

```

*****
12271 Wed Aug 3 14:06:32 2016
new/usr/src/man/man3c/rctlblk_set_value.3c
7264 Example code is rctlblk_set_value(3c) manpage does not compile.
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Toomas Soome <tsoome@me.com>
*****
1 \" te
2 .\" Copyright (c) 2006, 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 RCTLBLK_SET_VALUE 3C \"Aug 2, 2016\"
6 .TH RCTLBLK_SET_VALUE 3C \"May 15, 2006\"
7 .SH NAME
8 rctlblk_set_value, rctlblk_get_firing_time, rctlblk_get_global_action,
9 rctlblk_get_global_flags, rctlblk_get_local_action, rctlblk_get_local_flags,
10 rctlblk_get_privilege, rctlblk_get_recipient_pid, rctlblk_get_value,
11 rctlblk_get_enforced_value, rctlblk_set_local_action, rctlblk_set_local_flags,
12 rctlblk_set_privilege, rctlblk_set_recipient_pid, rctlblk_size \- manipulate
13 resource control blocks
14 .SH SYNOPSIS
15 .LP
16 .nf
17 #include <rctl.h>

19 \fBhrtime_t\fR \fBrcctlblk_get_firing_time\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
20 .fi

22 .LP
23 .nf
24 \fBint_t\fR \fBrcctlblk_get_global_action\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
25 .fi

27 .LP
28 .nf
29 \fBint_t\fR \fBrcctlblk_get_global_flags\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
30 .fi

32 .LP
33 .nf
34 \fBint_t\fR \fBrcctlblk_get_local_action\fR(\fBrcctlblk_t *\fR\fIrbk\fR, \fBint_t *\fR
35 .fi

37 .LP
38 .nf
39 \fBint_t\fR \fBrcctlblk_get_local_flags\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
40 .fi

42 .LP
43 .nf
44 \fBrcctl_priv_t\fR \fBrcctlblk_get_privilege\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
45 .fi

47 .LP
48 .nf
49 \fBbid_t\fR \fBrcctlblk_get_recipient_pid\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
50 .fi

52 .LP
53 .nf
54 \fBrcctl_qty_t\fR \fBrcctlblk_get_value\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
55 .fi

57 .LP
58 .nf

```

```

59 \fBrcctl_qty_t\fR \fBrcctlblk_get_enforced_value\fR(\fBrcctlblk_t *\fR\fIrbk\fR);
60 .fi

62 .LP
63 .nf
64 \fBvoid_t\fR \fBrcctlblk_set_local_action\fR(\fBrcctlblk_t *\fR\fIrbk\fR, \fBrcctl_a
65 \fBint_t\fR \fBisignal\fR);
66 .fi

68 .LP
69 .nf
70 \fBvoid_t\fR \fBrcctlblk_set_local_flags\fR(\fBrcctlblk_t *\fR\fIrbk\fR, \fBint_t\fR
71 .fi

73 .LP
74 .nf
75 \fBvoid_t\fR \fBrcctlblk_set_privilege\fR(\fBrcctlblk_t *\fR\fIrbk\fR, \fBrcctl_priv
76 .fi

78 .LP
79 .nf
80 \fBvoid_t\fR \fBrcctlblk_set_value\fR(\fBrcctlblk_t *\fR\fIrbk\fR, \fBrcctl_qty_t\fR
81 .fi

83 .LP
84 .nf
85 \fBvoid_t\fR \fBrcctlblk_set_recipient_pid\fR(\fBbid_t\fR\fIpid\fR);
86 .fi

88 .LP
89 .nf
90 \fBsize_t\fR \fBrcctlblk_size\fR(\fBvoid_t\fR);
91 .fi

93 .SH DESCRIPTION
94 .sp
94 .LP
95 The resource control block routines allow the establishment or retrieval of
96 values from a resource control block used to transfer information using the
97 \fBgctrctl\fR(2) and \fBsetrctl\fR(2) functions. Each of the routines accesses
98 or sets the resource control block member corresponding to its name. Certain
99 of these members are read-only and do not possess set routines.
100 .sp
101 .LP
102 The firing time of a resource control block is 0 if the resource control
103 action-value has not been exceeded for its lifetime on the process. Otherwise
104 the firing time is the value of \fBgethrtime\fR(3C) at the moment the action on
105 the resource control value was taken.
106 .sp
107 .LP
108 The global actions and flags are the action and flags set by \fBrcctladm\fR(1M).
109 These values cannot be set with \fBsetrctl\fR(2). Valid global actions are
110 listed in the table below. Global flags are generally a published property of
111 the control and are not modifiable.
112 .sp
113 .ne 2
114 .na
115 \fBFBRCCTL_GLOBAL_DENY_ALWAYS\fR
116 .ad
117 .RS 28n
118 The action taken when a control value is exceeded on this control will always
119 include denial of the resource.
120 .RE

122 .sp
123 .ne 2

```

```

124 .na
125 \fB\fBRCTL_GLOBAL_DENY_NEVER\fR\fR
126 .ad
127 .RS 28n
128 The action taken when a control value is exceeded on this control will always
129 exclude denial of the resource; the resource will always be granted, although
130 other actions can also be taken.
131 .RE

133 .sp
134 .ne 2
135 .na
136 \fB\fBRCTL_GLOBAL_SIGNAL_NEVER\fR\fR
137 .ad
138 .RS 28n
139 No signal actions are permitted on this control.
140 .RE

142 .sp
143 .ne 2
144 .na
145 \fB\fBRCTL_GLOBAL_CPU_TIME\fR\fR
146 .ad
147 .RS 28n
148 The valid signals available as local actions include the \fBSIGXCPU\fR signal.
149 .RE

151 .sp
152 .ne 2
153 .na
154 \fB\fBRCTL_GLOBAL_FILE_SIZE\fR\fR
155 .ad
156 .RS 28n
157 The valid signals available as local actions include the \fBSIGXFSZ\fR signal.
158 .RE

160 .sp
161 .ne 2
162 .na
163 \fB\fBRCTL_GLOBAL_INFINITE\fR\fR
164 .ad
165 .RS 28n
166 This resource control supports the concept of an unlimited value; generally
167 true only of accumulation-oriented resources, such as CPU time.
168 .RE

170 .sp
171 .ne 2
172 .na
173 \fB\fBRCTL_GLOBAL_LOWERABLE\fR\fR
174 .ad
175 .RS 28n
176 Non-privileged callers are able to lower the value of privileged resource
177 control values on this control.
178 .RE

180 .sp
181 .ne 2
182 .na
183 \fB\fBRCTL_GLOBAL_NOACTION\fR\fR
184 .ad
185 .RS 28n
186 No global action will be taken when a resource control value is exceeded on
187 this control.
188 .RE

```

```

190 .sp
191 .ne 2
192 .na
193 \fB\fBRCTL_GLOBAL_NOBASIC\fR\fR
194 .ad
195 .RS 28n
196 No values with the \fBRCPRIV_BASIC\fR privilege are permitted on this control.
197 .RE

199 .sp
200 .ne 2
201 .na
202 \fB\fBRCTL_GLOBAL_SYSLOG\fR\fR
203 .ad
204 .RS 28n
205 A standard message will be logged by the \fBsyslog\fR(3C) facility when any
206 resource control value on a sequence associated with this control is exceeded.
207 .RE

209 .sp
210 .ne 2
211 .na
212 \fB\fBRCTL_GLOBAL_SYSLOG_NEVER\fR\fR
213 .ad
214 .RS 28n
215 The resource control does not support the \fBsyslog()\fR global action.
216 Exceeding a resource control value on this control will not result in a message
217 logged by the \fBsyslog()\fR facility.
218 .RE

220 .sp
221 .ne 2
222 .na
223 \fB\fBRCTL_GLOBAL_UNOBSERVABLE\fR\fR
224 .ad
225 .RS 28n
226 The resource control (generally on a task- or project-related control) does not
227 support observational control values. An \fBRCPRIV_BASIC\fR privileged control
228 value placed by a process on the task or process will generate an action only
229 if the value is exceeded by that process.
230 .RE

232 .sp
233 .ne 2
234 .na
235 \fB\fBRCTL_GLOBAL_BYTES\fR\fR
236 .ad
237 .RS 28n
238 This resource control represents a number of bytes.
239 .RE

241 .sp
242 .ne 2
243 .na
244 \fB\fBRCTL_GLOBAL_SECONDS\fR\fR
245 .ad
246 .RS 28n
247 This resource control represents a quantity of time in seconds.
248 .RE

250 .sp
251 .ne 2
252 .na
253 \fB\fBRCTL_GLOBAL_COUNT\fR\fR
254 .ad
255 .RS 28n

```

```

256 This resource control represents an integer count.
257 .RE

259 .sp
260 .LP
261 The local action and flags are those on the current resource control value
262 represented by this resource control block. Valid actions and flags are listed
263 in the table below. In the case of \fBRCTL_LOCAL_SIGNAL\fR, the second argument
264 to \fBrctlblk_set_local_action()\fR contains the signal to be sent. Similarly,
265 the signal to be sent is copied into the integer location specified by the
266 second argument to \fBrctlblk_get_local_action()\fR. A restricted set of
267 signals is made available for normal use by the resource control facility:
268 \fBSIGBART\fR, \fBSIGXRES\fR, \fBSIGHUP\fR, \fBSIGSTOP\fR, \fBSIGTERM\fR, and
269 \fBSIGKILL\fR. Other signals are permitted due to global properties of a
270 specific control. Calls to \fBsetrctl()\fR with illegal signals will fail.
271 .sp
272 .ne 2
273 .na
274 \fB\fBRCTL_LOCAL_DENY\fR\fR
275 .ad
276 .RS 23n
277 When this resource control value is encountered, the request for the resource
278 will be denied. Set on all values if \fBRCTL_GLOBAL_DENY_ALWAYS\fR is set for
279 this control; cleared on all values if \fBRCTL_GLOBAL_DENY_NEVER\fR is set for
280 this control.
281 .RE

283 .sp
284 .ne 2
285 .na
286 \fB\fBRCTL_LOCAL_MAXIMAL\fR\fR
287 .ad
288 .RS 23n
289 This resource control value represents a request for the maximum amount of
290 resource for this control. If \fBRCTL_GLOBAL_INFINITY\fR is set for this
291 resource control, \fBRCTL_LOCAL_MAXIMAL\fR indicates an unlimited resource
292 control value, one that will never be exceeded.
293 .RE

295 .sp
296 .ne 2
297 .na
298 \fB\fBRCTL_LOCAL_NOACTION\fR\fR
299 .ad
300 .RS 23n
301 No local action will be taken when this resource control value is exceeded.
302 .RE

304 .sp
305 .ne 2
306 .na
307 \fB\fBRCTL_LOCAL_SIGNAL\fR\fR
308 .ad
309 .RS 23n
310 The specified signal, sent by \fBrctlblk_set_local_action()\fR, will be sent to
311 the process that placed this resource control value in the value sequence. This
312 behavior is also true for signal actions on project and task resource controls.
313 The specified signal is sent only to the recipient process, not all processes
314 within the project or task.
315 .RE

317 .sp
318 .LP
319 The \fBrctlblk_get_recipient_pid()\fR function returns the value of the process
320 ID that placed the resource control value for basic rctls. For privileged or
321 system rctls, \fBrctlblk_get_recipient_pid()\fR returns -1.

```

```

322 .sp
323 .LP
324 The \fBrctlblk_set_recipient_pid()\fR function sets the recipient \fIpid\fR for
325 a basic rctl. When \fBsetrctl()\fR(2) is called with the flag
326 \fBRCTL_USE_RECIPIENT_PID\fR, this \fIpid\fR is used. Otherwise, the PID of the
327 calling process is used. Only privileged users can set the recipient PID to one
328 other than the PID of the calling process. Process-scoped rctls must have a
329 recipient PID that matches the PID of the calling process.
330 .sp
331 .LP
332 The \fBrctlblk_get_privilege()\fR function returns the privilege of the
333 resource control block. Valid privileges are \fBRCPRIV_BASIC\fR,
334 \fBRCPRIV_PRIVILEGED\fR, and \fBRCPRIV_SYSTEM\fR. System resource controls are
335 read-only. Privileged resource controls require the {\fBPRIV_SYS_RESOURCE\fR}
336 privilege to write, unless the \fBRCTL_GLOBAL_LOWERABLE\fR global flag is set,
337 in which case unprivileged applications can lower the value of a privileged
338 control.
339 .sp
340 .LP
341 The \fBrctlblk_get_value()\fR and \fBrctlblk_set_value()\fR functions return or
342 establish the enforced value associated with the resource control. In cases
343 where the process, task, or project associated with the control possesses fewer
344 capabilities than allowable by the current value, the value returned by
345 \fBrctlblk_get_enforced_value()\fR will differ from that returned by
346 \fBrctlblk_get_value()\fR. This capability difference arises with processes
347 using an address space model smaller than the maximum address space model
348 supported by the system.
349 .sp
350 .LP
351 The \fBrctlblk_size()\fR function returns the size of a resource control block
352 for use in memory allocation. The \fBrctlblk_t *\fR type is an opaque pointer
353 whose size is not connected with that of the resource control block itself. Use
354 of \fBrctlblk_size()\fR is illustrated in the example below.
355 .SH RETURN VALUES
356 .sp
357 .LP
358 The various set routines have no return values. Incorrectly composed resource
359 control blocks will generate errors when used with \fBsetrctl()\fR(2) or
360 \fBgctrctl()\fR(2).
361 .SH ERRORS
362 .sp
363 .LP
364 No error values are returned. Incorrectly constructed resource control blocks
365 will be rejected by the system calls.
366 .SH EXAMPLES
367 .sp
368 .LP
369 The following example displays the contents of a fetched resource control
370 block.

372 .sp
373 .in +2
374 .nf
375 #include <rctl.h>
376 #include <stdio.h>
377 #include <stdlib.h>

379 int
380 main()
381 {
382     rctlblk_t *rblk;
383     int rsignal, raction;
384     rctlblk_t *rblk;
385     int rsignal;

```

```

384 int raction;

385     if ((rblk = malloc(rctlblk_size())) == NULL) {
386 if ((rblk = malloc(rctlblk_size())) == NULL) {
386     (void) perror("rblk malloc");
387     exit(1);
388     }
389 }

390     if (getrctl("process.max-cpu-time", NULL, rblk, RCTL_FIRST) == -1) {
391 if (getrctl("process.max-cpu-time", NULL, rblk, RCTL_FIRST) == -1) {
391     (void) perror("getrctl");
392     exit(1);
393     }
394 }

396 main()
397 {
398     raction = rctlblk_get_local_action(rblk, &rsignal),
399     (void) printf("Resource control for %s\n",
400     "process.max-cpu-time");
401     (void) printf("Process ID:      %d\n",
402     (int)rctlblk_get_recipient_pid(rblk));
403     (void) printf("Privilege:      %x\n",
404     rctlblk_get_recipient_pid(rblk));
405     (void) printf("Privilege:      %x\n",
406     rctlblk_get_privilege(rblk));
407     (void) printf("Global flags:   %x\n",
408     rctlblk_get_global_flags(rblk));
409     (void) printf("Global actions: %x\n",
410     rctlblk_get_global_action(rblk));
411     (void) printf("Local flags:    %x\n",
412     rctlblk_get_local_flags(rblk));
413     (void) printf("Local action:  %x (%d)\n",
414     raction, raction == RCTL_LOCAL_SIGNAL ? rsignal : 0);
415     (void) printf("Value:         %llu\n",
416     rctlblk_get_value(rblk));
417     (void) printf("\tEnforced value: %llu\n",
418     rctlblk_get_enforced_value(rblk));

419     return (0);
420 }
421 .fi
422 .in -2

423 .SH ATTRIBUTES
424 .sp
425 .LP
426 See \fBattributes\fR(5) for descriptions of the following attributes:
427 .sp

428 .sp
429 .TS
430 box;
431 c | c
432 l | l .
433 ATTRIBUTE TYPE  ATTRIBUTE VALUE
434 -
435 Interface Stability      Evolving
436 -
437 MT-Level                 MT-Safe
438 .TE

```

```

437 .SH SEE ALSO
438 .sp
439 .LP
440 \fBBrctladm\fR(1M), \fBgetrctl\fR(2), \fBsetrctl\fR(2), \fBgethrtime\fR(3C),
441 \fBattributes\fR(5)

```