Lab Worksheet 01 - Processes

Exercise 1: First process creations

Write two programs that create `N_CHILDREN` child processes.

1. The first program uses the iterative paradigm (loop)
2. The program uses recursion.

Note: This not a question about process chains: parent creates a child, child creates a grandchild, ... **All created processes must have the same parent**.

Exercise 2: First parent/child synchronizations

Repeat exercise 1, but this time use `'wait'` primitives so that the initial process waits for:

1. the termination of exactly one child (any child)
2. the termination of the last child it created (that one only)
3. the termination of all its children

Exercise 3: Parallel compilation

Write a C program whose arguments are a list of .c source file paths, compiles each of them separately and simultaneously, and then edits the links to produce an executable. This program must create a child process for each file in the list. Each child runs the gcc -c program on the file it gets assigned. The parent awaits the termination of all its children; if all children processes terminate correctly, the parent performs the link edition by running gcc on all the .o files produced by its children.

Exercise 4: Spy shell

Write a program 'spy' which acts as a wrapper to the `shell` program.
Once launched, 'spy' reads every command entered by the user (download the skeleton code "spy-incomplete.c" (https://newclasses.nyu.edu/access/content/group/51ce8755-5381-4dd5-bae5-8ae3b3c862d0/Worksheets/Skeleton-Code/spy-incomplete.c) for code that reads user input from the terminal) and executes this command with an exec call.
Use 'execlp' for commands entered with exactly one argument, 'execvp' for all the others.