NAME

mktemp — make temporary filename (unique)

SYNOPSIS

mktemp [**-V**] | [**-dqtu**] [**-p** *directory*] [*template*]

DESCRIPTION

The **mktemp** utility takes the given filename *template* and overwrites a portion of it to create a unique filename. The *template* may be any filename with some number of 'Xs' appended to it, for example /tmp/tfile.XXXXXXXXXX. If no *template* is specified a default of tmp.XXXXXXXXXX is used and the **-t** flag is implied (see below).

The trailing 'Xs' are replaced with a combination of the current process number and random letters. The name chosen depends both on the number of 'Xs' in the *template* and the number of collisions with pre-existing files. The number of unique filenames **mktemp** can return depends on the number of 'Xs' provided; ten 'Xs' will result in **mktemp** testing roughly 26 ** 10 combinations.

If **mktemp** can successfully generate a unique filename, the file (or directory) is created with file permissions such that it is only readable and writable by its owner (unless the **-u** flag is given) and the filename is printed to standard output.

mktemp is provided to allow shell scripts to safely use temporary files. Traditionally, many shell scripts take the name of the program with the PID as a suffix and use that as a temporary filename. This kind of naming scheme is predictable and the race condition it creates is easy for an attacker to win. A safer, though still inferior approach is to make a temporary directory using the same naming scheme. While this does allow one to guarantee that a temporary file will not be subverted, it still allows a simple denial of service attack. For these reasons it is suggested that **mktemp** be used instead.

The options are as follows:

-V Print the version and exit.

-d Make a directory instead of a file.

-p *directory*
Use the specified *directory* as a prefix when generating the temporary filename. The *directory* will be overridden by the user's TMPDIR environment variable if it is set. This option implies the **-t** flag (see below).

-q Fail silently if an error occurs. This is useful if a script does not want error output to go to standard error.

-t Generate a path rooted in a temporary directory. This directory is chosen as follows:

- If the user's TMPDIR environment variable is set, the directory contained therein is used.
- Otherwise, if the **-p** flag was given the specified directory is used.
- If none of the above apply, /tmp is used.

In this mode, the *template* (if specified) should be a directory component (as opposed to a full path) and thus should not contain any forward slashes.

-u Operate in "unsafe" mode. The temp file will be unlinked before **mktemp** exits. This is slightly better than **mktemp**(*3*) but still introduces a race condition. Use of this option is not encouraged.

The **mktemp** utility exits with a value of 0 on success or 1 on failure.

ENVIRONMENT

TMPDIR directory in which to place the temporary file when in **-t** mode

EXAMPLES

The following sh(1) fragment illustrates a simple use of **mktemp** where the script should quit if it cannot get a safe temporary file.

```
TMPFILE=‘mktemp /tmp/example.XXXXXXXXXX‘ || exit 1
echo "program output" >> $TMPFILE
```

The same fragment with support for a user’s TMPDIR environment variable can be written as follows.

```
TMPFILE=‘mktemp -t example.XXXXXXXXXX‘ || exit 1
echo "program output" >> $TMPFILE
```

This can be further simplified if we don’t care about the actual name of the temporary file. In this case the **-t** flag is implied.

```
TMPFILE=‘mktemp‘ || exit 1
echo "program output" >> $TMPFILE
```

In some cases, it may be desirable to use a default temporary directory other than /tmp. In this example the temporary file will be created in /extra/tmp unless the user’s TMPDIR environment variable specifies otherwise.

```
TMPFILE=‘mktemp -p /extra/tmp example.XXXXXXXXXX‘ || exit 1
echo "program output" >> $TMPFILE
```

In some cases, we want the script to catch the error. For instance, if we attempt to create two temporary files and the second one fails we need to remove the first before exiting.

```
TMP1=‘mktemp -t example.1.XXXXXXXXXX‘ || exit 1
TMP2=‘mktemp -t example.2.XXXXXXXXXX‘
if [ $? -ne 0 ]; then
        rm -f $TMP1
        exit 1
fi
```

Or perhaps you don’t want to exit if **mktemp** is unable to create the file. In this case you can protect that part of the script thusly.

```
TMPFILE=‘mktemp -q -t example.XXXXXXXXXX‘ && {
        # Safe to use $TMPFILE in this block
        echo data > $TMPFILE
        ...
        rm -f $TMPFILE
}
```

SEE ALSO

mkdtemp(3), mkstemp(3), mktemp(3)

HISTORY

The **mktemp** utility appeared in OpenBSD 2.1.