NAME

uuid_generate, uuid_generate_random, uuid_generate_time, uuid_generate_time_safe − create a new unique UUID value

SYNOPSIS

#include <uuid.h>

void uuid_generate(uuid_t *out***);**
void uuid_generate_random(uuid_t *out***);**
void uuid_generate_time(uuid_t *out***);**
int uuid_generate_time_safe(uuid_t *out***);**
void uuid_generate_md5(uuid_t *out***, const uuid_t** *ns***, const char** **name***, **size_t** *len***);**
void uuid_generate_sha1(uuid_t *out***, const uuid_t** *ns***, const char** **name***, **size_t** *len***);**

DESCRIPTION

The **uuid_generate** function creates a new universally unique identifier (UUID). The uuid will be generated based on high-quality randomness from */dev/urandom*, if available. If it is not available, then **uuid_generate** will use an alternative algorithm which uses the current time, the local ethernet MAC address (if available), and random data generated using a pseudo-random generator.

The **uuid_generate_random** function forces the use of the all-random UUID format, even if a high-quality random number generator (i.e., */dev/urandom*) is not available, in which case a pseudo-random generator will be substituted. Note that the use of a pseudo-random generator may compromise the uniqueness of UUIDs generated in this fashion.

The **uuid_generate_time** function forces the use of the alternative algorithm which uses the current time and the local ethernet MAC address (if available). This algorithm used to be the default one used to generate UUIDs, but because of the use of the ethernet MAC address, it can leak information about when and where the UUID was generated. This can cause privacy problems in some applications, so the **uuid_generate** function only uses this algorithm if a high-quality source of randomness is not available. To guarantee uniqueness of UUIDs generated by concurrently running processes, the uuid library uses a global clock state counter (if the process has permissions to gain exclusive access to this file) and/or the **uuidd** daemon, if it is running already or can be spawned by the process (if installed and the process has enough permissions to run it). If neither of these two synchronization mechanisms can be used, it is theoretically possible that two concurrently running processes obtain the same UUID(s). To tell whether the UUID has been generated in a safe manner, use **uuid_generate_time_safe**.

The **uuid_generate_time_safe** function is similar to **uuid_generate_time**, except that it returns a value which denotes whether any of the synchronization mechanisms (see above) has been used.

The UUID is 16 bytes (128 bits) long, which gives approximately 3.4x10^38 unique values (there are approximately 10^80 elementary particles in the universe according to Carl Sagan's *Cosmos*). The new UUID can reasonably be considered unique among all UUIDs created on the local system, and among UUIDs created on other systems in the past and in the future.

The **uuid_generate_md5** and **uuid_generate_sha1** functions generate an MD5 and SHA1 hashed (predictable) UUID based on a well-known UUID providing the namespace and an arbitrary binary string. The UUIDs conform to V3 and V5 UUIDs per RFC-4122.

RETURN VALUE

The newly created UUID is returned in the memory location pointed to by *out*. **uuid_generate_time_safe** returns zero if the UUID has been generated in a safe manner, −1 otherwise.

CONFORMING TO

This library generates UUIDs compatible with OSF DCE 1.1, and hash based UUIDs V3 and V5 compatible with RFC-4122.

AUTHOR

Theodore Y. Ts'o

AVAILABILITY

libuuid is part of the util-linux package since version 2.15.1 and is available from https://www.kernel.org/pub/linux/utils/util-linux/.

SEE ALSO

uuidgen(1), **uuid**(3), **uuid_clear**(3), **uuid_compare**(3), **uuid_copy**(3), **uuid_is_null**(3), **uuid_parse**(3), **uuid_time**(3), **uuid_unparse**(3), **uuidd**(8)