

Grade: 65/65 A+

File Operations and File Types within Operating Systems

Alejandro Ricciardi

Colorado State University Global

CSC300: Operating Systems and Architecture

Joe Rangitsch

July 23, 2023

File Operations and File Types within Operating Systems

In the context of operating systems, understanding file operations and file types is crucial for the efficient manipulation and management of files. In this essay, I will discuss the seven basic file operations and their purpose. Furthermore, I will discuss how file types are used to indicate the internal structure of a file. Finally, the seven basic file operations are crucial for the functionality of applications and operating systems, and understanding the association between file types and file internal structures is essential for effectively processing, manipulating, and storing files. Both the seven basic file operations and the file types concept work together to facilitate the efficient management, manipulation, and understanding of data files within an operating system.

The Purpose of the Seven Basic File Operations

Any file system not only provides the means to store data but also offers a set of operations that allow files to be manipulated (Stallings, 2018). The seven basic file operations are: create, delete, open, close, read, write, and copy.

1. The create operation's goal is to create a new file within a structure of files (Stallings, 2018).

In Linux systems, creating a simple text file can be accomplished by using the command 'touch' in the command line, or by using a simple text editor such as nano also from the command line, see Figures 2 and 3 for examples. To create a directory from the command line the command 'mkdir' is utilized, see Figure 1 for example.

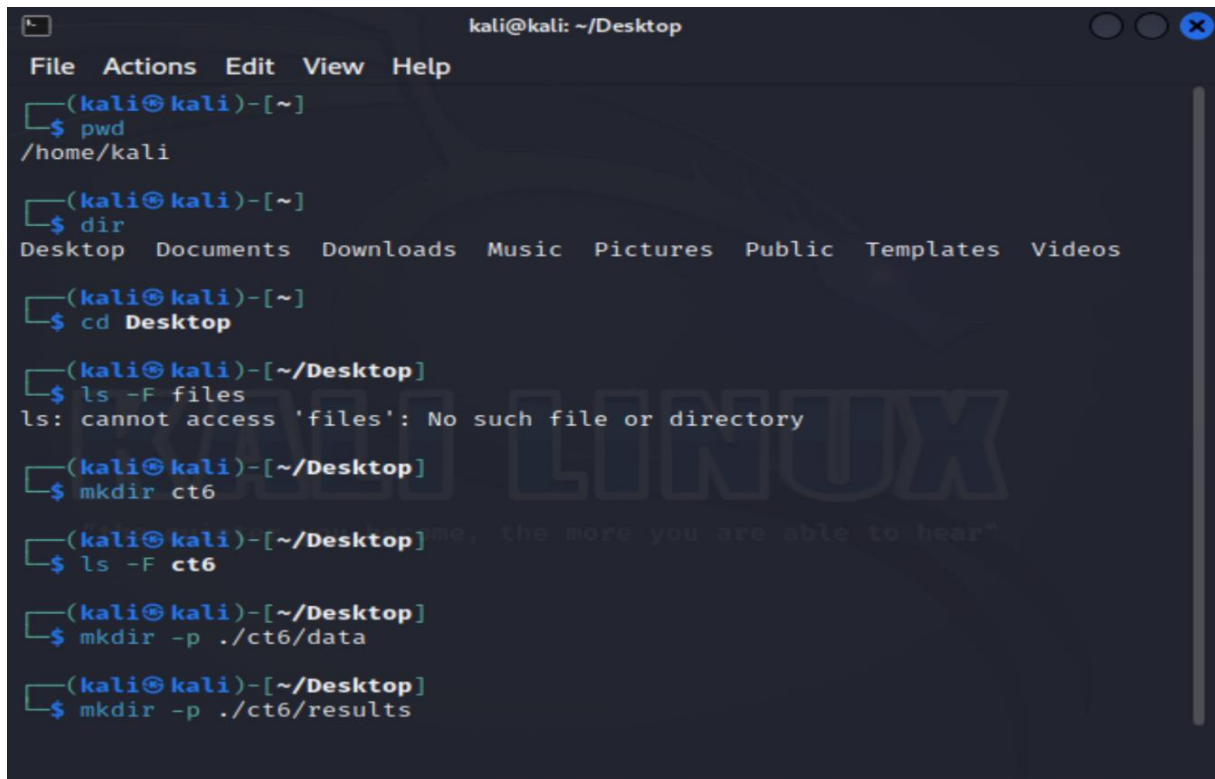
2. The delete operation's goal is to remove the file from the structure and destroy it. This can be done by applications, operation system services like file explorer for Windows system, or from the command line. In Linux systems, this can be accomplished by utilizing the command 'rm' from the command line, See Figure 1 for example.

3. The open operation's goal is to declare a file to be open allowing a process to perform functions on the file. This is usually utilized by applications to prevent other applications from accessing the file. Note that the command 'open' in the command line will usually open the file in a simple text editor, for example in Kali Linux the file will open in Mousepad.
4. The close operation's goal is to close the file with respect to a process. The process will no longer allow to perform functions on the file. This is usually utilized by applications to release a file so it can be used by other applications.
5. The read operation's goal is to read all or part of a file. This is usually utilized by applications to extract data information from a file.
6. The write operation's goal is to add, delete, or modify data in the file. This is usually utilized by applications to manipulate data in a file.
7. The copy operation's goal is to recreate a file from a location in the structure of files by pasting it to a new location in the structure, without removing it from its original location. The move operation is related to the copy operation, its goal is to transfer an existing file from a location within a structure of files to a different location in that structure. In other words, the move operation copies a file from its original location, past it to a new location, and removes it from its original location. In Linux, the command 'cp' copies a file from a location to a new location. The command 'mv', on the other hand, moves a file from one location to a new location. See Figure 4 for examples of the commands 'cp' and 'mv'.

Therefore, the seven basic file operations' purpose is to manipulate file data and manage file storage within a file structure. Additionally, each operation serves a unique purpose.

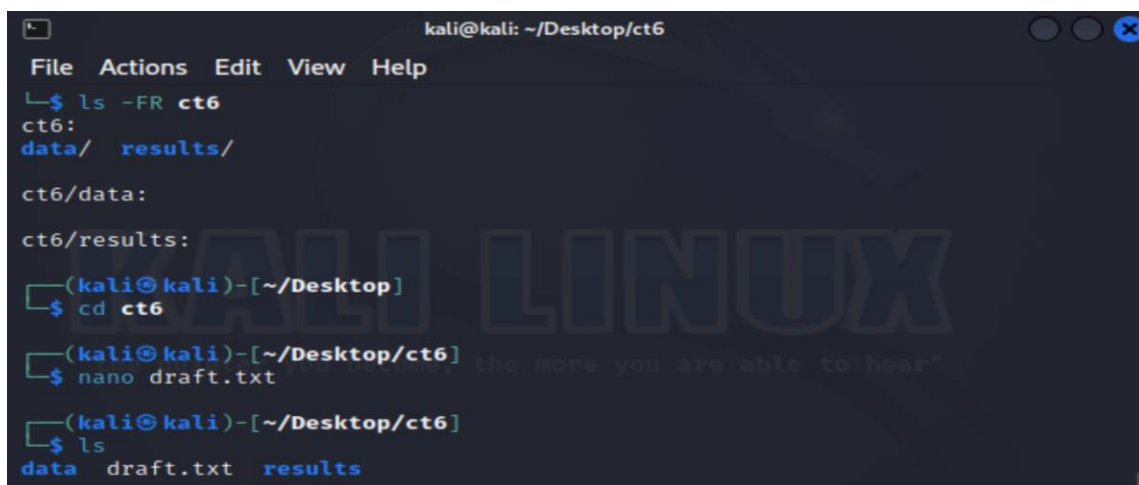
Furthermore, these operations are crucial for the functionality of applications and operating systems.

Figure 1

mkdirA terminal window titled 'kali@kali: ~/Desktop' with a menu bar (File, Actions, Edit, View, Help). The terminal shows a sequence of commands: 'pwd' returns '/home/kali'; 'dir' lists desktop icons; 'cd Desktop' changes the directory; 'ls -F files' fails with 'No such file or directory'; 'mkdir ct6' creates a directory; 'ls -F ct6' shows 'ct6/'; 'mkdir -p ./ct6/data' creates a subdirectory; and 'mkdir -p ./ct6/results' creates another subdirectory.

```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~]
$ pwd
/home/kali
(kali@kali)-[~]
$ dir
Desktop Documents Downloads Music Pictures Public Templates Videos
(kali@kali)-[~]
$ cd Desktop
(kali@kali)-[~/Desktop]
$ ls -F files
ls: cannot access 'files': No such file or directory
(kali@kali)-[~/Desktop]
$ mkdir ct6
(kali@kali)-[~/Desktop]
$ ls -F ct6
ct6/
(kali@kali)-[~/Desktop]
$ mkdir -p ./ct6/data
(kali@kali)-[~/Desktop]
$ mkdir -p ./ct6/results
```

Figure 2

Commands: ls -FR ct6, cd ct6, nano draft.txt, and lsA terminal window titled 'kali@kali: ~/Desktop/ct6' with a menu bar (File, Actions, Edit, View, Help). The terminal shows: 'ls -FR ct6' listing 'ct6/data/' and 'ct6/results/'; 'cd ct6' changing to the subdirectory; 'nano draft.txt' creating a file; and a final 'ls' command listing 'data', 'draft.txt', and 'results'.

```
kali@kali: ~/Desktop/ct6
File Actions Edit View Help
$ ls -FR ct6
ct6:
data/ results/
ct6/data:
ct6/results:
(kali@kali)-[~/Desktop]
$ cd ct6
(kali@kali)-[~/Desktop/ct6]
$ nano draft.txt
(kali@kali)-[~/Desktop/ct6]
$ ls
data draft.txt results
```

Figure 3

Nano

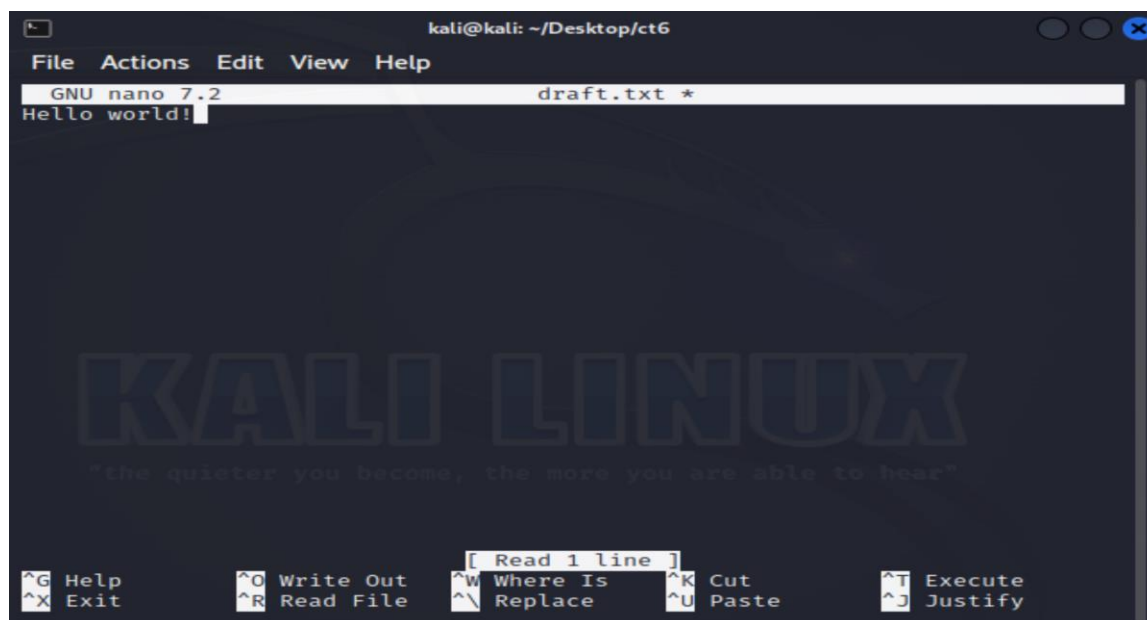
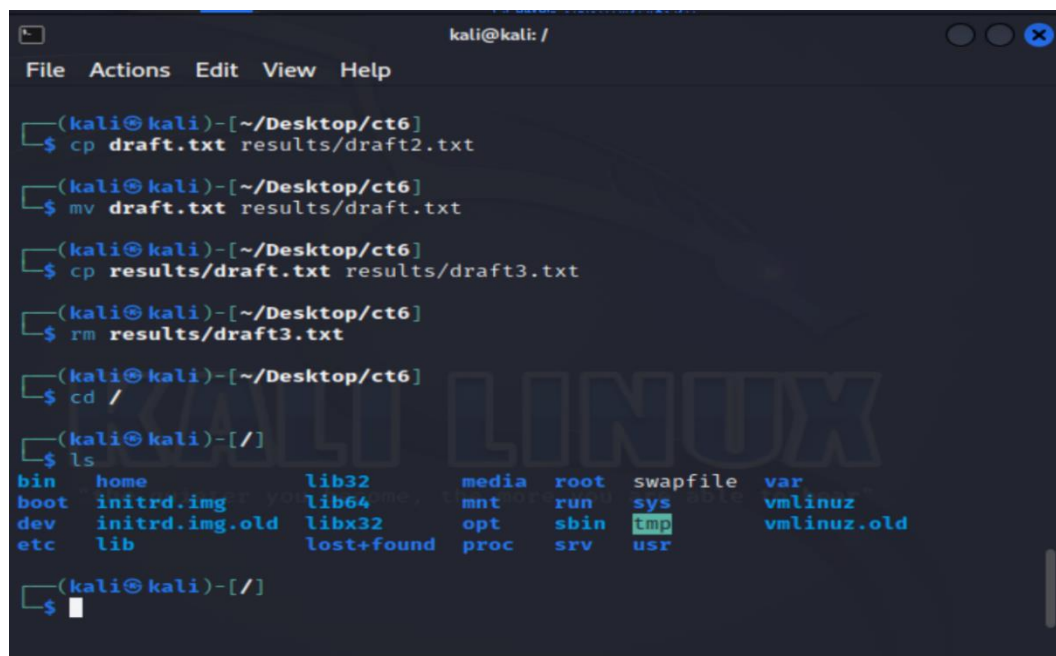


Figure 4

Command: rm, cp, and mv



How File Types Indicate the Internal Structure of a File

“A common technique for implementing file types is to include the type as part of the file name” (Silberschatz et al, 2018. p. 536). This allows the user and the operating system to tell the type of the file by the name alone. The Windows operating systems add a file-name extension to their files, for example, a file with ‘exe’ extension is an executable file, indicating that the file's internal structure is an executable machine language structure. A ‘gif’, ‘pdf’, or ‘jpg’, on the other hand, indicates to the user that the file is a picture, but also that the file's internal structure is an ASCII or binary structure, this is useful for image editing application to tell which of the two internal structures a picture file has. For more examples, see Figure 5. In comparison, the Unix operating system does allow file-name extension hints, but it does not enforce or rely on these extensions. Unix utilizes magic numbers at the beginning of files to distinguish between executable files (Underwood, 2009). Furthermore, “many other file types, including file formats used in other operating systems, now have a magic number somewhere in the file format” (Underwood, 2009, p. 5). However, not all files contain a magic number. In these cases, the file(1) command can be utilized, among other tests, to identify a file's type or magic number. The command returns the file type (Kerrisk, 2010). Finally, file types indicate the internal structure of files, allowing users, applications, and operating systems to identify the internal format. Therefore, understanding the

Figure 5*File-name Extensions*

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

Note. From *Operating System Concept*, by Silberschatz, Galvin, & Gagne, 2018, Figure 13.3. Copyright 2023 by Wiley.

association between file types and their internal structures is crucial for effectively processing, manipulating, and storing files within any operating system.

Conclusion

To summarize, I discussed the seven basic file operations and their purpose, which are create, delete, open, close, read, write, and copy. I also explained how file types are used to indicate the internal structure of a file. The purpose of the seven basic file operations is to manipulate file data and manage file storage within a file structure, with each operation serving a unique function. These operations are critical for the proper functioning of applications and operating systems. Additionally, understanding the relationship between file types and their internal structures is essential for effectively processing, manipulating, and storing files within any operating system. Thus, both the seven basic file operations and the concept of file types work together to ensure efficient management, manipulation, and understanding of data files within an operating system.

References

Kerrisk, M. (2010, October). *The Linux Programming Interface*. Man-Pages Project.

<https://man7.org/linux/man-pages/man1/file.1.html>

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* [PDF][Figure 5].

Wiley. Retrieved from: <https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf>

Stallings, W. (2018). *Operating systems: Internals and design principles*. Pearson.

Underwood, W. (2009, September). Extensions of the UNIX file command and magic file for file

type identification [PDF]. *Georgia Tech Research Institute*. Retrieve from:

<https://www.archives.gov/files/applied-research/papers/unix-file-command.pdf>