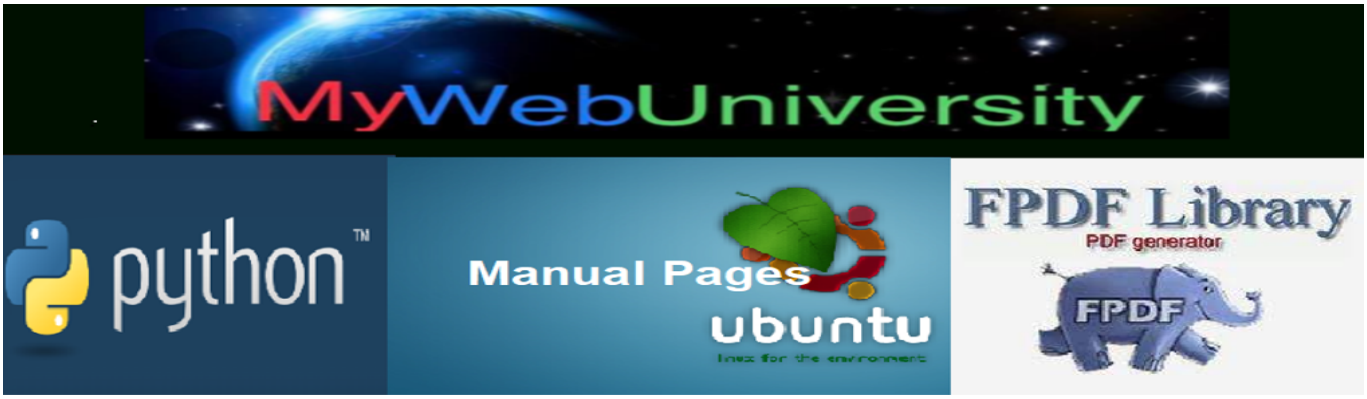


**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

## *Rocky Enterprise Linux 9.2 Manual Pages on command 'APR::Finfo.3pm'*

***$ man APR::Finfo.3pm***

libapache2-mod-perl2-2.0.12::doUseraContrilibapache2-mod-perl2-2.0.12::docs::api::APR::Finfo(3pm)

NAME

APR::Finfo - Perl API for APR fileinfo structure

Synopsis

```
use APR::Finfo ();
use APR::Const -compile => qw(FINFO_NORM);
my $finfo = APR::Finfo::stat("/tmp/test", APR::Const::FINFO_NORM, $pool);
$device = $finfo->device;     # (stat $file)[0]
$inode  = $finfo->inode;      # (stat $file)[1]
# stat returns an octal number while protection is hex
$prot   = $finfo->protection; # (stat $file)[2]
$nlink  = $finfo->nlink;      # (stat $file)[3]
$gid    = $finfo->group;      # (stat $file)[4]
$uid    = $finfo->user;       # (stat $file)[5]
$size   = $finfo->size;       # (stat $file)[7]
$atime  = $finfo->atime;      # (stat $file)[8]
$mtime  = $finfo->mtime;      # (stat $file)[9]
$ctime  = $finfo->ctime;      # (stat $file)[10]
$csize = $finfo->csize; # consumed size: not portable!
$filetype = $finfo->filetype; # file/dir/socket/etc
$fname = $finfo->fname;
$name  = $finfo->name;  # in filesystem case:
# valid fields that can be queried
```

```
$valid = $finfo->valid;
```

## Description

APR fileinfo structure provides somewhat similar information to Perl's "stat()" call, but you will want to use this module's API to query an already "stat()'ed" filehandle to avoid an extra system call or to query attributes specific to APR file handles.

During the HTTP request handlers coming after "PerlMapToStorageHandler", "$r->finfo" already contains the cached values from the apr's "stat()" call. So you don't want to perform it again, but instead get the "ARP::Finfo" object via:

```
my $finfo = $r->finfo;
```

## API

"APR::Finfo" provides the following functions and/or methods:

### "atime"

Get the time the file was last accessed:

```
$atime = $finfo->atime;
```

obj: $finfo ( "APR::Finfo object" )

return: $atime ( integer )

Last access time in seconds since the epoch

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[8]
```

Note that this method may not be reliable on all platforms, most notably Win32 -- FAT32 filesystems appear to work properly, but NTFS filesystems do not.

### "csize"

Get the storage size consumed by the file

```
$csize = $finfo->csize;
```

obj: $finfo ( "APR::Finfo object" )

return: $csize ( integer )

since: 2.0.00

Chances are that you don't want to use this method, since its functionality is not supported on most platforms (in which case it always returns 0).

### "ctime"

Get the time the file was last changed

```
$ctime = $finfo->ctime;
```

obj: $finfo ( "APR::Finfo object" )

return: $ctime ( integer )

Inode change time in seconds since the epoch

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[10]
```

The ctime field is non-portable. In particular, you cannot expect it to be a "creation time", see "Files and Filesystems" in the perlport manpage for details.

"device"

Get the id of the device the file is on.

```
$device = $finfo->device;
```

obj: $finfo ( "APR::Finfo object" )

return: $device ( integer )

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[0]
```

Note that this method is non-portable. It doesn't work on all platforms, most notably Win32.

"filetype"

Get the type of file.

```
$filetype = $finfo->filetype;
```

obj: $finfo ( "APR::Finfo object" )

return: $filetype ( ":filetype constant" )

since: 2.0.00

For example:

```
use APR::Pool;
use APR::Finfo;
use APR::Const -compile => qw(FILETYPE_DIR FILETYPE_REG FINFO_NORM);
my $pool  = APR::Pool->new();
my $finfo = APR::Finfo::stat("/tmp", APR::Const::FINFO_NORM, $pool);
my $finfo = $finfo->filetype;
if ($finfo == APR::Const::FILETYPE_REG) {
    print "regular file";
```

```
    }
    elsif ($finfo == APR::Const::FILETYPE_REG) {
        print "directory";
    }
    else {
        print "other file";
    }
```

Since /tmp is a directory, this will print:

```
  directory
```

"fname"

Get the pathname of the file (possibly unrooted)

```
  $fname = $finfo->fname;
```

obj: $finfo ( "APR::Finfo object" )

return: $filetype ( string )

since: 2.0.00

"group"

Get the group id that owns the file:

```
  $gid = $finfo->group;
```

obj: $finfo ( "APR::Finfo object" )

return: $gid ( number )

since: 2.0.00

This method returns the same value as Perl's:

```
  (stat $filename)[5]
```

Note that this method may not be meaningful on all platforms, most notably Win32.

Incorrect results have also been reported on some versions of OSX.

"inode"

Get the inode of the file.

```
  $inode = $finfo->inode;
```

obj: $finfo ( "APR::Finfo object" )

return: $inode ( integer )

since: 2.0.00

This method returns the same value as Perl's:

```
  (stat $filename)[1]
```

Note that this method may not be meaningful on all platforms, most notably Win32.

"mtime"

The time the file was last modified

```
$mtime = $finfo->mtime;
```

obj: $finfo ( "APR::Finfo object" )

return: $mtime ( integer )

Last modify time in seconds since the epoch

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[9]
```

"name"

Get the file's name (no path) in filesystem case:

```
$name = $finfo->name;
```

obj: $finfo ( "APR::Finfo object" )

return: $device ( string )

since: 2.0.00

"nlink"

Get the number of hard links to the file.

```
$nlink = $finfo->nlink;
```

obj: $finfo ( "APR::Finfo object" )

return: $nlink ( integer )

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[3]
```

"protection"

Get the access permissions of the file.  Mimics Unix access rights.

```
$prot = $finfo->protection;
```

obj: $finfo ( "APR::Finfo object" )

return: $prot ( ":fprot constant" )

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[2]
```

Note: Perl's stat returns an octal number while mod_perl's "protection" returns a hex

number.

See perldoc -f stat and APR's file_io for more information on each.

"size"

Get the size of the file

```
$size = $finfo->size;
```

obj: $finfo ( "APR::Finfo object" )

return: $size ( integer )

Total size of file, in bytes

since: 2.0.00

This method returns the same value as Perl's:

```
(stat $filename)[7]
```

"stat"

Get the specified file's stats.

```
$finfo = APR::Finfo::stat($fname, $wanted_fields, $p);
```

arg1: $fname ( string )

The path to the file to "stat()".

arg2: $wanted_fields ( ":finfo constant" )

The desired fields, as a bitmask flag of "APR::FINFO_*" constants.

Notice that you can also use the constants that already combine several elements in one. For example "APR::Const::FINFO_PROT" asks for all protection bits, "APR::Const::FINFO_MIN" asks for the following fields: type, mtime, ctime, atime, size and "APR::Const::FINFO_NORM" asks for all atomic unix "apr_stat()" fields (similar to perl's "stat()").

arg3: $p ( "APR::Pool object" )

the pool to use to allocate the file stat structure.

ret: $finfo ( "APR::Finfo object" )

since: 2.0.00

For example, here is how to get most of the "stat" fields:

```
use APR::Pool ();
use APR::Finfo ();
use APR::Const -compile => qw(FINFO_NORM);
my $pool = APR::Pool->new();
my $finfo = APR::Finfo::stat("/tmp/test", APR::Const::FINFO_NORM, $pool);
```

"user"

Get the user id that owns the file:

$uid = $finfo->user;

obj: $finfo ( "APR::Finfo object" )

return: $uid ( number )

since: 2.0.00

This method returns the same value as Perl's:

(stat $filename)[4]

Note that this method may not be meaningful on all platforms, most notably Win32.

"valid"

The bitmask describing valid fields of this apr_finfo_t structure including all available

'wanted' fields and potentially more

$valid = $finfo->valid;

obj: $finfo ( "APR::Finfo object" )

arg1: $valid ( bitmask )

This bitmask flag should be bit-OR'ed against ":finfo constant" constants.

since: 2.0.00

## See Also

mod_perl 2.0 documentation.

## Copyright



## Authors

The mod_perl development team and numerous contributors.