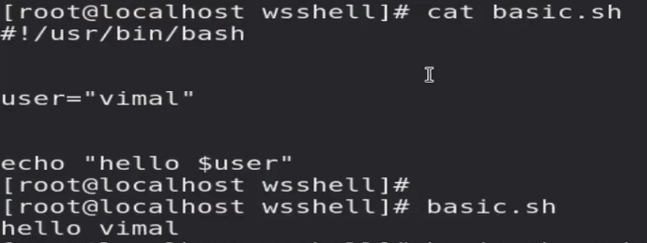
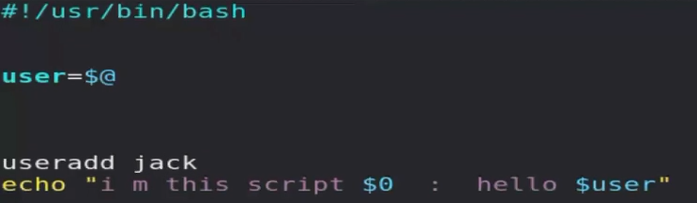
**Shell Scripting**

**Day 1**

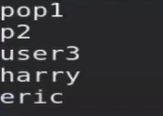
* Shell is just a concept.
* Behind the CLI this program is running.
* In linux most of places we use bash shell.
  + Shell scripting is a way to do **automation process**.
  + We have more automation tool like ansible, puppet and many more.
  + But every linux OS comes up with **shell script** by **default**.
* Some variable is provided by shell.
* For this you can use like echo **$?**
  + It will print the status of **last command**.
    - If the exit code is 0 then last command is success.
* We can use $PATH it will give you location where all the commands located.
* For giving path to your command or script you have to copy this inside that location or add another path.
  + **PATH=/wsshell:$PATH**
    - Here we use : so it will **add new path** without removing older paths.
    - If you want to use this your file must be executable.
  + For this you can use **chmod +x file.sh**
* Here our default shell is bash.
  + Else it will not work. Here we have to tell our file we want to use bash shell.
    - **#/bin/bash**
  + Add this above your file.
  + Here we have set the path but it is temporary, for making it permanent we have to write this inside **/root/.bashrc**

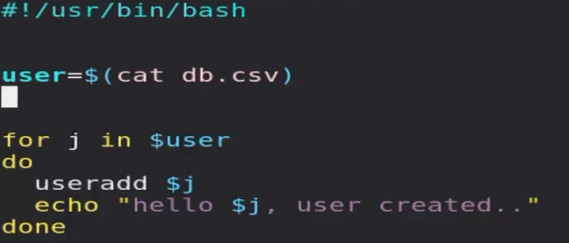


* + But this program is static instead of this we want to take input from user.
    - For this we can pass further values with command as a argument.
    - If you want to use this value inside script so you have to retrieve it.
  + This argument is stored inside variable 1.
    - Second value stored in variable 2.
    - This called **positional variable**.   
      
* Here we have set only two arguments.
  + But if we want to use multiple argument then we can use **@**
  + All the argument you pass is stored inside this variable.

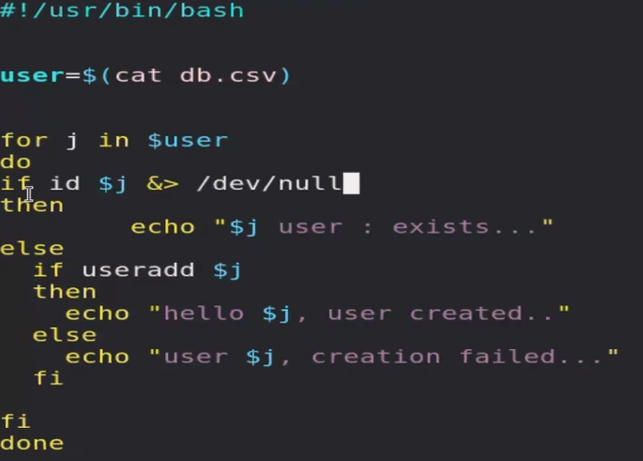


* Suppose you want to store any command in variable and want to use that variable so you can run that command anywhere.
  + X = date
  + Echo $X
    - But it will print date as a string.
  + X**=$(date)** or x**=`date`**
  + Echo $X
    - It will first run the command and store output inside x and we are printing x.
* Suppose you want to take a input from csv file, so you can use this syntax.
  + For I in $(cat db.csv) ; do echo $i; done

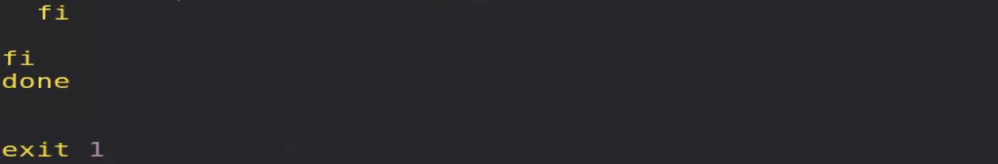
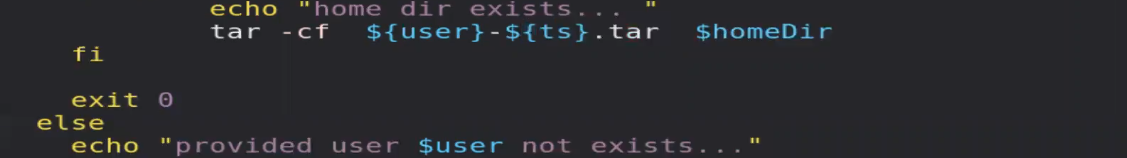
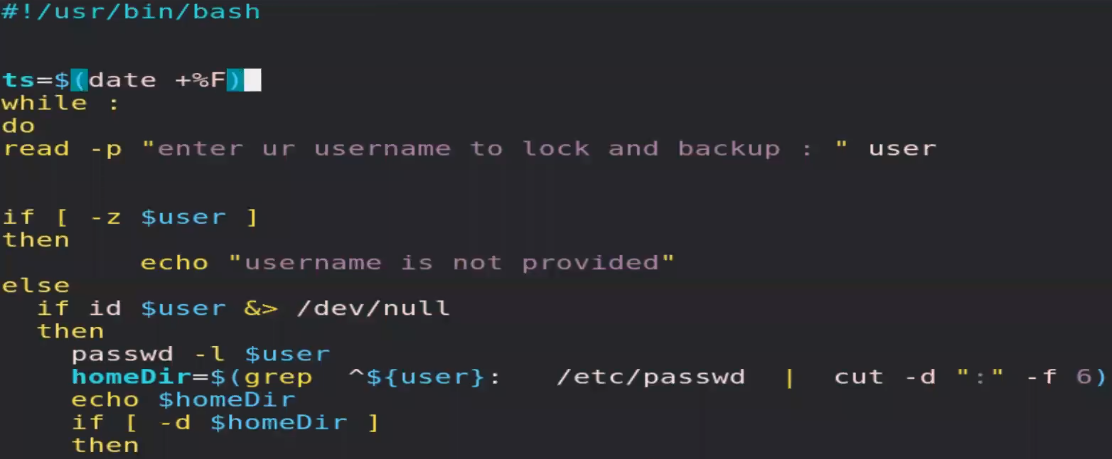




* If you want to lock any account, here we have to use **passwd**.
* For matching or testing something we have keyword called test.
  + Eg, **test 1 == 1**
  + But here < > >> has some different meaning, for this we are not using this symbol in the shell scripting.
  + Instead of this we use **–lt –ge –le –ne**
  + Here instead of test we can also use square bracket.
    - **[5 –lt 10]**
  + If [ 5 –lt 10 ]; then; echo “hi”; fi
* If you want to redirect success message then you have to use 1> or > but for failure you can redirect using 2>.
* If you want to send it to same file or do something on both results you can use &>.
  + It will work for both success & failure result.
  + We can throw this output to **garbage** as it is not important.
    - For this we have file **/dev/null**, it will work as a garbage for you.
* Using this we can optimize our code.

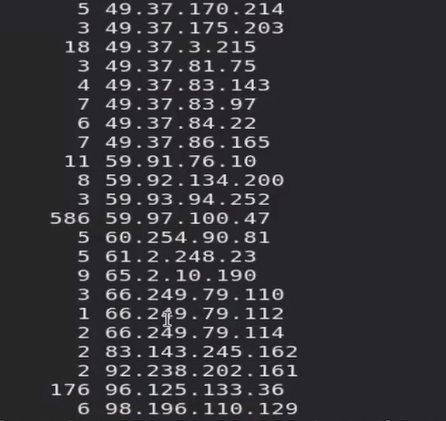


* If you want to take input from keyboard you have to use **read** keyword.
  + **Read myname** 
    - It will store data in myname variable.
* For lock user we have to use **passwd –l jack**
  + It will block jack user.
* But here we have to first check user has input username or not.
* For checking string is empty or not.
  + We can use **test –z $user**
    - It will return 0 if string is empty.
    - We can use –d option it is reverse of this.
    - It will return 0 if it is not empty.
  + Here we can add while loop if username is empty we can prompt to enter username again.
* It is also good practice to add exit code in script.
* Home directory of user is /home/.
* But we can change home directory while creating user.
  + Useradd –d /mnt/j2 j2
* We want to take a backup while locking a file.
  + But we have different home directory for different user.
  + So for retrieve this we can use /etc/passwd file.
  + It will also store home directory for all users.
* But we have find specific user info and inside that directory name.
  + For finding that user line we can use regex like.
  + **^root:**
  + For finding specific thing inside one line we can use cut command.
* **x=$(grep ^j2: /etc/passwd | cut –d “:” –f 6)**
  + here we are telling that our field separator is : and we want six field from line.
  + We are also storing this inside x variable.
* Test –e /etc/passwd
  + –x for executable file, –w for writable file.
  + –s for file exists & not empty, –r fo readable file.
  + –e file exists or not, –d directory exists or not.
* Now we have to create a backup.
  + For this we can use tar command.
  + Suppose you want to create daily backup.
  + So for this you can add time stamp in your tar file.

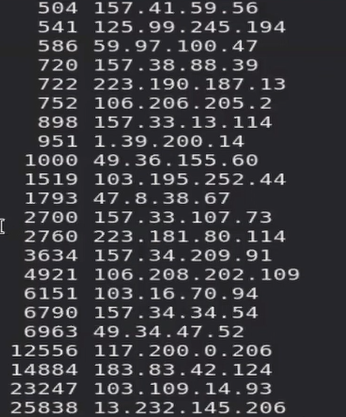


Day2

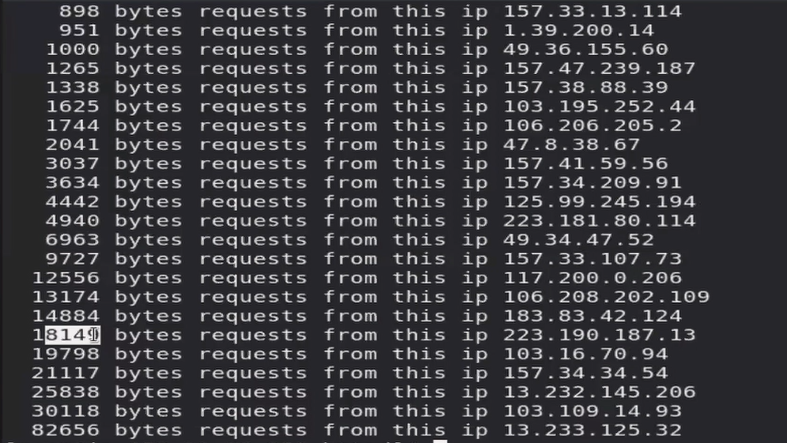
* On top of instance we have installed webserver and index.html page and start webserver.
  + Apache stores log inside **/var/log/httpd/access\_log**
  + If you want to connect to instance using redhat VM, ensure to remove read access to others.
    - Chmod o-r <key>.pem
    - Chmod g-r <key>.pem
    - Ssh only allows secure key.
* Here we want to find how muny unique users have connected.
  + For finding unique IP address from logs we will use **awk** command.
  + For retrieve column we can use this command.
  + By default they use space separator.
    - Awk -F: ‘{print $1 $6}’ /etc/passwd
      * It will give you column 1 & 6.
      * But they are merged (continuos) we can use this thing to separate them.
    - Awk -F: ‘{print $1 “ ” $6}’ /etc/passwd
      * You can also add something else instead of space.
  + Awk ‘{print $1}’ access\_log | sort
    - We can use this sorting them in ascending order.
  + **Awk ‘{print $1}’ access\_log | sort | uniq | wc -l** 
    - Uniq will give you unique things.
* We also see this IP has hit our webserver this much time.
  + Awk ‘{print $1}’ access\_log | sort | uniq -c
    - Here we got 2 fields.



* + Awk ‘{print $1}’ access\_log | sort | uniq -c |sort -n
    - Sort by default sort first column else we can tell using,
    - Sort -n k2



* + We can also use -r for recursive.
* We can also find total hits day wise.
  + For this we can use date command.
  + Total hit today…
  + Grep $(date +%e/%b/%G) access\_log | Awk ‘{print “bytes request from this ip” $1}’ | sort | uniq -c | sort -n -k1



* If you want to compare something before printing this we can compare it inside single quots but outside curly braces.
* Suppose we want retrieve unique IP address and pages accessed that does not exists.
  + Here we have to compare is status code = 404.
    - Awk ‘$9==404 { print $1 “ ” $7}’ access\_log | sort | uniq -c | sort -n
* You can write this command inside aws instance inside shell script, & run this whenever we want.
* Suppose we want to get data from file and for some operation we want every word in different line.
  + So we have to put other words in next line after space.
    - For this we can use tr.
  + But we may face one issue, we may have more space or tab.
  + So here we can use -s option.

