Kishmish *in a nutshell*

Because everything happens for a raisin

Introduction

This is a stripped-down shell implemented in C, as part of the submission for Assignment 2 and 3 of Computer Systems Engineering-1 Course (Monsoon 2019).

Functionality and Limitations

Kishmish supports the following features:

- Built-in commands: `ls [al]`, `echo`, `pwd`, `cd`, `quit`, `history`, `pinfo`, `nightswatch`, `setenv`, `unsetenv`, `jobs`, `kjobs`, `overkill`, `fg`, `bg`, `cronjob`
- Process management including foreground and background processes and switching between them.
- Piping and redirection
- Signal handling for signals like `SIGINT` and `SIGTSTP`
- A prompt that shows the username, hostname and the current working directory
- Command history and `up` arrow command recall across sessions
- Appropriate error handling

Kishmish does not support the following yet:

- Glob expansion and `tab` completion
- Support for quoting and escape sequences
- Aliasing
- User profiles
- Environment variables

and maybe a lot more...

All the functionality required for the assignment has been implemented.

Make instructions

In the directory where the source files reside, execute:

```
$ make
$ ./kishmish
```

To clean up object and binary files, execute:

```
$ make clean
```

File-Wise Code Breakdown

- `shell.c` : Contains the REPL of the shell
- `utils.c` : Contains several utility functions such as those for string processing, `~` expansion, storing and retrieving session history, initialization and cleanup
- `prompt.c` : Contains the function responsible for generating and printing the shell prompt
- `parse.c` : Contains functions that parse and sanitize the entered commands and call appropriate functions
- `external.c` : Contains functions that handle the execution of external programs in the foreground and background
- `exit.c` : Contains functions that implement the `exit` command
- `pwd.c` : Contains functions that implement the `pwd` command
- `cd.c` : Contains functions that implement the `cd` command

- `ls.c` : Contains functions that implement the `ls` command
- `echo.c` : Contains functions that implement the `echo` command
- `pinfo.c` : Contains functions that implement the `pinfo` command
- `history.c` : Contains functions that implement the `history` command
- `nightswatch.c` : Contains functions that implement the `nightswatch` command
- `bg.c` : Contains functions that implement the `bg` command
- `cronjob.c` : Contains functions that implement the `cronjob` command
- `env.c` : Contains functions that implement the `setenv` and `unsetenv` commands
- `fg.c` : Contains functions that implement the `fg` command
- `jobs.c` : Contains functions that implement the `jobs` command
- `kjob.c` : Contains functions that implement the `kjob` command
- `overkill.c` : Contains functions that implement the `overkill` command
- `pipe.c` : Contains function that help in parsing and setting up command pipelines
- `process.c` : Contains utility functions for process management
- `recall.c` : Contains functions that implement `up` arrow command recall
- `redirection.c` : Contains functions that implement input and output redirection
- `signals.c` : Contains functions that handle signals sent to the shell
- `makefile` : Contains the `make` rules for compiling the files
- `README.md` : The file you are reading now
- `README.pdf` : Another file you could be reading now
- `LICENSE` : This project is open-sourced under an MIT license. Details in this file. WARNING: Legal language ahead.

A header file accompanies each of the C files listed above

Implementation Details

The UI and the Prompt

- Kishmish indicates that it is waiting for input by printing a prompt that shows the current username, hostname and working directory address.
- The working directory address is shown relative to the *home directory*, which is defined as the directory where the shell executable resides[1].

Input Parsing

- Kishmish accepts *bash-like* commands and is capable of executing a list of semi-colon-separated ones.
- Kishmish is capable of parsing space-separated arguments
- Kishmish tries to handle all reasonable input errors appropriately.

Command History

- Kishmish stores command history across sessions, capped at a maximum of 20 commands[2].
- Command recall on pressing the `Up` arrow key is supported up to 20 commands[3]. Kishmish does not currently support editing the recalled command before execution.
- This is achieved by storing history data in a file at the end of this session and reading it at the beginning of the next. During a session, history data is managed internally in an array. This is to reduce the number of file accesses and improve performance.
- The history file, `.kishmish_history.dat`, is a binary file that is created automatically in the home directory, if it doesn't already exist. Deletion of this file is harmless, except the loss of history data from the previous sessions.
- The history data is written to file only when the shell exits normally, that is on receiving an `EOF` signal (`Ctrl`+`D` on Linux-based systems) or upon use of the `quit` command.

Execution of External Programs and Process Management

- Kishmish is capable of launching external programs in the foreground as well as in the background.
- This is achieved by forking the shell process into two, and replacing the child process's program image with that of the desired program.

- Depending on whether the program was invoked in the foreground or in the background, the parent process does or does not wait for the child process to terminate, respectively.
- Invocation in the background is requested by adding an ampersand symbol (`&`) after the complete command. It may or may not be separated by whitespace. Any part of the command after the `&` symbol is ignored.
- Kishmish prints an alert if a background job terminates.
- Kishmish supports switching jobs between foreground and background execution using the `fg` , `bg` commands and the `SIGTSTP` signal (`Ctrl`+`Z`) on Linux based systems.
- This is achieved by maintaining an internal list of processes, managing processing groups and manipulating the foreground process group of the attached terminal.

Piping and Redirection

- Kishmish supports input and output redirection using `<` , `>` and `>>` .
- This is achieved by duplicating the file descriptors of the standard I/O files.
- Kishmish also supports pipelining of commands using `|` , achieved using the `pipe` syscall.

Built-in Commands

- All built-in commands listed above behave as expected, mimicking the behaviour of bash.
- The syntax of some peculiar commands is elaborated below:
- - `cronjob -c command -t wait -p time` : Repeatedly run `command` after every `wait` seconds for a total duration of `time` seconds in a subshell in the background. The subshell is however granted access to terminal I/O and runs in the same process group as Kishmish.
- - `nightswatch -n wait [dirty | interrupt]` : This command prints certain system information to the screen every `wait` seconds. The command runs in Kishmish itself, not in a separate subshell, hence it cannot be sent to the background and cannot be interrupted. This is by design, because the command needs to periodically print data to the terminal. It can be exited using the sequence `Q`+`Enter`. The information displayed corresponds to the amount of dirty memory in the system cache if `dirty` is selected, or the number of keyboard interrupts per CPU core if `interrupt` is selected.

Have a grape day!

Jivitesh Jain, 2018101092, UG-2, CSE

1: As per assignment requirements

2: As per assignment requirements

3: As per assignment requirements