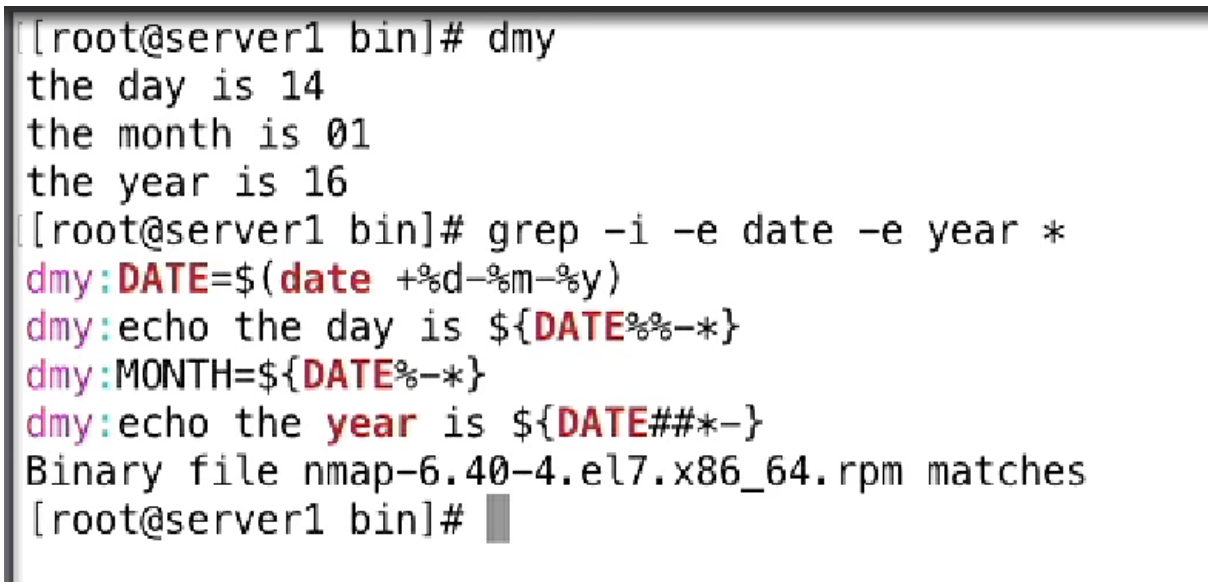
Lesson 4: Using Essential External Tools

grep is a very flexible tool to search for text patterns based on regular expressions

- **grep –i**: case insensitive
  - **grep –i what ***
- **grep –v**: exclude lines that match the pattern
  - **grep –v what ***
- **grep –r**: recursive
  - **grep –r what ***
- **grep –e (egrep)**: matches more regular expressions
  - **grep –e 'what' –e 'else' ***
- **grep –A5**: shows 5 lines after the matching regex
- **grep –B4**: shows 4 lines before the matching regex



test allows for testing of many items

- expression: **test (ls /etc/hosts)**
- string: **test –z $1**
- integers: **test $1 = 6**
- file comparisons: **test file1 –nt file2**
- file properties: **test –x file1**

Test -z $1 → variable is empty or not?

Test $1 = 6 → variable have some specific value or not?

Test file1 -nt file2 to test the files or test the file prop or test the file is empty or not

Three Ways to **test**

- **test –z $1**: old method, using an internal bash command
- **[-z $1]**: equivalent to test, using a bash internal
- **[[-z $1]]**: new improved version of [...]. Not as universal as [...]; it has && and || built in
- Best practice: if it doesn't work using [...], try using [[...]]
- If compatibility with older shells doesn't matter, use [[...]]
- Compare the following:
  - **[$BLAH = a*] || echo string does not start with a**
  - **[[$BLAH = a*]] || echo string does not start with a**

```
[root@server1 bin]# BLAH=abced
[root@server1 bin]# [ $BLAH = a* ] || echo string does not start with an a
string does not start with an a
[root@server1 bin]# [[ $BLAH = a* ]] || echo string does not start with an a
[root@server1 bin]#
```

Using **cut** and **sort**

- **cut** is used to filter a specific column or field out of a line
- **sort** is used to sort data in a specific order
- **cut** and **sort** are often seen together

cut and **sort** Examples

- **cut –f 1 –d : /etc/passwd**
- **sort /etc/passwd**
- **cut –f 2 –d : /etc/passwd | sort –n**
- **du –h | sort –rn**
- **sort –n –k2 –t : /etc/passwd**

```
[root@server1 home]# sort -n -r -k3 -t : /etc/passwd | head
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
boris:x:1004:1006::/home/boris:/bin/bash
paul:x:1003:1005::/home/paul:/bin/bash
mike:x:1002:1004::/home/mike:/bin/bash
linda:x:1001:1002::/home/linda:/bin/bash
user:x:1000:1000:user:/home/user:/bin/bash
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
unbound:x:998:997:Unbound DNS resolver:/etc/unbound:/sbin/nologin
colord:x:997:996:User for colord:/var/lib/colord:/sbin/nologin
saslauth:x:996:76:"Saslauthd user":/run/saslauthd:/sbin/nologin
[root@server1 home]#
```

Sort -rn → reverse number sort in reverse order

Sort -n -k2 -t : /etc/passwd → it will help you to sort the file with numeric sort and column with -k3 based on it but it will not cut anything

Using **tail** and **head**

- **tail** is used to display the last line(s) of a file
- **head** is used to display the first line(s) of a file
- **tail -2 /etc/passwd**
- **head -2 /etc/passwd**
- **head -5 /etc/passwd | tail -1**

Using **sed**

- **sed** is more than a text processing utility, it's a programming language with many features
- A limited set of these are useful in scripts
- **sed –n 5p /etc/passwd**
- **sed –i s/old-text/new-text/g ~/myfile**
- **sed –i –e '2d' ~/myfile**
- **sed –i –e '2d;20,25d' ~/myfile**

Last command will help us to delete the line 2nd and also from 20 to 25 it will delete the lines

Using awk

- Like **sed**, **awk** is a very rich language
- In scripts you'll appreciate it as an alternative to **cut** to filter information from text files based on regular expression-based patterns
- The basic usage is **awk '/search pattern/ {Actions}' file**
- **awk –F : '{ print $4 }' /etc/passwd**
- **awk –F : '/user/ { print $4 }' /etc/passwd**
- **awk –F : '{ print $1,$NF }' /etc/passwd** ($NF is the last field)
- **awk –F : '$3 > 500' /etc/passwd**
- **awk –F : '$NF ~/bash/' /etc/passwd**

```
[root@server1 ~]# useradd pete
[root@server1 ~]# awk -F : '/pete/ { print $4 }' /etc/passwd
1008
[root@server1 ~]# awk -F : '/pete/ { print $1, $NF }' /etc/passwd
pete /bin/bash
[root@server1 ~]#
```

```
[root@server1 ~]# awk -F : '$NF ~/bash/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1000:1000:user:/home/user:/bin/bash
linda:x:1001:1002::/home/linda:/bin/bash
mike:x:1002:1004::/home/mike:/bin/bash
paul:x:1003:1005::/home/paul:/bin/bash
boris:x:1004:1006::/home/boris:/bin/bash
pete:x:1005:1008::/home/pete:/bin/bash
[root@server1 ~]#
```

Using tr

- **tr** helps in transforming strings
- **echo hello | tr [a-z] [A-Z]**
- **echo hello | tr [:lower:] [:upper:]**

Exercise 4

- Create a script that transforms the string cn=lara,dc=example,dc=com in a way that the user name (lara) is extracted from the string. Also make sure that the result is written in all lowercase. Store the username in the variable USER and at the end of the script, echo the value of this variable.

Exercise 4 Solution

```
#!/bin/bash

USER=cn=lara,dc=example,dc=com

USER=${USER%%,*}

USER=${USER#*=}

USER=$(echo $USER | tr [:lower:] [:upper:])

echo the username is $USER
```