Module
Process::UID

Provides a higher-level (and more portable) interface to the underlying operating system's concepts of real, effective, and saved user IDs. For more information, see the introduction to Process::GID on page 648.

Module methods

change_privilege Process::UID.change_privilege(*uid*) → *uid*

Sets the real, effective, and saved user IDs to *uid*, raising an exception on failure (in which case the state of the IDs is not known). Not compatible with Process.uid=.

eid Process::UID.eid → *euid*

Returns the effective user ID for this process. Synonym for Process.euid.

eid= Process::UID.eid = *euid*

Synonym for Process::UID.grant_privilege.

grant_privilege Process::UID.grant_privilege(*euid*) → *euid*

Sets the effective user ID to *euid*, raising an exception on failure. One some environments this may also change the saved user ID.

re_exchange Process::UID.re_exchange → *euid*

Exchanges the real and effective user IDs, setting the saved user ID to the new effective user ID. Returns the new effective user ID.

re_exchangeable? Process::UID.re_exchangeable → true or false

Returns true if real and effective user IDs can be exchanged on the host operating system and returns false otherwise.

rid Process::UID.rid → *uid*

Returns the real user ID for this process. Synonym for Process.uid.

sid_available? Process::UID.sid_available? → true or false

Returns true if the underlying platform supports saved user IDs and returns false otherwise. Currently, Ruby assumes support if the operating system has setresuid(2) or seteuid(2) calls or if the configuration includes the POSIX_SAVED_IDS flag.

switch Process::UID.switch → *euid*
Process::UID.switch { *block* } → *obj*

Handles the toggling of user privilege. In the block form, automatically toggles the IDs back when the block terminates (as long as the block doesn't use other Process::UID calls to interfere). Without a block, returns the original effective user ID.