Assignment - 6

Operating Systems

Question 1

☞ copy the code fragment on the directory slide
to write a ‘mini-ls’ program
☞ it should list the file pointed to by its program first
argument (argv[1])
☞ compile and run it
☞ now modify it to use stat on each file
☞ get it to add a slash ’/’ to the end of directories
☞ use your imagination to add other status info!
☞ compile and run again
☞ if you have time, try adding a ‘-L’ option
when the ‘-L’ option is given, your program
should give the details of symbolic links themselves
as the ‘-L’ option does for the real ls

Group - 6

Abhijeet Sonkar

IIB2019009

Aditya Aggarwal

IIT2019210

Ambika Singh

IIB2019017

Avneesh Kumar

IIB2019010

Divy Agarwal

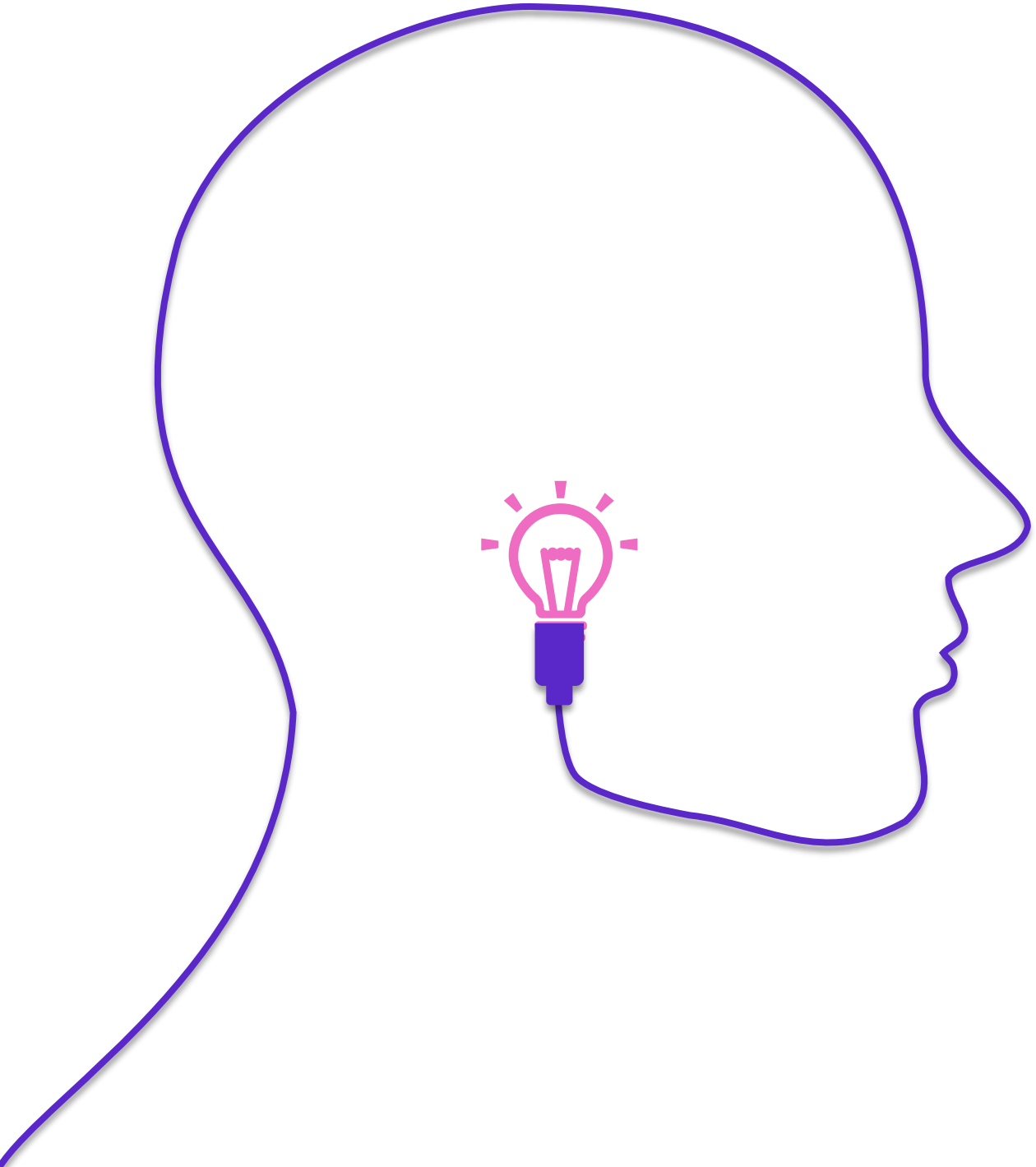
IIT2019211

Table of content

01 Basic Concepts

02 Code and Explanations

03 Output Screenshots



1. Basic Concepts

`ls` command

A Linux shell command that lists directory contents of files and directories. Below are some options available in ls command. They can be combine also as per requirement.

list all files including hidden file starting with '.'	`ls -a`	`ls -r`	**list in reverse order**
list file's inode index number	`ls -i`	`ls -s`	**list file size**
list with long format - show permissions	`ls -l`	`ls -t`	**sort by time & date**
sort by extension name	`ls -X`	`ls -S`	**sort by file size**

File System

- It is a structured collection of files on a disk drive or a partition.
- A partition is a segment of memory and contains some specific data. In our machine, there can be various partitions of the memory.
- It is generally a built-in layer of a Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file.
- It maintain consistency by treating everything as a file (even the hardware devices).
- The keyboard, mouse, printers, monitor, hard disk, processes, even the directories are treated as files in Linux.

inode in Unix

Inode: It is a data structure, contains a list of all the blocks in which a file is stored, the owner information for that file, permissions and all other attributes that are set for the file.

Inode number: is also known as index number. An inode is a unique number assigned to files and directories while it is created. The inode number will be unique to entire filesystem.

Links in Linux:

Link is a pointer to another file. There are two types of links.

Hard Link

- A hard link is a name that references an inode.
- It means that if 'file1' has a hard link named 'file2', then both of these files refer to same inode. So, when you create a hard link for a file, all you really do is add a new name to an inode.

Soft(Symbolic) Link

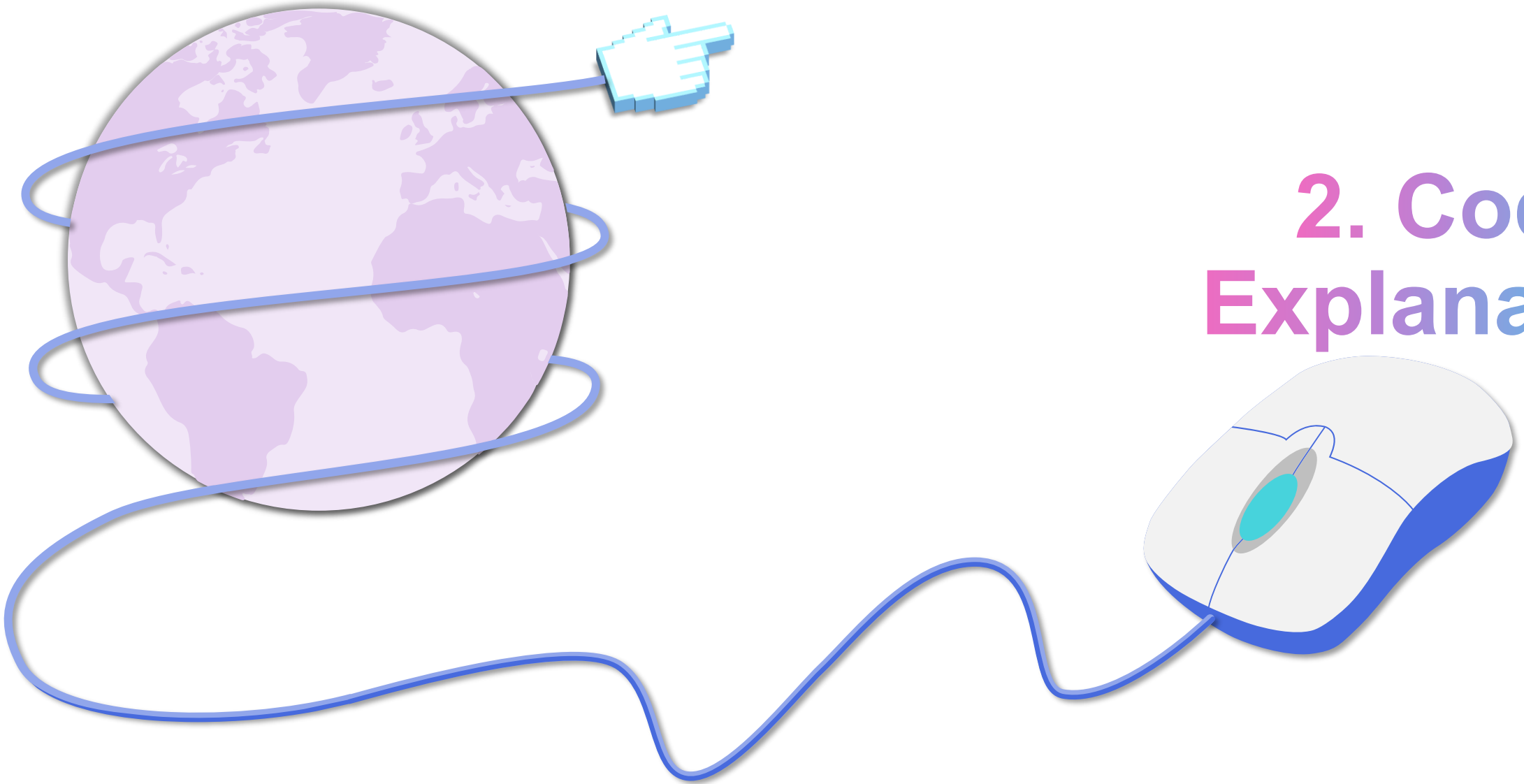
- It is a separate file whose contents point to the linked-to file.
- The original file is just a name that is connected directly to the inode, and the symbolic link refers to the name. The size of the symbolic link is the number of bytes in the name of the file it refers to, because no other information is available in the symbolic link.

tm Structure

Below are the fields present in tm structure.

```
struct tm {
   int tm_sec;         /* seconds,  range 0 to 59          */
   int tm_min;         /* minutes, range 0 to 59           */
   int tm_hour;        /* hours, range 0 to 23             */
   int tm_mday;        /* day of the month, range 1 to 31  */
   int tm_mon;         /* month, range 0 to 11             */
   int tm_year;        /* The number of years since 1900   */
   int tm_wday;        /* day of the week, range 0 to 6    */
   int tm_yday;        /* day in the year, range 0 to 365  */
   int tm_isdst;       /* daylight saving time             */

};
```



2. Code & Explanation

Here inside `main`. We check that valid arguments have been passed.

- If invalid arguments are passed then we print error on screen.
- If valid arguments are passed, then we proceed with printing directory details using `print_directory_details() function.`
- If only `./mini-ls` is passed then our program work on current directory.
- If `./mini-ls -L` are passed then we again work on current directory, but here we deal with symbolic links.
- If `./mini-ls -L “absolute path”` then works on given directory, but here we deal with symbolic links.
- If `./mini-ls “absolute path”` then works on given directory.

```
int main(int argc, char *argv[]){
    if (argc > 3 || (argc == 3 && strcmp(argv[1], "-L"))){
        perror("Invalid Request");
        exit(1);
    }
    if(argc==1){
        argc= 2;
        print_directory_details(argc,".");
    }
    else if(argc==2){
        if(!strcmp(argv[1],"-L")){
            argc=3;
            print_directory_details(argc,".");
        }
        else{
            print_directory_details(argc,argv[1]);
        }
    }
    else{
        print_directory_details(argc,argv[2]);
    }
    return 0;
}
```

`mode_t` Specifies the mode of the file. This includes file type information and the file permission bits.

`S_ISREG()`	returns non-zero if the file is a regular file.
`S_ISDIR()`	returns non-zero if the file is a directory.
`S_ISFIFO()`	returns non-zero if the file is a FIFO special file, or a pipe.
`S_ISSOCK()`	returns non-zero if the file is a socket.
`S_ISCHR()`	returns non-zero if the file is a character special file. (like Terminal)
`S_ISBLK()`	returns non-zero if the file is a block special file (a device like a disk).
`S_ISLNK()`	returns non-zero if the file is a symbolic link.

`S_IRUSR`	Read permission bit for the owner of the file. On many systems this bit is 0400.
`S_IWUSR`	Write permission bit for the owner of the file. Usually 0200.
`S_IXUSR`	Execute or search permission bit for the owner of the file. Usually 0100.
`S_IRGRP`	Read permission bit for the group owner of the file. Usually 040.
`S_IWGRP`	Write permission bit for the group owner of the file. Usually 020.
`S_IXGRP`	Execute or search permission bit for the group owner of the file. Usually 010.
`S_IROTH`	Read permission bit for other users. Usually 04.
`S_IWOTH`	Write permission bit for other users. Usually 02.
`S_IXOTH`	Execute or search permission bit for other users. Usually 01.

At the end we return string loaded with all permission eg. `-rw-rw-r--`

```
string permissions(mode_t st){
    char perms[11];
    if(S_ISREG(st))
        perms[0] = '-';
    else if(S_ISDIR(st))
        perms[0] = 'd';
    else if(S_ISFIFO(st))
        perms[0] = '|';
    else if(S_ISSOCK(st))
        perms[0] = 's';
    else if(S_ISCHR(st))
        perms[0] = 'c';
    else if(S_ISBLK(st))
        perms[0] = 'b';
    else if(S_ISLNK(st))
        perms[0] = 'l';
    perms[1]=(st&S_IRUSR)?'r':'-';
    perms[2]=(st&S_IWUSR)?'w':'-';
    perms[3]=(st&S_IXUSR)?'x':'-';
    perms[4]=(st&S_IRGRP)?'r':'-';
    perms[5]=(st&S_IWGRP)?'w':'-';
    perms[6]=(st&S_IXGRP)?'x':'-';
    perms[7]=(st&S_IROTH)?'r':'-';
    perms[8]=(st&S_IWOTH)?'w':'-';
    perms[9]=(st&S_IXOTH)?'x':'-';
    perms[10]='\0';
    string d=perms;
    return d;
}
```

buffer, details, s are of type stat. It is a structure which stores several information as inode number, protection, no. of hard links, etc.

The opendir() function opens a directory stream corresponding to the directory name(path), and returns a pointer to the directory stream. The stream is positioned at the first entry in the directory.

The readdir() function returns a pointer to a dirent structure(directory) representing the next directory entry in the directory stream pointed to by open_directory. It returns NULL on reaching the end of the directory stream or if an error occurred. So we have used a while loop to iterate over all the directory entries.

- stat() function is used to list properties of a file identified by path. It reads all file properties and dumps to buffer structure.
- lstat() function gets status information about a specified file and places it in the area of memory pointed to by details. If the named file is a symbolic link, lstat() returns information about the symbolic link itself.
- S_ISDIR() to check for a directory,

If it is a directory the '/' is added at the end and printed.

```
int print_directory_details(int argc,string str){
    struct stat buffer;
    struct stat details;
    string path=str;
    DIR *open_directory=opendir(path.c_str());
    directory=readdir(open_directory);
    int flag=0;
    while(directory!= NULL){
        string temp = directory->d_name;
        string s = path;
        s+="/";
        s+=temp;
        if(stat(s.c_str(),&buffer)==-1){
            perror("stat");
            return errno;
        }
        if(lstat(s.c_str(),&details)==-1){
            perror("stat");
            return errno;
        }
        if(S_ISDIR(buffer.st_mode)){
            temp+="/";
        }
        if(argc!=3)
            cout<<temp<<" ";
```

`permissions()` function is used to get string representing all the permissions. Then we print that out.

The `passwd` structure is returned by the `getpwnam()` `getpwnam_r()`, `getpwuid()` and `getpwuid_r()` functions. It provides information about a user account. The structure has fields for user login name, ID, grp ID, class etc.

The `group` structure is used to hold information about an entry in the system group database. It field as name, ID , and vector of pointers to the names of users of group.

`tm` structure stores the values that represent the corresponding local time. It contain field as explained earlier.

The `getpwuid_r()` function is a reentrant version of `getpwuid()`. It lets a process gain more knowledge about user with the given *uid* (`buffer.st_uid`). updates the passwd structure pointed to by `pwent` and stores a pointer to that structure at the location pointed by `pwentp`.

The `getgrgid_r()` function updates the group structure pointed to by `grp` and stores a pointer to that structure at the location pointed to by `grpt`. The structure contains an entry from the group database with a matching *gid* (`buffer.st_gid`).

```
if(argc==3){
            if(flag==0){
                cout<<"permission    no. of hardlinks   own
er of file   user group   size   date & time modified   dat
e & time accessed last   file or directory name\n\n";
                flag=1;
            }
            string permission=permissions(details.st_mode);
            cout<<permission<<"\t    ";
            printf("%d\t\t     ", (int)buffer.st_nlink);
            struct passwd pwent,*pwentp;
            struct group grp,*grpt;
            char datestring[256];
            struct tm time;
            char storage[128];
            if(!getpwuid_r(buffer.st_uid, &pwent, storage,
sizeof(storage), &pwentp)){
                printf("%s\t   ",pwent.pw_name);
            }
            else{
                printf("%d\t   ",buffer.st_uid);
            }
            if(!getgrgid_r(buffer.st_gid, &grp, storage, si
zeof(storage), &grpt)){
                printf("%s\t     ",grp.gr_name);
            }
            else{
                printf("%d\t     ",buffer.st_gid);
            }
```

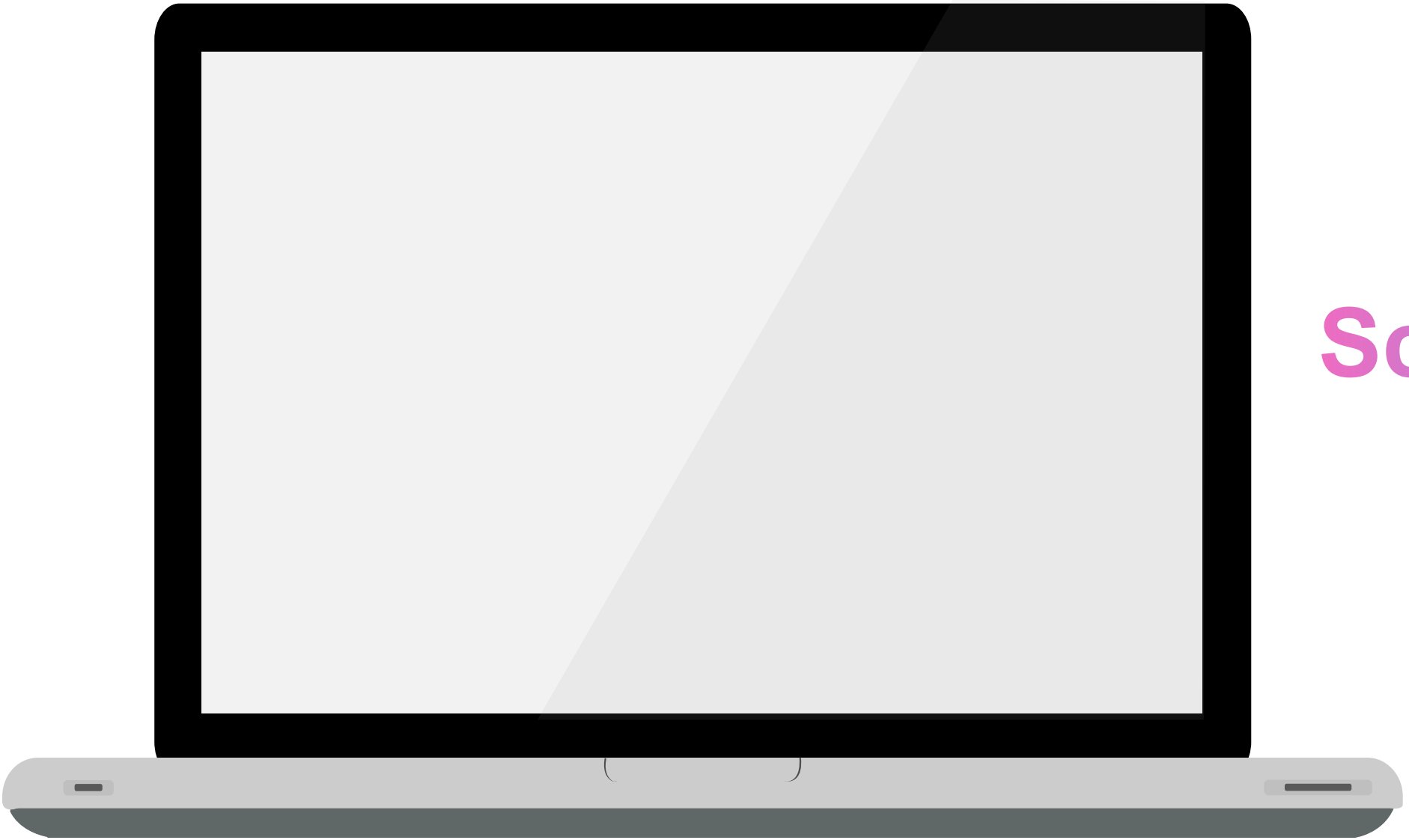
`localtime_r()` function is the restartable version of `localtime()`. It is the same as localtime() except that it passes in the place to store the returned structure `time`.

`strftime()` function formats the broken-down time (`time`) according to the formatting rules specified in format and store it in character array `datestring`.

`S_ISLNK()` to check whether a file has a symbolic link or not. If it is symbolic link then we use `readlink()` which places the contents of the symbolic link pathname(`s`) in the buffer (`link_to`), which has size `PATH_MAX`. readlink() does not append a null byte to buffer. It will truncate the contents (to a length of `PATH_MAX` characters), in case the buffer is too small to hold all of the contents.

`closedir()` closes the directory stream referred to by the argument `open_directory`. Upon return, the value of *argument* may no longer point to an accessible object of the type **DIR**. If a file descriptor is used to implement type **DIR**, that file descriptor shall be closed.

```
printf("%5d   ",(int)buffer.st_size);
            localtime_r(&buffer.st_mtime, &time);
            strftime(datestring,sizeof(datestring),
"%F %T",&time);
            printf(" %s   ",datestring);
            localtime_r(&buffer.st_atime, &time);
            strftime(datestring,sizeof(datestring),
"%F %T",&time);
            printf(" %s\t ",datestring);
            if(S_ISLNK(details.st_mode)){
                char link_to[PATH_MAX];
                ssize_t r=readlink(s.c_str(),link_t
o,PATH_MAX);
                if(r!=-1){
                    link_to[r]='\0';
                    cout<<temp;
                    printf(" -> %s\n",link_to);
                }
            }
            else{
                cout<<temp <<" ";
                cout<<endl;
            }
        }
        else{
            cout<<endl;
        }
        directory=readdir(open_directory);
    }
    closedir(open_directory);
```



3. Output Screenshots

Activities Terminal Sat 02:31

divy@anonymous: ~/Downloads/question-1

File Edit View Search Terminal Help

```
divy@anonymous:~/Downloads/question-1$ g++ mini-ls.cpp -o mini-ls
divy@anonymous:~/Downloads/question-1$ ./mini-ls -L /home/divy/Desktop/toc
permission    no. of hardlinks   owner of file   user group   size   date & time modified   date & time accessed last   file or directory name

-rw-rw-r--          1                divy          divy      26790    2020-11-03 16:48:02    2020-11-03 16:48:25         OUTPUT_1_IIT2019211.png
drwxrwxr-x          2                divy          divy       4096    2020-11-21 02:30:32    2020-11-21 02:30:36         ./
-rwxr-xr-x          1                divy          divy      16840    2020-11-03 17:03:09    2020-11-03 17:03:10         q
drwxr-xr-x          5                divy          divy      20480    2020-11-03 16:50:38    2020-11-20 18:29:12         ../
-rw-rw-r--          1                divy          divy      207718    2020-11-03 16:44:48    2020-11-03 16:49:07         Transition_Diagram.pdf
-rwxr-xr-x          1                divy          divy      13232    2020-11-20 20:50:21    2020-11-20 21:12:00         m
-rw-rw-r--          1                divy          divy      27572    2020-11-03 16:26:43    2020-11-03 16:46:22         OUTPUT_2_IIT2019211.png
-rw-rw-r--          1                divy          divy        792    2020-11-03 17:02:59    2020-11-03 17:03:07         q1.cpp
-rw-rw-r--          1                divy          divy      99649    2020-11-03 16:28:16    2020-11-03 16:46:49         OUTPUT_3_IIT2019211.png
-rw-rw-r--          1                divy          divy       2991    2020-11-03 16:22:56    2020-11-03 16:45:59         Question_1_IIT2019211.cpp
lrwxrwxrwx          1                divy          divy      16840    2020-11-03 17:03:09    2020-11-03 17:03:10         abc -> q
lrwxrwxrwx          1                divy          divy        792    2020-11-03 17:02:59    2020-11-03 17:03:07         abc1 -> q1.cpp
divy@anonymous:~/Downloads/question-1$ ./mini-ls /home/divy/Desktop/toc
OUTPUT_1_IIT2019211.png
./
q
../
Transition_Diagram.pdf
m
OUTPUT_2_IIT2019211.png
q1.cpp
OUTPUT_3_IIT2019211.png
Question_1_IIT2019211.cpp
abc
abc1
divy@anonymous:~/Downloads/question-1$
```

Activities Terminal Sat 02:32

divy@anonymous: ~/Downloads/question-1

File Edit View Search Terminal Help

```
divy@anonymous:~/Downloads/question-1$ ./mini-ls
./
mini-ls.cpp
../
m
mini-ls
readme.txt
divy@anonymous:~/Downloads/question-1$ ./mini-ls -L
permission    no. of hardlinks   owner of file   user group   size   date & time modified   date & time accessed last   file or directory name

drwxr-xr-x          2                divy          divy       4096    2020-11-21 02:30:59    2020-11-21 02:32:08         ./
-rw-r--r--          1                divy          divy       4207    2020-11-21 02:30:55    2020-11-21 02:30:58         mini-ls.cpp
drwxr-xr-x          6                divy          divy       4096    2020-11-20 20:50:20    2020-11-20 20:50:23         ../
-rwxr-xr-x          1                divy          divy      21872    2020-11-21 01:59:02    2020-11-21 01:59:08         m
-rwxr-xr-x          1                divy          divy      21872    2020-11-21 02:30:59    2020-11-21 02:31:01         mini-ls
-rw-r--r--          1                divy          divy        224    2020-11-21 02:06:07    2020-11-21 02:06:21         readme.txt
divy@anonymous:~/Downloads/question-1$
```

Activities Terminal Sat 02:32

divy@anonymous: ~/Downloads/question-1

File Edit View Search Terminal Help

```
divy@anonymous:~/Downloads/question-1$
```


THANK YOU