Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'readdir.3p' command

$ man readdir.3p

READDIR(3P) POSIX Programmer's Manual READDIR(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

readdir, readdir_r ? read a directory

SYNOPSIS

```
#include <dirent.h>
struct dirent *readdir(DIR *dirp);
int readdir_r(DIR *restrict dirp, struct dirent *restrict entry,
    struct dirent **restrict result);
```

DESCRIPTION

The type DIR, which is defined in the <dirent.h> header, represents a directory stream, which is an ordered sequence of all the directory en? tries in a particular directory. Directory entries represent files; files may be removed from a directory or added to a directory asyn? chronously to the operation of readdir().

The readdir() function shall return a pointer to a structure represent? ing the directory entry at the current position in the directory stream specified by the argument dirp, and position the directory stream at the next entry. It shall return a null pointer upon reaching the end of

the directory stream. The structure dirent defined in the <dirent.h> header describes a directory entry. The value of the structure's d_ino member shall be set to the file serial number of the file named by the d_name member. If the d_name member names a symbolic link, the value of the d_ino member shall be set to the file serial number of the symbolic link itself.

The readdir() function shall not return directory entries containing empty names. If entries for dot or dot-dot exist, one entry shall be returned for dot and one entry shall be returned for dot-dot; otherwise, they shall not be returned.

The application shall not modify the structure to which the return value of readdir() points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to readdir() on the same directory stream. They shall not be affected by a call to readdir() on a different directory stream. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

If a file is removed from or added to the directory after the most recent call to opendir() or rewinddir(), whether a subsequent call to readdir() returns an entry for that file is unspecified.

The readdir() function may buffer several directory entries per actual read operation; readdir() shall mark for update the last data access timestamp of the directory each time the directory is actually read.

After a call to fork(), either the parent or child (but not both) may continue processing the directory stream using readdir(), rewinddir(), or seekdir(). If both the parent and child processes use these functions, the result is undefined.

The readdir() function need not be thread-safe.

Applications wishing to check for error situations should set errno to 0 before calling readdir(). If errno is set to non-zero on return, an error occurred.

The readdir_r() function shall initialize the dirent structure refer?
enced by entry to represent the directory entry at the current position
in the directory stream referred to by dirp, store a pointer to this
structure at the location referenced by result, and position the direc?
tory stream at the next entry.

The storage pointed to by entry shall be large enough for a dirent with
an array of char d_name members containing at least {NAME_MAX}+1 ele?
ments.

Upon successful return, the pointer returned at *result shall have the
same value as the argument entry. Upon reaching the end of the direc?
tory stream, this pointer shall have the value NULL.

The readdir_r() function shall not return directory entries containing
empty names.

If a file is removed from or added to the directory after the most re?
cent call to opendir() or rewinddir(), whether a subsequent call to
readdir_r() returns an entry for that file is unspecified.

The readdir_r() function may buffer several directory entries per ac?
tual read operation; readdir_r() shall mark for update the last data
access timestamp of the directory each time the directory is actually
read.

RETURN VALUE

Upon successful completion, readdir() shall return a pointer to an ob?
ject of type struct dirent. When an error is encountered, a null
pointer shall be returned and errno shall be set to indicate the error.

When the end of the directory is encountered, a null pointer shall be
returned and errno is not changed.

If successful, the readdir_r() function shall return zero; otherwise,
an error number shall be returned to indicate the error.

ERRORS

These functions shall fail if:

EOVERFLOW

One of the values in the structure to be returned cannot be rep?
resented correctly.

These functions may fail if:

EBADF The dirp argument does not refer to an open directory stream.

ENOENT The current position of the directory stream is invalid.

The following sections are informative.

EXAMPLES

The following sample program searches the current directory for each of the arguments supplied on the command line.

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>
static void lookup(const char *arg)
{
    DIR *dirp;
    struct dirent *dp;
    if ((dirp = opendir(".")) == NULL) {
        perror("couldn't open '.'");
        return;
    }
    do {
        errno = 0;
        if ((dp = readdir(dirp)) != NULL) {
            if (strcmp(dp->d_name, arg) != 0)
                continue;
            (void) printf("found %s\n", arg);
            (void) closedir(dirp);
                return;
        }
    } while (dp != NULL);
    if (errno != 0)
        perror("error reading directory");
    else
        (void) printf("failed to find %s\n", arg);
```

```
        (void) closedir(dirp);
        return;
    }
    int main(int argc, char *argv[])
    {
        int i;
        for (i = 1; i < argc; i++)
            lookup(argv[i]);
        return (0);
    }
```

APPLICATION USAGE

The readdir() function should be used in conjunction with opendir(), closedir(), and rewinddir() to examine the contents of the directory.

The readdir_r() function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

RATIONALE

The returned value of readdir() merely represents a directory entry. No equivalence should be inferred.

Historical implementations of readdir() obtain multiple directory en? tries on a single read operation, which permits subsequent readdir() operations to operate from the buffered information. Any wording that required each successful readdir() operation to mark the directory last data access timestamp for update would disallow such historical perfor? mance-oriented implementations.

When returning a directory entry for the root of a mounted file system, some historical implementations of readdir() returned the file serial number of the underlying mount point, rather than of the root of the mounted file system. This behavior is considered to be a bug, since the underlying file serial number has no significance to applications.

Since readdir() returns NULL when it detects an error and when the end of the directory is encountered, an application that needs to tell the difference must set errno to zero before the call and check it if NULL

is returned. Since the function must not change errno in the second case and must set it to a non-zero value in the first case, a zero er? rno after a call returning NULL indicates end-of-directory; otherwise, an error.

Routines to deal with this problem more directly were proposed:

```
int derror (dirp)
DIR *dirp;
void clearderr (dirp)
DIR *dirp;
```

The first would indicate whether an error had occurred, and the second would clear the error indication. The simpler method involving errno was adopted instead by requiring that readdir() not change errno when end-of-directory is encountered.

An error or signal indicating that a directory has changed while open was considered but rejected.

The thread-safe version of the directory reading function returns val? ues in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call. Either the {NAME_MAX} com? pile-time constant or the corresponding pathconf() option can be used to determine the maximum sizes of returned pathnames.

FUTURE DIRECTIONS

None.

SEE ALSO

closedir(), dirfd(), exec, fdopendir(), fstatat(), rewinddir(), sym? link()

The Base Definitions volume of POSIX.1?2017, <dirent.h>, <sys_types.h>

COPYRIGHT

The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .