

util-vserver (libvserver) Reference Manual
0.30.210

Generated by Doxygen 1.4.6

Mon Apr 10 17:27:39 2006

Contents

1 util-vserver (libvserver) Module Index	1
2 util-vserver (libvserver) Hierarchical Index	1
3 util-vserver (libvserver) Data Structure Index	2
4 util-vserver (libvserver) File Index	2
5 util-vserver (libvserver) Module Documentation	3
6 util-vserver (libvserver) Data Structure Documentation	9
7 util-vserver (libvserver) File Documentation	15

1 util-vserver (libvserver) Module Index

1.1 util-vserver (libvserver) Modules

Here is a list of all modules:

Syscall wrappers	3
Helper functions	7

2 util-vserver (libvserver) Hierarchical Index

2.1 util-vserver (libvserver) Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Mapping_uint32	9
Mapping_uint64	10
vc_ctx_caps	10
vc_ctx_dlimit	11
vc_ctx_flags	11
vc_err_listparser	11
vc_ip_mask_pair	12
vc_net_caps	12
vc_net_flags	13

vc_net_nx	13
vc_nx_info	13
vc_rlimit	13
vc_rlimit_mask	14
vc_set_sched	15
vc_vx_info	15

3 util-vserver (libvserver) Data Structure Index

3.1 util-vserver (libvserver) Data Structures

Here are the data structures with brief descriptions:

Mapping_uint32	9
Mapping_uint64	10
vc_ctx_caps (Capabilities of process-contexts)	10
vc_ctx_dlimit	11
vc_ctx_flags (Flags of process-contexts)	11
vc_err_listparser (Information about parsing errors)	11
vc_ip_mask_pair	12
vc_net_caps	12
vc_net_flags	13
vc_net_nx	13
vc_nx_info	13
vc_rlimit (The limits of a resources)	13
vc_rlimit_mask (Masks describing the supported limits)	14
vc_set_sched	15
vc_vx_info	15

4 util-vserver (libvserver) File Index

4.1 util-vserver (libvserver) File List

Here is a list of all documented files with brief descriptions:

[internal.h](#) (Declarations which are used by util-vserver internally) 15

[vserver.h](#) (The public interface of the the libvserver library) 16

5 util-vserver (libvserver) Module Documentation

5.1 Syscall wrappers

Functions

- [int vc_syscall](#) (uint32_t cmd, [xid_t](#) xid, void *data)
The generic vserver syscall
This function executes the generic vserver syscall. It uses the correct syscallnumber (which may differ between the different architectures).
- [int vc_get_version](#) ()
Returns the version of the current kernel API.
- [xid_t vc_new_s_context](#) ([xid_t](#) ctx, unsigned int remove_cap, unsigned int flags)
Moves current process into a context
Puts current process into context ctx, removes the capabilities given in remove_cap and sets flags.
- [int vc_set_ipv4root](#) (uint32_t bcast, size_t nb, struct [vc_ip_mask_pair](#) const *ips)
Sets the ipv4root information.
- [xid_t vc_ctx_create](#) ([xid_t](#) xid)
Creates a context without starting it.
This functions initializes a new context. When already in a freshly created context, this old context will be discarded.
- [int vc_ctx_migrate](#) ([xid_t](#) xid)
Moves the current process into the specified context.
- [int vc_get_rlimit](#) ([xid_t](#) xid, int resource, struct [vc_rlimit](#) *lim)
Returns the limits of resource.
- [int vc_set_rlimit](#) ([xid_t](#) xid, int resource, struct [vc_rlimit](#) const *lim)
Sets the limits of resource.
- [int vc_ctx_kill](#) ([xid_t](#) ctx, pid_t pid, int sig)
Sends a signal to a context/pid
Special values for pid are:
 - -1 which means every process in ctx except the init-process
 - 0 which means every process in ctx inclusive the init-process.
- [int vc_get_iattr](#) (char const *filename, [xid_t](#) *xid, uint_least32_t *flags, uint_least32_t *mask)
Returns information about attributes and assigned context of a file.
This function returns the VC_IATTR_XXX flags and about the assigned context of a file. To request an information, the appropriate bit in mask must be set and the corresponding parameter (xid or flags) must not be NULL.

- `xid_t vc_get_task_xid (pid_t pid)`
Returns the context of the given process.
- `xid_t vc_getfilecontext (char const *filename)`
*Returns the context of filename
This function calls `vc_get_iattr()` with appropriate arguments to determine the context of filename. In error-case or when no context is assigned, VC_NOCTX will be returned. To differ between both cases, `errno` must be examined.*
- `int vc_wait_exit (xid_t xid)`
Waits for the end of a context.

5.1.1 Detailed Description

Functions which are calling the vserver syscall directly.

5.1.2 Function Documentation

5.1.2.1 `xid_t vc_ctx_create (xid_t xid)`

Creates a context without starting it.

This functions initializes a new context. When already in a freshly created context, this old context will be discarded.

Parameters:

- xid* The new context; special values are:
- VC_DYNAMIC_XID which means to create a dynamic context

Returns:

the xid of the created context, or VC_NOCTX on errors. `errno` will be set appropriately.

5.1.2.2 `int vc_ctx_migrate (xid_t xid)`

Moves the current process into the specified context.

Parameters:

xid The new context

Returns:

0 on success, -1 on errors

5.1.2.3 `int vc_get_iattr (char const * filename, xid_t * xid, uint_least32_t * flags, uint_least32_t * mask)`

Returns information about attributes and assigned context of a file.

This function returns the VC_IATTR_XXX flags and about the assigned context of a file. To request an information, the appropriate bit in `mask` must be set and the corresponding parameter (*xid* or *flags*) must not be NULL.

E.g. to receive the assigned context, the VC_IATTR_XID bit must be set in *mask*, and *xid* must point to valid memory.

Possible flags are VC_IATTR_ADMIN, VC_IATTR_WATCH , VC_IATTR_HIDE, VC_IATTR_BARRIER, VC_IATTR_IUNLINK and VC_IATTR_IMMUTABLE.

Parameters:

filename The name of the file whose attributes shall be determined.

xid When non-zero and the VC_IATTR_XID bit is set in *mask*, the assigned context of *filename* will be stored there.

flags When non-zero, a bitmask of current attributes will be stored there. These attributes must be requested explicitly by setting the appropriate bit in *mask*

mask Points to a bitmask which tells which attributes shall be determined. On return, it will masquerade the attributes which were determined.

Precondition:

`mask!=0 && !((*mask&VC_IATTR_XID) && xid==0) && !((*mask&~VC_IATTR_XID) && flags==0)`

5.1.2.4 int vc_get_rlimit ([xid_t](#) xid, int resource, struct [vc_rlimit](#) * lim)

Returns the limits of *resource*.

Parameters:

xid The id of the context

resource The resource which will be queried

lim The result which will be filled with the limits

Returns:

0 on success, and -1 on errors.

5.1.2.5 [xid_t](#) vc_get_task_xid (pid_t pid)

Returns the context of the given process.

Parameters:

pid the process-id whose xid shall be determined; pid==0 means the current process.

Returns:

the xid of process *pid* or -1 on errors

5.1.2.6 int vc_get_version ()

Returns the version of the current kernel API.

Returns:

The versionnumber of the kernel API

5.1.2.7 `xid_t vc_getfilecontext (char const *filename)`

Returns the context of `filename`

This function calls `vc_get_iattr()` with appropriate arguments to determine the context of `filename`. In error-case or when no context is assigned, `VC_NOCTX` will be returned. To differ between both cases, `errno` must be examined.

WARNING: this function can modify `errno` although no error happened.

Parameters:

filename The file to check

Returns:

The assigned context, or `VC_NOCTX` when an error occurred or no such assignment exists. `errno` will be 0 in the latter case

5.1.2.8 `xid_t vc_new_s_context (xid_t ctx, unsigned int remove_cap, unsigned int flags)`

Moves current process into a context

Puts current process into context `ctx`, removes the capabilities given in `remove_cap` and sets `flags`.

Parameters:

ctx The new context; special values for are

- `VC_SAMECTX` which means the current context (just for changing caps and flags)
- `VC_DYNAMIC_XID` which means the next free context; this value can be used by ordinary users also

remove_cap The linux capabilities which will be **removed**.

flags Special flags which will be set.

Returns:

The new context-id, or `VC_NOCTX` on errors; `errno` will be set appropriately

See <http://vserver.13thfloor.at/Stuff/Logic.txt> for details

5.1.2.9 `int vc_set_ipv4root (uint32_t bcast, size_t nb, struct vc_ip_mask_pair const *ips)`

Sets the ipv4root information.

Precondition:

`nb < NB_IPV4ROOT && ips != 0`

5.1.2.10 `int vc_set_rlimit (xid_t xid, int resource, struct vc_rlimit const *lim)`

Sets the limits of *resource*.

Parameters:

xid The id of the context

resource The resource which will be queried

lim The new limits

Returns:

0 on success, and -1 on errors.

5.1.2.11 int vc_syscall (uint32_t cmd, xid_t xid, void * data)

The generic vserver syscall

This function executes the generic vserver syscall. It uses the correct syscallnumber (which may differ between the different architectures).

Parameters:

- cmd* the command to be executed
- xid* the xid on which the cmd shall be applied
- data* additional arguments; depends on cmd

Returns:

depends on cmd; usually, -1 stands for an error

5.2 Helper functions

Data Structures

- struct [vc_err_listparser](#)
Information about parsing errors.

Functions

- size_t [vc_get_nb_ipv4root](#) () VC_ATTR_CONST
Returns the value of NB_IPV4ROOT.
*This function returns the value of NB_IPV4ROOT which was used when the library was built, but **not** the value which is used by the currently running kernel.*
- bool [vc_parseLimit](#) (char const *str, vc_limit_t *res)
Parses a string describing a limit
This function parses str and interprets special words like "inf" or suffixes. Valid suffixes are
 - k ... 1000
 - m ... 1000000
 - K ... 1024
 - M ... 1048576.
- uint_least64_t [vc_text2bcap](#) (char const *str, size_t len)
Converts a single string into bcapability.
- char const * [vc_lobcap2text](#) (uint_least64_t *val)
Converts the lowest bit of a bcapability or the entire value (when possible) to a textual representation.
- int [vc_list2bcap](#) (char const *str, size_t len, struct [vc_err_listparser](#) *err, struct [vc_ctx_caps](#) *cap)
Converts a string into a bcapability-bitmask
Syntax of str:.

5.2.1 Detailed Description

Functions which are doing general helper tasks like parameter parsing.

5.2.2 Function Documentation

5.2.2.1 `int vc_list2bcap (char const * str, size_t len, struct vc_err_listparser * err, struct vc_ctx_caps * cap)`

Converts a string into a bcapability-bitmask

Syntax of *str*:

```
LIST    <- ELEM | ELEM ' , ' LIST
ELEM    <- '~' ELEM | MASK | NAME
MASK    <- NUMBER | '^' NUMBER
NUMBER  <- 0[0-7]* | [1-9][0-9]* | 0x[0-9,a-f]+
NAME    <- <literal name> | "all" | "any" | "none"
```

When the '~' prefix is used, the bits will be unset and a '~' after another '~' will cancel both ones. The '^' prefix specifies a bitnumber instead of a bitmask.

"literal name" is everything which will be accepted by the `vc_text2bcap()` function. The special values for NAME will be recognized case insensitively

Parameters:

str The string to be parsed

len The length of the string, or 0 for automatic detection

err Pointer to a structure for error-information, or NULL.

cap Pointer to a `vc_ctx_caps` structure holding the results; only the *bcaps* and *bmask* fields will be changed and already set values will not be honored. When an error occurred, *cap* will have the value of all processed valid BCAP parts.

Returns:

0 on success, -1 on error. In error case, *err* will hold position and length of the first not understood BCAP part

Precondition:

str != 0 && *cap* != 0; *cap*->*bcaps* and *cap*->*bmask* must be initialized

5.2.2.2 `char const* vc_lobcap2text (uint_least64_t * val)`

Converts the lowest bit of a bcapability or the entire value (when possible) to a textual representation.

Parameters:

val The string to be converted; on success, the detected bit(s) will be unset, in errorcase only the lowest set bit

Returns:

A textual representation of *val* resp. of its lowest set bit; or NULL in errorcase.

Precondition:

val != 0

Postcondition:

```
*valold != 0 <-> *valold > *valnew
*valold == 0 --> result == 0
```

5.2.2.3 bool vc_parseLimit (char const * *str*, [vc_limit_t](#) * *res*)

Parses a string describing a limit

This function parses *str* and interprets special words like "inf" or suffixes. Valid suffixes are

- k ... 1000
- m ... 1000000
- K ... 1024
- M ... 1048576.

Parameters:

str The string which shall be parsed

res Will be filled with the interpreted value; in errorcase, this value is undefined.

Returns:

true, iff the string *str* could be parsed. *res* will be filled with the interpreted value in this case.

Precondition:

str!=0 && *res*!=0

5.2.2.4 uint_least64_t vc_text2bcap (char const * *str*, size_t *len*)

Converts a single string into bcability.

Parameters:

str The string to be parsed; both "CAP_XXX" and "XXX" will be accepted

len The length of the string, or 0 for automatic detection

Returns:

0 on error; a bitmask on success

Precondition:

str != 0

6 util-vserver (libvserver) Data Structure Documentation

6.1 Mapping_uint32 Struct Reference

Data Fields

- char const *const [id](#)
- size_t [len](#)
- uint_least32_t [val](#)

6.1.1 Detailed Description

Definition at line 61 of file internal.h.

The documentation for this struct was generated from the following file:

- [internal.h](#)

6.2 Mapping_uint64 Struct Reference

Data Fields

- char const *const [id](#)
- size_t [len](#)
- uint_least64_t [val](#)

6.2.1 Detailed Description

Definition at line 67 of file internal.h.

The documentation for this struct was generated from the following file:

- [internal.h](#)

6.3 vc_ctx_caps Struct Reference

Capabilities of process-contexts.

```
#include <vserver.h>
```

Data Fields

- uint_least64_t [bcaps](#)
Mask of set common system capabilities.
- uint_least64_t [bmask](#)
Mask of set and unset common system capabilities when used by set operations, or the modifiable capabilities when used by get operations.
- uint_least64_t [ccaps](#)
Mask of set process context capabilities.
- uint_least64_t [cmask](#)
Mask of set and unset process context capabilities when used by set operations, or the modifiable capabilities when used by get operations.

6.3.1 Detailed Description

Capabilities of process-contexts.

Definition at line 515 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.4 vc_ctx_dlimit Struct Reference

Data Fields

- uint_least32_t [space_used](#)
- uint_least32_t [space_total](#)
- uint_least32_t [inodes_used](#)
- uint_least32_t [inodes_total](#)
- uint_least32_t [reserved](#)

6.4.1 Detailed Description

Definition at line 688 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.5 vc_ctx_flags Struct Reference

Flags of process-contexts.

```
#include <vserver.h>
```

Data Fields

- uint_least64_t [flagword](#)
Mask of set context flags.
- uint_least64_t [mask](#)
Mask of set and unset context flags when used by set operations, or modifiable flags when used by get operations.

6.5.1 Detailed Description

Flags of process-contexts.

Definition at line 505 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.6 vc_err_listparser Struct Reference

Information about parsing errors.

```
#include <vserver.h>
```

Data Fields

- `char const * ptr`
Pointer to the first character of an erroneous string.
- `size_t len`
Length of the erroneous string.

6.6.1 Detailed Description

Information about parsing errors.

Definition at line 533 of file `vserver.h`.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.7 `vc_ip_mask_pair` Struct Reference

Data Fields

- `uint32_t ip`
- `uint32_t mask`

6.7.1 Detailed Description

Definition at line 233 of file `vserver.h`.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.8 `vc_net_caps` Struct Reference

Data Fields

- `uint_least64_t ncaps`
- `uint_least64_t cmask`

6.8.1 Detailed Description

Definition at line 426 of file `vserver.h`.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.9 vc_net_flags Struct Reference

Data Fields

- uint_least64_t [flagword](#)
- uint_least64_t [mask](#)

6.9.1 Detailed Description

Definition at line 417 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.10 vc_net_nx Struct Reference

Data Fields

- [vc_net_nx_type](#) type
- size_t [count](#)
- uint32_t [ip](#) [4]
- uint32_t [mask](#) [4]

6.10.1 Detailed Description

Definition at line 404 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.11 vc_nx_info Struct Reference

Data Fields

- [nid_t](#) [nid](#)

6.11.1 Detailed Description

Definition at line 393 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.12 vc_rlimit Struct Reference

The limits of a resources.

```
#include <vserver.h>
```

Data Fields

- [vc_limit_t min](#)
the guaranteed minimum of a resources
- [vc_limit_t soft](#)
the softlimit of a resource
- [vc_limit_t hard](#)
the absolute hardlimit of a resource

6.12.1 Detailed Description

The limits of a resources.

This is a triple consisting of a minimum, soft and hardlimit.

Definition at line 327 of file `vserver.h`.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.13 `vc_rlimit_mask` Struct Reference

Masks describing the supported limits.

```
#include <vserver.h>
```

Data Fields

- [uint_least32_t min](#)
masks the resources supporting a minimum limit
- [uint_least32_t soft](#)
masks the resources supporting a soft limit
- [uint_least32_t hard](#)
masks the resources supporting a hard limit

6.13.1 Detailed Description

Masks describing the supported limits.

Definition at line 334 of file `vserver.h`.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.14 vc_set_sched Struct Reference

Data Fields

- uint_least32_t [set_mask](#)
- int_least32_t [fill_rate](#)
- int_least32_t [interval](#)
- int_least32_t [tokens](#)
- int_least32_t [tokens_min](#)
- int_least32_t [tokens_max](#)
- int_least32_t [priority_bias](#)

6.14.1 Detailed Description

Definition at line 675 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

6.15 vc_vx_info Struct Reference

Data Fields

- [xid_t](#) [xid](#)
- [pid_t](#) [initpid](#)

6.15.1 Detailed Description

Definition at line 470 of file vserver.h.

The documentation for this struct was generated from the following file:

- [vserver.h](#)

7 util-vserver (libvserver) File Documentation

7.1 internal.h File Reference

Declarations which are used by util-vserver internally.

```
#include "fmt.h"
```

```
#include "vserver.h"
```

```
#include <stdlib.h>
```

```
#include <stdbool.h>
```

Include dependency graph for internal.h:

Data Structures

- struct [Mapping_uint32](#)
- struct [Mapping_uint64](#)

Functions

- char * **vc_getVserverByCtx_Internal** (xid_t ctx, vcCfgStyle *style, char const *revdir, bool validate_result)
- int **utilvserver_checkCompatVersion** ()
- bool **utilvserver_isDirectory** (char const *path, bool follow_link)
- bool **utilvserver_isFile** (char const *path, bool follow_link)
- bool **utilvserver_isLink** (char const *path)
- int **utilvserver_listparser_uint32** (char const *str, size_t len, char const **err_ptr, size_t *err_len, uint_least32_t *flag, uint_least32_t *mask, uint_least32_t(*func)(char const *, size_t, bool *)) NONNULL((1
- int **utilvserver_listparser_uint64** (char const *str, size_t len, char const **err_ptr, size_t *err_len, uint_least64_t *flag, uint_least64_t *mask, uint_least64_t(*func)(char const *, size_t, bool *)) NONNULL((1
- ssize_t **utilvserver_value2text_uint32** (char const *str, size_t len, struct [Mapping_uint32](#) const *map, size_t map_len) NONNULL((1
- ssize_t **utilvserver_value2text_uint64** (char const *str, size_t len, struct [Mapping_uint64](#) const *map, size_t map_len) NONNULL((1
- ssize_t **utilvserver_text2value_uint32** (uint_least32_t *val, struct [Mapping_uint32](#) const *map, size_t map_len) NONNULL((1
- ssize_t **utilvserver_text2value_uint64** (uint_least64_t *val, struct [Mapping_uint64](#) const *map, size_t map_len) NONNULL((1

7.1.1 Detailed Description

Declarations which are used by util-vserver internally.

Definition in file [internal.h](#).

7.2 vserver.h File Reference

The public interface of the the libvserver library.

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/types.h>
```

Include dependency graph for vserver.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vc_ip_mask_pair](#)
- struct [vc_rlimit](#)

The limits of a resources.

- struct [vc_rlimit_mask](#)

Masks describing the supported limits.

- struct [vc_nx_info](#)
- struct [vc_net_nx](#)
- struct [vc_net_flags](#)
- struct [vc_net_caps](#)
- struct [vc_vx_info](#)
- struct [vc_ctx_flags](#)

Flags of process-contexts.

- struct [vc_ctx_caps](#)

Capabilities of process-contexts.

- struct [vc_err_listparser](#)

Information about parsing errors.

- struct [vc_set_sched](#)
- struct [vc_ctx_dlimit](#)

Defines

- #define [VC_NOCTX](#) ((xid_t)(-1))
- #define [VC_NOXID](#) ((xid_t)(-1))
- #define [VC_DYNAMIC_XID](#) ((xid_t)(-1))
- #define [VC_SAMECTX](#) ((xid_t)(-2))
- #define [VC_NONID](#) ((nid_t)(-1))
- #define [VC_DYNAMIC_NID](#) ((nid_t)(-1))
- #define [VC_LIM_INFINITY](#) (~0ULL)
- #define [VC_LIM_KEEP](#) (~1ULL)
- #define [VC_CDLIM_UNSET](#) (0U)
- #define [VC_CDLIM_INFINITY](#) (~0U)
- #define [VC_CDLIM_KEEP](#) (~1U)
- #define [S_CTX_INFO_LOCK](#) 1
- #define [S_CTX_INFO_SCHED](#) 2
- #define [S_CTX_INFO_NPROC](#) 4
- #define [S_CTX_INFO_PRIVATE](#) 8
- #define [S_CTX_INFO_INIT](#) 16
- #define [S_CTX_INFO_HIDEINFO](#) 32
- #define [S_CTX_INFO_ULIMIT](#) 64
- #define [S_CTX_INFO_NAMESPACE](#) 128
- #define [VC_CAP_CHOWN](#) 0
- #define [VC_CAP_DAC_OVERRIDE](#) 1
- #define [VC_CAP_DAC_READ_SEARCH](#) 2
- #define [VC_CAP_FOWNER](#) 3
- #define [VC_CAP_FSETID](#) 4
- #define [VC_CAP_KILL](#) 5
- #define [VC_CAP_SETGID](#) 6

- `#define VC_CAP_SETUID` 7
- `#define VC_CAP_SETPCAP` 8
- `#define VC_CAP_LINUX_IMMUTABLE` 9
- `#define VC_CAP_NET_BIND_SERVICE` 10
- `#define VC_CAP_NET_BROADCAST` 11
- `#define VC_CAP_NET_ADMIN` 12
- `#define VC_CAP_NET_RAW` 13
- `#define VC_CAP_IPC_LOCK` 14
- `#define VC_CAP_IPC_OWNER` 15
- `#define VC_CAP_SYS_MODULE` 16
- `#define VC_CAP_SYS_RAWIO` 17
- `#define VC_CAP_SYS_CHROOT` 18
- `#define VC_CAP_SYS_PTRACE` 19
- `#define VC_CAP_SYS_PACCT` 20
- `#define VC_CAP_SYS_ADMIN` 21
- `#define VC_CAP_SYS_BOOT` 22
- `#define VC_CAP_SYS_NICE` 23
- `#define VC_CAP_SYS_RESOURCE` 24
- `#define VC_CAP_SYS_TIME` 25
- `#define VC_CAP_SYS_TTY_CONFIG` 26
- `#define VC_CAP_MKNOD` 27
- `#define VC_CAP_LEASE` 28
- `#define VC_CAP_AUDIT_WRITE` 29
- `#define VC_CAP_AUDIT_CONTROL` 30
- `#define VC_IMMUTABLE_FILE_FL` 0x0000010lu
- `#define VC_IMMUTABLE_LINK_FL` 0x0008000lu
- `#define VC_IMMUTABLE_ALL` (VC_IMMUTABLE_LINK_FL|VC_IMMUTABLE_FILE_FL)
- `#define VC_IATTR_XID` 0x01000000u
- `#define VC_IATTR_ADMIN` 0x00000001u
- `#define VC_IATTR_WATCH` 0x00000002u
- `#define VC_IATTR_HIDE` 0x00000004u
- `#define VC_IATTR_FLAGS` 0x00000007u
- `#define VC_IATTR_BARRIER` 0x00010000u
- `#define VC_IATTR_IUNLINK` 0x00020000u
- `#define VC_IATTR_IMMUTABLE` 0x00040000u
- `#define VC_VXF_INFO_LOCK` 0x00000001ull
- `#define VC_VXF_INFO_NPROC` 0x00000004ull
- `#define VC_VXF_INFO_PRIVATE` 0x00000008ull
- `#define VC_VXF_INFO_INIT` 0x00000010ull
- `#define VC_VXF_INFO_HIDEINFO` 0x00000020ull
- `#define VC_VXF_INFO_ULIMIT` 0x00000040ull
- `#define VC_VXF_INFO_NAMESPACE` 0x00000080ull
- `#define VC_VXF_SCHED_HARD` 0x00000100ull
- `#define VC_VXF_SCHED_PRIO` 0x00000200ull
- `#define VC_VXF_SCHED_PAUSE` 0x00000400ull
- `#define VC_VXF_VIRT_MEM` 0x00010000ull
- `#define VC_VXF_VIRT_UPTIME` 0x00020000ull
- `#define VC_VXF_VIRT_CPU` 0x00040000ull
- `#define VC_VXF_VIRT_LOAD` 0x00080000ull
- `#define VC_VXF_HIDE_MOUNT` 0x01000000ull

- `#define VC_VXF_HIDE_NETIF 0x02000000ull`
- `#define VC_VXF_STATE_SETUP (1ULL<<32)`
- `#define VC_VXF_STATE_INIT (1ULL<<33)`
- `#define VC_VXF_FORK_RSS (1ULL<<48)`
- `#define VC_VXF_PROLIFIC (1ULL<<49)`
- `#define VC_VXF_IGNEG_NICE (1ULL<<52)`
- `#define VC_VXC_SET_UTSNAME 0x00000001ull`
- `#define VC_VXC_SET_RLIMIT 0x00000002ull`
- `#define VC_VXC_RAW_ICMP 0x00000100ull`
- `#define VC_VXC_SYSLOG 0x00001000ull`
- `#define VC_VXC_SECURE_MOUNT 0x00010000ull`
- `#define VC_VXC_SECURE_REMOUNT 0x00020000ull`
- `#define VC_VXC_BINARY_MOUNT 0x00040000ull`
- `#define VC_VXC_QUOTA_CTL 0x00100000ull`
- `#define VC_VXSM_FILL_RATE 0x0001`
- `#define VC_VXSM_INTERVAL 0x0002`
- `#define VC_VXSM_TOKENS 0x0010`
- `#define VC_VXSM_TOKENS_MIN 0x0020`
- `#define VC_VXSM_TOKENS_MAX 0x0040`
- `#define VC_VXSM_PRIO_BIAS 0x0100`
- `#define VC_BAD_PERSONALITY ((uint_least32_t)(-1))`
- `#define VC_LIMIT_VSERVER_NAME_LEN 1024`
- `#define vcSKEL_INTERFACES 1u`
- `#define vcSKEL_PKGMGMT 2u`
- `#define vcSKEL_FILESYSTEM 4u`

Typedefs

- `typedef an_unsigned_integer_type xid_t`
- `typedef an_unsigned_integer_type nid_t`
- `typedef uint_least64_t vc_limit_t`

The type which is used for a single limit value.

Enumerations

- `enum vc_net_nx_type {`
`vcNET_IPV4 = 1, vcNET_IPV6 = 2, vcNET_IPV4B = 0x101, vcNET_IPV6B = 0x102,`
`vcNET_ANY = ~0 }`
- `enum vc_uts_type {`
`vcVHI_CONTEXT, vcVHI_SYSNAME, vcVHI_NODENAME, vcVHI_RELEASE,`
`vcVHI_VERSION, vcVHI_MACHINE, vcVHI_DOMAINNAME }`
- `enum vcFeatureSet {`
`vcFEATURE_VKILL, vcFEATURE_IATTR, vcFEATURE_RLIMIT, vcFEATURE_-`
`COMPAT,`
`vcFEATURE_MIGRATE, vcFEATURE_NAMESPACE, vcFEATURE_SCHED, vc-`
`FEATURE_VINFO,`
`vcFEATURE_VHI, vcFEATURE_VSHELPER0, vcFEATURE_VSHELPER, vcFEATURE_-`
`VWAIT,`
`vcFEATURE_VNET }`

- enum `vcXidType` {
`vcTYPE_INVALID`, `vcTYPE_MAIN`, `vcTYPE_WATCH`, `vcTYPE_STATIC`,
`vcTYPE_DYNAMIC` }
- enum `vcCfgStyle` {
`vcCFG_NONE`, `vcCFG_AUTO`, `vcCFG_LEGACY`, `vcCFG_RECENT_SHORT`,
`vcCFG_RECENT_FULL` }

Functions

- int `vc_syscall` (uint32_t cmd, `xid_t` xid, void *data)
The generic vserver syscall
This function executes the generic vserver syscall. It uses the correct syscallnumber (which may differ between the different architectures).
- int `vc_get_version` ()
Returns the version of the current kernel API.
- `xid_t` `vc_new_s_context` (`xid_t` ctx, unsigned int remove_cap, unsigned int flags)
Moves current process into a context
Puts current process into context ctx, removes the capabilities given in remove_cap and sets flags.
- int `vc_set_ipv4root` (uint32_t bcast, size_t nb, struct `vc_ip_mask_pair` const *ips)
Sets the ipv4root information.
- size_t `vc_get_nb_ipv4root` () VC_ATTR_CONST
Returns the value of NB_IPV4ROOT.
*This function returns the value of NB_IPV4ROOT which was used when the library was built, but **not** the value which is used by the currently running kernel.*
- `xid_t` `vc_ctx_create` (`xid_t` xid)
Creates a context without starting it.
This functions initializes a new context. When already in a freshly created context, this old context will be discarded.
- int `vc_ctx_migrate` (`xid_t` xid)
Moves the current process into the specified context.
- int `vc_get_rlimit` (`xid_t` xid, int resource, struct `vc_rlimit` *lim)
Returns the limits of resource.
- int `vc_set_rlimit` (`xid_t` xid, int resource, struct `vc_rlimit` const *lim)
Sets the limits of resource.
- int `vc_get_rlimit_mask` (`xid_t` xid, struct `vc_rlimit_mask` *lim)
- bool `vc_parseLimit` (char const *str, `vc_limit_t` *res)
Parses a string describing a limit
This function parses str and interprets special words like "inf" or suffixes. Valid suffixes are
 - k ... 1000
 - m ... 1000000
 - K ... 1024

– M ... 1048576.

- `int vc_ctx_kill (xid_t ctx, pid_t pid, int sig)`

Sends a signal to a context/pid

Special values for pid are:

- -1 which means every process in ctx except the init-process
- 0 which means every process in ctx inclusive the init-process.

- `nid_t vc_get_task_nid (pid_t pid)`
- `int vc_get_nx_info (nid_t nid, struct vc_nx_info *)`
- `nid_t vc_net_create (nid_t nid)`
- `int vc_net_migrate (nid_t nid)`
- `int vc_net_add (nid_t nid, struct vc_net_nx const *info)`
- `int vc_net_remove (nid_t nid, struct vc_net_nx const *info)`
- `int vc_get_nflags (nid_t, struct vc_net_flags *)`
- `int vc_set_nflags (nid_t, struct vc_net_flags const *)`
- `int vc_get_ncaps (nid_t, struct vc_net_caps *)`
- `int vc_set_ncaps (nid_t, struct vc_net_caps const *)`
- `int vc_set_iattr (char const *filename, xid_t xid, uint_least32_t flags, uint_least32_t mask)`
- `int vc_get_iattr (char const *filename, xid_t *xid, uint_least32_t *flags, uint_least32_t *mask)`

Returns information about attributes and assigned context of a file.

This function returns the VC_IATTR_XXX flags and about the assigned context of a file. To request an information, the appropriate bit in mask must be set and the corresponding parameter (xid or flags) must not be NULL.

- `xid_t vc_get_task_xid (pid_t pid)`

Returns the context of the given process.

- `int vc_get_vx_info (xid_t xid, struct vc_vx_info *info)`
- `int vc_set_vhi_name (xid_t xid, vc_uts_type type, char const *val, size_t len)`
- `int vc_get_vhi_name (xid_t xid, vc_uts_type type, char *val, size_t len)`
- `bool vc_is_dynamic_xid (xid_t xid)`
- `int vc_enter_namespace (xid_t xid)`
- `int vc_set_namespace ()`
- `int vc_cleanup_namespace ()`
- `int vc_get_cflags (xid_t xid, struct vc_ctx_flags *)`
- `int vc_set_cflags (xid_t xid, struct vc_ctx_flags const *)`
- `int vc_get_ccaps (xid_t xid, struct vc_ctx_caps *)`
- `int vc_set_ccaps (xid_t xid, struct vc_ctx_caps const *)`
- `uint_least64_t vc_text2bcap (char const *str, size_t len)`

Converts a single string into bcapability.

- `char const * vc_lobcap2text (uint_least64_t *val)`

Converts the lowest bit of a bcapability or the entire value (when possible) to a textual representation.

- `int vc_list2bcap (char const *str, size_t len, struct vc_err_listparser *err, struct vc_ctx_caps *cap)`

Converts a string into a bcapability-bitmask

Syntax of str:.

- `uint_least64_t vc_text2ccap (char const *, size_t len)`
- `char const * vc_loccap2text (uint_least64_t *)`

- int **vc_list2ccap** (char const *, size_t len, struct **vc_err_listparser** *err, struct **vc_ctx_caps** *)
- int **vc_list2cflag** (char const *, size_t len, struct **vc_err_listparser** *err, struct **vc_ctx_flags** *flags)
- uint_least64_t **vc_text2cflag** (char const *, size_t len)
- char const * **vc_locflag2text** (uint_least64_t *)
- uint_least32_t **vc_list2cflag_compat** (char const *, size_t len, struct **vc_err_listparser** *err)
- uint_least32_t **vc_text2cflag_compat** (char const *, size_t len)
- char const * **vc_hicflag2text_compat** (uint_least32_t)
- int **vc_text2cap** (char const *)
- char const * **vc_cap2text** (unsigned int)
- int **vc_list2nflag** (char const *, size_t len, struct **vc_err_listparser** *err, struct **vc_net_flags** *flags)
- uint_least64_t **vc_text2nflag** (char const *, size_t len)
- char const * **vc_lonflag2text** (uint_least64_t *)
- uint_least64_t **vc_text2ncap** (char const *, size_t len)
- char const * **vc_loncap2text** (uint_least64_t *)
- int **vc_list2ncap** (char const *, size_t len, struct **vc_err_listparser** *err, struct **vc_net_caps** *)
- uint_least64_t **vc_get_insecurebcaps** () VC_ATTR_CONST
- uint_least32_t **vc_text2personalityflag** (char const *str, size_t len)
- char const * **vc_lopersonality2text** (uint_least32_t *)
- int **vc_list2personalityflag** (char const *, size_t len, uint_least32_t *personality, struct **vc_err_listparser** *err)
- uint_least32_t **vc_str2personalitytype** (char const *, size_t len)
- **xid_t** **vc_getfilecontext** (char const *filename)

Returns the context of filename

*This function calls **vc_get_iattr()** with appropriate arguments to determine the context of filename. In error-case or when no context is assigned, VC_NOCTX will be returned. To differ between both cases, **errno** must be examined.*

- int **vc_set_sched** (**xid_t** xid, struct **vc_set_sched** const *)
- int **vc_add_dlimit** (char const *filename, **xid_t** xid, uint_least32_t flags)
- int **vc_rem_dlimit** (char const *filename, **xid_t** xid, uint_least32_t flags)
- int **vc_set_dlimit** (char const *filename, **xid_t** xid, uint_least32_t flags, struct **vc_ctx_dlimit** const *limits)
- int **vc_get_dlimit** (char const *filename, **xid_t** xid, uint_least32_t flags, struct **vc_ctx_dlimit** *limits)
- int **vc_wait_exit** (**xid_t** xid)

Waits for the end of a context.

- bool **vc_isSupported** (**vcFeatureSet**) VC_ATTR_CONST
- bool **vc_isSupportedString** (char const *)
- **vcXidType** **vc_getXIDType** (**xid_t** xid) VC_ATTR_CONST
- **xid_t** **vc_xidopt2xid** (char const *, bool honor_static, char const **err_info)
- **vcCfgStyle** **vc_getVserverCfgStyle** (char const *id)
- char * **vc_getVserverName** (char const *id, **vcCfgStyle** style)
- char * **vc_getVserverCfgDir** (char const *id, **vcCfgStyle** style)
- char * **vc_getVserverAppDir** (char const *id, **vcCfgStyle** style, char const *app)
- char * **vc_getVserverVdir** (char const *id, **vcCfgStyle** style, bool physical)
- **xid_t** **vc_getVserverCtx** (char const *id, **vcCfgStyle** style, bool honor_static, bool *is_running)
- char * **vc_getVserverByCtx** (**xid_t** ctx, **vcCfgStyle** *style, char const *revdir)
- int **vc_compareVserverById** (char const *lhs, **vcCfgStyle** lhs_style, char const *rhs, **vcCfgStyle** rhs_style)
- int **vc_createSkeleton** (char const *id, **vcCfgStyle** style, int flags)

7.2.1 Detailed Description

The public interface of the the libvserver library.

Definition in file [vserver.h](#).

7.2.2 Define Documentation

7.2.2.1 `#define VC_DYNAMIC_XID ((xid_t)(-1))`

the value which means a random (the next free) ctx

Definition at line 65 of file [vserver.h](#).

7.2.2.2 `#define VC_NOCTX ((xid_t)(-1))`

the value which is returned in error-case (no ctx found)

Definition at line 62 of file [vserver.h](#).

7.2.2.3 `#define VC_SAMECTX ((xid_t)(-2))`

the value which means the current ctx

Definition at line 67 of file [vserver.h](#).

7.2.3 Typedef Documentation

7.2.3.1 `typedef uint_least64_t vc_limit_t`

The type which is used for a single limit value.

Special values are

- `VC_LIM_INFINITY` ... which is the infinite value
- `VC_LIM_KEEP` ... which is used to mark values which shall not be modified by the [vc_set_rlimit\(\)](#) operation.

Else, the interpretation of the value depends on the corresponding resource; it might be bytes, pages, seconds or litres of beer.

Definition at line 322 of file [vserver.h](#).

7.2.3.2 `an_unsigned_integer_type xid_t`

The identifier of a context.

Definition at line 225 of file [vserver.h](#).

7.2.4 Function Documentation

7.2.4.1 `int vc_add_dlimit (char const * filename, xid_t xid, uint_least32_t flags)`

Add a disk limit to a file system.

7.2.4.2 int vc_createSkeleton (char const * *id*, *vcCfgStyle* style, int *flags*)

Create a basic configuration skeleton for a vserver plus toplevel directories for pkgmanagment and filesystem (when requested).

7.2.4.3 int vc_get_dlimit (char const * *filename*, *xid_t* *xid*, uint_least32_t *flags*, struct *vc_ctx_dlimit* * *limits*)

Get a disk limit.

7.2.4.4 char* vc_getVserverAppDir (char const * *id*, *vcCfgStyle* style, char const * *app*)

Returns the path of the configuration directory for the given application. The result will be allocated and must be freed by the caller.

7.2.4.5 char* vc_getVserverByCtx (*xid_t* *ctx*, *vcCfgStyle* * *style*, char const * *revdir*)

Resolves the cfg-path of the vserver owning the given ctx. 'revdir' will be used as the directory holding the mapping-links; when NULL, the default value will be assumed. The result will be allocated and must be freed by the caller.

7.2.4.6 char* vc_getVserverCfgDir (char const * *id*, *vcCfgStyle* style)

Returns the path of the vserver configuration directory. When the given vserver does not exist, or when it does not have such a directory, NULL will be returned. Else, the result will be allocated and must be freed by the caller.

7.2.4.7 *xid_t* vc_getVserverCtx (char const * *id*, *vcCfgStyle* style, bool *honor_static*, bool * *is_running*)

Returns the ctx of the given vserver. When vserver is not running and 'honor_static' is false, VC_NOCTX will be returned. Else, when 'honor_static' is true and a static assignment exists, those value will be returned. Else, the result will be VC_NOCTX.

When 'is_running' is not null, the status of the vserver will be assigned to this variable.

7.2.4.8 char* vc_getVserverName (char const * *id*, *vcCfgStyle* style)

Resolves the name of the vserver. The result will be allocated and must be freed by the caller.

7.2.4.9 char* vc_getVserverVdir (char const * *id*, *vcCfgStyle* style, bool *physical*)

Returns the path to the vserver root-directory. The result will be allocated and must be freed by the caller.

7.2.4.10 bool vc_is_dynamic_xid (*xid_t* *xid*)

Returns true iff *xid* is a dynamic xid

7.2.4.11 int vc_rem_dlimit (char const * *filename*, *xid_t* *xid*, uint_least32_t *flags*)

Remove a disk limit from a file system.

7.2.4.12 `int vc_set_dlimit (char const *filename, xid_t xid, uint_least32_t flags, struct vc_ctx_dlimit const * limits)`

Set a disk limit.

7.2.4.13 `xid_t vc_xidopt2xid (char const *, bool honor_static, char const ** err_info)`

Maps an xid given at '-xid' options to an xid_t

Index

helper

- vc_list2bcap, 8
- vc_lobcap2text, 8
- vc_parseLimit, 8
- vc_text2bcap, 9

Helper functions, 7

internal.h, 15

Mapping_uint32, 9

Mapping_uint64, 10

Syscall wrappers, 3

syscalls

- vc_ctx_create, 4
- vc_ctx_migrate, 4
- vc_get_iattr, 4
- vc_get_rlimit, 5
- vc_get_task_xid, 5
- vc_get_version, 5
- vc_getfilecontext, 5
- vc_new_s_context, 6
- vc_set_ipv4root, 6
- vc_set_rlimit, 6
- vc_syscall, 6

vc_add_dlimit vserver.h, 23

vc_createSkeleton vserver.h, 23

vc_ctx_caps, 10

vc_ctx_create syscalls, 4

vc_ctx_dlimit, 11

vc_ctx_flags, 11

vc_ctx_migrate syscalls, 4

VC_DYNAMIC_XID vserver.h, 23

vc_err_listparser, 11

vc_get_dlimit vserver.h, 24

vc_get_iattr syscalls, 4

vc_get_rlimit syscalls, 5

vc_get_task_xid syscalls, 5

vc_get_version syscalls, 5

vc_getfilecontext

syscalls, 5

vc_getVserverAppDir vserver.h, 24

vc_getVserverByCtx vserver.h, 24

vc_getVserverCfgDir vserver.h, 24

vc_getVserverCtx vserver.h, 24

vc_getVserverName vserver.h, 24

vc_getVserverVdir vserver.h, 24

vc_ip_mask_pair, 12

vc_is_dynamic_xid vserver.h, 24

vc_limit_t vserver.h, 23

vc_list2bcap helper, 8

vc_lobcap2text helper, 8

vc_net_caps, 12

vc_net_flags, 13

vc_net_nx, 13

vc_new_s_context syscalls, 6

VC_NOCTX vserver.h, 23

vc_nx_info, 13

vc_parseLimit helper, 8

vc_rem_dlimit vserver.h, 24

vc_rlimit, 13

vc_rlimit_mask, 14

VC_SAMECTX vserver.h, 23

vc_set_dlimit vserver.h, 24

vc_set_ipv4root syscalls, 6

vc_set_rlimit syscalls, 6

vc_set_sched, 15

vc_syscall syscalls, 6

vc_text2bcap helper, 9

vc_vx_info, 15

vc_xidopt2xid
 vserver.h, [25](#)
vserver.h, [16](#)
 vc_add_dlimit, [23](#)
 vc_createSkeleton, [23](#)
 VC_DYNAMIC_XID, [23](#)
 vc_get_dlimit, [24](#)
 vc_getVserverAppDir, [24](#)
 vc_getVserverByCtx, [24](#)
 vc_getVserverCfgDir, [24](#)
 vc_getVserverCtx, [24](#)
 vc_getVserverName, [24](#)
 vc_getVserverVdir, [24](#)
 vc_is_dynamic_xid, [24](#)
 vc_limit_t, [23](#)
 VC_NOCTX, [23](#)
 vc_rem_dlimit, [24](#)
 VC_SAMECTX, [23](#)
 vc_set_dlimit, [24](#)
 vc_xidopt2xid, [25](#)
 xid_t, [23](#)

xid_t
 vserver.h, [23](#)