Name: Abhinav Reddy Boddu

Roll Number: 2021101034

File structure

```
2021101034
├── commands
│   ├── commands.h
│   ├── bg.c
│   ├── cd.c
│   ├── discover.c
│   ├── echo.c
│   ├── fg.c
│   ├── history.c
│   ├── jobs.c
│   ├── ls.c
│   ├── pinfo.c
│   ├── pwd.c
│   └── sig.c
├── Helpers
│   ├── Helpers.h
│   ├── add_to_list.c
│   ├── check_invalid.c
│   ├── getinfo.c
│   ├── History.c               /* TO HANDLE HISTORY */
│   ├── interrupt_handlers.c
│   ├── ioredirect.c
│   ├── pipe.c
│   ├── process_creation.c
│   ├── splitter.c              /* split the command among cd,ls,echo,..... */
│   └── tokenize.c
├── io_module
│   ├── input.c                 /* Taking input ( raw mode ) and autocompletion
*/
│   ├── input.h
│   ├── print_error.c
│   ├── print_error.h
│   ├── prompt.c
│   └── prompt.h
├── Linked_list
│   └── ADT for Linked List
├── main.c
├── makefile
├── headers.h
├── README.md
└── README.pdf
```

folder structure:

> `commands:` contains implementations of built-in commands

> `Helpers:` contains some helper functions

> `Linked_list:` contains ADT for linked list (used in storing background process data)

> `io_module:` function for taking input, printing errors and prompt

To compile:

```
$ make
(or)
$ make main
```

To Run:

```
$ ./main
```

Assignment 3

> assuming precendence order is `';'` > `'&'` > `'|'` > (spaces and tabs) > (io redirection) . So tokenising accordingly.

1. Spec1:
   - Assuming that `<` , `>` , `>>` would always be followed by a file name (i.e commands like "`cat < >`" or "`cat <`" will give ambiguous results)
2. Spec2:
   - the commands when using pipe run one command after the other

     ```
     ~> sleep 5 | sleep 5
     ```

     takes 10 seconds to execute.
3. Spec4:
   - `bg` is having problems with "vi" / "vim" due to some unknown vi bugs.
4. Spec6:
   - Everything is implemented as mentioned in the PDF

Assignment 2

assumptions:

> assumed all lengths such as `Maximum input size`,`Maximum directory path length`,.... to be 1001

> assuming precendence order is `';'`>`'&'`>(spaces and tabs) . So tokenising accordingly.

> Please use a terminal window of sufficient width so that text wrat does not cause any problem

1. cd:
    - Not handling `~` when `~` is not at the start of the path (similar to bash).
    - maximum length of paths is `1001` chars
2. echo:
    - printed as is irrespective of Quotes and newlines (`",\n,'`).
    - handled space and '`\t`' characters similar to bash (replace multiple space / tabs with single space).
3. pwd:
    - Did not check for the number of arguments passed (since bash also does the same).
    - `absolute path`
4. ls:
    - implemented same as that of Bash.
    - assumed that there cant be paths like `/` , where there are some special files / folders which are not handlable.
    - if there are multiple files / folders as arguments , we printed in the same order as given to us (Bash prints for files first and directories next).
    - if the user name , or group name exceeds 15 characters , only the first 15 chars of their name are printed.
5. pinfo:
    - cannot read the executable path for certain processes like `systemd` (pid - 1) , where user does not have appropriate permissions.
6. History:
    - even if the input has only a combination of spaces and tabs , they are being added to history.
    - even if input is an invalid command it is being added to history.
    - refraing from adding to history only when the new input is exact same as old one (including space / tabs).
    - saving history in a hidden file `.shell_history.tmp`. any arbitrary changes made to this file while programme execution would not reflect and leads to unexpected errors.
7. discover:
    - assuming it can only be of the form:

      ```
      discover <directory to search for> <flags> <target name>
      ```

    - if no flags are given then considering it as `-d -f` only.
    - assuming that there cant be multiple directories in the arguments to search for (searched in only first appearing directory in the arguments).
    - target name must be in enclosed with `"`.
8. Background commands:
    - running `vi`/`vim` as a background process is creating problems due to some issues with inputs (stdin) on some machines. All other external commands work perfectly.
9. Foreground processes:
    - The time taken printed is the sum of individual parts of command

```
```

```
sleep 3 ; sleep 4
would print the time taken as 7s
```