

Basic tools : Introduction to Linux & GitHub

KASHTANOVA Victoriya

Inria, Epione

Linux is not equal to Unix

Operating system (OS) is system software that manages computer hardware, software resources, and provides common services for

Unix is the family of OS that derive from the original AT&T Unix (Bell labs, 1970)

Unix OS are :

- Multiuser
- Multitasking
- Respect POSIX standards

Examples :

- Mac OS X
- Android
- GNU/Linux

* Microsoft Windows family is another type of OS



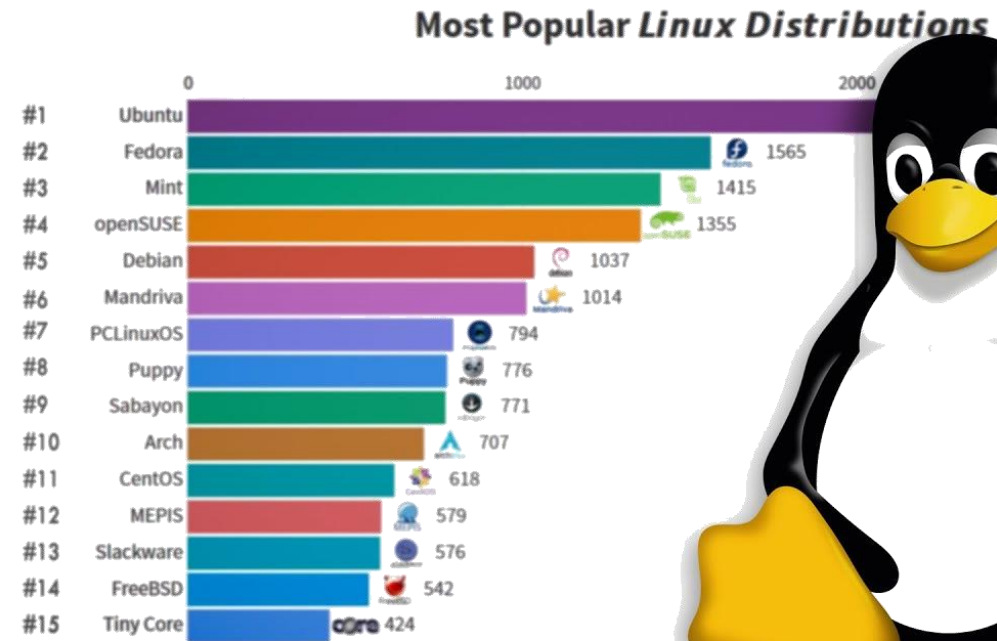
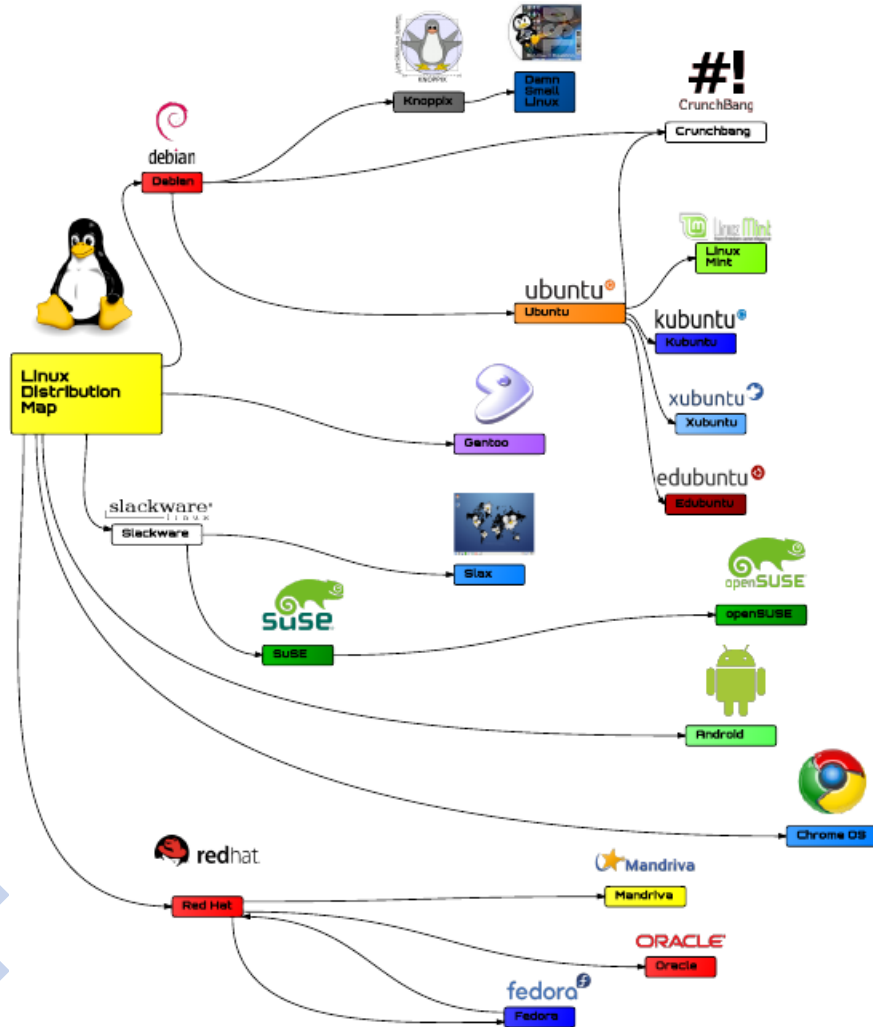
Linux is not equal to Unix

Linux = Kernel = Deepest layer of OS

Linux is free software which gives freedom to use, to study, to modify source code
→ wealth (or dispersion ?) of the free software ecosystem



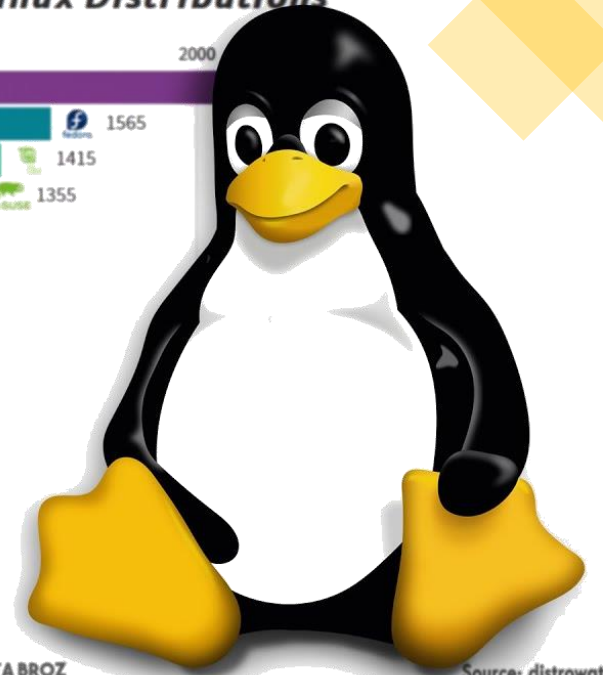
Popular Linux distributions



Values: Hits Per Day in Distrowatch

DATA BROZ

Source: distrowatch



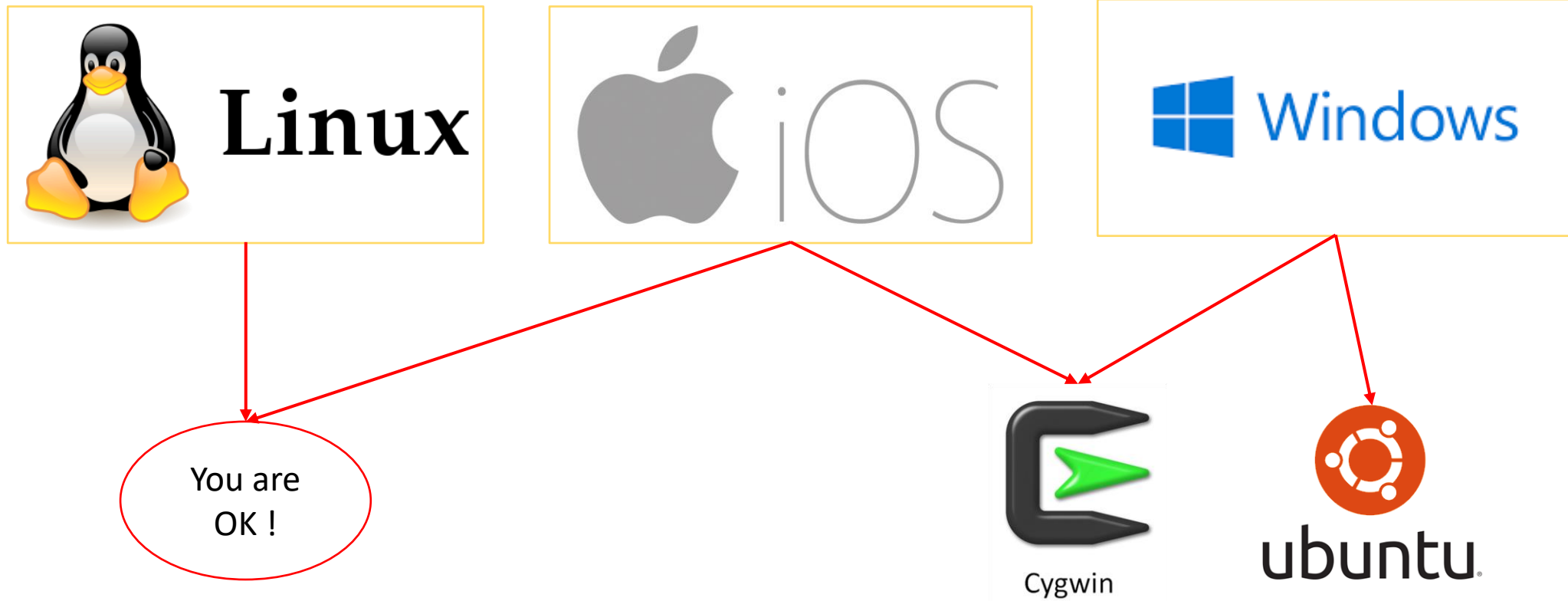
Why do we need to use Linux ?



Google Cloud Platform



What is your Operating System ?



Bash

- **Bash is a Unix shell and command language.**
- **Bash** is a command processor that typically runs in a text window where the user types commands that cause actions. Bash can also read and execute commands from a file, called a shell script.

To check that you use Bash, enter : **echo \$SHELL**



```
vkash@DESKTOP-3CT1O9E: ~  
vkash@DESKTOP-3CT1O9E:~$ echo $SHELL  
/bin/bash  
vkash@DESKTOP-3CT1O9E:~$
```

File system commands

cd – command to go to a specific folder

pwd - command to get a path of the current working directory

ls - command to get a list of contents of the current working directory
(or of any directory if we use ls + "directory path")

mkdir – command to create a new directory/directories

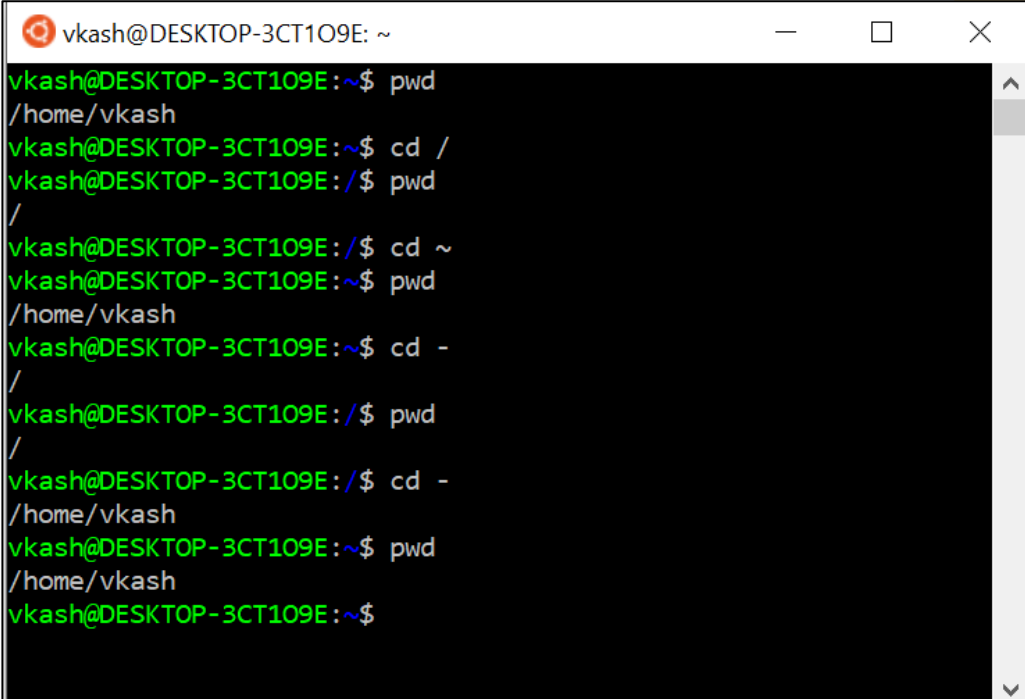
cp - command to copy a file

mv – command to rename or to move a file/files

ln - command to create a link to a file

rm – command to remove a file/files

rmdir - – command to remove a directory



```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ cd /  
vkash@DESKTOP-3CT109E:/$ pwd  
/  
vkash@DESKTOP-3CT109E:/$ cd ~  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ cd -  
/  
vkash@DESKTOP-3CT109E:/$ pwd  
/  
vkash@DESKTOP-3CT109E:/$ cd -  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$
```


File system commands

cd – command to go to a specific folder

pwd - command to get a path of the current working directory

ls - command to get a list of contents of the current working directory
(or of any directory if we use ls + "directory path")

mkdir – command to create a new directory/directories

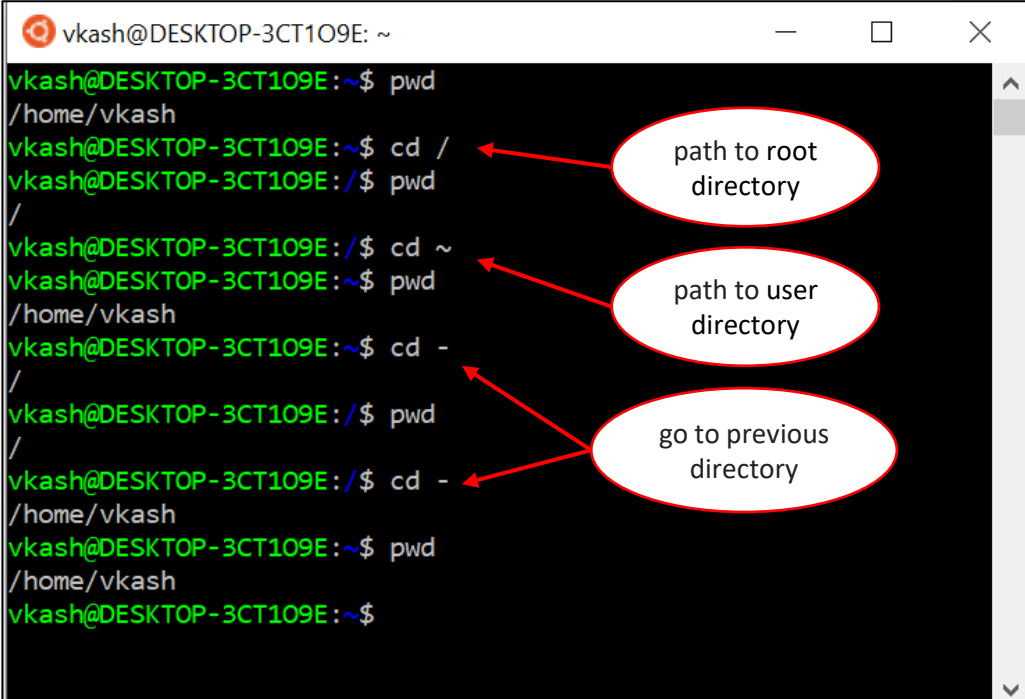
cp - command to copy a file

mv – command to rename or to move a file/files

ln - command to create a link to a file

rm – command to remove a file/files

rmdir - – command to remove a directory



```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ cd /  
vkash@DESKTOP-3CT109E:/$ pwd  
/  
vkash@DESKTOP-3CT109E:/$ cd ~  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ cd -  
/  
vkash@DESKTOP-3CT109E:/$ pwd  
/  
vkash@DESKTOP-3CT109E:/$ cd -  
/home/vkash  
vkash@DESKTOP-3CT109E:~$ pwd  
/home/vkash  
vkash@DESKTOP-3CT109E:~$
```

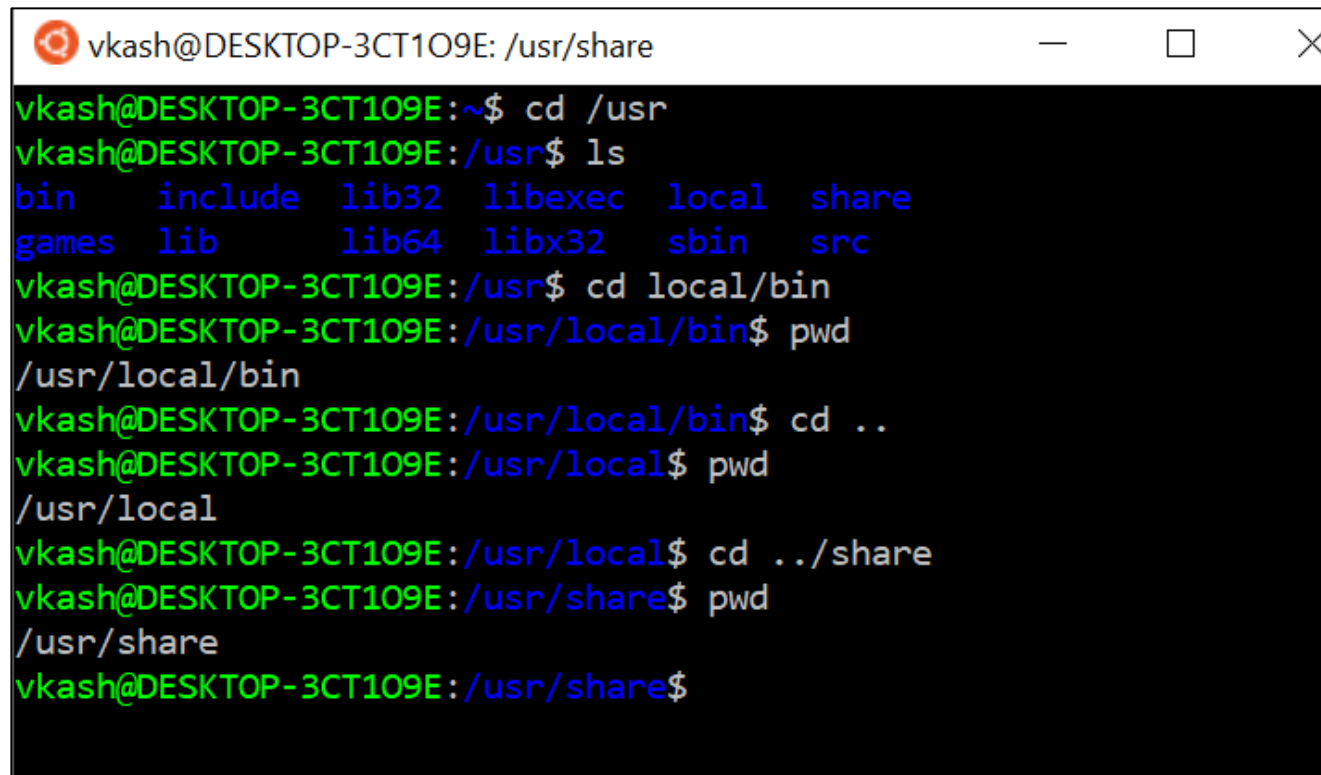
Annotations:

- path to root directory (points to `cd /`)
- path to user directory (points to `cd ~`)
- go to previous directory (points to `cd -`)

File system specificity

Absolute path: /repertoire/sous_repertoire/sous_sous_repertoire/fichier_ou_dossier

Relative path: sous_repertoire/fichier_ou_dossier (depends on current directory)

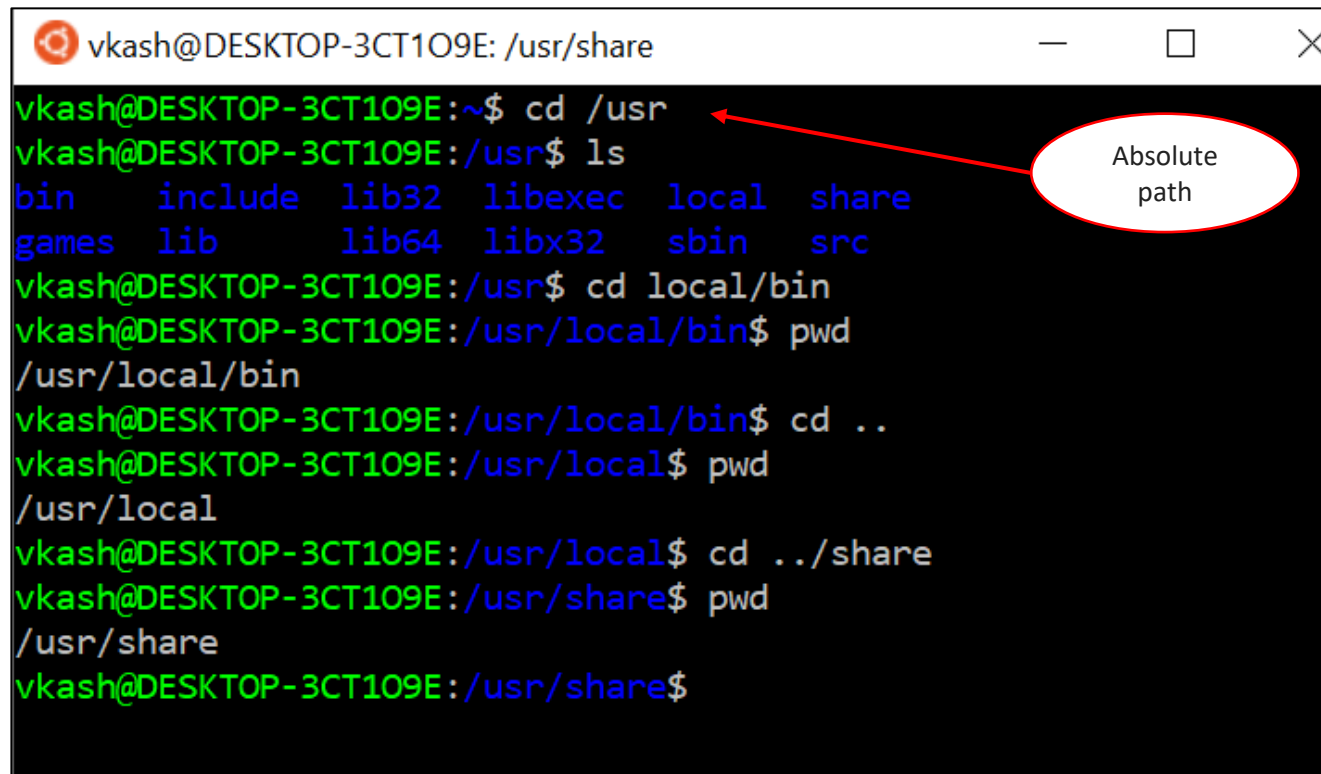


```
vkash@DESKTOP-3CT109E: /usr/share
vkash@DESKTOP-3CT109E:~$ cd /usr
vkash@DESKTOP-3CT109E:/usr$ ls
bin    include  lib32    libexec  local    share
games  lib      lib64    libx32   sbin     src
vkash@DESKTOP-3CT109E:/usr$ cd local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ pwd
/usr/local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ cd ..
vkash@DESKTOP-3CT109E:/usr/local$ pwd
/usr/local
vkash@DESKTOP-3CT109E:/usr/local$ cd ../share
vkash@DESKTOP-3CT109E:/usr/share$ pwd
/usr/share
vkash@DESKTOP-3CT109E:/usr/share$
```

File system specificity

Absolute path: /repertoire/sous_repertoire/sous_sous_repertoire/fichier_ou_dossier

Relative path: sous_repertoire/fichier_ou_dossier (depends on current directory)



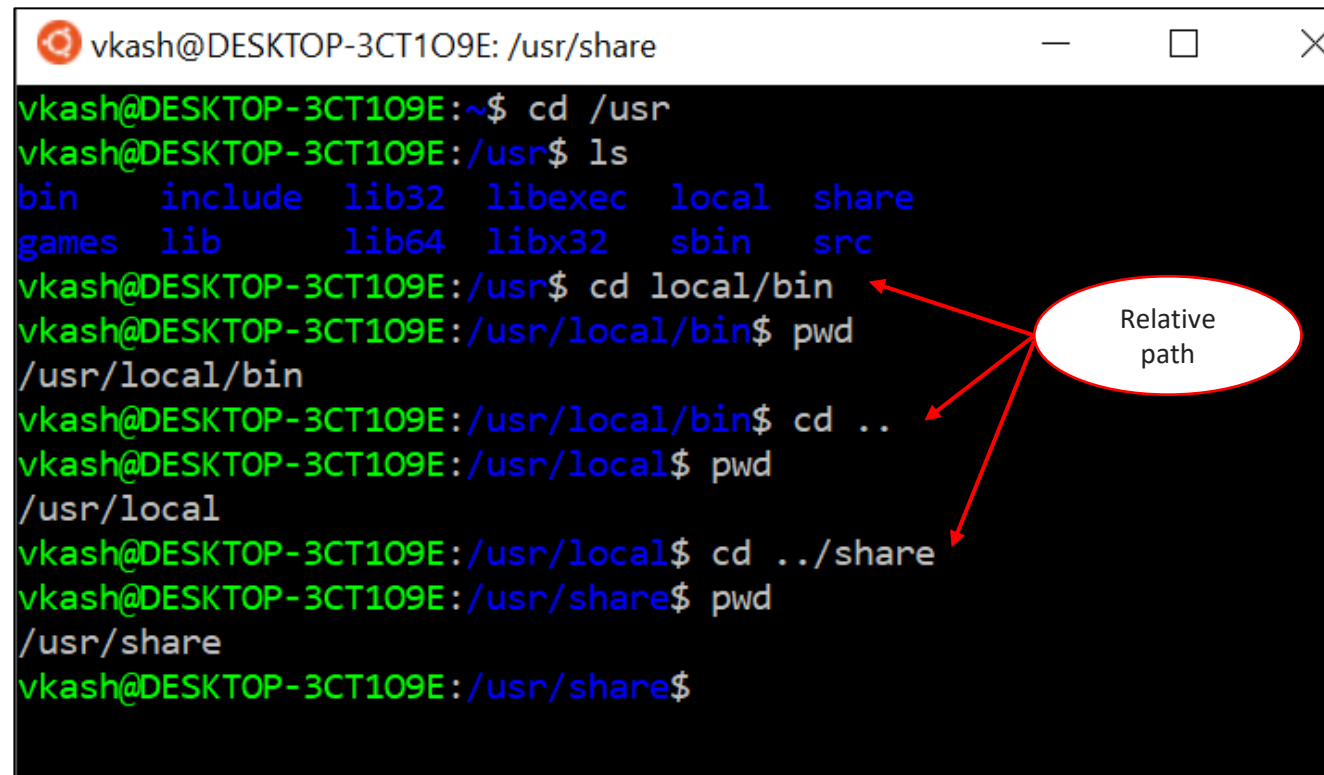
```
vkash@DESKTOP-3CT109E: /usr/share
vkash@DESKTOP-3CT109E:~$ cd /usr
vkash@DESKTOP-3CT109E:/usr$ ls
bin    include  lib32    libexec  local    share
games  lib      lib64    libx32   sbin     src
vkash@DESKTOP-3CT109E:/usr$ cd local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ pwd
/usr/local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ cd ..
vkash@DESKTOP-3CT109E:/usr/local$ pwd
/usr/local
vkash@DESKTOP-3CT109E:/usr/local$ cd ../share
vkash@DESKTOP-3CT109E:/usr/share$ pwd
/usr/share
vkash@DESKTOP-3CT109E:/usr/share$
```

A red arrow points from a red oval containing the text "Absolute path" to the `/usr` in the first command `cd /usr`.

File system specificity

Absolute path: /repertoire/sous_repertoire/sous_sous_repertoire/fichier_ou_dossier

Relative path: sous_repertoire/fichier_ou_dossier (depends on current directory)



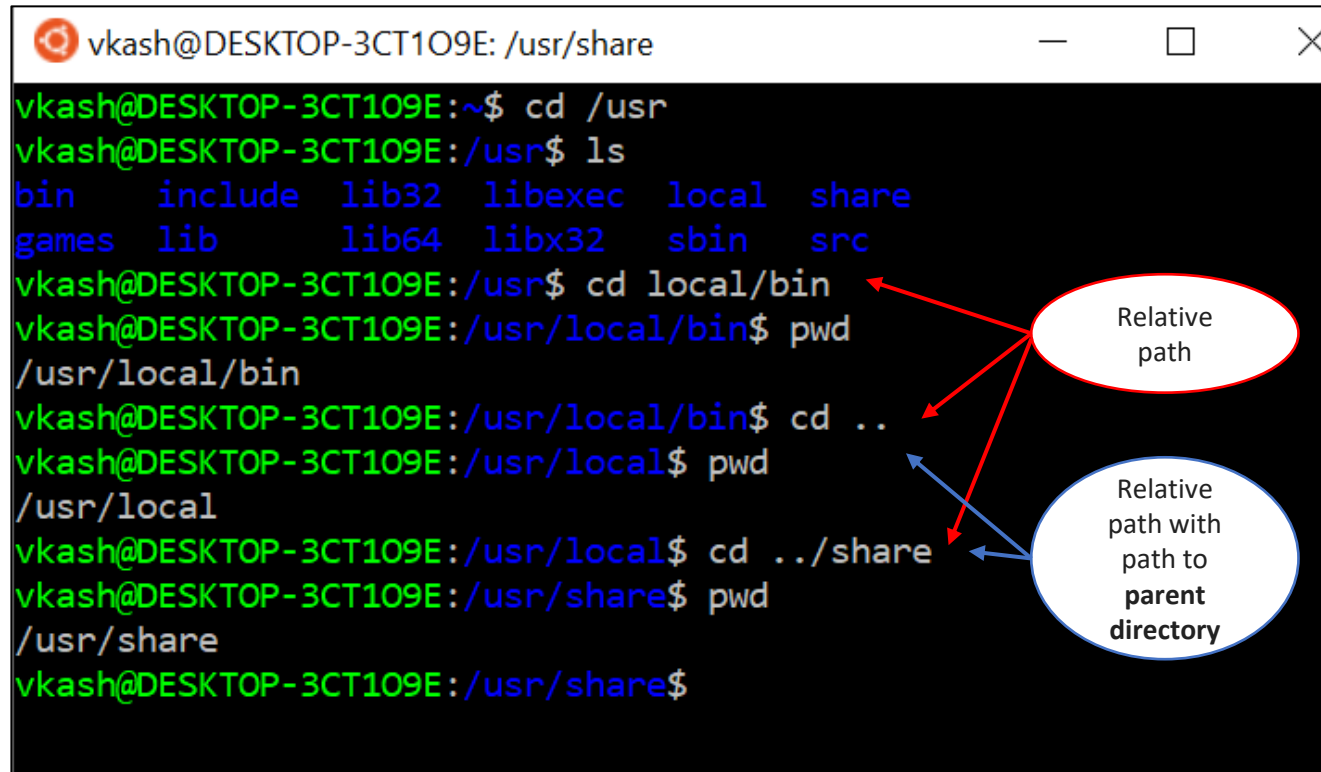
```
vkash@DESKTOP-3CT109E: /usr/share
vkash@DESKTOP-3CT109E:~$ cd /usr
vkash@DESKTOP-3CT109E:/usr$ ls
bin    include  lib32    libexec  local    share
games  lib      lib64    libx32   sbin     src
vkash@DESKTOP-3CT109E:/usr$ cd local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ pwd
/usr/local/bin
vkash@DESKTOP-3CT109E:/usr/local/bin$ cd ..
vkash@DESKTOP-3CT109E:/usr/local$ pwd
/usr/local
vkash@DESKTOP-3CT109E:/usr/local$ cd ../share
vkash@DESKTOP-3CT109E:/usr/share$ pwd
/usr/share
vkash@DESKTOP-3CT109E:/usr/share$
```

A terminal window titled "vkash@DESKTOP-3CT109E: /usr/share" showing a series of directory navigation commands. The user starts at the home directory, moves to /usr, lists its contents, moves to /usr/local/bin, and then uses relative paths (..) to move up to /usr/local and then to /usr/share. A red oval labeled "Relative path" has three arrows pointing to the relative path commands: "cd .." from /usr/local/bin to /usr/local, and "cd ../share" from /usr/local to /usr/share.

File system specificity

Absolute path: /repertoire/sous_repertoire/sous_sous_repertoire/fichier_ou_dossier

Relative path: sous_repertoire/fichier_ou_dossier (depends on current directory)



```
vkash@DESKTOP-3CT1O9E: /usr/share
vkash@DESKTOP-3CT1O9E:~$ cd /usr
vkash@DESKTOP-3CT1O9E:/usr$ ls
bin    include  lib32    libexec  local    share
games  lib      lib64    libx32   sbin     src
vkash@DESKTOP-3CT1O9E:/usr$ cd local/bin
vkash@DESKTOP-3CT1O9E:/usr/local/bin$ pwd
/usr/local/bin
vkash@DESKTOP-3CT1O9E:/usr/local/bin$ cd ..
vkash@DESKTOP-3CT1O9E:/usr/local$ pwd
/usr/local
vkash@DESKTOP-3CT1O9E:/usr/local$ cd ../share
vkash@DESKTOP-3CT1O9E:/usr/share$ pwd
/usr/share
vkash@DESKTOP-3CT1O9E:/usr/share$
```

Relative path

Relative path with path to parent directory

Exercise : **cd** & **pwd**

What folders will we get after running these commands ?

```
$ cd /bin  
$ cd ../usr/share/zoneinfo      => ?
```

```
$ cd /usr/X11R6/bin  
$ cd ../lib/X11                 => ?
```

```
$ cd /usr/bin  
$ cd ../bin/../bin              => ?
```

Exercise : **cd** & **pwd**

What folders will we get after running these commands ?

```
$ cd /bin
```

```
$ cd ../usr/share/zoneinfo
```

```
=> /usr/share/zoneinfo
```

```
$ cd /usr/X11R6/bin
```

```
$ cd ../lib/X11
```

```
=> /usr/X11R6/lib/X11
```

```
$ cd /usr/bin
```

```
$ cd ../bin/../bin
```

```
=> /usr/bin
```

Exercise : ls

```
vkash@DESKTOP-3CT1O9E: /usr
vkash@DESKTOP-3CT1O9E:~$ cd /usr
vkash@DESKTOP-3CT1O9E:/usr$ ls
bin    include  lib32    libexec  local  share
games  lib      lib64    libx32   sbin   src
vkash@DESKTOP-3CT1O9E:/usr$ ls -a
.  bin    include  lib32    libexec  local  share
.. games  lib      lib64    libx32   sbin   src
vkash@DESKTOP-3CT1O9E:/usr$ man ls_
```

list of
contents

full list of
contents,
hidden
content
included

manual for
command

How to get a full information about all
objects in the directory ?

```
vkash@DESKTOP-3CT1O9E: /usr
LS(1)                                User Commands                                LS(1)
NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory
    by default). Sort entries alphabetically if none of
    -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for
    short options too.

    -a, --all
        do not ignore entries starting with .

Manual page ls(1) line 1 (press h for help or q to quit)
```


Exercise : ls

vkash@DESKTOP-3CT1O9E: ~

```
vkash@DESKTOP-3CT1O9E:~$ ls -l /usr
```

total 0

drwxr-xr-x	1	root	root	4096	Sep 13 21:02	bin
drwxr-xr-x	1	root	root	4096	Apr 15 13:09	games
drwxr-xr-x	1	root	root	4096	Aug 5 00:07	include
drwxr-xr-x	1	root	root	4096	Aug 5 00:07	lib
drwxr-xr-x	1	root	root	4096	Aug 4 23:39	lib32
drwxr-xr-x	1	root	root	4096	Aug 4 23:39	lib64
drwxr-xr-x	1	root	root	4096	Aug 4 23:41	libexec
drwxr-xr-x	1	root	root	4096	Aug 4 23:39	libx32
drwxr-xr-x	1	root	root	4096	Aug 4 23:39	local
drwxr-xr-x	1	root	root	4096	Sep 13 21:02	sbin
drwxr-xr-x	1	root	root	4096	Aug 5 00:07	share
drwxr-xr-x	1	root	root	4096	Aug 4 23:47	src

also works (ls + "path") but doesn't change current directory

"long listing", detailed list of contents

access rights

number of references to this element of file system

list of owners and groups

object size

time of object last modification

object name

Exercise : mkdir & cp & mv

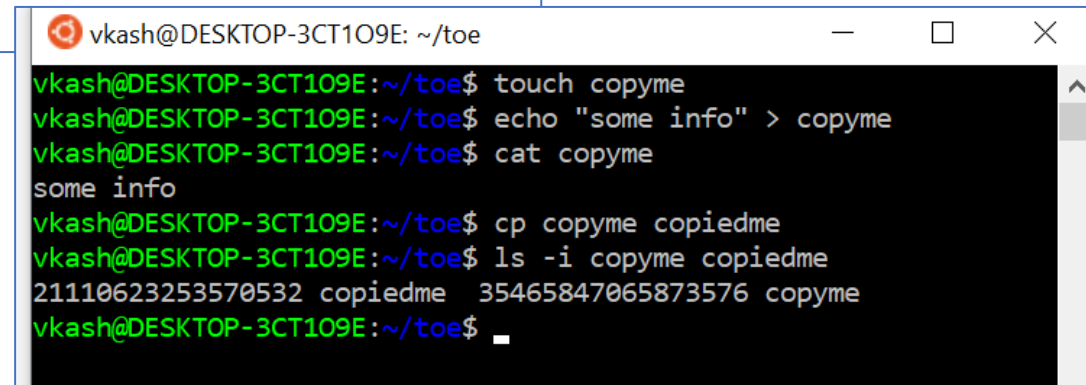
1. Go to your user directory and create 3 new directories : “tic”, “tac” and “toe”
2. Create a new directory “tic/ket”
3. Create a new directory “tac/o/s” (and solve the error)

1. Go to the “toe” directory
2. Create a zero-size file “copyme” using command “touch”
3. *Add in this file some information using command “echo”
4. Display the file content to the terminal using command “cat”
5. Create a new file “copiedme” like a copy of file “copyme”
6. Check that these files are different

Exercise : mkdir & cp & mv

1. Go to your user directory and create 3 new directories : “tic”, “tac” and “toe”
2. Create a new directory “tic/ket”
3. Create a new directory “tac/o/s” (and solve the error)

1. Go to the “toe” directory
2. Create a zero-size file “copyme” using command “touch”
3. *Add in this file some information using command “echo”
4. Display the file content to the terminal using command “cat”
5. Create a new file “copiedme” like a copy of file “copyme”
6. Check that these files are different



```
vkash@DESKTOP-3CT1O9E: ~/toe
vkash@DESKTOP-3CT1O9E:~/toe$ touch copyme
vkash@DESKTOP-3CT1O9E:~/toe$ echo "some info" > copyme
vkash@DESKTOP-3CT1O9E:~/toe$ cat copyme
some info
vkash@DESKTOP-3CT1O9E:~/toe$ cp copyme copiedme
vkash@DESKTOP-3CT1O9E:~/toe$ ls -i copyme copiedme
21110623253570532 copiedme 35465847065873576 copyme
vkash@DESKTOP-3CT1O9E:~/toe$
```

Exercise : mkdir & cp & mv

1. Go to your user directory and create 3 new directories : “tic”, “tac” and “toe”
2. Create a new directory “tic/ket”
3. Create a new directory “tac/o/s” (and solve the error)

1. Go to the “toe” directory
2. Create a zero-size file “copyme” using command “touch”
3. *Add in this file some information using command “echo”
4. Display the file content to the terminal using command “cat”
5. Create a new file “copiedme” like a copy of file “copyme”
6. Check that these files are different

1. Rename “copiedme” to “moveme”
2. Check that it still the same file
3. Move “moveme” to “tac/o/s” directory
4. Check that file is there and not in the current directory

Exercise : mkdir & cp & mv

1. Go to your user directory and create 3 new directories : “tic”, “tac” and “toe”
2. Create a new directory “tic/ket”
3. Create a new directory “tac/o/s” (and solve the error)

1. Go to the “toe” directory
2. Create a zero-size file “copyme” using `touch`
3. *Add in this file some information using `echo`
4. Display the file content to the terminal using `cat`
5. Create a new file “copiedme” like a copy of “copyme” using `cp`
6. Check that these files are different using `ls -li`

1. Rename “copiedme” to “moveme” using `mv`
2. Check that it still the same file using `ls -li`
3. Move “moveme” to “tac/o/s” directory using `mv`
4. Check that file is there and not in the current directory using `ls`

```
vkash@DESKTOP-3CT109E: ~/toe
vkash@DESKTOP-3CT109E:~/toe$ touch copyme
vkash@DESKTOP-3CT109E:~/toe$ echo "some info" > copyme
vkash@DESKTOP-3CT109E:~/toe$ cat copyme
some info
vkash@DESKTOP-3CT109E:~/toe$ cp copyme copiedme
vkash@DESKTOP-3CT109E:~/toe$ ls -li copiedme
21110623253570532 copiedme 35465847065873576 copyme
vkash@DESKTOP-3CT109E:~/toe$ mv copiedme moveme
vkash@DESKTOP-3CT109E:~/toe$ ls -li moveme
21110623253570532 moveme
vkash@DESKTOP-3CT109E:~/toe$ mv moveme ~/tac/o/s
vkash@DESKTOP-3CT109E:~/toe$ ls ../tac/o/s
moveme
vkash@DESKTOP-3CT109E:~/toe$ ls
copyme
vkash@DESKTOP-3CT109E:~/toe$
```

Exercise : rm & rmdir

1. Go to “tic/ket” directory
2. Create 2 zero-size files “file1” and “file2” using “touch” command
3. Delete this files
4. Check that these files are removed

Exercise : rm & rmdir

1. Go to "tic/ket" directory
2. Create 2 zero-size files "file1" and "file2" using "touch" command
3. Delete this files
4. Check that these files are removed

be careful with "rm"
command which
removes file
forever, it's better to
use "rm -i" to control
the process

```
vkash@DESKTOP-3CT109E: ~/tic/ket
vkash@DESKTOP-3CT109E:~/tic/ket$ touch file1 file2
vkash@DESKTOP-3CT109E:~/tic/ket$ ls -l
total 0
-rw-r--r-- 1 vkash vkash 0 Sep 16 11:48 file1
-rw-r--r-- 1 vkash vkash 0 Sep 16 11:48 file2
vkash@DESKTOP-3CT109E:~/tic/ket$ rm -i file1 file2
rm: remove regular empty file 'file1'? y
rm: remove regular empty file 'file2'? y
vkash@DESKTOP-3CT109E:~/tic/ket$ ls -l
total 0
vkash@DESKTOP-3CT109E:~/tic/ket$ ls -l file1 file2
ls: cannot access 'file1': No such file or directory
ls: cannot access 'file2': No such file or directory
vkash@DESKTOP-3CT109E:~/tic/ket$
```

Exercise : rm & rmdir

1. Go to “tic/ket” directory
2. Create 2 zero-size files “file1” and “file2” using “touch” command
3. Delete this files
4. Check that these files are removed

1. Go to your user directory and try to delete “tic” directory
2. Find a method to delete “tic” directory

Exercise : rm & rmdir

1. Go to “tic/ket” directory
2. Create 2 zero-size files “file1” and “file2” using “touch” command
3. Delete this files
4. Check that these files are removed

1. Go to your user directory and try to delete “tic” directory
2. Find a method to delete “tic” directory

- There are 2 options :
- Delete each element in the directory (too slow)
 - Use recursive force of “rm” (too dangerous)

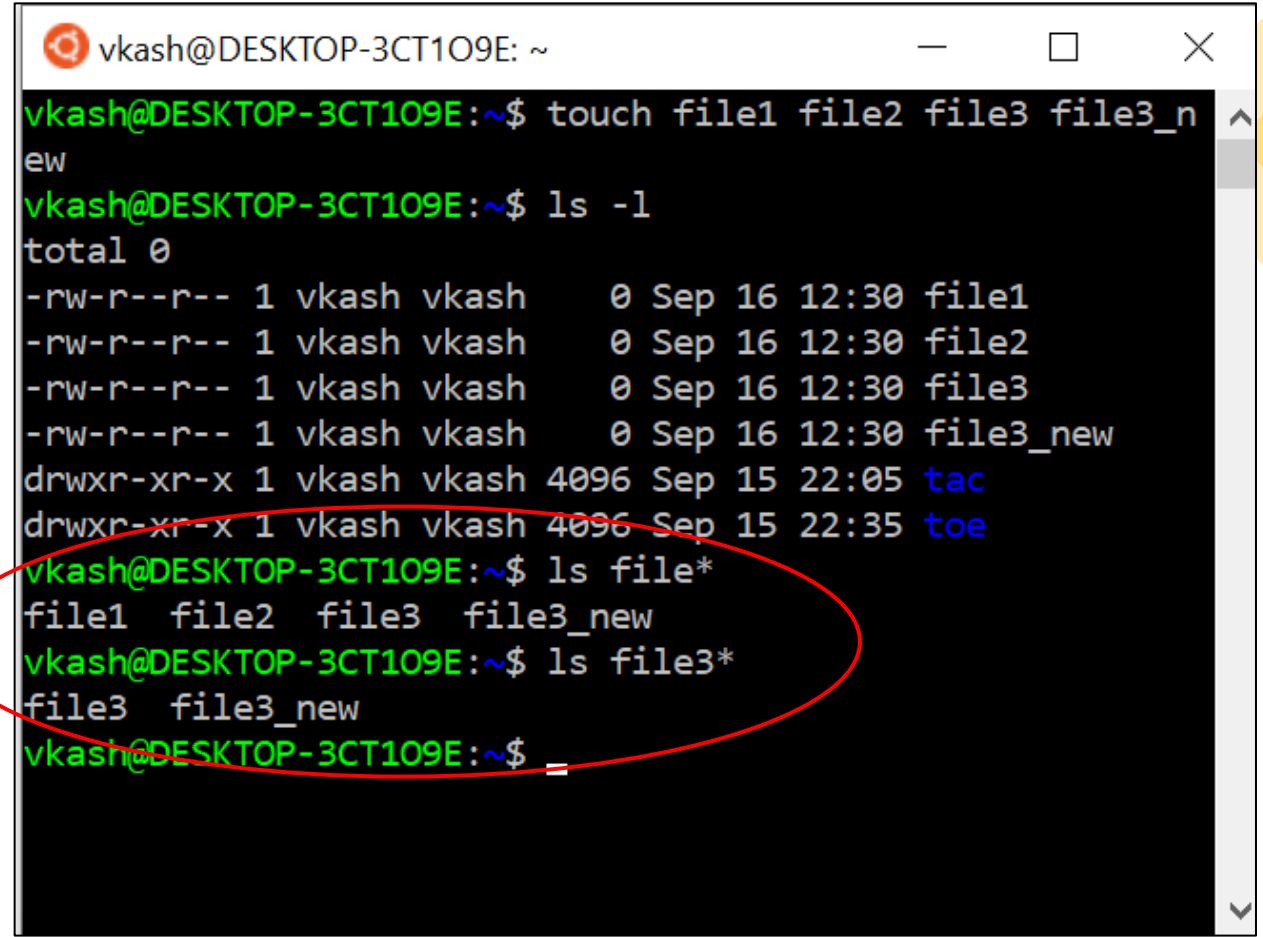
```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ ls  
tac tic toe  
vkash@DESKTOP-3CT109E:~$ rmdir tic  
rmdir: failed to remove 'tic': Directory not empty  
vkash@DESKTOP-3CT109E:~$ rmdir tic/ket  
vkash@DESKTOP-3CT109E:~$ rmdir tic  
vkash@DESKTOP-3CT109E:~$ ls -l tic  
ls: cannot access 'tic': No such file or directory  
vkash@DESKTOP-3CT109E:~$  
vkash@DESKTOP-3CT109E:~$ mkdir -p tic/ket  
vkash@DESKTOP-3CT109E:~$ rm -rf -i tic  
rm: descend into directory 'tic'? y  
rm: remove directory 'tic/ket'? y  
rm: remove directory 'tic'? y  
vkash@DESKTOP-3CT109E:~$ ls -l tic  
ls: cannot access 'tic': No such file or directory  
vkash@DESKTOP-3CT109E:~$
```

Linux File Globbing

- **Globbing** is also known as path name expansion. To learn about file globbing first we need to know about wildcards.
- **Wildcards pattern** are the **strings containing characters like '*', '?', '['**. It performs action on more than one file having same pattern or to find part of a phrase in a text file. Shell uses wildcards for file globbing.
- **Globbing** is an operation that recognizes the wildcard pattern and expands it into its path name.

Linux File Globbing

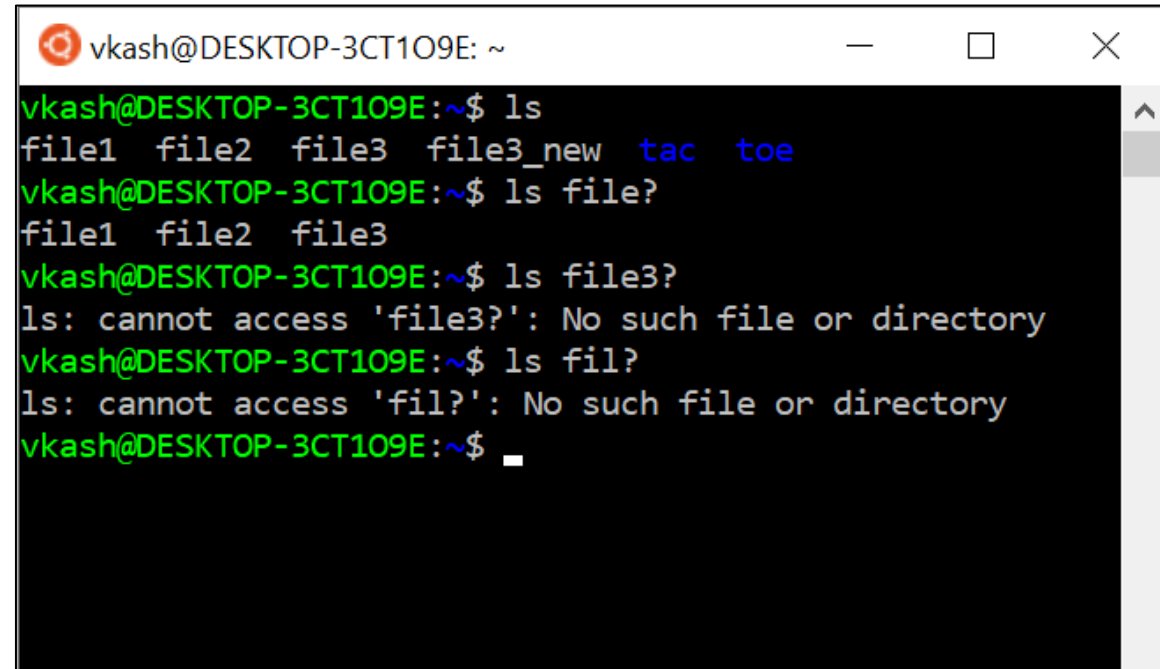
- * (Asterisk)
The asterisk is interpreted as the sign to generate matching file names. It is placed at the end of a line.
It matches the combination by any number of characters.

A terminal window titled 'vkash@DESKTOP-3CT109E: ~' with standard window controls. It shows a sequence of commands and their outputs. First, 'touch file1 file2 file3 file3_new' is executed. Then, 'ls -l' is run, showing a list of files including file1, file2, file3, file3_new, tac, and toe. Next, 'ls file*' is executed, and its output 'file1 file2 file3 file3_new' is circled in red. Finally, 'ls file3*' is run, showing 'file3 file3_new'.

```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ touch file1 file2 file3 file3_new  
vkash@DESKTOP-3CT109E:~$ ls -l  
total 0  
-rw-r--r-- 1 vkash vkash 0 Sep 16 12:30 file1  
-rw-r--r-- 1 vkash vkash 0 Sep 16 12:30 file2  
-rw-r--r-- 1 vkash vkash 0 Sep 16 12:30 file3  
-rw-r--r-- 1 vkash vkash 0 Sep 16 12:30 file3_new  
drwxr-xr-x 1 vkash vkash 4096 Sep 15 22:05 tac  
drwxr-xr-x 1 vkash vkash 4096 Sep 15 22:35 toe  
vkash@DESKTOP-3CT109E:~$ ls file*  
file1 file2 file3 file3_new  
vkash@DESKTOP-3CT109E:~$ ls file3*  
file3 file3_new  
vkash@DESKTOP-3CT109E:~$
```

Linux File Globbing

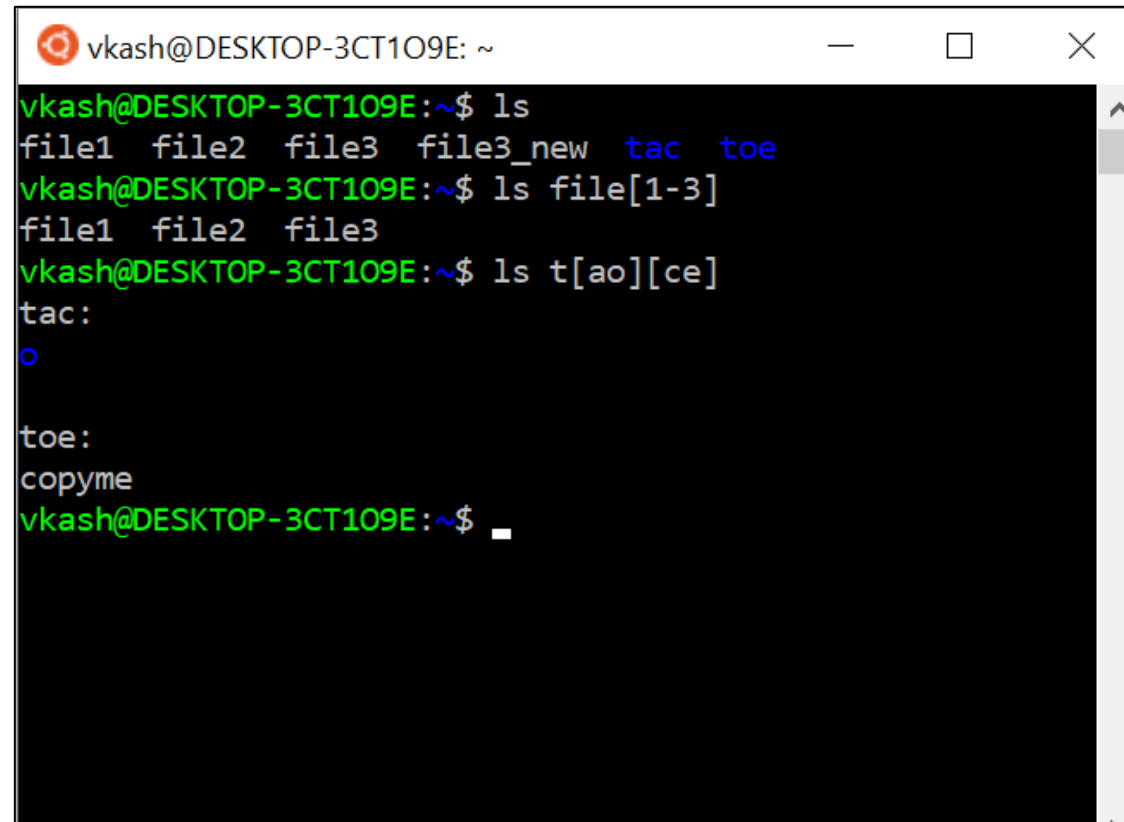
- ? (Question mark)
You can also use question mark sign in place of asterisk to generate matching file names. It is placed at the end of a line.
It matches the combination by exactly one character.

A terminal window titled 'vkash@DESKTOP-3CT109E: ~' with standard window controls. The terminal shows a series of commands and their outputs. The first command 'ls' lists files 'file1', 'file2', 'file3', 'file3_new', 'tac', and 'toe'. The second command 'ls file?' lists 'file1', 'file2', and 'file3'. The third command 'ls file3?' results in an error: 'ls: cannot access 'file3?': No such file or directory'. The fourth command 'ls fil?' also results in an error: 'ls: cannot access 'fil?': No such file or directory'. The prompt ends with a cursor on a new line.

```
vkash@DESKTOP-3CT109E: ~$ ls
file1 file2 file3 file3_new tac toe
vkash@DESKTOP-3CT109E: ~$ ls file?
file1 file2 file3
vkash@DESKTOP-3CT109E: ~$ ls file3?
ls: cannot access 'file3?': No such file or directory
vkash@DESKTOP-3CT109E: ~$ ls fil?
ls: cannot access 'fil?': No such file or directory
vkash@DESKTOP-3CT109E: ~$
```

Linux File Globbing

- **[] (Square Brackets)**
Square brackets are also used to generate matching file names inside the brackets and the first subsequent. Order inside the square bracket doesn't matter.
It matches the combination by exactly one character.



```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ ls  
file1 file2 file3 file3_new tac toe  
vkash@DESKTOP-3CT109E:~$ ls file[1-3]  
file1 file2 file3  
vkash@DESKTOP-3CT109E:~$ ls t[ao][ce]  
tac:  
o  
  
toe:  
copyme  
vkash@DESKTOP-3CT109E:~$
```

Linux File Globbing

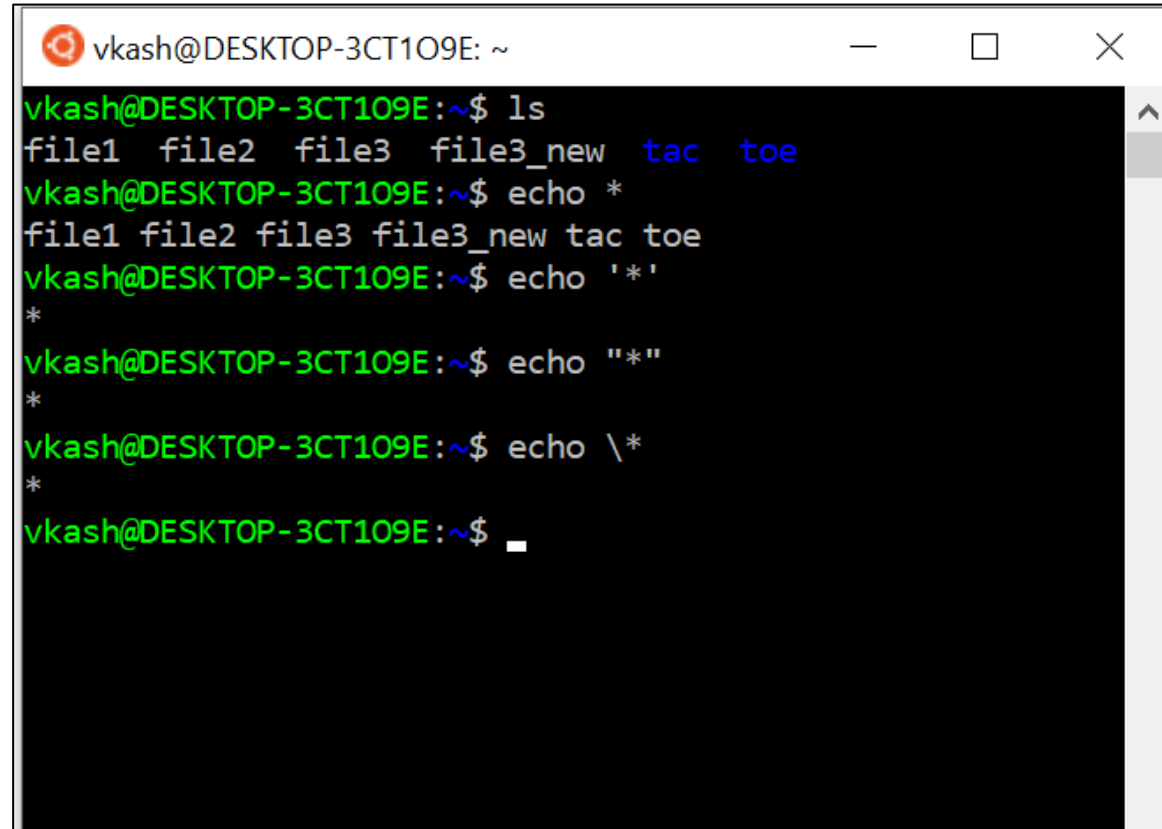
- **!** (exclamation mark)
Exclamation mark excludes characters from the list within the square bracket. And you can use the combination of asterisk (*), question mark (?) and square bracket [].

Try Globbing with different learned commands such as **echo**, **touch**, **mkdir**, **rm** etc.

```
vkash@DESKTOP-3CT109E: ~  
vkash@DESKTOP-3CT109E:~$ ls  
file1 file2 file3 file3_new tac toe  
vkash@DESKTOP-3CT109E:~$ ls file[!2]  
file1 file3  
vkash@DESKTOP-3CT109E:~$ ls file[!2]*  
file1 file3 file3_new  
vkash@DESKTOP-3CT109E:~$ ls file[!3]  
file1 file2  
vkash@DESKTOP-3CT109E:~$ ls file[!3]*  
file1 file2  
vkash@DESKTOP-3CT109E:~$ ls t[ao][!e]*  
o  
vkash@DESKTOP-3CT109E:~$ ls tac  
o  
vkash@DESKTOP-3CT109E:~$
```

Globbing Prevention

- The command **echo *** will print ***** when a directory is empty. But if not empty, then it will print all the files.
- To prevent it, special characters can be used like backward slash (\), single quote (') and double quote (").
- It works not only for *****, but for all globbing symbols.

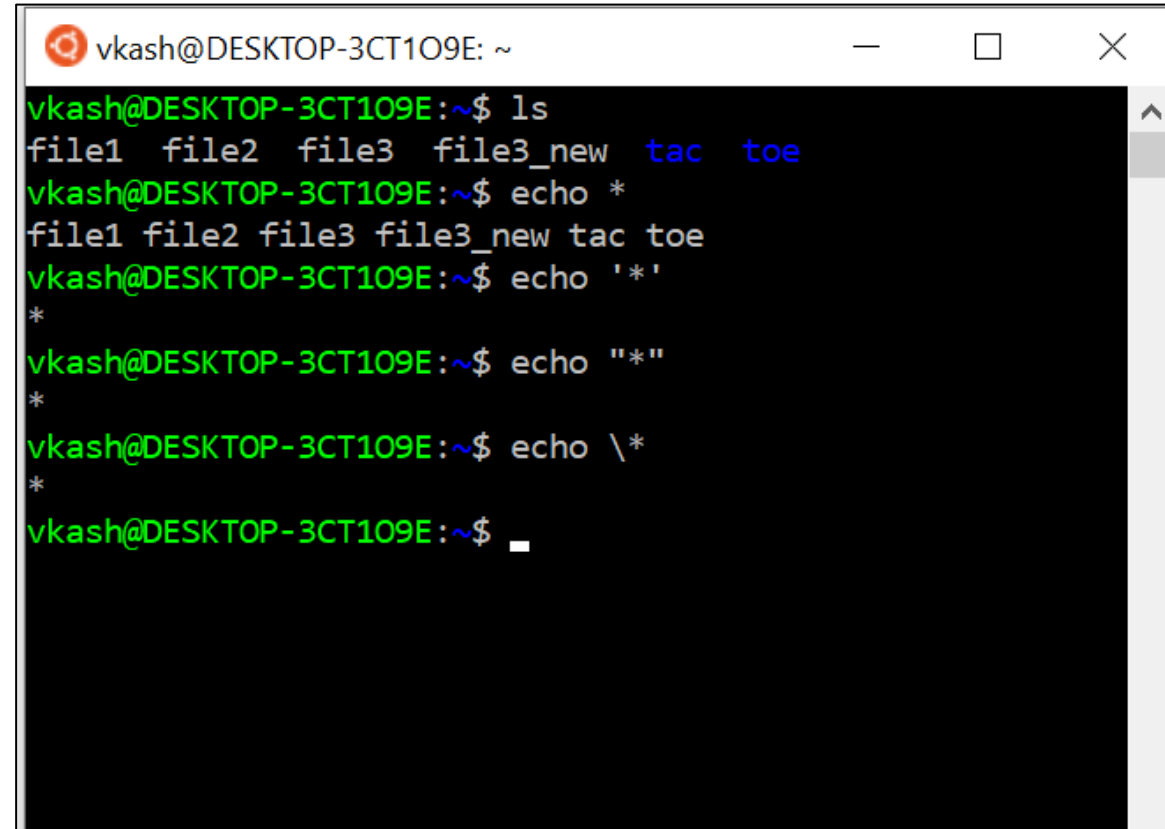
A terminal window titled 'vkash@DESKTOP-3CT109E: ~' with standard window controls. The terminal shows a series of commands and their outputs. First, 'ls' lists files: file1, file2, file3, file3_new, tac, toe. Then, 'echo *' outputs the same list. Next, 'echo \'' outputs a single asterisk. Then, 'echo "' outputs a double asterisk. Finally, 'echo \' outputs a single asterisk. The prompt 'vkash@DESKTOP-3CT109E:~\$' is followed by a cursor.

```
vkash@DESKTOP-3CT109E:~$ ls
file1 file2 file3 file3_new tac toe
vkash@DESKTOP-3CT109E:~$ echo *
file1 file2 file3 file3_new tac toe
vkash@DESKTOP-3CT109E:~$ echo '*'
*
vkash@DESKTOP-3CT109E:~$ echo "*"
*
vkash@DESKTOP-3CT109E:~$ echo \'
*
vkash@DESKTOP-3CT109E:~$
```

Globbing Prevention

- The command **echo *** will print ***** when a directory is empty. But if not empty, then it will print all the files.
- To prevent it, special characters can be used like backward slash (\), single quote (') and double quote (").
- It works not only for *, but for all globbing symbols.

Read manuals for all commands used today and try new configurations to better understand it

A terminal window titled 'vkash@DESKTOP-3CT109E: ~' with standard window controls. It shows a series of commands and their outputs to demonstrate globbing prevention. The commands are: 'ls' (listing files), 'echo *' (printing all files), 'echo \'' (printing a single quote), 'echo \"' (printing a double quote), and 'echo \'\"' (printing a backslash).

```
vkash@DESKTOP-3CT109E: ~$ ls
file1 file2 file3 file3_new tac toe
vkash@DESKTOP-3CT109E: ~$ echo *
file1 file2 file3 file3_new tac toe
vkash@DESKTOP-3CT109E: ~$ echo '*'
*
vkash@DESKTOP-3CT109E: ~$ echo "\""
*
vkash@DESKTOP-3CT109E: ~$ echo \'
*
vkash@DESKTOP-3CT109E: ~$
```