NAME

getopt - parse command options

SYNOPSIS

args=‘getopt *optstring $*‘; errcode=$?; set -- $args

DESCRIPTION

The **getopt** utility is used to break up options in command lines for easy parsing by shell procedures, and to check for legal options. *Optstring* is a string of recognized option letters (see getopt(3)); if a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space. The special option ‘--’ is used to delimit the end of the options. The **getopt** utility will place ‘--’ in the arguments at the end of the options, or recognize it if used explicitly. The shell arguments (**$1 $2** ...) are reset so that each option is preceded by a ‘-’ and in its own shell argument; each option argument is also in its own shell argument.

EXIT STATUS

The **getopt** utility prints an error message on the standard error output and exits with status > 0 when it encounters an option letter not included in *optstring*.

EXAMPLES

The following code fragment shows how one might process the arguments for a command that can take the options **-a** and **-b**, and the option **-o**, which requires an argument.

```
args=‘getopt abo: $*‘
# you should not use ‘getopt abo: "$@"‘ since that would parse
# the arguments differently from what the set command below does.
if [ $? -ne 0 ]; then
        echo ’Usage: ...’
        exit 2
fi
set -- $args
# You cannot use the set command with a backquoted getopt directly,
# since the exit code from getopt would be shadowed by those of set,
# which is zero by definition.
while :; do
        case "$1" in
        -a|-b)
                echo "flag $1 set"; sflags="${1#-}$sflags"
                shift
                ;;
```

```
        -o)
                echo "oarg is '$2'"; oarg="$2"
                shift; shift
                ;;
        --)
                shift; break
                ;;
        esac
done
echo "single-char flags: '$sflags'"
echo "oarg is '$oarg'"
```

This code will accept any of the following as equivalent:

```
cmd -aoarg file1 file2
cmd -a -o arg file1 file2
cmd -oarg -a file1 file2
cmd -a -oarg -- file1 file2
```

SEE ALSO

getopts(1), sh(1), getopt(3)

HISTORY

Written by Henry Spencer, working from a Bell Labs manual page. Behavior believed identical to the Bell version. Example changed in FreeBSD version 3.2 and 4.0.

BUGS

Whatever getopt(3) has.

Arguments containing white space or embedded shell metacharacters generally will not survive intact; this looks easy to fix but is not. People trying to fix **getopt** or the example in this manpage should check the history of this file in FreeBSD.

The error message for an invalid option is identified as coming from **getopt** rather than from the shell procedure containing the invocation of **getopt**; this again is hard to fix.

The precise best way to use the **set** command to set the arguments without disrupting the value(s) of shell options varies from one shell version to another.

Each shellscript has to carry complex code to parse arguments halfway correctly (like the example

presented here). A better getopt-like tool would move much of the complexity into the tool and keep the client shell scripts simpler.