Homework #8

Physics 129 Spring 2022

Problems due **Saturday, May 21, at 11:55 P.M.**

Please read the homework guidelines handout on the course web page.

Before attempting this assignment, ensure your RPi is connected to the Internet, then run the `update_physrpi` script.

Better answers and code will get better grades.

Reading

- Complete by **Monday, May 23**

- Read sections 6.2–6.3 and 6.5–6.9 in K&N.

Problems

1. **Final Project.** **For this problem only, the due date is Tuesday, June 7, at 11:55 P.M.** Your project should include:

   a. any Python code you have written for the project

   b. a pdf file produced with LaTeX that includes:

      i. a project title

      ii. your name

      iii. instructions for installing any necessary software that is not present on the standard Raspberry Pi updated for Physics 129

      iv. if relevant, a list of any external hardware needed to run your project, including make, model, and place of purchase for each item

      v. a brief but clear and specific description of what your project is supposed to do and how it works. This should be a maximum of one page long in 12-point, normally spaced text with normal margins.

      vi. your results. Include here any necessary instructions for how to run your software to get the output you describe.

This section is expected to be more extensive if the instructors will not be able to run your code and see the program output for themselves (for example, if you used external hardware). You may reference external audio or video files here. If you do, include them in the tar file. Images and plots should be incorporated in the LaTeX document. This section should not exceed three pages unless you have a really good reason, for example a set of images you'd like to display at full-page size.

For this problem only, **email a tar file containing your project code and all associated files to Prof. Lipman and the TAs by the due date.** Both the tar file and the directory it unpacks into should be named `<lastname>_project`. For example, if your name is Enrico Fermi, you should turn in a file called `fermi_project.tar.gz` which unpacks into a directory called `fermi_project/`. Do not include anything for this problem when you turn in the rest of this homework set.

2. **Polynomial Fits.** Write a program that generates a user-specified number N of uniformly distributed random points on a region of the x-y plane, with x and y both running from 0–100. Find fits to the set of points using polynomials of degree 1 (a line), $N - 3$, and $N - 1$. Turn in an EPS plot of the specified region of the x-y plane showing the set of points and the fit curves, in four different colors. Your program may limit the value of N to some reasonable range, and you may use NumPy functions to do the fitting.

3. **Monte Carlo Circle.**

   a. Write a program that generates a user-specified number N of uniformly distributed random points on a region of the x-y plane with x and y both running from 0–2. By counting the number of points lying within the circle of radius 1 centered at $(x, y) = (1, 1)$, determine the area of this circle and the corresponding value of π.

   b. Write a second program to plot the fractional error in the determined value of π as a function of N. Turn in the plot as an EPS file.

4. **`fork()` and `execv()`.**

   a. From an interactive Python prompt, use `os.execv()` to execute the `ls` command. Explain what you see after the command has finished running.

   b. Write a program that starts counting from 1, printing the next number every half second. Each time the program reaches a multiple of 10, it should announce that it is about to fork, and then fork. The child process should then announce that it is about to execute `ls`, and then execute `ls -l` using `os.execv()`.

Hints: See `fork_example.py` .
Try `man exec` if you don't understand what `os.execv()` is doing.
The first entry in an argument list is the name of the program being executed. Try running this shell script:

```
#!/bin/sh

echo $0 $1 $2 $3
```

with several different sets of arguments. Or print `sys.argv` from a Python script.

5. **Threaded Stripchart.** Modify your stripchart program from the previous homework set so that it displays the value of a global variable. Have your program spawn a thread that continuously prompts the user for a new value and places it in the global variable. The chart should then display the most recent number entered. Make sure your program can deal gracefully with non-numeric input.

   Hints: See `thread_example.py` and `error_handling.py` .
   You will have to declare the global variable using the `global` keyword in your thread function so that the input value is visible to the rest of the program.

6. **System Call Tracing.** Using the `trp` wrapper script for `strace`, run your program from the **`fork()` and `execv()`** problem. Kill the program after the first time it forks and executes `ls`.

   In the output containing the system calls (a file called `tr`), locate and explain in your answer file what you see in the following:

   a. the line where the process forks, including 5 lines before and 5 lines after
   b. the line where the child process exits after running `ls`, including the 5 preceding lines
   c. The line where the parent process receives the signal that kills it, including the 5 preceding lines.

   Hint: `os.fork()` is implemented using the `clone()` system call.