NAME

encrypt, setkey, encrypt_r, setkey_r - encrypt 64-bit messages

SYNOPSIS

#define _XOPEN_SOURCE /* See feature_test_macros(7) */
#include <unistd.h>

void encrypt(char *block*[**64**], **int** *edflag*);

#define _XOPEN_SOURCE /* See feature_test_macros(7) */
#include <stdlib.h>

**void setkey(const char ***key*);

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <crypt.h>

**void setkey_r(const char ***key*, **struct crypt_data ***data*);
void encrypt_r(char ***block*, **int *edflag*, **struct crypt_data ***data*);

Each of these requires linking with *-lcrypt*.

DESCRIPTION

These functions encrypt and decrypt 64-bit messages. The **setkey**() function sets the key used by **encrypt**(). The*key* argumen t used here is an array of 64 bytes, each of which has numerical value 1 or 0. The bytes key[n] where n=8*i-1 are ignored, so that the effective key length is 56 bits.

The **encrypt**() function modifies the passed buffer, encoding if *edflag* is 0, and decoding if 1 is being passed. Like the *key* argument, also *block* is a bit vector representation of the actual value that is encoded. The result is returned in that same vector.

These two functions are not reentrant, that is, the key data is kept in static storage. The functions **setkey_r**() and **encrypt_r**() are the reentrant versions. They use the following structure to hold the key data:

```
struct crypt_data {
char keysched[16 * 8];
char sb0[32768];
char sb1[32768];
char sb2[32768];
char sb3[32768];
char crypt_3_buf[14];
char current_salt[2];
long int current_saltbits;
int direction;
int initialized;
};
```

Before calling **setkey_r**() set *data->initialized* to zero.

RETURN VALUE

These functions do not return any value.

ERRORS

Set *errno* to zero before calling the above functions. On success, it is unchanged.

ENOSYS
The function is not provided. (For example because of former USA export restrictions.)

ATTRIBUTES

Multithreading (see pthreads(7))

The **encrypt**() and **setkey**() functions are not thread-safe.

The **encrypt_r**() and **setkey_r**() functions are thread-safe.

CONFORMING TO

The functions **encrypt**() and **setkey**() conform to SVr4, SUSv2, and POSIX.1-2001. The functions **encrypt_r**() and **setkey_r**() are GNU extensions.

NOTES

In glibc 2.2, these functions use the DES algorithm.

EXAMPLE

You need to link with libcrypt to compile this example with glibc. To do useful work, the *key[]* and *txt[]* arrays must be filled with a useful bit pattern.

```
#define _XOPEN_SOURCE
#include <unistd.h>
#include <stdlib.h>

int
main(void)
{
char key[64]; /* bit pattern for key */
char txt[64]; /* bit pattern for messages */

setkey(key);
encrypt(txt, 0); /* encode */
encrypt(txt, 1); /* decode */
}
```

SEE ALSO

cbc_crypt(3), crypt(3), ecb_crypt(3),

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-pages/.