NAME

regcmp, regex – compile and execute regular expression

SYNOPSIS

cc [*flag* ...] *file* ... **–lgen** [*library* ...]

#include <libgen.h>

char ∗**regcmp(const char** ∗*string1*, /∗ **char** ∗*string2* ∗/ **...**, *int* /∗**(char** ∗**)0**∗/**);**

char ∗**regex(const char** ∗*re*, **const char** ∗*subject*, /∗ **char** ∗*ret0* ∗/ **...);**

extern char ∗**__loc1;**

MT-LEVEL

MT-Safe

DESCRIPTION

regcmp( ) compiles a regular expression (consisting of the concatenated arguments) and returns a pointer to the compiled form. **malloc**(3C) is used to create space for the compiled form. It is the user's responsibility to free unneeded space so allocated. A **NULL** return from **regcmp( )** indicates an incorrect argument. **regcmp**(1) has been written to generally preclude the need for this routine at execution time.

regex( ) executes a compiled pattern against the subject string. Additional arguments are passed to receive values back. **regex( )** returns **NULL** on failure or a pointer to the next unmatched character on success. A global character pointer **__loc1** points to where the match began. **regcmp( )** and **regex( )** were mostly borrowed from the editor, **ed**(1); however, the syntax and semantics have been changed slightly. The following are the valid symbols and associated meanings.

[ ] ∗ . ^ This group of symbols retains its meaning as described on the **regexp**(5) manual page.

$ Matches the end of the string; **\n** matches a newline.

– Within brackets the minus means *through*. For example, **[a–z]** is equivalent to **[abcd...xyz]**. The – can appear as itself only if used as the first or last character. For example, the character class expression **[]–]** matches the characters **]** and –.

+ A regular expression followed by **+** means *one or more times*. For example, **[0–9]+** is equivalent to **[0–9][0–9]∗.**

{*m*} {*m,*} {*m,u*} Integer values enclosed in **{ }** indicate the number of times the preceding regular expression is to be applied. The value *m* is the minimum number and *u* is a number, less than 256, which is the maximum. If only *m* is present (that is, {*m*}), it indicates the exact number of times the regular expression is to be applied. The value {*m*,} is analogous to {*m,infinity*}. The plus (**+**) and star (∗) operations are equivalent to **{1,}** and **{0,}** respectively.

(...)$*n* The value of the enclosed regular expression is to be returned. The value will be stored in the (*n*+1)th argument following the subject argument. At most, ten enclosed regular expressions are allowed. **regex( )** makes its assignments unconditionally.

(. . .) Parentheses are used for grouping. An operator, for example, *****, **+**, **{ }**, can work on a single character or a regular expression enclosed in parentheses. For example, **(a*(cb+)*)$0**.

By necessity, all the above defined symbols are special. They must, therefore, be escaped with a \ (backslash) to be used as themselves.

EXAMPLES

The following example matches a leading newline in the subject string pointed at by cursor.

```
char *cursor, *newcursor, *ptr;
        . . .
newcursor = regex((ptr = regcmp("^\n", (char *)0)), cursor);
free(ptr);
```

The following example matches through the string **Testing3** and returns the address of the character after the last matched character (the ''**4**''). The string **Testing3** is copied to the character array **ret0**.

```
char ret0[9];
char *newcursor, *name;
        . . .
name = regcmp("([A–Za–z][A–za–z0–9]{0,7})$0", (char *)0);
newcursor = regex(name, "012Testing345", ret0);
```

The following example applies a precompiled regular expression in **file.i** (see **regcmp**(1)) against *string*.

```
#include "file.i"
char *string, *newcursor;
        . . .
newcursor = regex(name, string);
```

FILES

/usr/ccs/lib/libgen.a

SEE ALSO

ed(1), **regcmp**(1), **malloc**(3C), **regexp**(5)

NOTES

The user program may run out of memory if **regcmp( )** is called iteratively without freeing the vectors no longer required.

When compiling multi-thread applications, the **_REENTRANT** flag must be defined on the compile line. This flag should only be used in multi-thread applications.