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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'crypt.3p' command

$ man crypt.3p

CRYPT(3P) POSIX Programmer's Manual CRYPT(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

crypt ? string encoding function (CRYPT)

SYNOPSIS

```
#include <unistd.h>
char *crypt(const char *key, const char *salt);
```

DESCRIPTION

The crypt() function is a string encoding function. The algorithm is implementation-defined.

The key argument points to a string to be encoded. The salt argument shall be a string of at least two bytes in length not including the null character chosen from the set:

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9 . /

The first two bytes of this string may be used to perturb the encoding algorithm.

The return value of crypt() points to static data that is overwritten

by each call.

The crypt() function need not be thread-safe.

RETURN VALUE

Upon successful completion, crypt() shall return a pointer to the en? coded string. The first two bytes of the returned value shall be those of the salt argument. Otherwise, it shall return a null pointer and set errno to indicate the error.

ERRORS

The crypt() function shall fail if:

ENOSYS The functionality is not supported on this implementation.

The following sections are informative.

EXAMPLES

Encoding Passwords

The following example finds a user database entry matching a particular user name and changes the current password to a new password. The crypt() function generates an encoded version of each password. The first call to crypt() produces an encoded version of the old password; that encoded password is then compared to the password stored in the user database. The second call to crypt() encodes the new password be? fore it is stored.

The putpwent() function, used in the following example, is not part of POSIX.1?2008.

```
#include <unistd.h>
#include <pwd.h>
#include <string.h>
#include <stdio.h>
...
int valid_change;
int pfd;  /* Integer for file descriptor returned by open(). */
FILE *fpfd;  /* File pointer for use in putpwent(). */
struct passwd *p;
char user[100];
char oldpasswd[100];
```

```
        char newpasswd[100];
        char savepasswd[100];
        ...
        valid_change = 0;
        while ((p = getpwent()) != NULL) {
            /* Change entry if found. */
            if (strcmp(p->pw_name, user) == 0) {
                if (strcmp(p->pw_passwd, crypt(oldpasswd, p->pw_passwd)) == 0) {
                    strcpy(savepasswd, crypt(newpasswd, user));
                    p->pw_passwd = savepasswd;
                    valid_change = 1;
                }
                else {
                    fprintf(stderr, "Old password is not valid\n");
                }
            }
            /* Put passwd entry into ptmp. */
            putpwent(p, fpfd);
        }
```

APPLICATION USAGE

The values returned by this function need not be portable among XSI-conformant systems.

Several implementations offer extensions via characters outside of the set specified for the salt argument for specifying alternative algo? rithms; while not portable, these extensions may offer better security.

The use of crypt() for anything other than password hashing is not rec? ommended.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

encrypt(), setkey()

The Base Definitions volume of POSIX.1?2017, <unistd.h>

COPYRIGHT