File Systems

1. **NAME**

dirent.h — format of directory entries

SYNOPSIS

#include <dirent.h>

DESCRIPTION

The internal format of directories is unspecified. The *<dirent.h>* header shall define the following type:

DIR A type representing a directory stream. The **DIR** type may be an incomplete type.
It shall also define the structure **dirent** which shall include the following members:

ino_t d_ino File serial number.
char d_name [] Filename string of entry.

The array *d_name* is of unspecified size, but shall contain a filename of at most {NAME_MAX} bytes followed by a terminating null byte.

```
struct dirent {
 ino_t        d_ino;                    /* inode number */
 off_t        d_off;                    /* offset to the next dirent */
 unsigned short d_reclen;               /* length of this record */
 unsigned char  d_type;                 /* type of file;
char      d_name[256]; /* filename */ (This we have used in our ls-l Program)
}
```

2. NAME

sys/stat.h - data returned by the stat() function

SYNOPSIS

#include <sys/stat.h>

Description

The <*sys/stat.h*> header shall define the structure of the data returned by the functions *fstat*(), *lstat*(), and *stat*().

The **stat** structure shall contain at least the following members:

- mode_t st_mode Mode of file (see below).
- nlink_t st_nlink Number of hard links to the file.
- uid_t st_uid User ID of file.
- gid_t st_gid Group ID of file.
- off_t st_size For regular files, the file size in bytes.
    - For symbolic links, the length in bytes of the
    - pathname contained in the symbolic link.
- time_t st_atime Time of last access.
- time_t st_mtime Time of last data modification.
- time_t st_ctime Time of last status change.
- blksize_t st_blksize A file system-specific preferred I/O block size for
    - this object. In some file system types, this may
    - vary from file to file.
- blkcnt_t st_blocks Number of blocks allocated for this object.

- The following macros shall be provided to test whether a file is of the specified type. The value *m* supplied to the macros is the value of *st_mode* from a **stat** structure. The macro shall evaluate to a non-zero value if the test is true; 0 if the test is false.
- **S_ISBLK(*m*)**
- **Test for a block special file.**
- **S_ISCHR(*m*)**
- **Test for a character special file.**
- **S_ISDIR(*m*)**
- **Test for a directory.**
- **S_ISFIFO(*m*)**
- **Test for a pipe or FIFO special file.**
- **S_ISREG(*m*)**
- **Test for a regular file.**
- **S_ISLNK(*m*)**
- **Test for a symbolic link.**
- **S_ISSOCK(*m*)**
- **Test for a socket.**

3. NAME

pwd.h - password structure

SYNOPSIS

#include <pwd.h>

DESCRIPTION

The <*pwd.h*> header provides a definition for **struct passwd**, which includes at least the following members:

char *pw_name user's login name

uid_t pw_uid numerical user ID

gid_t pw_gid numerical group ID

char *pw_dir initial working directory

char *pw_shell program to use as shell

The following are declared as functions and may also be defined as macros. Function prototypes must be provided for use with an ISO C compiler.

- struct passwd *getpwnam(const char *);
- struct passwd *getpwuid(uid_t);
- int getpwnam_r(const char *, struct passwd *, char *,
  - size_t, struct passwd **);
- int getpwuid_r(uid_t, struct passwd *, char *,
  - size_t, struct passwd **);
- void endpwent(void);
- struct passwd *getpwent(void);
-

4. NAME

grp.h - group structure

SYNOPSIS

#include <grp.h>

DESCRIPTION

The <*grp.h*> header declares the structure **group** which includes the following members:

- char *gr_name the name of the group
- gid_t gr_gid numerical group ID
- char **gr_mem pointer to a null-terminated array of character

The following are declared as functions and may also be defined as macros. Function prototypes must be provided for use with an ISO C compiler.

- struct group *getgrgid(gid_t);
- struct group *getgrnam(const char *);
- int getgrgid_r(gid_t, struct group *, char *,
- size_t, struct group **);
- int getgrnam_r(const char *, struct group *, char *,
- size_t , struct group **);
- struct group *getgrent(void);

5. NAME

time.h - time types

SYNOPSIS

#include <time.h>

The following are declared as functions and may also be defined as macros

- struct tm *localtime(const time_t *);
- struct tm *localtime_r(const time_t *, struct tm *);
- time_t mktime(struct tm *);
- int nanosleep(const struct timespec *, struct timespec *);
- size_t strftime(char *, size_t, const char *, const struct tm *);
- char *strptime(const char *, const char *, struct tm *);

An **Arrow operator in C/C++** allows to access elements in Structures and Unions. It is used with a pointer variable pointing to a structure or union. The arrow operator is formed by using a minus sign, followed by the greater than symbol as shown below.

Syntax:

```
(pointer_name)->(variable_name)
```

```
de->d_name
```

CREATE SYSTEM CALL

```
#include <fcntl.h>

int creat(char* filename, mode_t mode)
```

- The mode
  - is an octal number
    - Example: 0444 indicates that r access for USER, GROUP and ALL for the file.
  - If the file exists, the creat is ignored and prior content and rights are maintained.

Opening Files

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(char* filename, int flags, mode_t mode);
```

Flags: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_TRUNC, O_APPEND

- Mode: Specifies permission bits of the file
  - S_IRUSR, S_IWUSR, S_IXUSR – owner permission
  - S_IRGRP, S_IWGRP, S_IXGRP – group permission
  - S_IROTH, S_IWOTH, S_IXOTH – other permission

Reading/Writing Files

- Low level read and write
- #include <unistd.h>
- ssize_t read(int fd, void *buf, size_t n);

Returns num bytes read or -1

- ssize_t write(int fd, const void *buf, size_t n);
  - Returns num bytes written or -1

What about size_t and ssize_t

- size_t – unsigned int
- ssize_t - signed int
- How does this affect the range of values in each type?
  - with 32-bit int?

Reading file metadata

- How can we find information about a file
- #include <unistd.h>
- #include <sys/stat.h>

  int stat(const char* filename, struct stat *buf);
- int fstat(int fd, struct stat *buf);

What is struct stat?

```
struct stat
{
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t  st_blocks;  /* number of blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

Accessing File Status

stat(char* file, struct stat *buf);

fstat(int fd, struct stat *buf);

struct stat buf; // defines a struct stat to hold file information

stat("filename", &buf) ; // now the file information is placed in the buf

st_atime --- Last access time

st_mtime --- last modify time

st_ctime --- Last status change time

st_size --- total size of file

st_uid – user ID of owner

st_mode – file status (directory or not)

Example

```
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
struct stat statbuf;


char dirpath[256];
getcwd(dirpath,256);

DIR *dir = opendir(dirpath);
struct dirent *dp;


for (dp=readdir(dir); dp != NULL ; dp=readdir(dir)){
        stat(dp->d_name, &statbuf);

        printf("the file name is %s \n", dp->d_name);
        printf("dir = %d\n", S_ISDIR(statbuf.st_mode));
        printf("file size is %ld in bytes \n", statbuf.st_size);

        printf("last modified time is %ld in seconds \n", statbuf.st_mtime);
        printf("last access time is %ld in seconds \n", statbuf.st_atime);
        printf("The device containing the file is %d\n", statbuf.st_dev);
        printf("File serial number is %d\n\n", statbuf.st_ino);
```

How to determine a file type

- S_ISREG
  - A regular file?
- S_ISDIR
  - Is a directory?
  - printf("dir = %d\n", S_ISDIR(statbuf.st_mode));
- S_ISSOCK

A network socket

The Breakdown of LS-L Program

DIR *the directory

The directory: it's the folder we're browsing (we'll use an argument (argv) in order to identify it)

struct dirent *de
A dirent structure contains the character pointer d_name, which points to a string that gives the name of a file in the directory. This string ends in a terminating NULL, and has a maximum of NAME_MAX characters.

struct stat buf;
// defines a struct stat to hold file information
buf
(Output) A pointer to the area to which the information should be written.

struct passwd *p;
struct group *p;
Will be used to determine the file owner & group

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

If a file is found (readdir returns a NOT NULL value), the loop starts/keep going until it has listed all of them.

```
while( (de=readdir(d))!=NULL)
```

Then we use stat function in order to retrieve information about the file
stat(buf, &thestat);

Then we will find the file permissions using the loop and group name, user id and time using the respective system calls

User Name

```
p=getpwuid(buf.st_uid);
printf(" %.8s",p->pw_name);
```

Group Name

```
g=getgrgid(buf.st_gid);
printf(" %-8.8s",g->gr_name);
```

For Time

The C library function struct tm *localtime(const time_t *timer) uses the time pointed by timer to fill a tm structure with the values that represent the corresponding local time. The value of timer is broken up into the structure tm and expressed in the local time zone.

Declaration

Following is the declaration for localtime() function.

```
struct tm *localtime(const time_t *timer)
```

strftime() is a function in C which is used to format date and time. It comes under the header file time.h, which also contains a structure named struct tm which is used to hold the time and date. The syntax of strftime() is as shown below

```
size_t strftime(char *s, size_t max, const char *format,   const struct tm *tm);
```