

```

*****
5814 Fri Feb 3 18:30:56 2017
new/usr/src/man/man9/vmem.9
code review from Josh and Robert
*****
1 .\"
2 .\" CDDL HEADER START
3 .\"
4 .\" The contents of this file are subject to the terms of the
5 .\" Common Development and Distribution License (the "License").
6 .\" You may not use this file except in compliance with the License.
7 .\"
8 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 .\" or http://www.opensolaris.org/os/licensing.
10 .\" See the License for the specific language governing permissions
11 .\" and limitations under the License.
12 .\"
13 .\" When distributing Covered Code, include this CDDL HEADER in each
14 .\" file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 .\" If applicable, add the following below this CDDL HEADER, with the
16 .\" fields enclosed by brackets "[]" replaced with your own identifying
17 .\" information: Portions Copyright [yyyy] [name of copyright owner]
18 .\"
19 .\" CDDL HEADER END
20 .\"
21 .\"
22 .\" Copyright 2010 Sun Microsystems, Inc. All rights reserved.
23 .\" Use is subject to license terms.
24 .\"
25 .\" Copyright (c) 2012, 2015 by Delphix. All rights reserved.
26 .\" Copyright (c) 2012, Joyent, Inc. All rights reserved.
27 .\"
28 .\" The text of this is derived from section 1 of the big theory statement in
29 .\" usr/src/uts/common/os/vmem.c, the traditional location of this text. They
30 .\" should largely be updated in tandem.
31 .Dd Jan 18, 2017
32 .Dt VMEM 9
33 .Os
34 .Sh NAME
35 .Nm vmem
36 .Nd virtual memory allocator
37 .Sh DESCRIPTION
38 .Ss Overview
39 An address space is divided into a number of logically distinct pieces, or
40 .Em arenas :
41 text, data, heap, stack, and so on.
42 Within these
43 arenas we often subdivide further; for example, we use heap addresses
44 not only for the kernel heap
45 .Po
46 .Fn kmem_alloc
47 space
48 .Pc ,
49 but also for DVMA,
50 .Fn bp_mapin ,
51 .Pa /dev/kmem ,
52 and even some device mappings.
53 .Pp
54 The kernel address space, therefore, is most accurately described as
55 a tree of arenas in which each node of the tree
56 .Em imports
57 some subset of its parent.
58 The virtual memory allocator manages these arenas
59 and supports their natural hierarchical structure.
60 .Ss Arenas
61 An arena is nothing more than a set of integers. These integers most

```

```

62 commonly represent virtual addresses, but in fact they can represent
63 anything at all. For example, we could use an arena containing the
64 integers minpid through maxpid to allocate process IDs. For uses of this
65 nature, prefer
66 .Xr id_space 9F
67 instead.
68 .Pp
69 .Fn vmem_create
70 and
71 .Fn vmem_destroy
72 create and destroy vmem arenas. In order to differentiate between arenas used
73 for addresses and arenas used for identifiers, the
74 for addresses and arenas used for identifiers, the
75 .Dv VMC_IDENTIFIER
76 flag is passed to
77 .Fn vmem_create .
78 This prevents identifier exhaustion from being diagnosed as general memory
79 failure.
80 .Ss Spans
81 We represent the integers in an arena as a collection of
82 .Em spans ,
83 or contiguous ranges of integers. For example, the kernel heap consists of
84 just one span:
85 .Li "[kernelheap, ekernelheap]" .
86 Spans can be added to an arena in two ways: explicitly, by
87 .Fn vmem_add ;
88 .Fn vmem_add ,
89 or implicitly, by importing, as described in
90 .Sx Imported Memory
91 below.
92 .Ss Segments
93 Spans are subdivided into
94 .Em segments ,
95 each of which is either allocated or free. A segment, like a span, is a
96 contiguous range of integers. Each allocated segment
97 .Li "[addr, addr + size)"
98 represents exactly one
99 .Li "vmem_alloc(size)"
100 that returned
101 .Sy addr .
102 Free segments represent the space between allocated segments. If two free
103 segments are adjacent, we coalesce them into one larger segment; that is, if
104 segments
105 .Li "[a, b)"
106 and
107 .Li "[b, c)"
108 are both free, we merge them into a single segment
109 .Li "[a, c)" .
110 The segments within a span are linked together in increasing-address
111 order so we can easily determine whether coalescing is possible.
112 .Pp
113 Segments never cross span boundaries. When all segments within an imported
114 span become free, we return the span to its source.
115 .Ss Imported Memory
116 As mentioned in the overview, some arenas are logical subsets of
117 other arenas. For example,
118 .Sy kmem_va_arena
119 (a virtual address cache
120 that satisfies most
121 .Fn kmem_slab_create
122 requests) is just a subset of
123 .Sy heap_arena
124 (the kernel heap) that provides caching for the most common slab sizes. When
125 runs out of virtual memory, it
126 .Em imports

```

```
126 more from the heap; we say that
125 .Em imports more from the heap; we
126 say that
127 .Sy heap_arena
128 is the
129 .Em "vmem source"
130 for
129 .Em "vmem source" for
131 .Sy kmem_va_arena.
132 .Fn vmem_create
133 allows you to specify any existing vmem arena as the source for your new
134 arena. Topologically, since every arena is a child of at most one source, the
135 set of all arenas forms a collection of trees.
136 .Ss Constrained Allocations
137 Some vmem clients are quite picky about the kind of address they want.
138 For example, the DVMA code may need an address that is at a particular
139 phase with respect to some alignment (to get good cache coloring), or
140 that lies within certain limits (the addressable range of a device),
141 or that doesn't cross some boundary (a DMA counter restriction) \(\em
142 or all of the above.
143 .Fn vmem_xalloc
144 allows the client to specify any or all of these constraints.
145 .Ss The Vmem Quantum
146 Every arena has a notion of
147 .Sq quantum ,
148 specified at
149 .Fn vmem_create
150 time, that defines the arena's minimum unit of currency. Most commonly the
151 quantum is either 1 or
152 .Dv PAGESIZE ,
153 but any power of 2 is legal. All vmem allocations are guaranteed to be
154 quantum\(-aligned.
155 .Ss Relationship to the Kernel Memory Allocator
156 Every kmem cache has a vmem arena as its slab supplier. The kernel memory
157 allocator uses
158 .Fn vmem_alloc
159 and
160 .Fn vmem_free
161 to create and destroy slabs.
162 .Sh SEE ALSO
163 .Xr id_space 9F ,
164 .Xr vmem_add 9F ,
165 .Xr vmem_alloc 9F ,
166 .Xr vmem_contains 9F ,
167 .Xr vmem_create 9F ,
168 .Xr vmem_walk 9F
169 .Pp
170 .Rs
171 .%A Jeff Bonwick
172 .%A Jonathan Adams
173 .%T Magazines and vmem: Extending the Slab Allocator to Many CPUs and Arbitrary
174 .%J Proceedings of the 2001 Usenix Conference
175 .%U http://www.usenix.org/event/usenix01/bonwick.html
176 .Re
```

```
*****
1572 Fri Feb 3 18:30:57 2017
new/usr/src/man/man9f/vmem_add.9f
code review from Josh and Robert
*****
```

```
1 .\"
2 .\" This file and its contents are supplied under the terms of the
3 .\" Common Development and Distribution License ("CDDL"), version 1.0.
4 .\" You may only use this file in accordance with the terms of version
5 .\" 1.0 of the CDDL.
6 .\"
7 .\" A full copy of the text of the CDDL should have accompanied this
8 .\" source. A copy of the CDDL is also available via the Internet at
9 .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright 2017, Richard Lowe.
13 .\"
14 .Dd Jan 18, 2017
15 .Dt VMEM_ADD 9F
16 .Os
17 .Sh NAME
18 .Nm vmem_add
19 .Nd add spans to a vmem arena
20 .Sh SYNOPSIS
21 .In sys/vmem.h
22 .Ft void *
23 .Fo vmem_add
24 .Fa "vmem_t *vmp"
25 .Fa "void *vaddr"
26 .Fa "size_t size"
27 .Fa "int vmflag"
28 .Fc
29 .Sh INTERFACE LEVEL
30 illumos DDI specific
31 .Sh PARAMETERS
32 .Bl -tag -width Ds
33 .It Fa vmp
34 The vmem arena to which the span should be added.
35 .It Fa vaddr
36 The base address of the span to add.
37 .It Fa size
38 The size of the span to add.
39 The size of the span to add
40 .It Fa vmflag
41 Flags affecting the allocation of the span to add.
42 .El
43 .Sh DESCRIPTION
44 The
45 .Fn vmem_add
46 function adds
47 .Fa size
48 bytes starting at
49 .Fa vaddr
50 to a vmem arena from which future calls to
51 .Fn vmem_alloc
52 may allocate.
53 .Pp
54 .Dv VM_SLEEP
55 or
56 .Dv VM_NOSLEEP
57 must be specified in
58 .Fa vmflag ,
59 and indicate whether the addition may block.
60 must be specified, and indicate whether the addition may block.
61 .Sh CONTEXT
```

```
60 This function can be called from either user or kernel context.
61 If the
62 .Dv VM_NOSLEEP
63 flag is specified, it may also be called from interrupt context.
64 .Sh RETURN VALUES
65 Upon success
66 .Fn vmem_add
67 returns
68 .Fa vaddr .
69 On failure,
70 .Dv NULL
71 is returned.
72 .Sh SEE ALSO
73 .Xr vmem 9 ,
74 .Xr vmem_alloc 9F ,
75 .Xr vmem_create 9F
```

4086 Fri Feb 3 18:30:59 2017

new/usr/src/man/man9f/vmem_alloc.9f

code review from Josh and Robert

```

1  \
2  \ This file and its contents are supplied under the terms of the
3  \ Common Development and Distribution License ("CDDL"), version 1.0.
4  \ You may only use this file in accordance with the terms of version
5  \ 1.0 of the CDDL.
6  \
7  \ A full copy of the text of the CDDL should have accompanied this
8  \ source. A copy of the CDDL is also available via the Internet at
9  \ http://www.illumos.org/license/CDDL.
10 \
11 \
12 \ Copyright 2017, Richard Lowe.
13 \
14 .Dd Jan 18, 2017
15 .Dt VMEM_ALLOC 9F
16 .Os
17 .Sh NAME
18 .Nm vmem_alloc ,
19 .Nm vmem_xalloc ,
20 .Nm vmem_free ,
21 .Nm vmem_xfree
22 .Nd allocate and free segments from a vmem arena
23 .Sh SYNOPSIS
24 .In sys/vmem.h
25 .Ft void *
26 .Fo vmem_alloc
27 .Fa "vmem_t *vmp"
28 .Fa "size_t size"
29 .Fa "int vmflag"
30 .Fc
31 .Ft void *
32 .Fo vmem_xalloc
33 .Fa "vmem_t *vmp"
34 .Fa "size_t size"
35 .Fa "size_t align_arg"
36 .Fa "size_t phase"
37 .Fa "size_t nocross"
38 .Fa "void *minaddr"
39 .Fa "void *maxaddr"
40 .Fa "int vmflag"
41 .Fc
42 .Ft void
43 .Fo vmem_free
44 .Fa "vmem_t *vmp"
45 .Fa "void *vaddr"
46 .Fa "size_t size"
47 .Fc
48 .Ft void
49 .Fo vmem_xfree
50 .Fa "vmem_t *vmp"
51 .Fa "void *vaddr"
52 .Fa "size_t size"
53 .Fc
54 .Sh INTERFACE LEVEL
55 illumos DDI specific
56 .Sh PARAMETERS
57 .Bl -tag -width Ds
58 .It Fa vmp
59 The vmem arena from which to allocate or free.
60 .It Fa size
61 The size of the segment to allocate or free.

```

```

62 .It Fa vmflag
63 A bitmask of flags controlling the behaviour of the allocation.
64 There are two meaningful groups of flags.
65 .Dv VM_SLEEP
66 or
67 .Dv VM_NOSLEEP
68 must be specified, and indicate whether the allocation may block. A
69 .Dv VM_SLEEP
70 allocation can never fail but may block indefinitely.
71 .Pp
72 The allocation policy may be specified by one of the following flags:
73 .Bl -tag -width Ds
74 .It Dv VM_BESTFIT
75 Take the segment from the smallest free segment that could satisfy this allocati
75 Take the segment from the smallest free segment that could satisfy this allocati
76 .It Dv VM_FIRSTFIT
77 Take the segment from the first free segment found that could satisfy this
78 allocation.
79 .It Dv VM_NEXTFIT
80 Take the segment from the segment after the one previously allocated. This
81 provides sequential behaviour useful when allocating identifiers from a
82 .Dv VMC_IDENTIFIER
83 arena.
84 .It Dv VM_ENDALLOC
85 May be specified in combination with
86 .Dv VM_BESTFIT ,
87 .Dv VM_FIRSTFIT
88 or the default policy to indicate that the higher addresses should be
89 preferred.
90 .El
91 .Pp
92 The default (unnamed) allocation policy is
93 .Dq "instant fit" ,
93 The default (un\ -named) allocation policy is
93 .Dq instant fit
94 an approximation of
95 .Dv VM_BESTFIT
96 in guaranteed constant time.
97 .It Fa align_arg
98 The minimum alignment of the allocation. If 0,
98 The minimum alignment of the allocation. If
99 .Ql 0
99 the allocated segment will be aligned as the arena's quantum.
100 .It Fa phase
101 The allocated segment must be
102 .Fa phase
103 bytes from the alignment boundary.
104 .It Fa nocross
105 The allocated segment may not straddle a
106 .Fa nocross
107 alignment boundary.
108 .It Fa minaddr
109 The minimum address at which the segment may be allocated.
110 .It Fa maxaddr
111 The maximum address which may be included in the segment.
112 .It Fa vaddr
113 The address of the segment which
114 .Fn vmem_free
115 or
116 .Fn vmem_xfree
117 should free.
118 .El
119 .Sh DESCRIPTION
120 The
121 .Fn vmem_alloc
122 and

```

```
123 .Fn vmem_xalloc
124 functions allocate a segment of
125 .Fa size
126 length from the vmem arena
127 .Fa vmp .
128 .Pp
129 The
130 .Fa vmflag
131 argument controls the behaviour of the allocation, as described in
132 .Sx PARAMETERS .
132 argument controls the behaviour of the allocation. As described in
133 .Sx PARAMETERS
133 .Pp
134 For allocations with complex requirements, such as those used for DMA
135 .Fn vmem_xalloc
136 takes additional arguments allowing those requirements to be expressed.
137 .Pp
138 Segments allocated with
139 .Fn vmem_xalloc
140 must always be freed with
141 .Fn vmem_xfree ,
142 since these allocations are uncached.
143 .Sh CONTEXT
144 This function can be called from either user or kernel context.
145 If the
146 .Dv VM_NOSLEEP
147 flag is specified, it may also be called from interrupt context.
148 .Sh RETURN VALUES
149 Upon successful completion the
150 .Fn vmem_alloc
151 and
152 .Fn vmem_xalloc
153 functions return a pointer to the beginning of the allocated segment. In the
154 case of a
155 .Dv VMC_IDENTIFIER
156 arena, the address of this pointer is the meaningful component, not the value
157 to which it points.
158 .Pp
159 On failure,
160 .Dv NULL
161 is returned.
162 When the
163 .Dv VM_SLEEP
164 flag is specified, these functions can never fail (but may block forever).
165 .Sh SEE ALSO
166 .Xr vmem 9 ,
167 .Xr vmem_contains 9F ,
168 #endif /* ! codereview */
169 .Xr vmem_create 9F
```

```
*****
1264 Fri Feb  3 18:31:01 2017
new/usr/src/man/man9f/vmem_contains.9f
code review from Josh and Robert
*****
```

```
1 .\"
2 .\" This file and its contents are supplied under the terms of the
3 .\" Common Development and Distribution License ("CDDL"), version 1.0.
4 .\" You may only use this file in accordance with the terms of version
5 .\" 1.0 of the CDDL.
6 .\"
7 .\" A full copy of the text of the CDDL should have accompanied this
8 .\" source. A copy of the CDDL is also available via the Internet at
9 .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright 2017, Richard Lowe.
13 .\"
14 .Dd Jan 18, 2017
15 .Dt VMEM_CONTAINS 9F
16 .Os
17 .Sh NAME
18 .Nm vmem_contains
19 .Nd check for membership in a vmem arena
20 .Sh SYNOPSIS
21 .In sys/vmem.h
22 .Ft int
23 .Fo vmem_contains
24 .Fa "vmem_t *vmp"
25 .Fa "void *vaddr"
26 .Fa "size_t *size"
27 .Fc
28 .Sh INTERFACE LEVEL
29 illumos DDI specific
30 .Sh PARAMETERS
31 .Bl -tag -width Ds
32 .It Fa vmp
33 The vmem arena
34 .It Fa vaddr
35 address of the segment to query.
36 .It Fa size
37 size of the segment to query
38 .El
39 .Sh DESCRIPTION
40 The
41 .Fn vmem_contains
42 function checks whether a segment of
43 .Fa size
44 bytes at
45 .Fa vaddr
46 exists within the vmem arena
47 .Fa vmp .
48 .Sh CONTEXT
49 This function may be called from user or kernel context.
50 .Sh RETURN VALUES
51 .Fn vmem_contains
52 returns non\0 if the segment exists, and 0 otherwise.
53 .Sh SEE ALSO
54 .Xr vmem 9 ,
55 .Xr vmem_add 9F ,
56 .Xr vmem_alloc 9F ,
57 .Xr vmem_create 9F
```

4846 Fri Feb 3 18:31:02 2017

new/usr/src/man/man9f/vmem_create.9f

code review from Josh and Robert

```

1 .\"
2 .\" This file and its contents are supplied under the terms of the
3 .\" Common Development and Distribution License ("CDDL"), version 1.0.
4 .\" You may only use this file in accordance with the terms of version
5 .\" 1.0 of the CDDL.
6 .\"
7 .\" A full copy of the text of the CDDL should have accompanied this
8 .\" source. A copy of the CDDL is also available via the Internet at
9 .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright 2017, Richard Lowe.
13 .\"
14 .Dd Jan 18, 2017
15 .Dt VMEM_CREATE 9f
16 .Os
17 .Sh NAME
18 .Nm vmem_create ,
19 .Nm vmem_xcreate ,
20 .Nm vmem_destroy
21 .Nd create and destroy vmem arenas
22 .Sh SYNOPSIS
23 .In sys/vmem.h
24 .Vt "typedef struct vmem vmem_t;"
25 .Vt "typedef void *(vmem_alloc_t)(vmem_t *, size_t, int);"
26 .Vt "typedef void (vmem_free_t)(vmem_t *, void *, size_t);"
27 .Vt "typedef void *(vmem_ximport_t)(vmem_t *, size_t *, size_t, int);"
28 .Ft vmem_t *
29 .Fo vmem_create
30 .Fa "const char *name"
31 .Fa "void *base"
32 .Fa "size_t size"
33 .Fa "size_t quantum"
34 .Fa "vmem_alloc_t *afunc"
35 .Fa "vmem_free_t *ffunc"
36 .Fa "vmem_t *source"
37 .Fa "size_t qcache_max"
38 .Fa "int vmflag"
39 .Fc
40 .Ft vmem_t *
41 .Fo vmem_xcreate
42 .Fa "const char *name"
43 .Fa "void *base"
44 .Fa "size_t size"
45 .Fa "size_t quantum"
46 .Fa "vmem_ximport_t *afunc"
47 .Fa "vmem_free_t *ffunc"
48 .Fa "vmem_t *source"
49 .Fa "size_t qcache_max"
50 .Fa "int vmflag"
51 .Fc
52 .Ft void
53 .Fo vmem_destroy
54 .Fa "vmem_t *vmp"
55 .Fc
56 .Sh INTERFACE LEVEL
57 illumos DDI specific
58 .Sh PARAMETERS
59 .Bl -tag -width Ds
60 .It Fa name
61 A character string giving a name to the vmem

```

```

62 arena to be created.
63 .It Fa base
64 An address indicating the lowest possible value in the arena.
65 .It Fa size
66 The size of the arena to create.
67 The size of the arena to create
68 .It Fa quantum
69 The arena's
70 .Dq quantum .
71 The granularity of the arena. The amount allocated at minimum by each
72 request. Must be a power of 2.
73 The granularity of the arena. The amount allocated at minimum by each request.
74 .It Fa afunc
75 A function which is called to import new spans from
76 .Fa source ,
77 which may be
78 .Fa source .
79 Which may be
80 .Dv NULL
81 if this arena does not import from another.
82 When calling
83 .Fn vmem_create
84 .Fa afunc
85 is an
86 .Vt vmem_alloc_t
87 a function taking three parameters and returning a pointer to
88 .Vt void
89 (the imported space):
90 .Bl -tag -width Ds
91 .It Fa "vmem_t *"
92 The source arena from which we'll import. The
93 .Fa source
94 argument to
95 .Fn vmem_create .
96 .It Fa size_t
97 The size to import
98 .It Fa int
99 The
100 .Fa vmflag
101 argument used for the import.
102 .El
103 .Pp
104 When calling
105 .Fn vmem_xcreate
106 .Fa afunc
107 is an
108 .Vt vmem_ximport_t
109 a function taking four parameters and returning a pointer to
110 .Vt void
111 (the imported space):
112 .Bl -tag -width Ds
113 .It Fa "vmem_t *"
114 The source arena from which we'll import. The
115 .Fa source
116 argument to
117 .Fn vmem_xcreate .
118 .It Fa "size_t *"
119 The size of the import,
120 .Fa afunc
121 may
122 .Em increase
123 this size if that is desirable, but must never decrease it.
124 .It Fa size_t
125 The desired alignment of the imported space.
126 .It Fa int
127 the

```

```

124 .Fa vmflag
125 argument used for the import.
126 .El
127 #endif /* ! codereview */
128 .It Fa ffunc
129 A function which is called to return spans to
130 .Fa source ,
131 which may be
132   77 .Fa source .
133   78 Which may be
132 .Dv NULL
133 if this arena does not import from another.
134 This is a
135 .Vt vmem_free_t
136 a function taking three parametes and returning void:
137 .Bl -tag -width Ds
138 .It Fa "vmem_t"
139 The arena to which space is being returned. The
140 .Fa source
141 argument to
142 .Fn vmem_create
143 or
144 .Fn vmem_xcreate .
145 .It Fa "void *"
146 The span being returned to the source arena.
147 .It Fa "size_t"
148 The size of the span being returned to the source arena.
149 .El
150 #endif /* ! codereview */
151 .It Fa source
152 An arena from which this arena will import,
153 which may be
154   81 An arena from which this arena will import.
155   82 Which may be
154 .Dv NULL
155 if this arena does not import from another.
156 .It Fa qcache_max
157 Each arena offers caching of integer multiples of
158 .Fa quantum
159 up to
160 .Fa qcache_max ,
161 which may be 0.
162   89 .Fa qcache_max .
162 .It Fa vmflag
163 A bitmask of flags indicating the characteristics of this arena.
164 .Bl -tag -width Ds
165 .It Dv VMC_IDENTIFIER
166 The arena represents arbitrary integer identifiers, rather than virtual
167 memory.
168 .El
169 .It Fa vmp
170 A pointer to the vmem arena to be destroyed.
171 .El
172 .Sh DESCRIPTION
173 A
174 .Em vmem arena
175 is a section of an arbitrary address space (a range of integer addresses).
176 This commonly represents virtual memory, but can in fact be an arbitrary set
177 of integers. The
178 .Dv VMC_IDENTIFIER
179 flag set at arena creation time differentiates between these two cases.
180 .Pp
181 The
182 .Fa afunc ,
183 .Fa ffunc , and
184 .Fa source

```

```

185 arguments combine to support a hierarchical structure of arenas, each
186 importing from a single parent (the
187 .Fa source ) .
188 The
189 .Fn vmem_create
190 and
191 .Fn vmem_xcreate
192 functions differ in that the latter provides an interface for
193 .Fa afunc
194 to alter the size of the span imported from
195 .Fa source .
196 It is only legal to
197 .Em increase
198 this size.
126 thise size.
199 .Sh CONTEXT
200 These functions can be called from user or kernel context.
201 .Sh RETURN VALUES
202 Upon successful completion the
203 .Fn vmem_create
204 and
131 .Fn vmem_create and
205 .Fn vmem_xcreate
206 functions return a pointer to a vmem arena. Otherwise,
207 .Dv NULL
208 is returned to indicate the arena could not be created.
209 .Sh SEE ALSO
210 .Xr vmem 9 ,
211 .Xr vmem_add 9F ,
212 .Xr vmem_alloc 9F

```


new/usr/src/man/man9f/vmem_walk.9f

1

1837 Fri Feb 3 18:31:04 2017

new/usr/src/man/man9f/vmem_walk.9f

code review from Josh and Robert

```
1 .\"
2 .\" This file and its contents are supplied under the terms of the
3 .\" Common Development and Distribution License ("CDDL"), version 1.0.
4 .\" You may only use this file in accordance with the terms of version
5 .\" 1.0 of the CDDL.
6 .\"
7 .\" A full copy of the text of the CDDL should have accompanied this
8 .\" source. A copy of the CDDL is also available via the Internet at
9 .\" http://www.illumos.org/license/CDDL.
10 .\"
11 .\"
12 .\" Copyright 2017, Richard Lowe.
13 .\"
14 .Dd Jan 18, 2017
15 .Dt VMEM_WALK 9F
16 .Os
17 .Sh NAME
18 .Nm vmem_walk ,
19 .Nm vmem_size
20 .Nd walk a (sub\-)set of the segments in a vmem arena
21 .Sh SYNOPSIS
22 .In sys/vmem.h
23 .Ft void
24 .Fo vmem_walk
25 .Fa "vmem_t *vmp"
26 .Fa "int typemask"
27 .Fa "void (*func)(void *, void *, size_t)"
28 .Fa "void *arg"
29 .Fc
30 .Ft size_t
31 .Fo vmem_size
32 .Fa "vmem_t *vmp"
33 .Fa "int typemask"
34 .Fc
35 .Sh INTERFACE LEVEL
36 illumos DDI specific
37 .Sh PARAMETERS
38 .Bl -tag -width Ds
39 .It Fa vmp
40 The vmem arena to walk.
40 The vmem arena to walk
41 .It Fa typemask
42 A bitmask indicating the types of segment to operate on.
42 A bitmask indicating the types of segment to operate on
43 .Bl -tag -width Ds
44 .It Dv VMEM_ALLOC
45 Allocated segments.
45 Allocated segments
46 .It Dv VMEM_FREE
47 Free segments.
47 Free segments
48 .El
49 .It Fa func
50 The function to apply to each segment matching
51 .Fa typemask .
52 .Fa func
53 should accept 3 arguments and return void:
53 should be of 3 values and return void:
54 .Bl -tag -width Ds
55 .It Fa "void *arg"
56 The
```

new/usr/src/man/man9f/vmem_walk.9f

2

```
57 .Fa arg
58 passed to
59 .Fn vmem_walk .
59 .Fn vmem_walk
60 .It Fa "void *vaddr"
61 The base address of the segment.
61 The base address of the segment
62 .It Fa "size_t size"
63 The size of the segment.
63 The size of the segment
64 .El
65 .It Fa arg
66 An arbitrary argument passed to each call to
67 .Fn func .
67 .Fn func
68 .El
69 .Sh DESCRIPTION
70 .Fn vmem_walk
71 walks each segment in the arena
72 .Fa vmp
73 and applies
74 .Fa func
75 to each which matches
76 .Fa typemask .
77 .Pp
78 .Fn vmem_size
79 walks each segment in the arena
80 .Fa vmp
81 and totals the size of each matching
82 .Fa typemask .
83 .Sh CONTEXT
84 This function may be called from user or kernel context.
85 .Sh SEE ALSO
86 .Xr vmem 9 ,
87 .Xr vmem_alloc 9F ,
88 .Xr vmem_create 9F
```