# From the point of view of the user, there are three kinds of files:

**-ordinary disk files:** No structure is expected by the system

eg: text files, user programs, binary files

# -directories

# Directories provide the mapping between the names of files and the files themselves,

# A directory behaves exactly like an ordinary file except that it cannot be written on by unprivileged programs, so that the system controls the contents of directories. However, anyone with appropriate permission may read a directory just like any other file

-e.g.: root directory

-The same nondirectory file may appear in several directories under different names. This feature is called **linking**; a directory entry for a file is sometimes called a link

-That is, **a file does not exist within a particular directory, the** directory entry for a file consists merely of its name and a pointer to the information describing the file. Thus, **a file exists independently of any directory entry, although** in practice a file is made to disappear along with the last link to it.

# -special files.

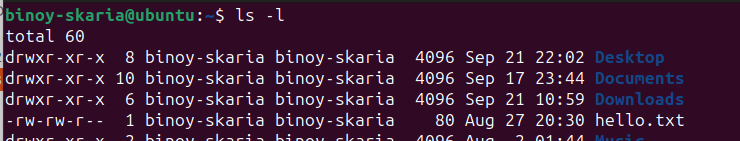
**-**each IO device

**-** Special files are read and written just like ordinary disk files, but requests to read or write result in activation of the associated device

**-** e. An entry for each special file resides in directory /dev

-e.g.: /dev/usb

# (image not important, just example)



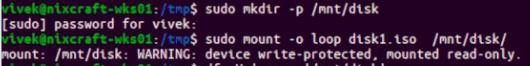
**How to mount a removable file system to a UNIX file system**

Call mount command

# Mount (name of existing ordinary file which we are going to mount on, name of special file (usb, HDD))

Mount replaces a leaf of the hierarchy tree (the ordinary file) by a whole new subtree (the hierarchy stored on the removable volume). After the mount, there is virtually no distinction between files on the removable volume and those in the permanent file system.

# (image not important, just example)



**Protection of a file**

Each user of the system is assigned a unique user identification number. When a file is created, it is marked with the user ID of its owner.

# Ususal bits, 6 bits here, 3 bits for owner rwx and 3 bits for others rwx

**7th bit is set-user-ID bit. If it is 1, then the file thinks that the current user is the owner.**

This mechanism is used to allow users to execute the

carefully written **commands(sudo)** which call **privileged system**

# Entries (apt install). Sudo sets set-user-ID bit to 1 and executes apt install.

**(don’t write sudo and apt install , they are not examples, they are part of linux, not old unix)**

# filep = open (name, flag)

**Name** indicates the **name of the file**. An arbitrary path name may be given.

The **flag** argument indicates **whether the file**

# is to be read, written, or “updated”.

The returned value **filep** is called a file descriptor. It is a **small integer used to identify the file.**

**It s**hould be said that the system **has sufficient internalinterlocks to maintain the logical consistency of the file system when two users engage simultaneously in such inconvenient activities as writing on the same file**, creating files in the same directory or deleting each other’s open files.

To create a new file or completely rewrite an old one,

there is a **create system call** which creates the given file if it does not exist

n = read(filep, buffer, count)

n= write(filep, buffer, count) , n represents, number of bytes read

**l**ocation = seek(filep, base, offset)

The

**actual offset from the beginning** of the file to which the

pointer was moved is returned in location.

**A** directory entry **contains for each file**:

only a **name for the associated file and a pointer to the file** itself. T**his pointer is an integer called the i-number (for index number) of the file. When the file is accessed, its i- number is used as an index into a system table (the i-list)** stored in a known part of the device on which the directory Resides. **I list / system table** contains all details of that file.

Eg: The purpose of an open or create system call is to turn the path name given by the user into an i-number by searching the explicitly or implicitly named directories. Once a file is

open, its device, i-number, and read/write pointer are stored in a system table indexed by the file descriptor returned by

the open or create. The file descriptor is an index of **I list / system table.**

**I** list / system table

1. Its owner.
2. Its protection bits.
3. The physical disk or tape addresses for the file contents.
4. Its size.
5. Time of last modification
6. The number of links to the file, that is, the number of times it appears in a directory

7.A bit indicating whether the file is a directory.

1. A bit indicating whether the file is a special file.
2. A bit indicating whether the file is “large” or “small.”

**A**n image is a computer execution environment. It

includes a core image, general register values, status of open files, current directory, and the like. An image is the current state of a pseudo computer.(context)

A process is the execution of an image. While the pro- cessor is executing on behalf of a process, the image must reside in core.

core image divided to 3 : text segment(code), data segment(variable), stack segment (could not see heap)

**PIPES:**

filep = pipe( ) returns a file descriptor filep and creates an interprocess channel called a pipe. This channel, like other open flies, is passed from parent to child process in the image by the fork call

used for inter process communication

**W**ait() :

processid = wait( )

causes its caller to suspend execution until one of its chil- dren has completed execution. Then wait returns the pro- cessid of the terminated process. An error return is taken if the calling process has no descendants.

**E**xit() : terminates a process, destroys its image, closes its open files, and obliterates it

**S**hell: communication with UNIX is carried out with SHELL, it is a command line interpreter.

command arg1 arg2 ⋅ ⋅ ⋅ argn… In command, specify the command file, if file path is not specified . the Shell prefixes the string /bin/ to command and attempts again to find the file.

WORKING

* Programs executed by the Shell, however, start off with two open files which have file descriptors 0 and 1. As such a program begins execution, file 1 is open for writing, and is best understood as the standard output file. Except under circumstances indicated below, this file is the user’s typewriter. Thus programs which wish to write informative or diagnostic information ordinarily use file descriptor 1. Conversely, file 0 starts off open for reading, and programs which wish to read messages typed by the user usually read this file.

**R**edirection: ls>output.txt , sort<input.txt

**F**ilters (means pipes) : cat input.txt | sort

**M**ultitasking : If a command is

followed by “&”, the Shell will not wait for the command

to finish before prompting again; instead, it is ready immediately to accept a new command. For example,

Eg1 : gedit &

Eg 2: Cmd1 & cmd2 & cmd3

**S**h < shellscript.sh : shell as a command, executes script in .sh file

**G**eneral Shell Implementation The Shell

analyzes the command line, putting the arguments in a form appropriate for execute. Then **fork** is called. The child pro- cess, whose code of course is still that of the Shell, attempts to perform an **execute** with the appropriate arguments. If successful, this will bring in and start execution of the pro- gram whose name was given. Meanwhile, the other process resulting from the fork, which is the parent process, waits for the child process to die.

**F**aults

-When an illegal action is caught, unless

other arrangements have been made, the system terminates the process and writes the user’s image on file core in the current directory.

-user has second thoughts may be

halted by the use of the **interrupt** signal, which is generated by typing the “delete” character. Unless special action has been taken, this signal simply causes the program to cease execution without producing a core image file.

-There is also a **quit** signal which is used to force a core image to be produced.