



Rocky Enterprise Linux 9.2 Manual Pages on command 'crypt_gensalt_rn.3'

C:\>man crypt_gensalt_rn.3

CRYPT_GENSALT(3) BSD Library Functions Manual CRYPT_GENSALT(3)

NAME

crypt_gensalt, crypt_gensalt_rn, crypt_gensalt_ra ? encode settings for passphrase hashing

LIBRARY

Crypt Library (libcrypt, -lcrypt)

SYNOPSIS

```
#include <crypt.h>
```

```
char *
```

```
crypt_gensalt(const char *prefix, unsigned long count, const char *rbytes,  
             int nrbytes);
```

```
char *
```

```
crypt_gensalt_rn(const char * prefix, unsigned long count, const char *rbytes,  
                int nrbytes, char * output, int output_size);
```

```
char *
```

```
crypt_gensalt_ra(const char *prefix, unsigned long count, const char *rbytes,  
                int nrbytes);
```

DESCRIPTION

The crypt_gensalt, crypt_gensalt_rn, and crypt_gensalt_ra functions compile a string for use as the setting argument to crypt, crypt_r, crypt_rn, and crypt_ra. prefix selects the hashing method to use. count controls the CPU time cost of the hash; the valid range for count and the exact meaning of ?CPU time cost? depends on the hashing

method, but larger numbers correspond to more costly hashes. `rbytes` should point to `nrbytes` cryptographically random bytes for use as `?salt?`

If `prefix` is a null pointer, the best available hashing method will be selected.

(CAUTION: if `prefix` is an empty string, the `?traditional?` DES-based hashing method will be selected; this method is unacceptably weak by modern standards.) If `count` is 0, a low default cost will be selected. If `rbytes` is a null pointer, an appropriate number of random bytes will be obtained from the operating system, and `nrbytes` is ignored.

See `crypt(5)` for other strings that can be used as `prefix`, and valid values of `count` for each.

RETURN VALUES

`crypt_gensalt`, `crypt_gensalt_rn`, and `crypt_gensalt_ra` return a pointer to an encoded setting string. This string will be entirely printable ASCII, and will not contain whitespace or the characters `?:?`, `?;?`, `?*?`, `?!?`, or `?\?`. See `crypt(5)` for more detail on the format of this string. Upon error, they return a null pointer and set `errno` to an appropriate error code.

`crypt_gensalt` places its result in a static storage area, which will be overwritten by subsequent calls to `crypt_gensalt`. It is not safe to call `crypt_gensalt` from multiple threads simultaneously. However, it is safe to pass the string returned by `crypt_gensalt` directly to `crypt` without copying it; each function has its own static storage area.

`crypt_gensalt_rn` places its result in the supplied output buffer, which has `output_size` bytes of storage available. `output_size` should be greater than or equal to `CRYPT_GENSALT_OUTPUT_SIZE`.

`crypt_gensalt_ra` allocates memory for its result using `malloc(3)`. It should be freed with `free(3)` after use.

Upon error, in addition to returning a null pointer, `crypt_gensalt` and `crypt_gensalt_rn` will write an invalid setting string to their output buffer, if there is enough space; this string will begin with a `?*?` and will not be equal to `prefix`.

ERRORS

`EINVAL` `prefix` is invalid or not supported by this implementation; `count` is invalid for the requested `prefix`; the input `nrbytes` is insufficient

