

Problem 1 -- The Shell

In this assignment, you will write an extremely simplistic UNIX shell which is capable of launching one program at a time, with arguments, waiting for and reporting the exit status and resource usage statistics.

Your shell shall accept lines of input from standard input until EOF. Don't worry about issuing a prompt, command-line editing, etc. Each line is a command to be executed. The shell should fork to create a new process in which to run the command, set up any requested I/O redirection (see below), exec the program, wait for and report the exit status, and report the real, user and system time consumed by the command (a la `time`). If you are adventurous, you might examine other members of the `rusage` structure and see how they are reported on different operating systems.

You may assume that each command is formatted as follows:

```
command {argument {argument...} } {redirection_operation {redirection_operation...}}
```

This is an extreme simplification of shell syntax, but this is after all a course in operating systems, not compilers. The above optional arguments and operators are whitespace-delimited, i.e. they are separated by one or more tabs or spaces. This will simplify parsing and you can use `strtok` to break up the input line into its components. The real shell accepts `command argument>output` as well, but you don't have to parse that if you don't want to. **A line that begins with the # character is a comment and should be ignored.** Report any errors related to command launching but, just like the real shell, do not exit the entire shell on error; go to the next command.

I/O redirection

Support the follow redirection operations (note that pipes are not required since we haven't talked about them yet):

<code><filename</code>	Open filename and redirect stdin
<code>>filename</code>	Open/Create/Truncate filename and redirect stdout
<code>2>filename</code>	Open/Create/Truncate filename and redirect stderr
<code>>>filename</code>	Open/Create/Append filename and redirect stdout
<code>2>>filename</code>	Open/Create/Append filename and redirect stderr

Note that a given command launch can have 0, 1, 2 or 3 possible redirection operators. A failure to establish any of the requested I/O redirections should result in an error message and the command should not be launched.

Clean File Descriptor Environment

Your shell should fork and exec the command with a standard, clean file descriptor environment. Only file descriptors 0, 1 and 2 should be open, possibly redirected as above. There should be no "dangling" file descriptors.

Example

```
$ mysh
ls -l >ls.out
Executing command ls with arguments "-l"
Command returned with return code 0,
consuming 0.005 real seconds, 0.002 user, 0.001 system
end of file
$ cat ls.out
mysh.c
mysh
$
```

Shell Scripts

Your shell must also support being invoked as a script interpreter:

```
mysh /tmp/script
```

If provided with an argument as above, your shell should open that and execute each line as a command. Otherwise, your shell should read commands from standard input until EOF.

Test this feature by creating an executable script file that calls your shell as the interpreter using the `#!` notation.

Exit status

The exit status of your shell itself should be 0 if no errors were encountered and all requested commands were launched, otherwise it should return a non-zero status.

Your submission should include: source code, "screenshot" demonstrating your shell working in an interactive case (commands from keyboard), screenshot showing your shell working as a script interpreter, including source code of shell script used to test this feature. You should also demonstrate I/O redirection.