Project

ECE 322

This is an individual assignment.
You may only request help or discuss with the course Instructor.
Presentation Due – Before Day of Final
Report Due – Day of Final

In this project, you will create a version of the pipe operations the Unix shell uses to connect standalone programs to each other.

Part 1: Standalone programs

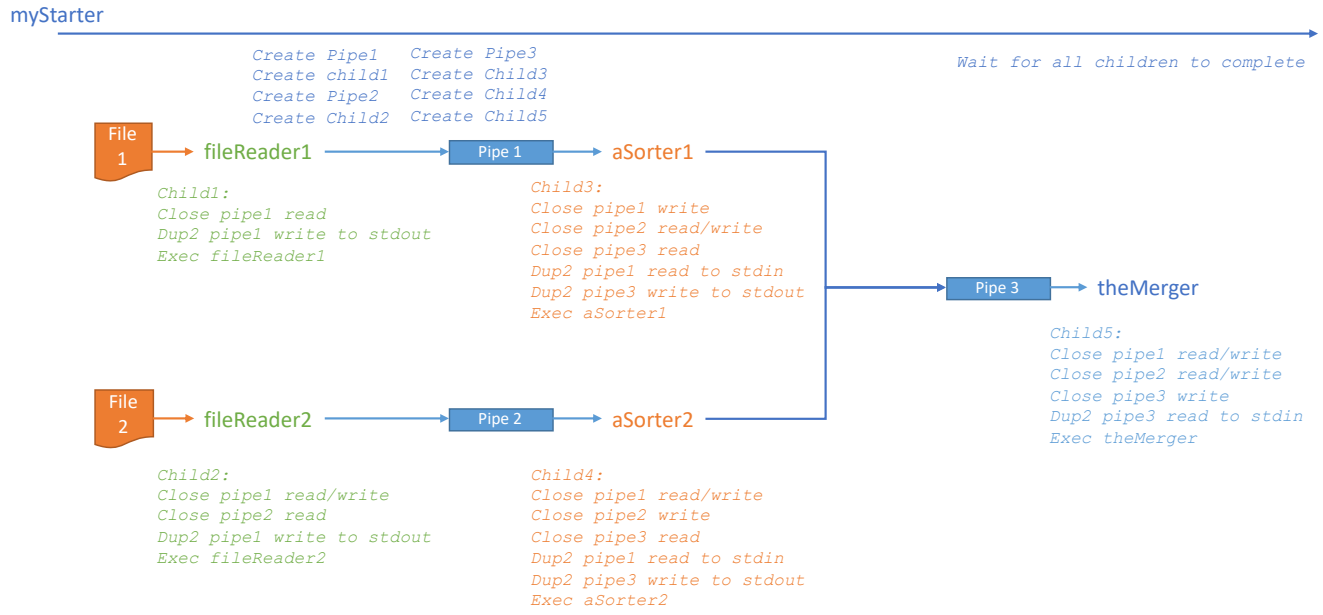
1. *fileReader <filename>*
    a. This program takes a single argument, the name of a text file
    b. It opens the file, reads each string in the file and writes it to STDOUT
    c. On EOF of the file, it closes the file and exits

2. *aSorter <number>*
    a. This program takes one argument, an integer
    b. It reads strings from STDIN and stores them in a self-sizing array
    c. On EOF on STDIN it:
        i. Sorts the array using bubble sort
        ii. Writes each element of the sorter array to STDOUT preceded by the number it received as an argument
    d. When done, it exits

3. *theMerger [filename]*
    a. This program takes one optional argument, a filename
    b. It reads data from STDIN until EOF
        i. Separates the data read into two arrays, based on the number preceding the data element
    c. It merges the two arrays of data it received to a third array
    d. If the argument was present, it writes the third array to the file provided
    e. OR if the arguments was not present, it writes the third array to STDOUT
    f. Exits

Part 2: Pipe Starter (simple)

The pipe starter is used to connect the standalone programs into a pipeline as shown below.



myStarter <child1/2> <child1 argument> <child2 argument> <child3/4> <child5> <child5 argument>

The *myStarter* program is used to start 5 children and connect them with three pipes into a pipeline as shown in the diagram.

The arguments are:

<child1/2>:	name of executable for child1 and child2 (*fileReader*)
<child1 argument>:	argument for child 1 (file1.txt)
<child2 argument>:	argument for child2 (file2.txt)
<child3/4>:	name of executable for child3 and child4 (*aSorter*)
<child5>:	name of executable for child5 (*theMerger*)
<child5 argument>:	argument for child5 (file3.txt)

So the final test will be:

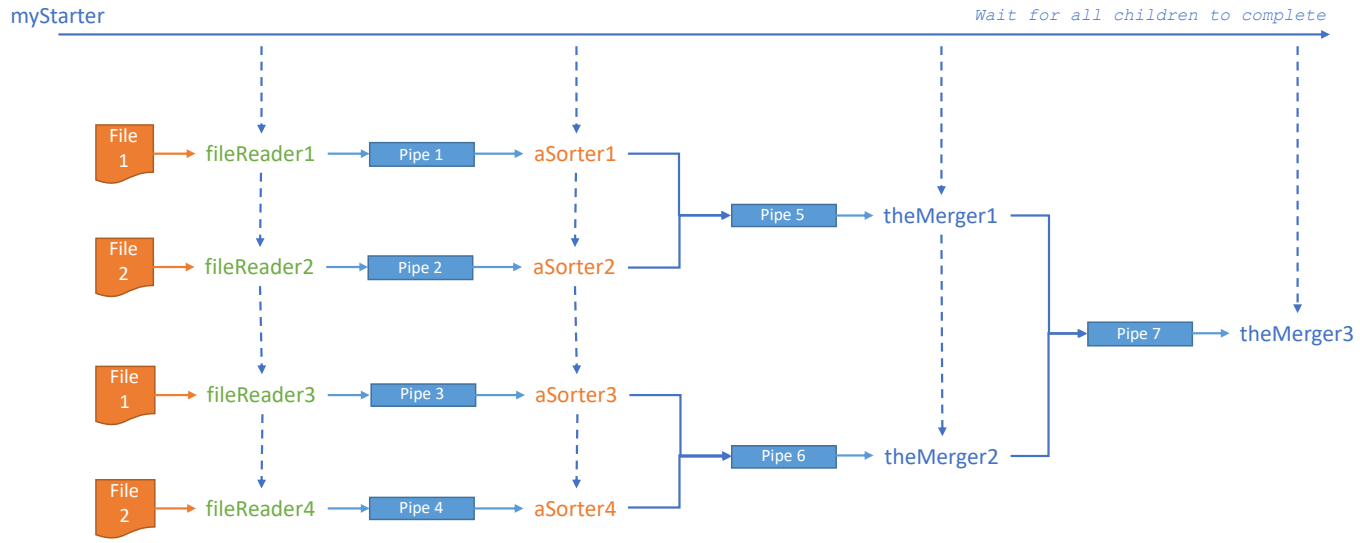
myStarter fileReader file1.txt file2.txt aSorter theMerger file3.txt

Part 3: Pipe Starter (more complex)
For this part, you have two options, you only need to complete 1 option.

Option 1: myStarter2
This option requires no changes in the standalone programs but has a more complex arrangement of pipes. In this case, you need to duplicate the pipeline from Part 2 and add another merger to merge the output of the first two mergers (note that these are started without an argument, so they write their outputs to STDOUT). It updates the *myStarter* to *myStarter2* with the following arguments:

myStarter2 fileReader file1.txt file2.txt file3.txt file4.txt aSorter theMerger file5.txt

The pipeline is now as shown:

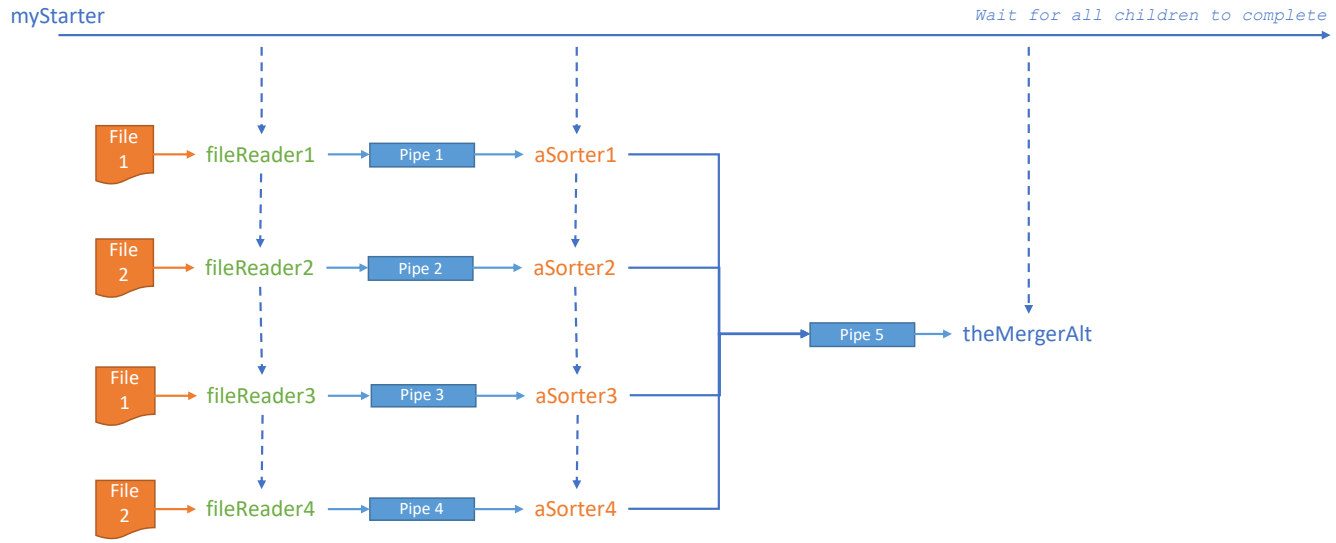


Option 2: myStarter3

This option has a simpler pipe setup for the pipeline but requires a modified version the *theMerger*. This modified merger, *theMerger2*, merges 4 arrays into a single array instead of just 2 arrays into a single array. It updates the *myStarter* to *myStarter3* with the following arguments:

myStarter3 fileReader file1.txt file2.txt file3.txt file4.txt aSorter theMerger2 file5.txt

The pipeline is now as shown:



Extra Credit: Only 1 option will be considered

Option 1:

Complete both Option 1 AND Option 2 of Part3

Option 2:

Use your version of stdio from HW3 in place of any STDIO calls in the system