NAME

makemsg – Write a message from within a Makefile.

SYNOPSIS

${HOME}/bin/exe/makemsg 'advisory message'
${HOME}/bin/exe/makemsg "'cat fyle'"
echo 'error message' | ${HOME}/bin/exe/makemsg
cat errors | ${HOME}/bin/exe/makemsg

DESCRIPTION

One function of the mymake script is to filter out unwanted text from /usr/bin/make. This program writes wanted text on /dev/tty so that it will not be eliminated by that filtering.

In the first prototype makemsg writes the given string, assumed to be advisory, and stops with a return code of 0 so that mymake can continue. The message is enclosed in forward single quotes. If makemsg finds a command line parameter it ignores standard-in.

In the second prototype makemsg writes the output of the cat command and stops with a return code of 0. The parameter is enclosed in double quotes and consists of the command enclosed in back quotes. If a command issued in this context fails, makemsg finds no parameter and blocks on trying to read from standard-in.

In the third and fourth prototypes makemsg writes the input text, assumed to be an error report, and stops with a return code of 1 so that mymake is interrupted.

UNITS and FILES

0	/dev/tty
5	standard in

SEE ALSO

The man page for mymake explains how to use this utility in a makefile.

BUGS

If no command line parameter is supplied and also no input is provided on standard-in, the program blocks on read. If more than one run-line parameter is supplied only the first is written.

EXAMPLES

```
unix[1] makemsg 'hello there'
hello there
unix[2] echo $?
0
```

This example shows how to write an informational message.

```
unix[3] echo 'error message' | makemsg
error message
unix[4] echo $?
1
```

This example shows how to write an error message.

```
unix[5] silent | makemsg
unix[6] echo $?
0
```

In this example, "silent" is a program that does not write any output. The pipeline presents an end-of-file to the standard-in of makemsg so it stops with return code 0.

```
unix[7] filter < input > output 2>&1 | makemsg
error message
unix[8] echo $?
1
```

In this example, "filter" is a program that reads from standard in, writes to standard out, and reports errors on standard error. The shell locution 2>&1 redirects standard error to the pipeline so that it reaches makemsg. If filter writes nothing to standard-error, then the pipeline presents an end-of-file to the standard-in of makemsg and it stops with return code 0. In Makefile, this locution stops mymake only in the event that filter writes an error message on standard-error.

```
unix[9] makemsg
^C
unix[10]
```

This example shows that if makemsg is provided with no input and no parameter, it blocks at an attempt to read from standard-in and must be interrupted by control-C.