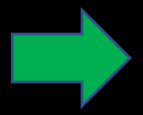
Operating Systems

Lab 04

Content



Content
Manipulating Files and Variables
Redirection
Pipelines and Filters
Permissions

Manipulating Files and Variables

$diff$ command

- Allows you to compare two directories and files line by line.
  - describes the positions and types of changes required to convert the first file to the second file.
- Compare f1.txt to f2.txt
  - $1d0$ and $4a4$ are change commands.
  - Each change command contains the following, from left to right:
    - line number, or range of numbers, corresponding to the first file
    - a letter (a for add, c for change, or d for delete)
    - line number, or range of numbers, corresponding to the second file.

```
lvl3@lvl3-vm:~/Desktop$ cat > f1.txt
a
b
c
d
lvl3@lvl3-vm:~/Desktop$ cat > f2.txt
b
c
d
e
lvl3@lvl3-vm:~/Desktop$ diff f1.txt f2.txt
1d0
< a
4a4
> e
```

Manipulating Files and Variables

- $diff$ change commands

Change	Description
`r1ar2`	Append the lines at the position *r2* in the second file to the position *r1* in the first file.
`r1cr2`	Change (replace) the lines at position *r1* with the lines at the position *r2* in the second file.
`r1dr2`	Delete the lines in the first file at position *r1*, which would have appeared at range *r2* in the second file

Manipulating Files and Variables

- The output here means:
  - Delete line 1 from the first file, to match line 0 (does not exist) in the second file.
    - The character to be removed (<) is a
  - Add line 4 to the first file, to match line 4 in the second file.
    - The character to be added (>) is e

```
lvl3@lvl3-vm:~/Desktop$ cat > f1.txt
a
b
c
d
lvl3@lvl3-vm:~/Desktop$ cat > f2.txt
b
c
d
e
lvl3@lvl3-vm:~/Desktop$ diff f1.txt f2.txt
1d0
< a
4a4
> e
```

Manipulating Files and Variables

- To compare directories

```
lvl3@lvl3-vm:~/Desktop$ diff dir1 dir4
Only in dir1: a.txt
Only in dir1: b.txt
Only in dir4: dir2
Only in dir4: file2.txt
Only in dir4: file.txt
Only in dir4: new.txt
```

Manipulating Files and Variables

- Compare a file to a directory
  - The file does not exist in the directory.

```
lvl3@lvl3-vm:~/Desktop$ diff b.txt dir4
diff: dir4/b.txt: No such file or directory
```

  - The file exist in the directory, but no changes. (no output)

```
lvl3@lvl3-vm:~/Desktop$ diff b.txt dir1
```

  - The file exist in the directory with changes

```
lvl3@lvl3-vm:~/Desktop$ diff b.txt dir1
1,3d0
< a
< x
<
```

Manipulating Files and Variables

$passwd$ command

- Used to change the user account passwords.
  - The root user can change the password for any user on the system
  - A normal user can only change the account password for his or her own account.

```
lvl3@lvl3-vm:~/Desktop$ passwd
Changing password for lvl3.
Current password:
New password:
Retype new password:
passwd: password updated successfully
```

  - To abort password change command, press $CTRL + U$ followed by $CTRL + D$.

Do NOT change it!

Manipulating Files and Variables

- We can see the user's password status using the –S option.

```
lvl3@lvl3-vm:~/Desktop$ passwd -S lvl3
lvl3 P 03/18/2022 0 99999 7 -1
```

  - The first field is the user's login name.
  - The second field indicates if the user account has a locked password (L), has no Password (NP), or has a usable password (P).
  - The third field gives the date of the last password change.
  - The next four fields are the minimum age, maximum age, warning period, and inactivity period for the password.
    - These ages are expressed in days.

Manipulating Files and Variables

alias command

- Replaces one string with another string while executing the commands.

```
lvl3@lvl3-vm:~/Desktop$ alias tt="ls"
lvl3@lvl3-vm:~/Desktop$ tt
b.txt  dir3  dir5    f2.txt  file.txt  i2.txt  new.txt  w.txt  xyz.txt
dir1   dir4  f1.txt  f3.txt  i2.jpg    mydir   o.txt    xyz
```

- Be CAREFULL, do not replace a command name by an alias.

How to fix that ??
Search google

```
lvl3@lvl3-vm:~/Desktop$ alias ls="tt"
lvl3@lvl3-vm:~/Desktop$ ls

Command 'tt' not found, but can be installed with:

sudo apt install treetop

lvl3@lvl3-vm:~/Desktop$ tt

Command 'tt' not found, but can be installed with:

sudo apt install treetop
```

Manipulating Files and Variables

sort command

- Sorts the contents of a file.

```
lvl3@lvl3-vm:~/Desktop$ cat > f.tx
Sayed
Ali
Mostafa
Belal
Xerox
abanob
```

```
lvl3@lvl3-vm:~/Desktop$ sort f.tx
abanob
Ali
Belal
Mostafa
Sayed
Xerox
```

- Use the $-r$ to reverse the output

```
lvl3@lvl3-vm:~/Desktop$ sort -r f.tx
Xerox
Sayed
Mostafa
Belal
Ali
abanob
```

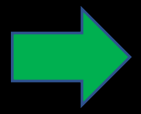
Manipulating Files and Variables

- We can sort multiple files in one line.

```
lvl3@lvl3-vm:~/Desktop$ sort f.tx new.txt xyz.txt o.txt

abanob
Ali
Belal
Goodbye
hello guys,
kjdisfiuhsfgs
Mostafa
ohidfuysgiygsf
Sayed
sdgsohfiushsgsfghsouhfs\
this is my text
welcome to Linux course.
Xerox
```

Content



Content
Manipulating Files and Variables
Redirection
Pipelines and Filters
Permissions

Redirection

- In Linux “everything is a file”
  - Programs send its output to a special file called *standard output (stdout)*.
  - Programs send its status messages of errors to a special file called *standard error (stderr)*.
  - Programs take input from a special file called *standard input (stdin)*.
- By default, both standard output and standard error are linked to the screen and not saved into a disk file.
- By default, the standard input is the keyboard.
- I/O redirection allows us to change where output goes and where input comes from.

Redirection

- We can redirect the output to another file using “>” symbol.
  - Save the result of the $sort$ command to a text file.

```
lvl3@lvl3-vm:~/Desktop$ sort -r f.tx > sorted_file.txt
```

  - Save the result of the $ls - l$ command to a file, then open the file using $nano$ command.

```
lvl3@lvl3-vm:~/Desktop$ ls -l /usr/bin > ls-output.txt
lvl3@lvl3-vm:~/Desktop$ nano ls-output.txt
```

  - We can append a data to an existing file instead of overwriting it using >> symbol.

```
lvl3@lvl3-vm:~/Desktop$ ls -l ~/Desktop/ >> ls-output.txt
lvl3@lvl3-vm:~/Desktop$ nano ls-output.txt
```

Redirection

- Let's try redirect the error message of *ls* to a file.
  - Refer to a directory that does not exist.

```
lvl3@lvl3-vm:~/Desktop$ ls /usr/jjjj > err_file.txt
ls: cannot access '/usr/jjjj': No such file or directory
```

The output is printed on the console, and the text file is empty!

- To redirect the standard error, we must refer to its *file descriptor*.
  - The shell references the standard input by file descriptor 0.
  - The shell references the standard output by the file descriptor 1
  - The shell references the standard error by the file descriptor 2.

Redirection

- We can redirect standard error with this notation:

```
lvl3@lvl3-vm:~/Desktop$ ls /usr/jjjj 2> err_file.txt
lvl3@lvl3-vm:~/Desktop$ more err_file.txt
ls: cannot access '/usr/jjjj': No such file or directory
```

  - The file descriptor “2” is placed immediately before the redirection operator.
- We can redirect the output and error to the same file.

```
lvl3@lvl3-vm:~/Desktop$ ls -l /usr/jjjj > out_err.txt 2>&1
lvl3@lvl3-vm:~/Desktop$ ls -l ~/Desktop/ >> out_err.txt 2>&1
```

  - When you open the *out_err* file, you will see the error message of the first command, then the output of the second command appended to it because we use the >> operator

Redirection

- We can redirect input using “<“ operator.

```
lvl3@lvl3-vm:~/Desktop$ cat < o.txt
ohidfuysgiygsf
sdgsohfiushsgsfghsouhfs\
kjdisfiuhsfgs
```

  - The result is the same as without using “<“, so it is useless.

```
lvl3@lvl3-vm:~/Desktop$ cat o.txt
ohidfuysgiygsf
sdgsohfiushsgsfghsouhfs\
kjdisfiuhsfgs
```

Redirection

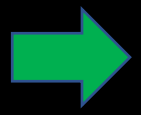
- Use the input and output redirection in one command line.

```
lvl3@lvl3-vm:~/Desktop$ sort < f.tx > new_result_file.txt
lvl3@lvl3-vm:~/Desktop$ more new_result_file.txt
abanob
Ali
Belal
Mostafa
Sayed
Xerox
```

  - In this example, we redirect the input of the $f.tx$ file to the $sort$ command, then redirect the output to the new_result_file.

Content

Content
Manipulating Files and Variables
Redirection
Pipelines and Filters
Permissions



Pipelines and Filters

- The pipeline operator "|" connects the output of one command with the input of a second command.
  - The standard output of one command can be piped into the standard input of another.
- Example, using the $ls - l\ /usr/bin/$command shows very long output.
  - Use the pipe operator to pass the output to $less$ command to show the output page by page.

```
lvl3@lvl3-vm:~/Desktop$ ls -l /usr/bin/ | less
```

Pipelines and Filters

- Filters take input, change it somehow, and then output it.
- Example, make a combined list of all the executable programs in $/usr/bin$, put them in sorted order and view the resulting list.

```
lvl3@lvl3-vm:~/Desktop$ ls /usr/bin | sort | less
```

- We can specify more than one directory.

```
lvl3@lvl3-vm:~/Desktop$ ls . /home | sort | less
```

  - Remember, the "." refers to the current working directory, which is "~/Desktop"

Pipelines and Filters

$uniq$ command

- Removes any duplicates from the list.

```
lvl3@lvl3-vm:~/Desktop$ cat > a1.txt
aa
aa
aa
xyz
hello
omar
ahmed
Zain
lvl3@lvl3-vm:~/Desktop$ uniq a1.txt
aa
xyz
hello
omar
ahmed
Zain
```

Pipelines and Filters

- Use pipelines to read a file, sort the content and remove duplicates.
  - Without using *uniq*:

```
lvl3@lvl3-vm:~/Desktop$ ls /bin /usr/bin | sort | less
```

  - With using *uniq*

```
lvl3@lvl3-vm:~/Desktop$ ls /bin /usr/bin | sort | uniq | less
```

```
[
[
aa-enabled
aa-enabled
aa-exec
aa-exec
aconnect
aconnect
acpi_listen
acpi_listen
```

```
[
aa-enabled
aa-exec
aconnect
acpi_listen
add-apt-repository
addpart
alsabat
alsaloop
alsamixer
```

Pipelines and Filters

wc command

- The wc (word count) command is used to display the number of lines, words, and bytes contained in files.

```
lvl3@lvl3-vm:~/Desktop$ wc ls-output.txt
 1364 12674 86542 ls-output.txt
```

  - it prints out three numbers: lines, words, and bytes contained in $ls - output.txt$.
- Use $-l$ option to print number of lines only.

```
lvl3@lvl3-vm:~/Desktop$ wc -l ls-output.txt
1364 ls-output.txt
```

Pipelines and Filters

grep command

- Used to find text patterns within files.

```
grep pattern [file...]
```

- Example, find any line in the */etc/passwd* file that contains word “usr”

```
lvl3@lvl3-vm:~/Desktop$ grep usr /etc/passwd
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

Pipelines and Filters

- Use pipelines to find all the files in our list of programs that had the word zip embedded in the name.

```
lvl3@lvl3-vm:~/Desktop$ ls /bin /usr/bin | sort | uniq | grep zip
bunzip2
bzip2
bzip2recover
funzip
gpg-zip
gunzip
gzip
mzip
preunzip
```

Pipelines and Filters

head and *tail* commands

- The *head* command prints the first ten lines of a file.

```
lvl3@lvl3-vm:~/Desktop$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

- The *tail* command prints the last ten lines of a file.

```
lvl3@lvl3-vm:~/Desktop$ tail /etc/passwd
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bi
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,
login
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nol
```

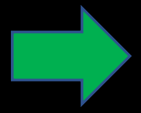
Pipelines and Filters

- Specify the number of lines to print using the $-n$ option.

```
lvl3@lvl3-vm:~/Desktop$ ls /usr/bin | tail -n 5
zipsplit
zjsdecode
zless
zmore
znew
```

Content

Content
Manipulating Files and Variables
Redirection
Pipelines and Filters
Permissions



Permissions

- Linux is a multi-user operating system.
  - It means that more than one person can be using the computer at the same time.
  - For example, if a computer is attached to a network, remote users can log in via ssh (secure shell) and operate the computer.
- We will look at part of system security and introduce the following commands:
  - id – Display user identity
  - $chmod$ – Change a file's mode
  - $umask$ – Set the default file permissions

Permissions

id command

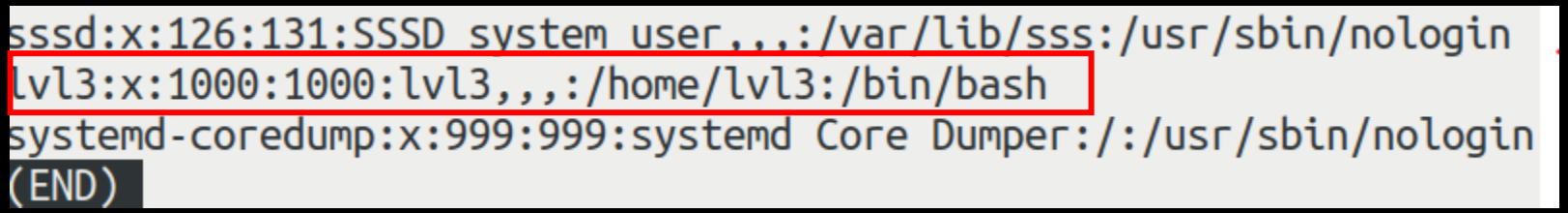
- Prints user and group information.

```
lvl3@lvl3-vm:~/Desktop$ id
uid=1000(lvl3) gid=1000(lvl3) groups=1000(lvl3),4(adm),24(cdrom),27(sudo)
,30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare)
```

  - uid is the current user ID.
  - gid is the primary group ID.
  - $groups$ is the groups that the user belongs to.
  - Other data are groups assigned to the current user, because Ubuntu manages privileges for system devices and services.

Permissions

- The file /etc/passwd contains user accounts. `less /etc/passwd`



  - Username: Password: UID: GID: account's name: Home directory: Login shell
  - The username field is the name used for login.
  - The x character in the password field means that the password is stored in /etc/shadow file in the encrypted format.
- The file /etc/group contains groups registered on the system.
- The file /etc/shadow contains information about the users password.

Permissions

- To see permissions of a file or a directory

```
lvl3@lvl3-vm:~/Desktop$ ls -l xyz.txt
-rw-rw-r-- 1 lvl3 lvl3 45 15:38 6  مار xyz.txt
```

Owner	Group	World
rwx	rwx	rwx

Attribute	Files	Directories
r	Allows a file to be opened and read.	Allows a directory's contents to be listed if the execute attribute is also set.
w	Allows a file to be written to or truncated, however this attribute does not allow files to be renamed or deleted. The ability to delete or rename files is determined by directory attributes.	Allows files within a directory to be created, deleted, and renamed if the execute attribute is also set.
x	Allows a file to be treated as a program and executed. Program files written in scripting languages must also be set as readable to be executed.	Allows a directory to be entered, e.g., cd *directory*.

Permissions

$chmod$ command – Change File Mode

- Changes the mode (permissions) of a file or directory.
  - only the file's owner or the superuser can change the mode of a file or directory.
- We can use the $chmod$ command in two ways:
  - octal number representation,
  - symbolic representation.

Permissions

- We use octal numbers to set the pattern of desired permissions.
  - An octal number represents three binary digits.
  - By using three octal digits, we can set the file mode for the owner, group owner, and world.
- Examples:
  - 600 = rw- --- ---
  - 777 = rwx rwx rwx
  - 123 = --x -w- -wx
  - 456 = r-- r-x rw-

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Permissions

- Example, give the read and write permissions to the user and remove all permissions from the group owner and world.
  - Use the permission 600.

```
lvl3@lvl3-vm:~/Desktop$ > dummy.txt
lvl3@lvl3-vm:~/Desktop$ ls -l dummy.txt
-rw-rw-r-- 1 lvl3 lvl3 0 21:34 26 مار dummy.txt
lvl3@lvl3-vm:~/Desktop$ chmod 600 dummy.txt
lvl3@lvl3-vm:~/Desktop$ ls -l dummy.txt
-rw------- 1 lvl3 lvl3 0 21:34 26 مار dummy.txt
```

Permissions

- Symbolic notation is divided into three parts.
  - Who the change will affect
    - u Short for “user” but means the file or directory owner.
    - g Group owner.
    - o Short for “others” but means world.
    - a Short for “all.” This is the combination of “u”, “g”, and “o”.
    - If no character is specified, “all” will be assumed.
  - Which operation will be performed
    - + means add a permission
    - - means remove a permission
    - = means add only the specified permissions and remove the others
  - What permission will be set.
    - r, w, x

Permissions

- Example

Notation	Meaning
`u+x`	Add execute permission for the owner.
`u-x`	Remove execute permission from the owner.
`+x`	Add execute permission for the owner, group, and world. This is equivalent to `a+x`.
`o-rw`	Remove the read and write permissions from anyone besides the owner and group owner.
`go=rw`	Set the group owner and anyone besides the owner to have read and write permission. If either the group owner or the world previously had execute permission, it is removed.
`u+x,go=rx`	Add execute permission for the owner and set the permissions for the group and others to read and execute. Multiple specifications may be separated by commas.

Permissions

- Example, give user read, write, and execute permissions. Give the group the read permission only and remove any other permission. Remove the write permission from the others.
  - $chmod\ u + rwx, g = r, o - w\ xyz.txt$
  - NO SPACE BETWEEEN COMMAS!

```
lvl3@lvl3-vm:~/Desktop$ chmod 777 xyz.txt
lvl3@lvl3-vm:~/Desktop$ ls -l xyz.txt
-rwxrwxrwx 1 lvl3 lvl3 45 15:38 6  مار xyz.txt
lvl3@lvl3-vm:~/Desktop$ chmod u+rwx,g=r,o-w xyz.txt
lvl3@lvl3-vm:~/Desktop$ ls -l xyz.txt
-rwxr--r-x 1 lvl3 lvl3 45 15:38 6  مار xyz.txt
```

Permissions

$umask$ – Set Default Permissions

- It controls the default permissions given to a file when it is created.
- It uses octal notation to express a mask of bits to be removed from a file's mode attributes.

```
lvl3@lvl3-vm:~/Desktop$ umask
0002
```

- When we create a new file, the user and the group get read and write permissions, while the world get read permission only.
  - This is because the value of the mask.

Permissions

How the mask works?

- The mask is represented in octal numbers.
- The default permissions of a file are: rw- rw- rw-
- Expand the permission with the binary of the mask
  - When a permission corresponds to a 1, it is removed.
  - This corresponds to (default_mode & ~mask)

Original file mode		`--- rw- rw- rw-`
Mask	**0002**	`000 000 000 010`
Result		`--- rw- rw- r--`

Original file mode		`--- rw- rw- rw-`
Mask	**0022**	`000 000 010 010`
Result		`--- rw- r-- r--`

Permissions

- Example, set the mask to 0022.

```
lvl3@lvl3-vm:~/Desktop$ umask 0022
lvl3@lvl3-vm:~/Desktop$ > foo.txt
lvl3@lvl3-vm:~/Desktop$ ls -l foo.txt
-rw-r--r-- 1 lvl3 lvl3 0 23:04 26 مار foo.txt
```

- Example, make the mask to give read permissions only to user and group and read and write permissions only to the world.
  - umask 0220

```
lvl3@lvl3-vm:~/Desktop$ umask 0220
lvl3@lvl3-vm:~/Desktop$ > new_file.txt
lvl3@lvl3-vm:~/Desktop$ ls -l new_file.txt
-r--r--rw- 1 lvl3 lvl3 0 23:16 26 مار new_file.txt
```

Permissions

- Can we define the mask with a execute permission?

Permissions

- We cannot define the mask with a execute permission. WHY?
- Since when the execution bits in file default mode are 0, perform bitwise AND “&” with 0 is always 0.
- You can view all possible permissions: https://www.linuxtrainingacademy.com/all-umasks/

Exercise

- Create $file1.txt$ that has the following content:
  Ubuntu
  Arch Linux
  Debian
  CentOS
  Fedora
- Create $file2.txt$ that has the following content:
  Kubuntu
  Ubuntu
  Debian
  Arch Linux
  Centos
  Fedora

Exercise

- Run the $diff$ tool between $file1.txt$ and $file2.txt$. Explain the output meaning.
- How to let $sort$ command to ignore blanks?
- Redirect the input of the ls command from $/usr$ directory and redirect the output to ls_res file.
- Display the first 8 lines of the /etc/passwd file.
- Copy the /etc/group file to the desktop folder with new name “grps_file”.
- Set the permissions of the “grps_file” to read and execute for the user and remove any permissions from the group and set the world to write only. (use two different ways)

Exercise

- Run the $diff$ tool between $file1.txt$ and $file2.txt$. Explain the output meaning.
- How to let $sort$ command to ignore blanks? Using –b operator.
- Redirect the input of the ls command from $/usr$ directory and redirect the output to ls_res file. ls < /usr/ > ls_res.txt
- Display the first 8 lines of the /etc/passwd file. head -n 8 /etc/passwd
- Copy the /etc/groups file to the desktop folder with new name “grps_file”.
  - cp /etc/group ./grps_file
- Set the permissions of the “grps_file” to read and execute for the user and remove any permissions from the group and set the world to write only. (use two different ways)
  - chmod 502 grps_file AND chmod u+r-w+x,g-rwx,o-r+w-x grps_file

Summary

- $diff$
- $passwd$
- $alias$
- $sort$
- < and > symbols
- | symbol
- $grep$
- $uniq$
- wc
- $head$
- $tail$
- id
- $chmod$
- $umask$

TASK

- What is the $-c$ option in $diff$ tool? Explain the output format.
- What is the $whatis$ command in Linux?
- Use the manual of $grep$ command. What $-i$ and $-v$ used for?
  - Apply the $-v$ command with pipelines to display set of programs in /usr/bin directory that does not contain word “zip”.
- How to set the default mask to rw--w--w-.