Linux commands:

Vi filename - creates the file with that name

1. Press insert
2. Then type as normal
3. After finished press escape
4. Then to :w, to write and :wq to write and quit

When trying to run this file it wont work as it was not given access
To give access we use

Chmod +x filename
Means to change mode

1. +x or 0766 means anyone
2. 0700 means only owner
3. Ls -1 script.sh *to see what the permission is*

Or we can say *bash script.sh*

We can comment using #message

And multiple line commenting using <<COMENT1 msg COMMENT1

debugging
Bash -x can be used to check the script's code for debugging
These below must be written into shell
Set -x displays commands as they are executed
Set - v displays only the inputs
Set -n only reads the code but does not execute

Say we want to wait for an input

```
echo "press any key"
while true; do
read -rsn1 key
if [[ -n "$key" ]]; then
echo "terminated"
break
fi
Done
```

If we want to store that input
*read(-t number **for a timer not needed**)-p "Enter your name : " name*
echo "Hi, $name. Let us be friends!"

Shell contains many variables

System Variable	Meaning	To View Variable Value Type
BASH_VERSION	Holds the version of this instance of bash.	echo $BASH_VERSION
HOSTNAME	The name of the your computer.	echo $HOSTNAME
CDPATH	The search path for the cd command.	echo $CDPATH
HISTFILE	The name of the file in which command history is saved.	echo $HISTFILE
HISTFILESIZE	The maximum number of lines contained in the history file.	echo $HISTFILESIZE
HISTSIZE	The number of commands to remember in the command history. The default value is 500.	echo $HISTSIZE
HOME	The home directory of the current user.	echo $HOME
IFS	The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the read builtin command. The default value is <space><tab><newline>.	echo $IFS
LANG	Used to determine the locale category for any category not specifically selected with a variable starting with LC_.	echo $LANG
PATH	The search path for commands. It is a colon-separated list of directories in which the shell looks for commands.	echo $PATH
PS1	Your prompt settings.	echo $PS1
TMOUT	The default timeout for the read builtin command. Alsom in an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the command. If not input provided it will logout user.	echo $TMOUT
TERM	Your login terminal type.	echo $TERM export TERM=vt100
SHELL	Set path to login shell.	echo $SHELL
DISPLAY	Set X display name	echo $DISPLAY export DISPLAY=:0.1
EDITOR	Set name of default text editor.	export EDITOR=/usr/bin/vim

To see the value of variables we type

Echo “$PATH” we need to write $ in front of the variable name the variable name can also be highlighted by using { … }

To set default values to variables we use :=
All user defined variables are local and so go away when we start a new instance, so intsead we can make them global using *export -p*

To delete variables we use *unset variable name*

To print we use *echo or printf(works like c)*

To compile a C file we use gcc -c file.c -o name.o *to compile c programs*
To link 2 files together gcc YourAssemblyFile.o YourMacroFile.o -o YourExecutable

Running a file for nexus a7,
Use *make* inside the directory containing .s file.
Then use make clean to clear old .bin files

Then we need to send it to the board
Inside the directory ~/VEGA/vega-tools/utils/eth_transfer
./send.sh ~ /file.bin riscv.dtb

To get to portal of board
Cd -
Dmesg
Minicom -D /dev/tty
Minicom -D /dev/ttyUSB1

Symbols used:

Quote type	Name	Meaning	Example (type at shell prompt)
"	The double quote	The double quote ("quote") protects everything enclosed between two double quote marks except $, ', " and \.Use the double quotes when you want only **variables and command substitution**. * **Variable** - Yes * **Wildcards** - No * **Command substitution** - yes	The double quotes allowes to print the value of $SHELL variable, disables the meaning of wildcards, and finally allows command substitution. echo "$SHELL" echo "/etc/*.conf" echo "Today is $(date)"
'	The single quote	The single quote ('quote') protects everything enclosed between two single quote marks. It is used to **turn off the special meaning** of all characters. * **Variable** - No * **Wildcards** - No * **Command substitution** - No	The single quotes prevents displaying variable $SHELL value, disabled the meaning of wildcards /etc/*.conf, and finally command substitution ($date) itself. echo '$SHELL' echo '/etc/*.conf' echo 'Today is $(date)'
\	The Backslash	Use backslash to change the special meaning of the characters or to escape special characters within the text such as quotation marks.	You can use \ before dollar sign to tell the shell to have no special meaning. Disable the meaning of the next character in $PATH (i.e. do not display value of $PATH variable): echo "Path is \$PATH" echo "Path is $PATH"