

util-vserver (libvserver)

0.30.216-pre3131

Generated by Doxygen 1.8.10

Sun May 1 2022 22:27:00

## Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>1</b>
2.1	Data Structures . . . . .	1
<b>3</b>	<b>File Index</b>	<b>2</b>
3.1	File List . . . . .	2
<b>4</b>	<b>Module Documentation</b>	<b>3</b>
4.1	Syscall wrappers . . . . .	3
4.1.1	Detailed Description . . . . .	4
4.1.2	Function Documentation . . . . .	4
4.2	Helper functions . . . . .	9
4.2.1	Detailed Description . . . . .	9
4.2.2	Function Documentation . . . . .	9
<b>5</b>	<b>Data Structure Documentation</b>	<b>12</b>
5.1	Mapping_uint32 Struct Reference . . . . .	12
5.1.1	Detailed Description . . . . .	12
5.2	Mapping_uint64 Struct Reference . . . . .	12
5.2.1	Detailed Description . . . . .	13
5.3	vc_ctx_caps Struct Reference . . . . .	13
5.3.1	Detailed Description . . . . .	13
5.4	vc_ctx_dlimit Struct Reference . . . . .	14
5.4.1	Detailed Description . . . . .	14
5.5	vc_ctx_flags Struct Reference . . . . .	14
5.5.1	Detailed Description . . . . .	15
5.6	vc_ctx_stat Struct Reference . . . . .	15
5.6.1	Detailed Description . . . . .	16
5.7	vc_err_listparser Struct Reference . . . . .	16
5.7.1	Detailed Description . . . . .	16
5.8	vc_ip_mask_pair Struct Reference . . . . .	17
5.8.1	Detailed Description . . . . .	17
5.9	vc_net_addr Struct Reference . . . . .	17
5.9.1	Detailed Description . . . . .	18
5.10	vc_net_caps Struct Reference . . . . .	18
5.10.1	Detailed Description . . . . .	19
5.11	vc_net_flags Struct Reference . . . . .	19
5.11.1	Detailed Description . . . . .	19

5.12	<a href="#">vc_nx_info Struct Reference</a>	19
5.12.1	<a href="#">Detailed Description</a>	20
5.13	<a href="#">vc_rlimit Struct Reference</a>	20
5.13.1	<a href="#">Detailed Description</a>	20
5.14	<a href="#">vc_rlimit_mask Struct Reference</a>	21
5.14.1	<a href="#">Detailed Description</a>	21
5.15	<a href="#">vc_rlimit_stat Struct Reference</a>	21
5.15.1	<a href="#">Detailed Description</a>	22
5.16	<a href="#">vc_sched_info Struct Reference</a>	23
5.16.1	<a href="#">Detailed Description</a>	23
5.17	<a href="#">vc_set_sched Struct Reference</a>	24
5.17.1	<a href="#">Detailed Description</a>	24
5.18	<a href="#">vc_umask Struct Reference</a>	24
5.18.1	<a href="#">Detailed Description</a>	25
5.19	<a href="#">vc_virt_stat Struct Reference</a>	25
5.19.1	<a href="#">Detailed Description</a>	26
5.20	<a href="#">vc_vx_info Struct Reference</a>	26
5.20.1	<a href="#">Detailed Description</a>	26
<b>6</b>	<b><a href="#">File Documentation</a></b>	<b>27</b>
6.1	<a href="#">internal.h File Reference</a>	27
6.1.1	<a href="#">Detailed Description</a>	28
6.2	<a href="#">vserver.h File Reference</a>	28
6.2.1	<a href="#">Detailed Description</a>	36
6.2.2	<a href="#">Macro Definition Documentation</a>	36
6.2.3	<a href="#">Typedef Documentation</a>	36
6.2.4	<a href="#">Function Documentation</a>	37
	<b><a href="#">Index</a></b>	<b>39</b>

## 1 Module Index

### 1.1 Modules

Here is a list of all modules:

**Syscall wrappers** **3**

**Helper functions** **9**

## 2 Data Structure Index

## 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Mapping_uint32</a>	12
<a href="#">Mapping_uint64</a>	12
<a href="#">vc_ctx_caps</a> Capabilities of process-contexts	13
<a href="#">vc_ctx_dlimit</a>	14
<a href="#">vc_ctx_flags</a> Flags of process-contexts	14
<a href="#">vc_ctx_stat</a> Statistics about a context	15
<a href="#">vc_err_listparser</a> Information about parsing errors	16
<a href="#">vc_ip_mask_pair</a>	17
<a href="#">vc_net_addr</a>	17
<a href="#">vc_net_caps</a>	18
<a href="#">vc_net_flags</a>	19
<a href="#">vc_nx_info</a>	19
<a href="#">vc_rlimit</a> The limits of a resources	20
<a href="#">vc_rlimit_mask</a> Masks describing the supported limits	21
<a href="#">vc_rlimit_stat</a> Statistics for a resource limit	21
<a href="#">vc_sched_info</a>	23
<a href="#">vc_set_sched</a>	24
<a href="#">vc_umask</a> Namespaces allowed to unshare	24
<a href="#">vc_virt_stat</a> Contains further statistics about a context	25
<a href="#">vc_vx_info</a>	26

## 3 File Index

### 3.1 File List

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

**internal.h**

Declarations which are used by util-vserver internally

27

**vserver.h**

The public interface of the the libvserver library

28

## 4 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.*
- vc\_vci\_t **vc\_get\_vci** ()
 

*Returns the kernel configuration bits.*
- 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, struct **vc\_ctx\_flags** \*flags)
 

*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, uint\_least64\_t flags)
 

*Moves the current process into the specified context.*
- int **vc\_ctx\_stat** (xid\_t xid, struct **vc\_ctx\_stat** \*stat)
 

*Get some statistics about a context.*
- int **vc\_virt\_stat** (xid\_t xid, struct **vc\_virt\_stat** \*stat)
 

*Get more statistics about a context.*
- int **vc\_ctx\_kill** (xid\_t ctx, pid\_t pid, int sig)
 

*Sends a signal to a context/pid*  
*Special values for pid are:*
- xid\_t **vc\_get\_task\_xid** (pid\_t pid)
 

*Returns the context of the given process.*
- int **vc\_wait\_exit** (xid\_t xid)
 

*Waits for the end of a 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\_rlimit\_stat** (xid\_t xid, int resource, struct **vc\_rlimit\_stat** \*stat)
 

*Returns the current stats of resource.*
- int **vc\_reset\_minmax** (xid\_t xid)
 

*Resets the minimum and maximum observed values of all resources.*
- 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_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.

#### 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, struct vc_ctx_flags * flags )`

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

<i>xid</i>	The new context; special values are: <ul style="list-style-type: none"> <li>• <code>VC_DYNAMIC_XID</code> which means to create a dynamic context</li> </ul>
------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

##### Returns

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

##### 4.1.2.2 `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

##### 4.1.2.3 `int vc_ctx_migrate ( xid_t xid, uint_least64_t flags )`

Moves the current process into the specified context.

##### Parameters

<i>xid</i>	The new context
<i>flags</i>	The flags, see <code>VC_VXM_*</code>

##### Returns

0 on success, -1 on errors

##### 4.1.2.4 `int vc_ctx_stat ( xid_t xid, struct vc_ctx_stat * stat )`

Get some statistics about a context.

## Parameters

<i>xid</i>	The context to get stats about
<i>stat</i>	Where to store the result

## Returns

0 on success, -1 on errors.

#### 4.1.2.5 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

<i>filename</i>	The name of the file whose attributes shall be determined.
<i>xid</i>	When non-zero and the VC_IATTR_XID bit is set in <i>mask</i> , the assigned context of <i>filename</i> will be stored there.
<i>flags</i>	When non-zero, a bitmask of current attributes will be stored there. These attributes must be requested explicitly by setting the appropriate bit in <i>mask</i>
<i>mask</i>	Points to a bitmask which tells which attributes shall be determined. On return, it will masquerade the attributes which were determined.

## Precondition

$mask \neq 0 \ \&\& \ !(( *mask \& VC\_IATTR\_XID) \ \&\& \ xid == 0) \ \&\& \ !(( *mask \& \sim VC\_IATTR\_XID) \ \&\& \ flags == 0)$

#### 4.1.2.6 int vc\_get\_rlimit ( xid\_t xid, int resource, struct vc\_rlimit \* lim )

Returns the limits of *resource*.

## Parameters

<i>xid</i>	The id of the context
<i>resource</i>	The resource which will be queried
<i>lim</i>	The result which will be filled with the limits

## Returns

0 on success, and -1 on errors.

#### 4.1.2.7 xid\_t vc\_get\_task\_xid ( pid\_t pid )

Returns the context of the given process.

## Parameters

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

## Returns

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

#### 4.1.2.8 `vc_vci_t vc_get_vci ( )`

Returns the kernel configuration bits.

##### Returns

The kernel configuration bits

#### 4.1.2.9 `int vc_get_version ( )`

Returns the version of the current kernel API.

##### Returns

The versionnumber of the kernel API

#### 4.1.2.10 `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

<i>filename</i>	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

#### 4.1.2.11 `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

<i>ctx</i>	The new context; special values for are <ul style="list-style-type: none"> <li>• <code>VC_SAMECTX</code> which means the current context (just for changing caps and flags)</li> <li>• <code>VC_DYNAMIC_XID</code> which means the next free context; this value can be used by ordinary users also</li> </ul>
<i>remove_cap</i>	The linux capabilities which will be <b>removed</b> .
<i>flags</i>	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

#### 4.1.2.12 `int vc_reset_minmax ( xid_t xid )`

Resets the minimum and maximum observed values of all resources.



## Parameters

<i>xid</i>	The id of the context
------------	-----------------------

## Returns

0 on success, and -1 on errors.

4.1.2.13 `int vc_rlimit_stat ( xid_t xid, int resource, struct vc_rlimit_stat * stat )`

Returns the current stats of *resource*.

## Parameters

<i>xid</i>	The id of the context
<i>resource</i>	The resource which will be queried
<i>stat</i>	The result which will be filled with the stats

## Returns

0 on success, and -1 on errors.

4.1.2.14 `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`

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

Sets the limits of *resource*.

## Parameters

<i>xid</i>	The id of the context
<i>resource</i>	The resource which will be queried
<i>lim</i>	The new limits

## Returns

0 on success, and -1 on errors.

4.1.2.16 `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

<i>cmd</i>	the command to be executed
<i>xid</i>	the xid on which the cmd shall be applied
<i>data</i>	additional arguments; depends on <i>cmd</i>

## Returns

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

4.1.2.17 `int vc_virt_stat ( xid_t xid, struct vc_virt_stat * stat )`

Get more statistics about a context.

**Parameters**

<i>xid</i>	The context to get stats about
<i>stat</i>	Where to store the result

**Returns**

0 on success, -1 on errors.

## 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.*
- `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:*  

```

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"

```

#### 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**

<i>str</i>	The string to be parsed
<i>len</i>	The length of the string, or 0 for automatic detection
<i>err</i>	Pointer to a structure for error-information, or <code>NULL</code> .
<i>cap</i>	Pointer to a <code>vc_ctx_caps</code> structure holding the results; only the <i>bcaps</i> and <i>bmask</i> fields will be changed and already set values will not be honored. When an error occurred, <i>cap</i> 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**

<i>val</i>	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**

$*val_{old} \neq 0 \leftrightarrow *val_{old} > *val_{new}$   
 $*val_{old} == 0 \leftrightarrow result == 0$

**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**

<i>str</i>	The string which shall be parsed
<i>res</i>	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 bcability.

**Parameters**

<i>str</i>	The string to be parsed; both "CAP_XXX" and "XXX" will be accepted
<i>len</i>	The length of the string, or 0 for automatic detection

**Returns**

0 on error; a bitmask on success

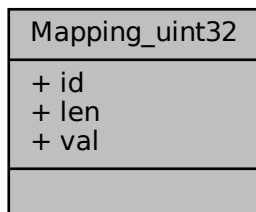
**Precondition**

*str* != 0

## 5 Data Structure Documentation

### 5.1 Mapping\_uint32 Struct Reference

Collaboration diagram for Mapping\_uint32:



#### Data Fields

- char const \*const **id**
- size\_t **len**
- uint\_least32\_t **val**

#### 5.1.1 Detailed Description

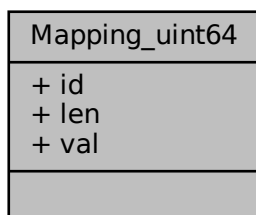
Definition at line 80 of file internal.h.

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

- [internal.h](#)

### 5.2 Mapping\_uint64 Struct Reference

Collaboration diagram for Mapping\_uint64:



## Data Fields

- char const \*const **id**
- size\_t **len**
- uint\_least64\_t **val**

## 5.2.1 Detailed Description

Definition at line 86 of file internal.h.

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

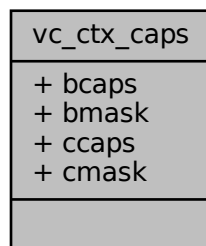
- [internal.h](#)

## 5.3 vc\_ctx\_caps Struct Reference

Capabilities of process-contexts.

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

Collaboration diagram for vc\_ctx\_caps:



## 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.

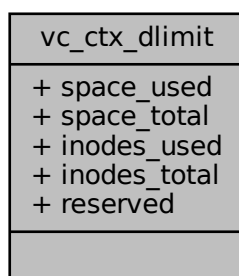
Definition at line 518 of file vserver.h.

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

- [vserver.h](#)

## 5.4 vc\_ctx\_dlimit Struct Reference

Collaboration diagram for vc\_ctx\_dlimit:



### 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 795 of file `vserver.h`.

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

- [vserver.h](#)

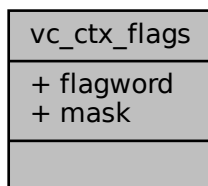
## 5.5 vc\_ctx\_flags Struct Reference

Flags of process-contexts.

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



Collaboration diagram for vc\_ctx\_flags:



#### 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 440 of file `vserver.h`.

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

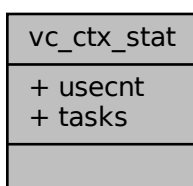
- [vserver.h](#)

## 5.6 vc\_ctx\_stat Struct Reference

Statistics about a context.

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

Collaboration diagram for `vc_ctx_stat`:



## Data Fields

- `uint_least32_t` [usecnt](#)  
*number of uses*
- `uint_least32_t` [tasks](#)  
*number of tasks*

### 5.6.1 Detailed Description

Statistics about a context.

Definition at line 471 of file `vserver.h`.

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

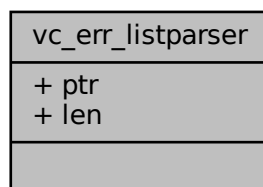
- [vserver.h](#)

## 5.7 `vc_err_listparser` Struct Reference

Information about parsing errors.

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

Collaboration diagram for `vc_err_listparser`:



## Data Fields

- `char const *` [ptr](#)  
*Pointer to the first character of an erroneous string.*
- `size_t` [len](#)  
*Length of the erroneous string.*

### 5.7.1 Detailed Description

Information about parsing errors.

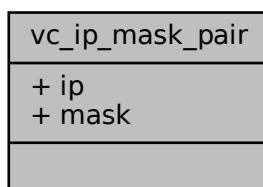
Definition at line 879 of file `vserver.h`.

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

- [vserver.h](#)

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

Collaboration diagram for vc\_ip\_mask\_pair:



### Data Fields

- `uint32_t ip`
- `uint32_t mask`

### 5.8.1 Detailed Description

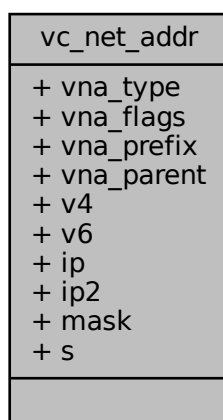
Definition at line 418 of file `vserver.h`.

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

- [vserver.h](#)

## 5.9 vc\_net\_addr Struct Reference

Collaboration diagram for vc\_net\_addr:



## Data Fields

- uint16\_t **vna\_type**
- uint16\_t **vna\_flags**
- uint16\_t **vna\_prefix**
- uint16\_t **vna\_parent**
- struct {
  - union {
    - struct in\_addr **v4**
    - struct in6\_addr **v6**
  - ip**
  - union {
    - struct in\_addr **v4**
    - struct in6\_addr **v6**
  - ip2**
  - union {
    - struct in\_addr **v4**
    - struct in6\_addr **v6**
  - mask**
- s**

## 5.9.1 Detailed Description

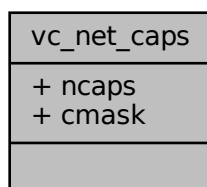
Definition at line 668 of file vserver.h.

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

- [vserver.h](#)

## 5.10 vc\_net\_caps Struct Reference

Collaboration diagram for vc\_net\_caps:



## Data Fields

- uint\_least64\_t **ncaps**
- uint\_least64\_t **cmask**

#### 5.10.1 Detailed Description

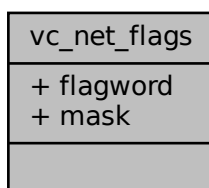
Definition at line 709 of file vserver.h.

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

- [vserver.h](#)

### 5.11 vc\_net\_flags Struct Reference

Collaboration diagram for vc\_net\_flags:



#### Data Fields

- `uint_least64_t` **flagword**
- `uint_least64_t` **mask**

#### 5.11.1 Detailed Description

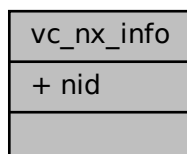
Definition at line 695 of file vserver.h.

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

- [vserver.h](#)

### 5.12 vc\_nx\_info Struct Reference

Collaboration diagram for vc\_nx\_info:



## Data Fields

- `nid_t nid`

### 5.12.1 Detailed Description

Definition at line 661 of file `vserver.h`.

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

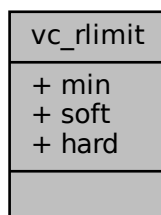
- [vserver.h](#)

## 5.13 `vc_rlimit` Struct Reference

The limits of a resources.

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

Collaboration diagram for `vc_rlimit`:



## 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*

### 5.13.1 Detailed Description

The limits of a resources.

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

Definition at line 584 of file `vserver.h`.

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

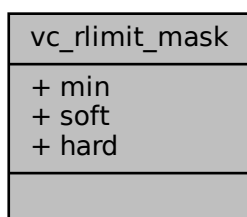
- [vserver.h](#)

## 5.14 `vc_rlimit_mask` Struct Reference

Masks describing the supported limits.

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

Collaboration diagram for `vc_rlimit_mask`:



### 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*

#### 5.14.1 Detailed Description

Masks describing the supported limits.

Definition at line 571 of file `vserver.h`.

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

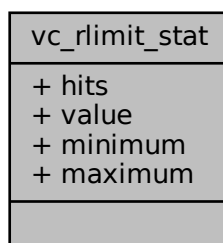
- `vserver.h`

## 5.15 `vc_rlimit_stat` Struct Reference

Statistics for a resource limit.

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

Collaboration diagram for `vc_rlimit_stat`:



#### Data Fields

- `uint_least32_t` [hits](#)

*number of hits on the limit*

- `vc_limit_t` [value](#)

*current value*

- `vc_limit_t` [minimum](#)

*minimum value observed*

- `vc_limit_t` [maximum](#)

*maximum value observed*

#### 5.15.1 Detailed Description

Statistics for a resource limit.

Definition at line 612 of file `vserver.h`.

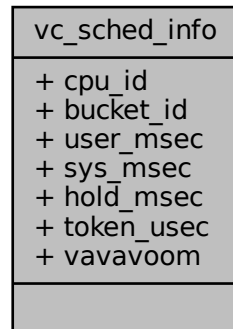
The documentation for this struct was generated from the following file:

- [vserver.h](#)



## 5.16 vc\_sched\_info Struct Reference

Collaboration diagram for vc\_sched\_info:



### Data Fields

- `int_least32_t` **cpu\_id**
- `int_least32_t` **bucket\_id**
- `uint_least64_t` **user\_msec**
- `uint_least64_t` **sys\_msec**
- `uint_least64_t` **hold\_msec**
- `uint_least32_t` **token\_usec**
- `int_least32_t` **vavavoom**

### 5.16.1 Detailed Description

Definition at line 847 of file `vserver.h`.

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

- [vserver.h](#)

## 5.17 vc\_set\_sched Struct Reference

Collaboration diagram for vc\_set\_sched:



### Data Fields

- uint\_least32\_t **set\_mask**
- int\_least32\_t **fill\_rate**
- int\_least32\_t **interval**
- int\_least32\_t **fill\_rate2**
- int\_least32\_t **interval2**
- int\_least32\_t **tokens**
- int\_least32\_t **tokens\_min**
- int\_least32\_t **tokens\_max**
- int\_least32\_t **priority\_bias**
- int\_least32\_t **cpu\_id**
- int\_least32\_t **bucket\_id**

### 5.17.1 Detailed Description

Definition at line 830 of file vserver.h.

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

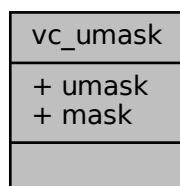
- [vserver.h](#)

## 5.18 vc\_umask Struct Reference

Namespaces allowed to unshare.

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

Collaboration diagram for vc\_umask:



#### Data Fields

- uint\_least64\_t **umask**
- uint\_least64\_t **mask**

##### 5.18.1 Detailed Description

Namespaces allowed to unshare.

Definition at line 868 of file vserver.h.

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

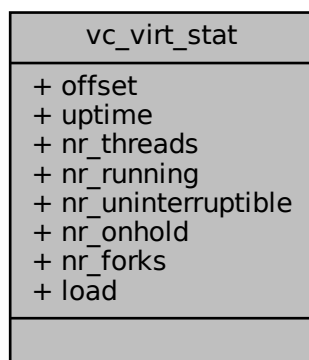
- [vserver.h](#)

## 5.19 vc\_virt\_stat Struct Reference

Contains further statistics about a context.

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

Collaboration diagram for vc\_virt\_stat:



## Data Fields

- `uint_least64_t offset`
- `uint_least64_t uptime`
- `uint_least32_t nr_threads`
- `uint_least32_t nr_running`
- `uint_least32_t nr_uninterruptible`
- `uint_least32_t nr_onhold`
- `uint_least32_t nr_forks`
- `uint_least32_t load` [3]

### 5.19.1 Detailed Description

Contains further statistics about a context.

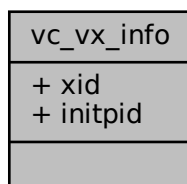
Definition at line 486 of file `vserver.h`.

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

- [vserver.h](#)

## 5.20 vc\_vx\_info Struct Reference

Collaboration diagram for `vc_vx_info`:



## Data Fields

- `xid_t xid`
- `pid_t initpid`

### 5.20.1 Detailed Description

Definition at line 536 of file `vserver.h`.

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

- [vserver.h](#)

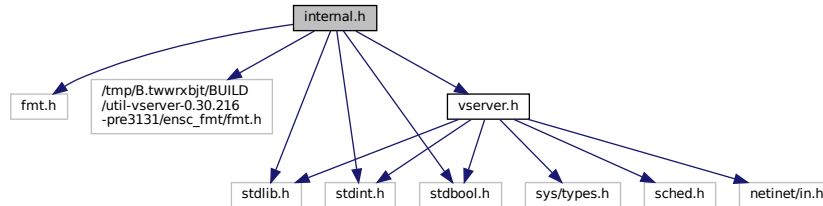
## 6 File Documentation

### 6.1 internal.h File Reference

Declarations which are used by util-vserver internally.

```
#include "fmt.h"
#include "vserver.h"
#include <stdbool.h>
```

Include dependency graph for internal.h:



#### Data Structures

- struct [Mapping\\_uint32](#)
- struct [Mapping\\_uint64](#)

#### Macros

- `#define _symbol_version(real, name, version)`
- `#define _default_symbol_version(real, name, version) extern __typeof (real) name __attribute__ ((alias (#name)))`
- `#define symbol_version(real, name, version) _symbol_version(real, name, version)`
- `#define default_symbol_version(real, name, version) _default_symbol_version(real, name, version)`

#### Functions

- `char * vc_getVserverByCtx_Internal (xid_t ctx, vcCfgStyle *style, char const *revdir, bool validate_result)`
- `int utilvserver_checkCompatVersion ()`
- `uint_least32_t utilvserver_checkCompatConfig ()`
- `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`

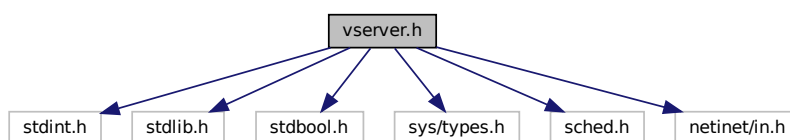
### 6.1.1 Detailed Description

Declarations which are used by util-vserver internally.

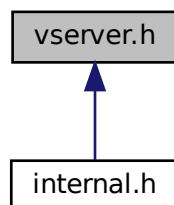
## 6.2 vserver.h File Reference

The public interface of the the libvserver library.

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sched.h>
#include <netinet/in.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\\_ctx\\_flags](#)  
*Flags of process-contexts.*
- struct [vc\\_ctx\\_stat](#)  
*Statistics about a context.*
- struct [vc\\_virt\\_stat](#)  
*Contains further statistics about a context.*
- struct [vc\\_ctx\\_caps](#)  
*Capabilities of process-contexts.*
- struct [vc\\_vx\\_info](#)

- struct [vc\\_rlimit\\_mask](#)  
*Masks describing the supported limits.*
- struct [vc\\_rlimit](#)  
*The limits of a resources.*
- struct [vc\\_rlimit\\_stat](#)  
*Statistics for a resource limit.*
- struct [vc\\_nx\\_info](#)
- struct [vc\\_net\\_addr](#)
- struct [vc\\_net\\_flags](#)
- struct [vc\\_net\\_caps](#)
- struct [vc\\_ctx\\_dlimit](#)
- struct [vc\\_set\\_sched](#)
- struct [vc\\_sched\\_info](#)
- struct [vc\\_umask](#)  
*Namespaces allowed to unshare.*
- struct [vc\\_err\\_listparser](#)  
*Information about parsing errors.*

## Macros

- #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\_CAP\_SETFCAP 31
- #define VC\_CAP\_MAC\_OVERRIDE 32
- #define VC\_CAP\_MAC\_ADMIN 33
- #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\_WRITE 0x00000008u
- #define VC\_IATTR\_FLAGS 0x0000000fu
- #define VC\_IATTR\_BARRIER 0x00010000u
- #define VC\_IATTR\_IUNLINK 0x00020000u
- #define VC\_IATTR\_IMMUTABLE 0x00040000u
- #define VC\_IATTR\_COW 0x00080000u
- #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\_VIRT\_TIME 0x00100000ull
- #define VC\_VXF\_HIDE\_MOUNT 0x01000000ull
- #define VC\_VXF\_HIDE\_NETIF 0x02000000ull
- #define VC\_VXF\_HIDE\_VINFO 0x04000000ull
- #define VC\_VXF\_STATE\_SETUP (1ULL<<32)
- #define VC\_VXF\_STATE\_INIT (1ULL<<33)
- #define VC\_VXF\_STATE\_ADMIN (1ULL<<34)
- #define VC\_VXF\_SC\_HELPER (1ULL<<36)



- `#define VC_VXF_REBOOT_KILL (1ULL<<37)`
- `#define VC_VXF_PERSISTENT (1ULL<<38)`
- `#define VC_VXF_FORK_RSS (1ULL<<48)`
- `#define VC_VXF_PROLIFIC (1ULL<<49)`
- `#define VC_VXF_IGNEG_NICE (1ULL<<52)`
- `#define VC_VXF_IGNEG_IONICE (1ULL<<53)`
- `#define VC_VXC_SET_UTSNAME 0x00000001ull`
- `#define VC_VXC_SET_RLIMIT 0x00000002ull`
- `#define VC_VXC_FS_SECURITY 0x00000004ull`
- `#define VC_VXC_FS_TRUSTED 0x00000008ull`
- `#define VC_VXC_TIOCSTI 0x00000010ull`
- `#define VC_VXC_RAW_ICMP 0x00000100ull`
- `#define VC_VXC_SYSLOG 0x00001000ull`
- `#define VC_VXC_OOM_ADJUST 0x00002000ull`
- `#define VC_VXC_AUDIT_CONTROL 0x00004000ull`
- `#define VC_VXC_SECURE_MOUNT 0x00010000ull`
- `#define VC_VXC_SECURE_REMOUNT 0x00020000ull`
- `#define VC_VXC_BINARY_MOUNT 0x00040000ull`
- `#define VC_VXC_DEV_MOUNT 0x00080000ull`
- `#define VC_VXC_QUOTA_CTL 0x00100000ull`
- `#define VC_VXC_ADMIN_MAPPER 0x00200000ull`
- `#define VC_VXC_ADMIN_CLOOP 0x00400000ull`
- `#define VC_VXC_KTHREAD 0x01000000ull`
- `#define VC_VXC_NAMESPACE 0x02000000ull`
- `#define VC_VXSM_FILL_RATE 0x0001`
- `#define VC_VXSM_INTERVAL 0x0002`
- `#define VC_VXSM_FILL_RATE2 0x0004`
- `#define VC_VXSM_INTERVAL2 0x0008`
- `#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_VXSM_CPU_ID 0x1000`
- `#define VC_VXSM_BUCKET_ID 0x2000`
- `#define VC_VXSM_IDLE_TIME 0x0200`
- `#define VC_VXSM_FORCE 0x0400`
- `#define VC_VXSM_MSEC 0x4000`
- `#define VC_VXSM_V3_MASK 0x0173`
- `#define VC_NXF_INFO_LOCK 0x00000001ull`
- `#define VC_NXF_INFO_PRIVATE 0x00000008ull`
- `#define VC_NXF_SINGLE_IP 0x00000100ull`
- `#define VC_NXF_LBACK_REMAP 0x00000200ull`
- `#define VC_NXF_LBACK_ALLOW 0x00000400ull`
- `#define VC_NXF_HIDE_NETIF 0x02000000ull`
- `#define VC_NXF_HIDE_LBACK 0x04000000ull`
- `#define VC_NXF_STATE_SETUP (1ULL<<32)`
- `#define VC_NXF_STATE_ADMIN (1ULL<<34)`
- `#define VC_NXF_SC_HELPER (1ULL<<36)`
- `#define VC_NXF_PERSISTENT (1ULL<<38)`
- `#define VC_NXC_TUN_CREATE 0x00000001ull`
- `#define VC_NXC_RAW_ICMP 0x00000100ull`
- `#define VC_VLIMIT_NSOCK 16`
- `#define VC_VLIMIT_OPENFD 17`
- `#define VC_VLIMIT_ANON 18`
- `#define VC_VLIMIT_SHMEM 19`

- **#define VC\_VLIMIT\_SEMARY** 20
- **#define VC\_VLIMIT\_NSEMS** 21
- **#define VC\_VLIMIT\_DENTRY** 22
- **#define VC\_VLIMIT\_MAPPED** 23
- **#define VC\_VCI\_NO\_DYNAMIC** (1 << 0)
- **#define VC\_VCI\_PROC\_SECURE** (1 << 4)
- **#define VC\_VCI\_HARDCPU** (1 << 5)
- **#define VC\_VCI\_IDLELIMIT** (1 << 6)
- **#define VC\_VCI\_IDLETIME** (1 << 7)
- **#define VC\_VCI\_COWBL** (1 << 8)
- **#define VC\_VCI\_FULLCOWBL** (1 << 9)
- **#define VC\_VCI\_SPACES** (1 << 10)
- **#define VC\_VCI\_NETV2** (1 << 11)
- **#define VC\_VCI\_MEMCG** (1 << 12)
- **#define VC\_VCI\_DEBUG** (1 << 16)
- **#define VC\_VCI\_HISTORY** (1 << 20)
- **#define VC\_VCI\_TAGGED** (1 << 24)
- **#define VC\_VCI\_PPTAG** (1 << 28)
- **#define VC\_DATTR\_CREATE** 0x00000001
- **#define VC\_DATTR\_OPEN** 0x00000002
- **#define VC\_DATTR\_REMAP** 0x00000010
- **#define VC\_VXM\_SET\_INIT** 0x00000001
- **#define VC\_VXM\_SET\_REAPER** 0x00000002
- **#define VC\_NXA\_TYPE\_IPV4** 0x0001
- **#define VC\_NXA\_TYPE\_IPV6** 0x0002
- **#define VC\_NXA\_TYPE\_NONE** 0x0000
- **#define VC\_NXA\_TYPE\_ANY** 0x00FF
- **#define VC\_NXA\_TYPE\_ADDR** 0x0010
- **#define VC\_NXA\_TYPE\_MASK** 0x0020
- **#define VC\_NXA\_TYPE\_RANGE** 0x0040
- **#define VC\_NXA\_MOD\_BCAST** 0x0100
- **#define VC\_NXA\_MOD\_LBACK** 0x0200
- **#define CLONE\_NEWNS** 0x00020000
- **#define CLONE\_NEWUTS** 0x04000000
- **#define CLONE\_NEWIPC** 0x08000000
- **#define CLONE\_NEWUSER** 0x10000000
- **#define CLONE\_NEWPID** 0x20000000
- **#define CLONE\_NEWNET** 0x40000000
- **#define VC\_BAD\_PERSONALITY** ((uint\_least32\_t)(-1))
- **#define vna\_v4\_ip** s.ip.v4
- **#define vna\_v4\_ip2** s.ip2.v4
- **#define vna\_v4\_mask** s.mask.v4
- **#define vna\_v6\_ip** s.ip.v6
- **#define vna\_v6\_ip2** s.ip2.v6
- **#define vna\_v6\_mask** s.mask.v6
- **#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 an\_unsigned\_integer\_type [tag\\_t](#)
- typedef uint64\_t [vc\\_vci\\_t](#)
- typedef uint\_least64\_t [vc\\_limit\\_t](#)

*The type which is used for a single limit value.*

## Enumerations

- 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](#), [vcFEATURE\\_VINFO](#),  
  [vcFEATURE\\_VHI](#), [vcFEATURE\\_VSHELPER0](#), [vcFEATURE\\_VSHELPER](#), [vcFEATURE\\_VWAIT](#),  
  [vcFEATURE\\_VNET](#), [vcFEATURE\\_VSTAT](#), [vcFEATURE\\_PPTAG](#), [vcFEATURE\\_PIDSPACE](#),  
  [vcFEATURE\\_SPACES](#), [vcFEATURE\\_PERSISTENT](#), [vcFEATURE\\_PIVOT\\_ROOT](#), [vcFEATURE\\_MEMCG](#),  
  [vcFEATURE\\_DYNAMIC](#), [vcFEATURE\\_BME](#) }
- 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](#) }
- enum [vcCtxType](#) { [vcCTX\\_XID](#) = 1, [vcCTX\\_NID](#), [vcCTX\\_TAG](#) }

## 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.*
- [vc\\_vci\\_t](#) [vc\\_get\\_vci](#) ()  
  *Returns the kernel configuration bits.*
- int [vc\\_get\\_kernel](#) ()
- [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, struct [vc\\_ctx\\_flags](#) \*flags)  
  *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, uint\_least64\_t flags)

- Moves the current process into the specified context.*

  - int `vc_ctx_stat` (`xid_t` xid, struct `vc_ctx_stat` \*stat)

*Get some statistics about a context.*
- int `vc_virt_stat` (`xid_t` xid, struct `vc_virt_stat` \*stat)

*Get more statistics about a context.*
- int `vc_ctx_kill` (`xid_t` ctx, `pid_t` pid, int sig)

*Sends a signal to a context/pid*  
*Special values for pid are:*
- 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 \*)
  - int `vc_get_vx_info` (`xid_t` xid, struct `vc_vx_info` \*info)
  - `xid_t` `vc_get_task_xid` (`pid_t` pid)

*Returns the context of the given process.*
- int `vc_wait_exit` (`xid_t` xid)

*Waits for the end of a context.*
- int `vc_get_rlimit_mask` (`xid_t` xid, struct `vc_rlimit_mask` \*lim)

*Returns the limits supported by the kernel.*
- 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_rlimit_stat` (`xid_t` xid, int resource, struct `vc_rlimit_stat` \*stat)

*Returns the current stats of resource.*
- int `vc_reset_minmax` (`xid_t` xid)

*Resets the minimum and maximum observed values of all resources.*
- 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.*
- `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_addr` const \*info)
  - int `vc_net_remove` (`nid_t` nid, struct `vc_net_addr` 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_fset_iattr` (int fd, `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.*
- int `vc_fget_iattr` (int fd, `xid_t` \*xid, `uint_least32_t` \*flags, `uint_least32_t` \*mask)
  - `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_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)

- int **vc\_enter\_namespace** (xid\_t xid, uint\_least64\_t mask, uint32\_t index)
- int **vc\_set\_namespace** (xid\_t xid, uint\_least64\_t mask, uint32\_t index)
- int **vc\_cleanup\_namespace** (void)
- uint\_least64\_t **vc\_get\_space\_mask** (void)
- uint\_least64\_t **vc\_get\_space\_default** (void)
- 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)
- tag\_t **vc\_get\_task\_tag** (pid\_t pid)
- int **vc\_tag\_create** (tag\_t tag)
- int **vc\_tag\_migrate** (tag\_t tag)
- int **vc\_set\_sched** (xid\_t xid, struct **vc\_set\_sched** const \*)
- int **vc\_get\_sched** (xid\_t xid, struct **vc\_set\_sched** \*)
- int **vc\_sched\_info** (xid\_t xid, struct **vc\_sched\_info** \*info)
- int **vc\_set\_mapping** (xid\_t xid, const char \*device, const char \*target, uint32\_t flags)
- int **vc\_unset\_mapping** (xid\_t xid, const char \*device, const char \*target, uint32\_t flags)
- int **vc\_get\_badness** (xid\_t xid, int64\_t \*badness)
- int **vc\_set\_badness** (xid\_t xid, int64\_t badness)
- int **vc\_get\_umask** (xid\_t xid, struct **vc\_umask** \*umask)
- int **vc\_set\_umask** (xid\_t xid, struct **vc\_umask** const \*umask)
- 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:*
  - 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"
  - .
- 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** \*)
- char const \* **vc\_loumask2text** (uint\_least64\_t \*)
- int **vc\_list2umask** (char const \*, size\_t len, struct **vc\_err\_listparser** \*err, struct **vc\_umask** \*)
- int **vc\_list2cflag** (char const \*, size\_t len, struct **vc\_err\_listparser** \*err, struct **vc\_ctx\_flags** \*flags)
- uint\_least64\_t **vc\_text2umask** (char const \*str, size\_t len)
- 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)
- bool **vc\_isSupported** (vcFeatureSet) VC\_ATTR\_CONST
- bool **vc\_isSupportedString** (char const \*)
- vcXidType **vc\_getXIDType** (xid\_t xid) VC\_ATTR\_CONST
- bool **vc\_is\_dynamic\_xid** (xid\_t xid)
- xid\_t **vc\_xidopt2xid** (char const \*, bool honor\_static, char const \*\*err\_info)
- nid\_t **vc\_nidopt2nid** (char const \*, bool honor\_static, char const \*\*err\_info)
- tag\_t **vc\_tagopt2tag** (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, vcCtxType type)
- char \* **vc\_getVserverByCtx** (xid\_t ctx, vcCfgStyle \*style, char const \*revdir)
- int **vc\_compareVserverByld** (char const \*lhs, vcCfgStyle lhs\_style, char const \*rhs, vcCfgStyle rhs\_style)
- void **vc\_exitLikeProcess** (int pid, int ret)
- int **vc\_createSkeleton** (char const \*id, vcCfgStyle style, int flags)

### 6.2.1 Detailed Description

The public interface of the the libvserver library.

### 6.2.2 Macro Definition 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 67 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 64 of file vserver.h.

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

the value which means the current ctx

Definition at line 69 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 568 of file vserver.h.

#### 6.2.3.2 an\_unsigned\_integer\_type xid\_t

The identifier of a context.

Definition at line 361 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 tag\_t vc\_get\_task\_tag ( pid\_t pid )

Get the filesystem tag for a process.

#### 6.2.4.5 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.

#### 6.2.4.6 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.

#### 6.2.4.7 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.

#### 6.2.4.8 xid\_t vc\_getVserverCtx ( char const \* id, vcCfgStyle style, bool honor\_static, bool \* is\_running, vcCtxType type )

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.

#### 6.2.4.9 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.

#### 6.2.4.10 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.

#### 6.2.4.11 `bool vc_is_dynamic_xid ( xid_t xid )`

Returns true iff *xid* is a dynamic xid

#### 6.2.4.12 `nid_t vc_nidopt2nid ( char const * , bool honor_static, char const ** err_info )`

Maps a nid given at '`--nid`' options to a `nid_t`

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

Remove a disk limit from a file system.

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

Set a disk limit.

#### 6.2.4.15 `int vc_tag_create ( tag_t tag )`

Create a new filesystem tag space.

#### 6.2.4.16 `int vc_tag_migrate ( tag_t tag )`

Migrate to an existing filesystem tag space.

#### 6.2.4.17 `tag_t vc_tagopt2tag ( char const * , bool honor_static, char const ** err_info )`

Maps a tag given at '`--tag`' options to a `tag_t`

#### 6.2.4.18 `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 functions, 9  
    vc\_list2bcap, 9  
    vc\_lobcap2text, 10  
    vc\_parseLimit, 10  
    vc\_text2bcap, 11

internal.h, 27

Mapping\_uint32, 12  
Mapping\_uint64, 12

Syscall wrappers, 3  
    vc\_ctx\_create, 4  
    vc\_ctx\_kill, 4  
    vc\_ctx\_migrate, 4  
    vc\_ctx\_stat, 4  
    vc\_get\_iattr, 4  
    vc\_get\_rlimit, 5  
    vc\_get\_task\_xid, 5  
    vc\_get\_vci, 5  
    vc\_get\_version, 5  
    vc\_getfilecontext, 6  
    vc\_new\_s\_context, 6  
    vc\_reset\_minmax, 6  
    vc\_rlimit\_stat, 6  
    vc\_set\_ipv4root, 8  
    vc\_set\_rlimit, 8  
    vc\_syscall, 8  
    vc\_virt\_stat, 8

VC\_DYNAMIC\_XID  
    vserver.h, 36

VC\_NOCTX  
    vserver.h, 36

VC\_SAMECTX  
    vserver.h, 36

vc\_add\_dlimit  
    vserver.h, 37

vc\_createSkeleton  
    vserver.h, 37

vc\_ctx\_caps, 13

vc\_ctx\_create  
    Syscall wrappers, 4

vc\_ctx\_dlimit, 14

vc\_ctx\_flags, 14

vc\_ctx\_kill  
    Syscall wrappers, 4

vc\_ctx\_migrate  
    Syscall wrappers, 4

vc\_ctx\_stat, 15  
    Syscall wrappers, 4

vc\_err\_listparser, 16

vc\_get\_dlimit  
    vserver.h, 37

vc\_get\_iattr  
    Syscall wrappers, 4

vc\_get\_rlimit  
    Syscall wrappers, 5

vc\_get\_task\_tag  
    vserver.h, 37

vc\_get\_task\_xid  
    Syscall wrappers, 5

vc\_get\_vci  
    Syscall wrappers, 5

vc\_get\_version  
    Syscall wrappers, 5

vc\_getVserverAppDir  
    vserver.h, 37

vc\_getVserverByCtx  
    vserver.h, 37

vc\_getVserverCfgDir  
    vserver.h, 37

vc\_getVserverCtx  
    vserver.h, 37

vc\_getVserverName  
    vserver.h, 37

vc\_getVserverVdir  
    vserver.h, 37

vc\_getfilecontext  
    Syscall wrappers, 6

vc\_ip\_mask\_pair, 17

vc\_is\_dynamic\_xid  
    vserver.h, 37

vc\_limit\_t  
    vserver.h, 36

vc\_list2bcap  
    Helper functions, 9

vc\_lobcap2text  
    Helper functions, 10

vc\_net\_addr, 17

vc\_net\_caps, 18

vc\_net\_flags, 19

vc\_new\_s\_context  
    Syscall wrappers, 6

vc\_nidopt2nid  
    vserver.h, 38

vc\_nx\_info, 19

vc\_parseLimit  
    Helper functions, 10

vc\_rem\_dlimit  
    vserver.h, 38

vc\_reset\_minmax  
    Syscall wrappers, 6

vc\_rlimit, 20

vc\_rlimit\_mask, 21

vc\_rlimit\_stat, 21  
    Syscall wrappers, 6

vc\_sched\_info, 23

vc\_set\_dlimit  
    vserver.h, 38

vc\_set\_ipv4root

- Syscall wrappers, 8
- vc\_set\_rlimit
  - Syscall wrappers, 8
- vc\_set\_sched, 24
- vc\_syscall
  - Syscall wrappers, 8
- vc\_tag\_create
  - vserver.h, 38
- vc\_tag\_migrate
  - vserver.h, 38
- vc\_tagopt2tag
  - vserver.h, 38
- vc\_text2bcap
  - Helper functions, 11
- vc\_umask, 24
- vc\_virt\_stat, 25
  - Syscall wrappers, 8
- vc\_vx\_info, 26
- vc\_xidopt2xid
  - vserver.h, 38
- vserver.h, 28
  - VC\_DYNAMIC\_XID, 36
  - VC\_NOCTX, 36
  - VC\_SAMECTX, 36
  - vc\_add\_dlimit, 37
  - vc\_createSkeleton, 37
  - vc\_get\_dlimit, 37
  - vc\_get\_task\_tag, 37
  - vc\_getVserverAppDir, 37
  - vc\_getVserverByCtx, 37
  - vc\_getVserverCfgDir, 37
  - vc\_getVserverCtx, 37
  - vc\_getVserverName, 37
  - vc\_getVserverVdir, 37
  - vc\_is\_dynamic\_xid, 37
  - vc\_limit\_t, 36
  - vc\_nidopt2nid, 38
  - vc\_rem\_dlimit, 38
  - vc\_set\_dlimit, 38
  - vc\_tag\_create, 38
  - vc\_tag\_migrate, 38
  - vc\_tagopt2tag, 38
  - vc\_xidopt2xid, 38
  - xid\_t, 37
- xid\_t
  - vserver.h, 37