ezsh User Manual

0. Contents

1. Dependencies

ezsh is dependant on:

- gcc version 7.3.0 or higher.
- GNU make version 4.1 or higher.
- tmux version 2.6 or higher.
- GNU readline version 7.0 or higher.
- ncurses version 6.1 or higher.
- TOIlet version 0.3 or higher.

2. Installation

2.1 Installing Compilers and Build Tools

Both the compiler `gcc` and build tool `make` usually come pre-installed with most Linux distributions. In the case they are not you can run:

- `sudo apt install gcc` to install the gcc compiler.
- `sudo apt install make` to install the make build tool.

2.2 Installing Shell Utilities

The shell utilities, that is, the dependencies which the shell requires to run, can be installed using the `requirements` shell script in `/code`. The dependencies installed by using this script are:

- tmux

- GNU readline
- ncurses
- TOIlet

To run the shell script you need to give it the correct permissions to allow execution. Using `chmod +x requirements` allows this action, and this script can be executed by running `./requirements`.

3. Build

Once all dependencies have been installed, you can build and run the project by running the `ezsh` script. This script takes care of all the setup for ezsh.

This script also needs to have the correct permissions to allow execution. Using `chmod +x ezsh` allows this action, and this script can be executed by running `./ezsh`.

4. Usage

4.1 Shell Layout

The above image shows the default layout of ezsh. There are three available panes in the above image.

The left-most pane will be referred to as the Explorer, the bottom-most pane will be referred to as the Messenger and the middle-most pane will be referred to as the Prompt.

All three of these panes are accesible to the user and may be swapped between by either clicking on them or holding `Alt` and one of the arrow keys.

Along with this, the size of each of these panes is adjustable, just simply click the edge of the pane and drag it to the desired position.

4.2 Explorer

- **Layout**

  The Explorer has a few components to it which allow it to function. First off is the list of files and directories in the current working directory. These items may be acted upon by the user.

  The top of the Explorer contains a command menu and the current command the user is executing on a given file or directory. Directories are coloured blue and files and coloured white.

  The bottom of the Explorer contains the number of files in the current working directory, the position of the user in respect to that and the full path of the current working directory.

  Finally below this, there is a flashing line of text which indicates to the user how they can access the help guide.

- **Navigation**

Use the `Up` and `Down` to navigate the current directory. Pressing `Enter` either opens the file in the users text editor, or opens that directory in the Explorer.

This action is indicated before it is taken in the top right hand side of the Explorer. This action will either indicated the file is to be opened or the directory is to be opened

- **Command Menu**

A sequence of commands which can be used to execute different operations is to be found at the top of the Explorer pane. These commands are:

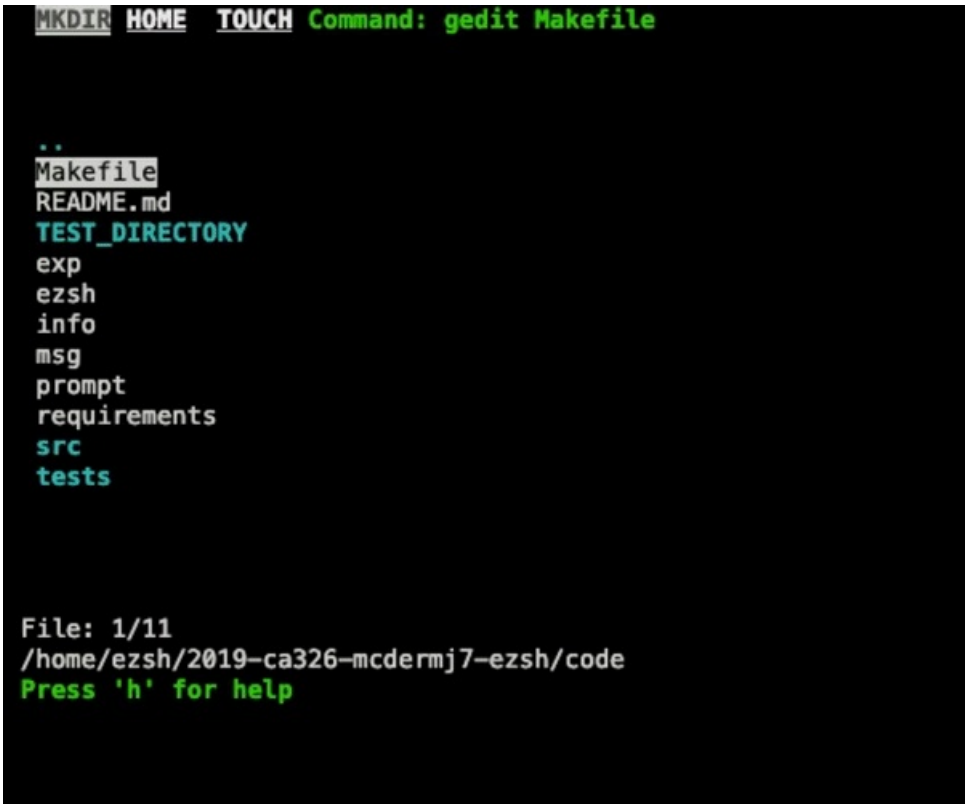
- **HOME** - this simple changes the user's directory to their home directory.
- **MKDIR** - this prompts the user for input and creates a new directory based on this input.
- **TOUCH** - this prompts the user for input and creates a new file based on this input.

Cycling between the commands in this menu is completed using the `Tab` key.

The execution of the desired shortcut is completed by pressing the `Spacebar` once it has been highlighted.

As described above, depending on the shortcut, the user may or may not be prompted for further information.

- **Command Shortcuts**



There exists a set of operations to be performed in the Explorer thorugh the use of key presses.

These key presses are:

- `r` - this key press will remove the chosen file or firectory. The user is then promted to confirm this removal.
- `h` - this key press will bring up the help guide, which may be exited by pressing the `q` key.

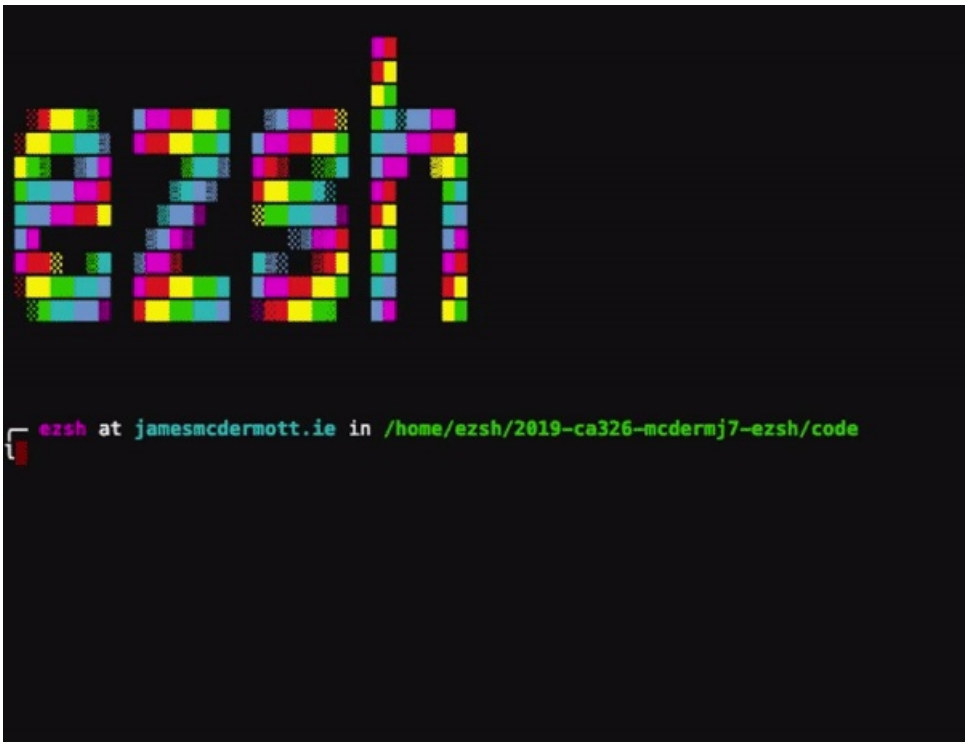
4.3 Prompt

- **Layout**

  The Prompt consists of a string of text at which users enter commands. This string of text is made up by the name of the user, the name of the machine they are working on, and the name of the current directory they are in.

This Prompt interface will refresh after each command entered and executed by the user.

- **Navigation**



Navigation in the prompt is completed through the use of the built-in `cd` command. This command allows the user to change directory, where the name of the desired directory follows the command in such formats as below:

  - `cd nameOfDirectory` to enter a specified directory.
  - `cd ..` to return to the previous directory.
  - `cd` to return to the user's home directory.

Names of directories (and other files) can be autocompleted if partially entered. Autocompleteion is achieved through the use of the `Tab` key.

The user can also navigate through any previously entered commands at the Prompt. The user can make use of the `Up` and `Down` keys to switch between these, which appear in chronological order.

- **Built-in Commands**

  - history

The `history` command allows users to view the entire history of what they have entered at the Prompt.

This action can be execute by simply entering `history`. This would yield output similar to the following:

```
1. cd ..
2. ls -al
3. mkdir new_directory
4. touch file.txt
5. mv file.txt new_directory
```

From here you can execute any line of your history using the `!` character.

Entering !2 at the prompt will execute the second line of your history file, just as !4 will execute the fourth line of your history file.

- star

The `star` command allows users to jump around their filesystem with ezsh's psuedo-bookmark system.

Stars can be added to your collection of stars using the `star` command followed by the directory you wish to add. Examples of this are found below:

`star code/` to add a directory called code.
`star ../` to add the directory one level above the user.
`star code/tests` to add the tests directory inside the code directory.

You can view your stars at anytime by simply running `star` with no following commands or directories. This would yield an output similar to this:

```
1. /home/username/code/
2. /home/username/
3. /home/username/code/tests
```

From here you can execute any line of your stars using the `*` character.

Entering *2 at the prompt will pop the user to their `/home/username`, just as *3 will pop the user into `/home/username/code/tests`.

- help
  The `help` command is a simplified version of the `man` command.

  `help` reduces screen clutter, and only presents relevant information regarding the command.

  You can run `help` in the following format:

  - `help mdkir` to show mkdir information.
  - `help rm` to show rm information.

- **Built-in Operations**
  - Logical And

    ezsh supports the Logical And operator. In ezsh, this is denoted by `&`. This allows the running of multiple commands asynchronously. Examples are found below:

    - `echo 'hi' & echo 'bye'`
    - `touch file.txt & ls`
  - Redirection

    ezsh supports redirection of file I/O. In ezsh you can redirect output from a command or file like so:

    - `cat file.txt > file2.txt`
    - `echo 'hi' > file.txt`
    - `python3 main.py > outputFile`
  - Pipes
    ezsh supports the use of pipes, meaning the output of one program as the input of another one. Pipes can be used like so:

    - `ls -al | grep 'd'`
    - `cat main.py | sed '51, 640'`
    - `history | grep 'cp -R'`

4.4 Messenger

```
Commands executed in ezsh:
Command --> ls -alR
```

- **Layout**

  The Messenger pane shows the commands which are executed in both the Explorer and the Prompt.

  This information is relayed to show what the user is executing on either side.

- **Features**

  Aside from relaying information from one side to the other, the Messenger pane has no other features.

4.5 Infograph

```
ls          cp
cd          echo
touch       chmod
cat         chown
gedit       custom1
mkdir       custom2
```

- **Layout**

  The Infograph pane (which at this point is options) shows the user popularly used commands in the shell environment.

- **Features**

  Aside from relaying information to the user, the Infograph pane has no other features at this time.

5. Plugins

5.1 Current Plugin Options

Currently, the are four available plugins for ezsh:

- Prompt
- Explorer
- Messenger
- Infograph

5.2 Adding Plugins

Adding any one of these plug-in programs to your ezsh environment to configure your own unique shell is covered it the next section.

6. Configuration

6.1 Configuration File Location

You can change many things about the shell so that is runs to suit your needs. The file used for configuartions is found at `~/.ezsh/.ezsh.conf`.

The following is an example config you can use to get your started.

6.2 Default ezsh Configuration

```
ezsh 0 tmux resize-pane -t 0 -x 3 && tty > .ezsh/.ezsh.tty && clear && ./exp
ezsh 1 tmux resize-pane -t 1 -y 3 && tty >> .ezsh/.ezsh.tty && clear && toilet -f big
ezsh 2 tmux resize-pane -t 2 -x 120 -y 3 && tty >> .ezsh/.ezsh.tty && clear && ./msg
```

Each line may be broken down as follows:

```
ezsh PANE_NUMBER tmux resize-pane -t TARGET_NUMBER -x WIDTH -y HEIGHT && tty > ./ezsh
```

6.3 Building your own ezsh Configuration

Keeping the breakdown of the line above in mind, the following parts of that line can be changed in accordance with the following restrictions:

- PANE_NUMBER:
  This is the number of the pane you are on.
  Assuming you start on line one, if you find yourself on the nth line, the PANE_NUMBER should be `n-1`.

- TARGET_NUMBER:
  This should always be the same as the current PANE_NUMBER.
- WIDTH:
  WIDTH can be an integer between 1 and 180 which corresponds to the width of that PANE_NUMBER
- HEIGHT:
  HEIGHT can be an integer between 1 and 180 which corresponds to the width of that PANE_NUMBER
- PROGRAM_NAME:
  The PROGRAM_NAME can be on of the following four plugins which ezsh can use:
  - ./prompt
  - ./exp
  - ./msg
  - ./info

6.4 Sample ezsh Configurations

- **Sample One**

```
ezsh 0 tmux resize-pane -t 0 -x 90 && tty > .ezsh/.ezsh.tty && clear && toilet -f big
ezsh 1 tmux resize-pane -t 1 -x 70 && tty >> .ezsh/.ezsh.tty && clear && ./exp
ezsh 2 tmux resize-pane -t 2 && tty >> .ezsh/.ezsh.tty && clear && ./info
ezsh 3 tmux resize-pane -t 3 -x 40 && tty >> .ezsh/.ezsh.tty && clear && ./msg
```

- **Sample Two**

```
ezsh 0 tmux resize-pane -t 0 -x 90 && tty > .ezsh/.ezsh.tty && clear && toilet -f big
ezsh 1 tmux resize-pane -t 1 -x 70 -y 25 && tty >> .ezsh/.ezsh.tty && clear && ./exp
ezsh 2 tmux resize-pane -t 2 && tty >> .ezsh/.ezsh.tty && clear && ./info
```