

COL1000: Introduction to Programming

File Systems, File I/O

Subodh Sharma | Lec 29 | Oct 24



Announcement

- **Monday's – 5 pm to 7 pm; Doubt learning sessions in Bharti 419**

What is a Filesystem?

- Imagine the disk as a **tree**:
 - Folders (or directories) are **nodes**
 - Files are **leaves**
 - Files: Containers in the storage system to store information
 - Common File type extensions: .txt, .doc, .jpg, .pdf, .mp3, .mov, .mp4, .exe, .app,

```
/ (Root Directory - The Disk)
├── folder1/
│   ├── fileA.txt
│   └── subfolder1a/
│       └── fileB.docx
├── folder2/
│   ├── image.jpg
│   ├── music.mp3
│   └── another_subfolder/
│       └── log.txt
└── fileC.py
```

What is a Filesystem?

- Each file has a:
 - **path** (its address),
 - **name**,
 - **size**,
 - **timestamp** (creation/modification),
 - **permissions** (who can read/write/execute)

```
Subodhs-MacBook-Pro:lec29 sv$ pwd
/Users/svs/svs-research/classes-IITD/col100-intro-to-compsci
ence/2025/lectures/lec29
```

```
Subodhs-MacBook-Pro:lec29 sv$ ls -alh
total 1984
drwxr-xr-x  3 sv$  staff   96B  24 Oct 10:01 .
drwxr-xr-x 30 sv$  staff  960B  24 Oct 09:37 ..
-rw-r--r--@ 1 sv$  staff  953K  24 Oct 10:01 lec29.key
```

\underbrace{d} \underbrace{rw}_{-} $\underbrace{-x}_{--}$ $\underbrace{r-x}_{--}$
dir/file/symlink Owner:read/write Group:execute Others:read/execute

A note about Paths

- **Path** can be:
 - **Absolute:** starts at the root, denoted by `/`
 - Eg: `/Users/svs/svs-research/classes_IITD/col100-intro-to-compscience/2025/lectures/lec29`
 - **Relative:** relative to the current working directory (CWD)
 - Eg: `Subodhs-MacBook-Pro:lec29 svs$ open ../lec29.key`
- Home directory is usually denoted by `~`
 - Eg: `Subodhs-MacBook-Pro:lec29 svs$ cd ~/svs-research/`
`Subodhs-MacBook-Pro:svs-research svs$`

Interacting with the FileSystem

Common Shell Commands

- Caution — while deleting, moving or copying files.

Task	macOS/Linux (bash/zsh)
Show path	<code>pwd</code>
List files	<code>ls -la</code>
Change dir	<code>cd path/</code>
Make dir	<code>mkdir newdir</code>
Copy file	<code>cp a.txt b.txt</code>
Move/rename	<code>mv a.txt b.txt</code>
Delete file	<code>rm file.txt</code>
Delete dir (recursive)	<code>rm -rf mydir</code> (danger!)
Show file	<code>cat file.txt</code>
View paged	<code>less file.txt</code>

Where do files become important?

- Commonly used for logs, storing configurations, datasets, reports etc.
- The **stored data persists** beyond program execution
 - Allows sharing of data across processes, devices, organisations, etc.
 - Provide **access-control** for **authorised** access
- Finally — it offers **great abstraction**; the OS presents the data to the program as a simple named file object while hiding the complex details of how and where it is stored/accessed on the disk

Python Support for File Management

- Use **pathlib**: to work with file paths
 - Handles file separators for you (/ or \)
- **File Input/Output Basics**
 - Opening a file: **open(<file_name>, mode, encoding)**
 - Eg:

```
f = open("data.csv", "r")  
content = f.read()  
f.close()
```



Explicitly closing the file

File Open — with exception handling

- Opening the file in a safer way:

```
try:
    with open("data.csv", "w") as f:
        content = f.read()
        print content
    #automatic closing of the file
except FileNotFoundError:
    print("The file was not found!")
```



Automatically closes the file

File Open — Modes

Mode	Symbol	Description
Read	'r'	Reads from a file. This is the default. Raises an error if the file does not exist.
Write	'w'	Writes to a file. Overwrites the entire file if it exists, or creates a new one if not.
Append	'a'	Appends to the end of a file. Creates a new file if it does not exist.
Create	'x'	Exclusively creates a new file. Fails with an error if the file already exists.
Text	't'	Opens in text mode . This is the default.
Binary	'b'	Opens in binary mode (for non-text files like images or executables).
Update	'+'	Opens for updating (reading and writing). Can be combined with other modes (e.g., <code>r+</code> , <code>w+</code>).

File Reading — Other options

- Reading one line: `f.readline()`
- Reading all lines as a list: `f.readlines()`
- Reading the content iteratively:

```
try:
    with open("data.csv", "r") as f:
        for line in f:
            print(line.strip())
except FileNotFoundError:
    print("The file was not found!")
```

File Reading—

- Reading all at once:

```
try:
    with open("data.csv", "w") as f:
        content = f.read()
        print content
    #automatic closing of the file
except FileNotFoundError:
    print("The file was not found!")
```



Reading everything at once