**LİNUX LESSON 1**

HEAD&TAİL COMMAND

Show the first 10 lines of clarusway.txt.

head clarusway.txt

Show the first 5 lines of clarusway.txt.

head -5 clarusway.txt

Show the last 10 lines of clarusway.txt.

tail clarusway.txt

Show the last 5 lines of clarusway.txt.

tail -5 clarusway.txt

CAT && ECHO COMMAND

Create three files with echo command and name them file1 file2 file3.

echo "this is file1" > file1

echo "this is file2" > file2

echo "this is file3" > file3

cat file1 file2 file3

Concatenate(combine) file1, file2 and file3 to all.txt file.

cat file1 file2 file3 > all.txt

cat > summer.txt

Today is cold.

Today is rainy

After the last line, type and hold the Control (Ctrl) key and press d.

FİND COMMAND

Find all the files whose name is clarusway.txt in a current working directory.

find . -name clarusway.txt

Find all the files under /home directory with name clarusway.txt.

find /home -name clarusway.txt

Find all txt files in the working directory.

find . -type f -name "\*.txt"

To find all 100MB files under home directory.

find /home -size 100M

Find all the files which are modified 10 days ago in /home directory.

find /home -mtime 10

GREP COMMAND

Grep is a Linux / Unix command-line tool used to search for a string of characters in a specified file.

Search linux.txt file for kernel.

grep "kernel" linux.txt

Search all files for is.

grep "is" \*

it didn't find linux expression. Because grep is case sensitive. Now try with the following command.

grep -i "linux" linux.txt

Search linux.txt file for ker.

grep -i "ker" linux.txt

Grep allows you to find and print the results for whole words only with -w flag. Let's try with the following command.

grep -w "kernel" linux.txt

We can display the lines that are not matched with the specified search sting pattern using the -v option.

grep -v "kernel" linux.txt

Use -A and a number of lines to display after a match.

grep -A3 "line 5" clarusway.txt

Use -B and a number of lines to display before a match.

grep -B2 "line 5" clarusway.txt

We can also use grep command with | (pipe).

man pwd | grep "print"

man find | grep -A5 "size"

history | grep "find"

**LİNUX-02 : LİNUX ENVİRONMENT VARİABLES**

Displays the current environment or sets the environment for the execution of a command. If env is run without any options, it prints the variables of the current environment. Otherwise, env sets each NAME to VALUE and executes COMMAND.

|  |  |
| --- | --- |
| -i, --ignore-environment | Start with an empty environment. |
| -0, --null | End each output line with a 0 ([null](https://www.computerhope.com/jargon/n/null.htm)) byte rather than a [newline](https://www.computerhope.com/jargon/n/newline.htm). |
| -u, --unset=NAME | Remove variable NAME from the environment. |
| --help | Display a help message and exit. |
| --version | Display version information and exit. |
| - | Same as -i. |

OPTİONS

Understanding the shell variable.​

CLARUS=way

After print CLARUS=way you can see that with env command

İf you print “set | grep CLARUS” you will see “way” value

echo $CLARUS

printenv prints the values of the specified environment VARIABLE(s). If no VARIABLE is specified, print name and value pairs for them all.

Remove the environment variable with unset command.​

export WAY=clarusway

env | grep WAY

unset WAY

env | grep WAY

PART 2 - PATH VARİABLE

PATH variable.​

printenv PATH

cd /bin

ls ca\*(see words that starting with “ca”)

Double Quotes.​

MYVAR=my value

echo $MYVAR

MYVAR="my value"

echo $MYVAR

MYNAME=james

MYVAR="my name is $MYNAME"

echo $MYVAR

MYNAME="james"

MYVAR="hello $MYNAME"

echo $MYVAR

MYVAR="hello \$MYNAME"

echo $MYVAR

Single Quotes.​

echo '$SHELL'

echo 'My\$SHELL'

**LİNUX-03: MANAGİNG USERS AND GROUPS**

Useradd

sudo useradd user1

cd /etc

ls login\*

cat login.defs

sudo nano login.defs # change the CREATE\_HOME variable's value to "no"

sudo useradd user4

cd /home && ls

cat /etc/passwd

sudo useradd -m user5 # force to system to create a home directory for user with -m option.

cd /home && ls

sudo useradd -m -d /home/user6home user6 # change the user's home directory name with -d option.

ls

sudo useradd -m -c "this guy is developer" user7 # give a descrpition to user with -c option.

cat /etc/passwd

cat /etc/passwd | grep user7

su user1 (switch user)

passwd user1(give password user1)

User management

cat /etc/passwd (see all user)

tail -3 /etc/passwd

userdel.

cat /etc/passwd

sudo userdel user5

cat /etc/passwd

cd /home && ls

sudo userdel -r user1 # delete user and its home directory with -r option.

cd /home && ls

Usermod.​

cat /etc/passwd

sudo usermod -c "this guy will be an aws solution architect" user7

cat /etc/passwd

sudo usermod --help

sudo usermod -l Superuser user2 # change the name of the user2 with -l option.

cat /etc/passwd

User Passwords

passwd-etc/shadow-etc/login.defs.​

sudo su

useradd user8

passwd user8

cd /etc

cat shadow

cat login.defs

Group Management

groups.​

groups

sudo groupadd linux

sudo groupadd aws

sudo groupadd python

cat /etc/group

groups

sudo usermod -a -G linux ec2-user # append ec2-user in linux group.

cat /etc/group

groups

sudo usermod -G aws ec2-user # this command deletes all groups that ec2-user in except default group of ec2-user and add ec2-user to aws group.

cat /etc/group

sudo groupmod -n my-linux linux # change the name of the linux group.

cat /etc/group

groups

cat /etc/group

sudo groupdel python

cat /etc/group

sudo gpasswd -a user7 aws # add a user to a group.

cat /etc/group

sudo gpasswd -d user7 aws # delete a user to a group.

cat /etc/group

**LİNUX-04 : USİNG PACKAGE MANAGERS İN LİNUX**

Update Amazon Linux Instance.​

sudo yum update

Update Ubuntu's package list. This command updates the local repo database but do not install any package.​

sudo apt update

Upgrade the packages. This command installs the listed available packages.

sudo apt upgrade

Install git on Amazon Linux instance.​

sudo yum install git -y

Uninstall git on Amazon Linux instance.​

sudo yum remove git –y

Uninstall git with dependencies on Amazon Linux instance without any interruption.​

sudo yum autoremove git –y

Check the info for the git package installed on Amazon Linux instance.​

sudo yum info git

List all available packages for Amazon Linux instance.​

sudo yum list

List all available git packages for Amazon Linux instance.​

sudo yum list git

List all available versions of git packages on Amazon Linux instance.​

sudo yum --showduplicates list git

**LİNUX-05: FİLTERS AND CONTROL OPERATORS**

Part 1 - Using Filters

​cat

concatenate files and print on the standard output​

tee

Read from standard input and write to standard output and files​

Write the content of the count.txt file in reverse order to another file named temp.txt and display the content of temp.txt in reverse order.

tac count.txt | tee temp.txt | tac (tac list reverse)

grep​

Print lines that match patterns. The most common use of grep is to filter lines of text containing (or not containing) a certain string.

cat > tennis.txt

Amelie Mauresmo, Fra

Justine Henin, BEL

Serena Williams, USA

Venus Williams, USA

press ctrl+d for EOF

Display only the lines of tennis.txt that includes 'Williams'.​

cat tennis.txt | grep Williams

Display only the lines of tennis.txt that includes 'us'.​

cat tennis.txt | grep us

cut

Display the owners column (3rd column) of all the files in current directory.​cut

The cut filter can select columns from files, depending on a delimiter or a count of bytes​

ls -l | cut -d' ' -f3

cut –c (only character)

cut –c2 days.txt(only second characters)

tr​

The command 'tr' stands for 'translate’. It is used to translate, like from lowercase to uppercase and vice versa or new lines into spaces

cat << EOF > clarusway.txt

Clarusway:Road to reinvent yourself.

EOF(End of File)

In the content of clarusway.txt, replace or translate aer letters with 'QAZ'.​

cat clarusway.txt | tr aer QAZ

Write the content of count.txt on the same line.​

cat count.txt | tr '\n' ' '

wc​

Print line, word, and charecters for each file.

Count the lines, words and letters of the content of count.txt.​

Find how many users are there in the computer.

wc -l /etc/passwd

sort​

The sort filter will default to an alphabetical sort. The sort filter will default to an alphabetical sort.

cat << EOF > marks.txt

aeron-9

albert-9

james-9

john-10

oliver-7

tom-7

victor-10

walter-8

EOF

Sort the content of marks.txt.​

sort marks.txt

Sort the content of marks.txt in reverse order.​

sort -r marks.txt

uniq​

report or omit repeated lines. With the help of uniq command you can form a sorted list in which every word will occur only once.

cat << EOF > trainees.txt

john

james

aeron

oliver

walter

albert

james

john

travis

mike

aeron

thomas

daniel

john

aeron

oliver

mike

john

EOF

Display the content of trainees.txt.​

cat trainees.txt

Display only the unique names in the content of trainees.txt.​before using uniq command, the file must be sorted​

sort trainees.txt | uniq

exercise: cut -d ',' -f3 country.txt | grep -i ^con

list words that starting with “con” third column

Using Control Operators

“ ;​ ”

More than one command can be used in a single line with ;.

Write two seperate cat command on the same line using ;.​

cat days.txt ; cat count.txt

echo Hello ; echo World!

&

When a line ends with an ampersand &, the shell will not wait for the command to finish. You will get your shell prompt back, and the command is executed in background. You will get a message when this command has finished executing in background.​

Run sleep 10 command and show that the kernel is busy until the process of this command ends.​

$?​

This control operator is used to check the status of last executed command. If status shows '0' then command was successfully executed and if shows '1' then command was a failure.

Run ls command and show that it is executed successfully.​

ls

echo $?

&&

The command shell interprets the && as the logical AND. When using this command, the second command will be executed only when the first one has been successfully executed.​

Display days.txt and if it runs properly display count.txt.​

cat days.txt && cat count.txt

||

The command shell interprets the (||) as the logical OR. This is opposite of logical AND. Means second command will execute only when first command will be a failure.​

Display days.txt or write 'clarusway' on the screen, then write 'one'.​

cat days.txt || echo clarusway ; echo one

&& and ||

We can use this logical AND and logical OR to write an if-then-else structure on the command line. This example uses echo to display whether the rm command was successful.​

Make a copy of file1.txt and named it file11.txt.​

cp file1.txt file11.txt

Delete file11.txt and write a message if it is deleted properly.​

rm file11.txt && echo 'it worked' || echo 'it failed'

#

Everything written after a pound sign (#) is ignored by the shell. This is useful to write a shell comment but has no influence on the command execution or shell expansion.​

Run the echo command and add a comment line.​

echo '# is the comment sign' # echo command displays the string comes after it.

echo # is the comment sign

echo \# is the comment sign

**LİNUX-06 : SHELL SCRİPTİNG BASİCS**

Part 1 - Shell Scripting Basics

Create a script file named basic.sh. Note all the scripts would have the .sh extension.

#!/bin/bash

echo "Hello World"

Before we add anything else to our script, we need to alert the system that a shell script is being started. This is done specifying #!/bin/bash on the first line, meaning that the script should always be run with bash, rather than another shell. #! is called a shebang because the # symbol is called a hash, and the ! symbol is called a bang.

After to save the above content, we need to make the script executable.

chmod +x basic.sh

Then we can execute the basic.sh. To execute basic.sh, it is required to add ./ beginning of the basic.sh. ./ means we're calling something in the current working directory. We have to specify the path for executables if they're outside our $PATH variable.

./basic.sh

Shell Comments

Bash ignores everything written on the line after the hash mark (#). The only exception to this rule is the first line of the script that starts with the #! characters.

Comments can be added at the beginning on the line or inline with other code. Let's update basic.sh.

#!/bin/bash

echo "hello"

# date

pwd # This is an inline comment

# ls

Part 2 - Shell Variables

A variable is pointer to the actual data. The shell enables us to create, assign, and delete variables.

Create a new file and name it variable.sh.

#!/bin/bash

NAME=Joe

echo $NAME

Command Substitution

Command substitution empowers us to take the output of a command or program (which would usually be written on the screen) and save it as the value of a variable. To do this we put it inside brackets, followed by a $ symbol.

content=$(ls)

echo $content

or we can use `(backtick)

content=`ls`

echo $content

example script:

#!/bin/bash

run=$(cut –d ‘ ‘ –f3- days.txt )

echo $run (see from third column to last column within of EOF.txt)

Console input

The Bash read command is a powerful built-in utility used take user input.

#!/bin/bash

echo "Enter your name: "

read NAME

echo "Welcome $NAME"

When writing interactive bash scripts, we can use the read command to get the user input. To specify a prompt string, use the -p option. The prompt is printed before the read is executed and doesn’t include a newline.

read -p "Enter your name: " NAME

echo "Welcome $NAME"

When entering sensitive information we do not want to display input coming. For this we can use read -s

read -p "Enter your name: " NAME

echo "Welcome $NAME"

read -s -p "Enter your password: " PASSWORD

echo -e "\nYour password is $PASSWORD"