COMP 2560 Winter 2024—Assignment 3

Due: 11: 59 PM, Mar. 3

Question 1

In the code “abort.c” (posted on the course website), it uses hard-coded bit-manipulation to access the child process termination status. Rewrite this code using the bit-manipulation macros discussed in the lecture.

Question 2

Using Unix system calls, fork(), wait(), read() and write(), write a C program for integer- basic Arithmetic (+, - , *, /). Your program should follow the sequential steps, given below.

- Prompts the message ”This program makes simple arithmetic”,
- Gets in an infinite loop then
  1. Writes the message “Enter an arithmetic statement, e.g., 34 + 132 > ”,
  2. Reads the whole input line,
  3. Forks and
     - the parent writes the message “Created a child to make your operation, waiting” then calls wait() to wait for its child.
     - the child process calls the function childFunction(char *) and never returns.

4. The child, through childFunction(char *line),
    - writes the message “I am a child working for my parent”
    - uses *sscanf()* to convert the input line into an integer, a character and an integer, respectively.
    - in case of a wrong statement, the child calls exit(50)
    - in case of division by zero, the child calls exit(100)
    - in case of a wrong op the child calls exit(200)
    - otherwise, it performs the appropriate arithmetic operation,
    - uses sprint() to create an output buffer consisting of “n1 op n2 = result”,
    - writes the output buffer to the screen
    - calls exit(0)
5. Once the child terminates, the parent checks the returned status value using bit-manipulation macros discussed in class and if it is 50, 100 or 200, writes “Wrong statement”, “Division by zero” or “Wrong operator”, respectively.
6. The parent goes back to step 1.

Important:

- All reads/writes must be done using read()/write(), even if it is to write to the console or read from the keyboard, unless otherwise specified.
- You can use the returned value of sscanf() for detecting a “Wrong statement” (sample code “ReadInt.c” shows how to use sscanf(..), or google online to see how to use sscanf())

Question 3

Write a C program to execute multiple Unix commands in parallel.

- The number of Unix commands to be executed is not fixed.
- There is no communication among the Unix commands.
- The Unix commands are given as command line arguments.
- For simplicity, you can assume that each Unix command has exactly one argument except that the last one can have either no argument or one argument.

For example,

```
> miniminishell cat openfile.c ls –t ps
```

includes three Unix commands: cat with one argument openfile.c, ls with one argument –t, and ps with no argument.

For each Unix command, use a separate child process to execute it. You need to print out each process's id.

Observe how the output from each child process is displayed. Are the outputs from all the child processes displayed in a certain fixed order or intermingled? In case you are interested and thinking about how multiple processes share the terminal window, you can refer to Ch.9.5.Ch.9.6 (not required by this course though).

Submission Requirements

For all questions, you need to submit your source code. Also, use the script command with the timing option to record you compiling and running your program and submit the script file and timing file for each question as well. Record a short video with time stamps for each question to explain your code design and compile and run your code. Please properly name all the submitted files.