Text Processing commands : ctrl+d = to escape from running cmd and go back to normal directory

1. Input Redirection (0): Read input from the file and display

Eg: cat 0< input.txt

Case : read from file and write again in another file

Eg cat 0< input.txt 1> output.txt

1. Output redirection (1): Write out put in file , not display in terminal

Eg: cat 1> output.txt

Apple

Mango

Orange

now do : cat output.txt --- >> it will give all the input given by standard input device (0) keyboard in ouput.txt which is stored in file output.txt

case : if we want to override the old data in output.txt file then 🡪 we can use single right angular bracket 1>

eg :cat 1> output.txt

Apple1

Mango1

Orange1

Now : cat output.txt will print only Apple1,Mango1,Orange1

Case : if we want to append in output.txt then we use : cat 1>> output.txt ( and write what we want to append or add in the given file)

Case : date 1>> output.txt =🡺> date will append in ourput.txt

1. Error Redirection (2)☹ This is also same as outpur redirection

Eg: cal 12 2042 2> errorMessage.txt

Cal 12 shshshs 2> errorMessage.txt ----🡪> now in the errorMessage.txt , we can see error message

Case : we need to first output redirect and if there is error while doing output redirection then error message should be redirected to errorMessage.txt

Eg: cal 12 shssh 1> output.txt 2> errorMessage.txt

Some common task processing are :

1. Displaying the content of a file efficiently
2. Search for a particular pattern in a file
3. Arranging the data in a file in order
4. Generating report from the data in a file

: need to give some time to learn

**Note : Practice is in Scripting folder under documents . And “ CgyWin” is can be used to unix which I have installed**

**Shell Scripting :**

**Shell variable :**

**grep in unix ?**

A variable is a string of character to which a value can be assigned . valid variable must be all in Upper case , not starting with number .

If we don’t want any user having access to the variable to change its value , we need to write “readonly”

Eg: $readonly stream=linux

Now this variable is readonly access if someone tries to change it gives error .

**Special variable in Unix :**

|  |
| --- |
| **$0**  The filename of the current script.  **echo $0**  **then it will give the filename of the current script** |
| 2 | **echo $n**  These variables correspond to the arguments with which a script was invoked. Here**n**is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on). |
| 3 | **$#**  The number of arguments supplied to a script. |
| 4 | **$\***  All the arguments are double quoted. If a script receives two arguments, $\* is equivalent to $1 $2. |
| 5 | **$@**  All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2. |
| 6 | **$?**  The exit status of the last command executed. |
| 7 | **$$**  The process number of the current shell. For shell scripts, this is the process ID under which they are executing. |
| 8 | **$!**  The process number of the last background com |

**Command line argument : syntax :** sh script.sh Alpah bravo Charlie delta

Here provided Alpha Bravo Charlie Delta are command line argument

**Positional parameter :** echo $1 = will print first argument Alpha and $2 print Bravo

**Command substitution ?**

**Bash comparison operators:**

**Integer comparison operator**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Usage** |
| **-eq** | **Equal to** | **[$a -eq $b]** |
| **-ne** | **Not equl to** | **[$a -ne $b]** |
| **-gt** | **Greater tahn** | **[$a -gt $b]** |
| **-lt** | **Lesser than** | **[$a -lt $b]** |
| **-le** | **Less than or equal to** | **[$a -le $b]** |
|  |  |  |

**String comparison**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Usage** |
| **==** | **Equal to** | **[“$a” == “$b”]** |
| **!=** | **Not equal to** | **[“$a” != “$b”]** |
| **-n** | **Not a null string** | **[-n “$a”]** |
| **-z** | **Null string** | **[-z “$a”]** |

**File operator**

|  |  |
| --- | --- |
| **File operator** | **True if …** |
| **-a file** | **File exist** |
| **-f file** | **File exist and is a regular file** |
| **-d file** | **File exist and is a directory** |
| **-r file** | **File is readable by the current user** |
| **-w file** | **Fie is writable by the current user** |
| **-x file** | **File is executable by the current user** |
| **-s file** | **File exists and in non empty** |

**Symbol and Command used in shell scripting :**

**Grep : global regular expression print**

1. **Example : grep “hi ram” name.txt --- >> here this will search given string in name.txt**
2. **` ` -> this grave accent symbol is used to give command like : `date` it will print date**
3. **~ -🡪 this is home directory eg : ~/hello**
4. **Gedit 🡪 command :**
5. **< 🡪input Redirection :**
6. **> 🡪 output redirection**
7. **| pipe --- > A pipe chains commands together . It takes outpur from one command and feeds to the next as input .**

**Selection construct:**

**The selection construct in UNIX are :**

**A,**

1. **If statement 2.if else statement 3. Nested if statement 4. Case statement**

**Case Statement**

: refer choices.sh in documents (prepare for practice : open gitbash and type vi choices.sh)

Now to switch the execution according to provided input we use command line argument . for example: sh choices.sh 10

Here 10 is passing from the command line .

**B , Iterative constructs**

For loop , while loop , until loop

For loop is count controlled and while loop & until loop are condition controlled

**Vi Editor in UNIX :**

1. How to open VI in UNIX : 1st : go to unix or gitbash # give vi and enter # Vi will open up .
2. Vi always open in Command mode . So if we want to make change in file then we need to switch to insert mode by typing “I” in key board .
3. To get back to command mode : press ESC

Commnads in Vi editor:

**VI Editing commands**

* i – Insert at cursor (goes into insert mode)
* a – Write after cursor (goes into insert mode)
* A – Write at the end of line (goes into insert mode)
* ESC – Terminate insert mode
* u – Undo last change
* U – Undo all changes to the entire line
* o – Open a new line (goes into insert mode)
* dd – Delete line
* 3dd – Delete 3 lines.
* D – Delete contents of line after the cursor
* C – Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
* dw – Delete word
* 4dw – Delete 4 words
* cw – Change word
* x – Delete character at the cursor
* r – Replace character
* R – Overwrite characters from cursor onward
* s – Substitute one character under cursor continue to insert
* S – Substitute entire line and begin to insert at the beginning of the line
* ~ – Change case of individual character

**Note**: You should be in the “**command mode” to execute these commands**. VI editor is **case-sensitive** so make sure you type the commands in the right letter-case.

Make sure you press the right command otherwise you will end up making undesirable changes to the file. You can also enter the insert mode by pressing a, A, o, as required.

**Moving within a file**

* k – Move cursor up
* j – Move cursor down
* h – Move cursor left
* l – Move cursor right

You need to be in the command mode to move within a file. The default keys for navigation are mentioned below else; You can **also use the arrow keys on the keyboard**.

**Saving and Closing and opening existing the file in vi(Semicolon must use to get cursor to write a command ):**

* Shift+zz – Save the file and quit
* :w – Save the file but keep it open
* :q – Quit without saving ( :q! it will override the
* :wq – Save the file and quit
* :w filename.txt (this will helps you give the name of vi file created )
* In unix terminal : .--- >> vi filename.sh ( it will open the scripting file or any of them)

You should be in the **command mode to exit the editor and save changes** to the file.