CSC 412 – Operating Systems

Lab Session 03, Spring 2022

Monday, February 28th and Wednesday, March 2nd, 2022

What this Lab Session Is About

This lab is about process creation using `fork()` and function from the `execXY()` family.

1 The Handout Code

1.1 Build, don't look

This time, we provide a large code handout. The good news is that you don't need to look at most of it. There is a script to build the different executables that we are going to use in this lab.

There is also startup code for the first version of our "dispacher" program.

1.2 What you just built

You built two small utility programs that perform elementary tasks on images in the uncompressed TGA file format[1] for flipping an image vertically and horizontally.

1.3 How to use them

Here is the basic syntax for the use of these utilities (there are more complex variations, but for this lab, these will suffice):

- `gray <image file path>` produced a gray version of the input image;
- `flipV <image file path>` flips the image about its horizontal middle axis (vertical flip);
- `flipH <image file path>` flips the image about its vertical middle axis (horizontal flip).

[1] This format is completely outdated for everyday use, but it is very convenient for assignments, because the code to read and write images in this format is tiny and completely cross-platform.

2 What to do

2.1 Version 1: No `fork`

The purpose of this version is simply to get a reference output and develop the required utility functions without dealing with the additional complexity (and associated points of failure) of process creation through `fork` and `exec`.

The `system()` function

For this purpose, we are going to use the `system()` function call (defined in `stdlib.h`), which lets one execute system commands within a C/C++ program. For example, the call

```
system("ls -l");
```

will print out to the standard output in "long form" the list of all files in the current working directory.

2.2 Task 2: Now We Get Some Action: `fork()` and `execXY()`

Note: I always use the generic term `execXY()` to refer to the family of functions `execve()`, `execlp()`, `execvp()`, etc. that flush the code (and entire partition) of the current process (generally inherited from their parent process by a `fork()`) to load and start executing a new program.

2.2.1 What we are going to do

In this version, the child processes are going to perform an `execXY()` to flush the code and data of the parent process and start running the code of one of the four image processing utilities. The basic idea is that the child process looks at the command string it got, and in particular at the first "word" in that string, which is the name of the executable to launch.

I remind you that there is no time for data validation in labs (unless some day we have a lab explicitly and exclusively about data validation). We have to assume that the input is valid, because there is not time to do checking. So here, the first work of a `TASK_LIST` entry can only be one of `flipV`, or `flipH`.

2.2.2 Version2a

I would suggest this intermediate step. For each of the four executables, hard-code a valid `execXY` command, to make sure that you actually launch the proper executable and that the command's arguments are passed along as you expect.

2.2.3 Version 2b

Now, you want to use the string you got from the `TASK_LIST` to produce a valid `execXY` command.