

util-vserver (libvserver) Reference Manual  
0.30.210

Generated by Doxygen 1.4.3

Sun Apr 23 22:13:41 2006

## Contents

<a href="#">1</a>	<a href="#">util-vserver (libvserver) Module Index</a>	<a href="#">1</a>
<a href="#">2</a>	<a href="#">util-vserver (libvserver) Data Structure Index</a>	<a href="#">1</a>
<a href="#">3</a>	<a href="#">util-vserver (libvserver) File Index</a>	<a href="#">2</a>
<a href="#">4</a>	<a href="#">util-vserver (libvserver) Module Documentation</a>	<a href="#">2</a>
<a href="#">5</a>	<a href="#">util-vserver (libvserver) Data Structure Documentation</a>	<a href="#">9</a>
<a href="#">6</a>	<a href="#">util-vserver (libvserver) File Documentation</a>	<a href="#">15</a>

## 1 util-vserver (libvserver) Module Index

### 1.1 util-vserver (libvserver) Modules

Here is a list of all modules:

<a href="#">Syscall wrappers</a>	<a href="#">2</a>
<a href="#">Helper functions</a>	<a href="#">6</a>

## 2 util-vserver (libvserver) Data Structure Index

### 2.1 util-vserver (libvserver) Data Structures

Here are the data structures with brief descriptions:

<a href="#">Mapping_uint32</a>	<a href="#">9</a>
<a href="#">Mapping_uint64</a>	<a href="#">9</a>
<a href="#">vc_ctx_caps (Capabilities of process-contexts )</a>	<a href="#">9</a>
<a href="#">vc_ctx_dlimit</a>	<a href="#">10</a>
<a href="#">vc_ctx_flags (Flags of process-contexts )</a>	<a href="#">10</a>
<a href="#">vc_err_listparser (Information about parsing errors )</a>	<a href="#">11</a>
<a href="#">vc_ip_mask_pair</a>	<a href="#">11</a>
<a href="#">vc_net_caps</a>	<a href="#">12</a>
<a href="#">vc_net_flags</a>	<a href="#">12</a>
<a href="#">vc_net_nx</a>	<a href="#">12</a>

<a href="#">vc_nx_info</a>	12
<a href="#">vc_rlimit</a> (The limits of a resources )	13
<a href="#">vc_rlimit_mask</a> (Masks describing the supported limits )	13
<a href="#">vc_set_sched</a>	14
<a href="#">vc_vx_info</a>	14

## 3 util-vserver (libvserver) File Index

### 3.1 util-vserver (libvserver) File List

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

<a href="#">internal.h</a> (Declarations which are used by util-vserver internally )	15
<a href="#">vserver.h</a> (The public interface of the the libvserver library )	16

## 4 util-vserver (libvserver) Module Documentation

### 4.1 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.*

### 4.1.1 Detailed Description

Functions which are calling the vserver syscall directly.

### 4.1.2 Function Documentation

#### 4.1.2.1 `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.

**4.1.2.2 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

**4.1.2.3 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)`

**4.1.2.4 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.

#### 4.1.2.5 `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

#### 4.1.2.6 `int vc_get_version ()`

Returns the version of the current kernel API.

**Returns:**

The versionnumber of the kernel API

#### 4.1.2.7 `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

#### 4.1.2.8 `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

**4.1.2.9** `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

**4.1.2.10** `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.

**4.1.2.11** `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

## 4.2 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`:*

### 4.2.1 Detailed Description

Functions which are doing general helper tasks like parameter parsing.

### 4.2.2 Function Documentation

#### 4.2.2.1 `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

**4.2.2.2 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:**

*\*val<sub>old</sub>* != 0 <-> *\*val<sub>old</sub>* > *\*val<sub>new</sub>*  
*\*val<sub>old</sub>* == 0 --> *result* == 0

**4.2.2.3 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

**4.2.2.4 uint\_least64\_t vc\_text2bcap (char const \* str, size\_t len)**

Converts a single string into bcapability.

**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

## 5 util-vserver (libvserver) Data Structure Documentation

### 5.1 Mapping\_uint32 Struct Reference

**Data Fields**

- char const \*const [id](#)
- size\_t [len](#)
- uint\_least32\_t [val](#)

#### 5.1.1 Detailed Description

Definition at line 61 of file internal.h.

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

- [internal.h](#)

### 5.2 Mapping\_uint64 Struct Reference

**Data Fields**

- char const \*const [id](#)
- size\_t [len](#)
- uint\_least64\_t [val](#)

#### 5.2.1 Detailed Description

Definition at line 67 of file internal.h.

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

- [internal.h](#)

### 5.3 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.*

#### 5.3.1 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](#)

## 5.4 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`

#### 5.4.1 Detailed Description

Definition at line 688 of file `vserver.h`.

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

- [vserver.h](#)

## 5.5 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.*

#### 5.5.1 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](#)

## 5.6 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.*

#### 5.6.1 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](#)

## 5.7 vc\_ip\_mask\_pair Struct Reference

### Data Fields

- `uint32_t` [ip](#)
- `uint32_t` [mask](#)

### 5.7.1 Detailed Description

Definition at line 233 of file vserver.h.

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

- [vserver.h](#)

## 5.8 vc\_net\_caps Struct Reference

### Data Fields

- uint\_least64\_t [ncaps](#)
- uint\_least64\_t [cmask](#)

### 5.8.1 Detailed Description

Definition at line 426 of file vserver.h.

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

- [vserver.h](#)

## 5.9 vc\_net\_flags Struct Reference

### Data Fields

- uint\_least64\_t [flagword](#)
- uint\_least64\_t [mask](#)

### 5.9.1 Detailed Description

Definition at line 417 of file vserver.h.

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

- [vserver.h](#)

## 5.10 vc\_net\_nx Struct Reference

### Data Fields

- [vc\\_net\\_nx\\_type](#) type
- size\_t [count](#)
- uint32\_t [ip](#) [4]
- uint32\_t [mask](#) [4]

### 5.10.1 Detailed Description

Definition at line 404 of file `vserver.h`.

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

- [vserver.h](#)

## 5.11 `vc_nx_info` Struct Reference

### Data Fields

- [nid\\_t nid](#)

### 5.11.1 Detailed Description

Definition at line 393 of file `vserver.h`.

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

- [vserver.h](#)

## 5.12 `vc_rlimit` Struct Reference

The limits of a resources.

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

### Data Fields

- [vc\\_limit\\_t min](#)  
*the guaranted minimum of a resources*
- [vc\\_limit\\_t soft](#)  
*the softlimit of a resource*
- [vc\\_limit\\_t hard](#)  
*the absolute hardlimit of a resource*

### 5.12.1 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](#)

## 5.13 `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*

### 5.13.1 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](#)

## 5.14 `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`

### 5.14.1 Detailed Description

Definition at line 675 of file `vserver.h`.

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

- [vserver.h](#)

## 5.15 vc\_vx\_info Struct Reference

### Data Fields

- [xid\\_t](#) xid
- [pid\\_t](#) initpid

### 5.15.1 Detailed Description

Definition at line 470 of file vserver.h.

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

- [vserver.h](#)

## 6 util-vserver (libvserver) File Documentation

### 6.1 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 ssize_t utilvserver_value2text_uint64` (char const \*str, size\_t len, struct [Mapping\\_uint64](#) const \*map, size\_t map\_len) NONNULL((1
- `ssize_t ssize_t ssize_t utilvserver_text2value_uint32` (uint\_least32\_t \*val, struct [Mapping\\_uint32](#) const \*map, size\_t map\_len) NONNULL((1
- `ssize_t ssize_t ssize_t ssize_t utilvserver_text2value_uint64` (uint\_least64\_t \*val, struct [Mapping\\_uint64](#) const \*map, size\_t map\_len) NONNULL((1

### 6.1.1 Detailed Description

Declarations which are used by util-vserver internally.

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

## 6.2 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 function 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)

### 6.2.1 Detailed Description

The public interface of the the libvserver library.

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

### 6.2.2 Define Documentation

#### 6.2.2.1 #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](#).

#### 6.2.2.2 #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](#).

#### 6.2.2.3 #define VC\_SAMECTX ((xid\_t)(-2))

the value which means the current ctx

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

### 6.2.3 Typedef Documentation

#### 6.2.3.1 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.

#### 6.2.3.2 an\_unsigned\_integer\_type [xid\\_t](#)

The identifier of a context.

Definition at line 225 of file vserver.h.

### 6.2.4 Function Documentation

#### 6.2.4.1 int [vc\\_add\\_dlimit](#) (char const \* *filename*, [xid\\_t](#) *xid*, uint\_least32\_t *flags*)

Add a disk limit to a file system.

#### 6.2.4.2 int [vc\\_createSkeleton](#) (char const \* *id*, [vcCfgStyle](#) *style*, int *flags*)

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

#### 6.2.4.3 int [vc\\_get\\_dlimit](#) (char const \* *filename*, [xid\\_t](#) *xid*, uint\_least32\_t *flags*, struct [vc\\_ctx\\_dlimit](#) \* *limits*)

Get a disk limit.

#### 6.2.4.4 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.

#### 6.2.4.5 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.

#### 6.2.4.6 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.



**6.2.4.7** `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.

**6.2.4.8** `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.

**6.2.4.9** `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.

**6.2.4.10** `bool vc_is_dynamic_xid (xid_t xid)`

Returns true iff *xid* is a dynamic xid

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

Remove a disk limit from a file system.

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

Set a disk limit.

**6.2.4.13** `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, 7  
    vc\_lobcap2text, 7  
    vc\_parseLimit, 8  
    vc\_text2bcap, 8  
Helper functions, 6  
  
internal.h, 15  
  
Mapping\_uint32, 9  
Mapping\_uint64, 9  
  
Syscall wrappers, 2  
syscalls  
    vc\_ctx\_create, 3  
    vc\_ctx\_migrate, 3  
    vc\_get\_iattr, 3  
    vc\_get\_rlimit, 4  
    vc\_get\_task\_xid, 4  
    vc\_get\_version, 4  
    vc\_getfilecontext, 5  
    vc\_new\_s\_context, 5  
    vc\_set\_ipv4root, 5  
    vc\_set\_rlimit, 5  
    vc\_syscall, 6  
  
vc\_add\_dlimit  
    vserver.h, 23  
vc\_createSkeleton  
    vserver.h, 23  
vc\_ctx\_caps, 9  
vc\_ctx\_create  
    syscalls, 3  
vc\_ctx\_dlimit, 10  
vc\_ctx\_flags, 10  
vc\_ctx\_migrate  
    syscalls, 3  
VC\_DYNAMIC\_XID  
    vserver.h, 22  
vc\_err\_listparser, 11  
vc\_get\_dlimit  
    vserver.h, 23  
vc\_get\_iattr  
    syscalls, 3  
vc\_get\_rlimit  
    syscalls, 4  
vc\_get\_task\_xid  
    syscalls, 4  
vc\_get\_version  
    syscalls, 4  
vc\_getfilecontext  
    syscalls, 5  
vc\_getVserverAppDir  
    vserver.h, 23  
vc\_getVserverByCtx  
    vserver.h, 23  
vc\_getVserverCfgDir  
    vserver.h, 23  
vc\_getVserverCtx  
    vserver.h, 23  
vc\_getVserverName  
    vserver.h, 24  
vc\_getVserverVdir  
    vserver.h, 24  
vc\_ip\_mask\_pair, 11  
vc\_is\_dynamic\_xid  
    vserver.h, 24  
vc\_limit\_t  
    vserver.h, 22  
vc\_list2bcap  
    helper, 7  
vc\_lobcap2text  
    helper, 7  
vc\_net\_caps, 12  
vc\_net\_flags, 12  
vc\_net\_nx, 12  
vc\_new\_s\_context  
    syscalls, 5  
VC\_NOCTX  
    vserver.h, 22  
vc\_nx\_info, 12  
vc\_parseLimit  
    helper, 8  
vc\_rem\_dlimit  
    vserver.h, 24  
vc\_rlimit, 13  
vc\_rlimit\_mask, 13  
VC\_SAMECTX  
    vserver.h, 22  
vc\_set\_dlimit  
    vserver.h, 24  
vc\_set\_ipv4root  
    syscalls, 5  
vc\_set\_rlimit  
    syscalls, 5  
vc\_set\_sched, 14  
vc\_syscall  
    syscalls, 6  
vc\_text2bcap  
    helper, 8  
vc\_vx\_info, 14

vc\_xidopt2xid  
    vserver.h, [24](#)  
vserver.h, [16](#)  
    vc\_add\_dlimit, [23](#)  
    vc\_createSkeleton, [23](#)  
    VC\_DYNAMIC\_XID, [22](#)  
    vc\_get\_dlimit, [23](#)  
    vc\_getVserverAppDir, [23](#)  
    vc\_getVserverByCtx, [23](#)  
    vc\_getVserverCfgDir, [23](#)  
    vc\_getVserverCtx, [23](#)  
    vc\_getVserverName, [24](#)  
    vc\_getVserverVdir, [24](#)  
    vc\_is\_dynamic\_xid, [24](#)  
    vc\_limit\_t, [22](#)  
    VC\_NOCTX, [22](#)  
    vc\_rem\_dlimit, [24](#)  
    VC\_SAMECTX, [22](#)  
    vc\_set\_dlimit, [24](#)  
    vc\_xidopt2xid, [24](#)  
    xid\_t, [23](#)  
  
xid\_t  
    vserver.h, [23](#)