NAME

pathchk – check path names

SYNOPSIS

pathchk [−**p**] *path* . . .

AVAILABILITY

SUNWcsu

DESCRIPTION

The **pathchk** command will check that one or more path names are valid (that is, they could be used to access or create a file without causing syntax errors) and portable (that is, no filename truncation will result). More extensive portability checks are provided by the −**p** option.

By default, **pathchk** will check each component of each *path* operand based on the underlying file system. A diagnostic will be written for each *path* operand that:

- is longer than **PATH_MAX** bytes.
- contains any component longer than **NAME_MAX** bytes in its containing directory
- contains any component in a directory that is not searchable
- contains any character in any component that is not valid in its containing directory.

The format of the diagnostic message is not specified, but will indicate the error detected and the corresponding *path* operand.

It will not be considered an error if one or more components of a *path* operand do not exist as long as a file matching the path name specified by the missing components could be created that does not violate any of the checks specified above.

OPTIONS

The following option is supported:

−**p** Instead of performing checks based on the underlying file system, write a diagnostic for each *path* operand that:

- is longer than **_POSIX_PATH_MAX** bytes
- contains any component longer than **_POSIX_NAME_MAX** bytes
- contains any character in any component that is not in the portable filename character set.

OPERANDS

The following operand is supported:

path A path to be checked.

EXAMPLES

To verify that all paths in an imported data interchange archive are legitimate and unambiguous on the current system:

```
pax –f archive | sed –e '/ == .*/s///' | xargs pathchk
if [ $? -eq 0 ]
then
        pax –r –f archive
else
        echo Investigate problems before importing files.
        exit 1
fi
```

To verify that all files in the current directory hierarchy could be moved to any system conforming to the X/Open specification that also supports the **pax**(1) command:

```
find . –print | xargs pathchk –p
if [ $? –eq 0 ]
then
        pax –w –f archive .
else
        echo Portable archive cannot be created.
        exit 1
fi
```

To verify that a user-supplied path names a readable file and that the application can create a file extending the given path without truncation and without overwriting any existing file:

```
case $- in
        *C*)   reset="";;
        *)     reset="set +C"
                set -C;;
esac
test -r "$path" && pathchk "$path.out" &&
        rm "$path.out" > "$path.out"
if [ $? -ne 0 ]; then
        printf "%s: %s not found or %s.out fails \
creation checks.\n" $0 "$path" "$path"
        $reset   # reset the noclobber option in case a trap
                 # on EXIT depends on it
        exit 1
fi
$reset
PROCESSING < "$path" > "$path.out"
```

The following assumptions are made in this example:

1. **PROCESSING** represents the code that will be used by the application to use **$path** once it is verified that **$path.out** will work as intended.

2. The state of the **noclobber** option is unknown when this code is invoked and should be set on exit to the state it was in when this code was invoked. (The **reset** variable is used in this example to restore the initial state.)
3. Note the usage of:

   rm "$path.out" > "$path.out"

   a. The **pathchk** command has already verified, at this point, that **$path.out** will not be truncated.
   b. With the **noclobber** option set, the shell will verify that **$path.out** does not already exist before invoking **rm**.
   c. If the shell succeeded in creating **$path.out**, **rm** will remove it so that the application can create the file again in the **PROCESSING** step.
   d. If the **PROCESSING** step wants the file to exist already when it is invoked, the:

      rm "$path.out" > "$path.out"

      should be replaced with:

      > "$path.out"

      which will verify that the file did not already exist, but leave **$path.out** in place for use by **PROCESSING**.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **pathchk**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 All *path* operands passed all of the checks.

>0 An error occurred.

SEE ALSO

pax(1), **test**(1), **environ**(5)