Oregon State University
College of Engineering

Lab 6: Big Benny

Please **read this entire assignment**, **every word**, before you start working on the code. There might be some things in here that make it easier to complete. This lab consists of multiple parts. **All Parts of this lab are July 27th by midnight.** Submit a single gzipped tar file to **TEACH**. The single gzipped tar file should contain the all source files (C source [*.c and *.h] and the `Makefile`). **You must have a single `Makefile` to build all portions of this assignment. If your `Makefile` does not build a portion, then it is a zero for that portion of the assignment.** Do yourself a favor, write your `Makefile` first.

Part 1 – Big Benny (250 points)

This is Part 2 of a 2-week assignment. That will be completed in 2 stages: Stage 1 and Stage 2.

- Stage 1 is/was due on July 20th. Stage 1 should be fully reusable as the beginning of Stage 2.
- **Stage 2 is due on July 27th. Stage 2 is the more challenging portion of the assignment.**

In this portion assignment (Stage 2), you will be *enhancing* your existing baby Benny UNIX/Linux shell, `BennySh`.

Stage 2 requirements:

1. **All the capabilities, requirements, and function of the Stage 1 portion of the assignment**.
2. External commands can be given as a **pipeline**. For example:

```
ls -la -F | wc -l
ps -elf | grep http
```

3. Pipelined commands can be of any length.

```
ps -elf | grep chaneyr | tr r j
ps -elf | grep chaneyr | tr r j | awk {printf("%s\n",$3);} > file1.txt
```

4. Redirection of `stdin` is only valid for the first command in a pipeline.
   4.1. E.g.: `wc < file1 | wc -l`
5. Redirection of `stdout` is only valid in the last command for a pipeline.
   5.1. E.g.: `ls -l | wc > file1`
6. There will always be at least 1 whitespace character on each side of a pipe on the user's command line (as shown in the examples above).
7. You will not be placing any child process as a background processing command (no use of the `&` on a command line).
8. You must not have any orphans or zombie processes.

Special note: As before, I know there is a web site out there called Write a Shell in C. It is a crappy web site. You can do better. I do not like how it implements commands. I discourage you from leaning on that code to complete this project. I've seen students use that web site and have it **add weeks** to their development time and not be able to reuse it in the second half of the shell assignment. Remember, if your assignment is more than 2 days late, it becomes a zero.

Other Requirements

You must have a single `Makefile` that compiles your program (`BennySh`). If you do not have a `Makefile` that builds our program, it will put a major dent in your grade for the assignment. Of course, all programs must be written in C. You should have no memory leaks. Your code must compile without any errors or warning from `gcc`. I will deduct 20% for each warning your code produces when compiled. Do not adorn your calls to `gcc` with any `-std=...` options.

Your `Makefile` must include the following targets:

- `all` – should build all out-of-date programs and prerequisites.
- `clean` – clean up the compiled files and editor chaff.
- `BennySh` – linking the part 1 program, rebuilding the `BennySh.o` file, if necessary
- `BennySh.o` – build the `BennySh.o` file, if necessary
- if your `BennySh` is built from multiple C files, you'll need to have additional targets to support the additional C files.

You must use the following `gcc` command line options in your `Makefile` when compiling your code.

```
-Wall
-Wshadow
-Wunreachable-code
-Wredundant-decls
-Wmissing-declarations
-Wold-style-definition
-Wmissing-prototypes
-Wdeclaration-after-statement
-Wno-return-local-addr
-Wuninitialized
```

The program must be named **`BennySh`**.

Final note

The labs in this course are intended to give you basic skills. In later labs, we will *assume* that you have **mastered** the skills introduced in earlier labs. We will assume you know how to do these things because you learned it in this lab. **If you don't understand, ask questions.**