NAME

exstr – extract strings from source files

SYNOPSIS

exstr *filename . . .*
exstr –e *filename . . .*
exstr –r [**–d**] *filename . . .*

DESCRIPTION

The **exstr** utility is used to extract strings from C-language source files and replace them by calls to the message retrieval function (see **gettxt**(3C)). This utility will extract all character strings surrounded by double quotes, not just strings used as arguments to the **printf** command or the **printf** routine. In the first form, **exstr** finds all strings in the source files and writes them on the standard output. Each string is preceded by the source file name and a colon.

The first step is to use **exstr –e** to extract a list of strings and save it in a file. Next, examine this list and determine which strings can be translated and subsequently retrieved by the message retrieval function. Then, modify this file by deleting lines that can't be translated and, for lines that can be translated, by adding the message file names and the message numbers as the fourth (*msgfile*) and fifth (*msgnum*) entries on a line. The message files named must have been created by **mkmsgs**(1) and exist in **/usr/lib/locale/***locale***/LC_MESSAGES**. (The directory *locale* corresponds to the language in which the text strings are written; see **setlocale**(3C)). The message numbers used must correspond to the sequence numbers of strings in the message files.

Now use this modified file as input to **exstr –r** to produce a new version of the original C-language source file in which the strings have been replaced by calls to the message retrieval function **gettxt**( ). The *msgfile* and *msgnum* fields are used to construct the first argument to **gettxt**( ). The second argument to **gettxt**( ) is printed if the message retrieval fails at run-time. This argument is the null string, unless the **–d** option is used.

This utility cannot replace strings in all instances. For example, a static initialized character string cannot be replaced by a function call. A second example is that a string could be in a form of an escape sequence which could not be translated. In order not to break existing code, the files created by invoking **exstr –e** must be examined and lines containing strings not replaceable by function calls must be deleted. In some cases the code may require modifications so that strings can be extracted and replaced by calls to the message retrieval function.

OPTIONS

–e Extract a list of strings from the named C-language source files, with positional information. This list is produced on standard output in the following format:

file:line:position:msgfile:msgnum:string

file	the name of a C-language source file
line	line number in the file
position	character position in the line
msgfile	null
msgnum	null
string	the extracted text string

Normally you would redirect this output into a file. Then you would edit this file to add the values you want to use for *msgfile* and *msgnum*:

msgfile the file that contains the text strings that will replace *string*. A file with this name must be created and installed in the appropriate place by the **mkmsgs**(1) utility.

msgnum the sequence number of the string in *msgfile*.

The next step is to use **exstr** −**r** to replace *string*s in *file*.

−**r** Replace strings in a C-language source file with function calls to the message retrieval function **gettxt**().

−**d** This option is used together with the −**r** option. If the message retrieval fails when **gettxt**() is invoked at run-time, then the extracted string is printed. You would use the capability provided by **exstr** on an application program that needs to run in an international environment and have messages print in more than one language. **exstr** replaces text strings with function calls that point at strings in a message data base. The data base used depends on the run-time value of the **LC_MESSAGES** environment variable (see **environ**(5)).

EXAMPLES

The following examples show uses of **exstr**.

Assume that the file **example.c** contains two strings:

```
main()
{
        printf("This is an example\n");
        printf("Hello world!\n");
}
```

The **exstr** utility, invoked with the argument **example.c** extracts strings from the named file and prints them on the standard output.

```
example% exstr example.c
```

produces the following output:

```
example.c:This is an example\n
example.c:Hello world!\n
```

```
example% exstr −e example.c > example.stringsout
```

produces the following output in the file **example.stringsout**:

```
example.c:3:8:::This is an example\n
example.c:4:8:::Hello world!\n
```

You must edit **example.stringsout** to add the values you want to use for the *msgfile* and *msgnum* fields before these strings can be replaced by calls to the retrieval function. If **UX** is the name of the message file, and the numbers **1** and **2** represent the sequence number of the strings in the file, here is what **example.stringsout** looks like after you add this information:

```
example.c:3:8:UX:1:This is an example\n
example.c:4:8:UX:2:Hello world!\n
```

The **exstr** utility can now be invoked with the −**r** option to replace the strings in the source file by calls to the message retrieval function **gettxt**().

```
example% exstr −r example.c <example.stringsout >intlexample.c
```

produces the following output:

```
extern char *gettxt();
main()
{
        printf(gettxt("UX:1", ""));
        printf(gettxt("UX:2", ""));
}
```

```
example% exstr −rd example.c <example.stringsout >intlexample.c
```

uses the extracted strings as a second argument to **gettxt**().

```
extern char *gettxt();
main()
{
        printf(gettxt("UX:1", "This is an example\n"));
        printf(gettxt("UX:2", "Hello world!\n"));
}
```

FILES

/usr/lib/locale/*locale***/LC_MESSAGES/***
    files created by **mkmsgs**(1)

SEE ALSO

gettxt(1), **mkmsgs**(1), **printf**(1), **srchtxt**(1), **gettxt**(3C), **printf**(3S), **setlocale**(3C), **environ**(5)

DIAGNOSTICS

The error messages produced by **exstr** are intended to be self-explanatory. They indicate errors in the command line or format errors encountered within the input file.