- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Assignment 9: Write Your Own Shell

Due 11:59 p.m., Monday, December 8

> **IMPORTANT!** This is an individual assignment. You may discuss broad issues of interpretation, understanding, and general approaches to a solution. However, the development of a specific solution or program code must be your own work. The assignment is expected to be entirely your own, designed and coded by you alone. If you need assistance, please consult your instructor or the TAs. Be sure to read the specific policies outlined in the **Academic Honesty in Computing** section.

The purpose of this assignment is to help you become more familiar with the concepts of process control in Linux-based systems. You'll accomplish this by writing a program called `tsh.c` ("Trinity Shell") that supports basic job control.

General Overview of Linux Shells

A *shell* is an interactive command-line interpreter that executes programs on behalf of the user. A typical shell:

- Displays a prompt,
- Waits for a command line from standard input,
- Parses and interprets the input,
- Executes the requested action.

If the command is built-in (e.g., `cd`, `echo`, `pwd`; see here for a complete list), the shell handles it directly in the current process. Otherwise, the shell:

1. Forks a new child process.
2. In the child, loads and runs the specified program.
3. Waits for the child to complete.

The set of child processes launched from a single command line is referred to as a *job*. While many jobs can consist of multiple processes connected by *pipes*, for this assignment you'll only implement single-process jobs.

The tsh Specification

Your `tsh` shell should implement the following features:

- The shell prompt should be:

`tsh>`

- The command line entered by the user should consist of a command name followed by zero or more arguments, all separated by one or more spaces.
- If the command is a built-in command, `tsh` should execute it immediately in the current process and then prompt the user again for the next command.
- The only built-in command supported by `tsh` is `quit`, which exits the shell.
- If the command is not built-in, `tsh` should treat the name as the path to an executable file:
  - It should `fork()` a child process,
  - The child should then use `execvp()` to load and execute the command,
  - The parent should wait for the child to complete by calling either `wait()` or `waitpid()`.

Programming Notes

- Use the `strtok()` function to break the input command line into tokens (words). Each token is separated by whitespace.
- To run an external command, use the `execvp()` function inside a child process.:

  ```
  int execvp(const char *command, char *argv[]);
  ```

  Here, `command` is the name of the executable, and `argv` is an array of strings (arguments), terminated by a `NULL` pointer.

- Compiler your shell with:

  ```
  $ gcc -o tsh tsh.c
  ```

- Run your shell with:

  ```
  $ ./tsh
  ```

- The `fork()` system call can fail (e.g., if the system runs out of memory or process IDs). You must check its return value: if `fork()` returns -1, your shell should print an error message (e.g., using `perror("fork")`) and continue the loop to prompt the user again.
- The `execvp()` function only returns if it fails (e.g., if the command path is invalid or the file is not executable). If it fails in the child process, you must immediately exit the child process to prevent it from continuing.
- The parent shell must call `wait()` to reclaim the resources of the terminated child. You should check the return value of `wait()` or `waitpid()` to ensure it was successful. Both functions return -1 on failure.

Sample Command Tests

```
tsh> echo Hello, world!
Hello, world!
tsh> ls
file1 file2
...
tsh> ps
tsh> ls -l
tsh> pwd
tsh> whoami
tsh> date
tsh> uname -a
tsh> id
tsh> hostname
tsh> quit
```

Handin

When completed, upload your C source (`tsh.c`) to the course website.

- **Welcome: Sean**
  - LogOut