DIRECTORY FILES API

Directory files in UNIX and POSIX systems are used to help users in organizing their files into some structure based on the specific use of file.

They are also used by the operating system to convert file path names to their inode numbers.

Directory files are created in BSD UNIX and POSIX.1 by mkdir API:

```
#include <sys/stat.h>
#include <unistd.h>
int mkdir ( const char* path_name, mode t mode );
```

1. The path_name argument is the path name of a directory to be created.
2. The mode argument specifies the access permission for the owner, group and others to be assigned to the file.
3. The return value of mkdir is 0 if it succeeds or -1 if it fails.

UNIX System V.3 uses the *mknod* API to create directory files.
UNIX System V.4 supports both the *mkdir* and *mknod APIs* for creating directory files.
The difference between the two *APIs is* that a directory created by ***mknod*** **does not contain the "."** and ".." links. On the other hand, a directory created by *mkdir* has the "." and ".." links created in one atomic operation, and it is ready to be used.
A directory file is a record-oriented file, where each record stores a file name and the mode number of a file that resides in that directory.

The following portable functions are defined for directory file browsing. These functions are defined in both the <dirent.h> and <sys/dir.h> headers.

```
#include <sys/types.h>
#if defined (BSD) && !_POSIX_SOURCE
#include <sys/dir.h>
 typedef struct direct Dirent;
#else
```

#include <dirent.h>

typedef struct dirent Dirent;

#endif

DIR* ***opendir*** (const char* path_name);

Dirent* *readdir* (DIR* dir_fdesc);

int *closedir* (DIR* dir_fdesc);

void ***rewinddir*** (DIR* dir_fdesc);

The uses of these functions are:

opendir: Opens a directory file for read-only. Returns a file handle DIR* for future reference of the file

readdir: Reads a record from a directory file referenced by *dir_fdesc* and returns that record information.

closedir: Closes a directory file referenced by *dir_fdesc*.

rewinddir: Resets the file pointer to the beginning of the directory file referenced by *dir_fdesc*. The next call to *readdir* will read the first record from the file.

UNIX systems support additional functions for random access of directory file records. These functions are not supported by POSIX.1:

telldir: Returns the file pointer of a given dir_fdesc.

seekdir: Changes the file pointer of a given dir_fdesc to a specified address.

Directory files are removed by the *rmdir* API. Its prototype is given below:

#include <unistd.h>

int *rmdir* (const char* path_name**);**

The following *list_dir.C* program illustrates uses of the *mkdir, opendir, readdir, closedir,* and rmdirAPIs:

#include <iostream.h>

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <string.h>

#include <sys/stat.h>

#if defined (BSD) && !_POSIX_SOURCE

```
#include <sys/dir.h>
  typedef struct direct Dirent;
#else
#include <dirent.h>
  typedef struct dirent Dirent;
#endif

int main (int argc, char* argv[ ])
 {
Dirent*  dp;
DIR*     dir_fdesc;
while (--argc > 0 ) {          /* do the following for each file */
if ( !(dir_fdesc = opendir( *++argv ) )) {
if (mkdir( *argv, S_ IRWXU|S_IRWXG|S_IRWXO) == -1 )
 perror( "opendir" );
continue;
}
/*scan each directory file twice*/
for (int i=0;i <2 ; i++ )  {
for ( int cnt=0; dp=readdir( dir_fdesc );) {
if (i) cout << dp->d_name << endl;
if (strcmp( dp->d_name, ".") && strcmp( dp->d_name, ".. ") )
 cnt++;                              /*count how many files in directory*/
if (!cnt) { rmdir( *argv ); break;) /* empty directory */
   rewinddir( dir fdesc );    / reset pointer for second round */
   }
   closedir( dir fdesc );
 }
}
```

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iv-unix-and-shell-programming-10cs44-notes.pdf