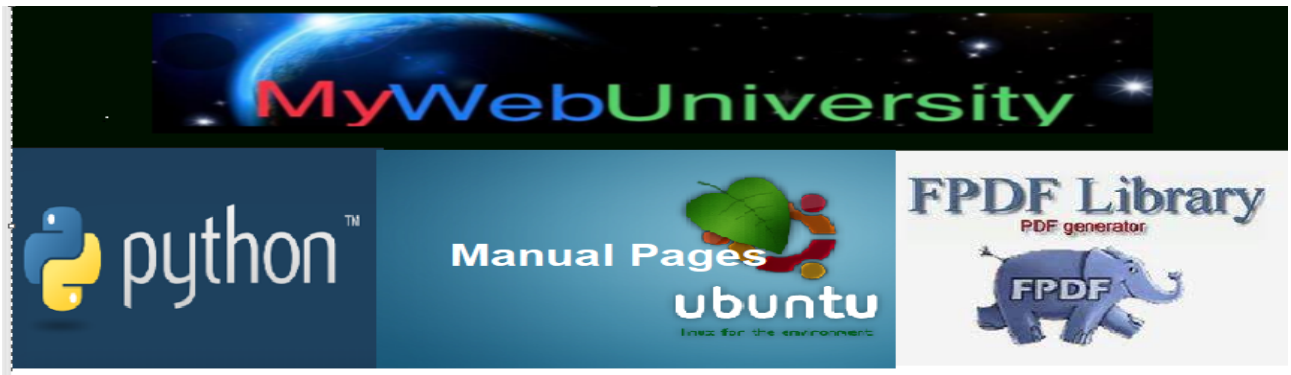

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'encrypt.3'

$ man encrypt.3

encrypt(3) Library Functions Manual encrypt(3)

NAME

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

LIBRARY

Password hashing library (libcrypt, -lcrypt)

SYNOPSIS

```
#define _XOPEN_SOURCE       /* See feature_test_macros(7) */
#include <unistd.h>
[[deprecated]] void encrypt(char block[64], int edflag);
#define _XOPEN_SOURCE       /* See feature_test_macros(7) */
#include <stdlib.h>
[[deprecated]] void setkey(const char *key);
#define _GNU_SOURCE         /* See feature_test_macros(7) */
#include <crypt.h>
[[deprecated]] void setkey_r(const char *key, struct crypt_data *data);
[[deprecated]] void encrypt_r(char *block, int edflag,
                              struct crypt_data *data);
```

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 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, errno is unchanged.

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

ATTRIBUTES

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

Interface	Attribute	Value
encrypt(), setkey()	Thread safety	MT-Unsafe race:crypt
encrypt_r(), setkey_r()	Thread safety	MT-Safe

STANDARDS

encrypt()

setkey()

POSIX.1-2008.

encrypt_r()

setkey_r()

None.

HISTORY

Removed in glibc 2.28.

Because they employ the DES block cipher, which is no longer considered secure, these func?

tions were removed from glibc. Applications should switch to a modern cryptography library,

such as libgcrypt.

encrypt()

setkey()

POSIX.1-2001, SUS, SVr4.

Availability in glibc

See crypt(3).

Features in glibc

In glibc 2.2, these functions use the DES algorithm.

EXAMPLES

```
#define _XOPEN_SOURCE
#include <crypt.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int
main(void)
{
    char key[64];
    char orig[9] = "eggplant";
    char buf[64];
    char txt[9];
    for (size_t i = 0; i < 64; i++) {
```

```
            key[i] = rand() & 1;
        }
        for (size_t i = 0; i < 8; i++) {
            for (size_t j = 0; j < 8; j++) {
                buf[i * 8 + j] = orig[i] >> j & 1;
            }
            setkey(key);
        }
        printf("Before encrypting: %s\n", orig);
        encrypt(buf, 0);
        for (size_t i = 0; i < 8; i++) {
            for (size_t 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 (size_t i = 0; i < 8; i++) {
            for (size_t 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)