UNIT-III

FILES and DIRECTORIES

FILES:

- All data in Linux is organized into files.
- All files are organized into directories.
- These directories are organized into a tree-like structure called the filesystem.

Three basic types of files

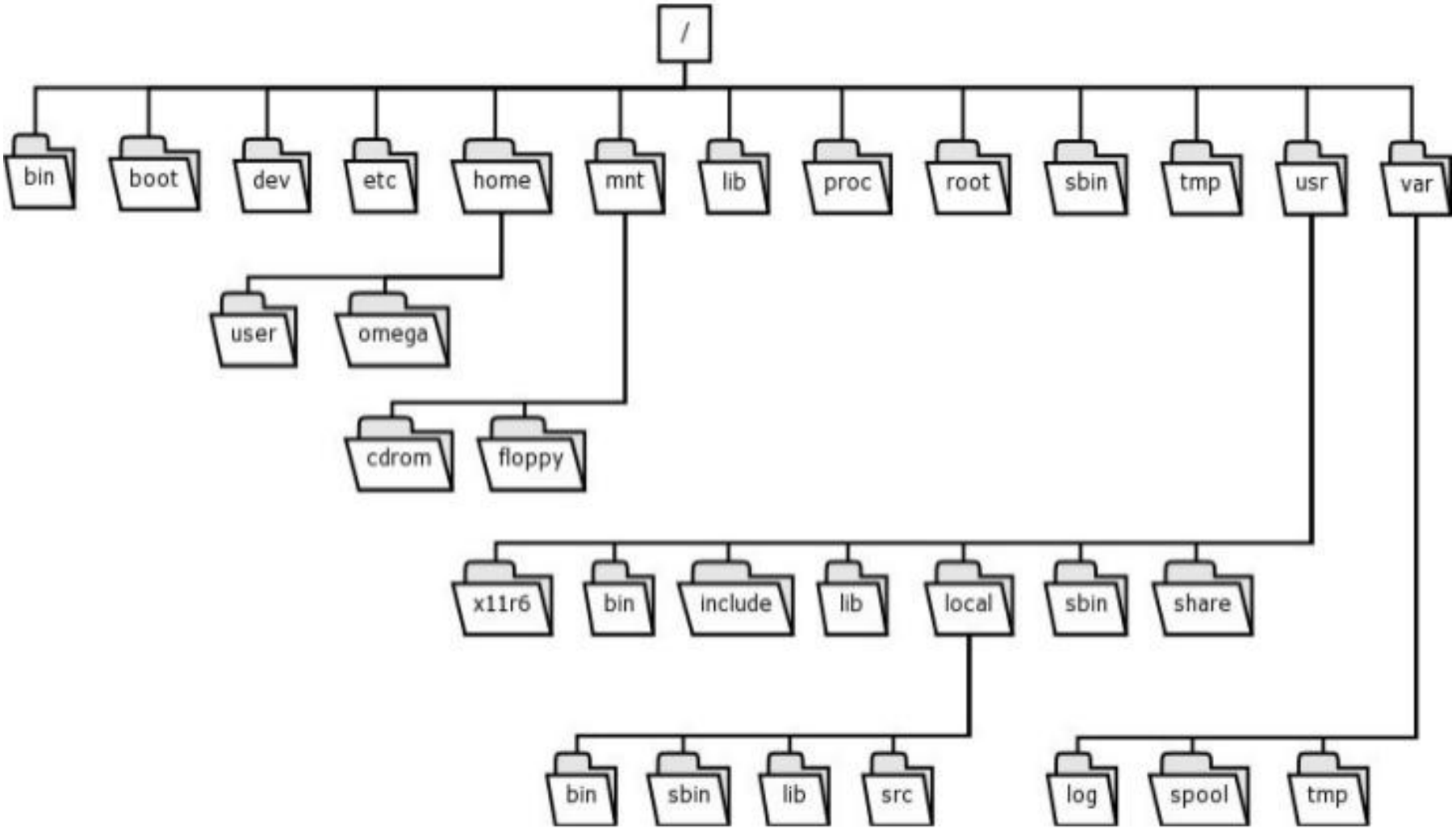
- **Ordinary Files** − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- Readable files
- Binary files
- Image files
- Compressed files and so on.

- **Directories** − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

- **Listing Files**
- **Hidden Files**
- **Creating Files**
- **Editing Files**
- **Display Content of a File**
- **Counting Words in a File**
- **Copying Files**
- **Renaming Files**
- **Deleting Files**
- **File permissions**

Linux File System Structure

Defines the directory structure and directory contents in Linux operating system



- /(**root**): All files and directory appears(starts) under the root directory /.
- Only root user has the right to write under this directory.
- /**bin:** essential command binaries that need to be available in single user mode; for all users, cat,ls cp….
- Commands used by all the users of the system are located here.
- /**boot** : boot loader files like kernels, initrd.

Linux File Tree (Continue)

- The following directories are required in /:
  - bin — Essential command binaries
  - boot — Static files of the boot loader
  - Dev — Device files
  - etc — Host-specific system configuration
  - lib — Essential shared libraries and kernel modules
  - media — Mount point for removeable media
  - mnt — Mount point for mounting a filesystem temporarily
  - opt — Add-on application software packages
  - sbin — Essential system binaries
  - srv — Data for services provided by this system
  - tmp — Temporary files
  - usr — Secondary hierarchy
  - var

Types of files

- **Regular file(-)**
- **Directory file(d)**
- **Special file**

- **Block file(b)-**These files are hardware files most of them are present in /dev.
- **Character device file(c)-**Provides a serial stream of input or output
- **Named pipe file or just a pipe file(p)**
- **Symbolic link file(l)**
- **Socket file(s)**

Inodes

The index node(inode) is a data structure that stores various information about a file in linux.

- Mode
- Ownership
- Group
- File type
- File size
- No. of links

Each inode is identified by an integer number, assigned to a file when it is created.

$ls –il

$ls –i myfile.txt

Inode doesn’t store the content of the file and filename

What happens with inode number when

- copy,
- move or
- delete a file on the filesystem.

"An inode is a data structure … … that stores all the information about a file except its name and its actual data."

In the case of inodes are full. we need to remove unused files from the filesystem to make Inode free. There is no option to increase/decrease inodes on disk. It only created during the creation of filesystem on any disk.

Library functions kernel support for files

Library functions can be of two types :

- Functions which do not call any system call.
- Functions that make a system call.

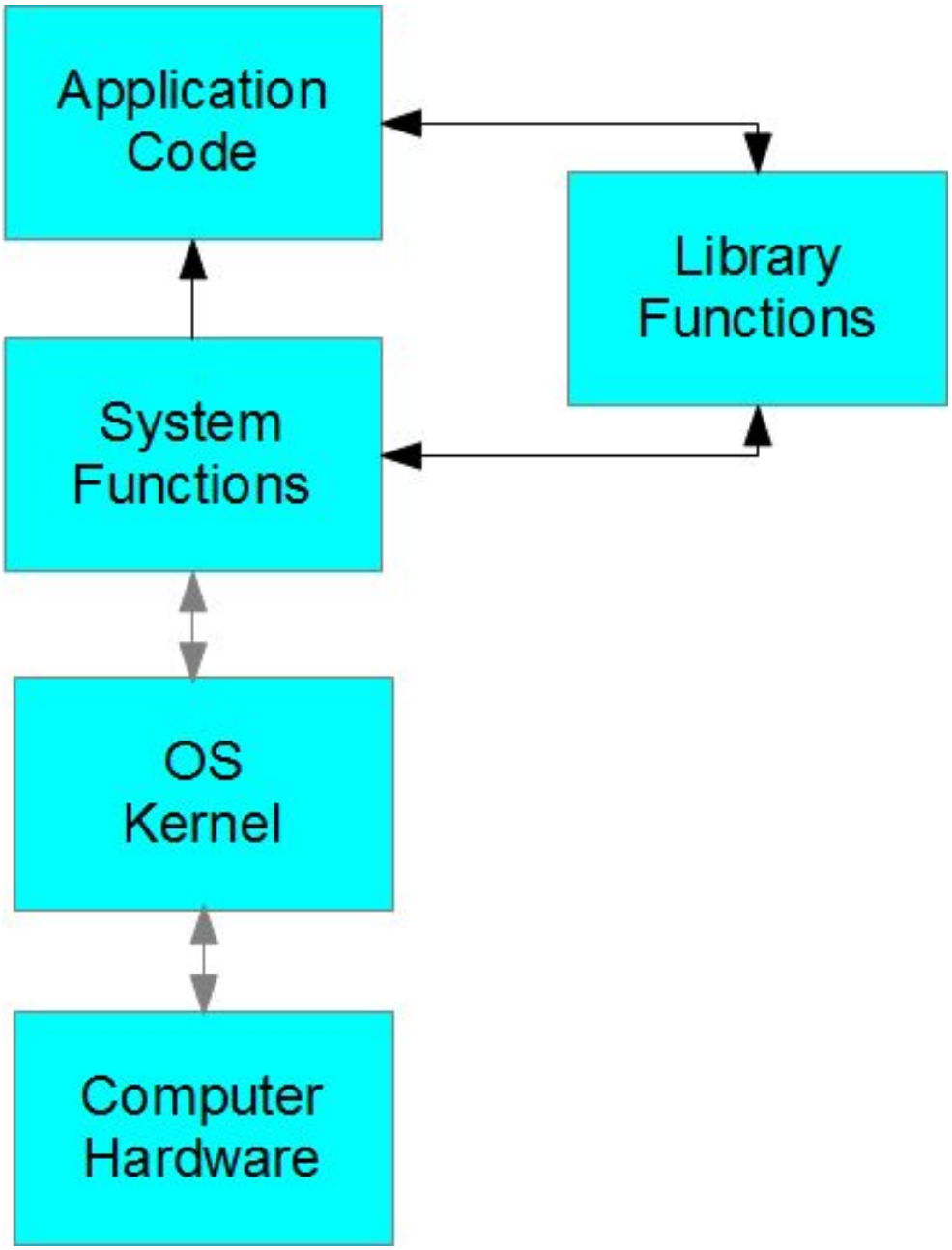
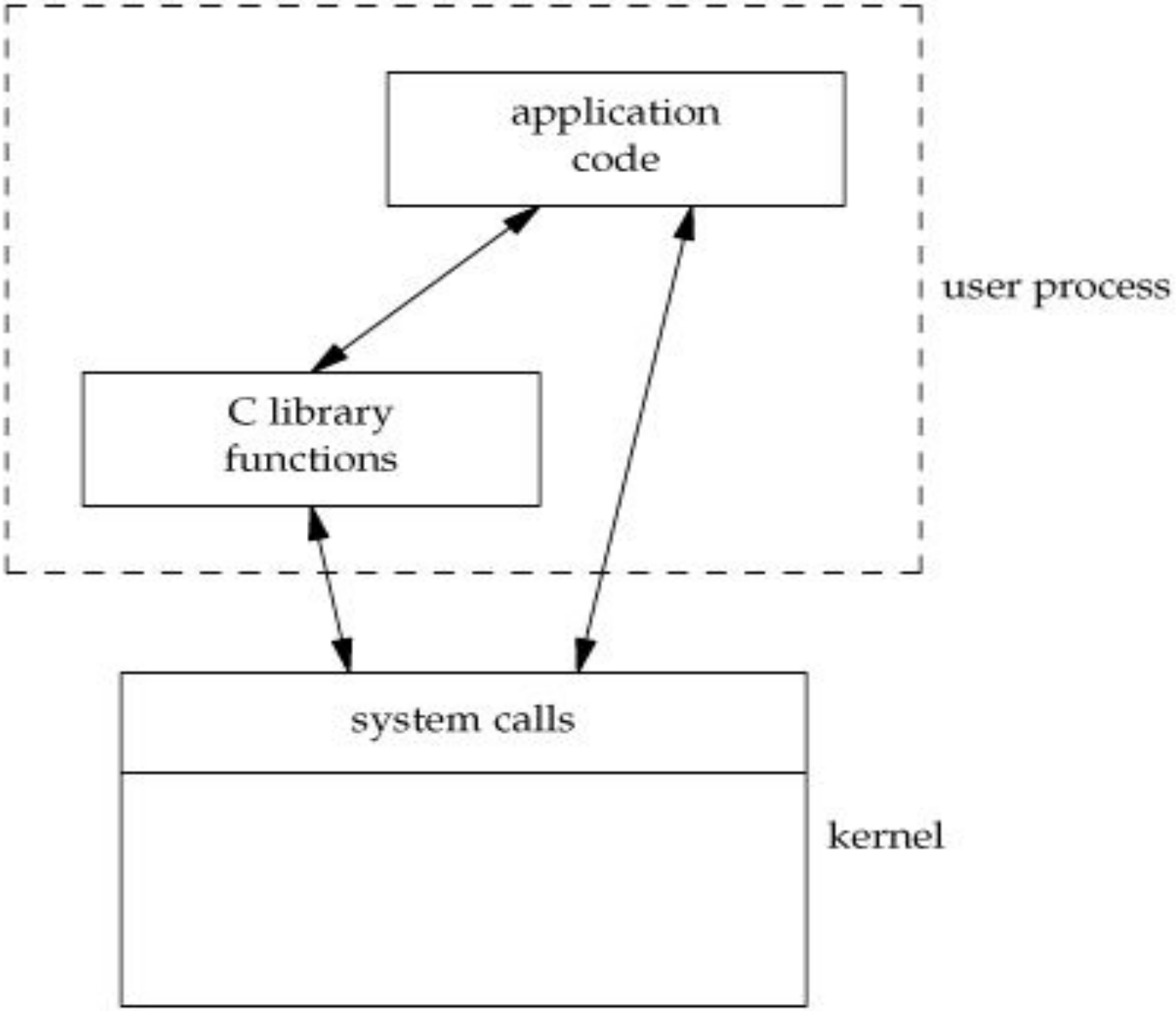

Application Code
Library Functions
System Functions
OS Kernel
Computer Hardware

Figure 1.12. Difference between C library functions and system calls



System calls for file I/O operations

- Open
- Create
- Read
- Write
- Close

- The **system call** is a way for programs to interact with the operating system.
- When the program makes a **system call** at that time it makes a request to the operating system's kernel.

- **O_RDONLY** : Opens the file for reading.
- **O_WRONLY**: Opens the file for writing.
- **O_RDWR**: The file is opened for reading and writing.
- **O_APPEND**: It writes successively to the end of the file.
- **O_CREAT** : The file is created in case it didn't already exist.
- **O_TRUNC :** If the file exists all of its content will be deleted.

- The function returns the file descriptor or in case of an error it returns the value -1.

This call is equivalent with:

open(path, O_WRONLY | O_CREAT | O_TRUNC, mod);

- For closing a file and thus eliminating the assigned descriptor we use the system call *close*.

```
#include <unistd.h>
int close(int fd);
```

```
#include <unistd.h>
ssize_t read(int fd, void* buf, size_t noct);
```

- The function returns the number of bytes read

```
#include <unistd.h>
ssize_t write(int fd, const void* buf, size_t noct);
```

The function returns the number of bytes written

-1 in case an error occurred.

links

- Soft/symbolic (pointer)
- Hard link (copy/Mirror image)