NAME

filefuncs − provide some file related functionality to gawk

SYNOPSIS

```
@load "filefuncs"

result = chdir("/some/directory")

result = stat("/some/path", statdata [, follow])

flags = or(FTS_PHYSICAL, ...)
result = fts(pathlist, flags, filedata)
```

DESCRIPTION

The *filefuncs* extension adds several functions that provide access to file-related facilities.

chdir()

The **chdir()** function is a direct hook to the *chdir*(2) system call to change the current directory. It returns zero upon success or less than zero upon error. In the latter case it updates **ERRNO**.

stat()

The **stat()** function provides a hook into the *stat*(2) system call. It returns zero upon success or less than zero upon error. In the latter case it updates **ERRNO**. By default, it uses *lstat*(2). However, if passed a third argument, it uses *stat*(2), instead.

In all cases, it clears the **statdata** array. When the call is successful, **stat()** fills the **statdata** array with information retrieved from the filesystem, as follows:

statdata["name"]
The name of the file.

statdata["dev"]
Corresponds to the *st_dev* field in the *struct stat*.

statdata["ino"]
Corresponds to the *st_ino* field in the *struct stat*.

statdata["mode"]
Corresponds to the *st_mode* field in the *struct stat*.

statdata["nlink"]
Corresponds to the *st_nlink* field in the *struct stat*.

statdata["uid"]
Corresponds to the *st_uid* field in the *struct stat*.

statdata["gid"]
Corresponds to the *st_gid* field in the *struct stat*.

statdata["size"]
Corresponds to the *st_size* field in the *struct stat*.

statdata["atime"]
Corresponds to the *st_atime* field in the *struct stat*.

statdata["mtime"]
Corresponds to the *st_mtime* field in the *struct stat*.

statdata["ctime"]
Corresponds to the *st_ctime* field in the *struct stat*.

statdata["rdev"]
Corresponds to the *st_rdev* field in the *struct stat*. This element is only present for device files.

statdata["major"]
Corresponds to the *st_major* field in the *struct stat*. This element is only present for device files.

statdata["minor"]
Corresponds to the *st_minor* field in the *struct stat*. This element is only present for device files.

statdata["blksize"]
Corresponds to the *st_blksize* field in the *struct stat*, if this field is present on your system. (It is present on all modern systems that we know of.)

statdata["pmode"]
A human-readable version of the mode value, such as printed by *ls*(1). For example, **"-rwxr-xr-x"**.

statdata["linkval"]
If the named file is a symbolic link, this element will exist and its value is the value of the symbolic link (where the symbolic link points to).

statdata["type"]
The type of the file as a string. One of **"file"**, **"blockdev"**, **"chardev"**, **"directory"**, **"socket"**, **"fifo"**, **"symlink"**, **"door"**, or **"unknown"**. Not all systems support all file types.

fts()
The **fts()** function provides a hook to the *fts*(3) set of routines for traversing file hierarchies. Instead of returning data about one file at a time in a stream, it fills in a multi-dimensional array with data about each file and directory encountered in the requested hierarchies.

The arguments are as follows:

pathlist
An array of filenames. The element values are used; the index values are ignored.

flags This should be the bitwise OR of one or more of the following predefined flag values. At least one of **FTS_LOGICAL** or **FTS_PHYSICAL** must be provided; otherwise **fts()** returns an error value and sets **ERRNO**.

FTS_LOGICAL
Do a "logical" file traversal, where the information returned for a symbolic link refers to the linked-to file, and not to the symbolic link itself. This flag is mutually exclusive with **FTS_PHYSICAL**.

FTS_PHYSICAL
Do a "physical" file traversal, where the information returned for a symbolic link refers to the symbolic link itself. This flag is mutually exclusive with **FTS_LOGICAL**.

FTS_NOCHDIR
As a performance optimization, the *fts*(3) routines change directory as they traverse a file hierarchy. This flag disables that optimization.

FTS_COMFOLLOW
Immediately follow a symbolic link named in **pathlist**, whether or not **FTS_LOGICAL** is set.

FTS_SEEDOT
By default, the *fts*(3) routines do not return entries for "." and "..". This option causes entries for ".." to also be included. (The AWK extension always includes an entry for ".", see below.)

FTS_XDEV
During a traversal, do not cross onto a different mounted filesystem.

filedata
The **filedata** array is first cleared. Then, **fts()** creates an element in **filedata** for every element in **pathlist**. The index is the name of the directory or file given in **pathlist**. The element for this

index is itself an array. There are two cases.

The path is a file.
In this case, the array contains two or three elements:

"path" The full path to this file, starting from the ‘‘root’’ that was given in the **pathlist** array.

"stat" This element is itself an array, containing the same information as provided by the **stat()** function described earlier for its **statdata** argument. The element may not be present if *stat*(2) for the file failed.

"error"
If some kind of error was encountered, the array will also contain an element named **"error"**, which is a string describing the error.

The path is a directory.
In this case, the array contains one element for each entry in the directory. If an entry is a file, that element is as for files, just described. If the entry is a directory, that element is (recursively), an array describing the subdirectory. If **FTS_SEEDOT** was provided in the flags, then there will also be an element named **".."**. This element will be an array containing the data as provided by **stat()**.

In addition, there will be an element whose index is **"."**. This element is an array containing the same two or three elements as for a file: **"path"**, **"stat"**, and **"error"**.

The **fts()** function returns 0 if there were no errors. Otherwise it returns −1.

NOTES

The AWK **fts()** extension does not exactly mimic the interface of the *fts*(3) routines, choosing instead to provide an interface that is based on associative arrays, which should be more comfortable to use from an AWK program. This includes the lack of a comparison function, since *gawk* already provides powerful array sorting facilities. While an *fts_read()*–like interface could have been provided, this felt less natural than simply creating a multi-dimensional array to represent the file hierarchy and its information.

Nothing prevents AWK code from changing the predefined **FTS_***xx* values, but doing so may cause strange results when the changed values are passed to **fts()**.

BUGS

There are many more file-related functions for which AWK interfaces would be desirable.

EXAMPLE

See **test/fts.awk** in the *gawk* distribution for an example.

SEE ALSO

GAWK: Effective AWK Programming, *fnmatch*(3am), *fork*(3am), *inplace*(3am), *ordchr*(3am), *readdir*(3am), *readfile*(3am), *revoutput*(3am), *rwarray*(3am), *time*(3am).

chdir(2), *fts*(3), *stat*(2).

AUTHOR

Arnold Robbins, **arnold@skeeve.com**.

COPYING PERMISSIONS