f1R);
      int z = x + y; fwrite(f2W, &z, sizeof(int));
} else {
      fclose(f1R); fclose(f2W);
      int x = userInput(), y = userInput();
      fwrite(f1W, &x, sizeof(int)); fwrite(f1W, &y, sizeof(int));
      int z; fread(f2R, &z, sizeof(int)); print(z);
}
```
6. Explain which line(s) cause one or other of the processes in Q5 to block. Why is the program in Q5 more efficient than the program we wrote in Q3?
7. Using the encoding of vertices and edges from lectures, deserialise the graph encoded as 5, 6, 1,2, 2,3, 3,4, 4,5, 5,3, 4,1.

Part II

1. Suggest an encoding for tree data structures.
2. What algorithm could two programs use with shared memory to know when a message is complete and ready to be consumed? I.e. how does the reading program avoid reading a message before the writer has finished it?

Part III

1. Research *Named Pipes*. How are they different from pipes? What problem do they solve? Show how to use them with pseudo-code or a real program.
2. For two programs you use regularly, figure out how they use interprocess communication. Explain using pseudo-code (or a real program) how you would implement them.