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 for the smallest valid salt with the requested *prefix*.

ERANGE
: **crypt_gensalt_rn** only: *output_size* is too small to hold the compiled *setting* string.

ENOMEM
: Failed to allocate internal scratch memory.
crypt_gensalt_ra only: failed to allocate memory for the compiled *setting* string.

ENOSYS, EACCES, EIO, etc.
: Obtaining random bytes from the operating system failed. This can only happen when *rbytes* is a null pointer.

FEATURE TEST MACROS

The following macros are defined by <crypt.h>:

CRYPT_GENSALT_IMPLEMENTS_DEFAULT_PREFIX
: A null pointer can be specified as the *prefix* argument.

CRYPT_GENSALT_IMPLEMENTS_AUTO_ENTROPY
: A null pointer can be specified as the *rbytes* argument.

PORTABILITY NOTES

The functions **crypt_gensalt**, **crypt_gensalt_rn**, and **crypt_gensalt_ra** are not part of any standard. They originate with the Openwall project. A function with the name **crypt_gensalt** also exists on Solaris 10 and newer, but its prototype and semantics differ.

The default prefix and auto entropy features are available since libxcrypt version 4.0.0. Portable software can use feature test macros to find out whether null pointers can be used for the *prefix* and *rbytes* arguments.

The set of supported hashing methods varies considerably from system to system.

ATTRIBUTES

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

Interface	Attribute	Value
crypt_gensalt	Thread safety	MT-Unsafe race:crypt_gensalt
crypt_gensalt_rn, **crypt_gensalt_ra**	Thread safety	MT-Safe

SEE ALSO

crypt(3), getpass(3), getpwent(3), shadow(3), login(1), passwd(1), crypt(5), passwd(5), shadow(5), pam(8)