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 of Computer Systems Engineering-1 Course (Monsoon 2019).

Functionality and Limitations

Kishmish supports the following features:

- Built-in commands: `ls [al]`, `echo`, `pwd`, `cd`, `exit`, `history`, `pinfo`, `nightswatch`
- Support for executing external programs in the foreground and background
- A prompt that shows the username, hostname and current working directory
- Command history across sessions
- Appropriate error handling

Kishmish does not support the following yet:

- Piping and redirection
- Signal and interrupt handling
- Glob expansion and `tab` completion
- Support for quoting and escape sequences
- Aliasing
- User profiles
- Environment variables
- Advanced process management, such as process suspension

and 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
- `makefile` : Contains the `make` rules for compiling the files
- `readme.md` : The file you are reading now

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].
- 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 `exit` command. Thus the use of `SIGINT` (Ctrl+C on Linux-based systems) to exit Kishmish should be avoided.

Execution of External Programs

- 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.
- The shell currently does not support commands like `fg` and `bg` that send processes to the foreground and to the background, however it does alert the user whenever a background process terminates.
- Programs that compete with the shell for I/O, such as the *Vi* and *Vim* text editors should not be invoked in the background[3].

Built-in Commands

- All built-in commands listed above behave as expected, mimicking the behaviour of bash.

Have a grape day!

Jivitesh Jain, 2018101092, UG-2, CSE

1: As per assignment requirements

2: As per assignment requirements

3: Such processes need to be suspended when in the background, and Kishmish doesn't currently support process suspension