CS3411 Project 4 - Husky Script : Script a Program

Due : November 12, 2024, Midnight

In this project, you will be developing a *script* program called *husky script* (hscript). Functioning similarly to the Unix `script` command, hscript forks a child and executes a program whose name is passed as an argument alongside with its own arguments and intercepts all output from the standard output and standard error of that program, as well as anything passed onto that programs's standard input. The program hence logs all input/output traffic to and from the executed program. The syntax for the execution of the *hscript* program is given by:

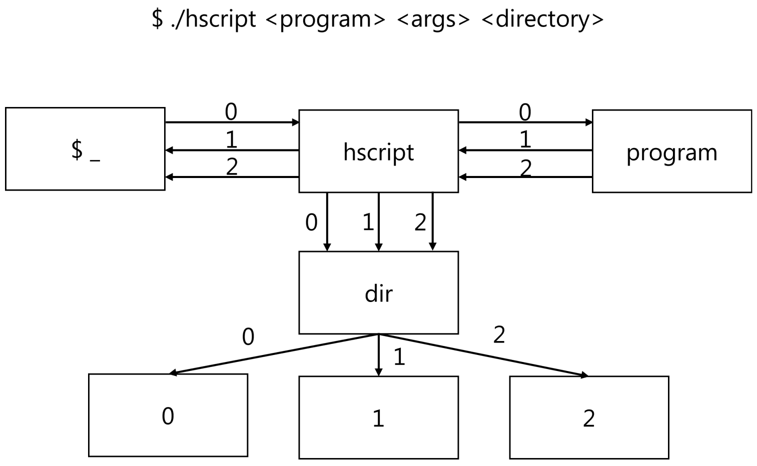
```
hscript <program name> <arguments> <directory>
```

where the `<program name>` is the full path to the to be executed program, `<arguments>` are the arguments which need to be passed to that program. The last argument `<directory>` is a simple name of the directory under which the traffic is placed. For example the command:

```
hscript /usr/bin/ls /home/myuserid blah
```

executes /usr/bin/ls with the argument /home/myuserid and creates three files titled, 0, 1, and 2 under the directory `blah`. File 0 should contain any typed input to the program (in this case none), 1 should contain the data output by the program to standard output and 2 should contain the data output by the program to standard error. The program should also send any data output by the program through its standard output and error to the corresponding descriptor of the hscript program (i.e., standard output of the executed program goes to the standard output of the hscript program). This ensures that hscript program can also be input/output directed if necessary.

Architecture Overview: Below is a high level overview of the design indicating how input/output is being passed and logged.



Requirements

1. If the directory indicated by the directory argument does not exists, the program creates it.
2. If there is a file with the indicated directory name, the program bails out.
3. The program should not *output direct* the executed program to the specified files; instead, hscript program itself should capture the data, read it and write it both to its standard output (error) to the appropriate file 1 (or 0) in the indicated directory.
4. Data capturing MUST be done using pipes.
5. The program should never block. Therefore, it is required that it uses select kernel call.

Your submission should include a tarred gzipped copy of :

1. Your program source file.
2. A Makefile.

Your makefile should include "all", and "clean" labels. When I type `make` or `make all` in the directory containing your recovered submission, a binary file named `hscript` should be created. Typing `make clean` should remove all the object files and the created binary `hscript`.

Ground Rules and Restrictions This assignment may be provided with updates as the project goes on. Make sure that you attend the classes and watch the class mailing list.
You may not borrow or reuse any code from other resources, and all the code must be your own. In addition, the following rules apply:

- You may discuss the program with others.
- You may not bring any printed/written material into the discussion with you. (You may not show anyone your code; you may not view the code of anyone else. This includes others both enrolled in the course and others not enrolled in the course.)
- You may generate written material during the discussion, but it must be destroyed immediately afterwards. Don't carry anything away from the discussion.
- If you are in doubt about the acceptability of any collaboration activity, I expect you to check with me before engaging in the activity.