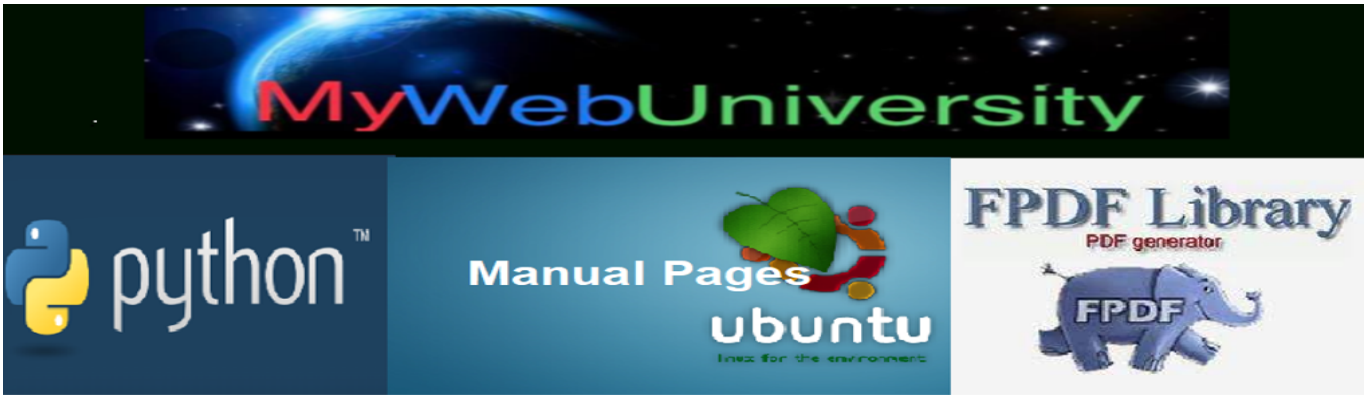

Rocky Enterprise Linux 9.2 Manual Pages on command 'filetest.3perl'

$ man filetest.3perl

filetest(3perl) Perl Programmers Reference Guide filetest(3perl)

NAME

filetest - Perl pragma to control the filetest permission operators

SYNOPSIS

```
$can_perhaps_read = -r "file";      # use the mode bits
{
    use filetest 'access';          # intuit harder
    $can_really_read = -r "file";
}
$can_perhaps_read = -r "file";      # use the mode bits again
```

DESCRIPTION

This pragma tells the compiler to change the behaviour of the filetest permission operators, "-r" "-w" "-x" "-R" "-W" "-X" (see perlfunc).

The default behaviour of file test operators is to use the simple mode bits as returned by the stat() family of system calls. However, many operating systems have additional features to define more complex access rights, for example ACLs (Access Control Lists). For such environments, "use filetest" may help the permission operators to return results more consistent with other tools.

The "use filetest" or "no filetest" statements affect file tests defined in their block, up to the end of the closest enclosing block (they are lexically block-scoped).

Currently, only the "access" sub-pragma is implemented. It enables (or disables) the use of access() when available, that is, on most UNIX systems and other POSIX environments. See details below.

Consider this carefully

The stat() mode bits are probably right for most of the files and directories found on your system, because few people want to use the additional features offered by access(). But you may encounter surprises if your program runs on a system that uses ACLs, since the stat() information won't reflect the actual permissions.

There may be a slight performance decrease in the filetest operations when the filetest pragma is in effect, because checking bits is very cheap.

Also, note that using the file tests for security purposes is a lost cause from the start: there is a window open for race conditions (who is to say that the permissions will not change between the test and the real operation?). Therefore if you are serious about security, just try the real operation and test for its success - think in terms of atomic operations. Filetests are more useful for filesystem administrative tasks, when you have no need for the content of the elements on disk.

The "access" sub-pragma

UNIX and POSIX systems provide an abstract access() operating system call, which should be used to query the read, write, and execute rights. This function hides various distinct approaches in additional operating system specific security features, like Access Control Lists (ACLs)

The extended filetest functionality is used by Perl only when the argument of the operators is a filename, not when it is a filehandle.

Limitation with regard to "_"

Because access() does not invoke stat() (at least not in a way visible to Perl), the stat result cache "_" is not set. This means that the outcome of the following two tests is different. The first has the stat bits of /etc/passwd in "_", and in the second case this still contains the bits of "/etc".

```
{ -d '/etc';
  -w '/etc/passwd';
  print -f _ ? 'Yes' : 'No';   # Yes
}
{ use filetest 'access';
  -d '/etc';
  -w '/etc/passwd';
  print -f _ ? 'Yes' : 'No';   # No
```

```
}
```

Of course, unless your OS does not implement access(), in which case the pragma is simply ignored. Best not to use "_" at all in a file where the filetest pragma is active!

As a side effect, as "_" doesn't work, stacked filetest operators ("-f -w $file") won't work either.

This limitation might be removed in a future version of perl.