



## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'setkey.3'***

**C:~>man setkey.3**

ENCRYPT(3)                      Linux Programmer's Manual                      ENCRYPT(3)

### 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 argument 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.)

#### VERSIONS

Because they employ the DES block cipher, which is no longer considered secure, `crypt()`, `crypt_r()`, `setkey()`, and `setkey_r()` were removed in glibc 2.28. Applications should switch to a modern cryptography library, such as `libgcrypt`.

#### ATTRIBUTES

For an explanation of the terms used in this section, see `attributes(7)`.

??

?Interface           ? Attribute   ? Value           ?

??



```

    setkey(key);
}
printf("Before encrypting: %s\n", orig);
encrypt(buf, 0);
for (i = 0; i < 8; i++) {
    for (j = 0, txt[i] = '\0'; j < 8; j++) {
        txt[i] |= buf[i * 8 + j] << j;
    }
    txt[8] = '\0';
}
printf("After encrypting: %s\n", txt);
encrypt(buf, 1);
for (i = 0; i < 8; i++) {
    for (j = 0, txt[i] = '\0'; j < 8; j++) {
        txt[i] |= buf[i * 8 + j] << j;
    }
    txt[8] = '\0';
}
printf("After decrypting: %s\n", txt);
exit(EXIT_SUCCESS);
}

```

#### SEE ALSO

cbc\_crypt(3), crypt(3), ecb\_crypt(3),

#### COLOPHON

This page is part of release 5.05 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 <https://www.kernel.org/doc/man-pages/>.

2018-04-30

ENCRYPT(3)