



# Finding and Managing Files

**Session 2**

# Table of contents

**01**

Finding files

**02**

Links

**03**

Compression and  
Archiving

**04**

Pipelining and  
Redirection



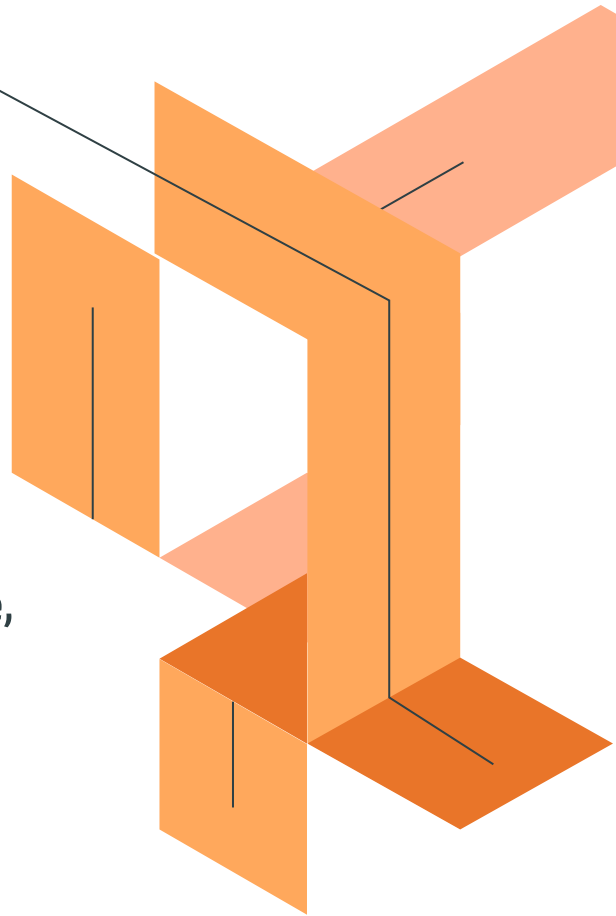
01

# Finding Files

# Finding Files

One of the ways that we can look for files in Linux is by using **find**.

This command allows us to search for files on our system based on different attributes, such as name, size, permissions, type, etc.



# Find Command

The general syntax for the find command is as follows:

```
find [path...] [expression]
```

- The **path...** attribute defines the starting directory or directories where find will search the files.
- The **expression** attribute is made up of options, search patterns, and actions separated by operators.



# Find Command - Options

Option	Description
<b>-name pattern</b>	Searches for files with a specific name or pattern.
<b>-iname pattern</b>	Case-insensitive version of -name. Searches for files with a specific name or pattern, regardless of case.
<b>-type type</b>	Specifies the type of file to search for (e.g., f for regular files, d for directories).
<b>-maxdepth</b>	limits how deep find will go into the directory tree. -> must be the first option

# Find Command - Size

- Searches for files based on size: `+n` finds larger files, `-n` finds smaller files. `n` measures size in characters.

You can use the following suffixes to specify the file size:

- **b** : 512-byte blocks (default)
- **c** : bytes
- **k** : Kilobytes
- **M** : Megabytes
- **G** : Gigabytes

## Example

- Case-Sensitive Search for a Specific File Name

```
fady@Ubuntu-labtop:~/Session2/MyFiles$ find . -name "main.txt"
./main.txt
```

- Case-Insensitive Search for a Specific File Name

```
fady@Ubuntu-labtop:~/Session2/MyFiles$ find . -iname "second.txt"
./SEcond.txt
./SECOND.TXT
./second.txt
```



## Example

### Find Text Files in the Current Directory Only

- Explanation: Finds all files with `.txt` extension (case-sensitive) only in the current directory (no subdirectories searched).

```
fady@Ubuntu-laptop:~/Session2/MyFiles$ find . -maxdepth 1 -name "*.txt"
./SEcond.txt
./main.txt
./second.txt
```

## Example

- Find All Directories in the Current Path

```
fady@Ubuntu-laptop:~/Session2/MyFiles$ find . -type d
.
./Dir1
```

- Find Files Larger Than 18 Bytes

```
fady@Ubuntu-laptop:~/Session2/MyFiles$ find . -size +18c
.
./SEcond.txt
./Dir1
```

# Find - Cheat Sheet

Option	Description	Notes
<code>-name</code>	Search by exact name (case-sensitive) or pattern	Matches exactly <code>file.txt</code> or any pattern <code>"*.txt"</code>
<code>-iname</code>	Search by name (case-insensitive)	Matches <code>file.txt</code> , <code>File.TXT</code> etc.
<code>-type</code>	Search by file type	<code>f</code> = file, <code>d</code> = directory,
<code>-size</code>	Search by file size	<code>+</code> greater, <code>-</code> smaller, no sign = exact size
		Units: <code>b</code> (512-bytes-block), <code>c</code> (bytes), <code>k</code> (KB), <code>M</code> (MB), <code>G</code> (GB)
<code>-maxdepth</code>	Limit search depth in directory hierarchy	1 = current dir only, 2 = include subdirs, etc.



# Hands on #1

---

# Find\_Practice

- Create **hands-on** directory in your home and go to it
- Create three directories and name it **OSC myDir1 myDir2**
- Create file with name **OSC.TXT**
- Create another file with name **osc.TXT**
- Now, use file command to search on:
  1. Any **file** start with name **osc** -> ignore Capital and small letters
  2. Any **directory** start with name **my**

# Find\_Practice Solution

- `mkdir ~/hands-on`
- `cd ~/hands-on`
- `mkdir OSC myDir1 myDir2`
- `touch OSC.TXT`
- `touch osc.TXT`
- `find . -type f -iname "osc*"`
- `find . -type d -name "my*"`



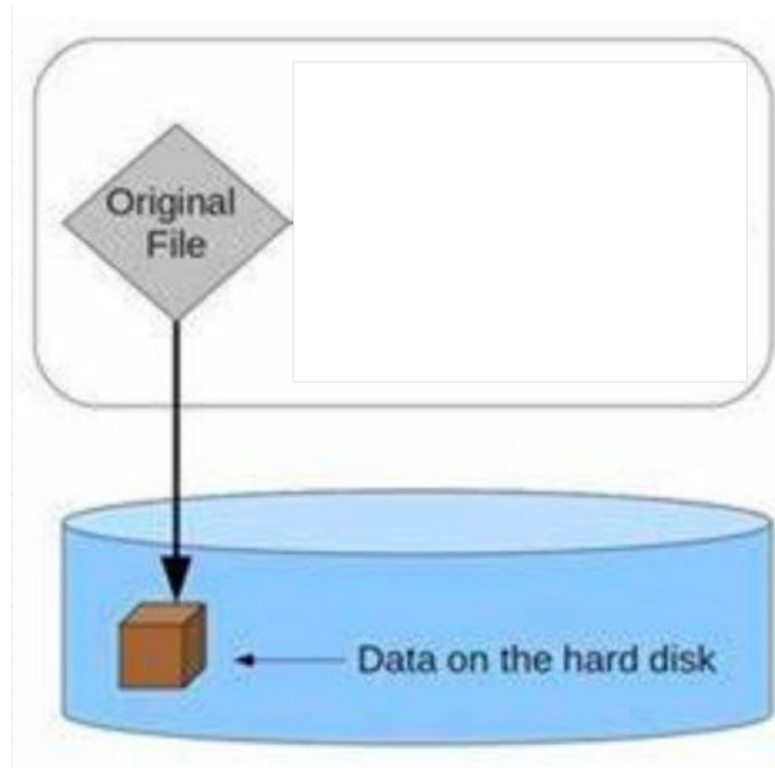


02

---

# Links

Any file in linux -> Point on his data -> Contain his address on disk



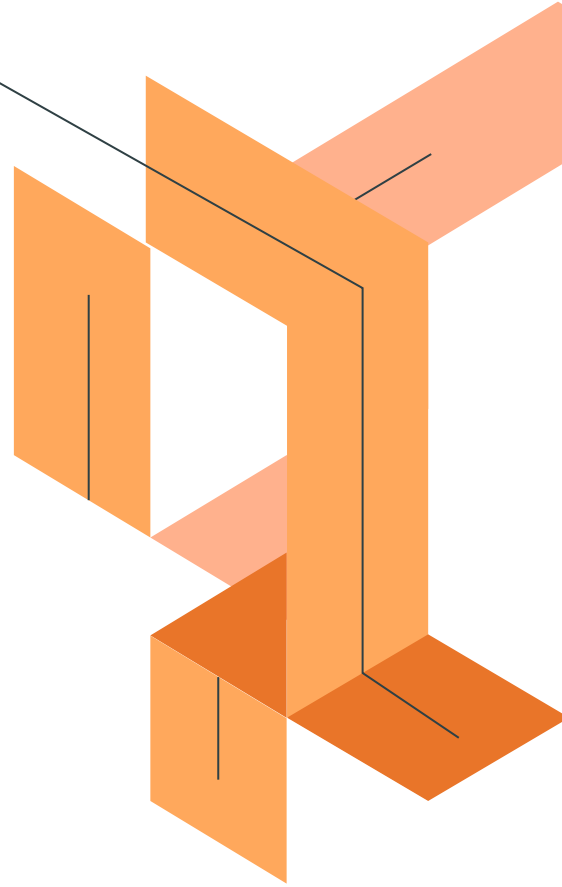


# What are Links?

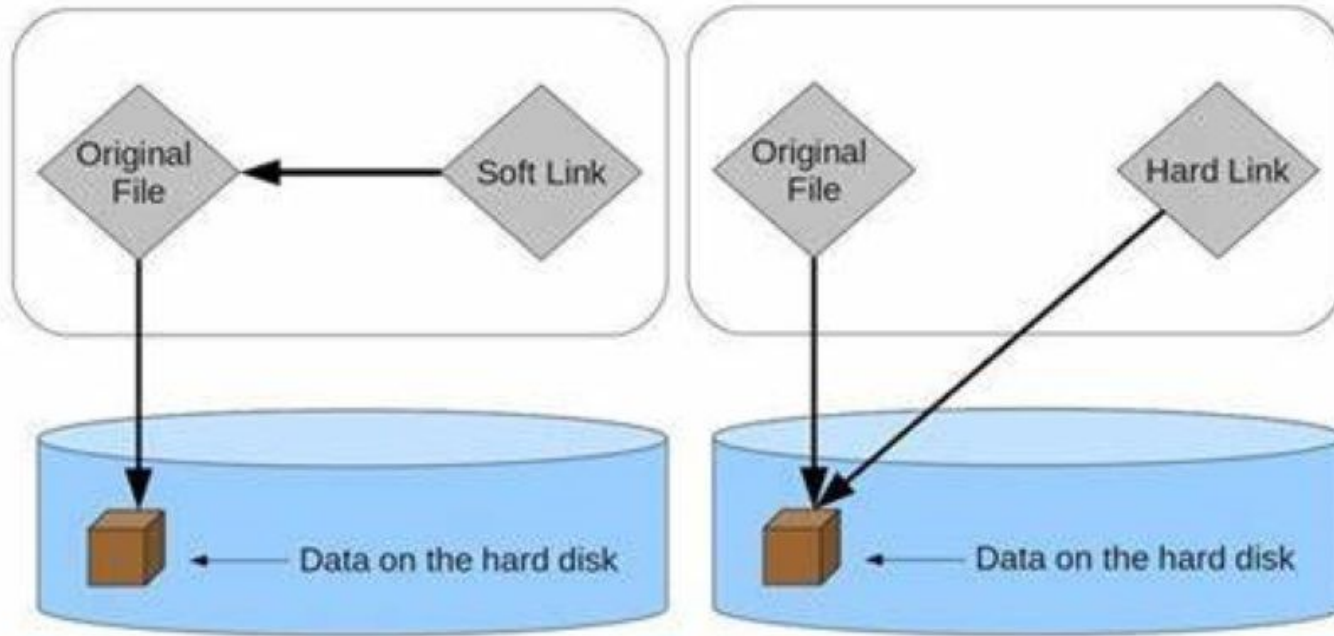
- A **link** in Linux is a file that points to another file/directory.
- Creating links is similar to creating shortcuts.
- A file can have **multiple links** linked to it. But a link can only be linked to (pointed to) one file.

There are two types of links:

1. Soft (Symbolic) link.
2. Hard link.

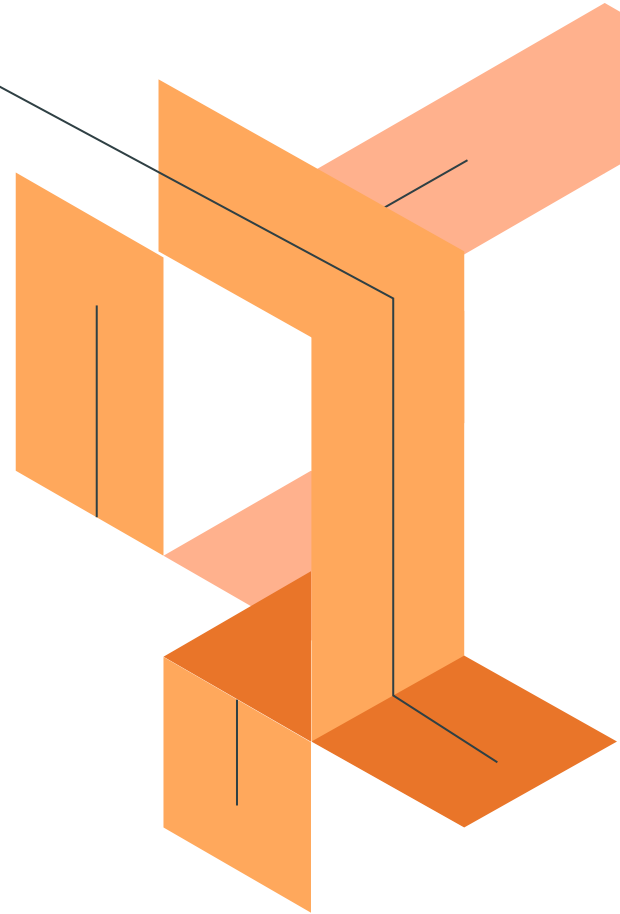


# Soft Link VS Hard Link



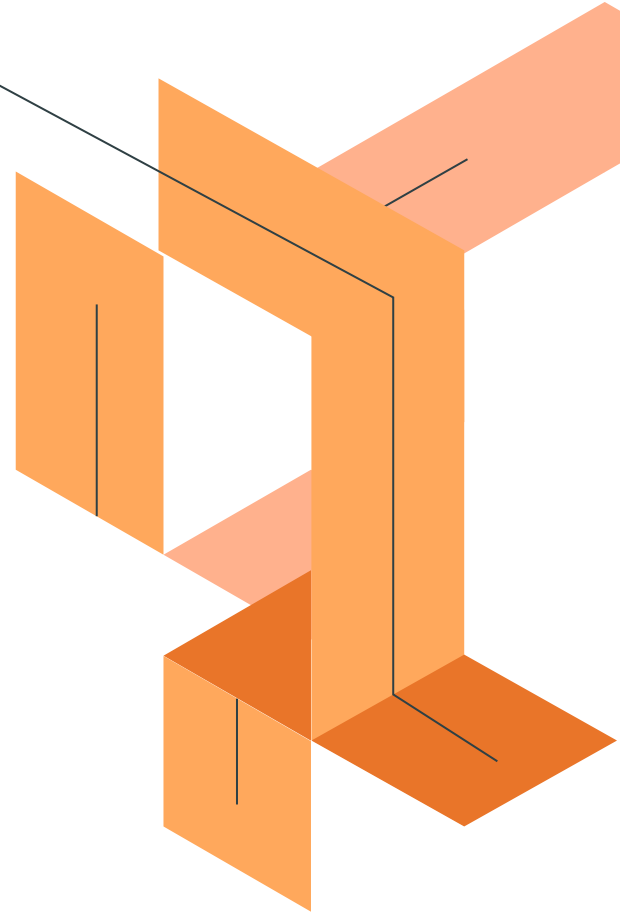
# Soft Link (Symbolic)

- They can point to a directory or special file, not just a regular file.
- If the original file is deleted it stop working
- Created by: **ln -s [file path] [link path]**
- Remove using: **rm [link path]** or **unlink [link path]**



# Hard Link

- Can only be used on files
- If the original file is deleted it still works
- Created by: **ln filename linkname**
- Remove using: **rm link** or **unlink link**
- You can verify if a file has hard links with the **ls -li** command.



# Example

- Create Soft Link

```
bebo@bebo-G3-3590:~/Links$ ls
file.txt
bebo@bebo-G3-3590:~/Links$ ln -s file.txt softlink.txt
bebo@bebo-G3-3590:~/Links$ ls -l
total 0
-rw-rw-r-- 1 bebo bebo 0 Aug  6 02:00 file.txt
lrwxrwxrwx 1 bebo bebo 8 Aug  6 02:05 softlink.txt -> file.txt
```

# Example

- Create Hard Link

```
bebo@bebo-G3-3590:~/Links$ ls
file.txt  softlink.txt
bebo@bebo-G3-3590:~/Links$ ln file.txt hardlink.txt
bebo@bebo-G3-3590:~/Links$ ls -l
total 0
-rw-rw-r-- 2 bebo bebo 0 Aug  6 02:00 file.txt
-rw-rw-r-- 2 bebo bebo 0 Aug  6 02:00 hardlink.txt
lrwxrwxrwx 1 bebo bebo 8 Aug  6 02:05 softlink.txt -> file.txt
```

## Deleting a Soft Link (Symbolic Link):

- **Effect:** Only the symbolic link itself is deleted.
- **Original File:** The original file remains unaffected and intact.
- **Other Links:** Any other links (hard or soft) to the original file remain unaffected.

## Deleting a Hard Link:

- **Effect:** The specific hard link is removed.
- **Original File:** The original file is not deleted as long as there is at least one remaining hard link to it.
- **Other Links:** Any other hard links to the file continue to function normally.
- **The file's content** remains accessible through these links





# Hands on #1-2

---

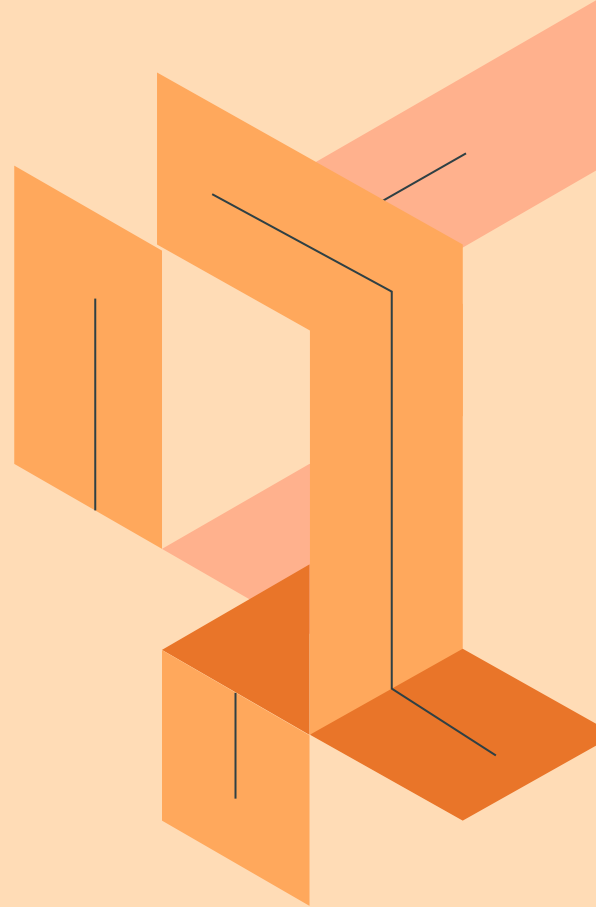
# Links Practice

- Use nano to create `file1.txt` & `file2.txt` with any content
- Create a hard link to `file1.txt` named `hardlink`.
- Create a soft link (symbolic) to `file2.txt` named `softlink`.
- Verify links
- Delete the original `file2.txt` and try to open `softlink`.

# Links Practice Solution

- Create a hard link to `file1.txt` named `hardlink`.
  - In `file1.txt` `hardlink`
- Create a soft (symbolic) link to `file2.txt` named `softlink`.
  - In `-s file2.txt softlink`
- Verify links
  - `ls -li`
- Delete the original `file2.txt` and try to open `softlink`
  - `rm file2.txt`
  - `cat softlink_to_file2` # will show "No such file or directory"

**Break..** 😊





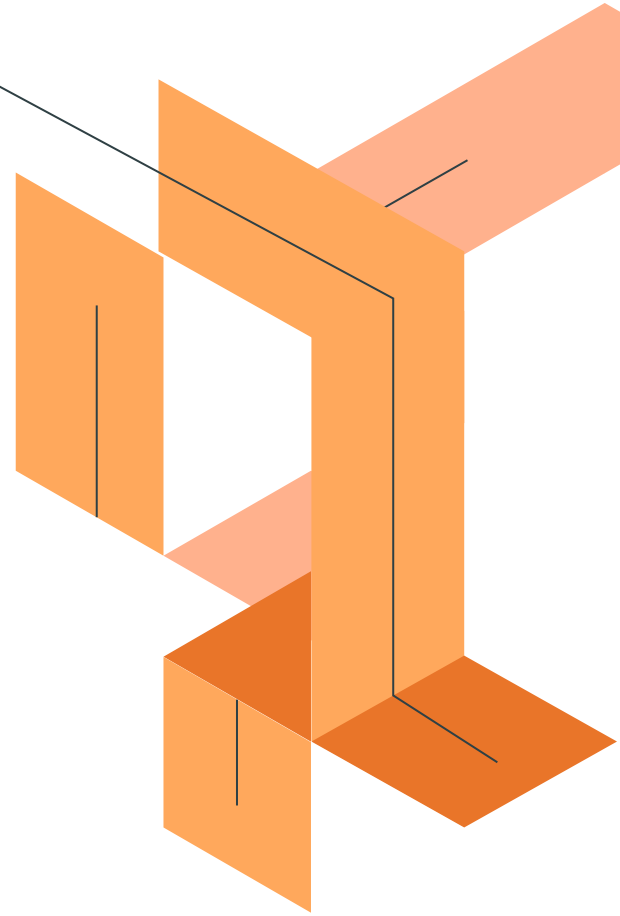
03

---

# **Compression & Archiving**

# Compression

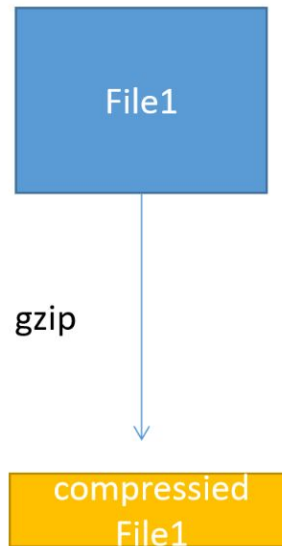
Compression, in computer science language, is a process that aims to represent the data contained in a file with less data.



# Compression

**Compression**, in computer science language, is a process that aims to represent the data contained in a file with less data.

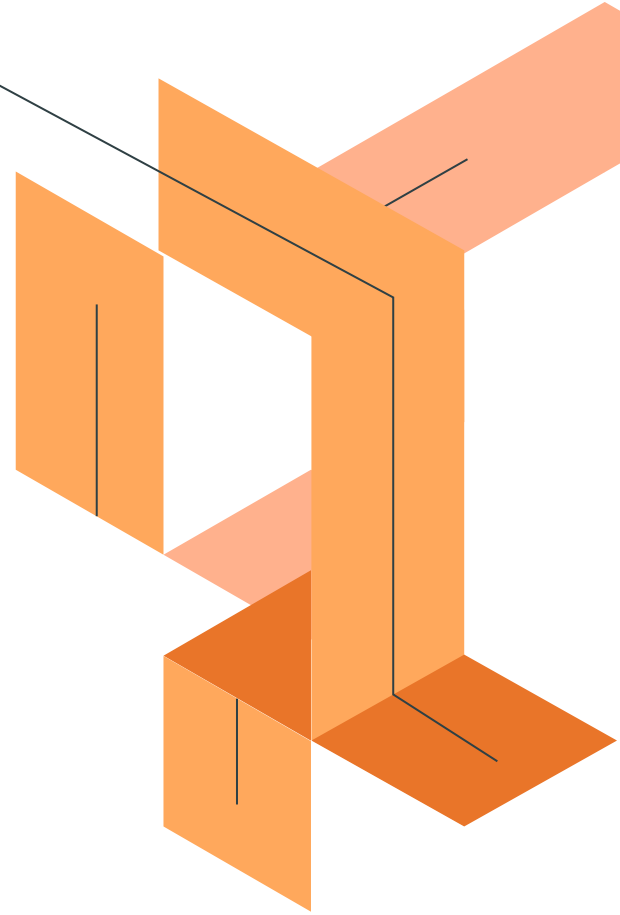
Compression



smaller file size

# Archiving

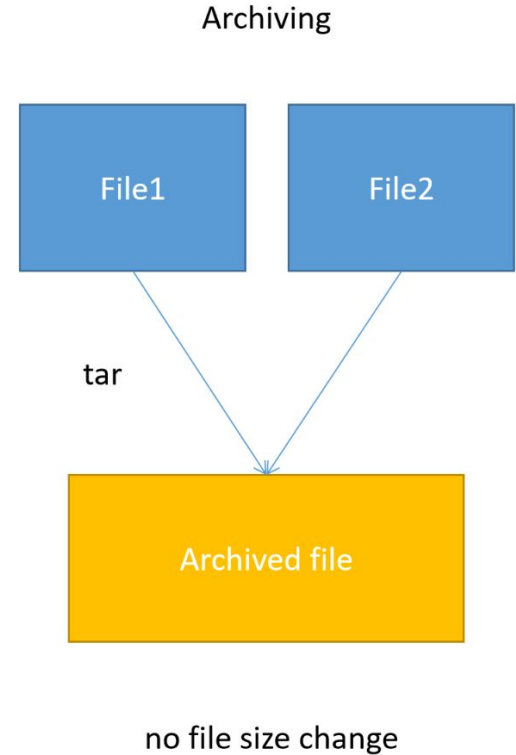
An **archive** file is a collection of files and directories that are stored in one file. The archive file is not compressed — it uses the same amount of disk space as all the individual files and directories combined.





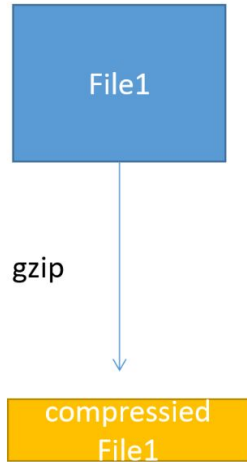
# Archiving

An **archive** file is a collection of files and directories that are stored in one file. The archive file is not compressed — it uses the same amount of disk space as all the individual files and directories combined.



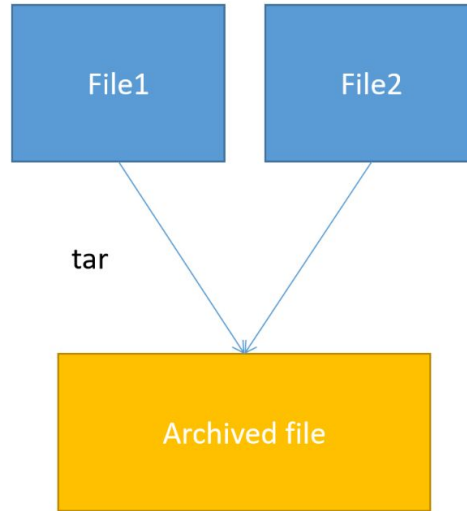
# Compression VS Archiving

## Compression



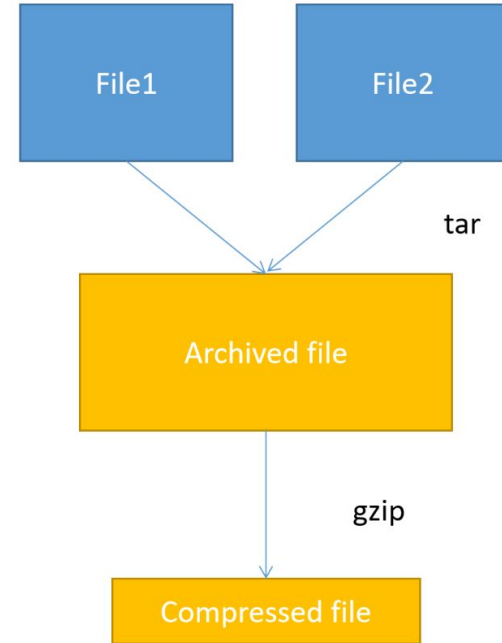
smaller file size

## Archiving



no file size change

## Archiving&Compression




# Tools

zip

gzip

bzip2

tar

An orange decorative shape, resembling a stylized arrow or a corner piece, is located on the left side of the slide.

<b>Tool</b>	<b>Compression Algorithm</b>	<b>Utility</b>	<b>Year</b>
zip	Deflate	Compressor and Archiver	1989
gzip	Deflate	Compressor	1992
bzip2	BWT + Huffman coding	Compressor	1998

# zip

This tool does both compression and archiving.

Tool	Extension	Decompression Tool	Example
zip	.zip	unzip	<b>Compress</b> : <code>zip file.zip fileName</code> <b>Decompress</b> <code>unzip fileName.zip</code>

- **To compress multiple files :**

`zip files.zip file1 file2 file3` → **Output** : files.zip

- **To compress multiple files and directories :**

`zip -r filename.zip file1 file2 file3 /usr/work/school`

- The above command compresses file1, file2, file3, and the contents of /usr/work/school and places them in an archive named **filename.zip**.

# zip

This tool does both compression and archiving.

Tool	Extension	Decompression Tool
zip	.zip	unzip

## Syntax:

```
zip [options] zipname.zip files
```

## Example:

- **Compress:** `zip file.zip fileName`

```
• Compression
>ls -lh
total 1.0M
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:50 file

• Compression
>zip file.zip file
  adding: file (deflated 100%)

• Compression
>ls -lh
total 1.1M
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:50 file
-rw-rw-r-- 1 hadeer hadeer 1.2K Aug  5 22:52 file.zip
```

## Example:

- Decompress `unzip fileName.zip`

• **Compression**

```
>ls  
file.zip
```

• **Compression**

```
>unzip file.zip  
Archive:  file.zip  
  inflating: file
```

• **Compression**

```
>ls  
file  file.zip
```



## Example:

- To compress multiple files :

`zip files.zip file1 file2 file3` → **Output:** files.zip

### • Compression

```
>zip files.zip file1 file2 file3
  adding: file1 (deflated 100%)
  adding: file2 (deflated 100%)
  adding: file3 (deflated 100%)
```

### • Compression

```
>ls -lh
total 3.1M
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:56 file1
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:56 file2
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:56 file3
-rw-rw-r-- 1 hadeer hadeer 3.5K Aug  5 22:57 files.zip
```

## Example:

- **To compress Directory:**  
`zip -r Dir.zip Dir/`

```
>ls
Dir

• Compression
>zip -r Dir.zip Dir/
  adding: Dir/ (stored 0%)
  adding: Dir/file3 (deflated 100%)
  adding: Dir/file1 (deflated 100%)
  adding: Dir/file2 (deflated 100%)

• Compression
>ls
Dir  Dir.zip
```

## Example:

- **To compress multiple files and directories :**  
`zip -r Data.zip file1 file2 file3 Dir/`

```
>ls
Dir  file2  file3

| Compression
>zip -r Data.zip file2 file3 Dir/
  adding: file2 (deflated 100%)
  adding: file3 (deflated 100%)
  adding: Dir/ (stored 0%)
  adding: Dir/file1 (deflated 100%)

| Compression
>ls
Data.zip  Dir  file2  file3
```

# gzip and bzip2

These are compression-only tools.

Compression Tool	Extension	Decompre-ssion Tool	Syntax
gzip	.gz	gunzip	<b>Compress:</b> gzip fileName <b>Decompress:</b> gunzip fileName.gz
bzip2	.bz2	bunzip2	<b>Compress:</b> bzip2 fileName <b>Decompress:</b> bunzip2 fileName.bz2

Option	Description
-1 .. -9	Set amount of compression. Default is 6. <ul style="list-style-type: none"><li>○ -1, --fast: <b>Compress</b> faster.</li><li>○ -9, --best: <b>Compress</b> better.</li></ul>
-f	<b>Force</b> compression even if a compressed version of the original file already exists.
-k	<b>Keep</b> a copy of the original file.
-v	<b>Verbose.</b>

## gzip Example:

- Compress & Decompress using gzip

```
>ls -lh
total 1.0M
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:56 file2

• Compression
>gzip file2

• Compression
>ls -lh
total 4.0K
-rw-rw-r-- 1 hadeer hadeer 1.1K Aug  5 22:56 file2.gz

• Compression
>gunzip file2.gz

• Compression
>ls -lh
total 1.0M
-rw-rw-r-- 1 hadeer hadeer 1.0M Aug  5 22:56 file2
```

## gzip Example:

- Compress -1 vs -9

• Compression

```
>gzip -1 file1
```

• Compression

```
>gzip -9 file2
```

• Compression

```
>ls -lh
```

```
total 12K
```

```
-rw-rw-r-- 1 hadeer hadeer 4.6K Aug 5 23:39 file1.gz
```

```
-rw-rw-r-- 1 hadeer hadeer 1.1K Aug 5 22:56 file2.gz
```

## gzip Example:

- Keep files and force compression

```
>gzip -k file1
```

• Compression

```
>ls
```

```
file1  file1.gz
```

• Compression

```
>echo "hi" >> file1
```

• Compression

```
>gzip -k file1
```

```
gzip: file1.gz already exists; do you wish to overwrite (y or n)? y
```

• Compression

```
>gzip -kf file1
```



## bzip2 Example:

- **Compress & Decompress**

```
>ls -lh
total 1.1M
-rw-rw-r-- 1 hadeer hadeer 1.1M Aug  6 00:40 file

• Compression
|
>bzip2 file

• Compression
|
>ls -lh
total 4.0K
-rw-rw-r-- 1 hadeer hadeer 54 Aug  6 00:40 file.bz2

• Compression
|
>bunzip2 file.bz2

• Compression
|
>ls
file
```

## bzip2 Example:

- Compress -1 vs -9

```
>ls -lh
total 10M
-rw-rw-r-- 1 hadeer hadeer 5.0M Aug  6 00:49 file1
-rw-rw-r-- 1 hadeer hadeer 5.0M Aug  6 00:49 file2

• Compression
>bzip2 -1 file1

• Compression
>bzip2 -9 file2

• Compression
>ls
file1.bz2  file2.bz2

• Compression
>ls -lh
total 8.0K
-rw-rw-r-- 1 hadeer hadeer 78 Aug  6 00:49 file1.bz2
-rw-rw-r-- 1 hadeer hadeer 48 Aug  6 00:49 file2.bz2
```

## bzip2 Example:

- Keep files and force compression

```
>bzip2 -k file
```

```
• Compression
```

```
>ls
```

```
file  file.bz2
```

```
• Compression
```

```
>bzip2 -k file
```

```
bzip2: Output file file.bz2 already exists.
```

```
• Compression
```

```
>bzip2 -kf file
```

```
• Compression
```

```
>bzip2 -kfv file
```

```
file: 109226.667:1, 0.000 bits/byte, 100.00% saved, 5242880 in, 48 out.
```

```
• Compression
```

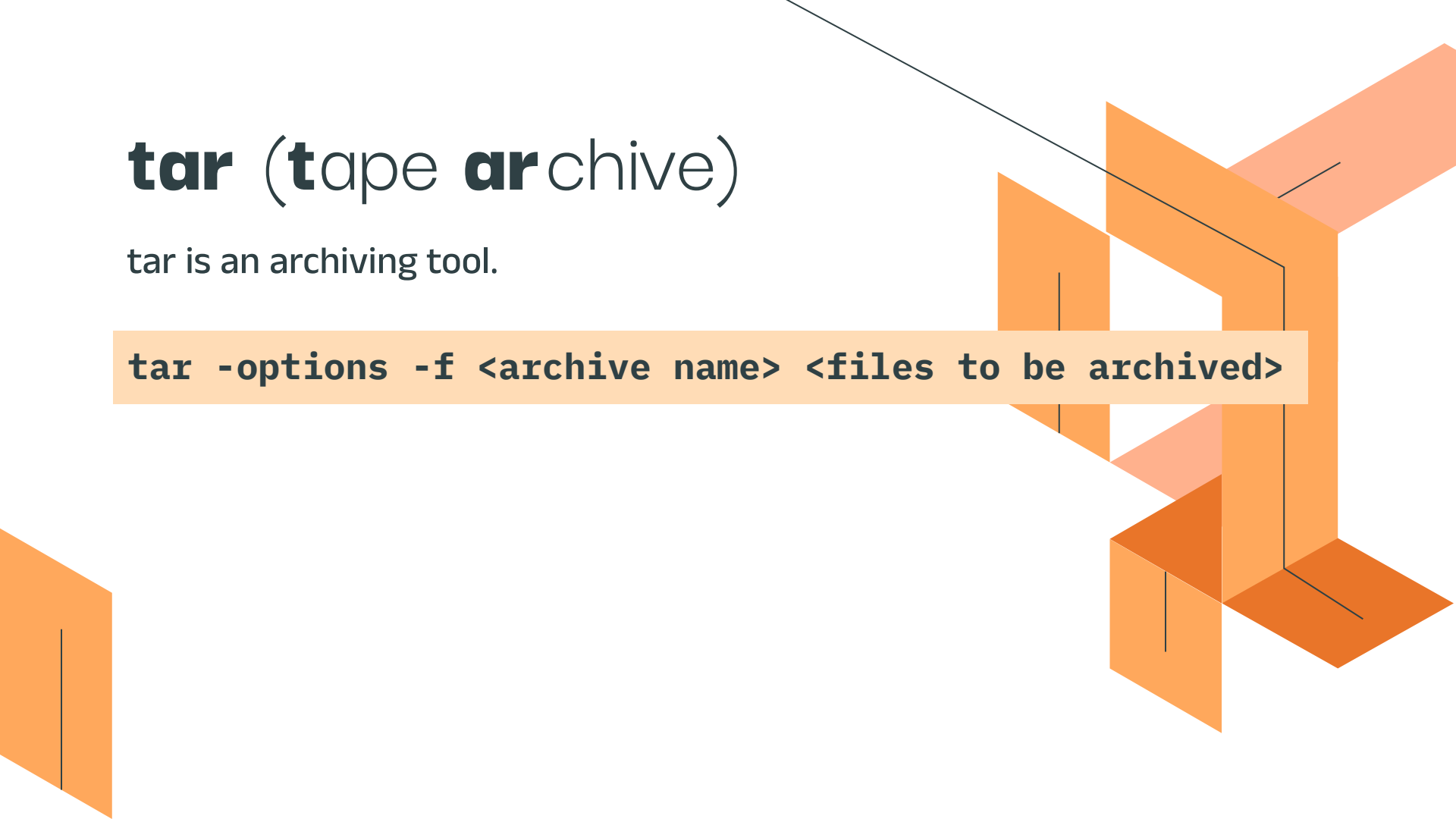
```
>ls
```

```
file  file.bz2
```

# **tar** (tape archive)

tar is an archiving tool.

```
tar -options -f <archive name> <files to be archived>
```



```
tar -options -f <archive name> <files to be archived>
```

What do you want to do?

- -c (create)
- -x (extract)
- -r (append)
- -t (list content)

Compress?

- -z (gzip)
- -j (bzip2)

Optional

- -v (verbose) (مطوّل)

Note: -f must be followed by the archive file name.

## Making a tarball (a tar archive)

```
myusername@ubuntu:~/osc/summer25/redpip$ tar -cvf mytar filestxt filespdf
filestxt
filespdf
myusername@ubuntu:~/osc/summer25/redpip$ tar -tvf mytar
-rw-rw-r-- myusername/myusername 24 2025-08-04 21:15 filestxt
-rw-rw-r-- myusername/myusername 48 2025-08-04 21:14 filespdf
```

## Appending to it

```
myusername@ubuntu:~/osc/summer25/redpip$ tar -rvf mytar filespng
filespng
myusername@ubuntu:~/osc/summer25/redpip$ tar -tvf mytar
-rw-rw-r-- myusername/myusername 24 2025-08-04 21:15 filestxt
-rw-rw-r-- myusername/myusername 48 2025-08-04 21:14 filespdf
-rw-rw-r-- myusername/myusername 36 2025-08-04 21:15 filespng
```

## Making a gzipped tar archive

```
myusername@ubuntu:~/osc/summer25/redpip$ tar -cvzf gzippedtar filestxt filespdf
filestxt
filespdf
myusername@ubuntu:~/osc/summer25/redpip$ file gzippedtar
gzippedtar: gzip compressed data, from Unix, original size modulo 2^32 10240
```

## Making a bziped tar archive

```
myusername@ubuntu:~/osc/summer25/redpip$ tar -cvjf bzipedtar filestxt filespdf
filestxt
filespdf
myusername@ubuntu:~/osc/summer25/redpip$ file bzipedtar
bzipedtar: bzip2 compressed data, block size = 900k
```



# Hands on #3

---



1. Create a file named file1.
2. Compress file1 using zip.
3. Compress file1 using gzip.
4. Compress file1 using bzip2.
5. Using nano text editor, edit file1.
6. Compress file1 using bzip2.
7. Using tar, archive the 3 compressed versions of file1.
8. Finally, decompress all the compressed files.



# Solution

---

- Create a file named file1.  
`touch file1`
- Compress file1 using zip.  
`zip file1.zip file1`
- Compress file1 using gzip.  
`gzip -k file1`
- Compress file1 using bzip2.  
`bzip2 -k file1`
- Using nano text editor, edit file1.  
`nano file1`

- Compress file1 using bzip2.

```
bzip2 -kf file1
```

- Using tar, archive the 3 compressed versions of file1.

```
tar -cf mytar.tar file1.zip file1.gz file1.bz2
```

- Finally, decompress all the compressed files.

```
unzip file1.zip
```

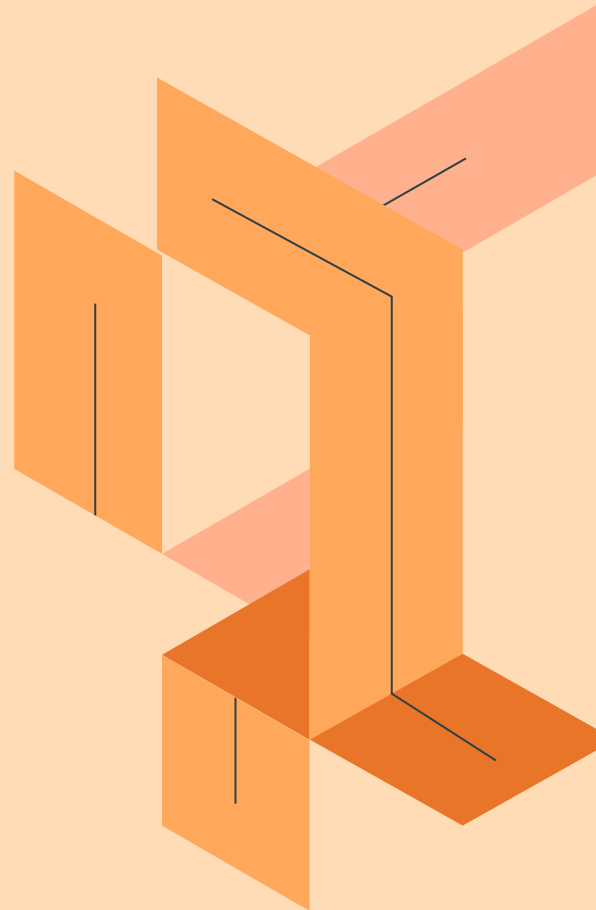
```
gunzip file1.gz
```

```
bunzip2 file1.bz2
```

04

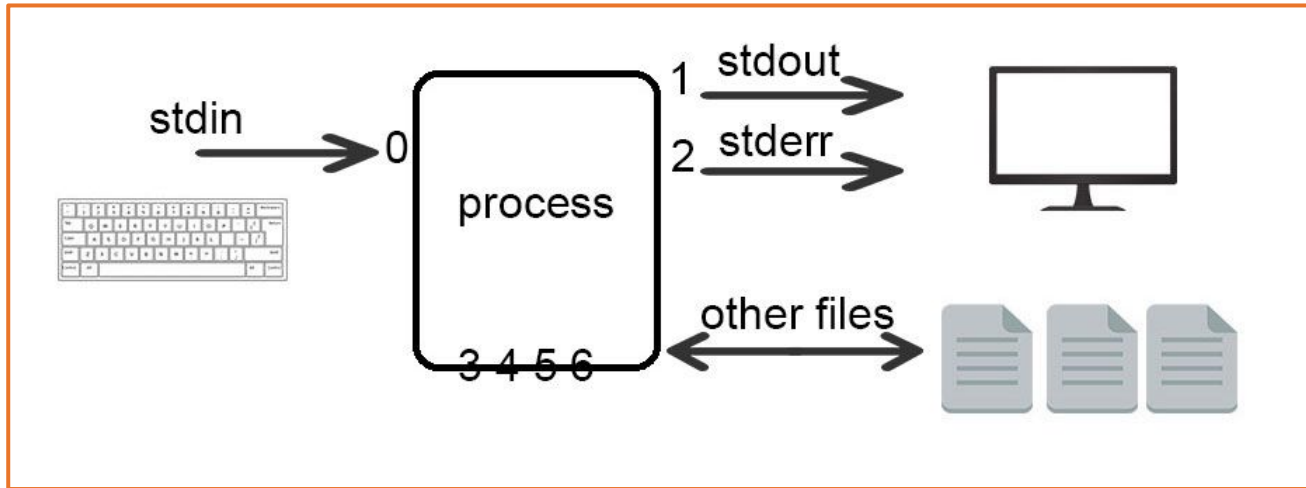
---

# Pipelining & Redirection



# Standard Channels

Processes use numbered **channels** to manage open files. Processes are created with default connections for channels 0, 1, and 2, known as standard input, standard output, and standard error. They use channels 3 and above to connect to other files. They use



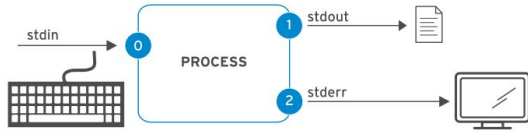
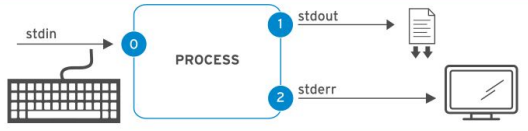
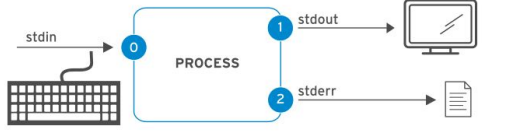
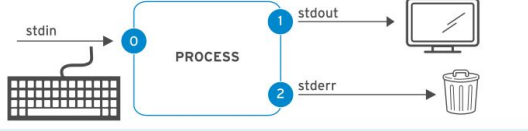
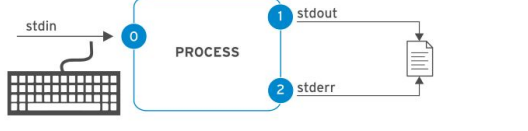
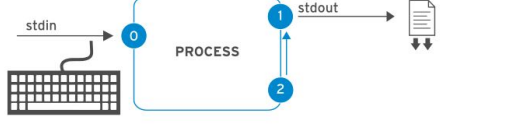
# Standard Channels

No.	Channel Name	Description	Default Connection	Usage
0	stdin	Standard input	Keyboard	read only
1	stdout	Standard output	Terminal	write only
2	stderr	Standard error	Terminal	write only
3+	filename	Other files	none	read/write

# Redirection

We can redirect output channels (stdout and stderr) to files instead of the terminal.

- **>** : Overwrite
- **>>** : Append

USAGE	EXPLANATION	VISUAL AID
<code>&gt; file</code>	redirect <b>stdout</b> to overwrite a file	
<code>&gt;&gt; file</code>	redirect <b>stdout</b> to append to a file	
<code>2&gt; file</code>	redirect <b>stderr</b> to overwrite a file	
<code>2&gt; /dev/null</code>	discard <b>stderr</b> error messages by redirecting to <b>/dev/null</b>	
<code>&gt; file 2&gt;&amp;1</code>	redirect <b>stdout</b> and <b>stderr</b> to overwrite the same file	
<code>&amp;&gt; file</code>		
<code>&gt;&gt; file 2&gt;&amp;1</code>	redirect <b>stdout</b> and <b>stderr</b> to append to the same file	
<code>&amp;&gt;&gt; file</code>		



# Redirection

**Redirection**, in simple words, is just the process of editing a file descriptor table entry.

We can redirect output channels (stdout and stderr) to files instead of the terminal.

i.e.:

- Redirecting stderr: `command 2> file.txt`
- Redirecting stdout: `command > file.txt`

Note:

- `>` : Overwrite
- `>>` : Append

## Writing to a file using redirection


```
myusername@ubuntu:~/osc/summer25$ cat hi.txt  
hi  
myusername@ubuntu:~/osc/summer25$ echo bye > hi.txt  
myusername@ubuntu:~/osc/summer25$ cat hi.txt  
bye
```

## appending to the file

```
myusername@ubuntu:~/osc/summer25$ cat hi.txt  
bye  
myusername@ubuntu:~/osc/summer25$ echo hi >> hi.txt  
myusername@ubuntu:~/osc/summer25$ cat hi.txt  
bye  
hi
```

## stderr redirection


```
myusername@ubuntu:~/osc/summer25$ ls
comp hi.txt redpip showfds
myusername@ubuntu:~/osc/summer25$ ls hi
ls: cannot access 'hi': No such file or directory
myusername@ubuntu:~/osc/summer25$ ls hi 2> errors.txt
myusername@ubuntu:~/osc/summer25$ ls
comp errors.txt hi.txt redpip showfds
myusername@ubuntu:~/osc/summer25$ cat errors.txt
ls: cannot access 'hi': No such file or directory
```



“What we want to do is a mechanism connecting programs together just like screwing pieces of garden hose together.”

- Doug McIlroy





“We should have some ways of connecting programs like garden hose..”

- Doug McIlroy



# Pipelining

A **pipeline** is a sequence of one or more commands separated by the pipe character (`|`).

A pipe connects the standard output of the first command to the standard input of the next command.



`wc -l` (a program that counts the number of lines)

```
myusername@ubuntu:~/osc/summer25/comp$ ls -l
total 0
-rw-rw-r-- 1 myusername myusername 0 Aug  5 22:07 file1
-rw-rw-r-- 1 myusername myusername 0 Aug  5 22:07 file2
-rw-rw-r-- 1 myusername myusername 0 Aug  5 22:07 file3
-rw-rw-r-- 1 myusername myusername 0 Aug  5 22:07 file4
-rw-rw-r-- 1 myusername myusername 0 Aug  5 22:07 file5
myusername@ubuntu:~/osc/summer25/comp$ ls -l | wc -l
6
```

`sort` (a program that sorts :))

```
myusername@ubuntu:~/osc/summer25/comp$ cat newfile
z
c
a
h
b
myusername@ubuntu:~/osc/summer25/comp$ cat newfile | sort
a
b
c
h
z
```

# Redirection vs Pipelining (Practical-wise)

Redirection: `command > file`

Pipelining: `command | command`







# Hands on #4

---

1. Create a file named **output.txt**
2. Redirect the output of this command **"echo hello"** to **output.txt**
3. Redirect the output of this command **"ls -l"** to **output.txt**
4. Count **output.txt** lines and redirect it to **count.txt**



# **Solution**

---

- Create a file named **output.txt**  
touch output.txt
- Redirect the output of this command “**echo hello**” to **output.txt**  
echo hello > output.txt
- Append the output of this command “**ls -l**” to **output.txt**  
ls -l >> output.txt
- Count **output.txt** lines and redirect it to **count.txt**  
cat output.txt | wc -l > count.txt

# Thanks!

