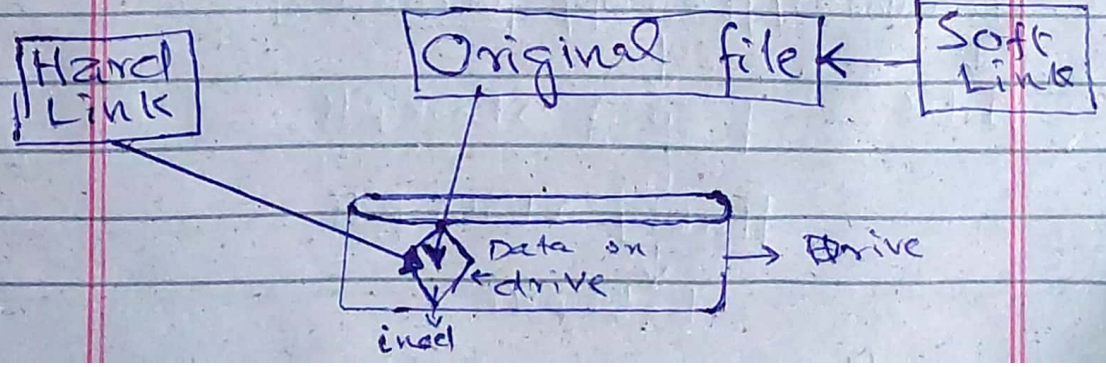
System Programming Notes

chapter # 5. Files and Directories

→ Unix file systems are tree structured, with nodes representing files and arcs representing the contains relationship.

→ Hard link: pointing directly to a structure containing the file location information. It is a file all its own. point to exact spot on a harddrive where inodes stores the data.

→ Soft link: points to name of the original file, rather than to a spout on hard drive. It isn't a seperate file. Sometime it is refer to symbolic link or symlink.

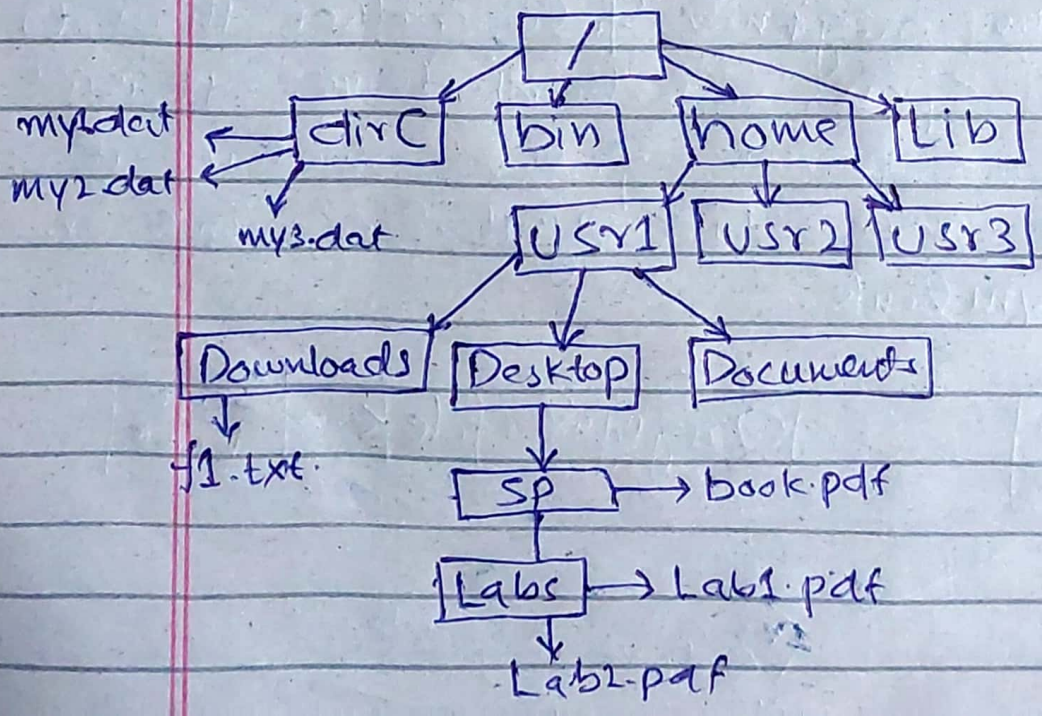


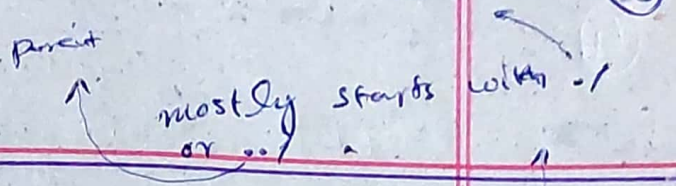
5.1 Unix file system navigation.

→ Filesystem is the way in which files, directories are named and where they are placed logically for storage and retrival. eg NTFS, FAT etc.

→ A file is any computer document while a directory is special file contains information about other files and folder called directory entries dirent.

→ The 'root' directory of filesystem is at top of tree and everything else are branches of it.





→ A relative path describe the location of a file relative to current working directory. An absolute path describe the location from root directory. mostly starts with /.

→ root directory has both dot and dot-dot pointing to itself.

→ The 'PWD' environment variable specify the current working directory of a process. Also 'getcwd' retreive the current working directory.

→ Synopsis:

```
#include <unistd.h>

int chdir(const char* path);
```

use to change the current working directory within a process.

~ * path : The specified path become the current working directory for calling process.

(4)

* 0: when successful
* -1: when unsuccessful and set 'errno'.

→ Synopsis

```
#include <unistd.h>
char *getcwd (char* buff,
    size_t size);
```

~ This function return the pathname of current working directory.

~ *buff: represents a user-supplied buffer for holding pathname of cwd.

~ size: specifies maximum length pathname that a buf can accomodate, including the trailing string terminator.

* a pointer to buff when successful
* returns NULL and set errno.

→ PATH_MAX is an optional POSIX constant specifying the maximum length of a pathname (including '\0') for the implementation. It may or may not be defined in 'limits.h'.

5.2 Directory Access:

→ Directory cannot be accessed with ordinary 'open', 'close', 'read' functions. Inste the required specilize function whose corresponding names end with 'dir'.

→ Synopsis:

```
#include <dirent.h>
DIR *opendir(const char* dirname);
```

This function provides a handle to a directory stream that is positioned at first entry in directory

~ *dirname: path to directory

* pointer to directory object when successful.

* NULL: returns a null pointer and set errno when unsuccessfull.

→ DIR type represent directory stream which is an order sequence of all of the directory entries in a particular order.

→ Synopsis:
#include <dirent.h>
struct dirent *readdir(DIR *dirp);

The 'readdir' function reads a directory by returning a successive entries in a directory stream pointed 'dirp'. It moves the stream to next position after each call

~ *dirp:

* struct dirent *: returns a pointer to the structure containing the information about next directory entry.
* NULL: also set errno.

→ readdir also return NULL to indicate the end of directory, but in this case it doesn't change errno.

→ Synopsis:

```
#include <dirent.h>
int closedir(DIR *dirp)
void rewinddir(DIR *dirp)
```

The 'closedir' closes a directory stream and the 'rewinddir' func repositions the directory stream to its beginning.

~ *dirp: correspond to an open directory stream

*s 0

*us -1 and set errno

→ Traditionally, UNIX directory entries only contains filenames/ directorynames and inode numbers. The inode number is an index into a table containing the other information about a file.

(8)

→ We have three functions for retreiving file status info.
- fstat access file with open file descriptor.
- stat and lstat access by file name.

→ Synopsis:
#include <sys/stat.h>

- int lstat (const char* restrict path, struct stat *restrict buf);
- int stat (const char *restrict path, struct stat *restrict buf);

~ path: name of symbolic link of a file whose status is to be returned
~ *buf; point to user supplied buffer into which the information is stored.

*S 0
*US -1 and set errno.

- int fstat(int fildes, struct stat *buf);

~ fildes: filedescriptor of open file.
~ buf: as above function.

* 0
* -1 and set errno.

→ 'ctime' function takes a time and return its readable string format. keep in mind ctime returned string end with new line. we can remove the new line by using strtok function as:
strtok(ctime(time), "\n");

→ The stat contain members which store time of last access, last data modification and last status chang. These are stored in 'time_t st_atime, time_t st_mtime and time_t st_ctime' respectively.

→ It has also a member named 'mode_t st_mode' specifies access permisson and type of

file.

- For file type Posix specifies macros (on page 197).

" S_ISDIR(st_mode) ? printf("d") : printf("-");"

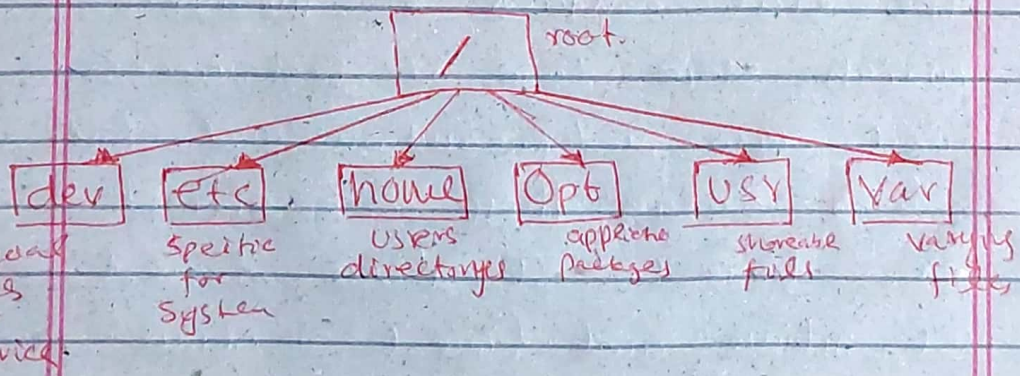
- For permission take bitwise and of permission bit and st_mod.

" S_IRUSR & st_mode ? printf("r") : printf("-");"

5.3 Unix file system Implementation:

→ Disk formatting divides a physical disk into regions called partitions.

→ All fully qualified paths in Unix start from root directory /.

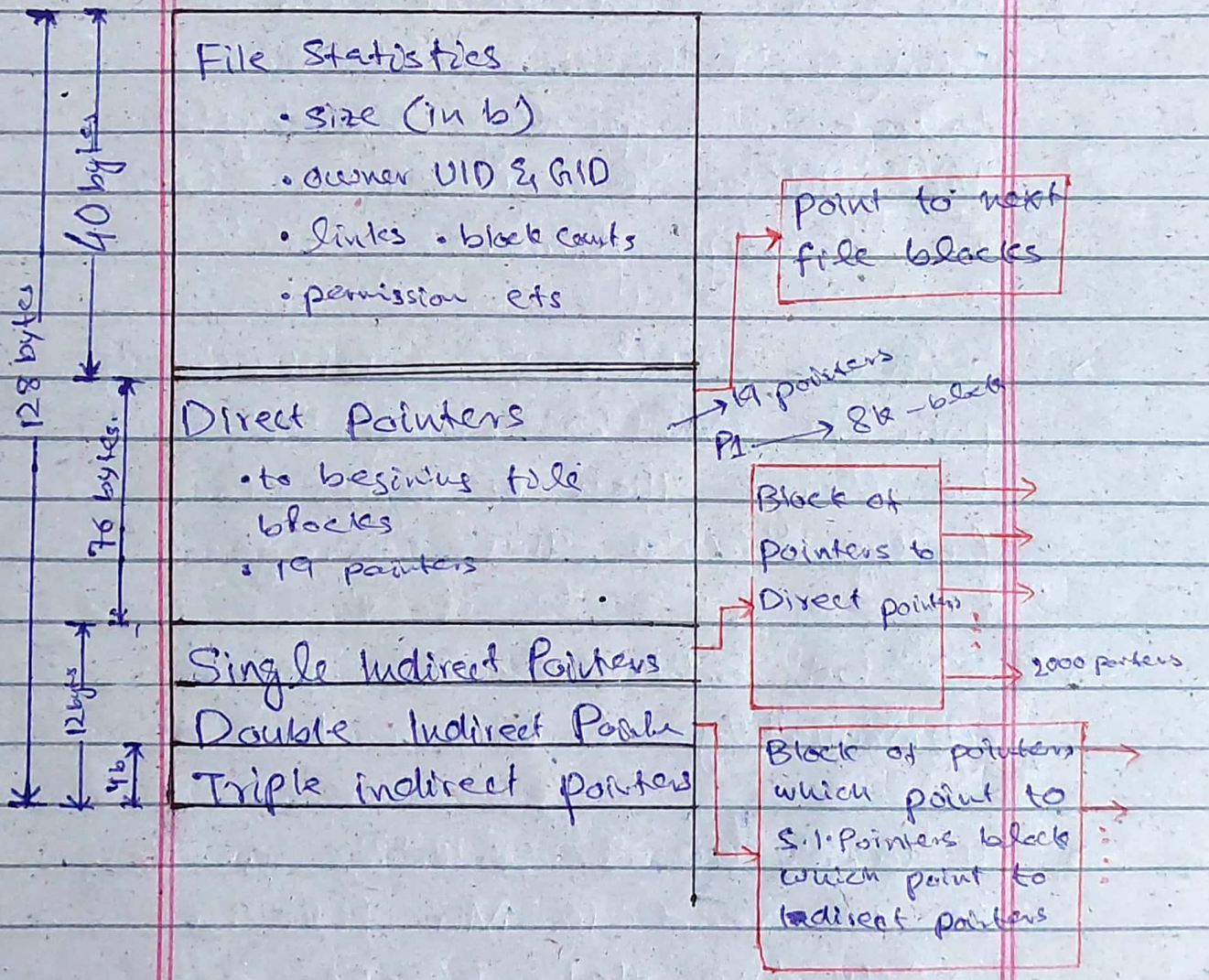


→ Directory entries contain filename and a reference to fixed-length structure called inod.

→ The inod contains information about

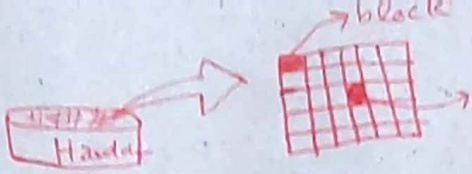
the file size, file location, owner of file, time of creation, last access and modification, permision and so on.

→ Below figure shows the inod structure for a typical file.



→ Direct pointer to first few data blocks of file.

→ If file is large, the indirect



pointer is a pointer to a block of pointers that points to additional data blocks.

→ Similarly double indirect pointers points to a block of single indirect pointers and so on.

→ The block can mean different things. In this context a block is typically 8k bytes. The number of bytes in a block is always power of 2.

→ Since $2048^3 = 2^{33}$ (T.I.P Case), pointers ~~will be~~ would need to be longe than 4 bytes to fully address this storage.

→ 32-bit addresses can access aproximatly 4 billion blocks.

→ It takes about 20 times as long as to read 16-MB file in which the data blocks are randomly placed than one in which the datablocks are contiguous.

→ Directory implementation that only contains name and inod has following advantages:

- Changing file name or moving a file (mv command) only require changing directory entry.
- only one physical copy of file needs to exists on disk.
- Directory entries are of variable length because the filename is of variable length.

5.4 Hard Links and Symbolic (Soft) Links::

→ A link, sometimes called a hard link, is a director entry.

→ A symbolic link sometimes called a soft link is a file that ~~use~~ stores a string used to modify the pathname when it is encountered during pathname resolution

→ See the figure on first page for better understanding.

→ We can create additional

links to a file with the "ln" shell commander the link function.

→ Creation of new link add directory entry and increment count of links pointing to inod. and by removing ('rm') the file it do vice versa.

→ Synopsis:

The link function creates a new entry for existing file: #include

int link(const char* path1, const char* path2);

* path1: path to file
* path2: new path of file.

* 0
* -1 and set errno.

→ Synopsis:

The unlink function removes the directory entry specified by

path. If link count is 0 and no process has opened it the unlink frees the space occupied by file.

```
#include <unistd.h>
int unlink(const char* path);
```

~ path: path to file to be removed.

*S 0

*US -1 and set errno

→ A symbolic link is a file that containing the name of another file or directory.

→ We can create a soft link by 'ln' command with '-s' option or by invoking 'symlink' function.

→ Synopsis:

```
#include <unistd.h>
int symlink(const char* path1,
const char* path2);
```

~ path1: path that is a string that will be content of link

~ path2: gives pathname of link.

(16)

*S 0; *US -1 and set errno.