Lab and Homework #2

Introduction to Operating Systems CS-UY 3224 | CS-UY 3224G

Mirna Džamonja, email md5961@nyu.edu

Due date for the Homework problems : September 18th, 2023 by 5 PM
Please hand in through the *Assignments* option on *Brightspace*.

Question 1: *A bit of commands in Unix/Linux.*

Please work on this question from the Terminal.

- Make a directory `Lab2` in your home directory and work from there.
- Read the manual page for the command `tty`.
- Display on the terminal the name of its absolute path.
- Display the permissions of the absolute path of the terminal
- Open another terminal and give it a command which will make sure that the word 'hello' appears on the first terminal you opened.

To do at home and hand in: What is the command you typed for the last item?

Question 2: *Creation of processes,* `fork` *and* `wait`. In this exercise you get some practice reading a code without testing it.

What does the following program do ? Please try to answer the question without running the code, and then run it (it is in the file `ex1.c` on the page of *Lab 2* on *Brightspace*) to test your answer.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
  pid_t r;
  switch (fork()) {
    case  -1:
      perror("fork");
      exit(1);
    case 0:
      write(STDOUT_FILENO, "a", 1);
      exit(0);
    default:
      write(STDOUT_FILENO, "b", 1);
      switch (r = fork()) {
        case  -1:
          perror("fork");
          exit(1);
        case 0:
          write(STDOUT_FILENO, "c", 1);
```

```
            exit(0);
          default:
            waitpid(r, NULL, 0);
            write(STDOUT_FILENO, "d", 1);
           printf("\n");
            exit(0);
      }
    }
}
```

To do at home and hand in: Explain the difference between a deterministic and a nondeterministic program. In this program, which outputs are possible ? Which part of the output is deterministic and which is not ? In practice, when you run the program many times, do you notice a pattern ? (Hint: most often you do, but it is a bias of the scheduler rather than an inherent feature of what you wrote.)

Question 2: *Writing process-creation programmes*

1. Write a C-program where a parent process crates 5 children.
2. Write a C-program where a parent process crates k children, where k is an integer variable taken as a user input from the screen. Save the program as `ex2i.c`. Run the program a few times and observe a non-determinism, appearing for example in the form of an output like this:

   ```
   [son] pid 8851 from [parent] pid 8842
   [son] pid 8852 from [parent] pid 1
   ```
3. Modify `ex2i.c` so that the parent waits for all children to finish before finishing itself. Notice that program has become deterministic.
4. Modify the program from Question 2.(3) so that the last process counts the number of words in the file `ex2i.c`.

   Hint: Consider the code for Chapter 5 of OSTEP, we have done a similar problem in the class.
5. Write a program in which an integer variable k taken as a user input from the screen is given and then a parent process creates a child, who creates a child, and so on, k times.

Question 3: *Makefile*

Write a makefile and a README fie that you can use to turn in the programs from Question 2 in the form of a folder which I can compile and read all the files necessary by using `make`. Make sure that the file containing the output of the word count from 2(4) is displayed.

To do at home and hand in: The folder created in Question 3, including the C-code and executables for Questions 2 along with a makefile and README files from Question 3.