Project Objectives and Test Plan

Interactive Mode

Shell will look like this:

```
user@ubuntu:/# ./hsh
($) /bin/ls
hsh main.c shell.c
($)
($) exit
user@ubuntu:/#
```

Non-interactive Mode

```
user@ubuntu:/# echo "/bin/ls" | ./hsh
hsh main.c shell.c test_ls_2

user@ubuntu:/# cat test_ls_2
/bin/ls
/bin/ls

user@ubuntu:/# cat test_ls_2 | ./hsh
hsh main.c shell.c test_ls_2
hsh main.c shell.c test_ls_2

user@ubuntu:/#
```

Error Handling

- program **must have the exact same output** as `sh` (`/bin/sh`) as well as the exact same error output.
- name of program must be equivalent to argv[0]

```
/bin/sh: 1: qwerty: not found
```

2. Simple shell 0.1

Usage: `simple_shell`

- [ ] When compiling use `simple_shell` (as opposed to `hsh`)

Basic Requirements:

- [ ] Display prompt "$" waits for user to type a command
- [ ] Command line always ends with a new line
- [ ] Prompt is displayed again after each command has been executed
- [ ] Command lines do NOT include - semicolons, pipes or redirections (or any other advanced features)
- [ ] Command lines are only made up of one word - no arguments are passed to programs
- [ ] If executable is not found print an error message and display prompt again
- [ ] Handle errors
- [ ] Handle EOF condition (CTRL+D)
- [ ] Pass `environ` to `execve`

NOT Required:

- [ ] Use PATH
- [ ] Implement built-ins
- [ ] Handle special characters such as `", ', , \, *, &, #`
- [ ] Moving the cursor
- [ ] Handle commands with arguments

3. Simple shell 0.2

Requirement

- [ ] Handle command lines with arguments

4. Simple shell 0.3

Requirements:

- [ ] Handle PATH -> handled by path.c
- [ ] `fork` must not be called if command doesn't exist

5. Simple shell 0.4

Requirements:

- [ ] Implement the `exit` built-in that exits the shell
- [ ] Usage: `exit`
- [ ] No need to handle any argument to built-in `exit`

6. Simple shell 1.0

- [ ] Implement the env built-in - that prints the current environment