```
***********************************************************
   15073 Thu Dec 19 21:45:34 2013
new/usr/src/man/man1/rpcgen.1
4329 rpcgen(1): Four output files are generated by default
***********************************************************
   1 '\" te
   2 .\" Copyright (C) 2009, Sun Microsystems, Inc. All Rights Reserved
   3 .\" Copyright 1989 AT&T
   4 .\" The contents of this file are subject to the terms of the Common Development
   5 .\"  See the License for the specific language governing permissions and limitat
   6 .\" fields enclosed by brackets "[]" replaced with your own identifying informat
   7 .TH RPCGEN 1 "Dec 16, 2013"
   7 .TH RPCGEN 1 "Aug 24, 2009"
   8 .SH NAME
   9 rpcgen \- an RPC protocol compiler
  10 .SH SYNOPSIS
  11 .LP
  12 .nf
  13 \fBrpcgen\fR \fIinfile\fR
  14 .fi

  16 .LP
  17 .nf
  18 \fBrpcgen\fR [\fB-a\fR] [\fB-A\fR] [\fB-b\fR] [\fB-C\fR] [\fB-D\fR \fIname\fR [=
  19     [\fB-I\fR [\fB-K\fR \fIseconds\fR]] [\fB-L\fR] [\fB-M\fR] [\fB-N\fR] [\fB-
  20     [\fB-Y\fR \fIpathname\fR] \fIinfile\fR
  21 .fi

  23 .LP
  24 .nf
  25 \fBrpcgen\fR [\fB-c\fR | \fB-h\fR | \fB-l\fR | \fB-m\fR | \fB-t\fR | \fB-Sc\fR |
  26     [\fB-o\fR \fIoutfile\fR] [\fIinfile\fR]
  27 .fi

  29 .LP
  30 .nf
  31 \fBrpcgen\fR [\fB-s\fR \fInettype\fR] [\fB-o\fR \fIoutfile\fR] [\fIinfile\fR]
  32 .fi

  34 .LP
  35 .nf
  36 \fBrpcgen\fR [\fB-n\fR \fInetid\fR] [\fB-o\fR \fIoutfile\fR] [\fIinfile\fR]
  37 .fi

  39 .SH DESCRIPTION
  40 .sp
  41 .LP
  42 The \fBrpcgen\fR utility is a tool that generates C code to implement an
  43 \fBRPC\fR protocol. The input to \fBrpcgen\fR is a language similar to C known
  44 as \fBRPC\fR Language (Remote Procedure Call Language).
  45 .sp
  46 .LP
  47 The \fBrpcgen\fR utility is normally used as in the first synopsis where it
  48 takes an input file and generates four output files. If the \fIinfile\fR is
  48 takes an input file and generates three output files. If the \fIinfile\fR is
  49 named \fBproto.x\fR, then \fBrpcgen\fR generates a header in \fBproto.h\fR,
  50 \fBXDR\fR routines in \fBproto_xdr.c\fR, server-side stubs in
  51 \fBproto_svc.c\fR, and client-side stubs in \fBproto_clnt.c\fR. With the
  52 \fB-T\fR option, it also generates the \fBRPC\fR dispatch table in
  53 \fBproto_tbl.i\fR.
  54 .sp
  55 .LP
  56 \fBrpcgen\fR can also generate sample client and server files that can be
  57 customized to suit a particular application. The \fB-Sc\fR, \fB-Ss\fR, and
  58 \fB-Sm\fR options generate sample client, server and makefile, respectively.
  59 The \fB-a\fR option generates all files, including sample files. If the infile
```

```
  60 is \fBproto.x\fR, then the client side sample file is written to
  61 \fBproto_client.c\fR, the server side sample file to \fBproto_server.c\fR and
  62 the sample makefile to \fBmakefile.proto\fR.
  63 .sp
  64 .LP
  65 The server created can be started both by the port monitors (for example,
  66 \fBinetd\fR or \fBlisten\fR) or by itself. When it is started by a port
  67 monitor, it creates servers only for the transport for which the file
  68 descriptor \fB0\fR was passed. The name of the transport must be specified by
  69 setting up the environment variable \fBPM_TRANSPORT\fR. When the server
  70 generated by \fBrpcgen\fR is executed, it creates server handles for all the
  71 transports specified in the \fBNETPATH\fR environment variable, or if it is
  72 unset, it creates server handles for all the visible transports from the
  73 \fB/etc/netconfig\fR file. Note: the transports are chosen at run time and not
  74 at compile time. When the server is self-started, it backgrounds itself by
  75 default. A special define symbol \fBRPC_SVC_FG\fR can be used to run the server
  76 process in foreground.
  77 .sp
  78 .LP
  79 The second synopsis provides special features which allow for the creation of
  80 more sophisticated \fBRPC\fR servers. These features include support for
  81 user-provided \fB#defines\fR and \fBRPC\fR dispatch tables. The entries in the
  82 \fBRPC\fR dispatch table contain:
  83 .RS +4
  84 .TP
  85 .ie t \(bu
  86 .el o
  87 pointers to the service routine corresponding to that procedure
  88 .RE
  89 .RS +4
  90 .TP
  91 .ie t \(bu
  92 .el o
  93 a pointer to the input and output arguments
  94 .RE
  95 .RS +4
  96 .TP
  97 .ie t \(bu
  98 .el o
  99 the size of these routines
 100 .RE
 101 .sp
 102 .LP
 103 A server can use the dispatch table to check authorization and then to execute
 104 the service routine. A client library can use the dispatch table to deal with
 105 the details of storage management and \fBXDR\fR data conversion.
 106 .sp
 107 .LP
 108 The other three synopses shown above are used when one does not want to
 109 generate all the output files, but only a particular one. See the EXAMPLES
 110 section below for examples of \fBrpcgen\fR usage. When \fBrpcgen\fR is executed
 111 with the \fB-s\fR option, it creates servers for that particular class of
 112 transports. When executed with the \fB-n\fR option, it creates a server for the
 113 transport specified by \fInetid\fR. If \fIinfile\fR is not specified,
 114 \fBrpcgen\fR accepts the standard input.
 115 .sp
 116 .LP
 117 All the options mentioned in the second synopsis can be used with the other
 118 three synopses, but the changes are made only to the specified output file.
 119 .sp
 120 .LP
 121 The C preprocessor \fBcc\fR \fB-E\fR is run on the input file before it is
 122 actually interpreted by \fBrpcgen\fR. For each type of output file,
 123 \fBrpcgen\fR defines a special preprocessor symbol for use by the \fBrpcgen\fR
 124 programmer:
 125 .sp
```

```
 126 .ne 2
 127 .na
 128 \fB\fBRPC_HDR\fR\fR
 129 .ad
 130 .RS 12n
 131 defined when compiling into headers
 132 .RE

 134 .sp
 135 .ne 2
 136 .na
 137 \fB\fBRPC_XDR\fR\fR
 138 .ad
 139 .RS 12n
 140 defined when compiling into \fBXDR\fR routines
 141 .RE

 143 .sp
 144 .ne 2
 145 .na
 146 \fB\fBRPC_SVC\fR\fR
 147 .ad
 148 .RS 12n
 149 defined when compiling into server-side stubs
 150 .RE

 152 .sp
 153 .ne 2
 154 .na
 155 \fB\fBRPC_CLNT\fR\fR
 156 .ad
 157 .RS 12n
 158 defined when compiling into client-side stubs
 159 .RE

 161 .sp
 162 .ne 2
 163 .na
 164 \fB\fBRPC_TBL\fR\fR
 165 .ad
 166 .RS 12n
 167 defined when compiling into \fBRPC\fR dispatch tables
 168 .RE

 170 .sp
 171 .LP
 172 Any line beginning with ''\fB%\fR'' is passed directly into the output file,
 173 uninterpreted by \fBrpcgen\fR, except that the leading ''\fB%\fR" is stripped
 174 off. To specify the path name of the C preprocessor, use the \fB-Y\fR flag.
 175 .sp
 176 .LP
 177 For every data type referred to in \fIinfile\fR, \fBrpcgen\fR assumes that
 178 there exists a routine with the string \fBxdr_\fR prepended to the name of the
 179 data type. If this routine does not exist in the \fBRPC\fR/\fBXDR\fR library,
 180 it must be provided. Providing an undefined data type allows customization of
 181 \fBXDR\fR routines.
 182 .SS "Server Error Reporting"
 183 .sp
 184 .LP
 185 By default, errors detected by \fBproto_svc.c\fR is reported to standard error
 186 and/or the system log.
 187 .sp
 188 .LP
 189 This behavior can be overridden by compiling the file with a definition of
 190 \fBRPC_MSGOUT\fR, for example, \fB-DRPC_MSGOUT=mymsgfunc\fR. The function
 191 specified is called to report errors. It must conform to the following
```

```
 192 \fBprintf\fR-like signature:
 193 .sp
 194 .in +2
 195 .nf
 196 extern void RPC_MSGOUT(const char *fmt, ...);
 197 .fi
 198 .in -2
 199 .sp

 201 .SH OPTIONS
 202 .sp
 203 .LP
 204 The following options are supported:
 205 .sp
 206 .ne 2
 207 .na
 208 \fB\fB-a\fR\fR
 209 .ad
 210 .RS 18n
 211 Generates all files, including sample files.
 212 .RE

 214 .sp
 215 .ne 2
 216 .na
 217 \fB\fB-A\fR\fR
 218 .ad
 219 .RS 18n
 220 Enables the Automatic \fBMT\fR mode in the server main program. In this mode,
 221 the \fBRPC\fR library automatically creates threads to service client requests.
 222 This option generates multithread-safe stubs by implicitly turning on the
 223 \fB-M\fR option. Server multithreading modes and parameters can be set using
 224 the \fBrpc_control\fR(3NSL) call. \fBrpcgen\fR generated code does not change
 225 the default values for the Automatic \fBMT\fR mode.
 226 .RE

 228 .sp
 229 .ne 2
 230 .na
 231 \fB\fB-b\fR\fR
 232 .ad
 233 .RS 18n
 234 Backward compatibility mode. Generates transport-specific \fBRPC\fR code for
 235 older versions of the operating system.
 236 .RE

 238 .sp
 239 .ne 2
 240 .na
 241 \fB\fB-c\fR\fR
 242 .ad
 243 .RS 18n
 244 Compiles into \fBXDR\fR routines.
 245 .RE

 247 .sp
 248 .ne 2
 249 .na
 250 \fB\fB-C\fR\fR
 251 .ad
 252 .RS 18n
 253 Generates header and stub files which can be used with ANSI C compilers.
 254 Headers generated with this flag can also be used with C++ programs.
 255 .RE

 257 .sp
```

```
258 .ne 2
259 .na
260 \fB\fB-D\fR\fIname\fR\fB[=\fR\fIvalue\fR\fB]\fR\fR
261 .ad
262 .RS 18n
263 Defines a symbol \fIname\fR. Equivalent to the \fB#define\fR directive in the
264 source. If no \fIvalue\fR is given, \fIvalue\fR is defined as \fB1\fR. This
265 option can be specified more than once.
266 .RE

268 .sp
269 .ne 2
270 .na
271 \fB\fB-h\fR\fR
272 .ad
273 .RS 18n
274 Compiles into \fBC\fR data-definitions (a header). The \fB-T\fR option can be
275 used in conjunction to produce a header which supports \fBRPC\fR dispatch
276 tables.
277 .RE

279 .sp
280 .ne 2
281 .na
282 \fB\fB-i\fR \fIsize\fR\fR
283 .ad
284 .RS 18n
285 Size at which to start generating inline code. This option is useful for
286 optimization. The default \fIsize\fR is 5.
287 .RE

289 .sp
290 .ne 2
291 .na
292 \fB\fB-I\fR\fR
293 .ad
294 .RS 18n
295 Compiles support for \fBinetd\fR(1M) in the server side stubs. Such servers can
296 be self-started or can be started by \fBinetd\fR. When the server is
297 self-started, it backgrounds itself by default. A special define symbol
298 \fBRPC_SVC_FG\fR can be used to run the server process in foreground, or the
299 user can simply compile without the \fB-I\fR option.
300 .sp
301 If there are no pending client requests, the \fBinetd\fR servers exit after 120
302 seconds (default). The default can be changed with the \fB-K\fR option. All of
303 the error messages for \fBinetd\fR servers are always logged with
304 \fBsyslog\fR(3C).
305 .sp
306 \fBNote:\fR This option is supported for backward compatibility only. It should
307 always be used in conjunction with the \fB-b\fR option which generates backward
308 compatibility code. By default (that is, when \fB-b\fR is not specified),
309 \fBrpcgen\fR generates servers that can be invoked through portmonitors.
310 .RE

312 .sp
313 .ne 2
314 .na
315 \fB\fB-K\fR \fIseconds\fR\fR
316 .ad
317 .RS 18n
318 By default, services created using \fBrpcgen\fR and invoked through port
319 monitors wait 120 seconds after servicing a request before exiting. That
320 interval can be changed using the \fB-K\fR flag. To create a server that exits
321 immediately upon servicing a request, use \fB-K\fR \fB0\fR. To create a server
322 that never exits, the appropriate argument is \fB-K\fR \fB\(mi1\fR\&.
323 .sp
```

```
324 When monitoring for a server, some portmonitors, like \fBlisten\fR(1M),
325 \fBalways\fR spawn a new process in response to a service request. If it is
326 known that a server are used with such a monitor, the server should exit
327 immediately on completion. For such servers, \fBrpcgen\fR should be used with
328 \fB-K\fR \fB0\fR.
329 .RE

331 .sp
332 .ne 2
333 .na
334 \fB\fB-l\fR\fR
335 .ad
336 .RS 18n
337 Compiles into client-side stubs.
338 .RE

340 .sp
341 .ne 2
342 .na
343 \fB\fB-L\fR\fR
344 .ad
345 .RS 18n
346 When the servers are started in foreground, uses \fBsyslog\fR(3C) to log the
347 server errors instead of printing them on the standard error.
348 .RE

350 .sp
351 .ne 2
352 .na
353 \fB\fB-m\fR\fR
354 .ad
355 .RS 18n
356 Compiles into server-side stubs, but do not generate a "main" routine. This
357 option is useful for doing callback-routines and for users who need to write
358 their own "main" routine to do initialization.
359 .RE

361 .sp
362 .ne 2
363 .na
364 \fB\fB-M\fR\fR
365 .ad
366 .RS 18n
367 Generates multithread-safe stubs for passing arguments and results between
368 \fBrpcgen\fR-generated code and user written code. This option is useful for
369 users who want to use threads in their code.
370 .RE

372 .sp
373 .ne 2
374 .na
375 \fB\fB-N\fR\fR
376 .ad
377 .RS 18n
378 This option allows procedures to have multiple arguments. It also uses the
379 style of parameter passing that closely resembles C. So, when passing an
380 argument to a remote procedure, you do not have to pass a pointer to the
381 argument, but can pass the argument itself. This behavior is different from the
382 old style of \fBrpcgen\fR-generated code. To maintain backward compatibility,
383 this option is not the default.
384 .RE

386 .sp
387 .ne 2
388 .na
389 \fB\fB-n\fR \fInetid\fR\fR
```

```
 390 .ad
 391 .RS 18n
 392 Compiles into server-side stubs for the transport specified by \fInetid\fR.
 393 There should be an entry for \fInetid\fR in the \fBnetconfig\fR database. This
 394 option can be specified more than once, so as to compile a server that serves
 395 multiple transports.
 396 .RE

 398 .sp
 399 .ne 2
 400 .na
 401 \fB\fB-o\fR \fIoutfile\fR\fR
 402 .ad
 403 .RS 18n
 404 Specifies the name of the output file. If none is specified, standard output is
 405 used (\fB-c\fR, \fB-h\fR, \fB-l\fR, \fB-m\fR, \fB-n\fR, \fB-s\fR, \fB-Sc\fR,
 406 \fB-Sm\fR, \fB-Ss\fR, and \fB-t\fR modes only).
 407 .RE

 409 .sp
 410 .ne 2
 411 .na
 412 \fB\fB-s\fR \fInettype\fR\fR
 413 .ad
 414 .RS 18n
 415 Compiles into server-side stubs for all the transports belonging to the class
 416 \fInettype\fR. The supported classes are \fBnetpath\fR, \fBvisible\fR,
 417 \fBcircuit_n\fR, \fBcircuit_v\fR, \fBdatagram_n\fR, \fBdatagram_v\fR,
 418 \fBtcp\fR, and \fBudp\fR (see \fBrpc\fR(3NSL) for the meanings associated with
 419 these classes). This option can be specified more than once. \fBNote:\fR The
 420 transports are chosen at run time and not at compile time.
 421 .RE
 423 .sp
 424 .ne 2
 425 .na
 426 \fB\fB-Sc\fR\fR
 427 .ad
 428 .RS 18n
 429 Generates sample client code that uses remote procedure calls.
 430 .RE

 432 .sp
 433 .ne 2
 434 .na
 435 \fB\fB-Sm\fR\fR
 436 .ad
 437 .RS 18n
 438 Generates a sample Makefile which can be used for compiling the application.
 439 .RE

 441 .sp
 442 .ne 2
 443 .na
 444 \fB\fB-Ss\fR\fR
 445 .ad
 446 .RS 18n
 447 Generates sample server code that uses remote procedure calls.
 448 .RE

 450 .sp
 451 .ne 2
 452 .na
 453 \fB\fB-t\fR\fR
 454 .ad
 455 .RS 18n
```

```
 456 Compiles into \fBRPC\fR dispatch table.
 457 .RE

 459 .sp
 460 .ne 2
 461 .na
 462 \fB\fB-T\fR\fR
 463 .ad
 464 .RS 18n
 465 Generates the code to support \fBRPC\fR dispatch tables.
 466 .sp
 467 The options \fB-c\fR, \fB-h\fR, \fB-l\fR, \fB-m\fR, \fB-s\fR, \fB-Sc\fR,
 468 \fB-Sm\fR, \fB-Ss\fR, and \fB-t\fR are used exclusively to generate a
 469 particular type of file, while the options \fB-D\fR and \fB-T\fR are global and
 470 can be used with the other options.
 471 .RE

 473 .sp
 474 .ne 2
 475 .na
 476 \fB\fB-v\fR\fR
 477 .ad
 478 .RS 18n
 479 Displays the version number.
 480 .RE

 482 .sp
 483 .ne 2
 484 .na
 485 \fB\fB-Y\fR \fIpathname\fR\fR
 486 .ad
 487 .RS 18n
 488 Gives the name of the directory where \fBrpcgen\fR starts looking for the C
 489 preprocessor.
 490 .RE

 492 .SH OPERANDS
 493 .sp
 494 .LP
 495 The following operand is supported:
 496 .sp
 497 .ne 2
 498 .na
 499 \fB\fIinfile\fR\fR
 500 .ad
 501 .RS 10n
 502 input file
 503 .RE

 505 .SH EXAMPLES
 506 .LP
 507 \fBExample 1 \fRGenerating the output files and dispatch table
 508 .sp
 509 .LP
 510 The following entry

 512 .sp
 513 .in +2
 514 .nf
 515 example% \fBrpcgen -T prot.x\fR
 516 .fi
 517 .in -2
 518 .sp

 520 .sp
 521 .LP
```

```
 522 generates all the five files: \fBprot.h\fR, \fBprot_clnt.c\fR,
 523 \fBprot_svc.c\fR, \fBprot_xdr.c\fR, and \fBprot_tbl.i\fR.

 525 .LP
 526 \fBExample 2 \fRSending headers to standard output
 527 .sp
 528 .LP
 529 The following example sends the C data-definitions (header) to the standard
 530 output:

 532 .sp
 533 .in +2
 534 .nf
 535 example% \fBrpcgen -h prot.x\fR
 536 .fi
 537 .in -2
 538 .sp

 540 .LP
 541 \fBExample 3 \fRSending a test version
 542 .sp
 543 .LP
 544 To send the test version of the \fB-DTEST\fR, server side stubs for all the
 545 transport belonging to the class \fBdatagram_n\fR to standard output, use:

 547 .sp
 548 .in +2
 549 .nf
 550 example% \fBrpcgen -s datagram_n -DTEST prot.x\fR
 551 .fi
 552 .in -2
 553 .sp

 555 .LP
 556 \fBExample 4 \fRCreating server side stubs
 557 .sp
 558 .LP
 559 To create the server side stubs for the transport indicated by \fInetid\fR
 560 \fBtcp\fR, use:

 562 .sp
 563 .in +2
 564 .nf
 565 example% \fBrpcgen -n tcp -o prot_svc.c prot.x\fR
 566 .fi
 567 .in -2
 568 .sp

 570 .SH EXIT STATUS
 571 .sp
 572 .ne 2
 573 .na
 574 \fB\fB0\fR\fR
 575 .ad
 576 .RS 6n
 577 Successful operation.
 578 .RE

 580 .sp
 581 .ne 2
 582 .na
 583 \fB\fB>0\fR\fR
 584 .ad
 585 .RS 6n
 586 An error occurred.
 587 .RE
```

```
 589 .SH SEE ALSO
 590 .sp
 591 .LP
 592 \fBinetd\fR(1M), \fBlisten\fR(1M), \fBrpc\fR(3NSL), \fBrpc_control\fR(3NSL),
 593 \fBrpc_svc_calls\fR(3NSL), \fBsyslog\fR(3C), \fBnetconfig\fR(4),
 594 \fBattributes\fR(5)
 595 .sp
 596 .LP
 597 The \fBrpcgen\fR chapter in the \fIONC+ Developer\&'s Guide\fR manual.
```

```
*********************************************************
    6960 Thu Dec 19 21:45:34 2013
new/usr/src/man/man3c/rwlock.3c
4327 rwlock(3c): Formatting issues and typos
*********************************************************
    1 '\" te
    2 .\"  Copyright (c) 1998 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 RWLOCK 3C "Dec 19, 2013"
    6 .TH RWLOCK 3C "May 14, 1998"
    7 .SH NAME
    8 rwlock, rwlock_init, rwlock_destroy, rw_rdlock, rw_wrlock, rw_tryrdlock,
    9 rw_trywrlock, rw_unlock \- multiple readers, single writer locks
   10 .SH SYNOPSIS
   11 .LP
   12 .nf
   13 cc -mt [ \fIflag\fR... ] \fIfile\fR...[ \fIlibrary\fR... ]

   15 #include <synch.h>

   17 \fBint\fR \fBrwlock_init\fR(\fBrwlock_t *\fR\fIrwlp\fR, \fBint\fR \fItype\fR, \f
   17 \fBint\fR \fBrwlock_init\fR(\fBrwlock_t *\fR\fIrwlp\fR, \fBint\fR \fItype\fR, \f
   18 .fi

   20 .LP
   21 .nf
   22 \fBint\fR \fBrwlock_destroy\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   23 .fi

   25 .LP
   26 .nf
   27 \fBint\fR \fBrw_rdlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   28 .fi

   30 .LP
   31 .nf
   32 \fBint\fR \fBrw_wrlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   33 .fi

   35 .LP
   36 .nf
   37 \fBint\fR \fBrw_unlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   38 .fi

   40 .LP
   41 .nf
   42 \fBint\fR \fBrw_tryrdlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   43 .fi

   45 .LP
   46 .nf
   47 \fBint\fR \fBrw_trywrlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
   48 .fi

   50 .SH DESCRIPTION
   51 .sp
   52 .LP
   53 Many threads can have simultaneous read-only access to data, while only one
   54 thread can have write access at any given time. Multiple read access with
   55 single write access is controlled by locks, which are generally used to protect
   56 data that is frequently searched.
   57 .sp
   58 .LP
   59 Readers/writer locks can synchronize threads in this process and other
```

```
   60 processes if they are allocated in writable memory  and shared among
   61 cooperating processes (see \fBmmap\fR(2)), and are initialized for this
   62 purpose.
   63 .sp
   64 .LP
   65 Additionally, readers/writer locks must be initialized prior to use.
   66 The readers/writer lock pointed to by \fIrwlp\fR is
   66 \fBrwlock_init()\fR The readers/writer lock pointed to by \fIrwlp\fR is
   67 initialized by  \fBrwlock_init()\fR. A readers/writer lock is capable of having
   68 several types of behavior, which is specified by \fItype\fR. \fIarg\fR is
   68 several types of behavior, which is specified by \fBtype\fR. \fIarg\fR is
   69 currently not used, although a future type may define  new behavior parameters
   70 by way of  \fIarg\fR.
   71 .sp
   72 .LP
   73 The \fItype\fR argument can be one of the following:
   74 .sp
   75 .ne 2
   76 .na
   77 \fB\fBUSYNC_PROCESS\fR \fR
   78 .ad
   79 .RS 18n
   80 The readers/writer lock can synchronize threads in this process and other
   81 processes. The readers/writer lock should be initialized by only one process.
   82 \fIarg\fR is ignored. A readers/writer lock initialized with this type, must be
   83 allocated in memory shared between processses, i.e. either in Sys V shared
   84 memory (see \fBshmop\fR(2)) or in memory mapped to a file (see \fBmmap\fR(2)).
   85 It is illegal to initialize the object this way and to not allocate it in such
   86 shared memory.
   87 .RE

   89 .sp
   90 .ne 2
   91 .na
   92 \fB\fBUSYNC_THREAD\fR \fR
   93 .ad
   94 .RS 18n
   95 The readers/writer lock can synchronize  threads in this process, only.
   96 \fIarg\fR is ignored.
   97 .RE

   99 .sp
  100 .LP
  101 Additionally, readers/writer locks can be initialized by allocation  in zeroed
  102 memory. A \fItype\fR of \fBUSYNC_THREAD\fR is assumed in this case. Multiple
  102 memory. A \fBtype\fR of \fBUSYNC_THREAD\fR is assumed in this case. Multiple
  103 threads must not simultaneously initialize the same  readers/writer lock. And a
  104 readers/writer lock must not be re-initialized while in use by other threads.
  105 .sp
  106 .LP
  107 The following are default readers/writer lock initialization (intra-process):
  108 .sp
  109 .in +2
  110 .nf
  111 rwlock_t rwlp;
  112 rwlock_init(&rwlp, NULL, NULL);

  114 .fi
  115 .in -2

  117 .sp
  118 .LP
  119 or
  120 .sp
  121 .in +2
  122 .nf
```

```
 123 rwlock_init(&rwlp, USYNC_THREAD, NULL);
 124 .fi
 125 .in -2

 127 .sp
 128 .LP
 129 or
 130 .sp
 131 .in +2
 132 .nf
 133 rwlock_t  rwlp  =  DEFAULTRWLOCK;
 134 .fi
 135 .in -2

 137 .sp
 138 .LP
 139 The following is a customized readers/writer lock  initialization
 140 (inter-process):
 141 .sp
 142 .in +2
 143 .nf
 144 rwlock_init(&rwlp, USYNC_PROCESS, NULL);
 145 .fi
 146 .in -2

 148 .sp
 149 .LP
 150 Any state associated with the readers/writer lock pointed to by \fIrwlp\fR are
 151 destroyed by  \fBrwlock_destroy()\fR and the readers/writer lock storage space
 152 is not released.
 153 .sp
 154 .LP
 155 \fBrw_rdlock()\fR gets a read lock on the readers/writer lock pointed to by
 156 \fIrwlp\fR. If the readers/writer lock is currently locked for writing, the
 157 calling thread blocks until the write lock is freed. Multiple threads may
 158 simultaneously hold a read lock on a  readers/writer lock.
 159 .sp
 160 .LP
 161 **\fBrw_tryrdlock()\fR tries to get a read lock on the readers/writer lock pointed**
 161 *\fBrw_tryrdlock()\fR trys to get a read lock on the readers/writer lock pointed*
 162 to by \fIrwlp\fR. If the readers/writer lock is locked for writing, it returns
 163 an error; otherwise, the read lock is acquired.
 164 .sp
 165 .LP
 166 \fBrw_wrlock()\fR gets a write lock on the readers/writer lock pointed to by
 167 \fIrwlp\fR. If the readers/writer lock is currently locked for reading or
 168 writing, the calling thread blocks until all the read and write locks are
 169 freed. At any given time, only one thread may have a write lock on a
 170 readers/writer lock.
 171 .sp
 172 .LP
 173 **\fBrw_trywrlock()\fR tries to get a write lock on the readers/writer lock**
 173 *\fBrw_trywrlock()\fR trys to get a write lock on the readers/writer lock*
 174 pointed to by \fIrwlp\fR. If the readers/writer lock is currently locked for
 175 reading or writing, it returns an error.
 176 .sp
 177 .LP
 178 \fBrw_unlock()\fR unlocks a readers/writer lock pointed to by  \fIrwlp\fR, if
 179 the readers/writer lock is locked and the calling thread holds the lock for
 180 either reading or writing. One of the other threads that is waiting for the
 181 **readers/writer  lock to be freed will be unblocked, provided there are other**
 181 *readers/writer  lock to be freed will be unblocked, provided there is other*
 182 waiting threads. If the calling thread does not hold the lock for either
 183 reading or writing, no error status is returned, and the program's behavior is
 184 unknown.
 185 .SH RETURN VALUES
```

```
 186 .sp
 187 .LP
 188 If successful, these functions return \fB0\fR. Otherwise, a non-zero value is
 189 returned to indicate the error.
 190 .SH ERRORS
 191 .sp
 192 .LP
 193 The \fBrwlock_init()\fR function will fail if:
 194 .sp
 195 .ne 2
 196 .na
 197 \fB\fBEINVAL\fR \fR
 198 .ad
 199 .RS 11n
 200 **\fItype\fR is invalid.**
 200 *\fBtype\fR is invalid.*
 201 .RE

 203 .sp
 204 .LP
 205 The  \fBrw_tryrdlock()\fR or \fBrw_trywrlock()\fR functions will fail if:
 206 .sp
 207 .ne 2
 208 .na
 209 \fB\fBEBUSY\fR \fR
 210 .ad
 211 .RS 10n
 212 **The readers/writer lock pointed to by \fIrwlp\fR was already locked.**
 212 *The reader or writer lock pointed to by \fIrwlp\fR was already locked.*
 213 .RE

 215 .sp
 216 .LP
 217 These functions may fail if:
 218 .sp
 219 .ne 2
 220 .na
 221 \fB\fBEFAULT\fR \fR
 222 .ad
 223 .RS 11n
 224 \fIrwlp\fR or \fIarg\fR points to an illegal address.
 225 .RE

 227 .SH ATTRIBUTES
 228 .sp
 229 .LP
 230 See \fBattributes\fR(5) for descriptions of the following attributes:
 231 .sp

 233 .sp
 234 .TS
 235 box;
 236 c | c
 237 l | l .
 238 ATTRIBUTE TYPE  ATTRIBUTE VALUE
 239 _
 240 MT-Level        MT-Safe
 241 .TE

 243 .SH SEE ALSO
 244 .sp
 245 .LP
 246 \fBmmap\fR(2), \fBattributes\fR(5)
 247 .SH NOTES
 248 .sp
 249 .LP
```

```
250 These interfaces also available by way of:
251 .sp
252 .LP
253 \fB#include\fR \fB<thread.h>\fR
254 .sp
255 .LP
256 If multiple threads are waiting for a readers/writer lock, the acquisition
257 order is random by default. However, some implementations may bias acquisition
258 order to avoid depriving writers. The current implementation favors writers
259 over readers.
```