C Programming/POSIX Reference/dirent.h

dirent.h is the header in the C POSIX library for the C programming language that contains constructs that facilitate directory traversing. The function is not part of the C standard, but is considered "pseudo-standard" and is usually portable between platforms.

Contents

Functions

Name	Notes
`int` **`closedir`**`(DIR* dirp)`	Closes the directory stream referred to by `dirp`. Upon return, `dirp` 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 will be closed. Upon successful completion, closedir() returns 0. Otherwise, -1 is returned and errno is set to indicate the error. `errno` Errors: EBADF means `dirp` does not refer to an open directory stream, EINTR means the function was interrupted by a signal.
`DIR*` **`opendir`**`(const char* dirname)`	Opens a directory stream corresponding to the directory named by `dirname`. The directory stream is positioned at the first entry. If the type DIR is implemented using a file descriptor, applications will only be able to open up to a total of `OPEN_MAX` files and directories. Upon successful completion, opendir() returns a pointer to an object of type DIR. Otherwise, a null pointer is returned and errno is set to indicate the error. `errno` Errors: EACCES means the search permission is denied for the component of the path prefix of dirname or read permission is denied for dirname. ELOOP means too many symbolic links were encountered in resolving path. ENAMETOOLONG means the length of the dirname argument exceeds PATH_MAX, or a pathname component is longer than NAME_MAX. ENOENT means a component of dirname does not name an existing directory or dirname is an empty string. ENOTDIR means a component of dirname is not a directory. EMFILE means OPEN_MAX file descriptors are currently open in the calling process. ENAMETOOLONG means the pathname resolution of a symbolic link produced an intermediate result whose length exceeds PATH_MAX. ENFILE means there are too many files are currently open in the system.
`struct dirent*` **`readdir`**`(DIR* dirp)`	Returns a pointer to a structure representing the directory entry at the current position in the directory stream specified by the argument `dirp`, and positions the directory stream at the next entry. It returns a null pointer upon reaching the end of the directory stream. If entries for dot or dot-dot exist, one entry will be returned for dot and one entry will be returned for dot-dot; otherwise they will not be returned. When an error is encountered, a null pointer is returned and errno is set to indicate the error. When the end of the directory is encountered, a null pointer is returned and errno is not changed. The memory location pointed to by the return value is managed by the library and may change on subsequent calls to readdir. It should not be free'd by the user. `errno` Errors: EOVERFLOW means one of the values in the structure to be returned cannot be represented correctly. EBADF means `dirp` does not refer to an open directory stream. ENOENT means the current position of the directory stream is invalid.
`int` **`readdir_r`**`(DIR* dirp, struct dirent* entry, struct dirent** result)`	Initialises `entry` to represent the directory entry at the current position in `dirp`, store a pointer to this structure at the location referenced by `result`, and positions the directory stream at the next entry. The storage pointed to by `entry` will be large enough for a dirent with an array of char d_name member containing at least `NAME_MAX` plus one elements. On successful return, the pointer returned at *result will have the same value as the argument entry. Upon reaching the end of the directory stream, this pointer will have the value NULL. `errno` Errors: EBADF means `dirp` does not refer to an open directory stream.

`void` **`rewinddir`**`(DIR* dirp)`	Resets the position of the directory stream to which dirp refers to the beginning of the directory. It also causes the directory stream to refer to the current state of the corresponding directory, as a call to opendir() would have done. If dirp does not refer to a directory stream, the effect is undefined.
`void` **`seekdir`**`(DIR* dirp, long int loc)`	Sets the position of the next readdir() operation on the directory stream specified by dirp to the position specified by loc. The value of loc should have been returned from an earlier call to telldir(). The new position reverts to the one associated with the directory stream when telldir() was performed. If the value of loc was not obtained from an earlier call to telldir() or if a call to rewinddir() occurred between the call to telldir() and the call to seekdir(), the results of subsequent calls to readdir() are unspecified.
`long int` **`telldir`**`(DIR* dirp)`	Obtains the current location associated with the directory stream specified by dirp. If the most recent operation on the directory stream was a seekdir(), the directory position returned from the telldir() is the same as that supplied as a loc argument for seekdir(). Upon successful completion, telldir() returns the current location of the specified directory stream.

Member constants

Constants defined in the `stdio.h` header include:

Name	Notes
`NAME_MAX` (or `FILENAME_MAX`)	The maximum length of the char array `d_name`.

Member types

Data types defined in the `dirent.h` header include:

- `DIR` - A structure representing a directory stream. Its structure is not defined by POSIX, and is usually opaque to users.
- `struct dirent` - A structure with the following members:
  - `ino_t d_ino` - file serial number
  - `char d_name[]` - name of entry (will not exceed a size of NAME_MAX)
- In addition, `struct dirent` may contain the following members, depending on the platform:
  - `off_t d_off` - file offset
  - `unsigned short int d_reclen` - length of the dirent record
  - `unsigned short int d_namlen` - length of name
  - `unsigned int d_type` - type of file

Standardization

dirent.h is included in most C/C++ libraries for the PC architecture.

dirent.h is known to be included in the following compilers:

- Turbo C++ (DOS)

- GCC (Cross-platform)
- MinGW (a Microsoft Windows version of GCC)
- Borland C++ Builder (Microsoft Windows)

Microsoft Visual C++ does **not** include dirent.h

Example

A short example of dirent.h usage is:

```
/*****************************************************************
 * A simpler and shorter implementation of ls(1)
 * ls(1) is very similar to the DIR command on DOS and Windows.
 *****************************************************************/
#include <stdio.h>
#include <dirent.h>

int listdir(const char *path)
{
  struct dirent *entry;
  DIR *dp;

  dp = opendir(path);
  if (dp == NULL)
  {
    perror("opendir");
    return -1;
  }

  while((entry = readdir(dp)))
    puts(entry->d_name);

  closedir(dp);
  return 0;
}

int main(int argc, char **argv) {
  int counter = 1;

  if (argc == 1)
    listdir(".");

  while (++counter <= argc) {
    printf("\nListing %s...\n", argv[counter-1]);
    listdir(argv[counter-1]);
  }

  return 0;
}
```

Put the source in a file (listdir.c) and compile (in a Linux shell) like this:

```
gcc listdir.c -o listdir
```

or like this:

```
gcc listdir.c -o listdir.exe
```

in a Windows/DOS environment.

Now, to run (in a Linux shell) type:

```
./listdir
```

or type:

```
listdir.exe
```

in a Windows/DOS shell.

References

- Free Windows implementation of dirent.h (https://web.archive.org/web/20110807110703/http://www.softagalleria.net/dirent.php)
- OpenGroup specification for dirent.h (http://www.opengroup.org/onlinepubs/007908799/xsh/dirent.h.html)
- GNU dirent specifications (http://www.delorie.com/gnu/docs/dirent/)