```
**********************************************************
   14734 Sun Aug  2 20:08:23 2015
new/usr/src/man/man3c/ftw.3c
1438 ftw(3C) should mention 'quit' member of struct FTW and possible values
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2007, Sun Microsystems, Inc.  All Rights Reserved.
   3 .\" Copyright 1989 AT&T
   4 .\" Portions Copyright (c) 2001, the Institute of Electrical and Electronics Eng
   5 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   6 .\" http://www.opengroup.org/bookstore/.
   7 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   8 .\"  This notice shall appear on any product containing this material.
   9 .\" The contents of this file are subject to the terms of the Common Development
  10 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
  11 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
  12 .TH FTW 3C "Jan 30, 2007"
  13 .SH NAME
  14 ftw, nftw \- walk a file tree
  15 .SH SYNOPSIS
  16 .LP
  17 .nf
  18 #include <ftw.h>

  20 \fBint\fR \fBftw\fR(\fBconst char *\fR\fIpath\fR, \fBint\fR (*\fIfn\fR) (\fBcons
  21      \fBconst struct stat *\fR, \fBint\fR), \fBint\fR \fIdepth\fR);
  22 .fi

  24 .LP
  25 .nf
  26 \fBint\fR \fBnftw\fR(\fBconst char *\fR\fIpath\fR, \fBint (*\fR\fIfn\fR) (\fBcon
  27      \fBconst struct stat *\fR, \fBint\fR, \fBstruct FTW *\fR), \fBint\fR \fIdep
  28      \fBint\fR \fIflags\fR);
  29 .fi

  31 .SH DESCRIPTION
  32 .sp
  33 .LP
  34 The \fBftw()\fR function recursively descends the directory hierarchy rooted in
  35 \fIpath\fR. For each object in the hierarchy, \fBftw()\fR calls the
  36 user-defined function \fIfn\fR, passing it a pointer to a null-terminated
  37 character string containing the name of the object, a pointer to a \fBstat\fR
  38 structure (see \fBstat\fR(2)) containing information about the object, and an
  39 integer. Possible values of the integer, defined in the <\fBftw.h\fR> header,
  40 are:
  41 .sp
  42 .ne 2
  43 .na
  44 \fB\fBFTW_F\fR\fR
  45 .ad
  46 .RS 11n
  47 The object is a file.
  48 .RE

  50 .sp
  51 .ne 2
  52 .na
  53 \fB\fBFTW_D\fR\fR
  54 .ad
  55 .RS 11n
  56 The object is a directory.
  57 .RE

  59 .sp
  60 .ne 2
  61 .na
```



```
  62 \fB\fBFTW_DNR\fR\fR
  63 .ad
  64 .RS 11n
  65 The object is a directory that cannot be read. Descendants of the directory are
  66 not processed.
  67 .RE

  69 .sp
  70 .ne 2
  71 .na
  72 \fB\fBFTW_NS\fR\fR
  73 .ad
  74 .RS 11n
  75 The \fBstat()\fR function failed on the object because of lack of appropriate
  76 permission or the object is a symbolic link that points to a non-existent file.
  77 The \fBstat\fR buffer passed to \fIfn\fR is undefined.
  78 .RE

  80 .sp
  81 .LP
  82 The \fBftw()\fR function visits a directory before visiting any of its
  83 descendants.
  84 .sp
  85 .LP
  86 The tree traversal continues until the tree is exhausted, an invocation of
  87 \fIfn\fR returns a non-zero value, or some error is detected within \fBftw()\fR
  88 (such as an I/O error). If the tree is exhausted, \fBftw()\fR returns \fB0\fR.
  89 If \fIfn\fR returns a non-zero value, \fBftw()\fR stops its tree traversal and
  90 returns whatever value was returned by \fIfn\fR.
  91 .sp
  92 .LP
  93 The  \fBnftw()\fR function is similar to \fBftw()\fR except that it takes the
  94 additional argument \fIflags\fR, which is a bitwise-inclusive OR of zero or
  95 more of the following flags:
  96 .sp
  97 .ne 2
  98 .na
  99 \fB\fBFTW_CHDIR\fR\fR
 100 .ad
 101 .RS 13n
 102 If set, \fBnftw()\fR changes the current working directory to each directory as
 103 it reports files in that directory. If clear, \fBnftw()\fR does not change the
 104 current working directory.
 105 .RE

 107 .sp
 108 .ne 2
 109 .na
 110 \fB\fBFTW_DEPTH\fR\fR
 111 .ad
 112 .RS 13n
 113 If set, \fBnftw()\fR reports all files in a directory before reporting the
 114 directory itself. If clear, \fBnftw()\fR reports any directory before reporting
 115 the files in that directory.
 116 .RE

 118 .sp
 119 .ne 2
 120 .na
 121 \fB\fBFTW_MOUNT\fR\fR
 122 .ad
 123 .RS 13n
 124 If set, \fBnftw()\fR reports only files in the same file system as path. If
 125 clear, \fBnftw()\fR reports all files encountered during the walk.
 126 .RE
```

```
.sp
.ne 2
.na
\fB\fBFTW_PHYS\fR\fR
.ad
.RS 13n
If set, \fBnftw()\fR performs a physical walk and does not follow symbolic
links.
.RE

.sp
.LP
If \fBFTW_PHYS\fR is clear and \fBFTW_DEPTH\fR is set, \fBnftw()\fR follows
links instead of reporting them, but does not report any directory that would
be a descendant of itself. If \fBFTW_PHYS\fR is clear and \fBFTW_DEPTH\fR is
clear, \fBnftw()\fR follows links instead of reporting them, but does not
report the contents of any directory that would be a descendant of itself.
.sp
.LP
At each file it encounters, \fBnftw()\fR calls the user-supplied function
\fIfn\fR with four arguments:
.RS +4
.TP
.ie t \(bu
.el o
The first argument is the pathname of the object.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The second argument is a pointer to the \fBstat\fR buffer containing
information on the object.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The third argument is an integer giving additional information. Its value is
one of the following:
.RS

.sp
.ne 2
.na
\fB\fBFTW_F\fR\fR
.ad
.RS 11n
The object is a file.
.RE

.sp
.ne 2
.na
\fB\fBFTW_D\fR\fR
.ad
.RS 11n
The object is a directory.
.RE

.sp
.ne 2
.na
\fB\fBFTW_DP\fR\fR
.ad
.RS 11n
```



```
The object is a directory and subdirectories have been visited. (This condition
only occurs if the FTW_DEPTH flag is included in flags.)
.RE

.sp
.ne 2
.na
\fB\fBFTW_SL\fR\fR
.ad
.RS 11n
The object is a symbolic link. (This condition only occurs if the FTW_PHYS flag
is included in flags.)
.RE

.sp
.ne 2
.na
\fB\fBFTW_SLN\fR\fR
.ad
.RS 11n
The object is a symbolic link that points to a non-existent file. (This
condition only occurs if the FTW_PHYS flag is not included in flags.)
.RE

.sp
.ne 2
.na
\fB\fBFTW_DNR\fR\fR
.ad
.RS 11n
The object is a directory that cannot be read.  The user-defined function
\fIfn\fR will not be called for any of its descendants.
.RE

.sp
.ne 2
.na
\fB\fBFTW_NS\fR\fR
.ad
.RS 11n
The \fBstat()\fR function failed on the object because of lack of appropriate
permission. The stat buffer passed to \fIfn\fR is undefined.  Failure of
\fBstat()\fR for any other reason is considered an error and \fBnftw()\fR
returns \(mi1.
.RE

.RE

.RE
.RS +4
.TP
.ie t \(bu
.el o
The fourth argument is a pointer to an \fBFTW\fR structure that contains the
following members:
.sp
.in +2
.nf
int   quit;
int   base;
int   level;
.fi
.in -2

The \fBquit\fR member has a default value of \fI0\fR, but can be set to the
following values:
```

```
.RS

.sp
.ne 2
.na
\fB\fBFTW_SKIP\fR\fR or \fB\fBFTW_PRUNE\fR\fR
.ad
.RS 11n
This object and its descendants are pruned from the search.
.RE

.sp
.ne 2
.na
\fB\fBFTW_FOLLOW\fR\fR
.ad
.RS 11n
If this object is a symbolic link, follow the link to its physical counterpart.
.RE

.sp
```

The \fBbase\fR member is the offset of the object’s filename in the pathname passed as the first argument to \fIfn\fR(). The value of \fBlevel\fR indicates the depth relative to the root of the walk, where the root level is 0.

.sp

The results are unspecified if the application-supplied \fIfn\fR() function does not preserve the current working directory.

.RE

.sp

.LP

Both \fBftw()\fR and \fBnftw()\fR use one file descriptor for each level in the tree. The \fIdepth\fR argument limits the number of file descriptors used. If \fIdepth\fR is zero or negative, the effect is the same as if it were 1. It must not be greater than the number of file descriptors currently available for use. The \fBftw()\fR function runs faster if \fIdepth\fR is at least as large as the number of levels in the tree. Both \fBftw()\fR and \fBnftw()\fR are able to descend to arbitrary depths in a file hierarchy and do not fail due to path length limitations unless either the length of the path name pointed to by the \fIpath\fR argument exceeds {\fBPATH_MAX\fR} requirements, or for \fBftw()\fR, the specified depth is less than 2, or for \fBnftw()\fR, the specified depth is less than 2 and \fBFTW_CHDIR\fR is not set. When \fBftw()\fR and \fBnftw()\fR return, they close any file descriptors they have opened; they do not close any file descriptors that might have been opened by \fIfn\fR.

.SH RETURN VALUES

.sp

.LP

If the tree is exhausted, \fBftw()\fR and \fBnftw()\fR return \fB0\fR. If the function pointed to by \fIfn\fR returns a non-zero value, \fBftw()\fR and \fBnftw()\fR stop their tree traversal and return whatever value was returned by the function pointed to by \fIfn\fR. If \fBftw()\fR and \fBnftw()\fR detect an error, they return \fB\(mi1\fR and set \fBerrno\fR to indicate the error.

.sp

.LP

If \fBftw()\fR and \fBnftw()\fR encounter an error other than \fBEACCES\fR (see \fBFTW_DNR\fR and \fBFTW_NS\fR above), they return \fB\(mi1\fR and set \fBerrno\fR to indicate the error. The external variable \fBerrno\fR can contain any error value that is possible when a directory is opened or when one of the \fBstat\fR functions is executed on a directory or file.

.SH ERRORS

.sp

.LP

The \fBftw()\fR and \fBnftw()\fR functions will fail if:

.sp

.ne 2

.na

\fB\fBELOOP\fR\fR

.ad

.RS 16n

A loop exists in symbolic links encountered during resolution of the \fIpath\fR argument

.RE

.sp

.ne 2

.na

\fB\fBENAMETOOLONG\fR\fR

.ad

.RS 16n

The length of the path name pointed to by the \fIpath\fR argument exceeds {\fBPATH_MAX\fR}, or a path name component is longer than {\fBNAME_MAX\fR}.

.RE

.sp

.ne 2

.na

\fB\fBENOENT\fR\fR

.ad

.RS 16n

A component of \fIpath\fR does not name an existing file or \fIpath\fR is an empty string.

.RE

.sp

.ne 2

.na

\fB\fBENOTDIR\fR\fR

.ad

.RS 16n

A component of \fIpath\fR is not a directory.

.RE

.sp

.ne 2

.na

\fB\fBEOVERFLOW\fR\fR

.ad

.RS 16n

A field in the \fBstat\fR structure cannot be represented correctly in the current programming environment for one or more files found in the file hierarchy.

.RE

.sp

.LP

The \fBftw()\fR function will fail if:

.sp

.ne 2

.na

\fB\fBEACCES\fR\fR

.ad

.RS 16n

Search permission is denied for any component of \fIpath\fR or read permission is denied for \fIpath\fR.

.RE

.sp

.ne 2

.na

\fB\fBENAMETOOLONG\fR\fR

.ad

.RS 16n

The \fBftw()\fR function has descended to a path that exceeds {\fBPATH_MAX\fR}

```
392 and the depth argument specified by the application is less than 2 and
393 \fBFTW_CHDIR\fR is not set.
394 .RE

396 .sp
397 .LP
398 The \fBnftw()\fR function will fail if:
399 .sp
400 .ne 2
401 .na
402 \fB\fBEACCES\fR\fR
403 .ad
404 .RS 10n
405 Search permission is denied for any component of \fIpath\fR or read permission
406 is denied for \fIpath\fR, or \fIfn\fR() returns \(mi1 and does not reset
407 \fBerrno\fR.
408 .RE

410 .sp
411 .LP
412 The \fBnftw()\fR and \fBftw()\fR functions may fail if:
413 .sp
414 .ne 2
415 .na
416 \fB\fBELOOP\fR\fR
417 .ad
418 .RS 16n
419 Too many symbolic links were encountered during resolution of the \fIpath\fR
420 argument.
421 .RE

423 .sp
424 .ne 2
425 .na
426 \fB\fBENAMETOOLONG\fR\fR
427 .ad
428 .RS 16n
429 Pathname resolution of a symbolic link in the path name pointed to by the
430 \fIpath\fR argument produced an intermediate result whose length exceeds
431 {\fBPATH_MAX\fR}.
432 .RE

434 .sp
435 .LP
436 The \fBftw()\fR function may fail if:
437 .sp
438 .ne 2
439 .na
440 \fB\fBEINVAL\fR\fR
441 .ad
442 .RS 10n
443 The value of the \fIdepth\fR argument is invalid.
444 .RE

446 .sp
447 .LP
448 The \fBnftw()\fR function may fail if:
449 .sp
450 .ne 2
451 .na
452 \fB\fBEMFILE\fR\fR
453 .ad
454 .RS 10n
455 There are {\fBOPEN_MAX\fR} file descriptors currently open in the calling
456 process.
457 .RE
```



```
459 .sp
460 .ne 2
461 .na
462 \fB\fBENFILE\fR\fR
463 .ad
464 .RS 10n
465 Too many files are currently open in the system.
466 .RE

468 .sp
469 .LP
470 If the function pointed to by \fIfn\fR encounters system errors, \fBerrno\fR
471 may be set accordingly.
472 .SH EXAMPLES
473 .LP
474 \fBExample 1 \fRWalk a directory structure using \fBftw()\fR.
475 .sp
476 .LP
477 The following example walks the current directory structure, calling the
478 \fIfn\fR() function for every directory entry, using at most 10 file
479 descriptors:

481 .sp
482 .in +2
483 .nf
484 #include <ftw.h>
485 \&...
486 if (ftw(".", fn, 10) != 0) {
487         perror("ftw"); exit(2);
488 }
```

_____***unchanged_portion_omitted_***