UNIX SHELL PROGRAMMING

Background and Basic Commands

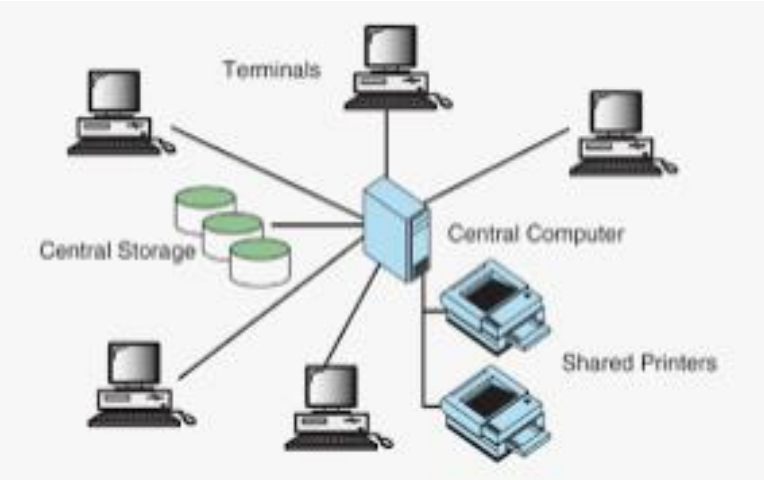
- Computer system is a combination of hardware & software that lets to work with the computer
  - Hardware (physical component)
  - Software(set of instructions or programs) that allows hardware to do its job .
    - System software
    - Application software

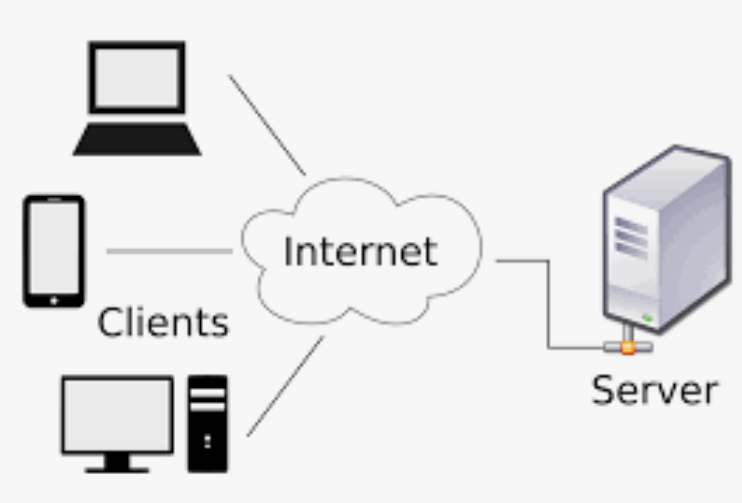
Operating System

- OS is a system software that manages all operating Facets of the computer
- Different types of OS – DOS ,Windows ,MAC OS, UNIX etc.
- Functions of OS –
  - Resource allocation and scheduling
  - Data management (file I/p & O/p)
  - System security
  - Copying , sorting, text creation , editing etc .

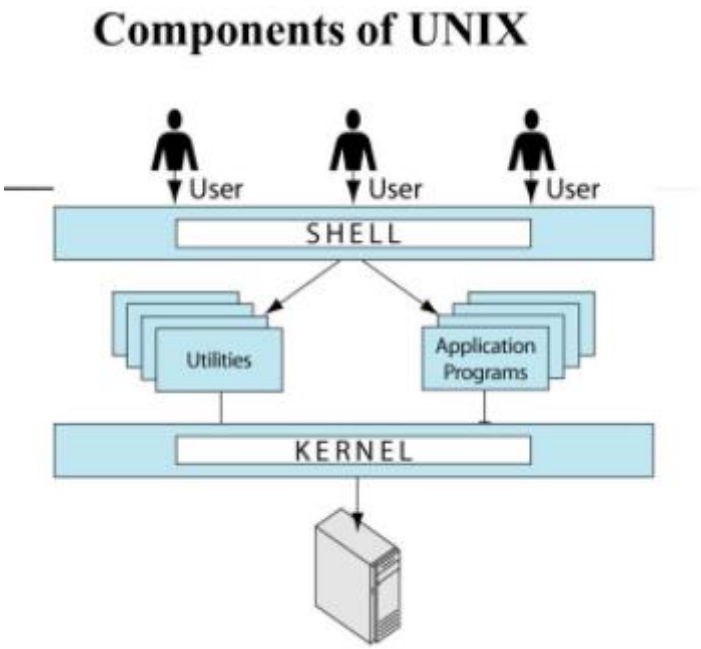
UNIX Environment

- UNIX is a multi user multi processing portable OS , designed to facilitate programming text processing , communication and many more task .
- Computing environment –
    1. Stand Alone
    2. Time sharing
    3. Client and server


Terminals
Central Storage
Central Computer
Shared Printers


Internet
Clients
Server

Components of UNIX



Components of UNIX (cont..)

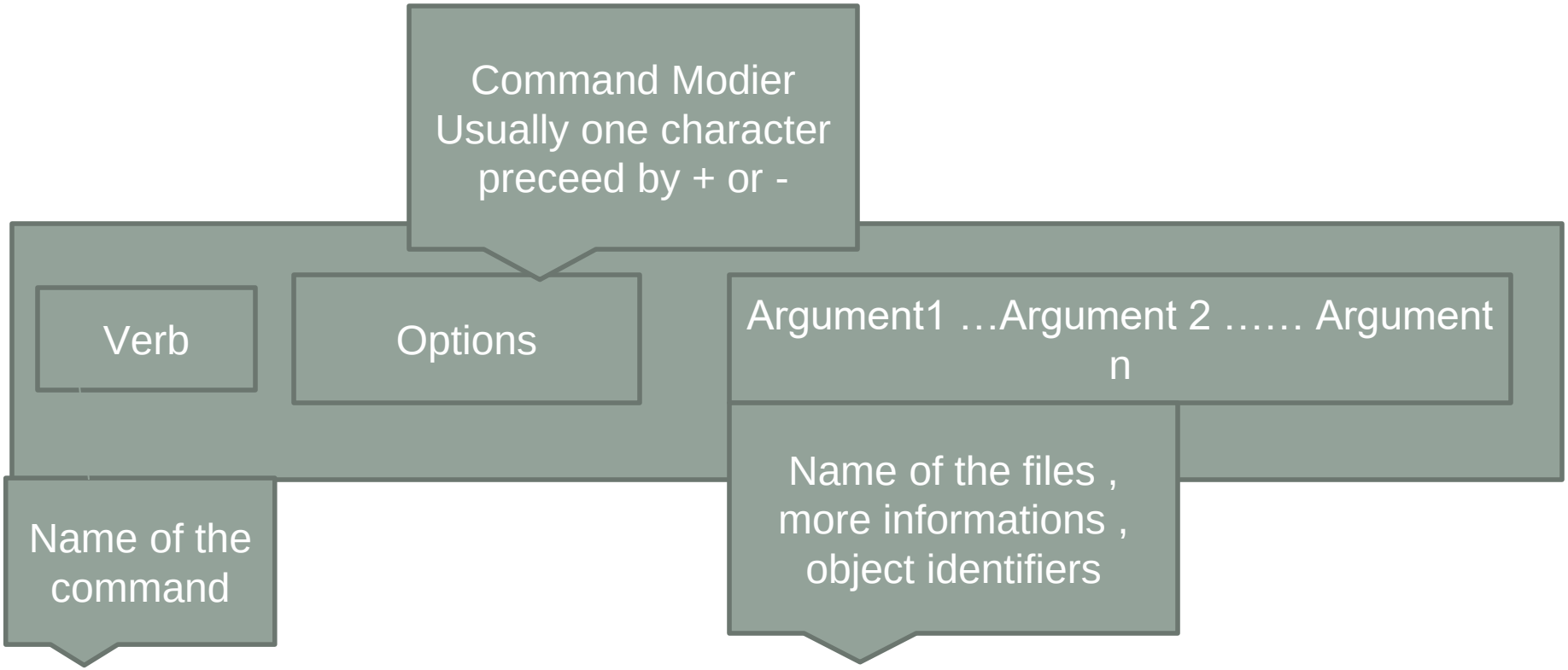
- Kernel – contains process control and resource management of the OS. All other components of the system call on the kernel to perform these services for them .
- Shell – receives and interprets the commands entered by the user.
- Types of Standard UNIX shells
  - BOURNE shell (developed by Steve Bourne at AT&T Labs)
  - BASH (BOURNE again shell)
  - C shell (Developed in Berkeley by Bill Joy)
  - Corn Shell (Developed by David Corn)

Components of UNIX (cont..)

- Two major parts of shell –
  - Interpreter
  - Shell script
- Utilities - Is a standard UNIX program that provide support process for users . Ex text editors , sort and search programs
- Application programs – are written by system administrators , professional programmers or users . They provide extended capability to the system

Commands

- A UNIX command is a action request given to UNIX shell for execution .



Basic commands

- **Date command –**
  - Syntax –

```
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

  - Examples - $ date

    $ date –u

    $ $date “+this month : %a”

format code	Part of Date	Desciption	Example Output
date +%a	Weekday	Name of weekday in short (like Sun, Mon, Tue, Wed, Thu, Fri, Sat)	Mon
date +%A	Weekday	Name of weekday in full (like Sunday, Monday, Tuesday)	Monday
date +%b	Month	Name of Month in short (like Jan, Feb, Mar)	Jan
date +%B	Month	Month name in full short (like January, February)	January
date +%d	Day	Day of month (e.g., 01)	4
date +%D	MM/DD/YY	Current Date; shown in MM/DD/YY	02/18/18
date +%F	YYYY-MM-DD	Date; shown in YYYY-MM-DD	19-01-2018
date +%H	Hour	Hour in 24-hour clock format	18
date +%I	Hour	Hour in 12-hour clock format	10
date +%j	Day	Day of year (001..366)	152
date +%m	Month	Number of month (01..12) (01 is January)	5
date +%M	Minutes	Minutes (00..59)	52
date +%S	Seconds	Seconds (00..59)	18
date +%N	Nanoseconds	Nanoseconds (000000000..999999999)	300231695
date +%T	HH:MM:SS	Time as HH:MM:SS (Hours in 24 Format)	18:55:42
date +%u	Day of Week	Day of week (1..7); 1 is Monday	7
date +%U	Week	Displays week number of year, with Sunday as first day of week (00..53)	23
date +%Y	Year	Displays full year i.e. YYYY	2018
date +%Z	Timezone	Time zone abbreviation (Ex: IST, GMT)	IST

Basic command (continue..)

- **Calendar command –**
  - Syntax – $cal [[month] year]
    - Example - $cal (displays current month)
      $cal 2001 (displays 2001 year calander)
      $cal 2 2002 (displays 2nd month of 2002 year).

- **Who command –**
  - Syntax - $who [-uH] [ARG 1 ARG2]
    - -u : shows how long it has been since there was any activity on the line(ideal time)
    - -H : shows the header that explain each column
    - If you want to see your user id “ $ Who am i ”

- **whoami –**
  - The system returns your user id.
  - Administrators use this command to find out who has left the terminal unguarded.

- **Change Password(passwd) Command –**
  - This command is used to change your password.
  - It begins by asking you to enter your old password, it is done for security reasons.
  - After you verify your password, the system asks you for a new password.

- **Print Message(echo) Command –**
  - The echo command copies its argument back to the terminal.
  - $ echo Hello World

    Hello World

- **Online Documentation (man) Command –**
  - One of the most important UNIX command is man.
  - The man command displays online documentation.
  - -k : man command with this option will display information, including commands, about the topic.

- **Print (lpr) Command –**
  - The most common print utility is line printer (lpr).
  - The line printer utility prints the contents of the specified files to either the default printer or to a specifies printer.
  - -p : to direct the output to a specified printer.

Other Useful Commands

- **Terminal (tty) Command –**
  - The tty utility is used to show the name of the terminal you are using.
  - $ tty
  - /dev/ttyq0

- **Clear screen (clear) Command –**
  - The clear command clears the screen and puts the cursor at the top.
  - $ clear

- **Set terminal (stty) Command –**
  - The set terminal command sets or unsets selected terminal input/output options.
  - When the terminal is not responding properly, the set terminal command can be used to reconfigure it.
  - -a : display current settings
  - -g : display current settings (in argument format)
  - stty **without any options and arguments**, it shows the current common setting for your terminal. Such as communication settings like baud rate, Delete key setting (default ctrl-h).

  - **Set Terminal with Arguments**
  - Set Erase and Kill (ek) – The ek argument sets the default erase (Delete key – ctrl+h) and kill (ctrl+c) to their defaults.
  - Set Terminal to General Configuration (sane) – The sane argument sets the terminal configuration to a reasonable setting that can be used with a majority of the terminals.

- Set Erase Key (erase) – By default, the Erase key is ctrl+h, it deletes the previous character typed.
- $ stty erase ^e (ctrl+e)
- Set Kill (kill) – The kill key deletes a whole line. By default, it is ctrl+u.
- We can change it using the set terminal command with the kill argument, but it's highly recommended that you don't use a single key because its easy to forget and use it accidently in typing normal commands or input.
- $ stty kill 9
- Set Interrupt Key (intr) – The interrupt key interrupts or suspend a command. By default, it is ctrl+c. It can be reset using the intr argument as shown.
- $ stty intr ^9

- There are many more stty command options and arguments. Many of them are applicable only to superusers.

TABLE 1.2 *Common Control Keys That Can Be Set*

Command	Attribute	Default
Erase text	erase	^h
End of file key	eof	^d
Kill command	kill	^u
Interrupt command	intr	^c
Start output	start	^q
Stop output		^d
Suspend command		^z

- **Record Session (script) Command –**
  - The script command can be used to record an interactive session.
  - When you want to start recording, key the command. To record a whole session, including the logout, make it the first command of the session.
  - To stop the recording, **key exit**. The session log is in a file named typescript.
  - After keying exit, you can print the script output file using the lpr command.
  - $ script

  Script started, file is typescript
  - $ script myfilename
  - Each script command execution erases the old script file output. To append to the file rather than erase it, we use the append option (-a).
  - $ script -a

- **System Name (uname) Command –**
  - Each UNIX system stores data, such as its name, about itself. To see these data, we use **uname** command.
  - -a : we can display all of the data.
  - -n : we can specify only the name.
  - -s : operating system.
  - -r : software release.
  - Options can be combined to obtain desired data.
  - -sr : to display the os and its release.
- **Calculator (bc) Command –**
  - The bc command is one of the most interesting commands in UNIX. It turns UNIX into a calculator.
  - In many respects, it is actually a language, similar to C, with a powerful math library.
  - To start the calculator, we simply key the bc command. To terminate it, we key end of file (ctrl+d).

- Simple Arithmetic is done at the command line. It supports addition, subtraction, multiplication, division, modulus, and power.

- **Floating-Point Calculations –**
  - To use floating point arithmetic, we must specify the number of decimal points to be used.
  - This is done with the scale expression, which sets the number of digits after the decimal in a floating-point number.
  - $ bc
  - 19/3 ->6
  - scale=2
  - ->6.33

- **Arithmetic Base Calculations –**
  - The bc calculator can be used in decimal, binary, octal, or hexadecimal bases.
  - The base is specified by one of two expression : ibase **or** obase.
  - The ibase expression specifies that input will be in the specified base.
  - The obase expression specifies the output base.
  - If the input or output base is not defined, it is assumed to be decimal.
  - You can change the base of your calculation by setting the input base(ibase) or output base(obase) to the base you want.
  - $ bc
  - ibase=2 obase=2
  - 111 ->7 5 ->101
  - ibase=16 obase=16
  - 1A ->26 26 ->1A

The Unix file system

- Unix file system is a logical method of **organizing and storing** large amounts of information in a way that makes it easy to manage.

- All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the **file system.**

File names

- There are few restrictions on how you make up filenames in UNIX.
- Some implementations limit the length of a filename to 14 characters, other have name as long as 255 characters.
- A filename can be any sequence of ASCII characters.
- Some characters cannot be used in a filename, such as greater than(>) and less than(<) character because they used for file redirection.
- A period in UNIX does not have a special meaning as it does in other operating system.
- In UNIX, you can use the period anywhere in a filename.

- **Rules to be followed for naming the file :**
  - Start your file name with an alphabetic character.
  - Use dividers to separate parts of the filename. Good dividers are the underscore, period, and the hyphen.
  - Use an extension at the end of the filename, even though UNIX doesn’t recognize extensions. Some application, such as compilers, require file extensions.
  - A few common extensions are .txt for text files, .dat for data files, .bin for binary files, and .c and .c++ for c and c++ files respectively.
  - **Never** start a filename with a period. Filename that start with a period are hidden files in UNIX.
  - Generally, hidden files are created and used by the system. They will not be seen when you list your files.

- **Wildcards**

  - Each filename must be unique. We may want to copy or list all files belonging to a project, we can group files together using wildcards that identify portions of filenames that are different.

  - A wildcard is a token that specifies that one or more different characters can be used to satisfy a specific request. Wildcards are likely blanks that can be filled in by any character.

  - There are three wildcards in UNIX : the single character(?) wildcard, the set([….])wildcard, the multiple character wildcard(*)wildcard.

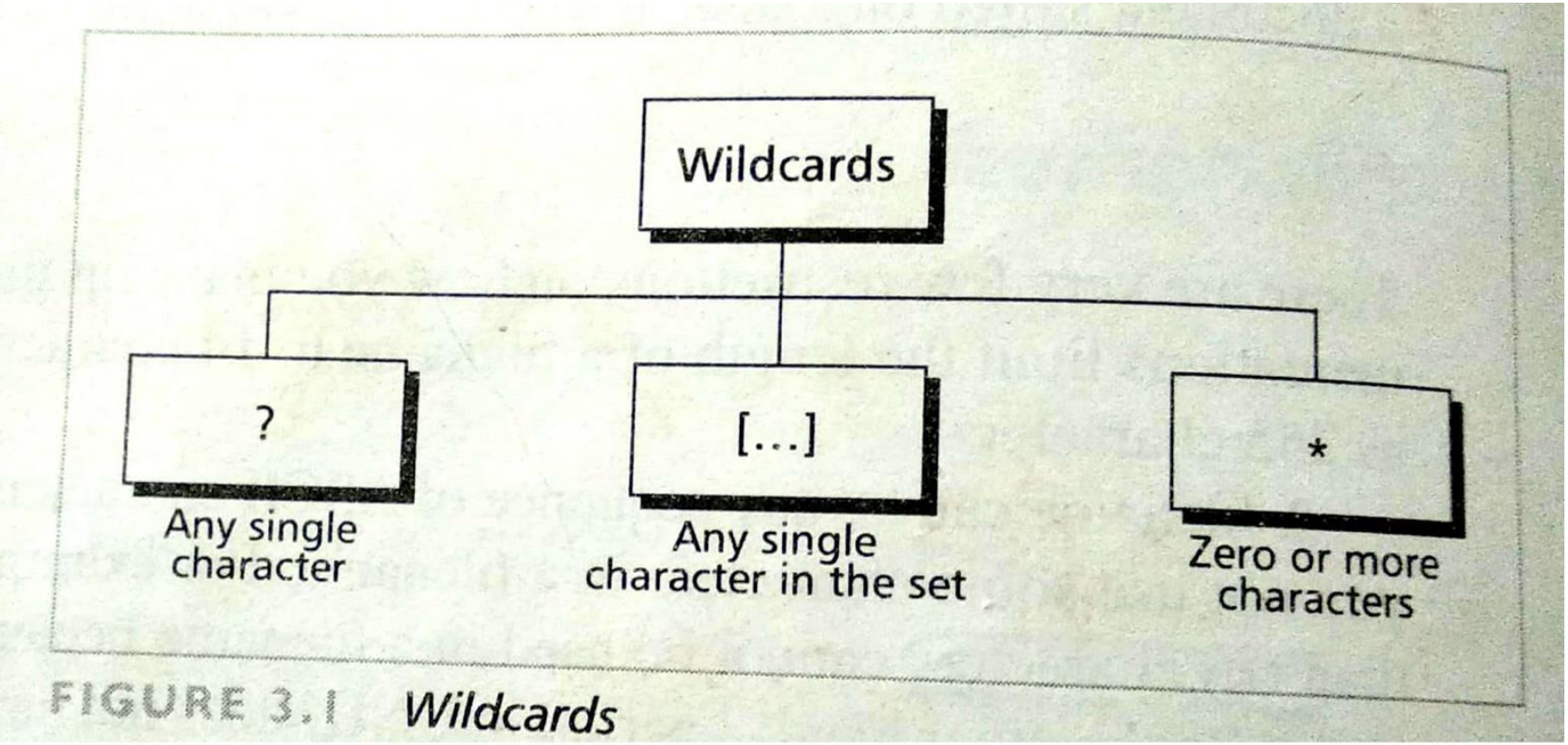


FIGURE 3.1 *Wildcards*

- **Matching any single character –**
  - Often it is necessary to group a set of files that all have the same filename except for one or two characters(Eg. Red1,red2,red3).
  - The single-character wildcard can be used more than once in a group. However, that each ? can match only one character.

TABLE 3.1 *Examples of Matching Any Single-Character (?)*

File	Matches				Does Not Match	
c?	c1	c2	c3	ca	ac	cat
c?t	cat	cet	cit	c1t	cad	dac
c??t	caat	cabt	cact	c12t	cat	daat
?a?	bat	car	far	mar	bed	cur

- **Matching a single character from a set –**
  - The single-character wildcard provides powerful capabilities to group files.
  - However, it is too powerful and we need to narrow the grouping. In this case, we use the character set.

TABLE 3.2 *Examples of the Character Set [...] Wildcard*

File	Matches				Does Not Match	
f[aoei]d	fad	fed	fod	fid	fud	fab
f[a-d]t	fat	fbt	fct	fdt	fab	fet
f[A-z][0-9]	fA3	fa3	fr2	f^2	FA3	fa33

  - To limit the test to only alphabetic characters, we should use [A-Za-z].

- **Matching Zero or More characters –**
  - The asterisk(*) is used to match zero or more characters in a filename.
  - The single-character wildcard matched one and only one character, the asterisk matches everything, and everything includes nothing.
  - Obviously, it is very powerful and you must be very careful when you use it.

TABLE 3.3 *Examples of Matching Zero or More Characters (*)*

Wildcard	Matches	Example	Does Not Match
*	every file		
f*	every file whose name begins with *f*	f5c2	afile, cat
*f	every file whose name ends in *f*	staff	f1, faF
.	every file whose name has a period	file.dat	bed, cur

File Types

- UNIX provides seven file types.

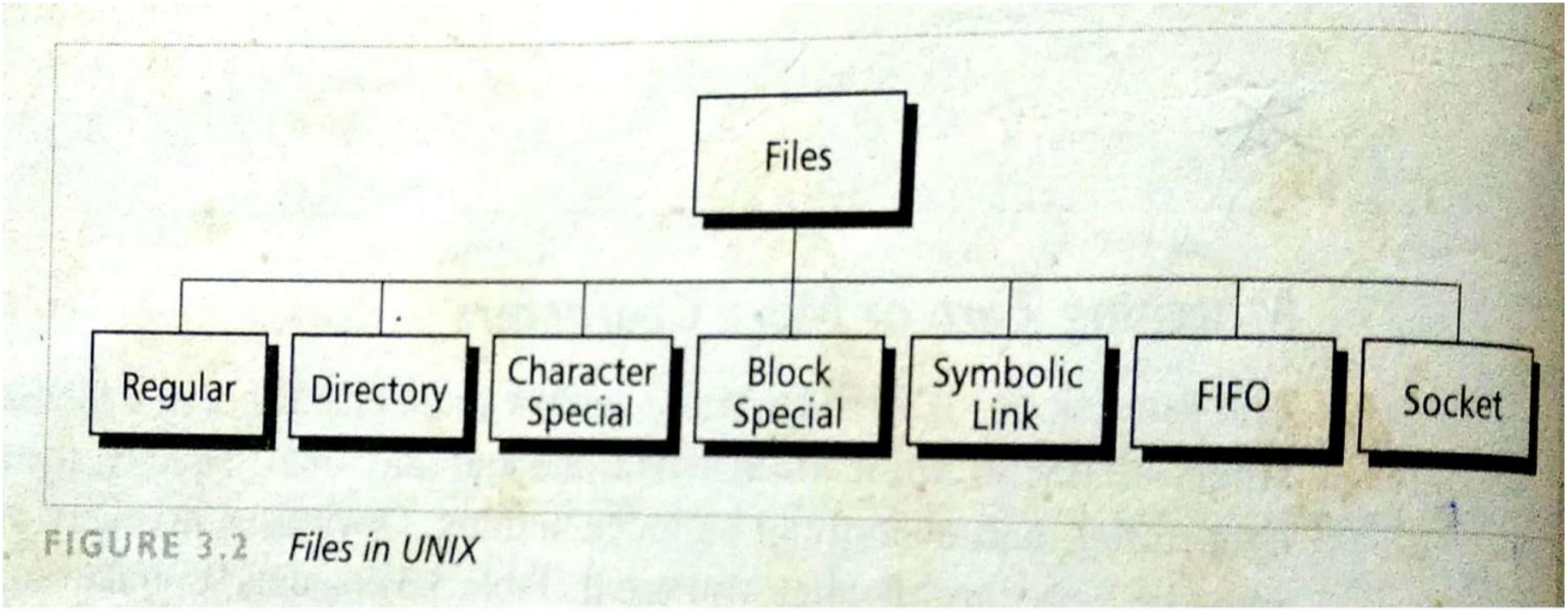


FIGURE 3.2 *Files in UNIX*

- **Regular Files –**
  - Regular files contains user data that need to be available for future processing. Sometimes called ordinary files, regular files are the most common files found in a system.
- **Directory Files –**
  - A directory is a file that contains the name and location of all the files stored on a physical device.
- **Character special Files –**
  - A character special file represents a physical devices, such as terminal, that reads or writes one character at a time.
- **Block special Files –**
  - A block special file represents a physical devices, such as a disk, that reads or writes data a block at a time.
- **Symbolic Link Files –**
  - A symbolic link is a logical file that defines the location of another file somewhere else in the system.

- **FIFO Files** –
  - A first-in, first-out file, also known as a named pipe, is a file that is used for interprocess communication.
- **Socket** –
  - A socket is a special file that is used for network communication.

- **Regular Files –**
  - The most common file in UNIX is the regular file. Regular files are divided by the physical format used to store the data as text or binary.
  - The physical format is controlled by the application program or utility that processes it.
  - UNIX views both format as a collection of bytes and leaves the interpretation of the file format to the program that processes it.

  - **Text file –**
  - A text file is a file of characters drawn from the computer's character set.
  - UNIX computers use the ASCII character set, the text file is the most UNIX file.

  - **Binary Files** –
  - A binary file is a collection of data stored in the internal format of the computer.

- There are two types of binary files :- **data files** and **program file**s.
- Data files contain application data.
- Program files contain instructions that make a program work.
- If you try to process a binary file with a text-processing utility, the output will look very strange because it is not in a format that can be read by people.

- **Directories –**
  - UNIX has a provision for organizing files by grouping them into directories.
  - A directory performs the same function as a folder in a filling cabinet.
  - It organizes related files and subdirectories in one place.

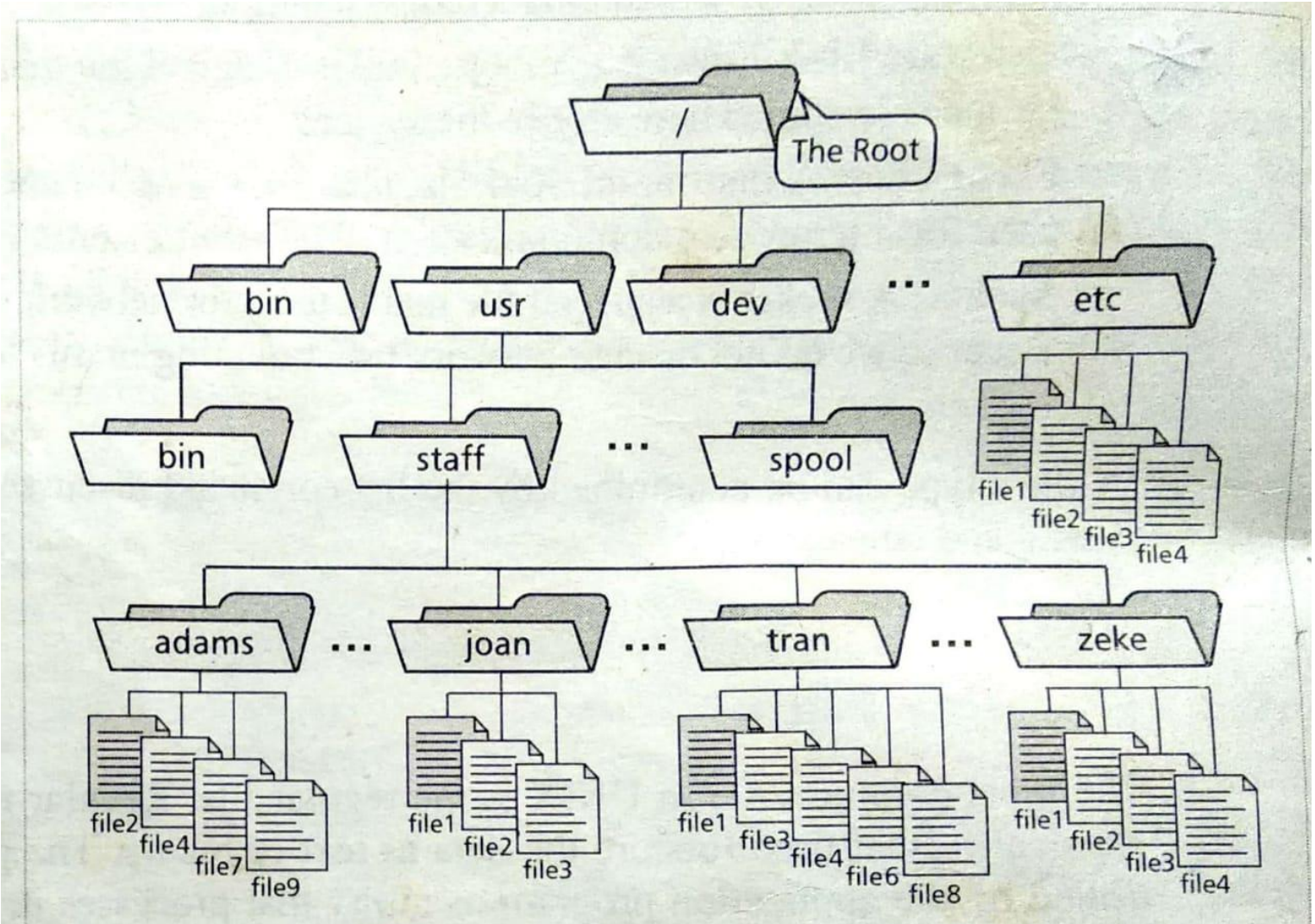


FIGURE 3.3 *A Directory Hierarchy*

- **Special directories** :
  - **Root** - is the highest level in the hierarchy . It doesn't have a parent. It can have several levels of sub directories . It can only be changed by the system admin.
  - **Home** - we use a home directory when we first login to the system . It contains all Files which we create. Each user has a home directory . The name of the home directory is the user login ID.
  - **Working** – the current/working directory is the one which we are in any point in a session . When we start , the working directory is the ‘Home’ directory.
  - **Parent** – is immediately above the working directory .

- **Path And Path Names –**
  - **Path** – to uniquely identify a file we require file's path .
    - **Absolute path name** - specifies the full path from the root to the desired directory or file.
    - **Relative path name** - is a path from the working directory to the file. Search starts from the working directory .

TABLE 3.4 *Absolute Pathnames for Some Files in Figure 3.3*

File	File Absolute Pathname	Directory Absolute Pathname
file1	/etc/file1	/etc
file2	/usr/staff/adams/file2	/usr/staff/adams
file1	/usr/staff/joan/file1	/usr/staff/joan

Continued

TABLE 3.4 *Absolute Pathnames for Some Files in Figure 3.3—Continued*

File	File Absolute Pathname	Directory Absolute Pathname
file1	/usr/staff/tran/file1	/usr/staff/tran
file1	/usr/staff/zeke/file1	/usr/staff/zeke

- **Relative path names abbreviations –**
  - Home – “ ~ “
  - Working – “ . ”
  - Parent – “ .. ”

TABLE 3.5 *Relative Pathname Examples*

Working Directory	Relative Path to `file3`
1. /etc	Use absolute pathname; it's faster.
2. /	usr/staff/joan/file3
3. /usr	staff/joan/file3
4. /usr/staff	joan/file3
5. /usr/staff/joan	file3
6. /usr/staff/adams	../joan/file3 or ~joan/file3
7. /usr/staff/adams/reports	../../joan/file3 or ~joan/file3

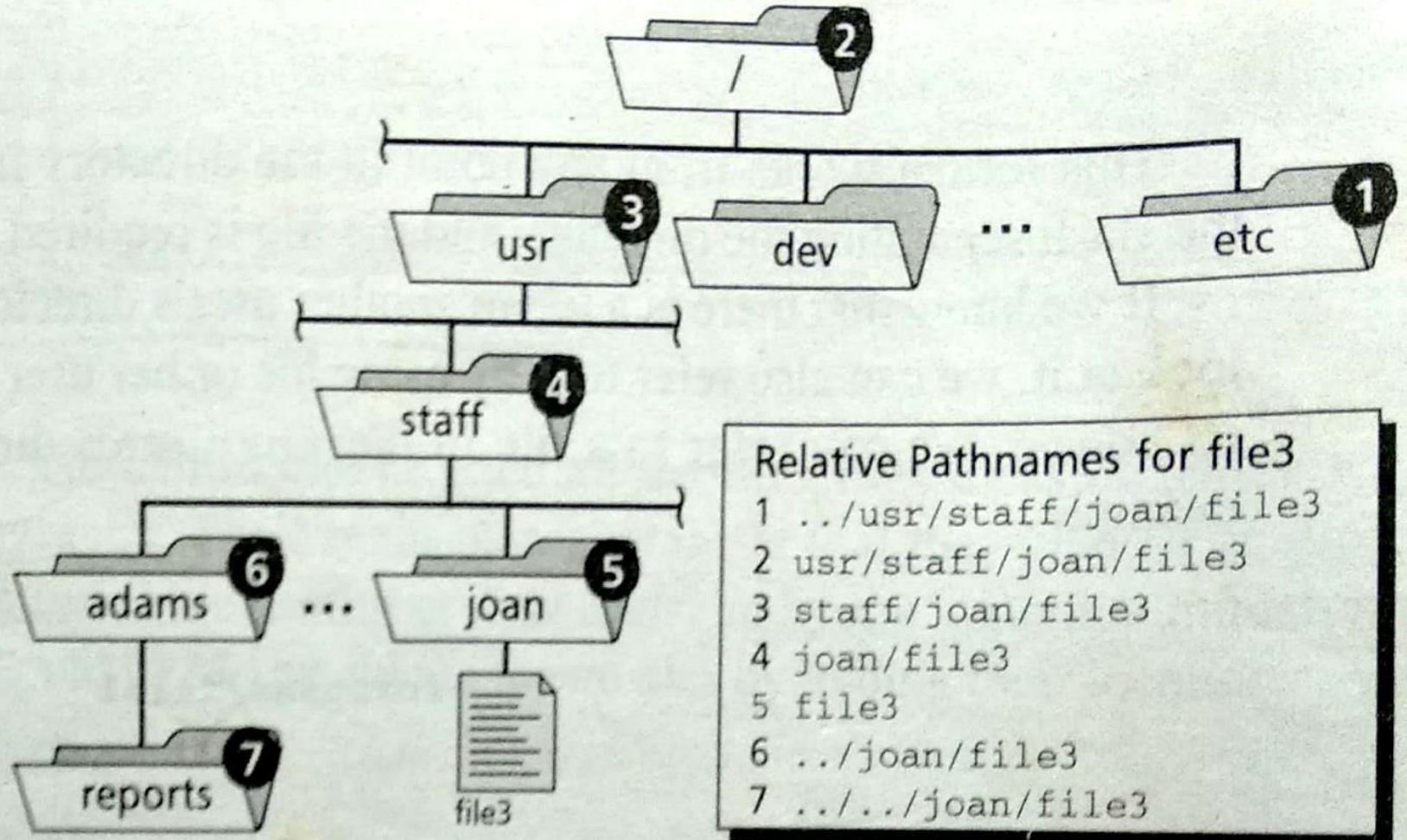
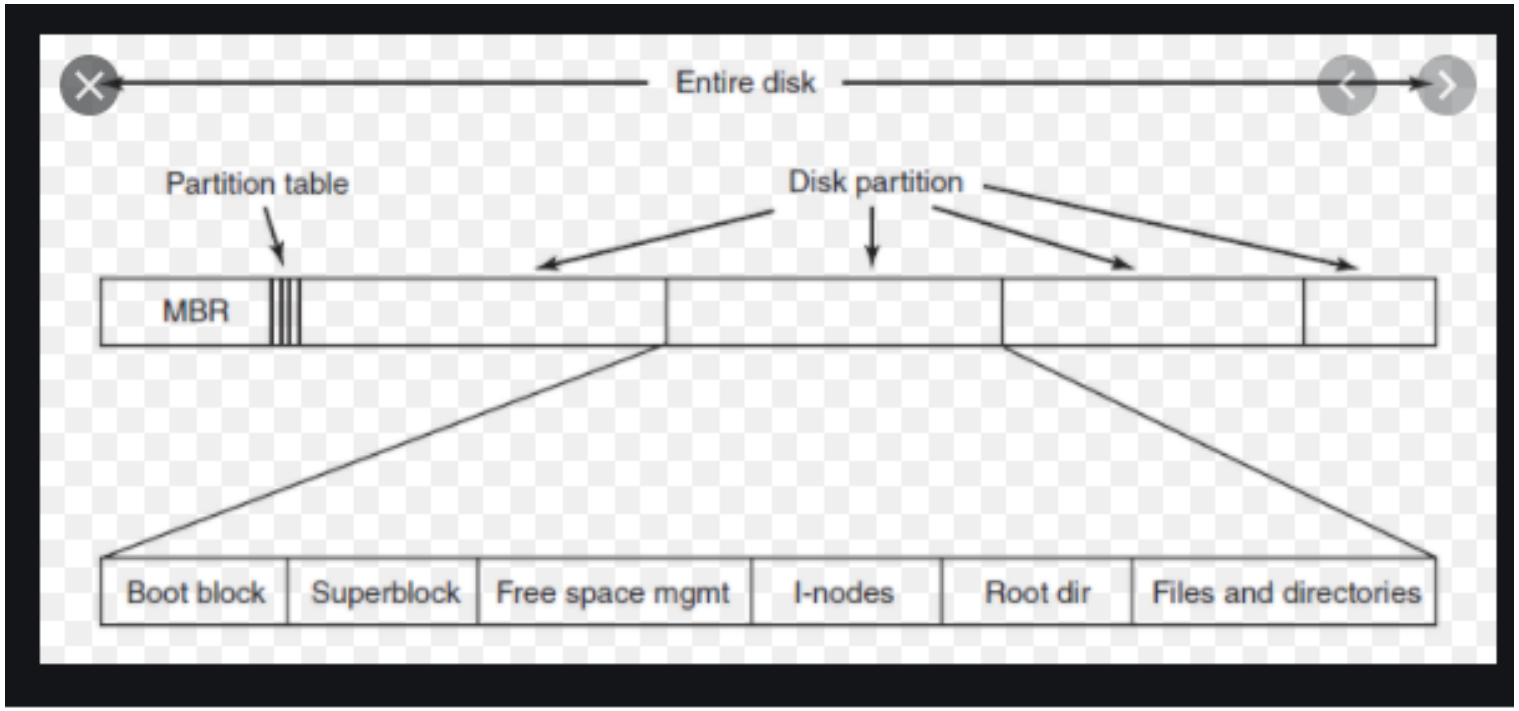


FIGURE 3.4 *Relative Pathnames for* **file3**

File System implementation



Boot Block	Super Block	Inode Block	Data Block

File Systems

Boot Block :- Boot program is used to load the kernel into memory, is found at the beginning of a disk in the boot block.

Super Block :- The Super block, contains information about the file system, such as total size of the disk, how many blocks are empty, and the location of bad blocks on the disks.

Inode Block :- The inode(information node) block, which contains information about each file in the data block. There is one inode for each file on the disk, they contain the information about the file like owner of file, its file type, permissions and address.

Data Blocks :- The data block contains several types of files. It contains all of the user files, it is where data is stored.

Inode Block

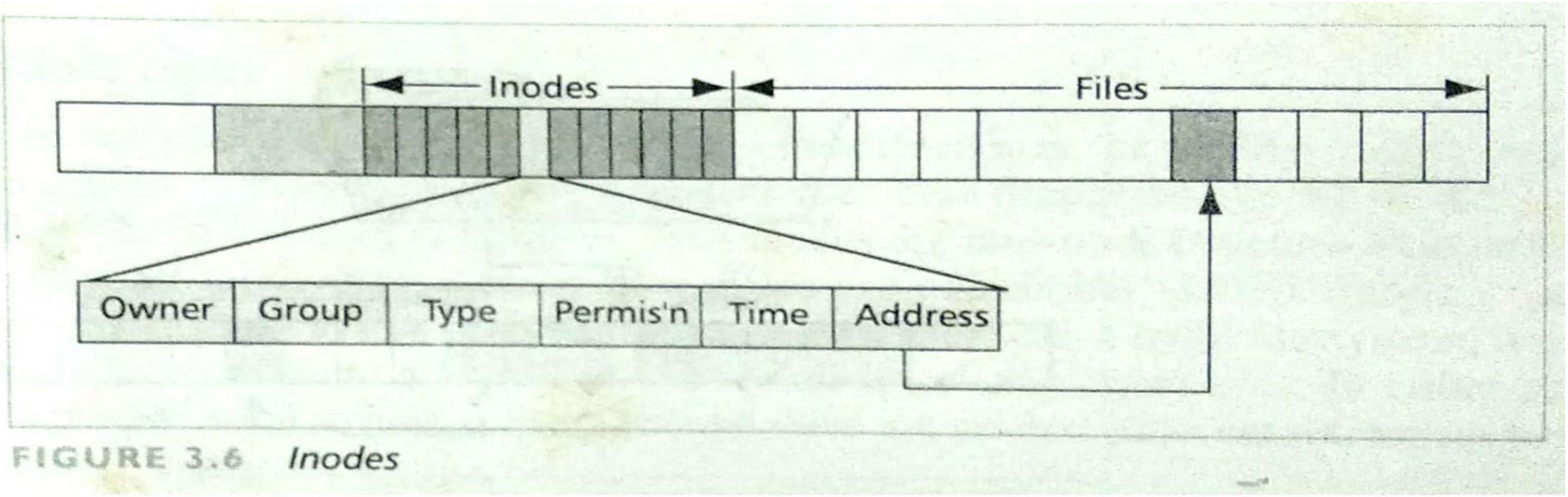
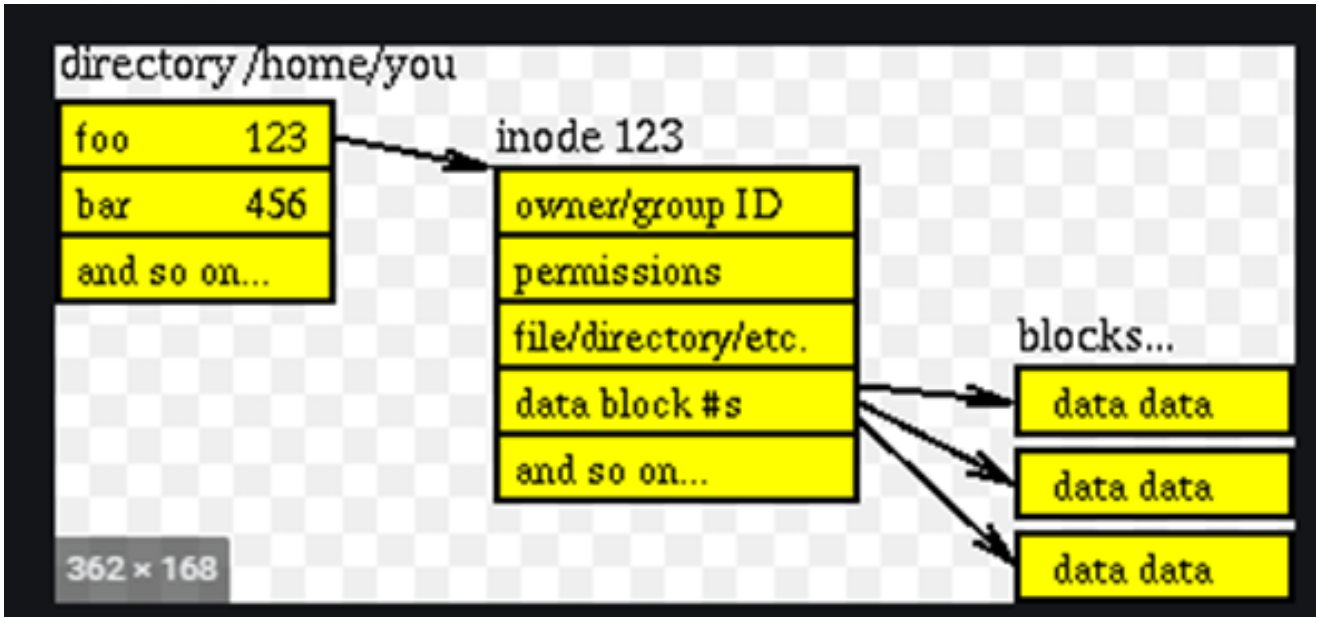
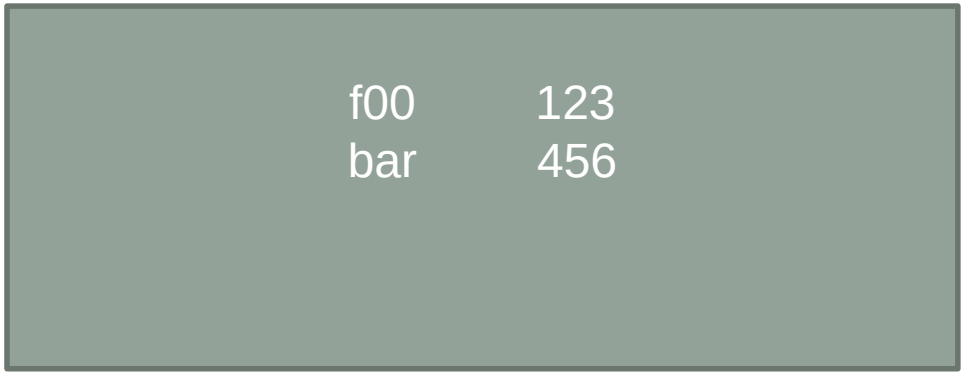


FIGURE 3.6 *Inodes*



Directory Contents

- Name of the directory: joan
- Contents of dir can viewed by using command: ls



Links

- Types of links
  - **Hard links** :- The inode in the directory links the filename directly to the physical file.

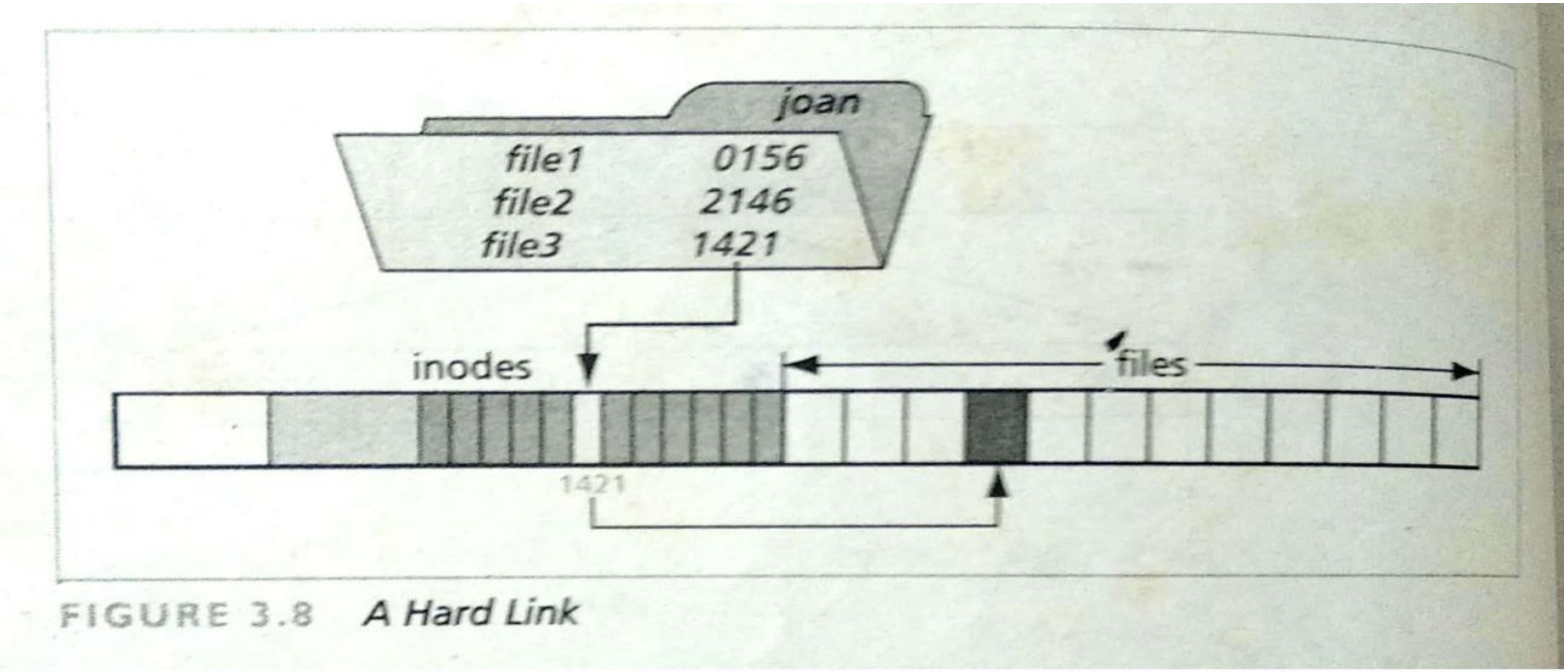


FIGURE 3.8 *A Hard Link*

- **Symbolic links** :- The inode is related to the physical file through a special file known as a symbolic link.

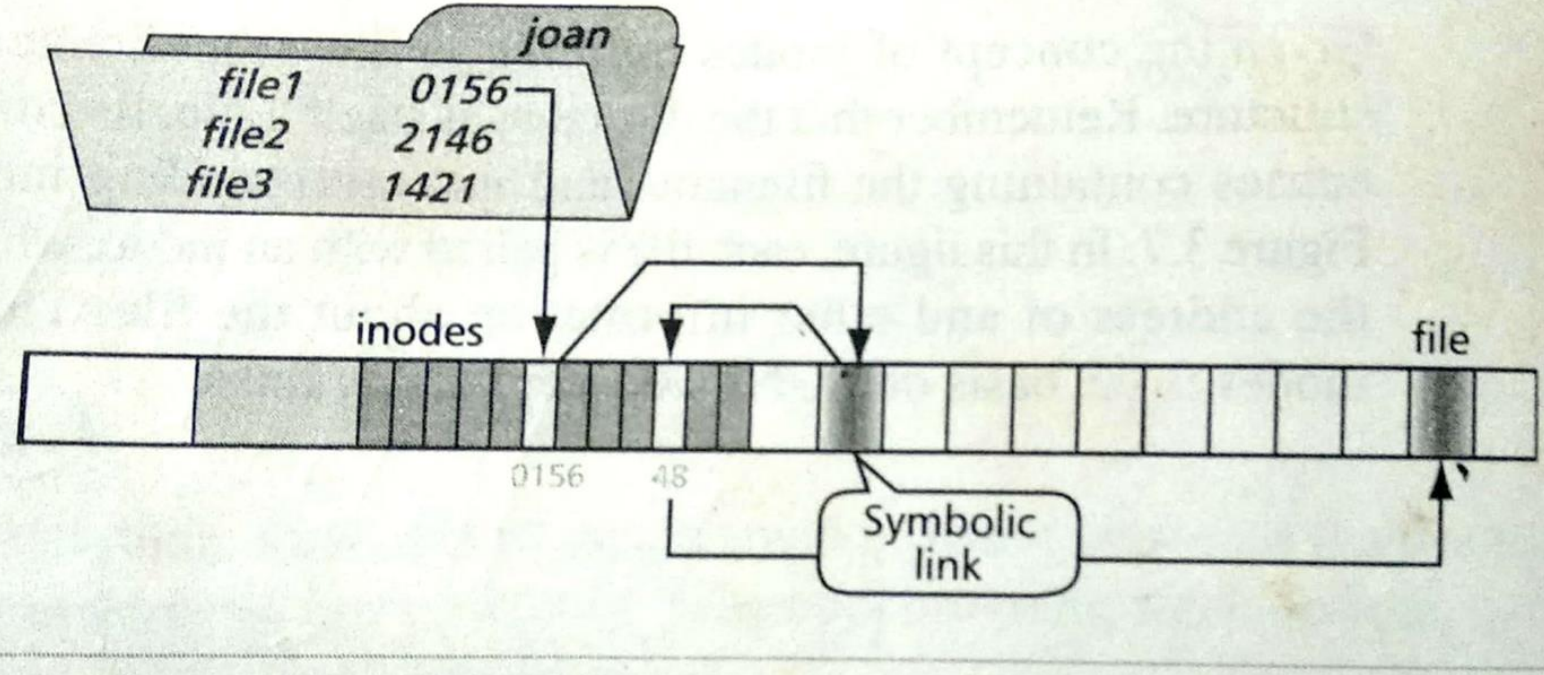
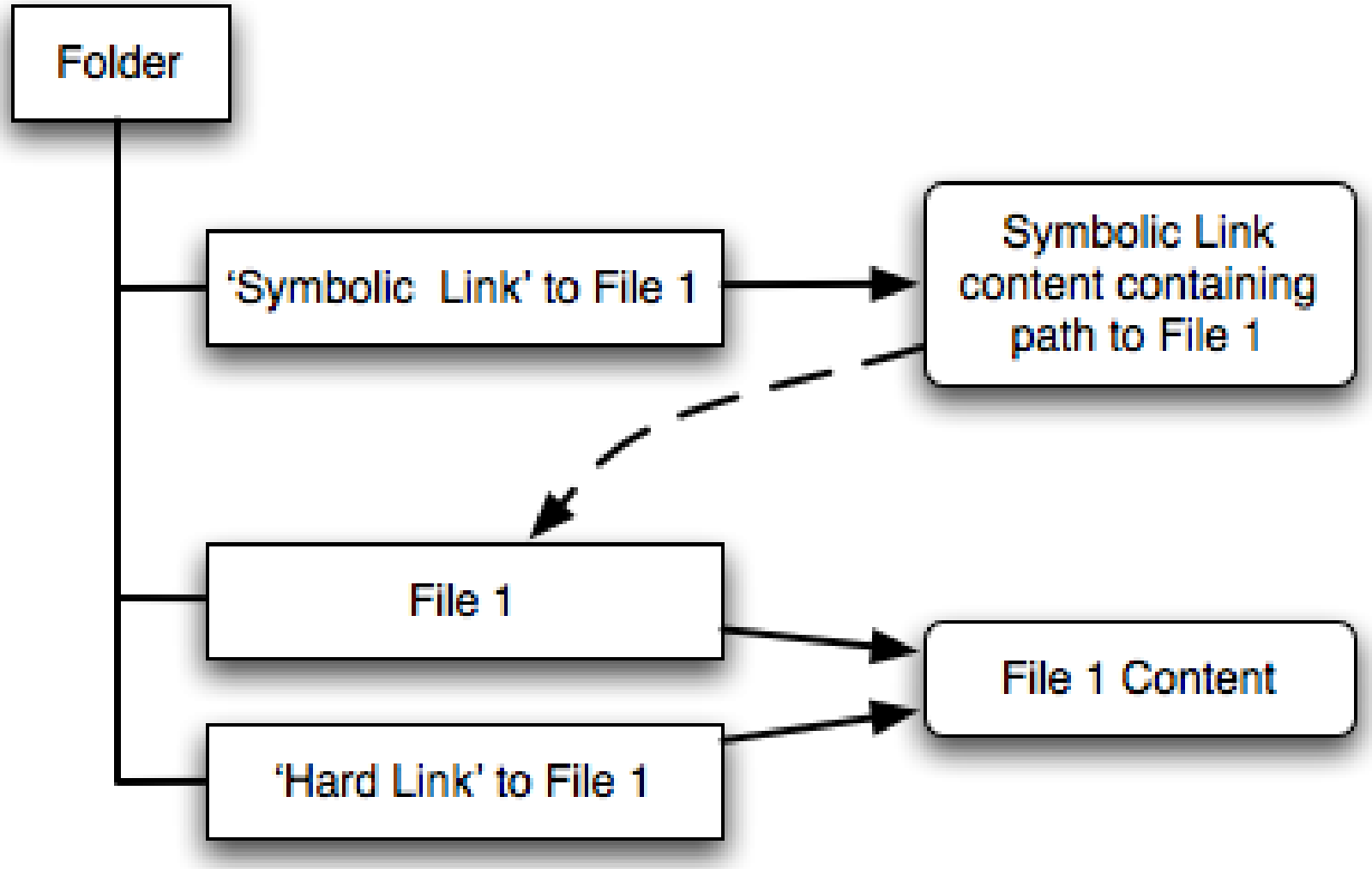


FIGURE 3.9 *A Symbolic Link*


Folder
'Symbolic Link' to File 1
Symbolic Link content containing path to File 1
File 1
File 1 Content
'Hard Link' to File 1

- **Multiple links** :- The inode design is the ability to link two or more different filenames to one physical file. This makes the multilink structure a convenient and efficient method of sharing files.

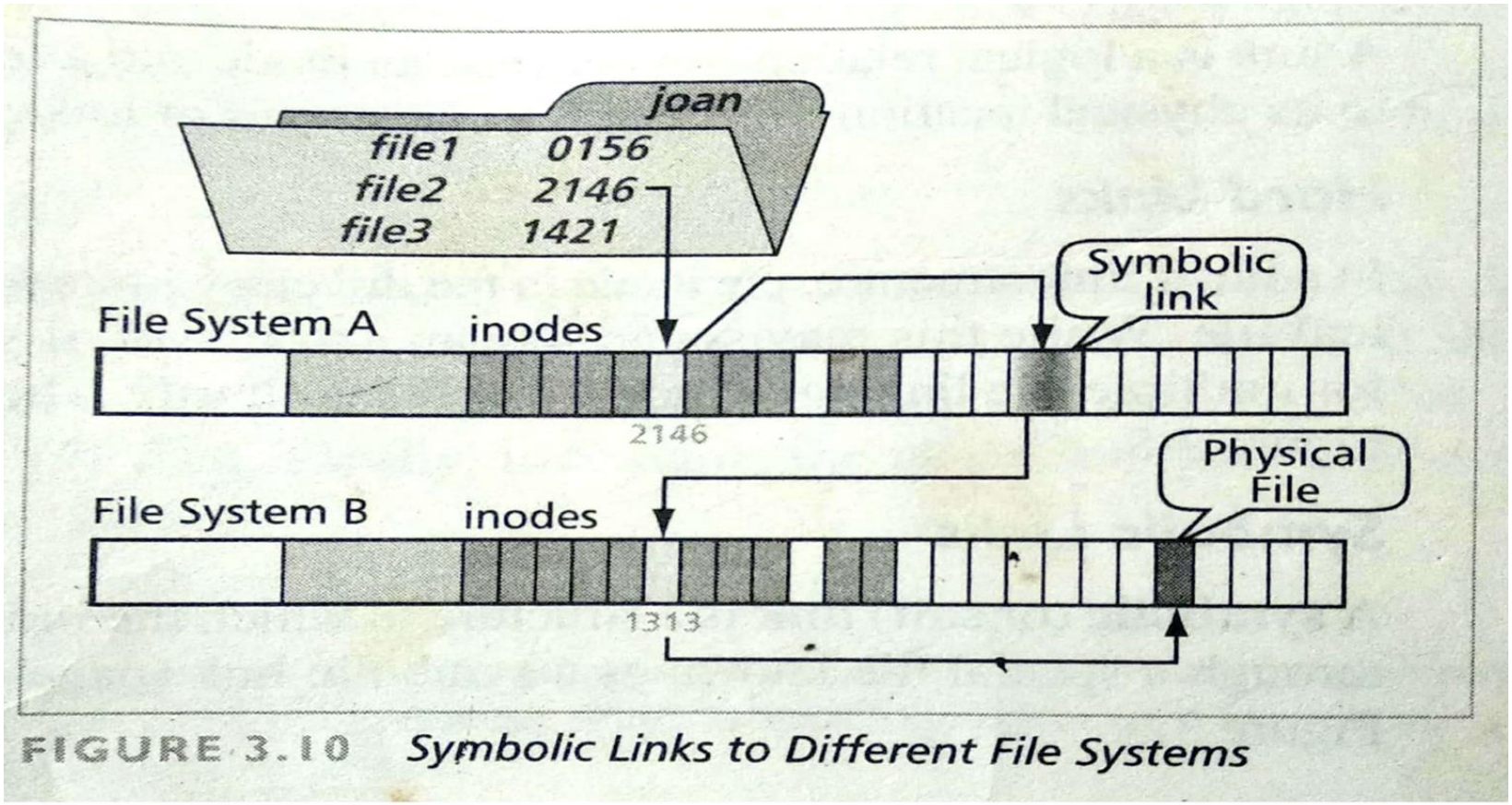


FIGURE 3.10 *Symbolic Links to Different File Systems*

Operations Unique to Directories

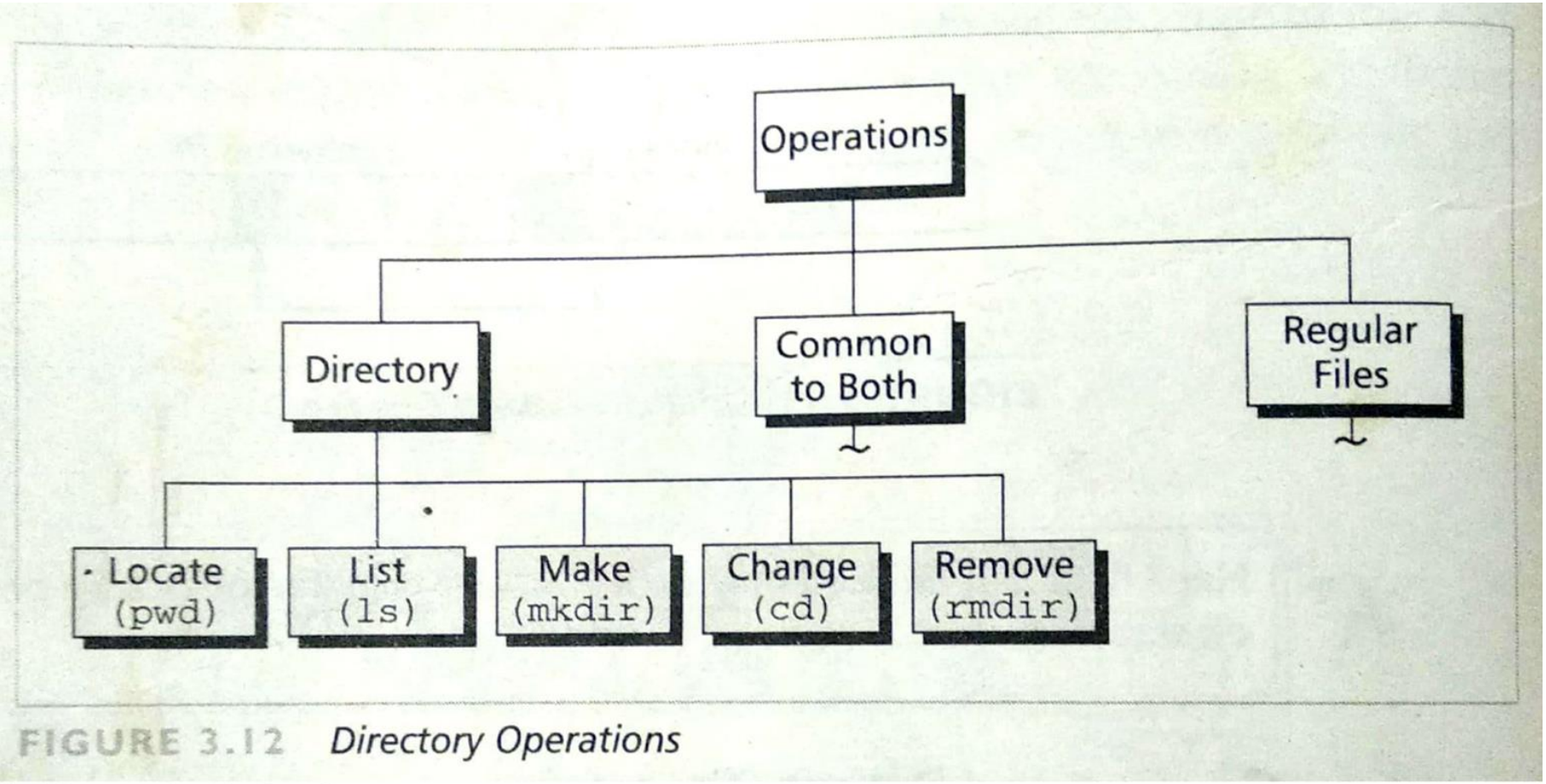


FIGURE 3.12 *Directory Operations*

Locate Directory (pwd) Command

- The command used to determine the current directory is **print working directory** (pwd). It has no options and no attributes. It prints the absolute pathname for the current directory.

- $ pwd
- /usr/~gilberg/tran

- **List Directory (ls) Command**
  - The list (**ls**) command lists the contents in a directory. It can list files, directories, or sub-directories. If the name of a directory is not provided, it displays the contents of the working directory.

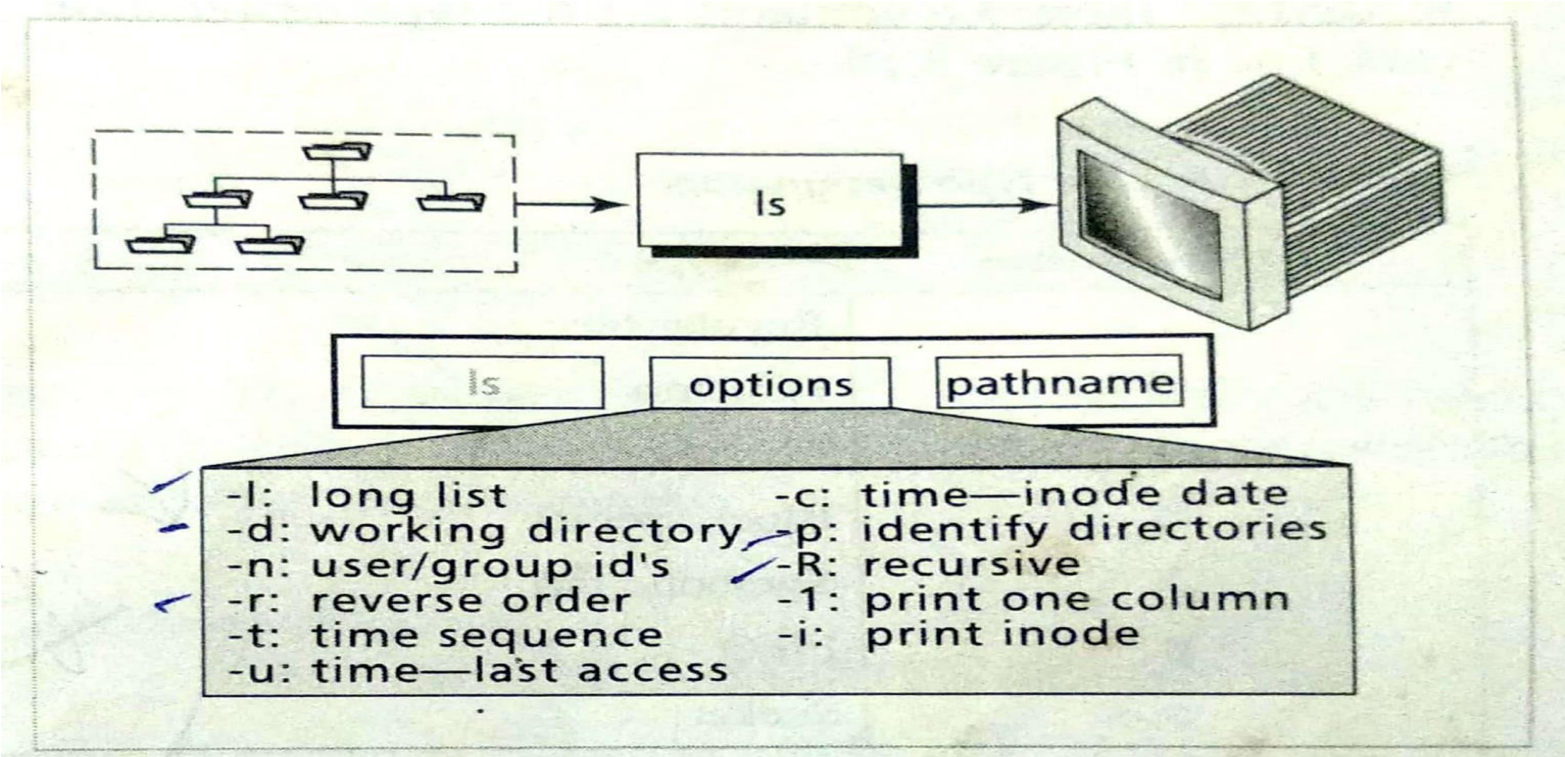


FIGURE 3.14 *The* **ls** *Command*

Long List

- The simple list command is good for a quick review of the files in a directory, but much more information is available with the long list.

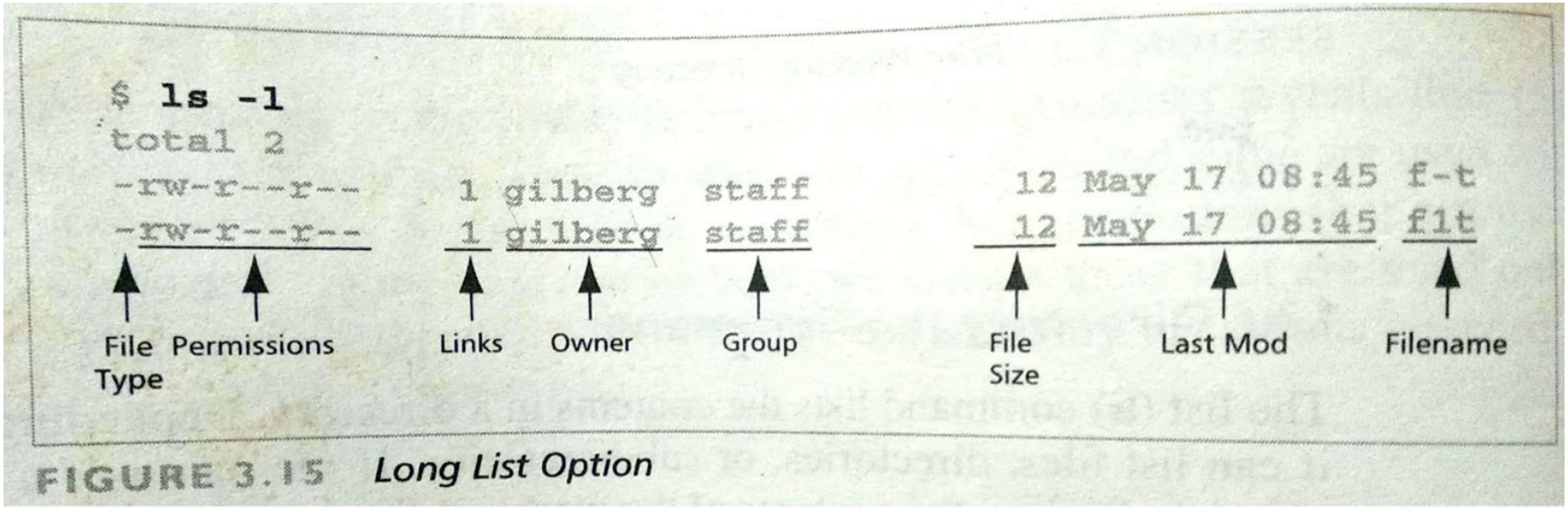


FIGURE 3.15 *Long List Option*

File type	File Designation
• Regular file	-
• Directory	d
• Character Special	c
• Block Special	b
• Symbolic link	l
• FIFO	p
• Socket	s

Recursive List

- In a list structure command, recursive means to list the directory, and then list another directory, and then another one. The recursive directory list is used primarily to study the structure of a directory.

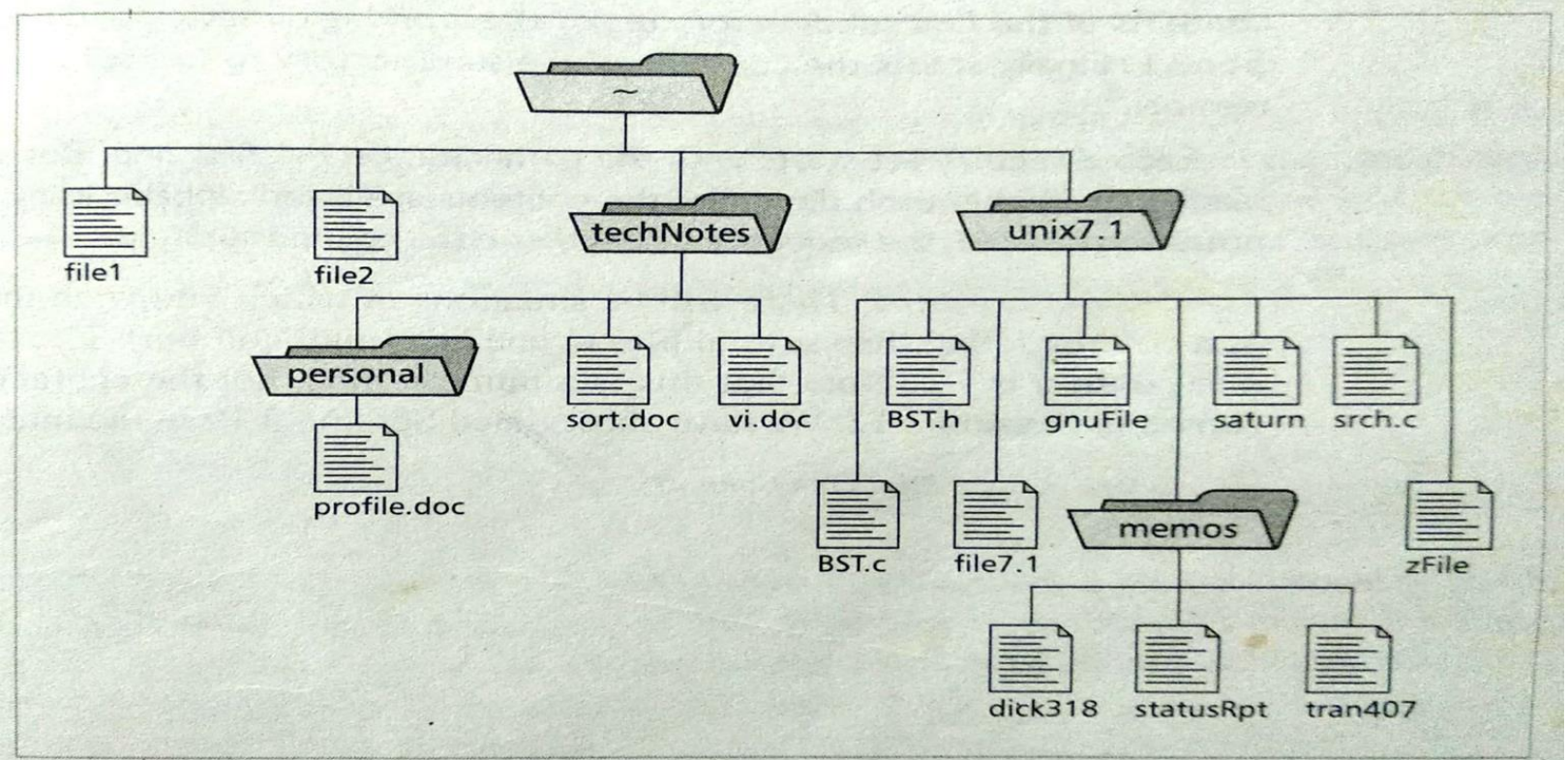


FIGURE 3.16 *Directory Structure for Recursive List*

Make Directory (mkdir) Command

- To create a new directory, use the make directory (mkdir) command.
- It has two options **permission mode** and **parent mode**.

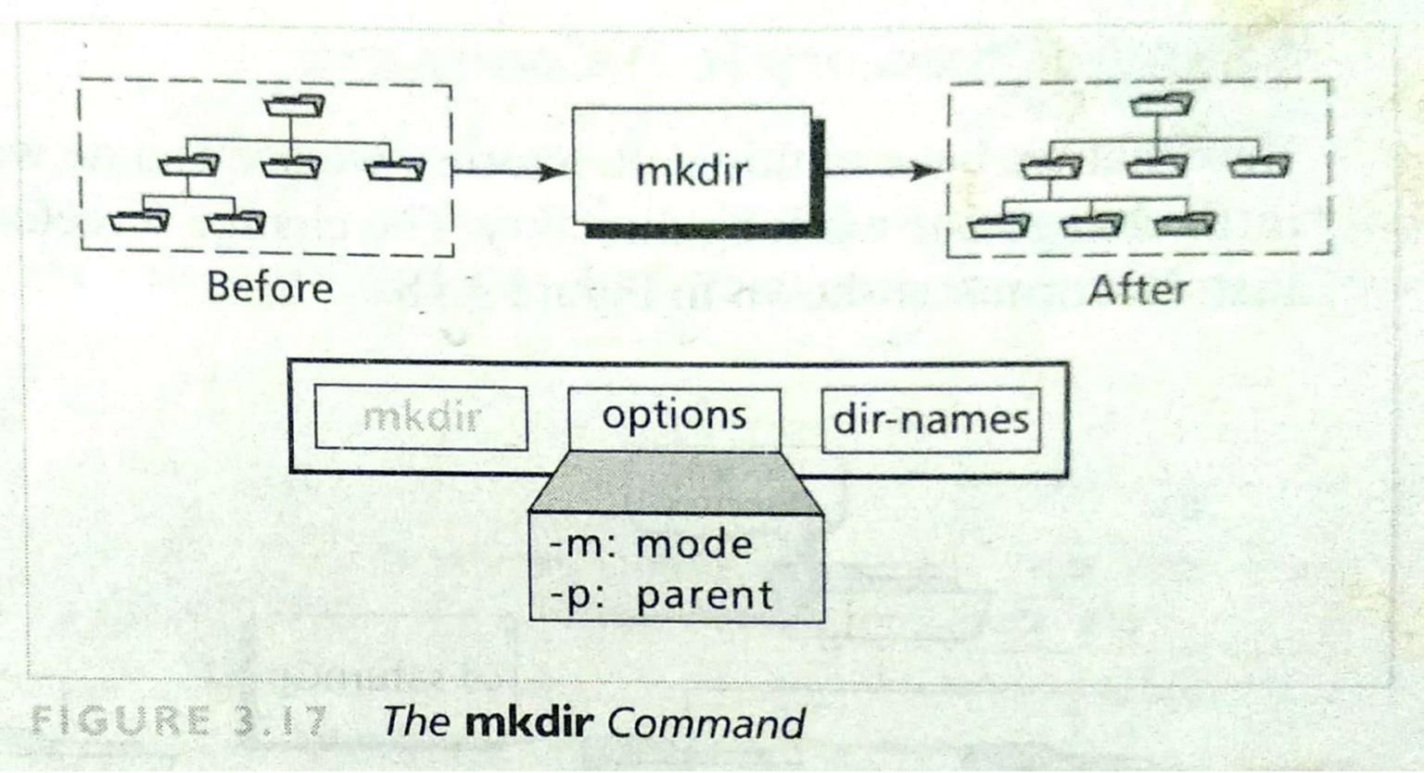


FIGURE 3.17 *The* **mkdir** *Command*

Change Directory (cd) Command

- The change directory (cd) command is used to change our working directory.
- There are no options for the change directory command.

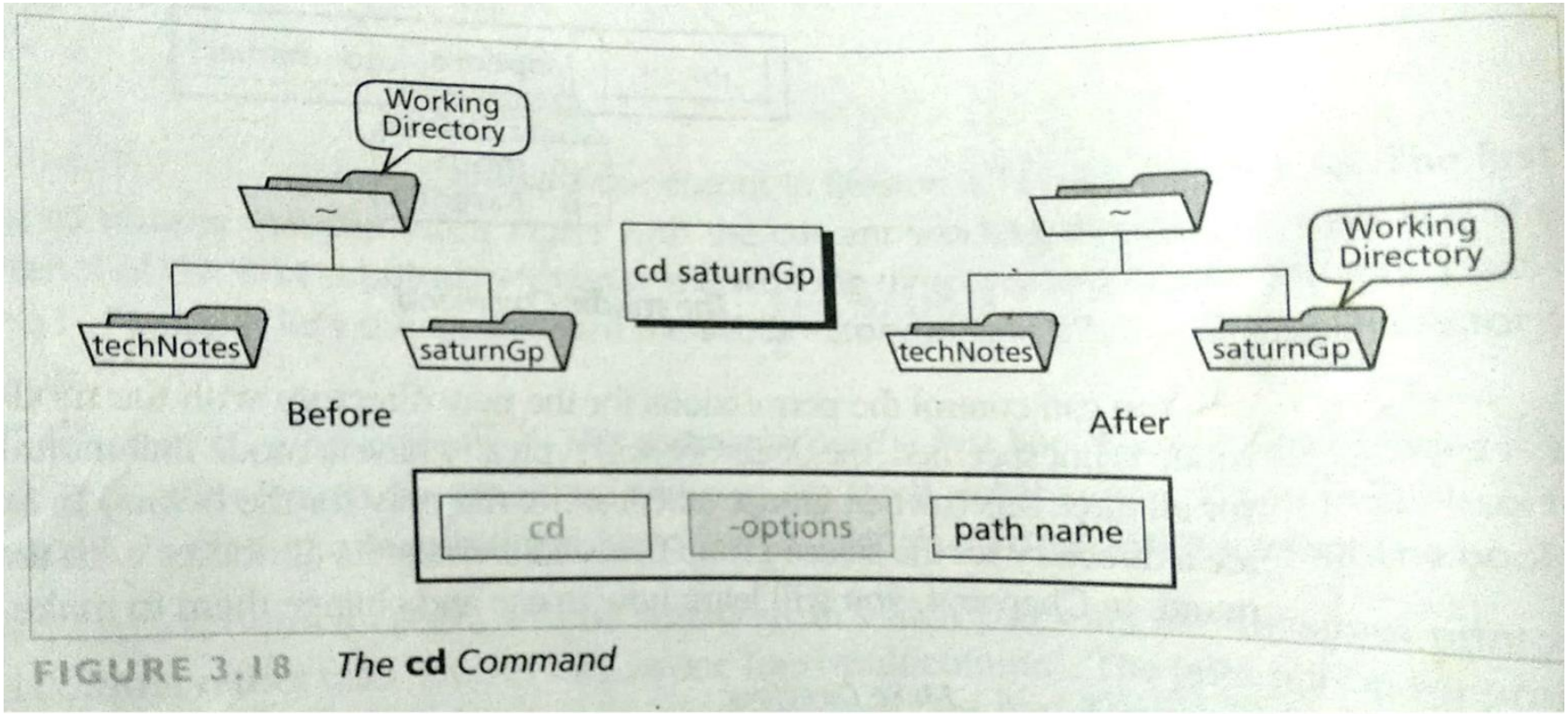


FIGURE 3.18 *The* **cd** *Command*

Remove Directory (rmdir) Command

- When a directory is no longer needed ,it should be removed. The remove directory (rmdir) command deletes directories.

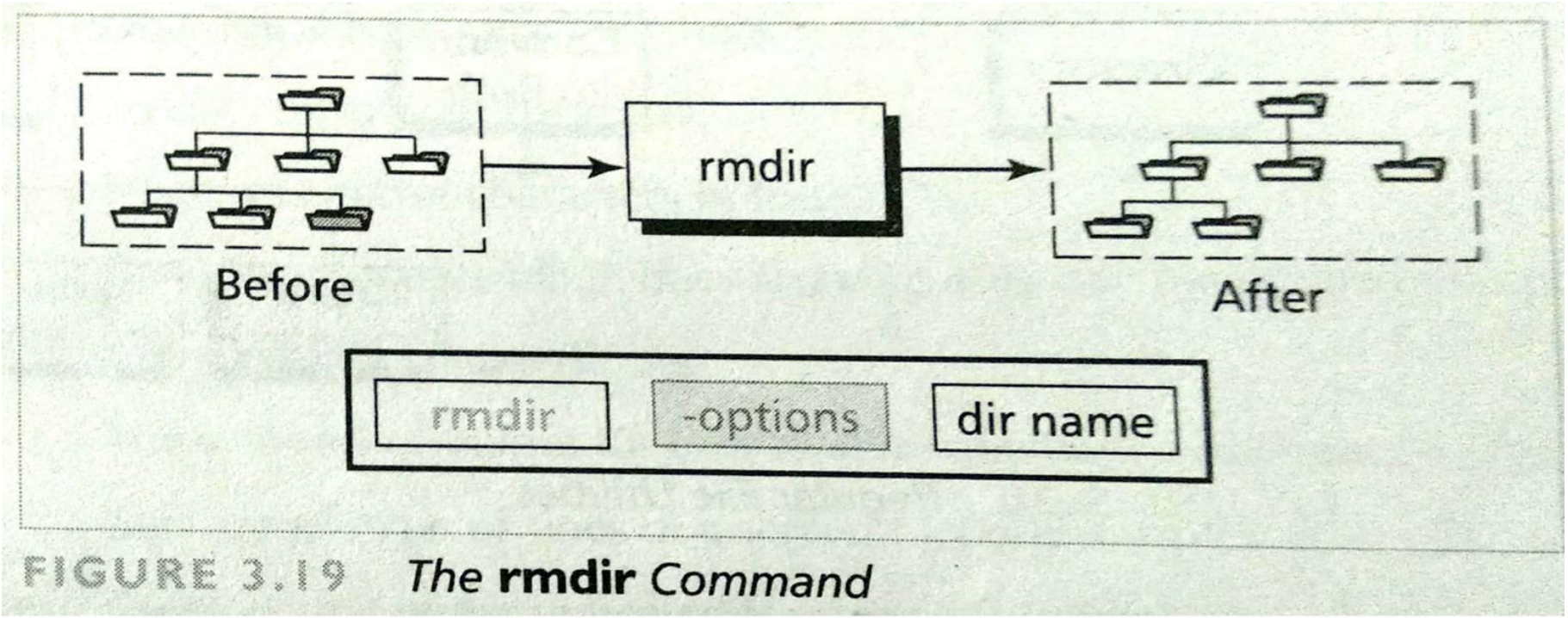


FIGURE 3.19 *The* **rmdir** *Command*

Operations Unique to Regular Files

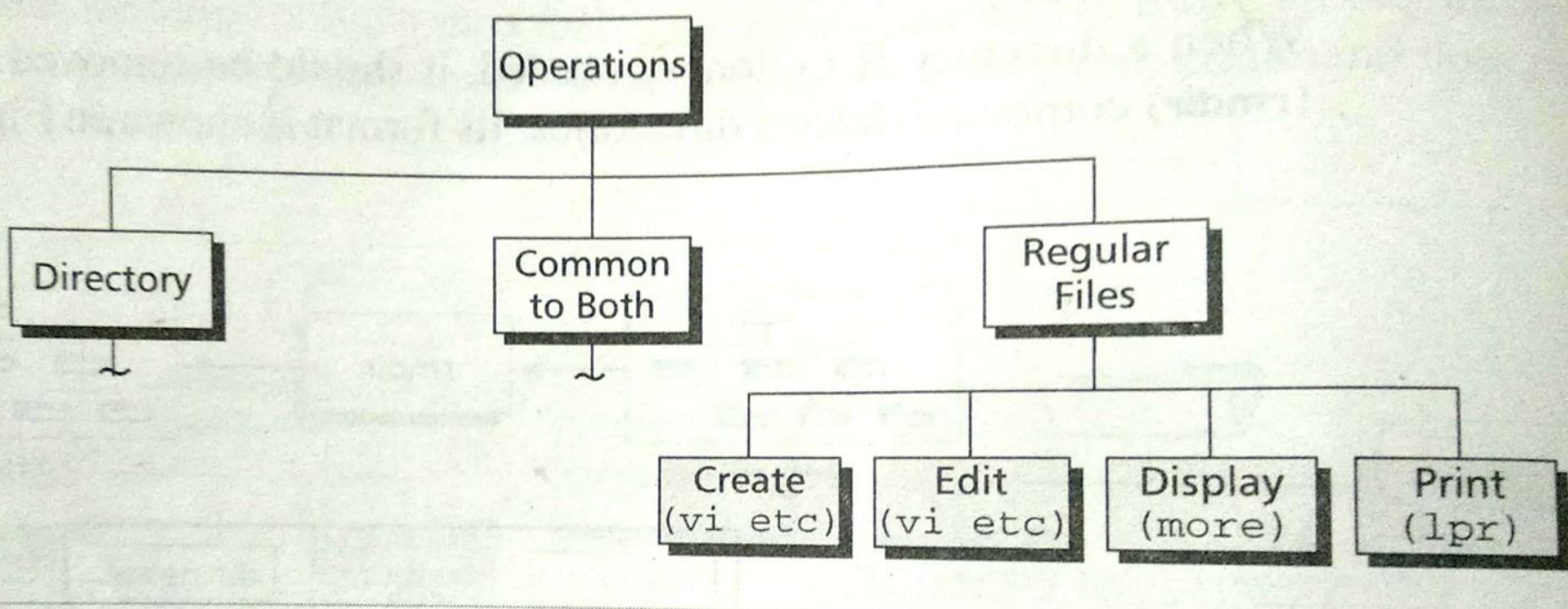


FIGURE 3.20 *Regular File Utilities*

- **Create File**
  - The most common tool to create a text file is a text editor such as **vi.**
  - The cat command is useful to create other files.
- **Edit File**
  - The most common tool to edit a text file is a text editor such as **vi.**
  - The sed is a powerful search and edit tool..

- **Display File (more) Command**
  - The most useful one to display a file is **more**. It allows to set the output page size and pauses at the end of each page to allow us to read the file.
  - After each page, we may request one or more lines, a new page, or quit.
  - To display the next screen, key the spacebar.

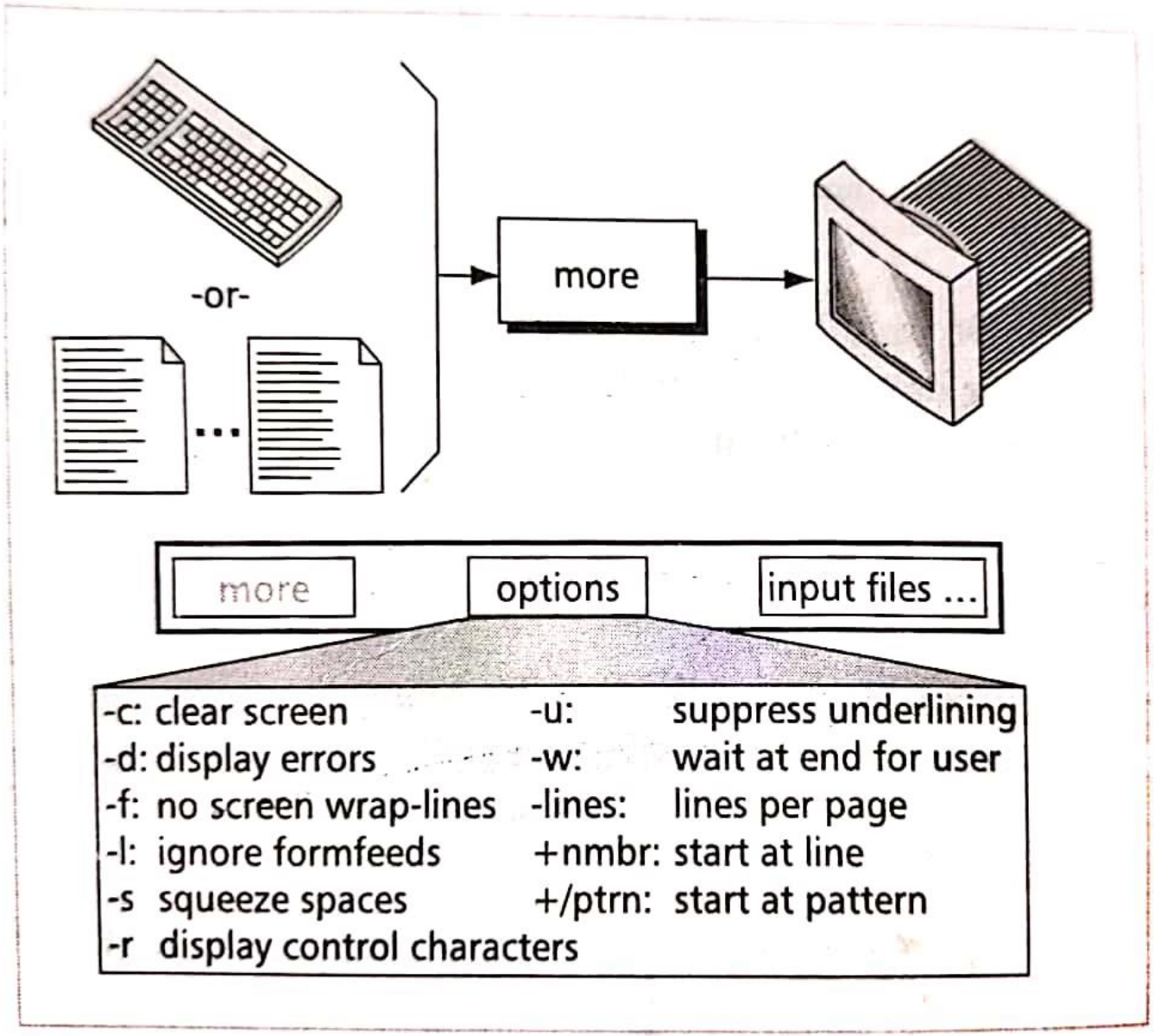


FIGURE 3.21 *The* **more** *Command*

Operations Common to Both

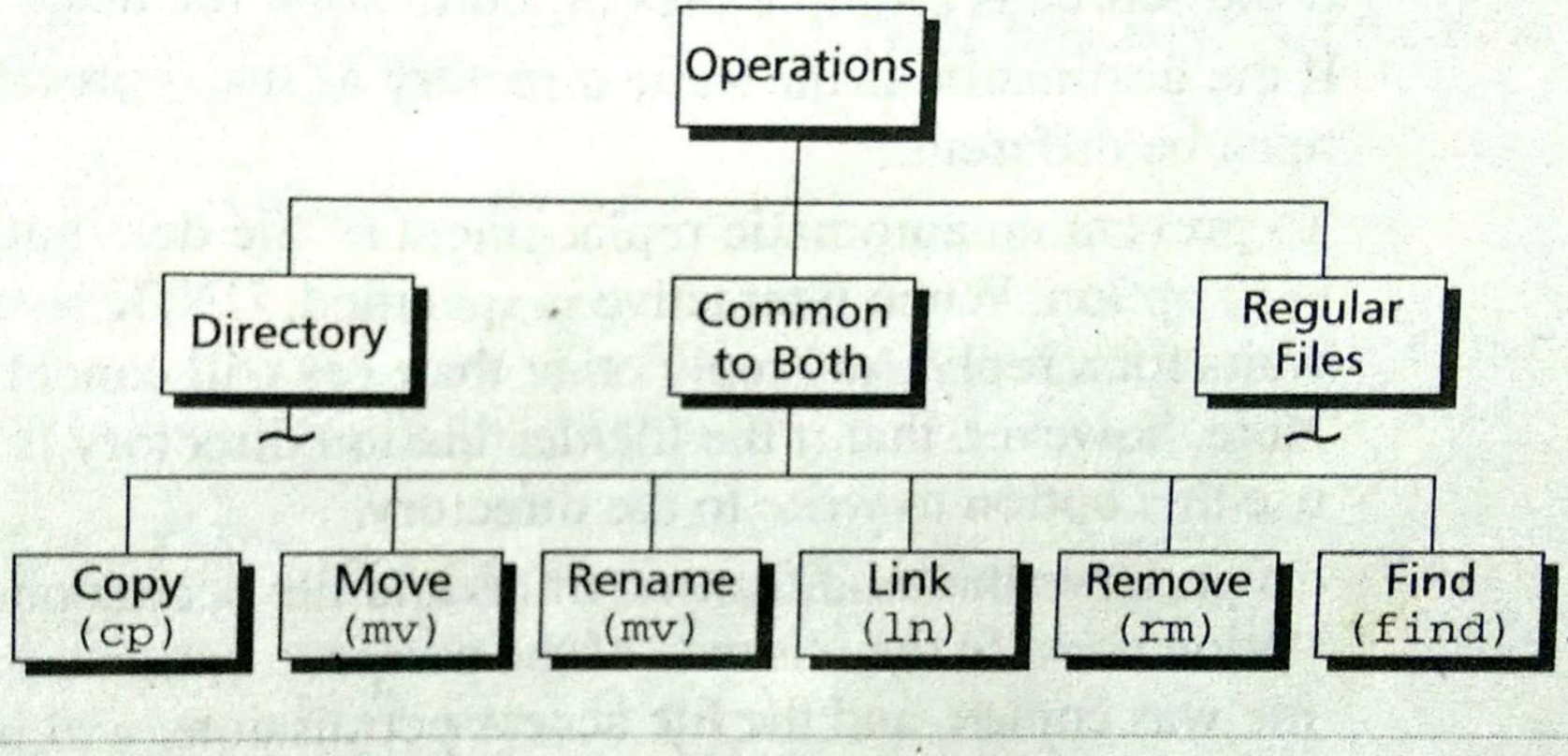


FIGURE 3.22 *Operations Common to Directories and Files*

- **Copy (cp) Command**
  - The copy (cp) utility creates a duplicate of a file, a set of files, or a directory. The cp command copies both text and binary files.

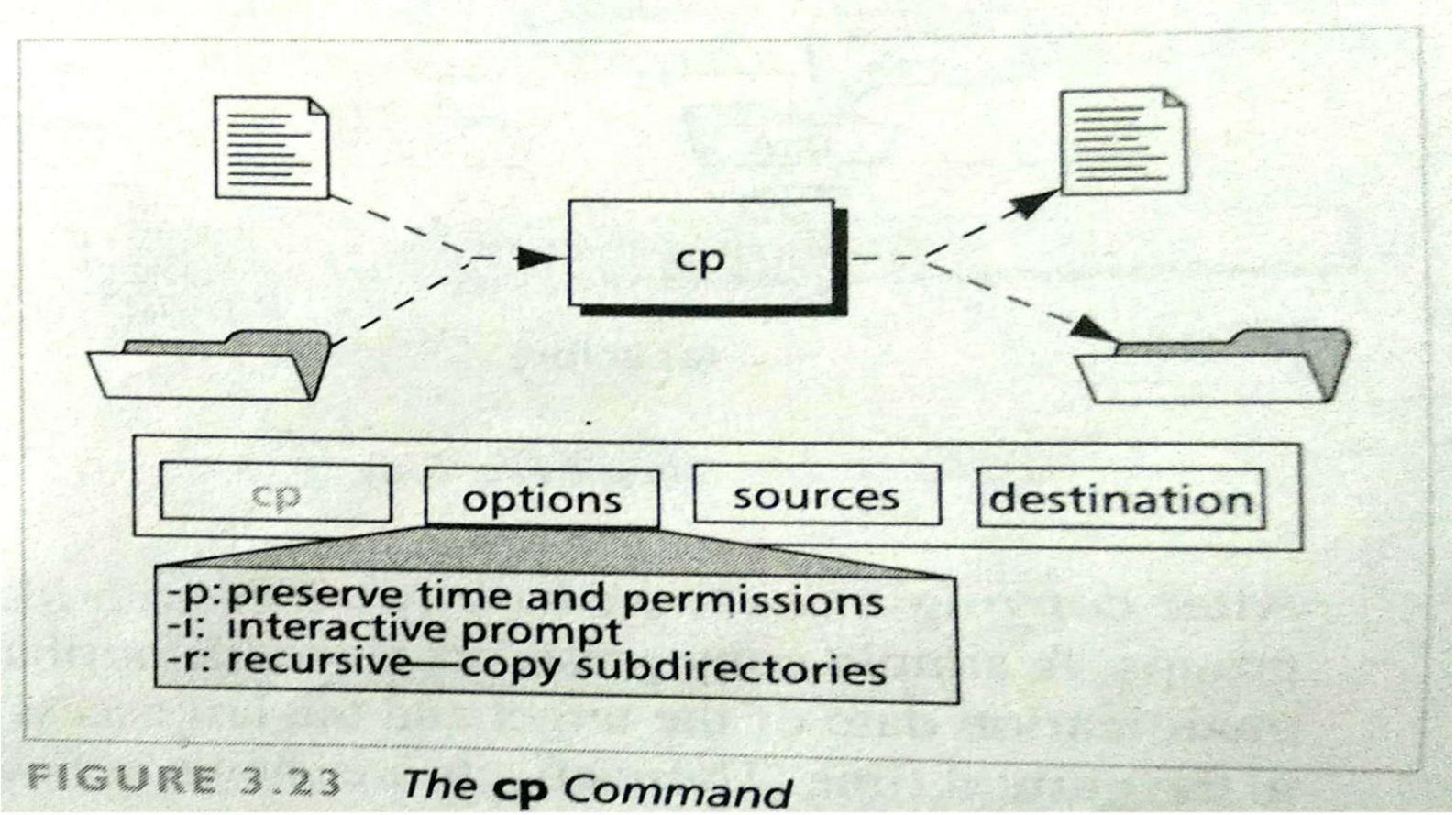


FIGURE 3.23 *The* **cp** *Command*

- **Rules to be followed to copy a file or directory**
  - The source must exist. Otherwise, UNIX prints the following error message. <source> - No such file or directory.
  - If no destination path is specified, UNIX assumes the destination is the current directory.
  - If the destination file does not exist, it is created, if it exists, it is replaced.
  - If the source is a multiple file or a directory, the destination should be directory.
  - If the destination is the same directory as the source, the destination filename must be different.
  - To prevent an automatic replacement of the destination file, use the interactive option (-i).
  - To preserve the modification times and file access permissions, use the preserve option (-p).

- Using the cp command

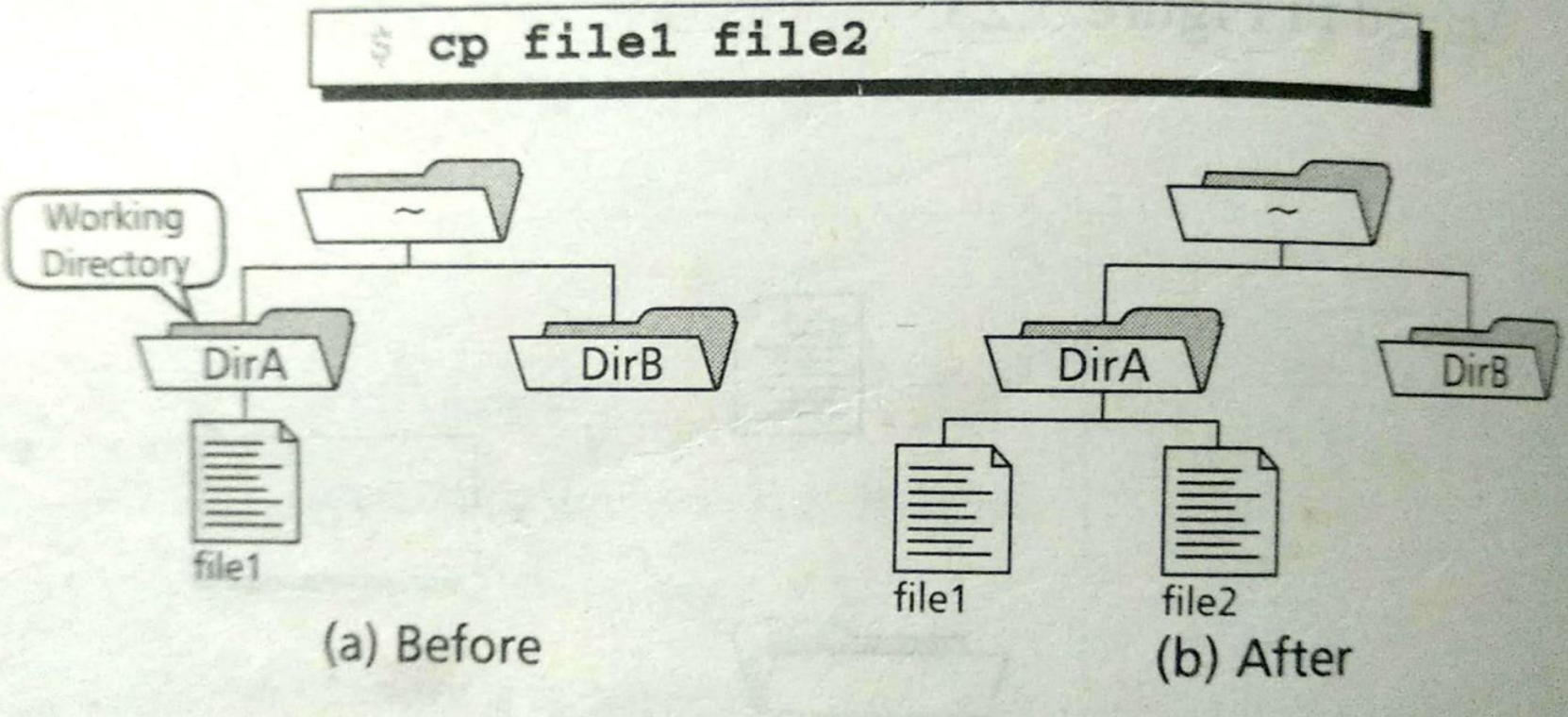


FIGURE 3.24 *Simple File Copy*

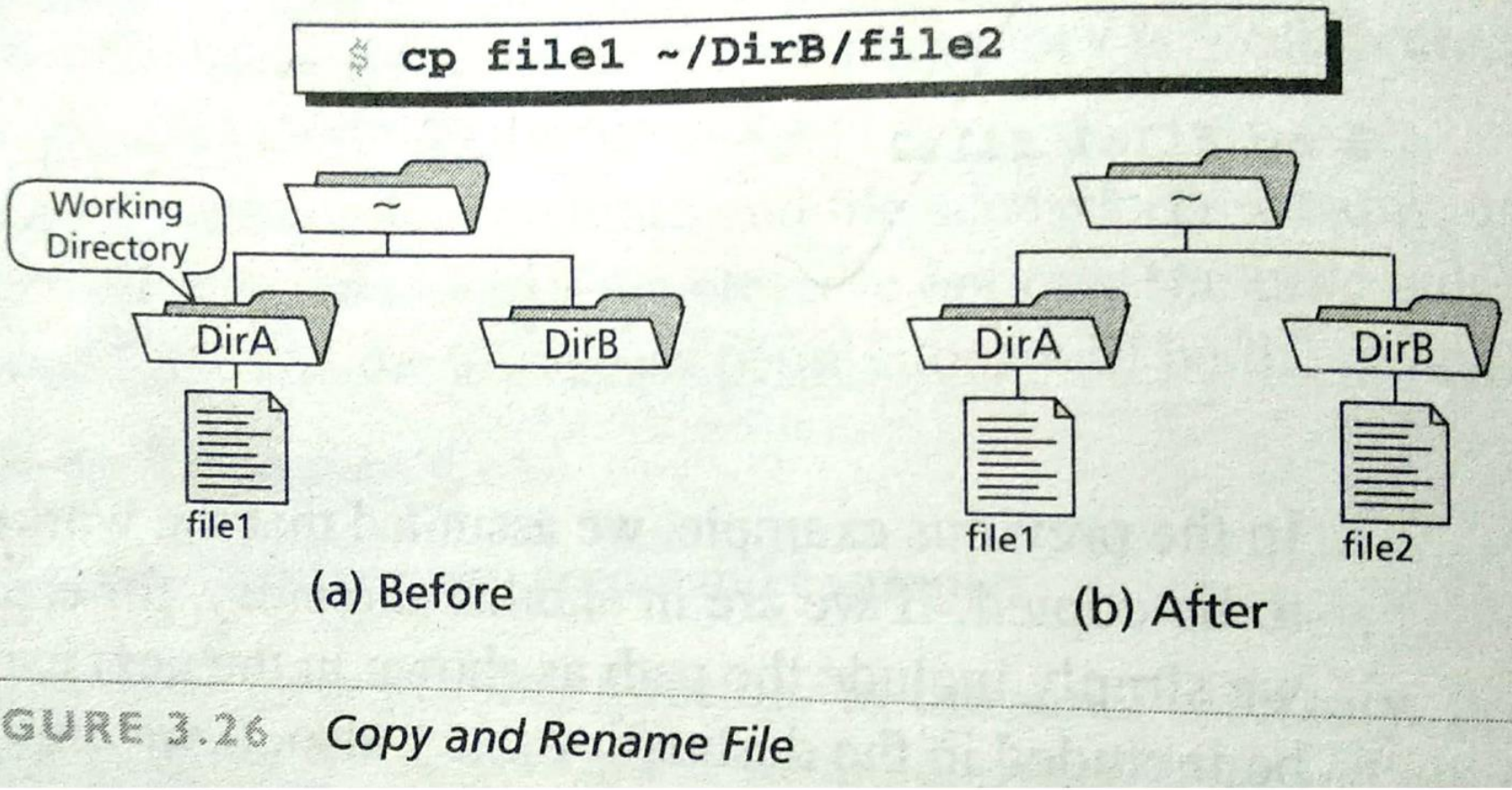


FIGURE 3.26 *Copy and Rename File*

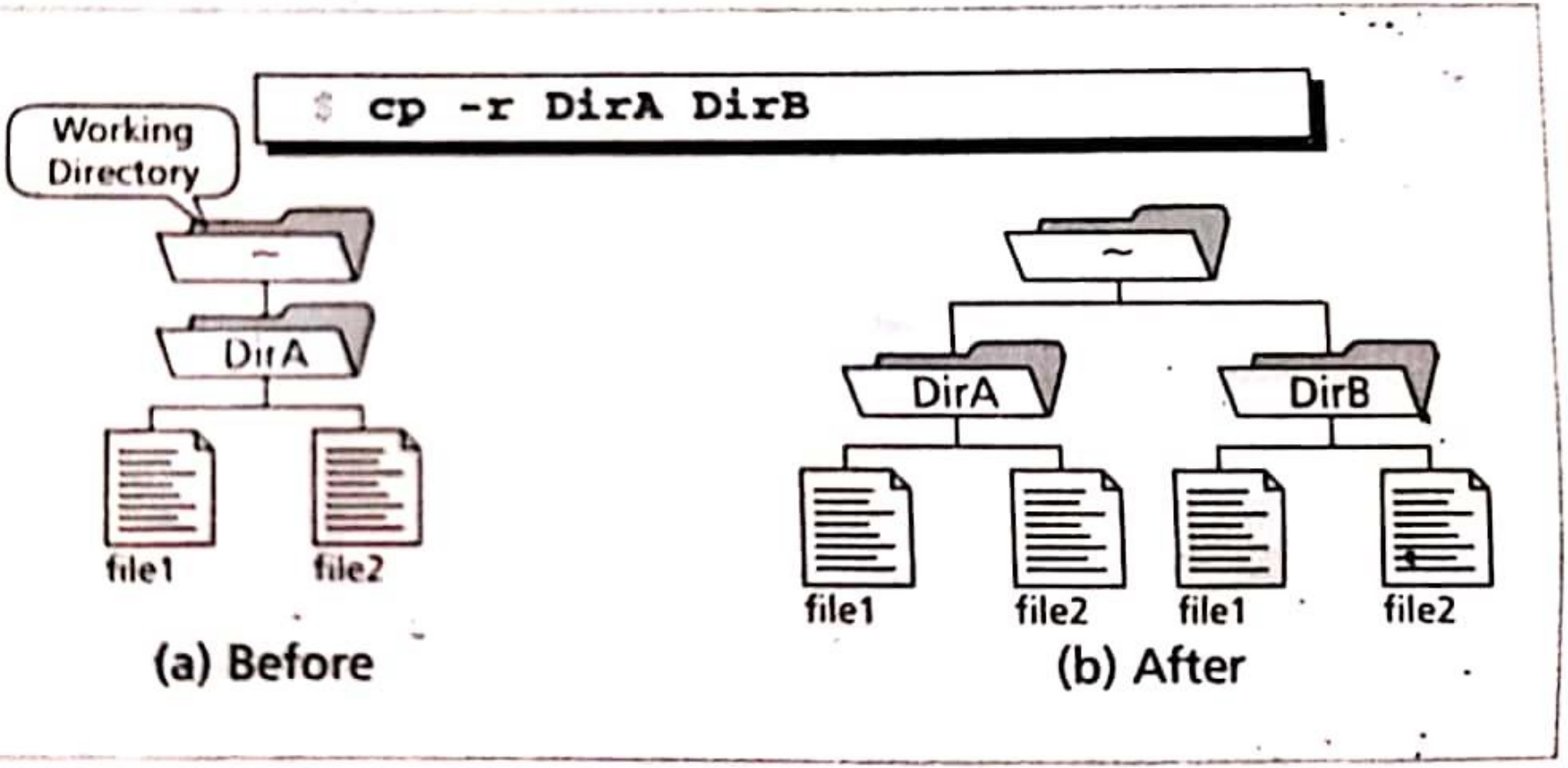


FIGURE 3.27 *Recursive Copy*

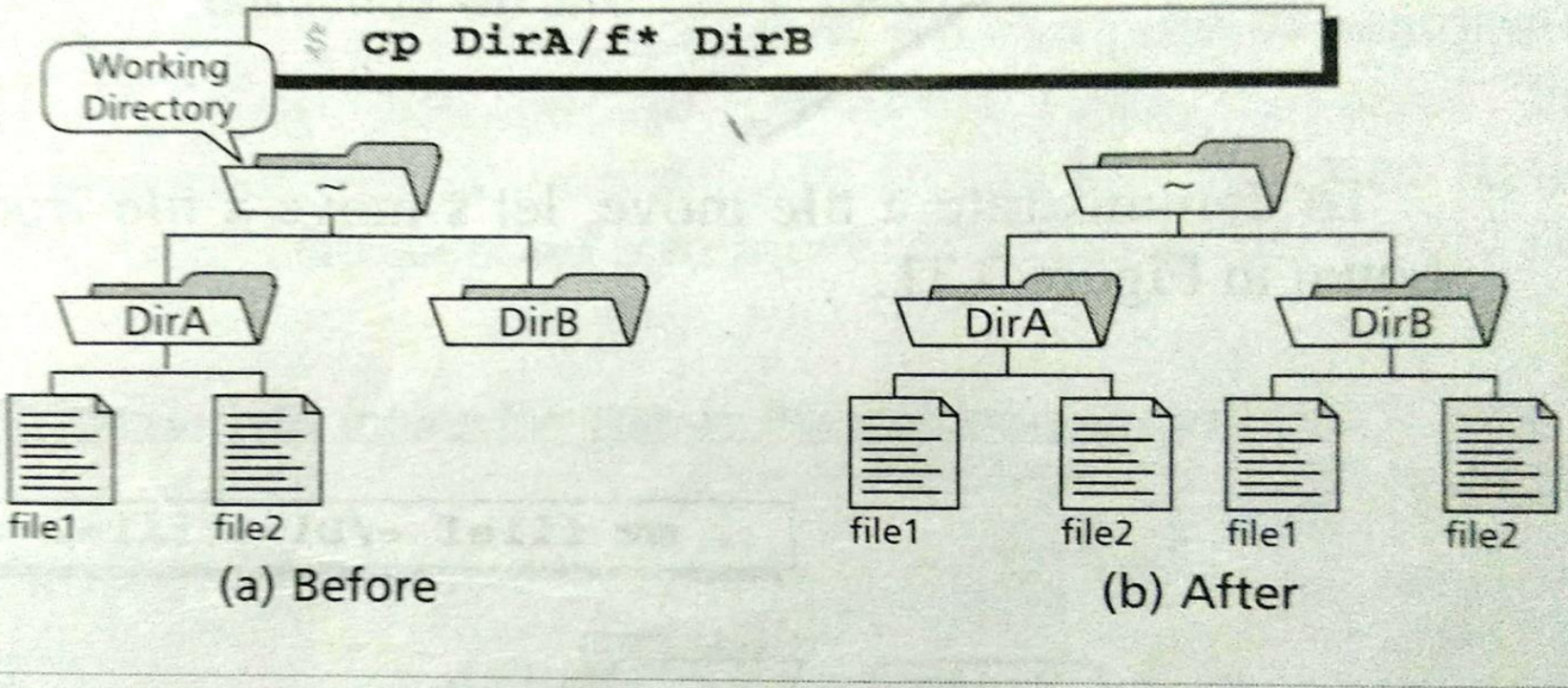


FIGURE 3.29 *Wildcard Copy*

- **Move (mv) Command**
  - The move (mv) command is used to move either an individual file, a list of files, or a directory. After a move, the old file name is gone and the new file name is found at the destination.

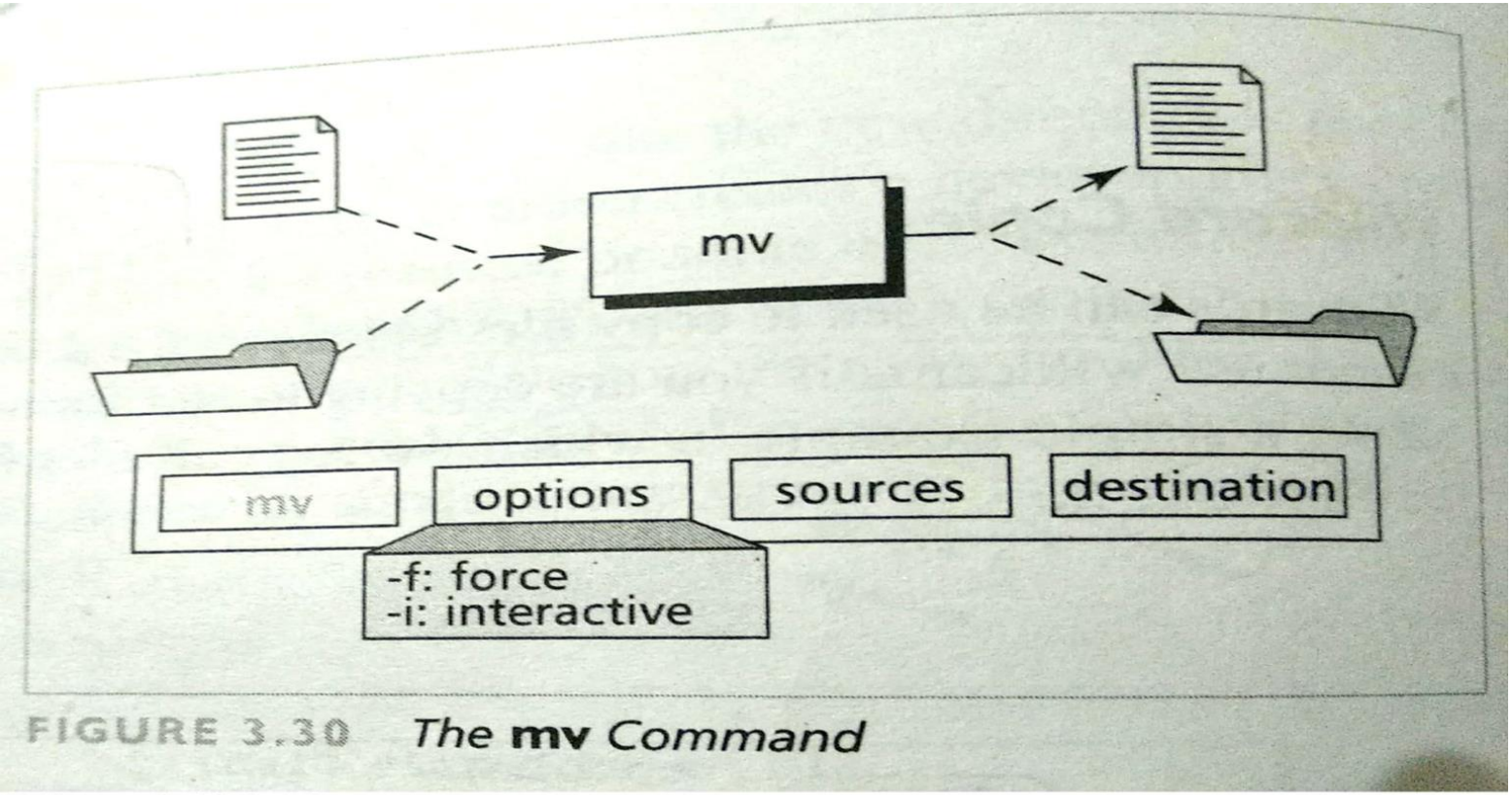


FIGURE 3.30 *The* **mv** *Command*

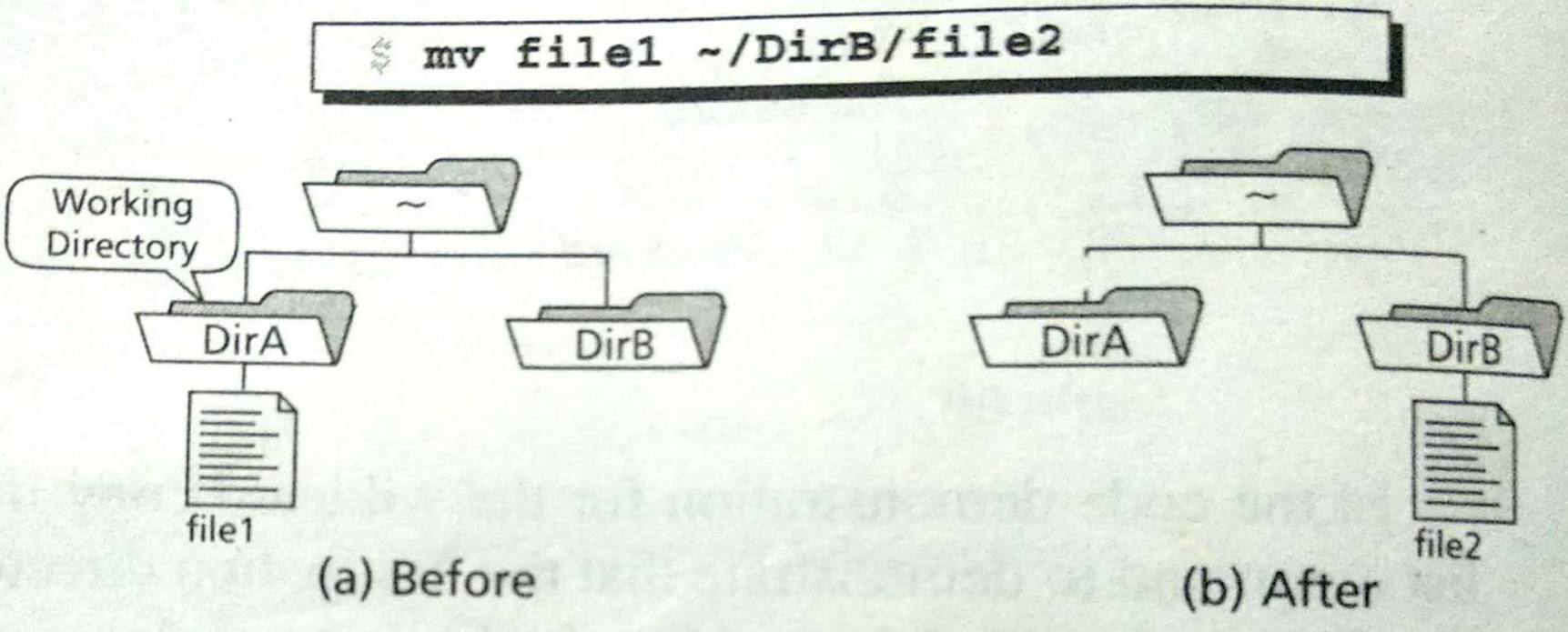


FIGURE 3.31 *Move File*

Move as only two options : interactive (-i) and force (-f)

- **Interactive** :- If the destination file already exists, its contents are destroyed unless we use the interactive flag (-i) to request that move warn us.
- **Force** :- If we are sure that we want to write file, even if it already exists, we can skip the interactive message with the force (-f) option.

- **Rename (mv) Command**
  - UNIX does not have a specific rename command.
  - If the destination file is in the same directory as the source file, the effect is renaming of the file.

  - Example :- mv file1 file0.

- **Link (ln) Command**
  - The link command receives either a file or directory as input, and its output is an updated directory.

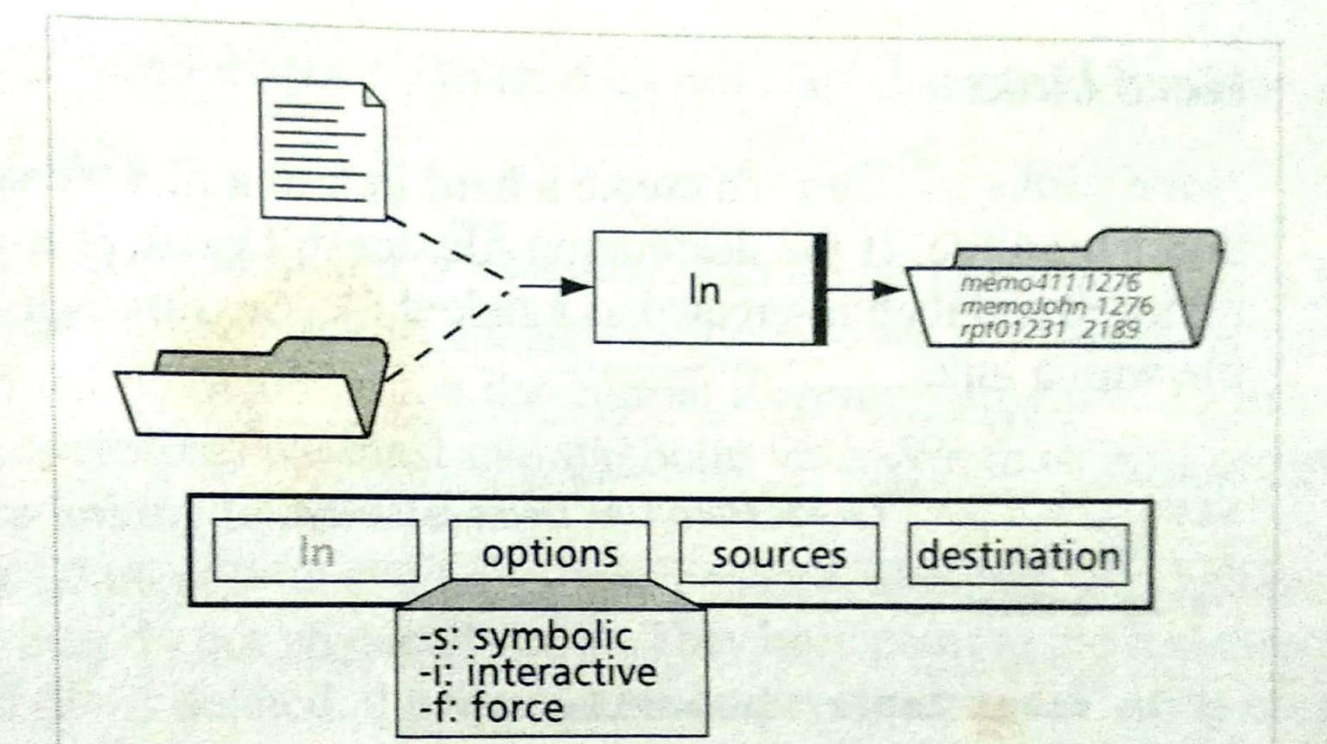


FIGURE 3.32 *The* **ln** *Command*

Hard links to Files

- To create a hard link to a file, we specify the source file and the destination file.
- If the destination file doesn't exist, it is created. If it exists, it is first removed and then re-created as a linked file.

```
$ ls lnDir
$
$ ln file1 lnDir/linkedFile
$

$ ls –i file1
79914  file1

$ ls –i lnDir/linkedFile
79914  lnDir/linkedFile
```

- **Symbolic Links**
  - To link to a different file system, therefore, we must use symbolic links. We must use symbolic links when we are linking to directories.
  - There is a danger with symbolic links because, although they behave like files and directories, they do not physically exists. If the physical file is deleted, the file will no longer appear on listing under its original name. It will still be available under its symbolic link name, but it is not accessible.
  - if we try to move to it, however, we receive a message that it doesn’t exist, to delete it, we must use the delete file command (rm), not the delete command (rmdir).

- Symbolic link to Files

```
$ ls lnDir
$ linkedFile

$ ln –s file2 lnDir

$ ls –i file2
79937  file2

$ ls –il lnDir
total 2
79898 lrwxr-xr-x 1 gilberg  … 5 May 28 15:39 file2 -> file2

79914 –rw-r--r-- 2 gilberg  … 120 May 28 15:39 linkedfile
```

- Symbolic link to Directory

$ ls symDir
Cannot access symDir: No such file or directory

$ ln –s Dirc symDir
$ # We have now created a symbolic directory as verified by inodes

$ ls –il symDir

79914 lrwxr-xr-x 1 gilberg staff 4 Sep 18 21:53 symDir -> DirC

- **Remove (rm) Command**
  - The remove (rm) utility deletes an entry from a directory by destroying its link to the file.
  - The file is deleted only if, after the remove, there are no more links to it.

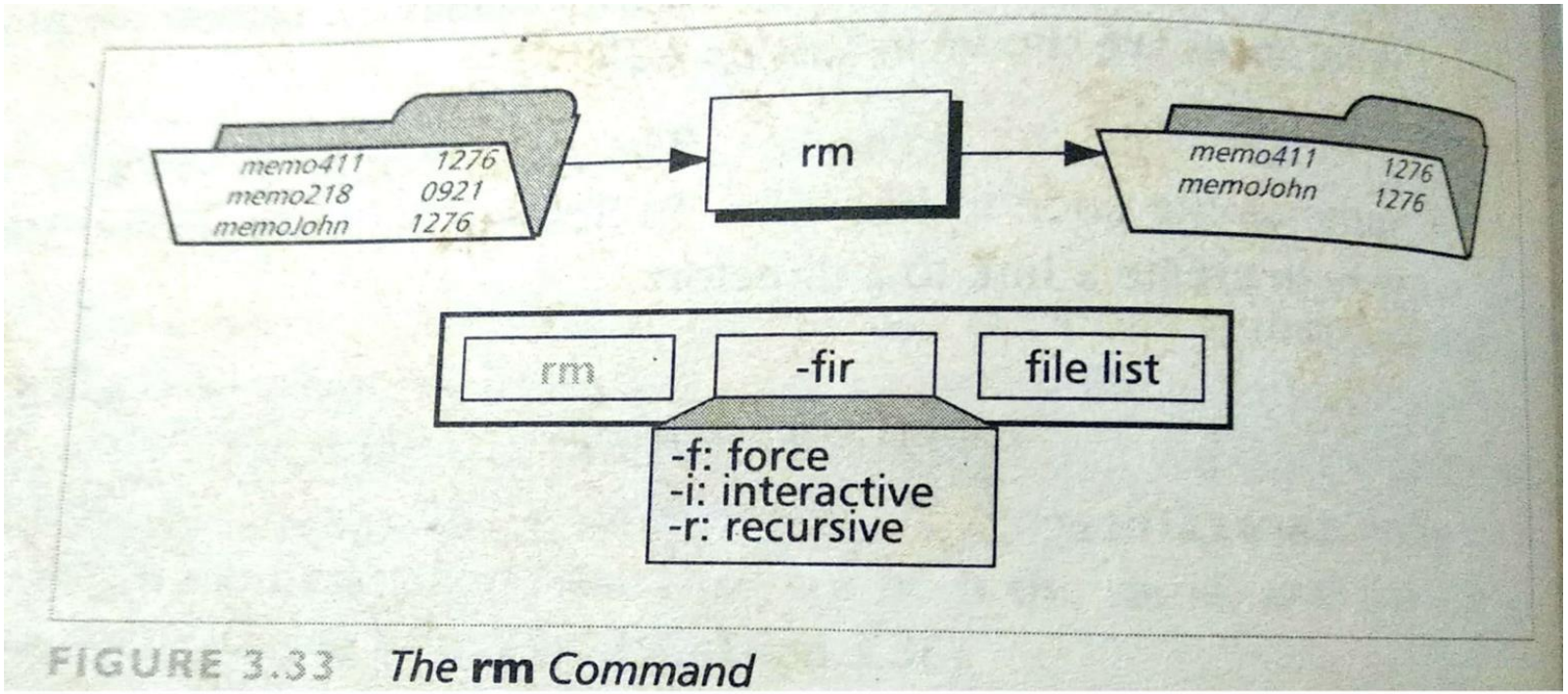


FIGURE 3.33 *The* **rm** *Command*

- **Find File (find) Command**
  - Given a large file environment, it can quickly become difficult to find a given file.
  - In UNIX, it is find. Find has no options. Its first argument is the path that we want to search, usually from our home directory. The second argument is the criterion that find needs to complete its search.

```
$ find DirC –name file3 –print
DirC/DirC1/file3
```

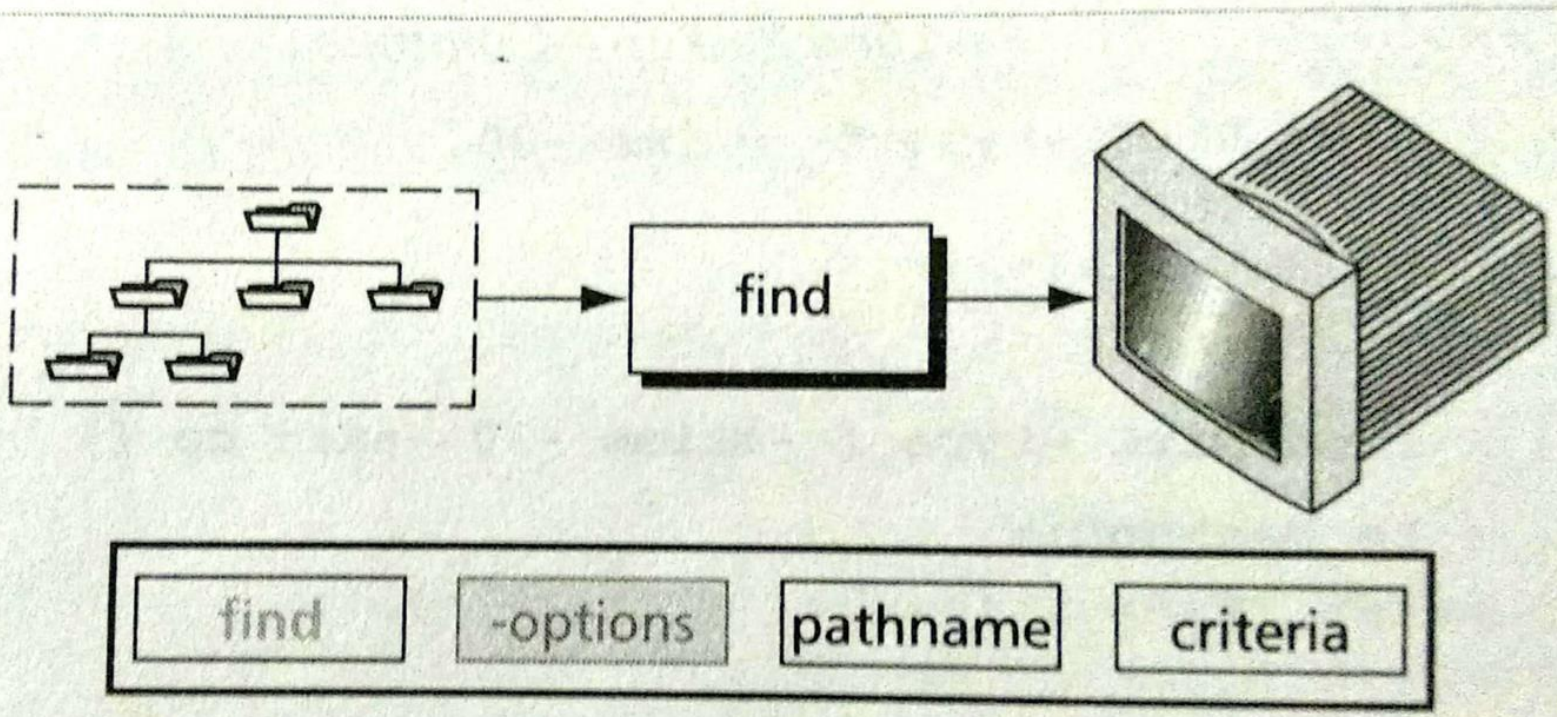


FIGURE 3.34 *The* **find** *Command*