İSİM

encrypt – 64 bitlik iletileri şifreler
setkey – encrypt() tarafından kullanılan anahtarı belirler
encrypt_r – 64 bitlik iletileri şifreler (evresel)
setkey_r – encrypt_r() tarafından kullanılan anahtarı belirler (evresel)

BİLDİRİM

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

void encrypt (char blok[64], int im);

#define _XOPEN_SOURCE
#include <stdlib.h>

void setkey (const char *anahtar);

#define _GNU_SOURCE
#include <crypt.h>

void setkey_r (const char *anahtar, struct crypt_data *veri)
void encrypt_r (char *blok, int im, struct crypt_data *veri);
```

Bunların herbiri **-lcrypt** ile ilintileme gerektirir.

AÇIKLAMA

Bu işlevler 64 bitlik iletileri şifreler ve deşifrelerler. **setkey()** işlevi **encrypt()** işlevi tarafından kullanılan anahtarı belirler. Burada kullanılan *anahtar* parametresi bir bayt dizisidir ve her bayt 1 ya da 0 sayısal değerine sahiptir. *anahtar*`[n]` dizisinin indis değeri `n=8*i-1` olan elemanları yoksayılır, bu durumda asıl anahtar uzunluğu 56 bit olur.

setkey() işlevi kendine aktarılan tamponu *im* olarak 0 verilmişse şifreleyerek, 0 verilmişse deşifreleyerek değiştirir. *anahtar* parametresi gibi *blok* parametresi de şifrelenmiş değerin bit gösteriminin vektörüdür. Sonuç aynı vektör içinde döner.

Bu iki işlev evresel değildir, yani anahtar verisi durağan bellek bölgesinde saklanır. **setkey_r()** ve **encrypt_r()** işlevleri evreseldir. ve anahtar verisini saklamak için **crypt_data** veri yapısını kullanırlar:

```
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, initialized;
};
```

DÖNÜŞ DEĞERİ

Bu işlevler herhangi bir değer döndürmezler.

HATALAR

Yukarıdaki işlevler çağrılmadan önce **errno** değişkeni sıfırlanır. Başarı durumunda değeri değişmez.

ENOSYS
İşlev kütüphanede bulunmamaktadır (Örneğin, ABD'nin ihracat sınırlamalarından dolayı).

ÖRNEK

Bu örneği glibc2.2 ile derlemek için `libcrypt` ile ilintilemeniz gerekir. Anlamlı bir çalıştırma için *anahtar[]* ve *ileti[]* dizilerini anlamlı bir bit kalıbı ile doldurmalısınız. `crypt.h` başlık dosyasının **setkey()** ve **encrypt()** işlevlerinin prototiplerini koşulsuz olarak verdiğini unutmayın.

```
#include <crypt.h>

main() {
  char anahtar[64];     /* anahtarın bit kalıbı */
  char ileti[64];       /* iletinin bit kalıbı */
  setkey(anahtar);
  encrypt(ileti, 0);    /* şifreler */
  encrypt(ileti, 1);    /* deşifreler */
}
```

NOTLAR

glibc2.2'de bu işlevler DES algoritmasını kullanır.

UYUMLULUK

encrypt() ve **setkey()** işlevleri SVID, SUSv2 ve POSIX 1003.1–2001 uyumludur. **encrypt_r()** ve **setkey_r()** işlevleri ise GNU oluşumudur.

İLGİLİ BELGELER

cbc_crypt(3), **crypt(3)**, **ecb_crypt(3)**, **fcrypt(3)**.

ÇEVİREN

Emin İslam Tatlı <eminislam (at) web.de>, Nisan 2004

YASAL UYARI

Özgün belgedeki telif hakkı beyanı

Modified 2002–07–23 19:21:35 CEST 2002 Walter Harms
<walter.harms@informatik.uni–oldenburg.de>

Modified 2003–04–04, aeb

Bu dosya (man3-encrypt.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

19 Ocak 2007