

\*\*\*\*\*

10282 Sun Jan 5 00:17:40 2014

new/usr/src/man/man1/pgrep.1

3563 pgrep not working as documented or documentation inaccurate

\*\*\*\*\*

```

1 \" te
2 .\" Copyright (c) 2004, Sun Microsystems, Inc. All Rights Reserved
3 .\" The contents of this file are subject to the terms of the Common Development
4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
6 .TH PGREP 1 "Dec 30, 2013"
7 .TH PGREP 1 "May 6, 2004"
8 .SH NAME
9 pgrep, pkill \- find or signal processes by name and other attributes
10 .SH SYNOPSIS
11 .LP
12 \fBpgrep\fR [\fB-flvx\fR] [\fB-n\fR | \fB-o\fR] [\fB-d\fR \fIdelim\fR] [\fB-P\fR
13 [\fB-g\fR \fIpgprlist\fR] [\fB-s\fR \fIsidlist\fR] [\fB-u\fR \fIeuidlist\fR
14 [\fB-G\fR \fIgidlist\fR] [\fB-J\fR \fIprojidlist\fR] [\fB-t\fR \fItermrlist\
15 [\fB-T\fR \fItaskidlist\fR] [\fB-c\fR \fIctidlist\fR] [\fB-z\fR \fIzoneidli
16 [\fIpattern\fR]
17 .fi
18 .LP
19 .LP
20 .nf
21 \fBpkill\fR [\fB-f\fIsignal\fR] [\fB-fvx\fR] [\fB-n\fR | \fB-o\fR] [\fB-P\fR \
22 [\fB-g\fR \fIpgprlist\fR] [\fB-s\fR \fIsidlist\fR] [\fB-u\fR \fIeuidlist\fR
23 [\fB-G\fR \fIgidlist\fR] [\fB-J\fR \fIprojidlist\fR] [\fB-t\fR \fItermrlist\
24 [\fB-T\fR \fItaskidlist\fR] [\fB-c\fR \fIctidlist\fR] [\fB-z\fR \fIzoneidli
25 [\fIpattern\fR]
26 .fi
27 .SH DESCRIPTION
28 .sp
29 .LP
30 .nf
31 The \fBpgrep\fR utility examines the active processes on the system and reports
32 the process \fBID\fRs of the processes whose attributes match the criteria
33 specified on the command line. Each process \fBID\fR is printed as a decimal
34 value and is separated from the next \fBID\fR by a delimiter string, which
35 defaults to a newline. For each attribute option, the user can specify a set of
36 possible values separated by commas on the command line. For example,
37 .sp
38 .in +2
39 .nf
40 \fBpgrep -G other,daemon\fR
41 .fi
42 .in -2
43 .sp
44 .sp
45 .sp
46 .LP
47 matches processes whose real group \fBID\fR is \fBOther\fR \fBBOR\fR
48 \fBdaemon\fR. If multiple criteria options are specified, \fBpgrep\fR matches
49 processes whose attributes match the logical \fBAND\fR of the criteria options.
50 For example,
51 .sp
52 .in +2
53 .nf
54 \fBpgrep -G other,daemon -U root,daemon\fR
55 .fi
56 .in -2
57 .sp
58 .sp
59 .sp
60 .LP

```

```

61 matches processes whose attributes are:
62 .br
63 .in +2
64 (real group \fBID\fR is \fBOther\fR \fBBOR\fR \fBdaemon\fR) \fBAND\fR
65 .in -2
66 .br
67 .in +2
68 (real user \fBID\fR is \fBroot\fR \fBBOR\fR \fBdaemon\fR)
69 .in -2
70 .sp
71 .LP
72 \fBpkill\fR functions identically to \fBpgrep\fR, except that each matching
73 process is signaled as if by \fBkill\fR(1) instead of having its process
74 \fBID\fR printed. A signal name or number may be specified as the first command
75 line option to \fBpkill\fR.
76 .SH OPTIONS
77 .sp
78 .LP
79 The following options are supported:
80 .sp
81 .ne 2
82 .na
83 \fB\fB-c\fR \fIctidlist\fR\fR
84 .ad
85 .RS 17n
86 Matches only processes whose process contract ID is in the given list.
87 .RE
88 .sp
89 .sp
90 .ne 2
91 .na
92 \fB\fB-d\fR \fIdelim\fR\fR
93 .ad
94 .RS 17n
95 Specifies the output delimiter string to be printed between each matching
96 process \fBID\fR. If no \fB-d\fR option is specified, the default is a newline
97 character. The \fB-d\fR option is only valid when specified as an option to
98 \fBpgrep\fR.
99 .RE
100 .sp
101 .sp
102 .ne 2
103 .na
104 \fB\fB-f\fR \fR
105 .ad
106 .RS 17n
107 The regular expression \fIpattern\fR should be matched against the full process
108 argument string (obtained from the \fBpr_psargs\fR field of the
109 \fB/proc/\fInnnn\fR/\fR/\fR file). If no \fB-f\fR option is specified, the
110 expression is matched only against the name of the executable file (obtained
111 from the \fBpr_fname\fR field of the \fB/proc/\fInnnn\fR/\fR/\fR file).
112 .RE
113 .sp
114 .sp
115 .ne 2
116 .na
117 \fB\fB-g\fR \fIpgprlist\fR\fR
118 .ad
119 .RS 17n
120 Matches only processes whose process group \fBID\fR is in the given list. If
121 group 0 is included in the list, this is interpreted as the process group
122 \fBID\fR of the \fBpgrep\fR or \fBpkill\fR process.
123 .RE
124 .sp
125 .sp
126 .ne 2

```

```

127 .na
128 \fB\fB-G\fR \fIgidlist\fR\fR
129 .ad
130 .RS 17n
131 Matches only processes whose real group \fBID\fR is in the given list. Each
132 group \fBID\fR may be specified as either a group name or a numerical group
133 \fBID\fR.
134 .RE

136 .sp
137 .ne 2
138 .na
139 \fB\fB-J\fR \fIprojidlist\fR\fR
140 .ad
141 .RS 17n
142 Matches only processes whose project \fBID\fR is in the given list. Each
143 project \fBID\fR may be specified as either a project name or a numerical
144 project \fBID\fR.
145 .RE

147 .sp
148 .ne 2
149 .na
150 \fB\fB-l\fR\fR
151 .ad
152 .RS 17n
153 Long output format. Prints the process name along with the process \fBID\fR of
154 each matching process. The process name is obtained from the \fBpr_psargs\fR or
155 \fBpr_fname\fR field, depending on whether the \fB-f\fR option was specified
156 (see above). The \fB-l\fR option is only valid when specified as an option to
157 \fBpgrep\fR.
158 .RE

160 .sp
161 .ne 2
162 .na
163 \fB\fB-n\fR\fR
164 .ad
165 .RS 17n
166 Matches only the newest (most recently created) process that meets all other
167 specified matching criteria. Cannot be used with option \fB-o\fR.
168 .RE

170 .sp
171 .ne 2
172 .na
173 \fB\fB-o\fR\fR
174 .ad
175 .RS 17n
176 Matches only the oldest (earliest created) process that meets all other
177 specified matching criteria. Cannot be used with option \fB-n\fR.
178 .RE

180 .sp
181 .ne 2
182 .na
183 \fB\fB-P\fR \fIppidlist\fR\fR
184 .ad
185 .RS 17n
186 Matches only processes whose parent process \fBID\fR is in the given list.
187 .RE

189 .sp
190 .ne 2
191 .na
192 \fB\fB-s\fR \fIsidlist\fR\fR

```

```

193 .ad
194 .RS 17n
195 Matches only processes whose process session \fBID\fR is in in the given list.
196 If \fBID\fR 0 is included in the list, this is interpreted as the session
197 \fBID\fR of the \fBpgrep\fR or \fBpkill\fR process.
198 .RE

200 .sp
201 .ne 2
202 .na
203 \fB\fB-t\fR \fItermlist\fR\fR
204 .ad
205 .RS 17n
206 Matches only processes which are associated with a terminal in the given list.
207 Each terminal is specified as the suffix following "/dev/" of the terminal's
208 device path name in \fB/dev\fR. For example, \fBterm/a\fR or \fBpts/0\fR.
209 .RE

211 .sp
212 .ne 2
213 .na
214 \fB\fB-T\fR \fItaskidlist\fR\fR
215 .ad
216 .RS 17n
217 Matches only processes whose task \fBID\fR is in the given list. If \fBID\fR 0
218 is included in the list, this is interpreted as the task \fBID\fR of the
219 \fBpgrep\fR or \fBpkill\fR process.
220 .RE

222 .sp
223 .ne 2
224 .na
225 \fB\fB-u\fR \fIeuidlist\fR\fR
226 .ad
227 .RS 17n
228 Matches only processes whose effective user \fBID\fR is in the given list. Each
229 user \fBID\fR may be specified as either a login name or a numerical user
230 \fBID\fR.
231 .RE

233 .sp
234 .ne 2
235 .na
236 \fB\fB-U\fR \fIuidlist\fR\fR
237 .ad
238 .RS 17n
239 Matches only processes whose real user \fBID\fR is in the given list. Each user
240 \fBID\fR may be specified as either a login name or a numerical user \fBID\fR.
241 .RE

243 .sp
244 .ne 2
245 .na
246 \fB\fB-v\fR\fR
247 .ad
248 .RS 17n
249 Reverses the sense of the matching. Matches all processes \fBexcept\fR those
250 which meet the specified matching criteria.
251 .RE

253 .sp
254 .ne 2
255 .na
256 \fB\fB-x\fR\fR
257 .ad
258 .RS 17n

```

```

259 Considers only processes whose executable file name \fBexactly\fR matches the
260 specified \fIpattern\fR. The pattern match is considered to be exact when all
261 characters in the executable file name match the pattern.
259 Considers only processes whose argument string or executable file name
260 \fBexactly\fR matches the specified \fIpattern\fR to be matching processes. The
261 pattern match is considered to be exact when all characters in the process
262 argument string or executable file name match the pattern.
262 .RE

264 .sp
265 .ne 2
266 .na
267 \fB\fB-z\fR \fIzoneidlist\fR\fR
268 .ad
269 .RS 17n
270 Matches only processes whose zone \fBID\fR is in the given list. Each zone
271 \fBID\fR may be specified as either a zone name or a numerical zone \fBID\fR.
272 This option is only useful when executed in the global zone. If the \fBkill\fR
273 utility is used to send signals to processes in other zones, the process must
274 have asserted the \fB{PRIV_PROC_ZONE}\fR privilege (see \fBprivileges\fR(5)).
275 .RE

277 .sp
278 .ne 2
279 .na
280 \fB\fB-\fR \fIsignal\fR\fR
281 .ad
282 .RS 17n
283 Specifies the signal to send to each matched process. If no signal is
284 specified, \fBSIGTERM\fR is sent by default. The value of \fIsignal\fR can be
285 one of the symbolic names defined in \fBsignal.h\fR(3HEAD) without the
286 \fBSIG\fR prefix, or the corresponding signal number as a decimal value. The
287 \fB-\fR \fIsignal\fR option is only valid when specified as the first option to
288 \fBkill\fR.
289 .RE

291 .SH OPERANDS
292 .sp
293 .LP
294 The following operand is supported:
295 .sp
296 .ne 2
297 .na
298 \fB\fIpattern\fR\fR
299 .ad
300 .RS 11n
301 Specifies an Extended Regular Expression (\fBERE\fR) pattern to match against
302 either the executable file name or full process argument string. See
303 \fBRegex\fR(5) for a complete description of the \fBERE\fR syntax.
304 .RE

306 .SH EXAMPLES
307 .LP
308 \fBExample 1 \fRObtaining a Process ID
309 .sp
310 .LP
311 Obtain the process \fBID\fR of \fBsendmail\fR:

313 .sp
314 .in +2
315 .nf
316 example% \fBpgrep -x -u root sendmail\fR
317 283
318 .fi
319 .in -2
320 .sp

```

```

322 .LP
323 \fBExample 2 \fRTerminating a Process
324 .sp
325 .LP
326 Terminate the most recently created \fBxterm\fR:

328 .sp
329 .in +2
330 .nf
331 example% \fBkill -n xterm\fR
332 .fi
333 .in -2
334 .sp

336 .SH EXIT STATUS
337 .sp
338 .LP
339 The following exit values are returned:
340 .sp
341 .ne 2
342 .na
343 \fB0\fR\fR
344 .ad
345 .RS 5n
346 One or more processes were matched.
347 .RE

349 .sp
350 .ne 2
351 .na
352 \fB1\fR\fR
353 .ad
354 .RS 5n
355 No processes were matched.
356 .RE

358 .sp
359 .ne 2
360 .na
361 \fB2\fR\fR
362 .ad
363 .RS 5n
364 Invalid command line options were specified.
365 .RE

367 .sp
368 .ne 2
369 .na
370 \fB3\fR\fR
371 .ad
372 .RS 5n
373 A fatal error occurred.
374 .RE

376 .SH FILES
377 .sp
378 .ne 2
379 .na
380 \fB/proc/\fInnnn\fR/\fBpsinfo\fR\fR
381 .ad
382 .RS 22n
383 Process information files
384 .RE

386 .SH SEE ALSO

```

387 .sp  
388 .LP  
389 \fBkill\fR(1), \fBproc\fR(1), \fBps\fR(1), \fBtruss\fR(1), \fBkill\fR(2),  
390 \fBsignal.h\fR(3HEAD), \fBproc\fR(4), \fBattributes\fR(5), \fBprivileges\fR(5),  
391 \fBregex\fR(5), \fBzones\fR(5)  
392 .SH NOTES  
393 .sp  
394 .LP  
395 Both utilities match the \fBERE\fR \fIpattern\fR argument against either the  
396 \fBpr\_fname\fR or \fBpr\_psargs\fR fields of the  
397 \fB/proc/\fR\fInnnn\fR\fB/psinfo\fR files. The lengths of these strings are  
398 limited according to definitions in \fB<sys/procfs.h>\fR&. Patterns which can  
399 match strings longer than the current limits may fail to match the intended set  
400 of processes.  
401 .sp  
402 .LP  
403 If the \fIpattern\fR argument contains \fBERE\fR meta-characters which are also  
404 shell meta-characters, it may be necessary to enclose the pattern with  
405 appropriate shell quotes.  
406 .sp  
407 .LP  
408 Defunct processes are never matched by either \fBpgrep\fR or \fBpkill\fR.  
409 .sp  
410 .LP  
411 The current \fBpgrep\fR or \fBpkill\fR process will never consider itself a  
412 potential match.

```

*****
13902 Sun Jan 5 00:17:41 2014
new/usr/src/man/man3nsl/rpc_svc_create.3nsl
4344 Minor typos in the 3nsl man pages
*****
1 \" te
2 .\" Copyright 1989 AT&T
3 .\" Copyright (C) 2005, Sun Microsystems, Inc. All Rights Reserved.
4 .\" The contents of this file are subject to the terms of the Common Development
5 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
6 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
7 .TH RPC_SVC_CREATE 3NSL "Dec 27, 2013"
7 .TH RPC_SVC_CREATE 3NSL "Mar 22, 2005"
8 .SH NAME
9 rpc_svc_create, svc_control, svc_create, svc_destroy, svc_dg_create,
10 svc_fd_create, svc_raw_create, svc_tli_create, svc_tp_create, svc_vc_create,
11 svc_door_create \- server handle creation routines
12 .SH SYNOPSIS
13 .LP
14 .nf
15 #include <rpc/rpc.h>

17 \fBbool_t\fR \fBsvc_control\fR(\fB SVCXPRT *\fR \fIsvcr\fR, \fBconst uint_t\fR \fIrr
18 .fi

20 .LP
21 .nf
22 \fBint\fR \fBsvc_create\fR(\fBconst void (*\fR \fIdispatch\fR)(\fBconst struct svc_r
23 const SVCXPRT *), \fBconst rpcprog_t\fR \fIiprognum\fR, \fBconst rpcvers_t\f
22 \fBint\fR \fBsvc_create\fR(\fBconst void (*\fR \fIdispatch\fR)\fBconst struct svc_re
23 const SVCXPRT *, \fBconst rpcprog_t\fR \fIiprognum\fR, \fBconst rpcvers_t\fR
24 \fBconst char *\fR \fIinetype\fR);
25 .fi

27 .LP
28 .nf
29 \fBvoid\fR \fBsvc_destroy\fR(\fB SVCXPRT *\fR \fIxprrt\fR);
30 .fi

32 .LP
33 .nf
34 \fB SVCXPRT *\fR \fBsvc_dg_create\fR(\fBconst int\fR \fIifildes\fR, \fBconst uint_t
35 \fBconst uint_t\fR \fIirecvsz\fR);
36 .fi

38 .LP
39 .nf
40 \fB SVCXPRT *\fR \fBsvc_fd_create\fR(\fBconst int\fR \fIifildes\fR, \fBconst uint_t
41 \fBconst uint_t\fR \fIirecvsz\fR);
42 .fi

44 .LP
45 .nf
46 \fB SVCXPRT *\fR \fBsvc_raw_create\fR(\fBvoid)
47 .fi

49 .LP
50 .nf
51 \fB SVCXPRT *\fR \fBsvc_tli_create\fR(\fBconst int\fR \fIifildes\fR, \fBconst struc
52 \fBconst struct t_bind *\fR \fIbind_addr\fR, \fBconst uint_t\fR \fIisendsz\fR
53 \fBconst uint_t\fR \fIirecvsz\fR);
54 .fi

56 .LP
57 .nf
58 \fB SVCXPRT *\fR \fBsvc_tp_create\fR(\fBconst void (*\fR \fIdispatch\fR)

```

```

59 (const struct svc_req *, const SVCXPRT *), \fBconst rpcprog_t\fR \fIiprognum
59 const struct svc_req *, const SVCXPRT *), \fBconst rpcprog_t\fR \fIiprognum\
60 \fBconst rpcvers_t\fR \fIiversnum\fR, \fBconst struct netconfig *\fR \fIinetco
61 .fi

63 .LP
64 .nf
65 \fB SVCXPRT *\fR \fBsvc_vc_create\fR(\fBconst int\fR \fIifildes\fR, \fBconst uint_t
66 \fBconst uint_t\fR \fIirecvsz\fR);
67 .fi

69 .LP
70 .nf
71 \fB SVCXPRT *\fR \fBsvc_door_create\fR(\fBvoid (*\fR \fIdispatch\fR)(\fBconst struct svc_req
72 \fBconst rpcprog_t\fR \fIiprognum\fR, \fBconst rpcvers_t\fR \fIiversnum\fR,
73 \fBconst uint_t\fR \fIisendsz\fR);
74 .fi

76 .SH DESCRIPTION
77 .sp
78 .LP
79 These routines are part of the \fB RPC\fR library which allows C language
80 programs to make procedure calls on servers across the network. These routines
81 deal with the creation of service handles. Once the handle is created, the
82 server can be invoked by calling \fB svc_run()\fR.
83 .SS "Routines"
84 .sp
85 .LP
86 See \fB rpc(3NSL)\fR for the definition of the \fB SVCXPRT\fR data structure.
87 .sp
88 .ne 2
89 .na
90 \fB svc_control()\fR \fR
91 .ad
92 .RS 21n
93 A function to change or retrieve information about a service object. \fIreq\fR
94 indicates the type of operation and \fIinfo\fR is a pointer to the information.
95 The supported values of \fIreq\fR, their argument types, and what they do are:
96 .sp
97 .ne 2
98 .na
99 \fB SVCGET_VERSQUIET\fR \fR
100 .ad
101 .RS 25n
102 If a request is received for a program number served by this server but the
103 version number is outside the range registered with the server, an
104 \fB RPC_PROGVERSMISMATCH\fR error will normally be returned. \fIinfo\fR should
105 be a pointer to an integer. Upon successful completion of the
106 \fB SVCGET_VERSQUIET\fR request, *\fIinfo\fR contains an integer which
107 describes the server's current behavior: \fB0\fR indicates normal server
108 behavior, that is, an \fB RPC_PROGVERSMISMATCH\fR error will be returned.
109 \fB1\fR indicates that the out of range request will be silently ignored.
110 .RE

112 .sp
113 .ne 2
114 .na
115 \fB SVCSET_VERSQUIET\fR \fR
116 .ad
117 .RS 25n
118 If a request is received for a program number served by this server but the
119 version number is outside the range registered with the server, an
120 \fB RPC_PROGVERSMISMATCH\fR error will normally be returned. It is sometimes
121 desirable to change this behavior. \fIinfo\fR should be a pointer to an integer
122 which is either \fB0\fR, indicating normal server behavior and an
123 \fB RPC_PROGVERSMISMATCH\fR error will be returned, or \fB1\fR, indicating that

```

```

124 the out of range request should be silently ignored.
125 .RE

127 .sp
128 .ne 2
129 .na
130 \fB\fBSVCGET_XID\fR\fR
131 .ad
132 .RS 25n
133 Returns the transaction \fBID\fR of connection\(\mioriented and connectionless
134 transport service calls. The transaction \fBID\fR assists in uniquely
135 identifying client requests for a given \fBRPC\fR version, program number,
136 procedure, and client. The transaction \fBID\fR is extracted from the service
137 transport handle \fIsvc\fR. \fIinfo\fR must be a pointer to an unsigned long.
138 Upon successful completion of the \fBSVCGET_XID\fR request, *\fIinfo\fR
139 contains the transaction \fBID\fR. Note that rendezvous and raw service
140 handles do not define a transaction \fBID\fR. Thus, if the service handle is
141 of rendezvous or raw type, and the request is of type \fBSVCGET_XID\fR
142 \fBsvc_control()\fR will return \fBFALSE\fR. Note also that the transaction
143 \fBID\fR read by the server can be set by the client through the suboption
144 \fBCLSET_XID\fR in \fBclnt_control()\fR. See \fBclnt_create\fR(3NSL)
145 .RE

147 .sp
148 .ne 2
149 .na
150 \fB\fBSVCSET_RECVERRHANDLER\fR\fR
151 .ad
152 .RS 25n
153 Attaches or detaches a disconnection handler to the service handle, \fIsvc\fR,
154 that will be called when a transport error arrives during the reception of a
155 request or when the server is waiting for a request and the connection shuts
156 down. This handler is only useful for a connection oriented service handle.
157 .sp
158 \fI*info\fR contains the address of the error handler to attach, or \fINULL\fR
159 to detach a previously defined one. The error handler has two arguments. It has
160 a pointer to the erroneous service handle. It also has an integer that
161 indicates if the full service is closed (when equal to zero), or that only one
162 connection on this service is closed (when not equal to zero).
163 .sp
164 .in +2
165 .nf
166 void handler (const SVCXPRT *svc, const bool_t isAConnection);
167 .fi
168 .in -2

170 With the service handle address, \fIsvc\fR, the error handler is able to detect
171 which connection has failed and to begin an error recovery process. The error
172 handler can be called by multiple threads and should be implemented in an
173 MT-safe way.
174 .RE

176 .sp
177 .ne 2
178 .na
179 \fB\fBSVCGET_RECVERRHANDLER\fR\fR
180 .ad
181 .RS 25n
182 Upon successful completion of the \fBSVCGET_RECVERRHANDLER\fR request,
183 \fI*info\fR contains the address of the handler for receiving errors. Upon
184 failure, \fI*info\fR contains \fINULL\fR.
185 .RE

187 .sp
188 .ne 2
189 .na

```

```

190 \fB\fBSVCSET_CONNMAXREC\fR\fR
191 .ad
192 .RS 25n
193 Set the maximum record size (in bytes) and enable non-blocking mode for this
194 service handle. Value can be set and read for both connection and
195 non-connection oriented transports, but is silently ignored for the
196 non-connection oriented case. The \fIinfo\fR argument should be a pointer to an
197 \fBInt\fR.
198 .RE

200 .sp
201 .ne 2
202 .na
203 \fB\fBSVCGET_CONNMAXREC\fR\fR
204 .ad
205 .RS 25n
206 Get the maximum record size for this service handle. Zero means no maximum in
207 effect and the connection is in blocking mode. The result is not significant
208 for non-connection oriented transports. The \fIinfo\fR argument should be a
209 pointer to an \fBInt\fR.
210 .RE

212 This routine returns TRUE if the operation was successful. Otherwise, it
213 returns false.
214 .RE

216 .sp
217 .ne 2
218 .na
219 \fB\fBsvc_create()\fR\fR
220 .ad
221 .RS 21n
222 \fBsvc_create()\fR creates server handles for all the transports belonging to
223 the class \fInettype\fR.
224 .sp
225 \fInettype\fR defines a class of transports which can be used for a particular
226 application. The transports are tried in left to right order in \fBNETPATH\fR
227 variable or in top to bottom order in the netconfig database. If \fInettype\fR
228 is \fINULL\fR it defaults to \fBnetpath\fR.
229 .sp
230 \fBsvc_create()\fR registers itself with the \fBbrpcbind\fR service (see
231 \fBbrpcbind\fR(1M)). \fBdispatch\fR is called when there is a remote procedure
232 call for the given \fIprognum\fR and \fIversnum\fR; this requires calling
233 \fBsvc_run()\fR (see \fBsvc_run()\fR in \fBbrpc_svc_reg\fR(3NSL)). If
234 \fBsvc_create()\fR succeeds, it returns the number of server handles it
235 created, otherwise it returns \fB0\fR and an error message is logged.
236 .RE

238 .sp
239 .ne 2
240 .na
241 \fB\fBsvc_destroy()\fR\fR
242 .ad
243 .RS 21n
244 A function macro that destroys the \fBRPC\fR service handle \fIxpirt\fR.
245 Destruction usually involves deallocation of private data structures, including
246 \fIxpirt\fR itself. Use of \fIxpirt\fR is undefined after calling this routine.
247 .RE

249 .sp
250 .ne 2
251 .na
252 \fB\fBsvc_dg_create()\fR\fR
253 .ad
254 .RS 21n
255 This routine creates a connectionless \fBRPC\fR service handle, and returns a

```

256 pointer to it. This routine returns \fINULL\fr if it fails, and an error  
 257 message is logged. \fIsendsz\fr and \fIrcvsv\fr are parameters used to specify  
 258 the size of the buffers. If they are \fB0\fr, suitable defaults are chosen. The  
 259 file descriptor \fIfildes\fr should be open and bound. The server is not  
 260 registered with \fBrpcbind\fr(1M).  
 261 .sp  
 262 Warning: since connectionless-based \fBRPC\fr messages can only hold limited  
 263 amount of encoded data, this transport cannot be used for procedures that take  
 264 large arguments or return huge results.  
 265 .RE

267 .sp  
 268 .ne 2  
 269 .na  
 270 \fB\fBsvc\_fd\_create()\fr\fr  
 271 .ad  
 272 .RS 21n  
 273 This routine creates a service on top of an open and bound file descriptor, and  
 274 returns the handle to it. Typically, this descriptor is a connected file  
 275 descriptor for a connection-oriented transport. \fIsendsz\fr and \fIrcvsv\fr  
 276 indicate sizes for the send and receive buffers. If they are \fB0\fr,  
 277 reasonable defaults are chosen. This routine returns \fINULL\fr if it fails,  
 278 and an error message is logged.  
 279 .RE

281 .sp  
 282 .ne 2  
 283 .na  
 284 \fB\fBsvc\_raw\_create()\fr\fr  
 285 .ad  
 286 .RS 21n  
 287 This routine creates an \fBRPC\fr service handle and returns a pointer to it.  
 288 The transport is really a buffer within the process's address space, so the  
 289 corresponding \fBRPC\fr client should live in the same address space; (see  
 290 \fBclnt\_raw\_create()\fr in \fBrpc\_clnt\_create\fr(3NSL)). This routine allows  
 291 simulation of \fBRPC\fr and acquisition of \fBRPC\fr overheads (such as round  
 292 trip times), without any kernel and networking interference. This routine  
 293 returns \fINULL\fr if it fails, and an error message is logged.  
 294 .sp  
 295 Note: \fBsvc\_run()\fr should not be called when the raw interface is being  
 296 used.  
 297 .RE

299 .sp  
 300 .ne 2  
 301 .na  
 302 \fB\fBsvc\_tli\_create()\fr\fr  
 303 .ad  
 304 .RS 21n  
 305 This routine creates an \fBRPC\fr server handle, and returns a pointer to it.  
 306 \fIfildes\fr is the file descriptor on which the service is listening. If  
 307 \fIfildes\fr is \fBRPC\_ANYFD\fr, it opens a file descriptor on the transport  
 308 specified by \fInetconf\fr. If the file descriptor is unbound and  
 309 \fIbindaddr\fr is non-null \fIfildes\fr is bound to the address specified by  
 310 \fIbindaddr\fr, otherwise \fIfildes\fr is bound to a default address chosen by  
 311 the transport. In the case where the default address is chosen, the number of  
 312 outstanding connect requests is set to 8 for connection-oriented transports.  
 313 The user may specify the size of the send and receive buffers with the  
 314 parameters \fIsendsz\fr and \fIrcvsv\fr \fI;\fr values of \fB0\fr choose  
 315 suitable defaults. This routine returns \fINULL\fr if it fails, and an error  
 316 message is logged. The server is not registered with the \fBrpcbind\fr(1M)  
 317 service.  
 318 .RE

320 .sp  
 321 .ne 2

322 .na  
 323 \fB\fBsvc\_tp\_create()\fr\fr  
 324 .ad  
 325 .RS 21n  
 326 \fBsvc\_tp\_create()\fr creates a server handle for the network specified by  
 327 \fInetconf\fr, and registers itself with the \fBrpcbind\fr service.  
 328 \fIdispatch\fr is called when there is a remote procedure call for the given  
 329 \fIprognum\fr and \fIversnum\fr; this requires calling \fBsvc\_run()\fr.  
 330 \fBsvc\_tp\_create()\fr returns the service handle if it succeeds, otherwise a  
 331 \fINULL\fr is returned and an error message is logged.  
 332 .RE

334 .sp  
 335 .ne 2  
 336 .na  
 337 \fB\fBsvc\_vc\_create()\fr\fr  
 338 .ad  
 339 .RS 21n  
 340 This routine creates a connection-oriented \fBRPC\fr service and returns a  
 341 pointer to it. This routine returns \fINULL\fr if it fails, and an error  
 342 message is logged. The users may specify the size of the send and receive  
 343 buffers with the parameters \fIsendsz\fr and \fIrcvsv\fr; values of \fB0\fr  
 344 choose suitable defaults. The file descriptor \fIfildes\fr should be open and  
 345 bound. The server is not registered with the \fBrpcbind\fr(1M) service.  
 346 .RE

348 .sp  
 349 .ne 2  
 350 .na  
 351 \fB\fBsvc\_door\_create()\fr\fr  
 352 .ad  
 353 .RS 21n  
 354 This routine creates an RPC server handle over doors and returns a pointer to  
 355 it. Doors is a transport mechanism that facilitates fast data transfer between  
 356 processes on the same machine. for the given program The user may set the size  
 357 of the send buffer with the parameter \fIsendsz\fr. If \fIsendsz\fr is 0, the  
 358 corresponding default buffer size is 16 Kbyte. If successful, the  
 359 \fBsvc\_door\_create()\fr routine returns the service handle. Otherwise it  
 360 returns \fINULL\fr and sets a value for \fBrpc\_createerr\fr. The server is not  
 361 registered with \fBrpcbind\fr(1M). The \fBsvc\_set\_connmaxrec\fr and  
 362 \fBsvc\_get\_connmaxrec\fr \fBsvc\_control()\fr requests can be used to set and  
 363 change the maximum allowed request size for the doors transport.  
 364 .RE

366 .SH ATTRIBUTES  
 367 .sp  
 368 .LP  
 369 See \fBattributes\fr(5) for descriptions of the following attributes:  
 370 .sp

372 .sp  
 373 .TS  
 374 box;  
 375 c | c  
 376 l | l  
 377 ATTRIBUTE TYPE ATTRIBUTE VALUE  
 378 \_  
 379 Architecture All  
 380 \_  
 381 Interface Stability Evolving  
 382 \_  
 383 MT-Level MT-Safe  
 384 .TE

386 .SH SEE ALSO  
 387 .sp

new/usr/src/man/man3nsl/rpc\_svc\_create.3nsl

7

```
388 .LP
389 \fBrpcbind\fR(1M), \fBrpc\fR(3NSL), \fBrpc_clnt_create\fR(3NSL),
390 \fBrpc_svc_calls\fR(3NSL), \fBrpc_svc_err\fR(3NSL), \fBrpc_svc_reg\fR(3NSL),
391 \fBattributes\fR(5)
```



```

*****
6620 Sun Jan 5 00:17:41 2014
new/usr/src/man/man3nsl/rpcbnd.3nsl
4344 Minor typos in the 3nsl man pages
*****
1 \" te
2.\" Copyright 1989 AT&T Copyright (c) 1997, Sun Microsystems, Inc. All Rights
3.\" The contents of this file are subject to the terms of the Common Development
4.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
5.\" When distributing Covered Code, include this CDDL HEADER in each file and in
6.TH RPCBIND 3NSL "Dec 27, 2013"
6.TH RPCBIND 3NSL "Feb 20, 1998"
7.SH NAME
8 rpcbind, rpcb_getmaps, rpcb_getaddr, rpcb_gettime, rpcb_rmtcall, rpcb_set,
9 rpcb_unset \- library routines for RPC bind service
10.SH SYNOPSIS
11.LP
12.nf
13 #include <rpc/rpc.h>

17 \fBstruct rpcblist *fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *fR\fBinetco
17 \fBstruct rpcblist *fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *fR\fBinetco
18 \fBconst char *fR\fBihost\fR);
19 .fi

21.LP
22.nf
23 \fBbool_trpcb_getaddr\fR(\fBconst rpcprog_tiprognum\fR, \fBconst
24 \fBconst struct netconfig *fR\fBinetconf\fR, \fBstruct netbuf *fR\fBissvcad
25 \fBconst char *fR\fBihost\fR);
26 .fi

28.LP
29.nf
30 \fBbool_trpcb_gettime\fR(\fBconst char *fR\fBihost\fR, \fBtime_t *fR\fBit
31 .fi

33.LP
34.nf
35 \fBenum clnt_statrpcb_rmtcall\fR(\fBconst struct netconfig *fR\fBinetconf
36 \fBconst char *fR\fBihost\fR, \fBconst rpcprog_tiprognum\fR,
37 \fBconst rpcvers_tiversnum\fR, \fBconst rpcproc_tiprocnun\fR,
38 \fBconst xdrproc_tlinproc\fR, \fBconst caddr_tlin\fR,
39 \fBconst xdrproc_tioutproc\fR, \fBconst tiout\fR,
39 \fBconst xdrproc_tioutproc\fR, \fBconst tiout\fR,,
40 \fBconst struct timevalitout\fR, \fBstruct netbuf *fR\fBissvcaddr\fR)
41 .fi

43.LP
44.nf
45 \fBbool_trpcb_set\fR(\fBconst rpcprog_tiprognum\fR, \fBconst rpcver
46 \fBconst struct netconfig *fR\fBinetconf\fR, \fBconst struct netbuf *fR\fBi
47 .fi

49.LP
50.nf
51 \fBbool_trpcb_unset\fR(\fBconst rpcprog_tiprognum\fR, \fBconst rpcv
52 \fBconst struct netconfig *fR\fBinetconf\fR);
53 .fi

55.SH DESCRIPTION
56.sp
57.LP
58 These routines allow client C programs to make procedure calls to the RPC

```

```

59 binder service. \fBrpcbnd\fR maintains a list of mappings between programs and
60 their universal addresses. See \fBrpcbnd\fR(1M).
61 .SS "Routines"
62 .sp
63 .ne 2
64 .na
65 \fBrpcb_getmaps\fR()
66 .ad
67 .RS 18n
68 An interface to the \fBrpcbnd\fR service, which returns a list of the current
69 \fBRPC\fR program-to-address mappings on \fBihost\fR. It uses the transport
70 specified through \fBinetconf\fR to contact the remote \fBrpcbnd\fR service on
71 \fBihost\fR. This routine will return \fBNULL\fR if the remote \fBrpcbnd\fR
72 could not be contacted.
73 .RE

75 .sp
76 .ne 2
77 .na
78 \fBrpcb_getaddr\fR()
79 .ad
80 .RS 18n
81 An interface to the \fBrpcbnd\fR service, which finds the address of the
82 service on \fBihost\fR that is registered with program number \fBiprognum\fR,
83 version \fBiversnum\fR, and speaks the transport protocol associated with
84 \fBinetconf\fR. The address found is returned in \fBissvcaddr\fR. \fBissvcaddr\fR
85 should be preallocated. This routine returns \fBTRUE\fR if it succeeds. A
86 return value of \fBFALSE\fR means that the mapping does not exist or that the
87 \fBRPC\fR system failed to contact the remote \fBrpcbnd\fR service. In the
88 latter case, the global variable \fBrpc_createerr\fR contains the \fBRPC\fR
89 status. See \fBrpc_clnt_create\fR(3NSL).
90 .RE

92 .sp
93 .ne 2
94 .na
95 \fBrpcb_gettime\fR()
96 .ad
97 .RS 18n
98 This routine returns the time on \fBihost\fR in \fBitimep\fR. If \fBihost\fR is
99 \fBINULL\fR, \fBrpcb_gettime\fR() returns the time on its own machine. This
100 routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR if it fails.
101 \fBrpcb_gettime\fR() can be used to synchronize the time between the client and
102 the remote server. This routine is particularly useful for secure RPC.
103 .RE

105 .sp
106 .ne 2
107 .na
108 \fBrpcb_rmtcall\fR()
109 .ad
110 .RS 18n
111 An interface to the \fBrpcbnd\fR service, which instructs \fBrpcbnd\fR on
112 \fBihost\fR to make an \fBRPC\fR call on your behalf to a procedure on that
113 host. The \fBnetconfig\fR structure should correspond to a connectionless
114 transport. The parameter \fBissvcaddr\fR will be modified to the server's
115 address if the procedure succeeds. See \fBrpc_call\fR() and \fBclnt_call\fR()
116 in \fBrpc_clnt_calls\fR(3NSL) for the definitions of other parameters.
117 .sp
118 This procedure should normally be used for a "ping" and nothing else. This
119 routine allows programs to do lookup and call, all in one step.
120 .sp
121 Note: Even if the server is not running \fBrpcbnd\fR does not return any error
122 messages to the caller. In such a case, the caller times out.
123 .sp
124 Note: \fBrpcb_rmtcall\fR() is only available for connectionless transports.

```

```

125 .RE

127 .sp
128 .ne 2
129 .na
130 \fB\fBrpcb_set()\fR\fR
131 .ad
132 .RS 18n
133 An interface to the \fBrpcbind\fR service, which establishes a mapping between
134 the triple [\fIprognum\fR, \fIversnum\fR, \fInetconf\fR->\fInetid\fR] and
135 \fIsvcadddr\fR on the machine's \fBrpcbind\fR service. The value of
136 \fInetid\fR must correspond to a network identifier that is defined by the
137 netconfig database. This routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR
138 otherwise. See also \fBsvc_reg()\fR in \fBrpc_svc_calls\fR(3NSL). If there
139 already exists such an entry with \fBrpcbind\fR, \fBrpcb_set()\fR will fail.
140 .RE

142 .sp
143 .ne 2
144 .na
145 \fB\fBrpcb_unset()\fR\fR
146 .ad
147 .RS 18n
148 An interface to the \fBrpcbind\fR service, which destroys the mapping between
149 the triple [\fIprognum\fR, \fIversnum\fR, \fInetconf\fR->\fInetid\fR] and
150 the address on the machine's \fBrpcbind\fR service. If \fInetconf\fR is
151 \fINULL\fR, \fBrpcb_unset()\fR destroys all mapping between the triple
152 [\fIprognum\fR, \fIversnum\fR, \fIall-transport\fR] and the addresses on the
153 machine's \fBrpcbind\fR service. This routine returns \fBTRUE\fR if it
154 succeeds, \fBFALSE\fR otherwise. Only the owner of the service or the
155 super-user can destroy the mapping. See also \fBsvc_unreg()\fR in
156 \fBrpc_svc_calls\fR(3NSL).
157 .RE

159 .SH ATTRIBUTES
160 .sp
161 .LP
162 See \fBattributes\fR(5) for descriptions of the following attributes:
163 .sp

165 .sp
166 .TS
167 box;
168 c | c
169 l | l .
170 ATTRIBUTE TYPE ATTRIBUTE VALUE
171 _
172 MT-Level MT-Safe
173 .TE

175 .SH SEE ALSO
176 .sp
177 .LP
178 \fBrpcbind\fR(1M), \fBrpcinfo\fR(1M), \fBrpc_clnt_calls\fR(3NSL),
179 \fBrpc_clnt_create\fR(3NSL), \fBrpc_svc_calls\fR(3NSL), \fBattributes\fR(5)

```

\*\*\*\*\*

12964 Sun Jan 5 00:17:41 2014

new/usr/src/man/man3nsl/t\_bind.3nsl

4344 Minor typos in the 3nsl man pages

\*\*\*\*\*

```

1  \" te
2  .\" Copyright 1994, The X/Open Company Ltd., All Rights Reserved. Portions Copy
3  .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
4  .\" http://www.opengroup.org/bookstore/.
5  .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
6  .\" This notice shall appear on any product containing this material.
7  .\" The contents of this file are subject to the terms of the Common Development
8  .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
9  .\" When distributing Covered Code, include this CDDL HEADER in each file and in
10 .TH T_BIND 3NSL "Dec 27, 2013"
10 .TH T_BIND 3NSL "May 7, 1998"
11 .SH NAME
12 t_bind \- bind an address to a transport endpoint
13 .SH SYNOPSIS
14 .LP
15 .nf
16 #include <xti.h>

21 \fBint\fR \fBt_bind\fR(\fBint\fR \fBifd\fR, \fBconst struct t_bind *\fR\fBireq\fR,
22 .fi

24 .SH DESCRIPTION
25 .sp
26 .LP
27 This routine is part of the \fBXTI\fR interfaces that evolved from the
28 \fBTLI\fR interfaces. \fBXTI\fR represents the future evolution of these
29 interfaces. However, \fBTLI\fR interfaces are supported for compatibility. When
30 using a \fBTLI\fR routine that has the same name as an \fBXTI\fR routine, the
31 \fBtiuser.h\fR header file must be used. Refer to the \fBTLI\fR
32 \fBtiuser.h\fR header file must be used. Refer to the \fBTLI\fR
33 \fBCOMPATIBILITY\fR section for a description of differences between the two
34 interfaces.
35 .sp
36 .LP
37 This function associates a protocol address with the transport endpoint
38 specified by \fBifd\fR and activates that transport endpoint. In connection
39 mode, the transport provider may begin enqueueing incoming connect indications,
40 or servicing a connection request on the transport endpoint. In
41 connectionless-mode, the transport user may send or receive data units through
42 the transport endpoint.
43 .sp
44 .LP
45 The \fBireq\fR and \fBiret\fR arguments point to a \fBt_bind\fR structure
46 containing the following members:
47 .sp
48 .in +2
49 .nf
50 struct netbuf  addr;
51 unsigned      qlen;
52 .fi
53 .in -2
54 .sp
55 .LP
56 The \fBifd\fR field of the \fBt_bind\fR structure specifies a protocol
57 address, and the \fBqlen\fR field is used to indicate the maximum number of
58 outstanding connection indications.
59 .sp

```

```

60 .LP
61 The parameter \fBireq\fR is used to request that an address, represented by the
62 \fBnetbuf\fR structure, be bound to the given transport endpoint. The parameter
63 \fBflen\fR specifies the number of bytes in the address, and \fBibuf\fR points to
64 the address buffer. The parameter \fBimaxlen\fR has no meaning for the \fBireq\fR
65 argument. On return, \fBiret\fR contains an encoding for the address that the
66 transport provider actually bound to the transport endpoint; if an address was
67 specified in \fBireq\fR, this will be an encoding of the same address. In
68 \fBiret\fR, the user specifies \fBimaxlen\fR which is the maximum size of the
69 address buffer, and \fBibuf\fR which points to the buffer where the address is
70 to be placed. On return, \fBflen\fR specifies the number of bytes in the bound
71 address, and \fBibuf\fR points to the bound address. If \fBimaxlen\fR equals
72 zero, no address is returned. If \fBimaxlen\fR is greater than zero and less
73 than the length of the address, \fBt_bind()\fR fails with \fBt_errno\fR set to
74 \fBTBUFOVFLW\fR.
75 .sp
76 .LP
77 If the requested address is not available, \fBt_bind()\fR will return -1 with
78 \fBt_errno\fR set as appropriate. If no address is specified in \fBireq\fR (the
79 \fBflen\fR field of \fBifd\fR in \fBireq\fR is zero or \fBireq\fR is
80 \fBNULL),\fR the transport provider will assign an appropriate address to be
81 bound, and will return that address in the \fBifd\fR field of \fBiret\fR. If
82 the transport provider could not allocate an address, \fBt_bind()\fR will fail
83 with \fBt_errno\fR set to \fBTNOADDR\fR.
84 .sp
85 .LP
86 The parameter \fBireq\fR may be a null pointer if the user does not wish to
87 specify an address to be bound. Here, the value of \fBqlen\fR is assumed to be
88 zero, and the transport provider will assign an address to the transport
89 endpoint. Similarly, \fBiret\fR may be a null pointer if the user does not care
90 what address was bound by the provider and is not interested in the negotiated
91 value of \fBqlen\fR. It is valid to set \fBireq\fR and \fBiret\fR to the null
92 pointer for the same call, in which case the provider chooses the address to
93 bind to the transport endpoint and does not return that information to the
94 user.
95 .sp
96 .LP
97 The \fBqlen\fR field has meaning only when initializing a connection-mode
98 service. It specifies the number of outstanding connection indications that the
99 transport provider should support for the given transport endpoint. An
100 outstanding connection indication is one that has been passed to the transport
101 user by the transport provider but which has not been accepted or rejected. A
102 value of \fBqlen\fR greater than zero is only meaningful when issued by a
103 passive transport user that expects other users to call it. The value of
104 \fBqlen\fR will be negotiated by the transport provider and may be changed if
105 the transport provider cannot support the specified number of outstanding
106 connection indications. However, this value of \fBqlen\fR will never be
107 negotiated from a requested value greater than zero to zero. This is a
108 requirement on transport providers; see \fBWARNINGS\fR below. On return, the
109 \fBqlen\fR field in \fBiret\fR will contain the negotiated value.
110 .sp
111 .LP
112 If \fBifd\fR refers to a connection-mode service, this function allows more than
113 one transport endpoint to be bound to the same protocol address. But it is not
114 possible to bind more than one protocol address to the same transport endpoint.
115 However, the transport provider must also support this capability. If a user
116 binds more than one transport endpoint to the same protocol address, only one
117 endpoint can be used to listen for connection indications associated with that
118 protocol address. In other words, only one \fBt_bind()\fR for a given protocol
119 address may specify a value of \fBqlen\fR greater than zero. In this way, the
120 transport provider can identify which transport endpoint should be notified of
121 an incoming connection indication. If a user attempts to bind a protocol
122 address to a second transport endpoint with a value of \fBqlen\fR greater than
123 zero, \fBt_bind()\fR will return -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR.
124 When a user accepts a connection on the transport endpoint that is being used

```

```

125 as the listening endpoint, the bound protocol address will be found to be busy
126 for the duration of the connection, until a \fBt_unbind\fR(3NSL) or
127 \fBt_close\fR(3NSL) call has been issued. No other transport endpoints may be
128 bound for listening on that same protocol address while that initial listening
129 endpoint is active (in the data transfer phase or in the \fBT_IDLE\fR state).
130 This will prevent more than one transport endpoint bound to the same protocol
131 address from accepting connection indications.
132 .sp
133 .LP
134 If \fIfd\fR refers to connectionless mode service, this function allows for
135 more than one transport endpoint to be associated with a protocol address,
136 where the underlying transport provider supports this capability (often in
137 conjunction with value of a protocol-specific option). If a user attempts to
138 bind a second transport endpoint to an already bound protocol address when such
139 capability is not supported for a transport provider, \fBt_bind()\fR will
140 return -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR.
141 .SH RETURN VALUES
142 .sp
143 .LP
144 Upon successful completion, a value of 0 is returned. Otherwise, a value of
145 -1 is returned and \fBt_errno\fR is set to indicate an error.
146 .SH VALID STATES
147 .sp
148 .LP
149 \fBT_UNBND\fR
150 .SH ERRORS
151 .sp
152 .LP
153 On failure, \fBt_errno\fR is set to one of the following:
154 .sp
155 .ne 2
156 .na
157 \fB\FBTACCES\fR
158 .ad
159 .RS 13n
160 The user does not have permission to use the specified address.
161 .RE
162 .sp
163 .ne 2
164 .na
165 \fB\FBTADDRBUSY\fR
166 .ad
167 .RS 13n
168 The requested address is in use.
169 .RE
170 .RE
171 .sp
172 .ne 2
173 .na
174 \fB\FBTBADADDR\fR
175 .ad
176 .RS 13n
177 The specified protocol address was in an incorrect format or contained illegal
178 information.
179 .RE
180 .RE
181 .sp
182 .ne 2
183 .na
184 \fB\FBTBADF\fR
185 .ad
186 .RS 13n
187 The specified file descriptor does not refer to a transport endpoint.
188 .RE
189 .RE

```

```

191 .sp
192 .ne 2
193 .na
194 \fB\FBTBUFOVFLW\fR
195 .ad
196 .RS 13n
197 The number of bytes allowed for an incoming argument \fI(maxlen)\fR is greater
198 than 0 but not sufficient to store the value of that argument. The provider's
199 state will change to \fBT_IDLE\fR and the information to be returned in
200 \fIret\fR will be discarded.
201 .RE
202 .sp
203 .ne 2
204 .na
205 \fB\FBTOUTSTATE\fR
206 .ad
207 .RS 13n
208 The communications endpoint referenced by \fIfd\fR is not in one of the states
209 in which a call to this function is valid.
210 .RE
211 .RE
212 .sp
213 .ne 2
214 .na
215 \fB\FBTNOADDR\fR
216 .ad
217 .RS 13n
218 The transport provider could not allocate an address.
219 .RE
220 .RE
221 .sp
222 .ne 2
223 .na
224 \fB\FBTPROTO\fR
225 .ad
226 .RS 13n
227 This error indicates that a communication problem has been detected between XTI
228 and the transport provider for which there is no other suitable XTI error
229 \fB(t_errno)\fR.
230 .RE
231 .RE
232 .sp
233 .ne 2
234 .na
235 \fB\FBTSYSERR\fR
236 .ad
237 .RS 13n
238 A system error has occurred during execution of this function.
239 .RE
240 .RE
241 .SH TLI COMPATIBILITY
242 .sp
243 .LP
244 The \fBXTI\fR and \fBTLI\fR interface definitions have common names but use
245 different header files. This, and other semantic differences between the two
246 interfaces are described in the subsections below.
247 .SS "Interface Header"
248 .sp
249 .LP
250 The \fBXTI\fR interfaces use the header file, \fBxti.h\fR. \fBTLI\fR interfaces
251 should \fInot\fR use this header. They should use the header:
252 .sp
253 .LP
254 \fB#include\fR \fB<tiuser.h>\fR
255 .SS "Address Bound"
256 .sp

```

```

257 .sp
258 .LP
259 The user can compare the addresses in \fIreq\fR and \fIret\fR to determine
260 whether the transport provider bound the transport endpoint to a different
261 address than that requested.
262 .SS "Error Description Values"
263 .sp
264 .LP
265 The \fBt_errno\fR values \fBTPROTO\fR and \fBTADDRBUSY\fR can be set by the
266 \fBXTI\fR interface but cannot be set by the \fBTLI\fR interface.
267 .sp
268 .LP
269 A \fBt_errno\fR value that this routine can return under different
270 circumstances than its \fBXTI\fR counterpart is \fBTBUFOVFLW\fR. It can be
271 returned even when the \fBmaxlen\fR field of the corresponding buffer has been
272 set to zero.
273 .SH ATTRIBUTES
274 .sp
275 .LP
276 See \fBattributes\fR(5) for descriptions of the following attributes:
277 .sp

279 .sp
280 .TS
281 box;
282 c | c
283 l | l .
284 ATTRIBUTE TYPE ATTRIBUTE VALUE
285 _
286 MT Level Safe
287 .TE

289 .SH SEE ALSO
290 .sp
291 .LP
292 \fBaccept\fR(3NSL), \fBalloc\fR(3NSL), \fBclose\fR(3NSL),
293 \fBconnect\fR(3NSL), \fBunbind\fR(3NSL), \fBattributes\fR(5)
294 .SH WARNINGS
295 .sp
296 .LP
297 The requirement that the value of \fIqlen\fR never be negotiated from a
298 requested value greater than zero to zero implies that transport providers,
299 rather than the XTI implementation itself, accept this restriction.
300 .sp
301 .LP
302 An implementation need not allow an application explicitly to bind more than
303 one communications endpoint to a single protocol address, while permitting more
304 than one connection to be accepted to the same protocol address. That means
305 that although an attempt to bind a communications endpoint to some address with
306 \fIqlen=0\fR might be rejected with \fBTADDRBUSY\fR, the user may nevertheless
307 use this (unbound) endpoint as a responding endpoint in a call to
308 \fBaccept\fR(3NSL). To become independent of such implementation differences,
309 the user should supply unbound responding endpoints to \fBaccept\fR(3NSL).
310 .sp
311 .LP
312 The local address bound to an endpoint may change as result of a
313 \fBaccept\fR(3NSL) or \fBconnect\fR(3NSL) call. Such changes are not
314 necessarily reversed when the connection is released.

```