Unix Systems Programming (COMP2002)
Assignment for Semester 1, 2023

Due: Tuesday 30 May, 23:59 (GMT+8)
Weight: 20% of the unit mark

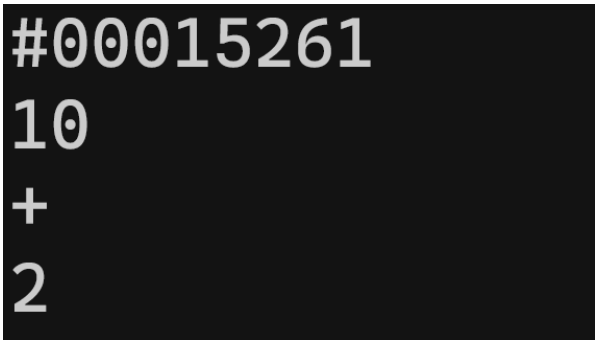
This assignment focus on the use of **pipe(), fork(), open(), close(), read(), and write()** system calls in C Programming to process text files in Unix.

1 Overall Task

In this assignment, you will be provided with some sample files with extension ".usp". Your program will retrieve the list of all files of that extension, and create one child process to handle each file. The result will be passed back to parent process which will put everything together in one text file.

1.1 Content of ".usp" file

The following figure shows one sample of the file:



The first number that starts with '#' is the unique ID of that file. Keep in mind that the file name is not necessarily the same as the unique ID inside it. The next three lines represents a simple mathematical operation. It will always be operation between two non-negative integers, and the result is also always a non-negative number. All the possible operators are:

- + : Addition operator
- - : Substraction operator
- * : Multiplication operator
- / : Division operator

> **Note**
>
> Since we are dealing with integers, please apply integer division instead. For example, 17/3 == 5 (remainder is discarded). You can assume the result will not exceed the maximum value of integer datatype.

1.2 Main steps of your program

Please name your executable as "calculator". Here are the features that you need to implement in your program:

- Retrieving all ".usp" files

  The first step of your program is to retrieve the list of all the ".usp" files in the same directory as your program. You can use some functions from <dirent.h> and <string.h> to achieve this (see practical 5). If you want, you can store this filename in an array of strings.

- Creating sufficient pipes

  Once you obtain the file list, you need to create the same amount of pipes as the amount of the ".usp" files. You can create an array of pipe variables for this task. You may use malloc(). Please do error/fail checking for pipe().

- Creating child processes

  Your program will create one child process for each ".usp" file. For example, if there are 5 ".usp" files, the parent process will create 5 child processes. Please do error/fail checking for fork().

- Parent process responsibilities
  - As the parent process, it has to maintain all the pipes. Each pipe is used as a two-way communication to each child process.
  - Once a child process is created, one ".usp" filename is passed to the child process via the pipe with write() system call.
  - After passing all the filenames to all child processes, the parent process will use read() system call on each pipe to obtain the result from each child process. All these result will be written to a single file called "result.txt" by **appending**. One result should

be written on a separate line. The order of the result writing does not matter as long as all the results present in the correct format. (For example, you can write the result from child 2 before child 1 if you want.) When we test your program, we will remove existing "result.txt" so the program will start with fresh state.

- Parent process then need to use wait() system call to handle ALL child processes correctly to prevent them becoming zombie processes.
- Lastly, parent process release all necessary resources used in the program. (e.g close() and free())

- Child process responsibilities
  - As the child process, it only need to maintain its corresponding pipe. For example, child 1 uses pipe[0], child 2 uses pipe[1] and so on. Therefore, each child process need to close all other irrelevant pipes.
  - After closing other pipes, each child process call read() system call on the pipe to receive the filename.
  - Child process then opens the file with open() system call (and close it later with close() system call). The child process need to read the unique ID in the file and then solve the mathematical operation from the file. Library call atoi() may be useful here.
  - Once the mathematical operation is solved, the child process need to write back to parent in the string format of "<#ID>: <result>". For example, if the file contains unique ID "#00152724" and the mathematical operation "39 / 5", the child process will write the string "#00152724:7" to the parent process. Library call itoa() and strcat() may be useful here.
  - Lastly, the child process release the necessary resources (e.g close() and free()) and can end the process normally. The parent process will wait() on each children later on.

Note

Keep in mind that read() system call return the amount of successful byte read. This information can be used to know the length of the received data.

1.3 Prohibited Library Calls

Since you need to use **open(), close(), read() and write()** system calls for this assignment, any related library calls are NOT allowed. This includes some functions such as fopen(), fclose(), fprintf(), fscanf(), fgetc(), fputc(), fgets(), fputs(), etc (basically library calls related to file IO). If you use any of these system calls, that task will not be marked. If you need to print something to the terminal for debugging purposes OR just to inform user on the progress (e.g "Child 1 complete"), you can use printf() function.

1.4 Allowed Library Calls

Anything that is NOT related to reading or writing to files are allowed. For example, malloc(), calloc(), atoi(), itoa(), opendir(), readdir(), and anything from <string.h> are allowed.

2 Marking Criteria

Your work will be marked out of 50 marks, as follows:

1. Retrieving ".usp" files list
    - **5 marks** — Using functions from <dirent.h> to retrieve file name list in the same directory.
    - **5 marks** — Able to filter in the ".usp" files and count them correctly.
2. Creating pipe and child processes correctly
    - **5 marks** — Pipe are created correctly with right amount.
    - **5 marks** — Child processes are fork()-ed correctly for each ".usp" file.
3. Parent process behaviour
    - **2 marks** — Pass each ".usp" filename to each child process via correct pipe with write() system call.
    - **2 marks** — Result from each child process is received correctly with read() system call.
    - **2 marks** — File "result.txt" is opened correctly for appending.
    - **2 marks** — All results from child processes are written correctly to file "result.txt" with write() system call.
    - **2 marks** — Wait() for each child process.

4. Child process behaviour
    - **2 marks** — Each child process close the irrelevant pipes.
    - **2 marks** — Filename is received correctly via pipe with read() system call.
    - **4 marks** — The ".usp" file is opened correctly for reading. The unique ID is retrieved, and the mathematical operation is evaluated correctly.
    - **2 marks** — Pass the result back to the parent process with correct format with write() system call.
5. Code Quality
    - **4 marks** – Good program readability/commenting.
    - **3 marks** – Error/fail checking for pipe() and fork().
    - **3 marks** – Releasing resources properly. (close() and free())

3 Submission

Submit all the necessary source codes, makefile, and a signed declaration of originality, to the appropriate area on Blackboard. Include everything in a single .zip file. You do not need to include any compiled code. Keep in mind that we will test your program with our own ".usp" files. Please name your ZIP file with the following format:

student-ID_Full-name_USP-2023.zip

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

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 academicintegrity.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.