Assignment 2: A RUSTy Shell

Dr. William Kreahling

March 17, 2021

1 Overview

A *shell* is a program that prompts users for commands to execute, reads those commands, and executes them. Every time you log into a Unix/Linux machine, you interact with a shell; on agora that shell is typically Bash (`/bin/bash`). There are, however, other shells including Bourne shell (`sh`), Korn Shell (`ksh`), C Shell (`csh`), TENEX C Shell (`tcsh`), and others. Each of these shells offers a specific set of features (e.g., tab completion, aliases, command history, functions) and a scripting language for automating tasks. In this assignment we will take a look at how shells read commands, start new programs, redirect input and output, and create pipelines between processes.

2 Background

This assignment uses some applications with which you might not be familiar; specifically `pest`. Pest is an elegant parser – a tool that reads a specification (here, `grammer.pest`) to recognize the specified tokens. This type of tool is frequently used in compiler implementation; however, here we're using to help us easily parse commands that are read by our shell. It is not important that you understand how this crate works. If you're interested there are resources out there.

3 Details

You must fill in the implementation of various parts of the files `main.rs`, `utils.rs`, `builtin.rs`, `history.rs`, and `redirect.rs`. Additional information about each of the changes can be found in a "TODO" comment in the source code at the appropriate location in the code. I *strongly* suggest you implement and test each of these changes one at a time. For full credit there should be no unsafe code.

4 Caution

Please start **immediately**; this may be a time-consuming assignment. I suspect that you are not accustomed to reading someone else's code. For this assignment you will need to do so, and you will need to understand how to make useful changes to that code. While the individual pieces do not involve large portions of code, understanding what is needed where may take some time.

5 Notes on Collaboration

You may work in teams of up to two on this assignment. Note that all members of a team will receive the same grade on the assignment.

6 Hand-In Instructions

This assignment is due by 11:59 PM on Monday March 29th. A single version version of the assignment is due from each team. Submit all source files associated with the program, do not submit the target directory as part of your submission. To submit your files, use the *handin* command on agora. Handin works as follows:

```
handin.<course#>.<section#>  <assignment#>  <files>
```

Therefore, to submit this assignment, you must use the following command (assuming you have used tat to compress your directory (without the target) into rShell.tbz):

```
handin.370.1  2 rShell.tbz
```