

```

*****
77394 Thu Aug 1 18:13:05 2013
new/usr/src/cmd/mdb/common/mdb/mdb_print.c
3953 Calling ::list without specifying the name of the next member causes mdb to
Reviewed by: Christopher Siden <christopher.siden@delphix.com>
Reviewed by: Matthew Ahrens <mahrens@delphix.com>
*****
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 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

26 /*
27 * Copyright (c) 2013 by Delphix. All rights reserved.
28 * Copyright (c) 2012 by Delphix. All rights reserved.
29 * Copyright (c) 2012 Joyent, Inc. All rights reserved.
30 */

31 #include <mdb/mdb_modapi.h>
32 #include <mdb/mdb_target.h>
33 #include <mdb/mdb_argvec.h>
34 #include <mdb/mdb_string.h>
35 #include <mdb/mdb_stdlib.h>
36 #include <mdb/mdb_err.h>
37 #include <mdb/mdb_debug.h>
38 #include <mdb/mdb_fmt.h>
39 #include <mdb/mdb_ctf.h>
40 #include <mdb/mdb_ctf_impl.h>
41 #include <mdb/mdb.h>
42 #include <mdb/mdb_tab.h>

44 #include <sys/isa_defs.h>
45 #include <sys/param.h>
46 #include <sys/sysmacros.h>
47 #include <netinet/in.h>
48 #include <strings.h>
49 #include <libctf.h>
50 #include <ctype.h>

52 typedef struct holeinfo {
53     ulong_t hi_offset;          /* expected offset */
54     uchar_t hi_isunion;        /* represents a union */
55 } holeinfo_t;
56
57 #ifndef UNCHANGED_PORTION_OMITTED
58 #endif
59
60 #endif
61
62 #endif
63
64 #endif
65
66 #endif
67
68 #endif
69
70 #endif
71
72 #endif
73
74 #endif
75
76 #endif
77
78 #endif
79
80 #endif
81
82 #endif
83
84 #endif
85
86 #endif
87
88 #endif
89
90 #endif
91
92 #endif
93
94 #endif
95
96 #endif
97
98 #endif
99
100 #endif
101
102 #endif
103
104 #endif
105
106 #endif
107
108 #endif
109
110 #endif
111
112 #endif
113
114 #endif
115
116 #endif
117
118 #endif
119
120 #endif
121
122 #endif
123
124 #endif
125
126 #endif
127
128 #endif
129
130 #endif
131
132 #endif
133
134 #endif
135
136 #endif
137
138 #endif
139
140 #endif
141
142 #endif
143
144 #endif
145
146 #endif
147
148 #endif
149
150 #endif
151
152 #endif
153
154 #endif
155
156 #endif
157
158 #endif
159
160 #endif
161
162 #endif
163
164 #endif
165
166 #endif
167
168 #endif
169
170 #endif
171
172 #endif
173
174 #endif
175
176 #endif
177
178 #endif
179
180 #endif
181
182 #endif
183
184 #endif
185
186 #endif
187
188 #endif
189
190 #endif
191
192 #endif
193
194 #endif
195
196 #endif
197
198 #endif
199
200 #endif
201
202 #endif
203
204 #endif
205
206 #endif
207
208 #endif
209
210 #endif
211
212 #endif
213
214 #endif
215
216 #endif
217
218 #endif
219
220 #endif
221
222 #endif
223
224 #endif
225
226 #endif
227
228 #endif
229
230 #endif
231
232 #endif
233
234 #endif
235
236 #endif
237
238 #endif
239
240 #endif
241
242 #endif
243
244 #endif
245
246 #endif
247
248 #endif
249
250 #endif
251
252 #endif
253
254 #endif
255
256 #endif
257
258 #endif
259
260 #endif
261
262 #endif
263
264 #endif
265
266 #endif
267
268 #endif
269
270 #endif
271
272 #endif
273
274 #endif
275
276 #endif
277
278 #endif
279
280 #endif
281
282 #endif
283
284 #endif
285
286 #endif
287
288 #endif
289
290 #endif
291
292 #endif
293
294 #endif
295
296 #endif
297
298 #endif
299
300 #endif
301
302 #endif
303
304 #endif
305
306 #endif
307
308 #endif
309
310 #endif
311
312 #endif
313
314 #endif
315
316 #endif
317
318 #endif
319
320 #endif
321
322 #endif
323
324 #endif
325
326 #endif
327
328 #endif
329
330 #endif
331
332 #endif
333
334 #endif
335
336 #endif
337
338 #endif
339
340 #endif
341
342 #endif
343
344 #endif
345
346 #endif
347
348 #endif
349
350 #endif
351
352 #endif
353
354 #endif
355
356 #endif
357
358 #endif
359
360 #endif
361
362 #endif
363
364 #endif
365
366 #endif
367
368 #endif
369
370 #endif
371
372 #endif
373
374 #endif
375
376 #endif
377
378 #endif
379
380 #endif
381
382 #endif
383
384 #endif
385
386 #endif
387
388 #endif
389
390 #endif
391
392 #endif
393
394 #endif
395
396 #endif
397
398 #endif
399
400 #endif
401
402 #endif
403
404 #endif
405
406 #endif
407
408 #endif
409
410 #endif
411
412 #endif
413
414 #endif
415
416 #endif
417
418 #endif
419
420 #endif
421
422 #endif
423
424 #endif
425
426 #endif
427
428 #endif
429
430 #endif
431
432 #endif
433
434 #endif
435
436 #endif
437
438 #endif
439
440 #endif
441
442 #endif
443
444 #endif
445
446 #endif
447
448 #endif
449
450 #endif
451
452 #endif
453
454 #endif
455
456 #endif
457
458 #endif
459
460 #endif
461
462 #endif
463
464 #endif
465
466 #endif
467
468 #endif
469
470 #endif
471
472 #endif
473
474 #endif
475
476 #endif
477
478 #endif
479
480 #endif
481
482 #endif
483
484 #endif
485
486 #endif
487
488 #endif
489
490 #endif
491
492 #endif
493
494 #endif
495
496 #endif
497
498 #endif
499
500 #endif
501
502 #endif
503
504 #endif
505
506 #endif
507
508 #endif
509
510 #endif
511
512 #endif
513
514 #endif
515
516 #endif
517
518 #endif
519
520 #endif
521
522 #endif
523
524 #endif
525
526 #endif
527
528 #endif
529
530 #endif
531
532 #endif
533
534 #endif
535
536 #endif
537
538 #endif
539
540 #endif
541
542 #endif
543
544 #endif
545
546 #endif
547
548 #endif
549
550 #endif
551
552 #endif
553
554 #endif
555
556 #endif
557
558 #endif
559
560 #endif
561
562 #endif
563
564 #endif
565
566 #endif
567
568 #endif
569
570 #endif
571
572 #endif
573
574 #endif
575
576 #endif
577
578 #endif
579
580 #endif
581
582 #endif
583
584 #endif
585
586 #endif
587
588 #endif
589
590 #endif
591
592 #endif
593
594 #endif
595
596 #endif
597
598 #endif
599
600 #endif
601
602 #endif
603
604 #endif
605
606 #endif
607
608 #endif
609
610 #endif
611
612 #endif
613
614 #endif
615
616 #endif
617
618 #endif
619
620 #endif
621
622 #endif
623
624 #endif
625
626 #endif
627
628 #endif
629
630 #endif
631
632 #endif
633
634 #endif
635
636 #endif
637
638 #endif
639
640 #endif
641
642 #endif
643
644 #endif
645
646 #endif
647
648 #endif
649
650 #endif
651
652 #endif
653
654 #endif
655
656 #endif
657
658 #endif
659
660 #endif
661
662 #endif
663
664 #endif
665
666 #endif
667
668 #endif
669
670 #endif
671
672 #endif
673
674 #endif
675
676 #endif
677
678 #endif
679
680 #endif
681
682 #endif
683
684 #endif
685
686 #endif
687
688 #endif
689
690 #endif
691
692 #endif
693
694 #endif
695
696 #endif
697
698 #endif
699
700 #endif
701
702 #endif
703
704 #endif
705
706 #endif
707
708 #endif
709
710 #endif
711
712 #endif
713
714 #endif
715
716 #endif
717
718 #endif
719
720 #endif
721
722 #endif
723
724 #endif
725
726 #endif
727
728 #endif
729
730 #endif
731
732 #endif
733
734 #endif
735
736 #endif
737
738 #endif
739
740 #endif
741
742 #endif
743
744 #endif
745
746 #endif
747
748 #endif
749
750 #endif
751
752 #endif
753
754 #endif
755
756 #endif
757
758 #endif
759
760 #endif
761
762 #endif
763
764 #endif
765
766 #endif
767
768 #endif
769
770 #endif
771
772 #endif
773
774 #endif
775
776 #endif
777
778 #endif
779
780 #endif
781
782 #endif
783
784 #endif
785
786 #endif
787
788 #endif
789
790 #endif
791
792 #endif
793
794 #endif
795
796 #endif
797
798 #endif
799
800 #endif
801
802 #endif
803
804 #endif
805
806 #endif
807
808 #endif
809
810 #endif
811
812 #endif
813
814 #endif
815
816 #endif
817
818 #endif
819
820 #endif
821
822 #endif
823
824 #endif
825
826 #endif
827
828 #endif
829
830 #endif
831
832 #endif
833
834 #endif
835
836 #endif
837
838 #endif
839
840 #endif
841
842 #endif
843
844 #endif
845
846 #endif
847
848 #endif
849
850 #endif
851
852 #endif
853
854 #endif
855
856 #endif
857
858 #endif
859
860 #endif
861
862 #endif
863
864 #endif
865
866 #endif
867
868 #endif
869
870 #endif
871
872 #endif
873
874 #endif
875
876 #endif
877
878 #endif
879
880 #endif
881
882 #endif
883
884 #endif
885
886 #endif
887
888 #endif
889
890 #endif
891
892 #endif
893
894 #endif
895
896 #endif
897
898 #endif
899
900 #endif
901
902 #endif
903
904 #endif
905
906 #endif
907
908 #endif
909
910 #endif
911
912 #endif
913
914 #endif
915
916 #endif
917
918 #endif
919
920 #endif
921
922 #endif
923
924 #endif
925
926 #endif
927
928 #endif
929
930 #endif
931
932 #endif
933
934 #endif
935
936 #endif
937
938 #endif
939
940 #endif
941
942 #endif
943
944 #endif
945
946 #endif
947
948 #endif
949
950 #endif
951
952 #endif
953
954 #endif
955
956 #endif
957
958 #endif
959
960 #endif
961
962 #endif
963
964 #endif
965
966 #endif
967
968 #endif
969
970 #endif
971
972 #endif
973
974 #endif
975
976 #endif
977
978 #endif
979
980 #endif
981
982 #endif
983
984 #endif
985
986 #endif
987
988 #endif
989
990 #endif
991
992 #endif
993
994 #endif
995
996 #endif
997
998 #endif
999
1000 #endif

```

```

697 int
698 cmd_list(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
699 {
700     int offset;
701     uintptr_t a, tmp;
702     int ret;

704     if (!(flags & DCMD_ADDRSPEC) || argc == 0)
705         return (DCMD_USAGE);

707     if (argv->a_type != MDB_TYPE_STRING) {
708         /*
709          * We are being given a raw offset in lieu of a type and
710          * member; confirm the number of arguments and argument
711          * type.
712          * member; confirm the arguments.
713          */
714         if (argc != 1 || argv->a_type != MDB_TYPE_IMMEDIATE)
715             if (argv->a_type != MDB_TYPE_IMMEDIATE)
716                 return (DCMD_USAGE);

718         offset = argv->a_un.a_val;
719         argv++;
720         argc--;

722         if (offset % sizeof (uintptr_t)) {
723             mdb_warn("offset must fall on a word boundary\n");
724             return (DCMD_ABORT);
725         } else {
726             const char *member;
727             char buf[MDB_SYM_NAMLEN];
728             int ret;

730             /*
731              * Check that we were provided 2 arguments: a type name
732              * and a member of that type.
733              */
734             if (argc != 2)
735                 return (DCMD_USAGE);

737             ret = args_to_typename(&argc, &argv, buf, sizeof (buf));
738             if (ret != 0)
739                 return (ret);

741             argv++;
742             argc--;

744             member = argv->a_un.a_str;
745             offset = mdb_ctf_offsetof_by_name(buf, member);
746             if (offset == -1)
747                 return (DCMD_ABORT);

749             argv++;
750             argc--;

752             if (offset % (sizeof (uintptr_t)) != 0) {
753                 mdb_warn("%s is not a word-aligned member\n", member);
754                 return (DCMD_ABORT);
755             }
756         }

758     /*
759      * If we have any unchewed arguments, a variable name must be present.
760      */

```

```
761     if (argc == 1) {
762         if (argv->a_type != MDB_TYPE_STRING)
763             return (DCMD_USAGE);
764
765         if ((ret = setup_vcb(argv->a_un.a_str, addr)) != 0)
766             return (ret);
767
768     } else if (argc != 0) {
769         return (DCMD_USAGE);
770     }
771
772     a = addr;
773
774     do {
775         mdb_printf("%lr\n", a);
776
777         if (mdb_vread(&tmp, sizeof (tmp), a + offset) == -1) {
778             mdb_warn("failed to read next pointer from object %p",
779                 a);
780             return (DCMD_ERR);
781         }
782
783         a = tmp;
784     } while (a != addr && a != NULL);
785
786     return (DCMD_OK);
787 }
```

unchanged portion omitted