

util-vserver (libvserver)  
0.30.216-pre2955

Generated by Doxygen 1.7.3

Mon Apr 25 2011 00:23:46

# 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>2</b>
4.1	Syscall wrappers . . . . .	2
4.1.1	Detailed Description . . . . .	4
4.1.2	Function Documentation . . . . .	4
4.2	Helper functions . . . . .	9
4.2.1	Detailed Description . . . . .	10
4.2.2	Function Documentation . . . . .	10
<b>5</b>	<b>Data Structure Documentation</b>	<b>13</b>
5.1	Mapping_uint32 Struct Reference . . . . .	13
5.1.1	Detailed Description . . . . .	13
5.2	Mapping_uint64 Struct Reference . . . . .	13
5.2.1	Detailed Description . . . . .	13
5.3	vc_ctx_caps Struct Reference . . . . .	13
5.3.1	Detailed Description . . . . .	14
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 . . . . .	15
5.7	vc_err_listparser Struct Reference . . . . .	15
5.7.1	Detailed Description . . . . .	16
5.8	vc_ip_mask_pair Struct Reference . . . . .	16
5.8.1	Detailed Description . . . . .	16
5.9	vc_net_addr Struct Reference . . . . .	16
5.9.1	Detailed Description . . . . .	17
5.10	vc_net_caps Struct Reference . . . . .	17
5.10.1	Detailed Description . . . . .	17
5.11	vc_net_flags Struct Reference . . . . .	17
5.11.1	Detailed Description . . . . .	17
5.12	vc_nx_info Struct Reference . . . . .	18
5.12.1	Detailed Description . . . . .	18
5.13	vc_rlimit Struct Reference . . . . .	18
5.13.1	Detailed Description . . . . .	18
5.14	vc_rlimit_mask Struct Reference . . . . .	18
5.14.1	Detailed Description . . . . .	19
5.15	vc_rlimit_stat Struct Reference . . . . .	19
5.15.1	Detailed Description . . . . .	19

## 1 Module Index 1

---

5.16	<a href="#">vc_sched_info Struct Reference</a>	20
5.16.1	<a href="#">Detailed Description</a>	20
5.17	<a href="#">vc_set_sched Struct Reference</a>	20
5.17.1	<a href="#">Detailed Description</a>	20
5.18	<a href="#">vc_virt_stat Struct Reference</a>	21
5.18.1	<a href="#">Detailed Description</a>	21
5.19	<a href="#">vc_vx_info Struct Reference</a>	21
5.19.1	<a href="#">Detailed Description</a>	21
<b>6</b>	<b>File Documentation</b>	<b>21</b>
6.1	<a href="#">internal.h File Reference</a>	21
6.1.1	<a href="#">Detailed Description</a>	23
6.2	<a href="#">vserver.h File Reference</a>	23
6.2.1	<a href="#">Detailed Description</a>	34
6.2.2	<a href="#">Define Documentation</a>	34
6.2.3	<a href="#">Typedef Documentation</a>	34
6.2.4	<a href="#">Function Documentation</a>	35

## 1 Module Index

### 1.1 Modules

Here is a list of all modules:

<a href="#">Syscall wrappers</a>	<a href="#">2</a>
<a href="#">Helper functions</a>	<a href="#">9</a>

## 2 Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Mapping_uint32</a>	<a href="#">13</a>
<a href="#">Mapping_uint64</a>	<a href="#">13</a>
<a href="#">vc_ctx_caps</a> (Capabilities of process-contexts )	<a href="#">13</a>
<a href="#">vc_ctx_dlimit</a>	<a href="#">14</a>
<a href="#">vc_ctx_flags</a> (Flags of process-contexts )	<a href="#">14</a>
<a href="#">vc_ctx_stat</a> (Statistics about a context )	<a href="#">15</a>
<a href="#">vc_err_listparser</a> (Information about parsing errors )	<a href="#">15</a>

<a href="#">vc_ip_mask_pair</a>	16
<a href="#">vc_net_addr</a>	16
<a href="#">vc_net_caps</a>	17
<a href="#">vc_net_flags</a>	17
<a href="#">vc_nx_info</a>	18
<a href="#">vc_rlimit</a> (The limits of a resources )	18
<a href="#">vc_rlimit_mask</a> (Masks describing the supported limits )	18
<a href="#">vc_rlimit_stat</a> (Statistics for a resource limit )	19
<a href="#">vc_sched_info</a>	20
<a href="#">vc_set_sched</a>	20
<a href="#">vc_virt_stat</a> (Contains further statistics about a context )	21
<a href="#">vc_vx_info</a>	21

## 3 File Index

### 3.1 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 )	21
<a href="#">vserver.h</a> (The public interface of the the libvserver library )	23

## 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) VC_ATTR_NONNULL((3))`  
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) VC_ATTR_NONNULL((2))`  
  
Get some statistics about a context.
- `int vc_virt_stat (xid_t xid, struct vc_virt_stat *stat) VC_ATTR_NONNULL((2))`  
  
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:
  - -1 which means every process in ctx except the init-process
  - 0 which means every process in ctx inclusive the init-process.
- `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) VC_ATTR_NONNULL((3))`  
Returns the limits of resource.
- `int vc_set_rlimit (xid_t xid, int resource, struct vc_rlimit const *lim) VC_ATTR_NONNULL((3))`

*Sets the limits of resource.*

- `int vc_rlimit_stat(xid_t xid, int resource, struct vc_rlimit_stat *stat) VC_ATTR_NONNULL((3))`

*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) VC_ATTR_NONNULL((1))`

*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) VC_ATTR_NONNULL((1))`

*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>• VC_DYNAMIC_XID 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_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 VC_VXM_*

##### Returns

0 on success, -1 on errors

#### 4.1.2.3 `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.

#### 4.1.2.4 `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!=0 && !((*mask&VC_IATTR_XID) && xid==0) && !((*mask&~VC_IATTR_XID) && flags==0)
```

**4.1.2.5 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.

**4.1.2.6 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

**4.1.2.7 vc\_vci\_t vc\_get\_vci ( )**

Returns the kernel configuration bits.



**Returns**

The kernel configuration bits

**4.1.2.8 int vc\_get\_version ( )**

Returns the version of the current kernel API.

**Returns**

The versionnumber of the kernel API

**4.1.2.9 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

**4.1.2.10 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>VC_SAMECTX which means the current context (just for changing caps and flags)</li> <li>VC_DYNAMIC_XID 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

**4.1.2.11 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.

**4.1.2.12 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.

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

Sets the ipv4root information.

#### Precondition

$nb < \text{NB\_IPV4ROOT} \ \&\& \ ip s \neq 0$

**4.1.2.14** `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.

**4.1.2.15** `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

**4.1.2.16** `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) VC\_ATTR\_NONNULL((1)  
  
*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) VC\_ATTR\_NONNULL((1))  
*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) VC\_ATTR\_NONNULL((1)  
*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 NULL.
<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**

$str \neq 0$  &&  $cap \neq 0$ ;  $cap \rightarrow bcaps$  and  $cap \rightarrow bmask$  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**

$val \neq 0$

**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

**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

### 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>
```

### 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 516 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 793 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 438 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>
```

**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 469 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>
```

**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 868 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****Data Fields**

- `uint32_t ip`

- uint32\_t **mask**

#### 5.8.1 Detailed Description

Definition at line 416 of file vserver.h.

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

- [vserver.h](#)

### 5.9 vc\_net\_addr Struct Reference

#### 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 666 of file vserver.h.

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

- [vserver.h](#)

### 5.10 vc\_net\_caps Struct Reference

#### Data Fields

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

### 5.10.1 Detailed Description

Definition at line 707 of file vserver.h.

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

- [vserver.h](#)

## 5.11 **vc\_net\_flags Struct Reference**

### Data Fields

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

### 5.11.1 Detailed Description

Definition at line 693 of file vserver.h.

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

- [vserver.h](#)

## 5.12 **vc\_nx\_info Struct Reference**

### Data Fields

- `nid_t` **nid**

### 5.12.1 Detailed Description

Definition at line 659 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>
```

### 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 582 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>
```

#### 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 569 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>
```

### 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 610 of file `vserver.h`.

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

- [vserver.h](#)

## 5.16 `vc_sched_info` Struct Reference

### 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 845 of file `vserver.h`.

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

- [vserver.h](#)

## 5.17 **vc\_set\_sched Struct Reference**

### 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 828 of file `vserver.h`.

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

- [vserver.h](#)

## 5.18 **vc\_virt\_stat Struct Reference**

Contains further statistics about a context.

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

### 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]

### 5.18.1 Detailed Description

Contains further statistics about a context.

Definition at line 484 of file vserver.h.

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

- [vserver.h](#)

## 5.19 vc\_vx\_info Struct Reference

### Data Fields

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

### 5.19.1 Detailed Description

Definition at line 534 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 <stdlib.h>
#include <stdbool.h>
```

Include dependency graph for internal.h:

### Data Structures

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

### Defines

- `#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.

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 <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\\_err\\_listparser](#)  
*Information about parsing errors.*

### 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\_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** 0x000000fu
- #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\_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\_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) VC_ATTR_NONNULL((3))`  
*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) VC_ATTR_NONNULL((2))`  
*Get some statistics about a context.*
- `int vc_virt_stat (xid_t xid, struct vc_virt_stat *stat) VC_ATTR_NONNULL((2))`  
*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:*



- *-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_cflags(xid_t xid, struct vc_ctx_flags *) VC_ATTR_NONNULL((2))`
- `int vc_set_cflags(xid_t xid, struct vc_ctx_flags const *) VC_ATTR_NONNULL((2))`
- `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) VC_ATTR_NONNULL((2))`
- `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) VC_ATTR_NONNULL((2))`  
  
*Returns the limits supported by the kernel.*
- `int vc_get_rlimit(xid_t xid, int resource, struct vc_rlimit *lim) VC_ATTR_NONNULL((3))`  
*Returns the limits of resource.*
- `int vc_set_rlimit(xid_t xid, int resource, struct vc_rlimit const *lim) VC_ATTR_NONNULL((3))`  
*Sets the limits of resource.*
- `int vc_rlimit_stat(xid_t xid, int resource, struct vc_rlimit_stat *stat) VC_ATTR_NONNULL((3))`  
*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) VC_ATTR_NONNULL((1))`  
  
*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.*
- `nid_t vc_get_task_nid(pid_t pid)`

- `int vc_get_nx_info` (`nid_t` nid, `struct vc_nx_info *`) VC\_ATTR\_NONNULL((2))
- `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) VC\_ATTR\_NONNULL((1))
- `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) VC\_ATTR\_NONNULL((1))
 

*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) VC\_ATTR\_NONNULL((4))
- `xid_t vc_getfilecontext` (`char` const \*filename) VC\_ATTR\_NONNULL((1))
 

*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) VC\_ATTR\_NONNULL((3))
- `int vc_get_vhi_name` (`xid_t` xid, `vc_uts_type` type, `char` \*val, `size_t` len) VC\_ATTR\_NONNULL((3))
- `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) VC\_ATTR\_NONNULL((1))
- `int vc_rem_dlimit` (`char` const \*filename, `xid_t` xid, `uint_least32_t` flags) VC\_ATTR\_NONNULL((1))
- `int vc_set_dlimit` (`char` const \*filename, `xid_t` xid, `uint_least32_t` flags, `struct vc_ctx_dlimit` const \*limits) VC\_ATTR\_NONNULL((1))
- `int vc_get_dlimit` (`char` const \*filename, `xid_t` xid, `uint_least32_t` flags, `struct vc_ctx_dlimit` \*limits) VC\_ATTR\_NONNULL((1))
- `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 *) VC_ATTR_NONNULL((2))`
- `int vc_get_sched(xid_t xid, struct vc_set_sched *) VC_ATTR_NONNULL((2))`
- `int vc_sched_info(xid_t xid, struct vc_sched_info *info) VC_ATTR_NONNULL((2))`
- `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)`
- `uint_least64_t vc_text2bcap(char const *str, size_t len)`  
*Converts a single string into bcapability.*
- `char const * vc_jobcap2text(uint_least64_t *val) VC_ATTR_NONNULL((1))`  
*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) VC_ATTR_NONNULL((1))`  
*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"

```
- `int 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) VC\_ATTR\_NONNULL((1))
- char const \* **vc\_lopersonality2text** (uint\_least32\_t \*) VC\_ATTR\_NONNULL((1))
- int **vc\_list2personalityflag** (char const \*, size\_t len, uint\_least32\_t \*personality, struct [vc\\_err\\_listparser](#) \*err) VC\_ATTR\_NONNULL((1))
- int uint\_least32\_t **vc\_str2personalitytype** (char const \*, size\_t len) VC\_ATTR\_NONNULL((1))
- 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\_compareVserverById** (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.

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 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 566 of file vserver.h.

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

The identifier of a context.

Definition at line 359 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 pkgmanagment and filesystem (when requested).

**6.2.4.3** `int 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  
    vc\_list2bcap, 10  
    vc\_lobcap2text, 11  
    vc\_parseLimit, 11  
    vc\_text2bcap, 12  
Helper functions, 9  
  
internal.h, 21  
  
Mapping\_uint32, 13  
Mapping\_uint64, 13  
  
Syscall wrappers, 2  
syscalls  
    vc\_ctx\_create, 4  
    vc\_ctx\_migrate, 4  
    vc\_ctx\_stat, 4  
    vc\_get\_iattr, 5  
    vc\_get\_rlimit, 5  
    vc\_get\_task\_xid, 6  
    vc\_get\_vci, 6  
    vc\_get\_version, 6  
    vc\_getfilecontext, 6  
    vc\_new\_s\_context, 7  
    vc\_reset\_minmax, 7  
    vc\_rlimit\_stat, 8  
    vc\_set\_ipv4root, 8  
    vc\_set\_rlimit, 8  
    vc\_syscall, 8  
    vc\_virt\_stat, 9  
  
vc\_add\_dlimit  
    vserver.h, 35  
vc\_createSkeleton  
    vserver.h, 35  
vc\_ctx\_caps, 13  
vc\_ctx\_create  
    syscalls, 4  
vc\_ctx\_dlimit, 14  
vc\_ctx\_flags, 14  
vc\_ctx\_migrate  
    syscalls, 4  
vc\_ctx\_stat, 15  
    syscalls, 4  
VC\_DYNAMIC\_XID  
    vserver.h, 34  
vc\_err\_listparser, 15  
vc\_get\_dlimit  
    vserver.h, 35  
vc\_get\_iattr  
    syscalls, 5  
vc\_get\_rlimit  
    syscalls, 5  
vc\_get\_task\_tag  
    vserver.h, 35  
vc\_get\_task\_xid  
    syscalls, 6  
vc\_get\_vci  
    syscalls, 6  
vc\_get\_version  
    syscalls, 6  
vc\_getfilecontext  
    syscalls, 6  
vc\_getVserverAppDir  
    vserver.h, 35  
vc\_getVserverByCtx  
    vserver.h, 35  
vc\_getVserverCfgDir  
    vserver.h, 35  
vc\_getVserverCtx  
    vserver.h, 35  
vc\_getVserverName  
    vserver.h, 36  
vc\_getVserverVdir  
    vserver.h, 36  
vc\_ip\_mask\_pair, 16  
vc\_is\_dynamic\_xid  
    vserver.h, 36  
vc\_limit\_t  
    vserver.h, 34  
vc\_list2bcap  
    helper, 10  
vc\_lobcap2text  
    helper, 11  
vc\_net\_addr, 16  
vc\_net\_caps, 17  
vc\_net\_flags, 17  
vc\_new\_s\_context  
    syscalls, 7  
vc\_nidopt2nid  
    vserver.h, 36  
VC\_NOCTX  
    vserver.h, 34  
vc\_nx\_info, 18  
vc\_parseLimit



---

- helper, 11
- vc\_rem\_dlimit
  - vserver.h, 36
- vc\_reset\_minmax
  - syscalls, 7
- vc\_rlimit, 18
- vc\_rlimit\_mask, 18
- vc\_rlimit\_stat, 19
  - syscalls, 8
- VC\_SAMECTX
  - vserver.h, 34
- vc\_sched\_info, 20
- vc\_set\_dlimit
  - vserver.h, 36
- vc\_set\_ipv4root
  - syscalls, 8
- vc\_set\_rlimit
  - syscalls, 8
- vc\_set\_sched, 20
- vc\_syscall
  - syscalls, 8
- vc\_tag\_create
  - vserver.h, 36
- vc\_tag\_migrate
  - vserver.h, 36
- vc\_tagopt2tag
  - vserver.h, 36
- vc\_text2bcap
  - helper, 12
- vc\_virt\_stat, 21
  - syscalls, 9
- vc\_vx\_info, 21
- vc\_xidopt2xid
  - vserver.h, 37
- vserver.h, 23
  - vc\_add\_dlimit, 35
  - vc\_createSkeleton, 35
  - VC\_DYNAMIC\_XID, 34
  - vc\_get\_dlimit, 35
  - vc\_get\_task\_tag, 35
  - vc\_getVserverAppDir, 35
  - vc\_getVserverByCtx, 35
  - vc\_getVserverCfgDir, 35
  - vc\_getVserverCtx, 35
  - vc\_getVserverName, 36
  - vc\_getVserverVdir, 36
  - vc\_is\_dynamic\_xid, 36
  - vc\_limit\_t, 34
  - vc\_nidopt2nid, 36
  - VC\_NOCTX, 34
  - vc\_rem\_dlimit, 36
  - VC\_SAMECTX, 34
  - vc\_set\_dlimit, 36
  - vc\_tag\_create, 36
  - vc\_tag\_migrate, 36
  - vc\_tagopt2tag, 36
  - vc\_xidopt2xid, 37
  - xid\_t, 34
- xid\_t
  - vserver.h, 34