libvserver Reference Manual

2.0_rc1

Generated by Doxygen 1.5.2

Wed Jun 20 19:50:08 2007

Contents

Chapter 1

libvserver Module Index

1.1 libvserver Modules

Here is a list of all modules:

Chapter 2

libvserver Data Structure Index

2.1 libvserver Data Structures

Here are the data structures with brief descriptions:

Chapter 3

libvserver File Index

3.1 libvserver File List

Here is a list of all files with brief descriptions:

Chapter 4

libvserver Module Documentation

4.1 Context commands

Data Structures

- struct **_vx_info**
  Context information.
- struct **_vx_stat**
  Context statistics.
- struct **_vx_flags**
  Context/migration flags.
- struct **_vx_uname**
  Virtual system information data.
- struct **_vx_wait**
  Wait results.

Defines

- #define **CAP_CHOWN** 0
- #define **CAP_DAC_OVERRIDE** 1
- #define **CAP_DAC_READ_SEARCH** 2
- #define **CAP_FOWNER** 3
- #define **CAP_FSETID** 4
- #define **CAP_KILL** 5
- #define **CAP_SETGID** 6
- #define **CAP_SETUID** 7
- #define **CAP_SETPCAP** 8
- #define **CAP_LINUX_IMMUTABLE** 9
- #define **CAP_NET_BIND_SERVICE** 10
- #define **CAP_NET_BROADCAST** 11

- #define **CAP_NET_ADMIN** 12
- #define **CAP_NET_RAW** 13
- #define **CAP_IPC_LOCK** 14
- #define **CAP_IPC_OWNER** 15
- #define **CAP_SYS_MODULE** 16
- #define **CAP_SYS_RAWIO** 17
- #define **CAP_SYS_CHROOT** 18
- #define **CAP_SYS_PTRACE** 19
- #define **CAP_SYS_PACCT** 20
- #define **CAP_SYS_ADMIN** 21
- #define **CAP_SYS_BOOT** 22
- #define **CAP_SYS_NICE** 23
- #define **CAP_SYS_RESOURCE** 24
- #define **CAP_SYS_TIME** 25
- #define **CAP_SYS_TTY_CONFIG** 26
- #define **CAP_MKNOD** 27
- #define **CAP_LEASE** 28
- #define **CAP_AUDIT_WRITE** 29
- #define **CAP_AUDIT_CONTROL** 30
- #define **CAP_CONTEXT** 31
- #define **VXC_SET_UTSNAME** 0x00000001
- #define **VXC_SET_RLIMIT** 0x00000002
- #define **VXC_RAW_ICMP** 0x00000100
- #define **VXC_SYSLOG** 0x00001000
- #define **VXC_SECURE_MOUNT** 0x00010000
- #define **VXC_SECURE_REMOUNT** 0x00020000
- #define **VXC_BINARY_MOUNT** 0x00040000
- #define **VXC_QUOTA_CTL** 0x00100000
- #define **VXC_ADMIN_MAPPER** 0x00200000
- #define **VXC_ADMIN_CLOOP** 0x00400000
- #define **VXF_INFO_SCHED** 0x00000002
- #define **VXF_INFO_NPROC** 0x00000004
- #define **VXF_INFO_PRIVATE** 0x00000008
- #define **VXF_INFO_INIT** 0x00000010
- #define **VXF_INFO_HIDE** 0x00000020
- #define **VXF_INFO_ULIMIT** 0x00000040
- #define **VXF_INFO_NSPACE** 0x00000080
- #define **VXF_SCHED_HARD** 0x00000100
- #define **VXF_SCHED_PRIO** 0x00000200
- #define **VXF_SCHED_PAUSE** 0x00000400
- #define **VXF_VIRT_MEM** 0x00010000
- #define **VXF_VIRT_UPTIME** 0x00020000
- #define **VXF_VIRT_CPU** 0x00040000
- #define **VXF_VIRT_LOAD** 0x00080000
- #define **VXF_VIRT_TIME** 0x00100000
- #define **VXF_HIDE_MOUNT** 0x01000000
- #define **VXF_HIDE_NETIF** 0x02000000
- #define **VXF_HIDE_VINFO** 0x04000000
- #define **VXF_STATE_SETUP** (1ULL<<32)
- #define **VXF_STATE_INIT** (1ULL<<33)

- #define **VXF_STATE_ADMIN** (1ULL<<34)
- #define **VXF_SC_HELPER** (1ULL<<36)
- #define **VXF_REBOOT_KILL** (1ULL<<37)
- #define **VXF_PERSISTENT** (1ULL<<38)
- #define **VXF_FORK_RSS** (1ULL<<48)
- #define **VXF_PROLIFIC** (1ULL<<49)
- #define **VXF_IGNEG_NICE** (1ULL<<52)
- #define **VXM_SET_INIT** 0x00000001
- #define **VXM_SET_REAPER** 0x00000002
- #define **VHIN_CONTEXT** 0
- #define **VHIN_SYSNAME** 1
- #define **VHIN_NODENAME** 2
- #define **VHIN_RELEASE** 3
- #define **VHIN_VERSION** 4
- #define **VHIN_MACHINE** 5
- #define **VHIN_DOMAINNAME** 6

Typedefs

- typedef uint32_t **xid_t**
- typedef **_vx_info vx_info_t**

  Context information.

- typedef **_vx_stat vx_stat_t**

  Context statistics.

- typedef **_vx_flags vx_flags_t**

  Context/migration flags.

- typedef **_vx_uname vx_uname_t**

  Virtual system information data.

- typedef **_vx_wait vx_wait_t**

  Wait results.

Functions

- int **vx_create** (**xid_t** xid, **vx_flags_t** *data)

  Create a new context.

- int **vx_migrate** (**xid_t** xid, **vx_flags_t** *data)

  Migrate to an existing context.

- int **vx_task_xid** (pid_t pid)

  Get the context ID of a process.

- int **vx_info** (**xid_t** xid, **vx_info_t** *data)

  Get context information.

- int **vx_stat** (**xid_t** xid, **vx_stat_t** *data)
  Get context statistics.
- int **vx_bcaps_set** (**xid_t** xid, **vx_flags_t** *data)
  Set system capabilities.
- int **vx_bcaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get system capabilities.
- int **vx_ccaps_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context capabilities.
- int **vx_ccaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context capabilities.
- int **vx_flags_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context flags.
- int **vx_flags_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context flags.
- int **vx_uname_set** (**xid_t** xid, **vx_uname_t** *data)
  Set virtual system information.
- int **vx_uname_get** (**xid_t** xid, **vx_uname_t** *data)
  Get virtual system information.
- int **vx_kill** (**xid_t** xid, pid_t pid, int sig)
  Kill one or more processes.
- int **vx_wait** (**xid_t** xid, **vx_wait_t** *data)
  Wait for context death.

4.1.1 Define Documentation

4.1.1.1 #define CAP_CHOWN 0

Definition at line 85 of file vserver.h.

4.1.1.2 #define CAP_DAC_OVERRIDE 1

Definition at line 86 of file vserver.h.

4.1.1.3 #define CAP_DAC_READ_SEARCH 2

Definition at line 87 of file vserver.h.

4.1.1.4 #define CAP_FOWNER 3

Definition at line 88 of file vserver.h.

4.1.1.5 #define CAP_FSETID 4

Definition at line 89 of file vserver.h.

4.1.1.6 #define CAP_KILL 5

Definition at line 90 of file vserver.h.

4.1.1.7 #define CAP_SETGID 6

Definition at line 91 of file vserver.h.

4.1.1.8 #define CAP_SETUID 7

Definition at line 92 of file vserver.h.

4.1.1.9 #define CAP_SETPCAP 8

Definition at line 93 of file vserver.h.

4.1.1.10 #define CAP_LINUX_IMMUTABLE 9

Definition at line 94 of file vserver.h.

4.1.1.11 #define CAP_NET_BIND_SERVICE 10

Definition at line 95 of file vserver.h.

4.1.1.12 #define CAP_NET_BROADCAST 11

Definition at line 96 of file vserver.h.

4.1.1.13 #define CAP_NET_ADMIN 12

Definition at line 97 of file vserver.h.

4.1.1.14 #define CAP_NET_RAW 13

Definition at line 98 of file vserver.h.

4.1.1.15 #define CAP_IPC_LOCK 14

Definition at line 99 of file vserver.h.

4.1.1.16 #define CAP_IPC_OWNER 15

Definition at line 100 of file vserver.h.

4.1.1.17 #define CAP_SYS_MODULE 16

Definition at line 101 of file vserver.h.

4.1.1.18 #define CAP_SYS_RAWIO 17

Definition at line 102 of file vserver.h.

4.1.1.19 #define CAP_SYS_CHROOT 18

Definition at line 103 of file vserver.h.

4.1.1.20 #define CAP_SYS_PTRACE 19

Definition at line 104 of file vserver.h.

4.1.1.21 #define CAP_SYS_PACCT 20

Definition at line 105 of file vserver.h.

4.1.1.22 #define CAP_SYS_ADMIN 21

Definition at line 106 of file vserver.h.

4.1.1.23 #define CAP_SYS_BOOT 22

Definition at line 107 of file vserver.h.

4.1.1.24 #define CAP_SYS_NICE 23

Definition at line 108 of file vserver.h.

4.1.1.25 #define CAP_SYS_RESOURCE 24

Definition at line 109 of file vserver.h.

4.1.1.26 #define CAP_SYS_TIME 25

Definition at line 110 of file vserver.h.

4.1.1.27 #define CAP_SYS_TTY_CONFIG 26

Definition at line 111 of file vserver.h.

4.1.1.28 #define CAP_MKNOD 27

Definition at line 112 of file vserver.h.

4.1.1.29 #define CAP_LEASE 28

Definition at line 113 of file vserver.h.

4.1.1.30 #define CAP_AUDIT_WRITE 29

Definition at line 114 of file vserver.h.

4.1.1.31 #define CAP_AUDIT_CONTROL 30

Definition at line 115 of file vserver.h.

4.1.1.32 #define CAP_CONTEXT 31

Definition at line 116 of file vserver.h.

4.1.1.33 #define VXC_SET_UTSNAME 0x00000001

Allow setdomainname(2) and sethostname(2)

Definition at line 120 of file vserver.h.

4.1.1.34 #define VXC_SET_RLIMIT 0x00000002

Allow setrlimit(2)

Definition at line 121 of file vserver.h.

4.1.1.35 #define VXC_RAW_ICMP 0x00000100

Allow raw ICMP sockets

Definition at line 122 of file vserver.h.

4.1.1.36 #define VXC_SYSLOG 0x00001000

Allow syslog(2)

Definition at line 123 of file vserver.h.

4.1.1.37 #define VXC_SECURE_MOUNT 0x00010000

Allow secure mount(2)

Definition at line 124 of file vserver.h.

4.1.1.38 #define VXC_SECURE_REMOUNT 0x00020000

Allow secure remount

Definition at line 125 of file vserver.h.

4.1.1.39 #define VXC_BINARY_MOUNT 0x00040000

Allow binary/network mounts

Definition at line 126 of file vserver.h.

4.1.1.40 #define VXC_QUOTA_CTL 0x00100000

Allow quota ioctl

Definition at line 127 of file vserver.h.

4.1.1.41 #define VXC_ADMIN_MAPPER 0x00200000

Allow access to device mapper

Definition at line 128 of file vserver.h.

4.1.1.42 #define VXC_ADMIN_CLOOP 0x00400000

Allow access to loop devices

Definition at line 129 of file vserver.h.

4.1.1.43 #define VXF_INFO_SCHED 0x00000002

Account all processes as one (L)

Definition at line 131 of file vserver.h.

4.1.1.44 #define VXF_INFO_NPROC 0x00000004

Apply process limits to context (L)

Definition at line 132 of file vserver.h.

4.1.1.45 #define VXF_INFO_PRIVATE 0x00000008

Context cannot be entered

Definition at line 133 of file vserver.h.

4.1.1.46 #define VXF_INFO_INIT 0x00000010

Show a fake init process

Definition at line 134 of file vserver.h.

4.1.1.47 #define VXF_INFO_HIDE 0x00000020

Hide context information in task status

Definition at line 135 of file vserver.h.

4.1.1.48 #define VXF_INFO_ULIMIT 0x00000040

Apply ulimits to context (L)

Definition at line 136 of file vserver.h.

4.1.1.49 #define VXF_INFO_NSPACE 0x00000080

Use private namespace (L)

Definition at line 137 of file vserver.h.

4.1.1.50 #define VXF_SCHED_HARD 0x00000100

Enable hard scheduler

Definition at line 138 of file vserver.h.

4.1.1.51 #define VXF_SCHED_PRIO 0x00000200

Enable priority scheduler

Definition at line 139 of file vserver.h.

4.1.1.52 #define VXF_SCHED_PAUSE 0x00000400

Pause context (unschedule)

Definition at line 140 of file vserver.h.

4.1.1.53 #define VXF_VIRT_MEM 0x00010000

Virtualize memory information

Definition at line 141 of file vserver.h.

4.1.1.54 #define VXF_VIRT_UPTIME 0x00020000

Virtualize uptime information

Definition at line 142 of file vserver.h.

4.1.1.55 #define VXF_VIRT_CPU 0x00040000

Virtualize cpu usage information

Definition at line 143 of file vserver.h.

4.1.1.56 #define VXF_VIRT_LOAD 0x00080000

Virtualize load average information

Definition at line 144 of file vserver.h.

4.1.1.57 #define VXF_VIRT_TIME 0x00100000

Allow per guest time offsets

Definition at line 145 of file vserver.h.

4.1.1.58 #define VXF_HIDE_MOUNT 0x01000000

Hide entries in /proc/$pid/mounts

Definition at line 146 of file vserver.h.

4.1.1.59 #define VXF_HIDE_NETIF 0x02000000

Hide foreign network interfaces

Definition at line 147 of file vserver.h.

4.1.1.60 #define VXF_HIDE_VINFO 0x04000000

Hide context information in task status

Definition at line 148 of file vserver.h.

4.1.1.61 #define VXF_STATE_SETUP (1ULL<<32)

Enable setup state

Definition at line 149 of file vserver.h.

4.1.1.62 #define VXF_STATE_INIT (1ULL<<33)

Enable init state

Definition at line 150 of file vserver.h.

4.1.1.63 #define VXF_STATE_ADMIN (1ULL<<34)

Enable admin state

Definition at line 151 of file vserver.h.

4.1.1.64 #define VXF_SC_HELPER (1ULL<<36)

Context state change helper

Definition at line 152 of file vserver.h.

4.1.1.65 #define VXF_REBOOT_KILL (1ULL<<37)

Kill all processes on reboot(2)

Definition at line 153 of file vserver.h.

4.1.1.66 #define VXF_PERSISTENT (1ULL<<38)

Make context persistent

Definition at line 154 of file vserver.h.

4.1.1.67 #define VXF_FORK_RSS (1ULL<<48)

Block fork when over RSS

Definition at line 155 of file vserver.h.

4.1.1.68 #define VXF_PROLIFIC (1ULL<<49)

Allow context to create new contexts

Definition at line 156 of file vserver.h.

4.1.1.69 #define VXF_IGNEG_NICE (1ULL<<52)

Ignore priority raise

Definition at line 157 of file vserver.h.

4.1.1.70 #define VXM_SET_INIT 0x00000001

Make current process the new init

Definition at line 159 of file vserver.h.

4.1.1.71 #define VXM_SET_REAPER 0x00000002

Make current process the new reaper

Definition at line 160 of file vserver.h.

4.1.1.72 #define VHIN_CONTEXT 0

Definition at line 164 of file vserver.h.

4.1.1.73 #define VHIN_SYSNAME 1

Definition at line 165 of file vserver.h.

4.1.1.74 #define VHIN_NODENAME 2

Definition at line 166 of file vserver.h.

4.1.1.75 #define VHIN_RELEASE 3

Definition at line 167 of file vserver.h.

4.1.1.76 #define VHIN_VERSION 4

Definition at line 168 of file vserver.h.

4.1.1.77 #define VHIN_MACHINE 5

Definition at line 169 of file vserver.h.

4.1.1.78 #define VHIN_DOMAINNAME 6

Definition at line 170 of file vserver.h.

4.1.2 Typedef Documentation

4.1.2.1 typedef uint32_t xid_t

Context ID type
Definition at line 173 of file vserver.h.

4.1.2.2 typedef struct _vx_info vx_info_t

Context information.

4.1.2.3 typedef struct _vx_stat vx_stat_t

Context statistics.

4.1.2.4 typedef struct _vx_flags vx_flags_t

Context/migration flags.

4.1.2.5 typedef struct _vx_uname vx_uname_t

Virtual system information data.

4.1.2.6 typedef struct _vx_wait vx_wait_t

Wait results.

4.1.3 Function Documentation

4.1.3.1 int vx_create (xid_t *xid*, vx_flags_t * *data*)

Create a new context.

Parameters:

xid Context ID

data Initial context flags

Definition at line 61 of file context.c.

References _vx_flags::flags, and vserver().

```
62 {
63         struct vcmd_ctx_create kdata = {
64                 .flagword = 0,
65         };
66
67         if (data)
68                 kdata.flagword = data->flags;
69
70         return vserver(VCMD_ctx_create, xid, &kdata);
71 }
```

4.1.3.2 int vx_migrate (xid_t *xid*, vx_flags_t * *data*)

Migrate to an existing context.

Parameters:

xid Context ID

data Migration flags

Definition at line 73 of file context.c.

References _vx_flags::flags, and vserver().

```
74 {
75         struct vcmd_ctx_migrate kdata = {
76                 .flagword = 0,
77         };
78
79         if (data)
80                 kdata.flagword = data->flags;
81
82         return vserver(VCMD_ctx_migrate, xid, &kdata);
83 }
```

4.1.3.3 int vx_task_xid (pid_t *pid*)

Get the context ID of a process.

Parameters:

pid Process ID

Returns:

Context ID

Definition at line 85 of file context.c.

References vserver().

```
86 {
87         return vserver(VCMD_task_xid, pid, NULL);
88 }
```

4.1.3.4 int vx_info (xid_t *xid*, vx_info_t * *data*)

Get context information.

Parameters:

xid Context ID

data Empty vx_info_t struct to be filled

Definition at line 90 of file context.c.

References _vx_info::initpid, vserver(), and _vx_info::xid.

```
91 {
92         struct vcmd_vx_info_v0 kdata;
93
94         int rc = vserver(VCMD_vx_info, xid, &kdata);
95
96         if (rc == -1)
97                 return rc;
98
99         if (data) {
100                 data->xid     = kdata.xid;
101                 data->initpid = kdata.initpid;
102         }
103
104         return rc;
105 }
```

4.1.3.5 int vx_stat (xid_t *xid*, vx_stat_t * *data*)

Get context statistics.

Parameters:

xid Context ID

data Empty vx_stat_t struct to be filled

Definition at line 107 of file context.c.

References _vx_stat::load, _vx_stat::nr_forks, _vx_stat::nr_onhold, _vx_stat::nr_running, _vx_stat::nr_threads, _vx_stat::nr_unintr, _vx_stat::offset, _vx_stat::tasks, _vx_stat::uptime, _vx_stat::usecnt, and vserver().

```
108 {
109         int rc;
110         struct vcmd_ctx_stat_v0 kdata1;
111         struct vcmd_virt_stat_v0 kdata2;
112 
113         rc = vserver(VCMD_ctx_stat, xid, &kdata1);
114 
115         if (rc == -1)
116                 return rc;
117 
118         rc = vserver(VCMD_virt_stat, xid, &kdata2);
119 
120         if (rc == -1)
121                 return rc;
122 
123         if (data) {
124                 data->usecnt     = kdata1.usecnt;
125                 data->tasks      = kdata1.tasks;
126                 data->nr_threads = kdata2.nr_threads;
127                 data->nr_running = kdata2.nr_running;
128                 data->nr_unintr  = kdata2.nr_uninterruptible;
129                 data->nr_onhold  = kdata2.nr_onhold;
130                 data->nr_forks   = kdata2.nr_forks;
131                 data->load[0]    = kdata2.load[0];
132                 data->load[1]    = kdata2.load[1];
133                 data->load[2]    = kdata2.load[2];
134                 data->offset     = kdata2.offset;
135                 data->uptime     = kdata2.uptime;
136         }
137 
138         return rc;
139 }
```

4.1.3.6 int vx_bcaps_set (xid_t *xid*, vx_flags_t * *data*)

Set system capabilities.

Parameters:

xid Context ID

data System capabilities

Definition at line 141 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
142 {
143         struct vcmd_bcaps kdata;
144 
145         if (!data)
146                 return errno = EINVAL, -1;
147 
148         kdata.bcaps = data->flags;
149         kdata.bmask = data->mask;
150 
151         return vserver(VCMD_set_bcaps, xid, &kdata);
152 }
```

4.1.3.7 int vx_bcaps_get (xid_t *xid*, vx_flags_t * *data*)

Get system capabilities.

Parameters:

xid Context ID

data Empty vx_flags_t struct to be filled

Definition at line 154 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
155 {
156         struct vcmd_bcaps kdata;
157
158         if (!data)
159                 return errno = EINVAL, -1;
160
161         int rc = vserver(VCMD_get_bcaps, xid, &kdata);
162
163         if (rc == -1)
164                 return rc;
165
166         data->flags = kdata.bcaps;
167         data->mask  = kdata.bmask;
168
169         return rc;
170 }
```

4.1.3.8 int vx_ccaps_set (xid_t *xid*, vx_flags_t * *data*)

Set context capabilities.

Parameters:

xid Context ID

data Context capabilities

Definition at line 172 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
173 {
174         struct vcmd_ctx_caps_v1 kdata;
175
176         if (!data)
177                 return errno = EINVAL, -1;
178
179         kdata.ccaps = data->flags;
180         kdata.cmask = data->mask;
181
182         return vserver(VCMD_set_ccaps, xid, &kdata);
183 }
```

4.1.3.9 int vx_ccaps_get (xid_t *xid*, vx_flags_t * *data*)

Get context capabilities.

Parameters:

xid Context ID

data Empty vx_flags_t struct to be filled

Definition at line 185 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
186 {
187         struct vcmd_ctx_caps_v1 kdata;
188
189         if (!data)
190                 return errno = EINVAL, -1;
191
192         int rc = vserver(VCMD_get_ccaps, xid, &kdata);
193
194         if (rc == -1)
195                 return rc;
196
197         data->flags = kdata.ccaps;
198         data->mask  = kdata.cmask;
199
200         return rc;
201 }
```

4.1.3.10 int vx_flags_set (xid_t *xid*, vx_flags_t * *data*)

Set context flags.

Parameters:

xid Context ID

data Context flags

Definition at line 203 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
204 {
205         struct vcmd_ctx_flags_v0 kdata;
206
207         if (!data)
208                 return errno = EINVAL, -1;
209
210         kdata.flagword = data->flags;
211         kdata.mask     = data->mask;
212
213         return vserver(VCMD_set_cflags, xid, &kdata);
214 }
```

4.1.3.11 int vx_flags_get (xid_t *xid*, vx_flags_t * *data*)

Get context flags.

Parameters:

xid Context ID

data Empty vx_flags_t struct to be filled

Definition at line 216 of file context.c.

References _vx_flags::flags, _vx_flags::mask, and vserver().

```
217 {
218         struct vcmd_ctx_flags_v0 kdata;
219
220         if (!data)
221                 return errno = EINVAL, -1;
222
223         int rc = vserver(VCMD_get_cflags, xid, &kdata);
224
225         if (rc == -1)
226                 return rc;
227
228         data->flags = kdata.flagword;
229         data->mask  = kdata.mask;
230
231         return rc;
232 }
```

4.1.3.12 int vx_uname_set (xid_t *xid*, vx_uname_t * *data*)

Set virtual system information.

Parameters:

xid Context ID

data Virtual system information data

Definition at line 234 of file context.c.

References _vx_uname::id, _vx_uname::value, and vserver().

```
235 {
236         struct vcmd_vhi_name_v0 kdata;
237
238         if (!data)
239                 return errno = EINVAL, -1;
240
241         if (str_len(data->value) >= sizeof(kdata.name))
242                 return errno = EINVAL, -1;
243
244         kdata.field = data->id;
245
246         str_zero(kdata.name, sizeof(kdata.name));
247         str_cpyn(kdata.name, data->value, sizeof(kdata.name) - 1);
248
249         return vserver(VCMD_set_vhi_name, xid, &kdata);
250 }
```

4.1.3.13 int vx_uname_get (xid_t *xid*, vx_uname_t * *data*)

Get virtual system information.

Parameters:

xid Context ID

data Empty vx_uname_t struct to be filled

Definition at line 252 of file context.c.

References _vx_uname::id, _vx_uname::value, and vserver().

```
253 {
254         int rc;
255         struct vcmd_vhi_name_v0 kdata;
256
257         if (!data)
258                 return errno = EINVAL, -1;
259
260         kdata.field = data->id;
261
262         rc = vserver(VCMD_get_vhi_name, xid, &kdata);
263
264         if (rc == -1)
265                 return rc;
266
267         str_zero(data->value, sizeof(kdata.name));
268         str_cpyn(data->value, kdata.name, sizeof(kdata.name) - 1);
269
270         return rc;
271 }
```

4.1.3.14 int vx_kill (xid_t *xid*, pid_t *pid*, int *sig*)

Kill one or more processes.

Parameters:

xid Context ID

pid Process ID

sig Signal number

Definition at line 273 of file context.c.

References vserver().

```
274 {
275         struct vcmd_ctx_kill_v0 kdata;
276
277         kdata.pid = pid;
278         kdata.sig = sig;
279
280         return vserver(VCMD_ctx_kill, xid, &kdata);
281 }
```

4.1.3.15 int vx_wait (xid_t *xid*, vx_wait_t * *data*)

Wait for context death.

Parameters:

xid Context ID

data Empty vx_wait_t struct to be filled

Definition at line 283 of file context.c.

References _vx_wait::exit_code, _vx_wait::reboot_cmd, and vserver().

```
284 {
285         struct vcmd_wait_exit_v0 kdata;
286
287         int rc;
288
289         rc = vserver(VCMD_wait_exit, xid, &kdata);
290
291         if (rc == -1)
292                 return -1;
293
294         if (data) {
295                 data->reboot_cmd = kdata.reboot_cmd;
296                 data->exit_code  = kdata.exit_code;
297         }
298
299         return rc;
300 }
```

4.2 CPU scheduler commands

Data Structures

- struct **_vx_sched**
  Scheduler values.
- struct **_vx_sched_info**
  Scheduler information.

Defines

- #define **VXSM_FILL_RATE** 0x0001
- #define **VXSM_INTERVAL** 0x0002
- #define **VXSM_FILL_RATE2** 0x0004
- #define **VXSM_INTERVAL2** 0x0008
- #define **VXSM_TOKENS** 0x0010
- #define **VXSM_TOKENS_MIN** 0x0020
- #define **VXSM_TOKENS_MAX** 0x0040
- #define **VXSM_PRIO_BIAS** 0x0100
- #define **VXSM_IDLE_TIME** 0x0200
- #define **VXSM_FORCE** 0x0400
- #define **VXSM_CPU_ID** 0x1000
- #define **VXSM_BUCKET_ID** 0x2000
- #define **VXSM_MSEC** 0x4000
- #define **VXSM_V3_MASK** 0x0173
- #define **VXSM_SET_MASK** 0x01FF

Typedefs

- typedef **_vx_sched vx_sched_t**
  Scheduler values.
- typedef **_vx_sched_info vx_sched_info_t**
  Scheduler information.

Functions

- int **vx_sched_set** (**xid_t** xid, **vx_sched_t** *data)
  Set scheduler values.
- int **vx_sched_get** (**xid_t** xid, **vx_sched_t** *data)
  Get scheduler values.
- int **vx_sched_info** (**xid_t** xid, **vx_sched_info_t** *data)
  Get scheduler information.

4.2.1 Define Documentation

4.2.1.1 #define VXSM_FILL_RATE 0x0001

Fill Rate

Definition at line 353 of file vserver.h.

4.2.1.2 #define VXSM_INTERVAL 0x0002

Interval

Definition at line 354 of file vserver.h.

4.2.1.3 #define VXSM_FILL_RATE2 0x0004

IDLE Fill Rate

Definition at line 355 of file vserver.h.

4.2.1.4 #define VXSM_INTERVAL2 0x0008

IDLE Interval

Definition at line 356 of file vserver.h.

4.2.1.5 #define VXSM_TOKENS 0x0010

Amount of tokens

Definition at line 357 of file vserver.h.

4.2.1.6 #define VXSM_TOKENS_MIN 0x0020

Minimum amount of tokens

Definition at line 358 of file vserver.h.

4.2.1.7 #define VXSM_TOKENS_MAX 0x0040

Maximum amount of tokens

Definition at line 359 of file vserver.h.

4.2.1.8 #define VXSM_PRIO_BIAS 0x0100

Priority bias

Definition at line 360 of file vserver.h.

4.2.1.9 #define VXSM_IDLE_TIME 0x0200

Use IDLE time settings

Definition at line 362 of file vserver.h.

4.2.1.10 #define VXSM_FORCE 0x0400

Force scheduler reload (SMP only)

Definition at line 363 of file vserver.h.

4.2.1.11 #define VXSM_CPU_ID 0x1000

CPU ID (SMP only)

Definition at line 364 of file vserver.h.

4.2.1.12 #define VXSM_BUCKET_ID 0x2000

Bucket ID

Definition at line 365 of file vserver.h.

4.2.1.13 #define VXSM_MSEC 0x4000

??

Definition at line 366 of file vserver.h.

4.2.1.14 #define VXSM_V3_MASK 0x0173

Mask all fields for scheduler v3

Definition at line 368 of file vserver.h.

4.2.1.15 #define VXSM_SET_MASK 0x01FF

Mask all fields for set_sched

Definition at line 369 of file vserver.h.

4.2.2 Typedef Documentation

4.2.2.1 typedef struct _vx_sched vx_sched_t

Scheduler values.

4.2.2.2 typedef struct _vx_sched_info vx_sched_info_t

Scheduler information.

4.2.3 Function Documentation

4.2.3.1 int vx_sched_set (xid_t *xid*, vx_sched_t * *data*)

Set scheduler values.

Parameters:

xid Context ID

data Scheduler values

Definition at line 53 of file sched.c.

References _vx_sched::cpu_id, _vx_sched::fill_rate, _vx_sched::interval, _vx_sched::mask, _vx_sched::prio_bias, _vx_sched::tokens, _vx_sched::tokens_max, _vx_sched::tokens_min, and vserver().

```
54 {
55         struct vcmd_sched_v5 kdata;
56
57         if (!data)
58                 return errno = EINVAL, -1;
59
60         kdata.mask         = data->mask;
61         kdata.cpu_id       = data->cpu_id;
62         kdata.fill_rate[0] = data->fill_rate[0];
63         kdata.fill_rate[1] = data->fill_rate[1];
64         kdata.interval[0]  = data->interval[0];
65         kdata.interval[1]  = data->interval[1];
66         kdata.tokens       = data->tokens;
67         kdata.tokens_min   = data->tokens_min;
68         kdata.tokens_max   = data->tokens_max;
69         kdata.prio_bias    = data->prio_bias;
70
71         return vserver(VCMD_set_sched, xid, &kdata);
72 }
```

4.2.3.2 int vx_sched_get (xid_t *xid*, vx_sched_t * *data*)

Get scheduler values.

Parameters:

xid Context ID

data Scheduler values

Definition at line 25 of file sched.c.

References _vx_sched::bucket_id, _vx_sched::cpu_id, _vx_sched::fill_rate, _vx_sched::interval, _vx_sched::mask, _vx_sched::prio_bias, _vx_sched::tokens, _vx_sched::tokens_max, _vx_sched::tokens_min, and vserver().

```
26 {
27         struct vcmd_sched_v5 kdata;
28
29         if (!data)
30                 return errno = EINVAL, -1;
31
```

```
32         kdata.cpu_id    = data->cpu_id;
33         kdata.bucket_id = data->bucket_id;
34         kdata.mask      = data->mask;
35
36         int rc = vserver(VCMD_get_sched, xid, &kdata);
37
38         if (rc == -1)
39                 return -1;
40
41         data->fill_rate[0] = kdata.fill_rate[0];
42         data->fill_rate[1] = kdata.fill_rate[1];
43         data->interval[0]  = kdata.interval[0];
44         data->interval[1]  = kdata.interval[1];
45         data->tokens       = kdata.tokens;
46         data->tokens_min   = kdata.tokens_min;
47         data->tokens_max   = kdata.tokens_max;
48         data->prio_bias    = kdata.prio_bias;
49
50         return rc;
51 }
```

4.2.3.3 int vx_sched_info (xid_t *xid*, vx_sched_info_t * *data*)

Get scheduler information.

Parameters:

xid Context ID

data Scheduler information

Definition at line 74 of file sched.c.

References _vx_sched_info::bucket_id, _vx_sched_info::cpu_id, _vx_sched_info::hold_msec, _vx_sched_info::sys_msec, _vx_sched_info::token_usec, _vx_sched_info::user_msec, _vx_sched_info::vavavoom, and vserver().

```
75 {
76         struct vcmd_sched_info kdata;
77
78         if (!data)
79                 return errno = EINVAL, -1;
80
81         kdata.cpu_id = data->cpu_id;
82         kdata.bucket_id = data->bucket_id;
83
84         int rc = vserver(VCMD_sched_info, xid, &kdata);
85
86         if (rc == -1)
87                 return -1;
88
89         data->user_msec  = kdata.user_msec;
90         data->sys_msec   = kdata.sys_msec;
91         data->hold_msec  = kdata.hold_msec;
92         data->token_usec = kdata.token_usec;
93         data->vavavoom   = kdata.vavavoom;
94
95         return rc;
96 }
```

4.3 Resource limit commands

Data Structures

- struct **_vx_limit**
  Resource limits.
- struct **_vx_limit_stat**
  Resource limit accounting.

Defines

- #define **CRLIM_UNSET** (0ULL)
- #define **CRLIM_INFINITY** (~0ULL)
- #define **CRLIM_KEEP** (~1ULL)
- #define **VLIMIT_NSOCK** 16
- #define **VLIMIT_OPENFD** 17
- #define **VLIMIT_ANON** 18
- #define **VLIMIT_SHMEM** 19
- #define **VLIMIT_SEMARY** 20
- #define **VLIMIT_NSEMS** 21
- #define **VLIMIT_DENTRY** 22
- #define **VLIMIT_MAPPED** 23

Typedefs

- typedef **_vx_limit** **vx_limit_t**
  Resource limits.
- typedef **_vx_limit_stat** **vx_limit_stat_t**
  Resource limit accounting.

Functions

- int **vx_limit_mask_get** (**vx_limit_t** *data)
  Get resource limits mask.
- int **vx_limit_set** (**xid_t** xid, **vx_limit_t** *data)
  Set resource limit.
- int **vx_limit_get** (**xid_t** xid, **vx_limit_t** *data)
  Get resource limits.
- int **vx_limit_stat** (**xid_t** xid, **vx_limit_stat_t** *data)
  Get resource limit accounting data.
- int **vx_limit_reset** (**xid_t** xid)
  Reset resource limit accounting data.

4.3.1 Define Documentation

4.3.1.1 #define CRLIM_UNSET (0ULL)

Unset resource limit

Definition at line 434 of file vserver.h.

4.3.1.2 #define CRLIM_INFINITY (~0ULL)

Infinity (no limit)

Definition at line 435 of file vserver.h.

4.3.1.3 #define CRLIM_KEEP (~1ULL)

Keep current value

Definition at line 436 of file vserver.h.

4.3.1.4 #define VLIMIT_NSOCK 16

Number of open sockets

Definition at line 440 of file vserver.h.

4.3.1.5 #define VLIMIT_OPENFD 17

Number of open file descriptors

Definition at line 441 of file vserver.h.

4.3.1.6 #define VLIMIT_ANON 18

Amount of anonymous memory

Definition at line 442 of file vserver.h.

4.3.1.7 #define VLIMIT_SHMEM 19

Amount of shared memory

Definition at line 443 of file vserver.h.

4.3.1.8 #define VLIMIT_SEMARY 20

Size of semary

Definition at line 444 of file vserver.h.

4.3.1.9 #define VLIMIT_NSEMS 21

Number of semaphores

Definition at line 445 of file vserver.h.

4.3.1.10 #define VLIMIT_DENTRY 22

Size of the dentry cache

Definition at line 446 of file vserver.h.

4.3.1.11 #define VLIMIT_MAPPED 23

??

Definition at line 447 of file vserver.h.

4.3.2 Typedef Documentation

4.3.2.1 typedef struct _vx_limit vx_limit_t

Resource limits.

4.3.2.2 typedef struct _vx_limit_stat vx_limit_stat_t

Resource limit accounting.

4.3.3 Function Documentation

4.3.3.1 int vx_limit_mask_get (vx_limit_t * *data*)

Get resource limits mask.

Parameters:

data Empty vx_limit_t struct to be filled

Definition at line 26 of file limit.c.

References _vx_limit::maximum, _vx_limit::minimum, _vx_limit::softlimit, and vserver().

```
27 {
28         int rc;
29         struct vcmd_ctx_rlimit_mask_v0 kdata;
30
31         if (!data)
32                 return errno = EINVAL, -1;
33
34         rc = vserver(VCMD_get_rlimit_mask, 0, &kdata);
35
36         if (rc == -1)
37                 return rc;
38
39         data->minimum   = kdata.minimum;
40         data->softlimit = kdata.softlimit;
41         data->maximum   = kdata.maximum;
42
43         return rc;
44 }
```

4.3.3.2 int vx_limit_set (xid_t *xid*, vx_limit_t * *data*)

Set resource limit.

Parameters:

xid Context ID
data Resource limits

Definition at line 46 of file limit.c.

References _vx_limit::id, _vx_limit::maximum, _vx_limit::minimum, _vx_limit::softlimit, and vserver().

```
47 {
48         struct vcmd_ctx_rlimit_v0 kdata;
49
50         if (!data)
51                 return errno = EINVAL, -1;
52
53         kdata.id        = data->id;
54         kdata.minimum   = data->minimum;
55         kdata.softlimit = data->softlimit;
56         kdata.maximum   = data->maximum;
57
58         return vserver(VCMD_set_rlimit, xid, &kdata);
59 }
```

4.3.3.3 int vx_limit_get (xid_t *xid*, vx_limit_t * *data*)

Get resource limits.

Parameters:

xid Context ID
data Empty vx_limit_t struct to be filled

Definition at line 61 of file limit.c.

References _vx_limit::id, _vx_limit::maximum, _vx_limit::minimum, _vx_limit::softlimit, and vserver().

```
62 {
63         int rc;
64         struct vcmd_ctx_rlimit_v0 kdata;
65
66         if (!data)
67                 return errno = EINVAL, -1;
68
69         kdata.id = data->id;
70
71         rc = vserver(VCMD_get_rlimit, xid, &kdata);
72
73         if (rc == -1)
74                 return rc;
75
76         data->minimum   = kdata.minimum;
77         data->softlimit = kdata.softlimit;
78         data->maximum   = kdata.maximum;
79
80         return rc;
81 }
```

4.3.3.4 int vx_limit_stat (xid_t *xid*, vx_limit_stat_t * *data*)

Get resource limit accounting data.

Parameters:

xid Context ID

data Empty vx_limit_stat_t struct to be filled

Definition at line 83 of file limit.c.

References _vx_limit_stat::hits, _vx_limit_stat::id, _vx_limit_stat::maximum, _vx_limit_stat::minimum, _vx_limit_stat::value, and vserver().

```
84 {
85         int rc;
86         struct vcmd_rlimit_stat_v0 kdata;
87
88         if (!data)
89                 return errno = EINVAL, -1;
90
91         kdata.id = data->id;
92
93         rc = vserver(VCMD_rlimit_stat, xid, &kdata);
94
95         if (rc == -1)
96                 return rc;
97
98         data->hits    = kdata.hits;
99         data->value   = kdata.value;
100         data->minimum = kdata.minimum;
101         data->maximum = kdata.maximum;
102
103         return rc;
104 }
```

4.3.3.5 int vx_limit_reset (xid_t *xid*)

Reset resource limit accounting data.

Parameters:

xid Context ID

Definition at line 106 of file limit.c.

References vserver().

```
107 {
108         return vserver(VCMD_reset_minmax, xid, NULL);
109 }
```

4.4 Disk limit commands

Data Structures

- struct **_dx_limit**
  Disk limit values.

Defines

- #define **CDLIM_UNSET** ((uint32_t)0UL)
- #define **CDLIM_INFINITY** ((uint32_t)~0UL)
- #define **CDLIM_KEEP** ((uint32_t)~1UL)

Typedefs

- typedef **_dx_limit dx_limit_t**
  Disk limit values.

Functions

- int **dx_limit_add** (**xid_t** xid, **dx_limit_t** *data)
  Add disk limit entry.
- int **dx_limit_remove** (**xid_t** xid, **dx_limit_t** *data)
  Remove disk limit.
- int **dx_limit_set** (**xid_t** xid, **dx_limit_t** *data)
  Set disk limit values.
- int **dx_limit_get** (**xid_t** xid, **dx_limit_t** *data)
  Get disk limit values.

4.4.1 Define Documentation

4.4.1.1 #define CDLIM_UNSET ((uint32_t)0UL)

Unset disk limit

Definition at line 517 of file vserver.h.

4.4.1.2 #define CDLIM_INFINITY ((uint32_t)~0UL)

Infinity (no limit)

Definition at line 518 of file vserver.h.

4.4.1.3 #define CDLIM_KEEP ((uint32_t)~1UL)

Keep current value

Definition at line 519 of file vserver.h.

4.4.2 Typedef Documentation

4.4.2.1 typedef struct _dx_limit dx_limit_t

Disk limit values.

4.4.3 Function Documentation

4.4.3.1 int dx_limit_add (xid_t *xid*, dx_limit_t * *data*)

Add disk limit entry.

Parameters:

xid Context ID

data Disk limit information

Definition at line 26 of file dlimit.c.

References _dx_limit::filename, _dx_limit::flags, and vserver().

```
27 {
28         struct vcmd_ctx_dlimit_base_v0 kdata;
29
30         if (!data)
31                 return errno = EINVAL, -1;
32
33         kdata.name  = data->filename;
34         kdata.flags = data->flags;
35
36         return vserver(VCMD_add_dlimit, xid, &kdata);
37 }
```

4.4.3.2 int dx_limit_remove (xid_t *xid*, dx_limit_t * *data*)

Remove disk limit.

Parameters:

xid Context ID

data Disk limit information

Definition at line 39 of file dlimit.c.

References _dx_limit::filename, and vserver().

```
40 {
41         struct vcmd_ctx_dlimit_base_v0 kdata;
```

```
42
43         if (!data)
44                 return errno = EINVAL, -1;
45
46         kdata.name = data->filename;
47
48         return vserver(VCMD_rem_dlimit, xid, &kdata);
49 }
```

4.4.3.3 int dx_limit_set (xid_t *xid*, dx_limit_t * *data*)

Set disk limit values.

Parameters:

xid Context ID

data Disk limit values

Definition at line 51 of file dlimit.c.

References _dx_limit::filename, _dx_limit::flags, _dx_limit::inodes_total, _dx_limit::inodes_-used, _dx_limit::reserved, _dx_limit::space_total, _dx_limit::space_used, and vserver().

```
52 {
53         struct vcmd_ctx_dlimit_v0 kdata;
54
55         if (!data)
56                 return errno = EINVAL, -1;
57
58         kdata.name         = data->filename;
59         kdata.space_used   = data->space_used;
60         kdata.space_total  = data->space_total;
61         kdata.inodes_used  = data->inodes_used;
62         kdata.inodes_total = data->inodes_total;
63         kdata.reserved     = data->reserved;
64         kdata.flags        = data->flags;
65
66         return vserver(VCMD_set_dlimit, xid, &kdata);
67 }
```

4.4.3.4 int dx_limit_get (xid_t *xid*, dx_limit_t * *data*)

Get disk limit values.

Parameters:

xid Context ID

data Empty dx_limit_t struct to be filled

Definition at line 69 of file dlimit.c.

References _dx_limit::filename, _dx_limit::flags, _dx_limit::inodes_total, _dx_limit::inodes_-used, _dx_limit::reserved, _dx_limit::space_total, _dx_limit::space_used, and vserver().

```
70 {
71         int rc;
72         struct vcmd_ctx_dlimit_v0 kdata;
```

```
73
74         if (!data)
75                 return errno = EINVAL, -1;
76
77         kdata.name = data->filename;
78
79         rc = vserver(VCMD_get_dlimit, xid, &kdata);
80
81         if (rc == -1)
82                 return rc;
83
84         data->space_used   = kdata.space_used;
85         data->space_total  = kdata.space_total;
86         data->inodes_used  = kdata.inodes_used;
87         data->inodes_total = kdata.inodes_total;
88         data->reserved     = kdata.reserved;
89         data->flags        = kdata.flags;
90
91         return rc;
92 }
```

4.5 Inode attribute commands

Data Structures

- struct **_ix_attr**

  Inode attributes.

Defines

- #define **IATTR_TAG** 0x01000000
- #define **IATTR_ADMIN** 0x00000001
- #define **IATTR_WATCH** 0x00000002
- #define **IATTR_HIDE** 0x00000004
- #define **IATTR_FLAGS** 0x00000007
- #define **IATTR_BARRIER** 0x00010000
- #define **IATTR_IUNLINK** 0x00020000
- #define **IATTR_IMMUTABLE** 0x00040000

Typedefs

- typedef **_ix_attr ix_attr_t**

  Inode attributes.

Functions

- int **ix_attr_set** (**ix_attr_t** *data)

  Set inode attributes.

- int **ix_attr_get** (**ix_attr_t** *data)

  Get inode attributes.

4.5.1 Define Documentation

4.5.1.1 #define IATTR_TAG 0x01000000

File is xid tagged

Definition at line 575 of file vserver.h.

4.5.1.2 #define IATTR_ADMIN 0x00000001

Accessible in xid=0 (only for /proc)

Definition at line 576 of file vserver.h.

4.5.1.3 #define IATTR_WATCH 0x00000002

Accessible in xid=1 (only for /proc)

Definition at line 577 of file vserver.h.

4.5.1.4 #define IATTR_HIDE 0x00000004

Not Accessible in xid!=(0|1) (only for /proc)

Definition at line 578 of file vserver.h.

4.5.1.5 #define IATTR_FLAGS 0x00000007

Flag mask for /proc flags

Definition at line 579 of file vserver.h.

4.5.1.6 #define IATTR_BARRIER 0x00010000

Directory barrier

Definition at line 580 of file vserver.h.

4.5.1.7 #define IATTR_IUNLINK 0x00020000

Unlink file

Definition at line 581 of file vserver.h.

4.5.1.8 #define IATTR_IMMUTABLE 0x00040000

File is immutable

Definition at line 582 of file vserver.h.

4.5.2 Typedef Documentation

4.5.2.1 typedef struct _ix_attr ix_attr_t

Inode attributes.

4.5.3 Function Documentation

4.5.3.1 int ix_attr_set (ix_attr_t * *data*)

Set inode attributes.

Parameters:

data Inode attributes

Definition at line 26 of file inode.c.

References _ix_attr::filename, _ix_attr::flags, _ix_attr::mask, vserver(), and _ix_attr::xid.

```
27 {
28         struct vcmd_ctx_iattr_v1 kdata;
29
30         if (!data)
31                 return errno = EINVAL, -1;
32
33         kdata.name  = data->filename;
34         kdata.xid   = data->xid;
35         kdata.flags = data->flags;
36         kdata.mask  = data->mask;
37
38         return vserver(VCMD_set_iattr, 0, &kdata);
39 }
```

4.5.3.2 int ix_attr_get (ix_attr_t * *data*)

Get inode attributes.

Parameters:

data Empty ix_attr_t struct to be filled

Definition at line 41 of file inode.c.

References _ix_attr::filename, _ix_attr::flags, _ix_attr::mask, vserver(), and _ix_attr::xid.

```
42 {
43         int rc;
44         struct vcmd_ctx_iattr_v1 kdata;
45
46         if (!data)
47                 return errno = EINVAL, -1;
48
49         kdata.name = data->filename;
50
51         rc = vserver(VCMD_get_iattr, 0, &kdata);
52
53         if (rc == -1)
54                 return rc;
55
56         data->xid   = kdata.xid;
57         data->flags = kdata.flags;
58         data->mask  = kdata.mask;
59
60         return rc;
61 }
```

4.6 Namespace commands

Defines

- #define **CLONE_VM** 0x00000100
- #define **CLONE_FS** 0x00000200
- #define **CLONE_FILES** 0x00000400
- #define **CLONE_SIGHAND** 0x00000800
- #define **CLONE_PTRACE** 0x00002000
- #define **CLONE_VFORK** 0x00004000
- #define **CLONE_PARENT** 0x00008000
- #define **CLONE_THREAD** 0x00010000
- #define **CLONE_NEWNS** 0x00020000
- #define **CLONE_SYSVSEM** 0x00040000
- #define **CLONE_SETTLS** 0x00080000
- #define **CLONE_PARENT_SETTID** 0x00100000
- #define **CLONE_CHILD_CLEARTID** 0x00200000
- #define **CLONE_DETACHED** 0x00400000
- #define **CLONE_UNTRACED** 0x00800000
- #define **CLONE_CHILD_SETTID** 0x01000000
- #define **CLONE_STOPPED** 0x02000000
- #define **CLONE_NEWUTS** 0x04000000
- #define **CLONE_NEWIPC** 0x08000000
- #define **CLONE_KTHREAD** 0x10000000

Functions

- int **ns_clone** (int flags, void *child_stack)

  Clone the current namespace (FS/IPC/UTS).

- int **ns_enter** (**xid_t** xid, uint64_t mask)

  Enter namespace.

- int **ns_set** (**xid_t** xid, uint64_t mask)

  Set namespace.

4.6.1 Define Documentation

4.6.1.1 #define CLONE_VM 0x00000100

Definition at line 617 of file vserver.h.

4.6.1.2 #define CLONE_FS 0x00000200

Definition at line 618 of file vserver.h.

4.6.1.3 #define CLONE_FILES 0x00000400

Definition at line 619 of file vserver.h.

4.6.1.4 #define CLONE_SIGHAND 0x00000800

Definition at line 620 of file vserver.h.

4.6.1.5 #define CLONE_PTRACE 0x00002000

Definition at line 621 of file vserver.h.

4.6.1.6 #define CLONE_VFORK 0x00004000

Definition at line 622 of file vserver.h.

4.6.1.7 #define CLONE_PARENT 0x00008000

Definition at line 623 of file vserver.h.

4.6.1.8 #define CLONE_THREAD 0x00010000

Definition at line 624 of file vserver.h.

4.6.1.9 #define CLONE_NEWNS 0x00020000

Definition at line 625 of file vserver.h.
Referenced by ns_clone().

4.6.1.10 #define CLONE_SYSVSEM 0x00040000

Definition at line 626 of file vserver.h.

4.6.1.11 #define CLONE_SETTLS 0x00080000

Definition at line 627 of file vserver.h.

4.6.1.12 #define CLONE_PARENT_SETTID 0x00100000

Definition at line 628 of file vserver.h.

4.6.1.13 #define CLONE_CHILD_CLEARTID 0x00200000

Definition at line 629 of file vserver.h.

4.6.1.14 #define CLONE_DETACHED 0x00400000

Definition at line 630 of file vserver.h.

4.6.1.15 #define CLONE_UNTRACED 0x00800000

Definition at line 631 of file vserver.h.

4.6.1.16 #define CLONE_CHILD_SETTID 0x01000000

Definition at line 632 of file vserver.h.

4.6.1.17 #define CLONE_STOPPED 0x02000000

Definition at line 633 of file vserver.h.

4.6.1.18 #define CLONE_NEWUTS 0x04000000

Definition at line 634 of file vserver.h.

Referenced by ns_clone().

4.6.1.19 #define CLONE_NEWIPC 0x08000000

Definition at line 635 of file vserver.h.

Referenced by ns_clone().

4.6.1.20 #define CLONE_KTHREAD 0x10000000

Definition at line 636 of file vserver.h.

4.6.2 Function Documentation

4.6.2.1 int ns_clone (int *flags*, void * *child_stack*)

Clone the current namespace (FS/IPC/UTS).

Parameters:

flags Clone flags

child_stack Child stack

Definition at line 25 of file namespace.c.

References clone(), CLONE_NEWIPC, CLONE_NEWNS, and CLONE_NEWUTS.

```
26 {
27         return clone(flags|CLONE_NEWNS|CLONE_NEWUTS|CLONE_NEWIPC, child_stack);
28 }
```

4.6.2.2 int ns_enter (xid_t *xid*, uint64_t *mask*)

Enter namespace.

Parameters:

xid Context ID

Definition at line 30 of file namespace.c.

References vserver().

```
31 {
32         struct vcmd_space_mask kdata = { .mask = mask };
33         return vserver(VCMD_enter_space, xid, &kdata);
34 }
```

4.6.2.3 int ns_set (xid_t *xid*, uint64_t *mask*)

Set namespace.

Parameters:

xid Context ID

Definition at line 36 of file namespace.c.

References vserver().

```
37 {
38         struct vcmd_space_mask kdata = { .mask = mask };
39         return vserver(VCMD_set_space, xid, &kdata);
40 }
```

4.7 Network context commands

Data Structures

- struct **_nx_info**
  Network context information.
- struct **_nx_addr**
  Network address information.
- struct **_nx_flags**
  Network context flags.
- struct **_nx_sock_stat**
  Accounting data.

Defines

- #define **NXF_INFO_PRIVATE** 0x00000008
- #define **NXF_STATE_SETUP** (1ULL<<32)
- #define **NXF_STATE_ADMIN** (1ULL<<34)
- #define **NXF_SC_HELPER** (1ULL<<36)
- #define **NXF_PERSISTENT** (1ULL<<38)
- #define **NXA_TYPE_IPV4** 1
- #define **NXA_TYPE_IPV6** 2
- #define **NXA_MOD_BCAST** (1<<8)
- #define **NXA_TYPE_ANY** ((uint16_t)-1)
- #define **VXA_SOCK_UNSPEC** 0
- #define **VXA_SOCK_UNIX** 1
- #define **VXA_SOCK_INET** 2
- #define **VXA_SOCK_INET6** 3
- #define **VXA_SOCK_PACKET** 4
- #define **VXA_SOCK_OTHER** 5
- #define **NXA_SOCK_UNSPEC** VXA_SOCK_UNSPEC
- #define **NXA_SOCK_UNIX** VXA_SOCK_UNIX
- #define **NXA_SOCK_INET** VXA_SOCK_INET
- #define **NXA_SOCK_INET6** VXA_SOCK_INET6
- #define **NXA_SOCK_PACKET** VXA_SOCK_PACKET
- #define **NXA_SOCK_OTHER** VXA_SOCK_OTHER

Typedefs

- typedef uint32_t **nid_t**
- typedef **_nx_info nx_info_t**
  Network context information.
- typedef **_nx_addr nx_addr_t**
  Network address information.

- typedef **_nx_flags nx_flags_t**
  Network context flags.
- typedef **_nx_sock_stat nx_sock_stat_t**
  Accounting data.

Functions

- int **nx_create** (**nid_t** nid, **nx_flags_t** *data)
  Create network context.
- int **nx_migrate** (**nid_t** nid)
  Migrate to an existing network context.
- int **nx_task_nid** (pid_t pid)
  Get the network context ID of a process.
- int **nx_info** (**nid_t** nid, **nx_info_t** *data)
  Get network context information.
- int **nx_addr_add** (**nid_t** nid, **nx_addr_t** *data)
  Add network context addresses.
- int **nx_addr_remove** (**nid_t** nid, **nx_addr_t** *data)
  Remove network context addresses.
- int **nx_flags_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context flags.
- int **nx_flags_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context flags.
- int **nx_caps_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context capabilities.
- int **nx_caps_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context capabilities.
- int **nx_sock_stat** (**nid_t** nid, **nx_sock_stat_t** *data)
  Get network socket accounting data.

4.7.1 Define Documentation

4.7.1.1 #define NXF_INFO_PRIVATE 0x00000008

Network context cannot be entered

Definition at line 669 of file vserver.h.

4.7.1.2 #define NXF_STATE_SETUP (1ULL<<32)

Network context is in setup state

Definition at line 670 of file vserver.h.

4.7.1.3 #define NXF_STATE_ADMIN (1ULL<<34)

Context is in admin state

Definition at line 671 of file vserver.h.

4.7.1.4 #define NXF_SC_HELPER (1ULL<<36)

Network state change helper

Definition at line 672 of file vserver.h.

4.7.1.5 #define NXF_PERSISTENT (1ULL<<38)

Make network context persistent

Definition at line 673 of file vserver.h.

4.7.1.6 #define NXA_TYPE_IPV4 1

Address is IPv4

Definition at line 675 of file vserver.h.

4.7.1.7 #define NXA_TYPE_IPV6 2

Address is IPv6

Definition at line 676 of file vserver.h.

4.7.1.8 #define NXA_MOD_BCAST (1<<8)

Address is Broadcast

Definition at line 677 of file vserver.h.

4.7.1.9 #define NXA_TYPE_ANY ((uint16_t)-1)

Matches any address

Definition at line 678 of file vserver.h.

4.7.1.10 #define VXA_SOCK_UNSPEC 0

Definition at line 682 of file vserver.h.

4.7.1.11 #define VXA_SOCK_UNIX 1

Definition at line 683 of file vserver.h.

4.7.1.12 #define VXA_SOCK_INET 2

Definition at line 684 of file vserver.h.

4.7.1.13 #define VXA_SOCK_INET6 3

Definition at line 685 of file vserver.h.

4.7.1.14 #define VXA_SOCK_PACKET 4

Definition at line 686 of file vserver.h.

4.7.1.15 #define VXA_SOCK_OTHER 5

Definition at line 687 of file vserver.h.

4.7.1.16 #define NXA_SOCK_UNSPEC VXA_SOCK_UNSPEC

Definition at line 690 of file vserver.h.

4.7.1.17 #define NXA_SOCK_UNIX VXA_SOCK_UNIX

Definition at line 691 of file vserver.h.

4.7.1.18 #define NXA_SOCK_INET VXA_SOCK_INET

Definition at line 692 of file vserver.h.

4.7.1.19 #define NXA_SOCK_INET6 VXA_SOCK_INET6

Definition at line 693 of file vserver.h.

4.7.1.20 #define NXA_SOCK_PACKET VXA_SOCK_PACKET

Definition at line 694 of file vserver.h.

4.7.1.21 #define NXA_SOCK_OTHER VXA_SOCK_OTHER

Definition at line 695 of file vserver.h.

4.7.2 Typedef Documentation

4.7.2.1 typedef uint32_t nid_t

Network context ID type

Definition at line 697 of file vserver.h.

4.7.2.2 typedef struct _nx_info nx_info_t

Network context information.

4.7.2.3 typedef struct _nx_addr nx_addr_t

Network address information.

4.7.2.4 typedef struct _nx_flags nx_flags_t

Network context flags.

4.7.2.5 typedef struct _nx_sock_stat nx_sock_stat_t

Accounting data.

4.7.3 Function Documentation

4.7.3.1 int nx_create (nid_t *nid*, nx_flags_t * *data*)

Create network context.

Parameters:

nid Network context ID

data Initial network context flags

Definition at line 60 of file network.c.

References _nx_flags::flags, and vserver().

```
61 {
62         struct vcmd_net_create kdata = {
63                 .flagword = 0,
64         };
65
66         if (data)
67                 kdata.flagword = data->flags;
68
69         return vserver(VCMD_net_create, nid, &kdata);
70 }
```

4.7.3.2 int nx_migrate (nid_t *nid*)

Migrate to an existing network context.

Parameters:

nid Network context ID

Definition at line 72 of file network.c.

References vserver().

```
73 {
74         return vserver(VCMD_net_migrate, nid, NULL);
75 }
```

4.7.3.3 int nx_task_nid (pid_t *pid*)

Get the network context ID of a process.

Parameters:

pid Process ID

Returns:

Network context ID

Definition at line 77 of file network.c.

References vserver().

```
78 {
79         return vserver(VCMD_task_nid, pid, NULL);
80 }
```

4.7.3.4 int nx_info (nid_t *nid*, nx_info_t * *data*)

Get network context information.

Parameters:

nid Network context ID

data Empty nx_info_t struct to be filled

Definition at line 82 of file network.c.

References _nx_info::nid, and vserver().

```
83 {
84         struct vcmd_nx_info_v0 kdata;
85
86         int rc = vserver(VCMD_nx_info, nid, &kdata);
87
88         if (rc == -1)
```

```
89                 return rc;
90
91         if (data)
92                 data->nid = kdata.nid;
93
94         return rc;
95 }
```

4.7.3.5 int nx_addr_add (nid_t *nid*, nx_addr_t * *data*)

Add network context addresses.

Parameters:

nid Network context ID

data Network address information

Definition at line 97 of file network.c.

References _nx_addr::count, _nx_addr::ip, _nx_addr::mask, _nx_addr::type, and vserver().

```
98 {
99         struct vcmd_net_addr_v0 kdata;
100
101         if (!data)
102                 return errno = EINVAL, -1;
103
104         kdata.type  = data->type;
105         kdata.count = data->count;
106
107         str_cpyn(kdata.ip,   data->ip,   sizeof(kdata.ip));
108         str_cpyn(kdata.mask, data->mask, sizeof(kdata.mask));
109
110         return vserver(VCMD_net_add, nid, &kdata);
111 }
```

4.7.3.6 int nx_addr_remove (nid_t *nid*, nx_addr_t * *data*)

Remove network context addresses.

Parameters:

nid Network context ID

data Network address information

Definition at line 113 of file network.c.

References _nx_addr::count, _nx_addr::ip, _nx_addr::mask, _nx_addr::type, and vserver().

```
114 {
115         struct vcmd_net_addr_v0 kdata;
116
117         if (!data)
118                 return errno = EINVAL, -1;
119
120         kdata.type  = data->type;
121         kdata.count = data->count;
```

```
122
123         str_cpyn(kdata.ip,   data->ip,   sizeof(kdata.ip));
124         str_cpyn(kdata.mask, data->mask, sizeof(kdata.mask));
125
126         return vserver(VCMD_net_remove, nid, &kdata);
127 }
```

4.7.3.7 int nx_flags_set (nid_t *nid*, nx_flags_t * *data*)

Set network context flags.

Parameters:

nid Network context ID

data Network context flags

Definition at line 129 of file network.c.

References _nx_flags::flags, _nx_flags::mask, and vserver().

```
130 {
131         struct vcmd_net_flags_v0 kdata;
132
133         if (!data)
134                 return errno = EINVAL, -1;
135
136         kdata.flagword = data->flags;
137         kdata.mask     = data->mask;
138
139         return vserver(VCMD_set_nflags, nid, &kdata);
140 }
```

4.7.3.8 int nx_flags_get (nid_t *nid*, nx_flags_t * *data*)

Get network context flags.

Parameters:

nid Network context ID

data Empty nx_flags_t struct to be filled

Definition at line 142 of file network.c.

References _nx_flags::flags, _nx_flags::mask, and vserver().

```
143 {
144         struct vcmd_net_flags_v0 kdata;
145
146         if (!data)
147                 return errno = EINVAL, -1;
148
149         int rc = vserver(VCMD_get_nflags, nid, &kdata);
150
151         if (rc == -1)
152                 return rc;
153
154         data->flags = kdata.flagword;
```

```
155         data->mask  = kdata.mask;
156
157         return rc;
158 }
```

4.7.3.9 int nx_caps_set (nid_t *nid*, nx_flags_t * *data*)

Set network context capabilities.

Parameters:

nid Network context ID

data Network context capabilities

Definition at line 160 of file network.c.

References _nx_flags::flags, _nx_flags::mask, and vserver().

```
161 {
162         struct vcmd_net_caps_v0 kdata;
163
164         if (!data)
165                 return errno = EINVAL, -1;
166
167         kdata.ncaps = data->flags;
168         kdata.cmask = data->mask;
169
170         return vserver(VCMD_set_ncaps, nid, &kdata);
171 }
```

4.7.3.10 int nx_caps_get (nid_t *nid*, nx_flags_t * *data*)

Get network context capabilities.

Parameters:

nid Network context ID

data Empty nx_flags_t struct to be filled

Definition at line 173 of file network.c.

References _nx_flags::flags, _nx_flags::mask, and vserver().

```
174 {
175         struct vcmd_net_caps_v0 kdata;
176
177         if (!data)
178                 return errno = EINVAL, -1;
179
180         int rc = vserver(VCMD_get_ncaps, nid, &kdata);
181
182         if (rc == -1)
183                 return rc;
184
185         data->flags = kdata.ncaps;
186         data->mask  = kdata.cmask;
187
188         return rc;
189 }
```

4.7.3.11 int nx_sock_stat (nid_t *nid*, nx_sock_stat_t * *data*)

Get network socket accounting data.

Parameters:

nid Network context ID

data Empty nx_sock_stat_t struct to be filled

Definition at line 191 of file network.c.

References _nx_sock_stat::count, _nx_sock_stat::id, _nx_sock_stat::total, and vserver().

```
192 {
193         struct vcmd_sock_stat_v0 kdata;
194 
195         if (!data)
196                 return errno = EINVAL, -1;
197 
198         kdata.field = data->id;
199 
200         int rc = vserver(VCMD_sock_stat, nid, &kdata);
201 
202         if (rc == -1)
203                 return rc;
204 
205         data->count[0] = kdata.count[0];
206         data->count[1] = kdata.count[1];
207         data->count[2] = kdata.count[2];
208 
209         data->total[0] = kdata.total[0];
210         data->total[1] = kdata.total[1];
211         data->total[2] = kdata.total[2];
212 
213         return rc;
214 }
```

Chapter 5

libvserver Data Structure Documentation

5.1 _dx_limit Struct Reference

`#include <vserver.h>`

5.1.1 Detailed Description

Disk limit values.

Definition at line 525 of file vserver.h.

Data Fields

- const char * **filename**
- uint32_t **space_used**
- uint32_t **space_total**
- uint32_t **inodes_used**
- uint32_t **inodes_total**
- uint32_t **reserved**
- uint32_t **flags**

5.1.2 Field Documentation

5.1.2.1 const char* _dx_limit::filename

Mount point

Definition at line 526 of file vserver.h.

Referenced by dx_limit_add(), dx_limit_get(), dx_limit_remove(), and dx_limit_set().

5.1.2.2 uint32_t _dx_limit::space_used

Currently used space

Definition at line 527 of file vserver.h.

Referenced by dx_limit_get(), and dx_limit_set().

5.1.2.3 uint32_t _dx_limit::space_total

Total space

Definition at line 528 of file vserver.h.

Referenced by dx_limit_get(), and dx_limit_set().

5.1.2.4 uint32_t _dx_limit::inodes_used

Currently used inodes

Definition at line 529 of file vserver.h.

Referenced by dx_limit_get(), and dx_limit_set().

5.1.2.5 uint32_t _dx_limit::inodes_total

Total inodes

Definition at line 530 of file vserver.h.

Referenced by dx_limit_get(), and dx_limit_set().

5.1.2.6 uint32_t _dx_limit::reserved

Space reserved for the root user

Definition at line 531 of file vserver.h.

Referenced by dx_limit_get(), and dx_limit_set().

5.1.2.7 uint32_t _dx_limit::flags

Disk limit flags

Definition at line 532 of file vserver.h.

Referenced by dx_limit_add(), dx_limit_get(), and dx_limit_set().

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

- **vserver.h**

5.2 _ix_attr Struct Reference

`#include <vserver.h>`

5.2.1 Detailed Description

Inode attributes.

Definition at line 588 of file vserver.h.

Data Fields

- const char * **filename**
- **xid_t xid**
- uint32_t **flags**
- uint32_t **mask**

5.2.2 Field Documentation

5.2.2.1 const char* _ix_attr::filename

Filename

Definition at line 589 of file vserver.h.

Referenced by ix_attr_get(), and ix_attr_set().

5.2.2.2 xid_t _ix_attr::xid

Context ID

Definition at line 590 of file vserver.h.

Referenced by ix_attr_get(), and ix_attr_set().

5.2.2.3 uint32_t _ix_attr::flags

Inode flags

Definition at line 591 of file vserver.h.

Referenced by ix_attr_get(), and ix_attr_set().

5.2.2.4 uint32_t _ix_attr::mask

Set mask

Definition at line 592 of file vserver.h.

Referenced by ix_attr_get(), and ix_attr_set().

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

- **vserver.h**

5.3 _nx_addr Struct Reference

`#include <vserver.h>`

5.3.1 Detailed Description

Network address information.

Definition at line 709 of file vserver.h.

Data Fields

- uint16_t **type**
- uint16_t **count**
- uint32_t **ip** [4]
- uint32_t **mask** [4]

5.3.2 Field Documentation

5.3.2.1 uint16_t _nx_addr::type

Address type

Definition at line 710 of file vserver.h.

Referenced by nx_addr_add(), and nx_addr_remove().

5.3.2.2 uint16_t _nx_addr::count

Number of addresses in ip/mask

Definition at line 711 of file vserver.h.

Referenced by nx_addr_add(), and nx_addr_remove().

5.3.2.3 uint32_t _nx_addr::ip[4]

Up to four adresses

Definition at line 712 of file vserver.h.

Referenced by nx_addr_add(), and nx_addr_remove().

5.3.2.4 uint32_t _nx_addr::mask[4]

Up to four netmasks

Definition at line 713 of file vserver.h.

Referenced by nx_addr_add(), and nx_addr_remove().

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

- **vserver.h**

5.4 _nx_flags Struct Reference

`#include <vserver.h>`

5.4.1 Detailed Description

Network context flags.

Definition at line 719 of file vserver.h.

Data Fields

- uint64_t **flags**
- uint64_t **mask**

5.4.2 Field Documentation

5.4.2.1 uint64_t _nx_flags::flags

Network context flags

Definition at line 720 of file vserver.h.

Referenced by nx_caps_get(), nx_caps_set(), nx_create(), nx_flags_get(), and nx_flags_set().

5.4.2.2 uint64_t _nx_flags::mask

Set mask

Definition at line 721 of file vserver.h.

Referenced by nx_caps_get(), nx_caps_set(), nx_flags_get(), and nx_flags_set().

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

- **vserver.h**

5.5 _nx_info Struct Reference

`#include <vserver.h>`

5.5.1 Detailed Description

Network context information.

Definition at line 702 of file vserver.h.

Data Fields

- **nid_t nid**

5.5.2 Field Documentation

5.5.2.1 nid_t _nx_info::nid

Network context ID

Definition at line 703 of file vserver.h.

Referenced by nx_info().

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

- **vserver.h**

5.6 _nx_sock_stat Struct Reference

`#include <vserver.h>`

5.6.1 Detailed Description

Accounting data.

Definition at line 727 of file vserver.h.

Data Fields

- uint32_t **id**
- uint32_t **count** [3]
- uint64_t **total** [3]

5.6.2 Field Documentation

5.6.2.1 uint32_t _nx_sock_stat::id

Socket type ID

Definition at line 728 of file vserver.h.

Referenced by nx_sock_stat().

5.6.2.2 uint32_t _nx_sock_stat::count[3]

Number of packets

Definition at line 729 of file vserver.h.

Referenced by nx_sock_stat().

5.6.2.3 uint64_t _nx_sock_stat::total[3]

Number of bytes

Definition at line 730 of file vserver.h.

Referenced by nx_sock_stat().

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

- **vserver.h**

5.7 _vx_flags Struct Reference

`#include <vserver.h>`

5.7.1 Detailed Description

Context/migration flags.

Definition at line 202 of file vserver.h.

Data Fields

- uint64_t **flags**
- uint64_t **mask**

5.7.2 Field Documentation

5.7.2.1 uint64_t _vx_flags::flags

Flags

Definition at line 203 of file vserver.h.

Referenced by vx_bcaps_get(), vx_bcaps_set(), vx_ccaps_get(), vx_ccaps_set(), vx_create(), vx_flags_get(), vx_flags_set(), and vx_migrate().

5.7.2.2 uint64_t _vx_flags::mask

Set mask

Definition at line 204 of file vserver.h.

Referenced by vx_bcaps_get(), vx_bcaps_set(), vx_ccaps_get(), vx_ccaps_set(), vx_flags_get(), and vx_flags_set().

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

- **vserver.h**

5.8 _vx_info Struct Reference

`#include <vserver.h>`

5.8.1 Detailed Description

Context information.

Definition at line 178 of file vserver.h.

Data Fields

- **xid_t xid**
- pid_t **initpid**

5.8.2 Field Documentation

5.8.2.1 xid_t _vx_info::xid

Context ID

Definition at line 179 of file vserver.h.

Referenced by vx_info().

5.8.2.2 pid_t _vx_info::initpid

Process ID of init

Definition at line 180 of file vserver.h.

Referenced by vx_info().

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

- **vserver.h**

5.9 _vx_limit Struct Reference

`#include <vserver.h>`

5.9.1 Detailed Description

Resource limits.

Definition at line 453 of file vserver.h.

Data Fields

- uint32_t **id**
- uint64_t **minimum**
- uint64_t **softlimit**
- uint64_t **maximum**

5.9.2 Field Documentation

5.9.2.1 uint32_t _vx_limit::id

Limit ID

Definition at line 454 of file vserver.h.

Referenced by vx_limit_get(), and vx_limit_set().

5.9.2.2 uint64_t _vx_limit::minimum

Minimum

Definition at line 455 of file vserver.h.

Referenced by vx_limit_get(), vx_limit_mask_get(), and vx_limit_set().

5.9.2.3 uint64_t _vx_limit::softlimit

Softlimit

Definition at line 456 of file vserver.h.

Referenced by vx_limit_get(), vx_limit_mask_get(), and vx_limit_set().

5.9.2.4 uint64_t _vx_limit::maximum

Maximum

Definition at line 457 of file vserver.h.

Referenced by vx_limit_get(), vx_limit_mask_get(), and vx_limit_set().

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

- **vserver.h**

5.10 _vx_limit_stat Struct Reference

`#include <vserver.h>`

5.10.1 Detailed Description

Resource limit accounting.

Definition at line 463 of file vserver.h.

Data Fields

- uint32_t **id**
- uint32_t **hits**
- uint64_t **value**
- uint64_t **minimum**
- uint64_t **maximum**

5.10.2 Field Documentation

5.10.2.1 uint32_t _vx_limit_stat::id

Limit ID

Definition at line 464 of file vserver.h.

Referenced by vx_limit_stat().

5.10.2.2 uint32_t _vx_limit_stat::hits

Number of hits

Definition at line 465 of file vserver.h.

Referenced by vx_limit_stat().

5.10.2.3 uint64_t _vx_limit_stat::value

Current value

Definition at line 466 of file vserver.h.

Referenced by vx_limit_stat().

5.10.2.4 uint64_t _vx_limit_stat::minimum

Minimum value

Definition at line 467 of file vserver.h.

Referenced by vx_limit_stat().

5.10.2.5 uint64_t _vx_limit_stat::maximum

Maximum value

Definition at line 468 of file vserver.h.

Referenced by vx_limit_stat().

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

- **vserver.h**

5.11 _vx_sched Struct Reference

`#include <vserver.h>`

5.11.1 Detailed Description

Scheduler values.

Definition at line 375 of file vserver.h.

Data Fields

- uint32_t **mask**
- int32_t **cpu_id**
- int32_t **bucket_id**
- int32_t **fill_rate** [2]
- int32_t **interval** [2]
- int32_t **tokens**
- int32_t **tokens_min**
- int32_t **tokens_max**
- int32_t **prio_bias**

5.11.2 Field Documentation

5.11.2.1 uint32_t _vx_sched::mask

Set mask

Definition at line 376 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.2 int32_t _vx_sched::cpu_id

CPU ID (for SMP machines)

Definition at line 377 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.3 int32_t _vx_sched::bucket_id

Token Bucket ID

Definition at line 378 of file vserver.h.

Referenced by vx_sched_get().

5.11.2.4 int32_t _vx_sched::fill_rate[2]

Fill rate

Definition at line 379 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.5 int32_t _vx_sched::interval[2]

Interval between fills

Definition at line 380 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.6 int32_t _vx_sched::tokens

Number of tokens in the bucket

Definition at line 381 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.7 int32_t _vx_sched::tokens_min

Minimum tokens to unhold the context

Definition at line 382 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.8 int32_t _vx_sched::tokens_max

Maximum number of tokens in the bucket

Definition at line 383 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

5.11.2.9 int32_t _vx_sched::prio_bias

Priority bias

Definition at line 384 of file vserver.h.

Referenced by vx_sched_get(), and vx_sched_set().

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

- **vserver.h**

5.12 _vx_sched_info Struct Reference

`#include <vserver.h>`

5.12.1 Detailed Description

Scheduler information.

Definition at line 390 of file vserver.h.

Data Fields

- int32_t **cpu_id**
- int32_t **bucket_id**
- uint64_t **user_msec**
- uint64_t **sys_msec**
- uint64_t **hold_msec**
- uint32_t **token_usec**
- int32_t **vavavoom**

5.12.2 Field Documentation

5.12.2.1 int32_t _vx_sched_info::cpu_id

Definition at line 391 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.2 int32_t _vx_sched_info::bucket_id

Definition at line 392 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.3 uint64_t _vx_sched_info::user_msec

Definition at line 393 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.4 uint64_t _vx_sched_info::sys_msec

Definition at line 394 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.5 uint64_t _vx_sched_info::hold_msec

Definition at line 395 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.6 uint32_t _vx_sched_info::token_usec

Definition at line 396 of file vserver.h.

Referenced by vx_sched_info().

5.12.2.7 int32_t _vx_sched_info::vavavoom

Definition at line 397 of file vserver.h.

Referenced by vx_sched_info().

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

- **vserver.h**

5.13 _vx_stat Struct Reference

`#include <vserver.h>`

5.13.1 Detailed Description

Context statistics.

Definition at line 186 of file vserver.h.

Data Fields

- uint32_t **usecnt**
- uint32_t **tasks**
- uint32_t **nr_threads**
- uint32_t **nr_running**
- uint32_t **nr_unintr**
- uint32_t **nr_onhold**
- uint32_t **nr_forks**
- uint32_t **load** [3]
- uint64_t **offset**
- uint64_t **uptime**

5.13.2 Field Documentation

5.13.2.1 uint32_t _vx_stat::usecnt

Number of context references

Definition at line 187 of file vserver.h.

Referenced by vx_stat().

5.13.2.2 uint32_t _vx_stat::tasks

Number of tasks

Definition at line 188 of file vserver.h.

Referenced by vx_stat().

5.13.2.3 uint32_t _vx_stat::nr_threads

Total number of threads

Definition at line 189 of file vserver.h.

Referenced by vx_stat().

5.13.2.4 uint32_t _vx_stat::nr_running

Number of running threads

Definition at line 190 of file vserver.h.

Referenced by vx_stat().

5.13.2.5 uint32_t _vx_stat::nr_unintr

Number of uninterruptible threads

Definition at line 191 of file vserver.h.

Referenced by vx_stat().

5.13.2.6 uint32_t _vx_stat::nr_onhold

Number of threads being held

Definition at line 192 of file vserver.h.

Referenced by vx_stat().

5.13.2.7 uint32_t _vx_stat::nr_forks

Total number of forks since context startup

Definition at line 193 of file vserver.h.

Referenced by vx_stat().

5.13.2.8 uint32_t _vx_stat::load[3]

Load average

Definition at line 194 of file vserver.h.

Referenced by vx_stat().

5.13.2.9 uint64_t _vx_stat::offset

Offset to the system time

Definition at line 195 of file vserver.h.

Referenced by vx_stat().

5.13.2.10 uint64_t _vx_stat::uptime

Context uptime

Definition at line 196 of file vserver.h.

Referenced by vx_stat().

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

- vserver.h

5.14 _vx_uname Struct Reference

`#include <vserver.h>`

5.14.1 Detailed Description

Virtual system information data.

Definition at line 210 of file vserver.h.

Data Fields

- uint32_t **id**
- char **value** [65]

5.14.2 Field Documentation

5.14.2.1 uint32_t _vx_uname::id

Name ID

Definition at line 211 of file vserver.h.

Referenced by vx_uname_get(), and vx_uname_set().

5.14.2.2 char _vx_uname::value[65]

Name value

Definition at line 212 of file vserver.h.

Referenced by vx_uname_get(), and vx_uname_set().

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

- **vserver.h**

5.15 _vx_wait Struct Reference

`#include <vserver.h>`

5.15.1 Detailed Description

Wait results.

Definition at line 218 of file vserver.h.

Data Fields

- int32_t **reboot_cmd**
- int32_t **exit_code**

5.15.2 Field Documentation

5.15.2.1 int32_t _vx_wait::reboot_cmd

Context reboot command

Definition at line 219 of file vserver.h.

Referenced by vx_wait().

5.15.2.2 int32_t _vx_wait::exit_code

Context exit code

Definition at line 220 of file vserver.h.

Referenced by vx_wait().

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

- **vserver.h**

Chapter 6

libvserver File Documentation

6.1 context.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/context_cmd.h"
#include "linux/vserver/cvirt_cmd.h"
#include "linux/vserver/signal_cmd.h"
#include "vserver.h"
```

Include dependency graph for context.c:

Functions

- int **vx_create** (**xid_t** xid, **vx_flags_t** *data)
  Create a new context.
- int **vx_migrate** (**xid_t** xid, **vx_flags_t** *data)
  Migrate to an existing context.
- int **vx_task_xid** (pid_t pid)
  Get the context ID of a process.
- int **vx_info** (**xid_t** xid, **vx_info_t** *data)
  Get context information.
- int **vx_stat** (**xid_t** xid, **vx_stat_t** *data)
  Get context statistics.
- int **vx_bcaps_set** (**xid_t** xid, **vx_flags_t** *data)

Set system capabilities.

- int **vx_bcaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get system capabilities.
- int **vx_ccaps_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context capabilities.
- int **vx_ccaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context capabilities.
- int **vx_flags_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context flags.
- int **vx_flags_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context flags.
- int **vx_uname_set** (**xid_t** xid, **vx_uname_t** *data)
  Set virtual system information.
- int **vx_uname_get** (**xid_t** xid, **vx_uname_t** *data)
  Get virtual system information.
- int **vx_kill** (**xid_t** xid, pid_t pid, int sig)
  Kill one or more processes.
- int **vx_wait** (**xid_t** xid, **vx_wait_t** *data)
  Wait for context death.

6.2 dlimit.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/dlimit_cmd.h"
#include "vserver.h"
```

Include dependency graph for dlimit.c:

Functions

- int **dx_limit_add** (**xid_t** xid, **dx_limit_t** *data)
  Add disk limit entry.
- int **dx_limit_remove** (**xid_t** xid, **dx_limit_t** *data)
  Remove disk limit.
- int **dx_limit_set** (**xid_t** xid, **dx_limit_t** *data)
  Set disk limit values.
- int **dx_limit_get** (**xid_t** xid, **dx_limit_t** *data)
  Get disk limit values.

6.3 inode.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/inode_cmd.h"
#include "vserver.h"
```

Include dependency graph for inode.c:

Functions

- int **ix_attr_set** (**ix_attr_t** *data)
  Set inode attributes.
- int **ix_attr_get** (**ix_attr_t** *data)
  Get inode attributes.

6.4 limit.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/limit_cmd.h"
#include "vserver.h"
```

Include dependency graph for limit.c:

Functions

- int **vx_limit_mask_get** (**vx_limit_t** *data)
  Get resource limits mask.
- int **vx_limit_set** (**xid_t** xid, **vx_limit_t** *data)
  Set resource limit.
- int **vx_limit_get** (**xid_t** xid, **vx_limit_t** *data)
  Get resource limits.
- int **vx_limit_stat** (**xid_t** xid, **vx_limit_stat_t** *data)
  Get resource limit accounting data.
- int **vx_limit_reset** (**xid_t** xid)
  Reset resource limit accounting data.

6.5 namespace.c File Reference

```
#include <stdint.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/space_cmd.h"
#include "vserver.h"
```

Include dependency graph for namespace.c:

Functions

- int **ns_clone** (int flags, void *child_stack)
  Clone the current namespace (FS/IPC/UTS).
- int **ns_enter** (**xid_t** xid, uint64_t mask)
  Enter namespace.
- int **ns_set** (**xid_t** xid, uint64_t mask)
  Set namespace.

6.6 network.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/cacct_cmd.h"
#include "linux/vserver/network_cmd.h"
#include "vserver.h"
```

Include dependency graph for network.c:

Functions

- int **nx_create** (**nid_t** nid, **nx_flags_t** *data)
  Create network context.
- int **nx_migrate** (**nid_t** nid)
  Migrate to an existing network context.
- int **nx_task_nid** (pid_t pid)
  Get the network context ID of a process.
- int **nx_info** (**nid_t** nid, **nx_info_t** *data)
  Get network context information.
- int **nx_addr_add** (**nid_t** nid, **nx_addr_t** *data)
  Add network context addresses.
- int **nx_addr_remove** (**nid_t** nid, **nx_addr_t** *data)
  Remove network context addresses.
- int **nx_flags_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context flags.
- int **nx_flags_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context flags.
- int **nx_caps_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context capabilities.
- int **nx_caps_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context capabilities.
- int **nx_sock_stat** (**nid_t** nid, **nx_sock_stat_t** *data)
  Get network socket accounting data.

6.7 sched.c File Reference

```
#include <errno.h>
#include "vserver.h"
#include "linux/vserver/switch.h"
#include "linux/vserver/sched_cmd.h"
```

Include dependency graph for sched.c:

Functions

- int **vx_sched_get** (**xid_t** xid, **vx_sched_t** *data)
  Get scheduler values.
- int **vx_sched_set** (**xid_t** xid, **vx_sched_t** *data)
  Set scheduler values.
- int **vx_sched_info** (**xid_t** xid, **vx_sched_info_t** *data)
  Get scheduler information.

6.8 switch.c File Reference

```
#include <stdint.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/debug_cmd.h"
#include "vserver.h"
```

Include dependency graph for switch.c:

Functions

- int **vs_get_version** (void)
  Get vserver version of running kernel.
- int **vs_get_config** (void)
  Get vserver configuration of running kernel.

6.8.1 Function Documentation

6.8.1.1 int vs_get_version (void)

Get vserver version of running kernel.

Returns:

Kernel version

Definition at line 25 of file switch.c.

References vserver().

```
26 {
27         return vserver(VCMD_get_version, 0, NULL);
28 }
```

6.8.1.2 int vs_get_config (void)

Get vserver configuration of running kernel.

Returns:

Kernel configuration

Definition at line 30 of file switch.c.

References vserver().

```
31 {
32         return vserver(VCMD_get_vci, 0, NULL);
33 }
```

6.9 syscall.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "syscall.h"
#include "vserver.h"
```

Include dependency graph for syscall.c:

6.10 vserver.h File Reference

6.10.1 Detailed Description

Interface to the vserver syscalls.

Definition in file **vserver.h**.

```
#include <sys/types.h>
```

```
#include <stdint.h>
```

Include dependency graph for vserver.h:

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

Data Structures

- struct **_vx_info**
  Context information.
- struct **_vx_stat**
  Context statistics.
- struct **_vx_flags**
  Context/migration flags.
- struct **_vx_uname**
  Virtual system information data.
- struct **_vx_wait**
  Wait results.
- struct **_vx_sched**
  Scheduler values.
- struct **_vx_sched_info**
  Scheduler information.
- struct **_vx_limit**
  Resource limits.
- struct **_vx_limit_stat**
  Resource limit accounting.
- struct **_dx_limit**
  Disk limit values.

- struct **_ix_attr**
  Inode attributes.
- struct **_nx_info**
  Network context information.
- struct **_nx_addr**
  Network address information.
- struct **_nx_flags**
  Network context flags.
- struct **_nx_sock_stat**
  Accounting data.

Defines

- #define **LIBVSERVER_API_MAJOR** 2
- #define **LIBVSERVER_API_MINOR** 0
- #define **CAP_CHOWN** 0
- #define **CAP_DAC_OVERRIDE** 1
- #define **CAP_DAC_READ_SEARCH** 2
- #define **CAP_FOWNER** 3
- #define **CAP_FSETID** 4
- #define **CAP_KILL** 5
- #define **CAP_SETGID** 6
- #define **CAP_SETUID** 7
- #define **CAP_SETPCAP** 8
- #define **CAP_LINUX_IMMUTABLE** 9
- #define **CAP_NET_BIND_SERVICE** 10
- #define **CAP_NET_BROADCAST** 11
- #define **CAP_NET_ADMIN** 12
- #define **CAP_NET_RAW** 13
- #define **CAP_IPC_LOCK** 14
- #define **CAP_IPC_OWNER** 15
- #define **CAP_SYS_MODULE** 16
- #define **CAP_SYS_RAWIO** 17
- #define **CAP_SYS_CHROOT** 18
- #define **CAP_SYS_PTRACE** 19
- #define **CAP_SYS_PACCT** 20
- #define **CAP_SYS_ADMIN** 21
- #define **CAP_SYS_BOOT** 22
- #define **CAP_SYS_NICE** 23
- #define **CAP_SYS_RESOURCE** 24
- #define **CAP_SYS_TIME** 25
- #define **CAP_SYS_TTY_CONFIG** 26
- #define **CAP_MKNOD** 27
- #define **CAP_LEASE** 28
- #define **CAP_AUDIT_WRITE** 29

- #define **CAP_AUDIT_CONTROL** 30
- #define **CAP_CONTEXT** 31
- #define **VXC_SET_UTSNAME** 0x00000001
- #define **VXC_SET_RLIMIT** 0x00000002
- #define **VXC_RAW_ICMP** 0x00000100
- #define **VXC_SYSLOG** 0x00001000
- #define **VXC_SECURE_MOUNT** 0x00010000
- #define **VXC_SECURE_REMOUNT** 0x00020000
- #define **VXC_BINARY_MOUNT** 0x00040000
- #define **VXC_QUOTA_CTL** 0x00100000
- #define **VXC_ADMIN_MAPPER** 0x00200000
- #define **VXC_ADMIN_CLOOP** 0x00400000
- #define **VXF_INFO_SCHED** 0x00000002
- #define **VXF_INFO_NPROC** 0x00000004
- #define **VXF_INFO_PRIVATE** 0x00000008
- #define **VXF_INFO_INIT** 0x00000010
- #define **VXF_INFO_HIDE** 0x00000020
- #define **VXF_INFO_ULIMIT** 0x00000040
- #define **VXF_INFO_NSPACE** 0x00000080
- #define **VXF_SCHED_HARD** 0x00000100
- #define **VXF_SCHED_PRIO** 0x00000200
- #define **VXF_SCHED_PAUSE** 0x00000400
- #define **VXF_VIRT_MEM** 0x00010000
- #define **VXF_VIRT_UPTIME** 0x00020000
- #define **VXF_VIRT_CPU** 0x00040000
- #define **VXF_VIRT_LOAD** 0x00080000
- #define **VXF_VIRT_TIME** 0x00100000
- #define **VXF_HIDE_MOUNT** 0x01000000
- #define **VXF_HIDE_NETIF** 0x02000000
- #define **VXF_HIDE_VINFO** 0x04000000
- #define **VXF_STATE_SETUP** (1ULL<<32)
- #define **VXF_STATE_INIT** (1ULL<<33)
- #define **VXF_STATE_ADMIN** (1ULL<<34)
- #define **VXF_SC_HELPER** (1ULL<<36)
- #define **VXF_REBOOT_KILL** (1ULL<<37)
- #define **VXF_PERSISTENT** (1ULL<<38)
- #define **VXF_FORK_RSS** (1ULL<<48)
- #define **VXF_PROLIFIC** (1ULL<<49)
- #define **VXF_IGNEG_NICE** (1ULL<<52)
- #define **VXM_SET_INIT** 0x00000001
- #define **VXM_SET_REAPER** 0x00000002
- #define **VHIN_CONTEXT** 0
- #define **VHIN_SYSNAME** 1
- #define **VHIN_NODENAME** 2
- #define **VHIN_RELEASE** 3
- #define **VHIN_VERSION** 4
- #define **VHIN_MACHINE** 5
- #define **VHIN_DOMAINNAME** 6
- #define **VXSM_FILL_RATE** 0x0001
- #define **VXSM_INTERVAL** 0x0002

- #define **VXSM_FILL_RATE2** 0x0004
- #define **VXSM_INTERVAL2** 0x0008
- #define **VXSM_TOKENS** 0x0010
- #define **VXSM_TOKENS_MIN** 0x0020
- #define **VXSM_TOKENS_MAX** 0x0040
- #define **VXSM_PRIO_BIAS** 0x0100
- #define **VXSM_IDLE_TIME** 0x0200
- #define **VXSM_FORCE** 0x0400
- #define **VXSM_CPU_ID** 0x1000
- #define **VXSM_BUCKET_ID** 0x2000
- #define **VXSM_MSEC** 0x4000
- #define **VXSM_V3_MASK** 0x0173
- #define **VXSM_SET_MASK** 0x01FF
- #define **CRLIM_UNSET** (0ULL)
- #define **CRLIM_INFINITY** (~0ULL)
- #define **CRLIM_KEEP** (~1ULL)
- #define **VLIMIT_NSOCK** 16
- #define **VLIMIT_OPENFD** 17
- #define **VLIMIT_ANON** 18
- #define **VLIMIT_SHMEM** 19
- #define **VLIMIT_SEMARY** 20
- #define **VLIMIT_NSEMS** 21
- #define **VLIMIT_DENTRY** 22
- #define **VLIMIT_MAPPED** 23
- #define **CDLIM_UNSET** ((uint32_t)0UL)
- #define **CDLIM_INFINITY** ((uint32_t)~0UL)
- #define **CDLIM_KEEP** ((uint32_t)~1UL)
- #define **IATTR_TAG** 0x01000000
- #define **IATTR_ADMIN** 0x00000001
- #define **IATTR_WATCH** 0x00000002
- #define **IATTR_HIDE** 0x00000004
- #define **IATTR_FLAGS** 0x00000007
- #define **IATTR_BARRIER** 0x00010000
- #define **IATTR_IUNLINK** 0x00020000
- #define **IATTR_IMMUTABLE** 0x00040000
- #define **CLONE_VM** 0x00000100
- #define **CLONE_FS** 0x00000200
- #define **CLONE_FILES** 0x00000400
- #define **CLONE_SIGHAND** 0x00000800
- #define **CLONE_PTRACE** 0x00002000
- #define **CLONE_VFORK** 0x00004000
- #define **CLONE_PARENT** 0x00008000
- #define **CLONE_THREAD** 0x00010000
- #define **CLONE_NEWNS** 0x00020000
- #define **CLONE_SYSVSEM** 0x00040000
- #define **CLONE_SETTLS** 0x00080000
- #define **CLONE_PARENT_SETTID** 0x00100000
- #define **CLONE_CHILD_CLEARTID** 0x00200000
- #define **CLONE_DETACHED** 0x00400000
- #define **CLONE_UNTRACED** 0x00800000

- #define **CLONE_CHILD_SETTID** 0x01000000
- #define **CLONE_STOPPED** 0x02000000
- #define **CLONE_NEWUTS** 0x04000000
- #define **CLONE_NEWIPC** 0x08000000
- #define **CLONE_KTHREAD** 0x10000000
- #define **NXF_INFO_PRIVATE** 0x00000008
- #define **NXF_STATE_SETUP** (1ULL<<32)
- #define **NXF_STATE_ADMIN** (1ULL<<34)
- #define **NXF_SC_HELPER** (1ULL<<36)
- #define **NXF_PERSISTENT** (1ULL<<38)
- #define **NXA_TYPE_IPV4** 1
- #define **NXA_TYPE_IPV6** 2
- #define **NXA_MOD_BCAST** (1<<8)
- #define **NXA_TYPE_ANY** ((uint16_t)-1)
- #define **VXA_SOCK_UNSPEC** 0
- #define **VXA_SOCK_UNIX** 1
- #define **VXA_SOCK_INET** 2
- #define **VXA_SOCK_INET6** 3
- #define **VXA_SOCK_PACKET** 4
- #define **VXA_SOCK_OTHER** 5
- #define **NXA_SOCK_UNSPEC** VXA_SOCK_UNSPEC
- #define **NXA_SOCK_UNIX** VXA_SOCK_UNIX
- #define **NXA_SOCK_INET** VXA_SOCK_INET
- #define **NXA_SOCK_INET6** VXA_SOCK_INET6
- #define **NXA_SOCK_PACKET** VXA_SOCK_PACKET
- #define **NXA_SOCK_OTHER** VXA_SOCK_OTHER

Typedefs

- typedef uint32_t **xid_t**
- typedef **_vx_info vx_info_t**

  Context information.

- typedef **_vx_stat vx_stat_t**

  Context statistics.

- typedef **_vx_flags vx_flags_t**

  Context/migration flags.

- typedef **_vx_uname vx_uname_t**

  Virtual system information data.

- typedef **_vx_wait vx_wait_t**

  Wait results.

- typedef **_vx_sched vx_sched_t**

  Scheduler values.

- typedef **_vx_sched_info vx_sched_info_t**

Scheduler information.

- typedef **_vx_limit vx_limit_t**
  Resource limits.
- typedef **_vx_limit_stat vx_limit_stat_t**
  Resource limit accounting.
- typedef **_dx_limit dx_limit_t**
  Disk limit values.
- typedef **_ix_attr ix_attr_t**
  Inode attributes.
- typedef uint32_t **nid_t**
- typedef **_nx_info nx_info_t**
  Network context information.
- typedef **_nx_addr nx_addr_t**
  Network address information.
- typedef **_nx_flags nx_flags_t**
  Network context flags.
- typedef **_nx_sock_stat nx_sock_stat_t**
  Accounting data.

Functions

- int **vserver** (uint32_t cmd, uint32_t id, void *data)
  Main vserver syscall interface.
- int **clone** (int flags, void *child_stack)
  Clone system call.
- int **vs_get_version** (void)
  Get vserver version of running kernel.
- int **vs_get_config** (void)
  Get vserver configuration of running kernel.
- int **vx_create** (**xid_t** xid, **vx_flags_t** *data)
  Create a new context.
- int **vx_migrate** (**xid_t** xid, **vx_flags_t** *data)
  Migrate to an existing context.
- int **vx_task_xid** (pid_t pid)
  Get the context ID of a process.

- int **vx_info** (**xid_t** xid, **vx_info_t** *data)
  Get context information.
- int **vx_stat** (**xid_t** xid, **vx_stat_t** *data)
  Get context statistics.
- int **vx_bcaps_set** (**xid_t** xid, **vx_flags_t** *data)
  Set system capabilities.
- int **vx_bcaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get system capabilities.
- int **vx_ccaps_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context capabilities.
- int **vx_ccaps_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context capabilities.
- int **vx_flags_set** (**xid_t** xid, **vx_flags_t** *data)
  Set context flags.
- int **vx_flags_get** (**xid_t** xid, **vx_flags_t** *data)
  Get context flags.
- int **vx_uname_set** (**xid_t** xid, **vx_uname_t** *data)
  Set virtual system information.
- int **vx_uname_get** (**xid_t** xid, **vx_uname_t** *data)
  Get virtual system information.
- int **vx_kill** (**xid_t** xid, pid_t pid, int sig)
  Kill one or more processes.
- int **vx_wait** (**xid_t** xid, **vx_wait_t** *data)
  Wait for context death.
- int **vx_sched_set** (**xid_t** xid, **vx_sched_t** *data)
  Set scheduler values.
- int **vx_sched_get** (**xid_t** xid, **vx_sched_t** *data)
  Get scheduler values.
- int **vx_sched_info** (**xid_t** xid, **vx_sched_info_t** *data)
  Get scheduler information.
- int **vx_limit_mask_get** (**vx_limit_t** *data)
  Get resource limits mask.
- int **vx_limit_set** (**xid_t** xid, **vx_limit_t** *data)

Set resource limit.

- int **vx_limit_get** (**xid_t** xid, **vx_limit_t** *data)
  Get resource limits.
- int **vx_limit_stat** (**xid_t** xid, **vx_limit_stat_t** *data)
  Get resource limit accounting data.
- int **vx_limit_reset** (**xid_t** xid)
  Reset resource limit accounting data.
- int **dx_limit_add** (**xid_t** xid, **dx_limit_t** *data)
  Add disk limit entry.
- int **dx_limit_remove** (**xid_t** xid, **dx_limit_t** *data)
  Remove disk limit.
- int **dx_limit_set** (**xid_t** xid, **dx_limit_t** *data)
  Set disk limit values.
- int **dx_limit_get** (**xid_t** xid, **dx_limit_t** *data)
  Get disk limit values.
- int **ix_attr_set** (**ix_attr_t** *data)
  Set inode attributes.
- int **ix_attr_get** (**ix_attr_t** *data)
  Get inode attributes.
- int **ns_clone** (int flags, void *child_stack)
  Clone the current namespace (FS/IPC/UTS).
- int **ns_enter** (**xid_t** xid, uint64_t mask)
  Enter namespace.
- int **ns_set** (**xid_t** xid, uint64_t mask)
  Set namespace.
- int **nx_create** (**nid_t** nid, **nx_flags_t** *data)
  Create network context.
- int **nx_migrate** (**nid_t** nid)
  Migrate to an existing network context.
- int **nx_task_nid** (pid_t pid)
  Get the network context ID of a process.
- int **nx_info** (**nid_t** nid, **nx_info_t** *data)
  Get network context information.

- int **nx_addr_add** (**nid_t** nid, **nx_addr_t** *data)
  Add network context addresses.
- int **nx_addr_remove** (**nid_t** nid, **nx_addr_t** *data)
  Remove network context addresses.
- int **nx_flags_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context flags.
- int **nx_flags_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context flags.
- int **nx_caps_set** (**nid_t** nid, **nx_flags_t** *data)
  Set network context capabilities.
- int **nx_caps_get** (**nid_t** nid, **nx_flags_t** *data)
  Get network context capabilities.
- int **nx_sock_stat** (**nid_t** nid, **nx_sock_stat_t** *data)
  Get network socket accounting data.

6.10.2 Define Documentation

6.10.2.1 #define LIBVSERVER_API_MAJOR 2

API major version

Definition at line 28 of file vserver.h.

6.10.2.2 #define LIBVSERVER_API_MINOR 0

API minor version

Definition at line 29 of file vserver.h.

6.10.3 Function Documentation

6.10.3.1 int vserver (uint32_t *cmd*, uint32_t *id*, void * *data*)

Main vserver syscall interface.

Parameters:

cmd Command number

id Context ID (sometimes process id)

data Data structures

Returns:

Syscall return code

See also:

Referenced by dx_limit_add(), dx_limit_get(), dx_limit_remove(), dx_limit_set(), ix_attr_get(), ix_attr_set(), ns_enter(), ns_set(), nx_addr_add(), nx_addr_remove(), nx_caps_get(), nx_caps_set(), nx_create(), nx_flags_get(), nx_flags_set(), nx_info(), nx_migrate(), nx_sock_stat(), nx_task_nid(), vs_get_config(), vs_get_version(), vx_bcaps_get(), vx_bcaps_set(), vx_ccaps_get(), vx_ccaps_set(), vx_create(), vx_flags_get(), vx_flags_set(), vx_info(), vx_kill(), vx_limit_get(), vx_limit_mask_get(), vx_limit_reset(), vx_limit_set(), vx_limit_stat(), vx_migrate(), vx_sched_get(), vx_sched_info(), vx_sched_set(), vx_stat(), vx_task_xid(), vx_uname_get(), vx_uname_set(), and vx_wait().

6.10.3.2 int clone (int *flags*, void * *child_stack*)

Clone system call.

Parameters:

flags Clone flags

child_stack Child stack

Returns:

Process ID in parent, 0 in child, -1 on error

See also:

clone(2)

Referenced by ns_clone().

6.10.3.3 int vs_get_version (void)

Get vserver version of running kernel.

Returns:

Kernel version

Definition at line 25 of file switch.c.

References vserver().

```
26 {
27         return vserver(VCMD_get_version, 0, NULL);
28 }
```

6.10.3.4 int vs_get_config (void)

Get vserver configuration of running kernel.

Returns:

Kernel configuration

Definition at line 30 of file switch.c.

References vserver().

```
31 {
32         return vserver(VCMD_get_vci, 0, NULL);
33 }
```

Index