

new/usr/src/uts/common/krtld/kobj.c

1

```
*****  
111823 Mon Sep 9 20:46:53 2013  
new/usr/src/uts/common/krtld/kobj.c  
4122 do_sysfile_cmd colon-separates the module path, and then we can't parse it  
*****  
_____unchanged_portion_omitted_____  
  
3433 /*  
3434 * fullname is dynamically allocated to be able to hold the  
3435 * maximum size string that can be constructed from name.  
3436 * path is exactly like the shell PATH variable.  
3437 */  
3438 struct _buf *  
3439 kobj_open_path(char *name, int use_path, int use_moddir_suffix)  
3440 {  
3441     char *p, *q;  
3442     char *pathp;  
3443     char *pathpsave;  
3444     char *fullname;  
3445     int maxpathlen;  
3446     struct _buf *file;  
  
3448 #if !defined(MODDIR_SUFFIX)  
3449     use_moddir_suffix = B_FALSE;  
3450 #endif  
  
3452     if (!use_path)  
3453         pathp = ""; /* use name as specified */  
3454     else  
3455         pathp = kobj_module_path; /* use configured default path */  
  
3458     pathpsave = pathp; /* keep this for error reporting */  
  
3460     /*  
3461      * Allocate enough space for the largest possible fullname.  
3462      * since path is of the form <directory> : <directory> : ...  
3463      * we're potentially allocating a little more than we need to  
3464      * but we'll allocate the exact amount when we find the right directory.  
3465      * (The + 3 below is one for NULL terminator and one for the '/'  
3466      * we might have to add at the beginning of path and one for  
3467      * the '/' between path and name.)  
3468     */  
3469     maxpathlen = strlen(pathp) + strlen(name) + 3;  
3470     /* sizeof includes null */  
3471     maxpathlen += sizeof(slash_moddir_suffix_slash) - 1;  
3472     fullname = kobj_zalloc(maxpathlen, KM_WAIT);  
  
3474     for (;;) {  
3475         p = fullname;  
3476         if (*pathp != '\0' && *pathp != '/')  
3477             *p++ = '/'; /* path must start with '/' */  
3478         while (*pathp && *pathp != ':' && *pathp != ' ')  
3479             *p++ = *pathp++;  
3480         if (p != fullname && p[-1] != '/')  
3481             *p++ = '/';  
3482         if (use_moddir_suffix) {  
3483             char *b = basename(name);  
3484             char *s;  
  
3486             /* copy everything up to the base name */  
3487             q = name;  
3488             while (q != b && *q)  
3489                 *p++ = *q++;  
3490             s = slash_moddir_suffix_slash;  
3491             while (*s)
```

new/usr/src/uts/common/krtld/kobj.c

2

```
3492                                         *p++ = *s++;  
3493                                         /* copy the rest */  
3494                                         while (*b)  
3495                                             *p++ = *b++;  
3496                                         } else {  
3497                                             q = name;  
3498                                             while (*q)  
3499                                                 *p++ = *q++;  
3500                                         }  
3501                                         *p = 0;  
3502                                         if ((file = kobj_open_file(fullname)) != (struct _buf *)-1) {  
3503                                             kobj_free(fullname, maxpathlen);  
3504                                             return (file);  
3505                                         }  
3506                                         while (*pathp == ' ' || *pathp == ':')  
3506                                         while (*pathp == ' ')  
3507                                             pathp++;  
3508                                         if (*pathp == 0)  
3509                                             break;  
3511                                         }  
3512                                         kobj_free(fullname, maxpathlen);  
3513                                         if (_moddebug & MODDEBUG_ERRMSG) {  
3514                                             _kobj_printf(ops, "can't open %s, %s", name);  
3515                                             _kobj_printf(ops, " path is %s\n", pathpsave);  
3516                                         }  
3517                                         return ((struct _buf *)-1);  
3518 }  
_____unchanged_portion_omitted_____
```

```
*****
74163 Mon Sep  9 20:46:54 2013
new/usr/src/uts/common/os/modsysfile.c
4122 do_sysfile_cmd colon-separates the module path, and then we can't parse it
*****
```

```

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 2008 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

26 #pragma ident "%Z%%M% %I%     %E% SMI"

26 #include <sys/types.h>
27 #include <sys/inttypes.h>
28 #include <sys/param.h>
29 #include <sys/sysm.h>
30 #include <sys/user.h>
31 #include <sys/disp.h>
32 #include <sys/conf.h>
33 #include <sys/bootconf.h>
34 #include <sys/sysconf.h>
35 #include <sys/sunddi.h>
36 #include <sys/esunddi.h>
37 #include <sys/ddi_impldefs.h>
38 #include <sys/kmem.h>
39 #include <sys/vmem.h>
40 #include <sys/fs/ufs_fsdir.h>
41 #include <sys/hwconf.h>
42 #include <sys/modctl.h>
43 #include <sys/cmn_err.h>
44 #include <sys/kobj.h>
45 #include <sys/kobj_lexer.h>
46 #include <sys/errno.h>
47 #include <sys/debug.h>
48 #include <sys/autoconf.h>
49 #include <sys/callb.h>
50 #include <sys/sysmacros.h>
51 #include <sys/dacf.h>
52 #include <vm/seg_kmem.h>

54 struct hwc_class *hcl_head;      /* head of list of classes */
55 static kmutex_t hcl_lock;        /* for accessing list of classes */

57 #define DAFILE          "/etc/driver_aliases"
58 #define CLASSFILE        "/etc/driver_classes"
59 #define DACFFILE         "/etc/dacf.conf"
```

```

61 static char class_file[] = CLASSFILE;
62 static char dafile[] = DAFILE;
63 static char dacffile[] = DACFFILE;

65 char *systemfile = "/etc/system";           /* name of ascii system file */
66 static struct sysparam *sysparam_hd;          /* head of parameters list */
67 static struct sysparam *sysparam_tl;          /* tail of parameters list */
68 static vmem_t *mod_sysfile_arena;             /* parser memory */

69 char obp_bootpath[BO_MAXOBJNAME];
70 char svm_bootpath[BO_MAXOBJNAME];            /* bootpath from obp */
71                                         /* bootpath redirected via rootdev */

72 #if defined(_PSM_MODULES)
73 struct psm_mach {
74     struct psm_mach *m_next;
75     char          *m_machname;
76 };
77 unchanged portion omitted

493 static char bad_op[] = "illegal operator '%' used on a string";
494 static char colon_err[] = "A colon ':' must follow the '%' command";
495 static char tok_err[] = "Unexpected token '%'";
496 static char extra_err[] = "extraneous input ignored starting at '%'";
497 static char oversize_err[] = "value too long";

498 static struct sysparam *
499 do_sysfile_cmd(struct _buf *file, const char *cmd)
500 {
501     struct sysparam *sysp;
502     struct modcmd *mcp;
503     token_t token, op;
504     char *cp;
505     int ch;
506     char tok1[MOD_MAXPATH + 1]; /* used to read the path set by 'moddir' */
507     char tok2[64];
508
509     for (mcp = modcmd; mcp->mc_cmdname != NULL; mcp++) {
510         if (strcmp(mcp->mc_cmdname, cmd) == 0)
511             break;
512     }
513
514     sysp = vmem_alloc(mod_sysfile_arena, sizeof (struct sysparam),
515                       VM_SLEEP);
516     bzero(sysp, sizeof (struct sysparam));
517     sysp->sys_op = SETOP_NONE; /* set op to noop initially */
518
519     switch (sysp->sys_type = mcp->mc_type) {
520     case MOD_INCLUDE:
521     case MOD_EXCLUDE:
522     case MOD_FORCELOAD:
523         /*
524          * Are followed by colon.
525         */
526     case MOD_ROOTFS:
527     case MOD_SWAPFS:
528         if ((token = kobj_lexer(file, tok1, sizeof (tok1))) == COLON) {
529             token = kobj_lexer(file, tok1, sizeof (tok1));
530         } else {
531             kobj_file_err(CE_WARN, file, colon_err, cmd);
532         }
533         if (token != NAME) {
534             kobj_file_err(CE_WARN, file, "value expected");
535             goto bad;
536         }
537     }
538
539     if (sysp->sys_type == MOD_FORCELOAD) {
540         if (token == NAME) {
541             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
542                 token = kobj_lexer(file, tok1, sizeof (tok1));
543             } else {
544                 kobj_file_err(CE_WARN, file, colon_err, cmd);
545             }
546         }
547     }
548
549     if (sysp->sys_type == MOD_INCLUDE) {
550         if (token == NAME) {
551             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
552                 token = kobj_lexer(file, tok1, sizeof (tok1));
553             } else {
554                 kobj_file_err(CE_WARN, file, colon_err, cmd);
555             }
556         }
557     }
558
559     if (sysp->sys_type == MOD_EXCLUDE) {
560         if (token == NAME) {
561             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
562                 token = kobj_lexer(file, tok1, sizeof (tok1));
563             } else {
564                 kobj_file_err(CE_WARN, file, colon_err, cmd);
565             }
566         }
567     }
568
569     if (sysp->sys_type == MOD_FORCELOAD) {
570         if (token == NAME) {
571             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
572                 token = kobj_lexer(file, tok1, sizeof (tok1));
573             } else {
574                 kobj_file_err(CE_WARN, file, colon_err, cmd);
575             }
576         }
577     }
578
579     if (sysp->sys_type == MOD_ROOTFS) {
580         if (token == NAME) {
581             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
582                 token = kobj_lexer(file, tok1, sizeof (tok1));
583             } else {
584                 kobj_file_err(CE_WARN, file, colon_err, cmd);
585             }
586         }
587     }
588
589     if (sysp->sys_type == MOD_SWAPFS) {
590         if (token == NAME) {
591             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
592                 token = kobj_lexer(file, tok1, sizeof (tok1));
593             } else {
594                 kobj_file_err(CE_WARN, file, colon_err, cmd);
595             }
596         }
597     }
598
599     if (sysp->sys_type == MOD_FORCELOAD) {
600         if (token == NAME) {
601             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
602                 token = kobj_lexer(file, tok1, sizeof (tok1));
603             } else {
604                 kobj_file_err(CE_WARN, file, colon_err, cmd);
605             }
606         }
607     }
608
609     if (sysp->sys_type == MOD_INCLUDE) {
610         if (token == NAME) {
611             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
612                 token = kobj_lexer(file, tok1, sizeof (tok1));
613             } else {
614                 kobj_file_err(CE_WARN, file, colon_err, cmd);
615             }
616         }
617     }
618
619     if (sysp->sys_type == MOD_EXCLUDE) {
620         if (token == NAME) {
621             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
622                 token = kobj_lexer(file, tok1, sizeof (tok1));
623             } else {
624                 kobj_file_err(CE_WARN, file, colon_err, cmd);
625             }
626         }
627     }
628
629     if (sysp->sys_type == MOD_FORCELOAD) {
630         if (token == NAME) {
631             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
632                 token = kobj_lexer(file, tok1, sizeof (tok1));
633             } else {
634                 kobj_file_err(CE_WARN, file, colon_err, cmd);
635             }
636         }
637     }
638
639     if (sysp->sys_type == MOD_ROOTFS) {
640         if (token == NAME) {
641             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
642                 token = kobj_lexer(file, tok1, sizeof (tok1));
643             } else {
644                 kobj_file_err(CE_WARN, file, colon_err, cmd);
645             }
646         }
647     }
648
649     if (sysp->sys_type == MOD_SWAPFS) {
650         if (token == NAME) {
651             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
652                 token = kobj_lexer(file, tok1, sizeof (tok1));
653             } else {
654                 kobj_file_err(CE_WARN, file, colon_err, cmd);
655             }
656         }
657     }
658
659     if (sysp->sys_type == MOD_FORCELOAD) {
660         if (token == NAME) {
661             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
662                 token = kobj_lexer(file, tok1, sizeof (tok1));
663             } else {
664                 kobj_file_err(CE_WARN, file, colon_err, cmd);
665             }
666         }
667     }
668
669     if (sysp->sys_type == MOD_INCLUDE) {
670         if (token == NAME) {
671             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
672                 token = kobj_lexer(file, tok1, sizeof (tok1));
673             } else {
674                 kobj_file_err(CE_WARN, file, colon_err, cmd);
675             }
676         }
677     }
678
679     if (sysp->sys_type == MOD_EXCLUDE) {
680         if (token == NAME) {
681             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
682                 token = kobj_lexer(file, tok1, sizeof (tok1));
683             } else {
684                 kobj_file_err(CE_WARN, file, colon_err, cmd);
685             }
686         }
687     }
688
689     if (sysp->sys_type == MOD_FORCELOAD) {
690         if (token == NAME) {
691             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
692                 token = kobj_lexer(file, tok1, sizeof (tok1));
693             } else {
694                 kobj_file_err(CE_WARN, file, colon_err, cmd);
695             }
696         }
697     }
698
699     if (sysp->sys_type == MOD_ROOTFS) {
700         if (token == NAME) {
701             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
702                 token = kobj_lexer(file, tok1, sizeof (tok1));
703             } else {
704                 kobj_file_err(CE_WARN, file, colon_err, cmd);
705             }
706         }
707     }
708
709     if (sysp->sys_type == MOD_SWAPFS) {
710         if (token == NAME) {
711             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
712                 token = kobj_lexer(file, tok1, sizeof (tok1));
713             } else {
714                 kobj_file_err(CE_WARN, file, colon_err, cmd);
715             }
716         }
717     }
718
719     if (sysp->sys_type == MOD_FORCELOAD) {
720         if (token == NAME) {
721             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
722                 token = kobj_lexer(file, tok1, sizeof (tok1));
723             } else {
724                 kobj_file_err(CE_WARN, file, colon_err, cmd);
725             }
726         }
727     }
728
729     if (sysp->sys_type == MOD_INCLUDE) {
730         if (token == NAME) {
731             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
732                 token = kobj_lexer(file, tok1, sizeof (tok1));
733             } else {
734                 kobj_file_err(CE_WARN, file, colon_err, cmd);
735             }
736         }
737     }
738
739     if (sysp->sys_type == MOD_EXCLUDE) {
740         if (token == NAME) {
741             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
742                 token = kobj_lexer(file, tok1, sizeof (tok1));
743             } else {
744                 kobj_file_err(CE_WARN, file, colon_err, cmd);
745             }
746         }
747     }
748
749     if (sysp->sys_type == MOD_FORCELOAD) {
750         if (token == NAME) {
751             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
752                 token = kobj_lexer(file, tok1, sizeof (tok1));
753             } else {
754                 kobj_file_err(CE_WARN, file, colon_err, cmd);
755             }
756         }
757     }
758
759     if (sysp->sys_type == MOD_ROOTFS) {
760         if (token == NAME) {
761             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
762                 token = kobj_lexer(file, tok1, sizeof (tok1));
763             } else {
764                 kobj_file_err(CE_WARN, file, colon_err, cmd);
765             }
766         }
767     }
768
769     if (sysp->sys_type == MOD_SWAPFS) {
770         if (token == NAME) {
771             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
772                 token = kobj_lexer(file, tok1, sizeof (tok1));
773             } else {
774                 kobj_file_err(CE_WARN, file, colon_err, cmd);
775             }
776         }
777     }
778
779     if (sysp->sys_type == MOD_FORCELOAD) {
780         if (token == NAME) {
781             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
782                 token = kobj_lexer(file, tok1, sizeof (tok1));
783             } else {
784                 kobj_file_err(CE_WARN, file, colon_err, cmd);
785             }
786         }
787     }
788
789     if (sysp->sys_type == MOD_INCLUDE) {
790         if (token == NAME) {
791             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
792                 token = kobj_lexer(file, tok1, sizeof (tok1));
793             } else {
794                 kobj_file_err(CE_WARN, file, colon_err, cmd);
795             }
796         }
797     }
798
799     if (sysp->sys_type == MOD_EXCLUDE) {
800         if (token == NAME) {
801             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
802                 token = kobj_lexer(file, tok1, sizeof (tok1));
803             } else {
804                 kobj_file_err(CE_WARN, file, colon_err, cmd);
805             }
806         }
807     }
808
809     if (sysp->sys_type == MOD_FORCELOAD) {
810         if (token == NAME) {
811             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
812                 token = kobj_lexer(file, tok1, sizeof (tok1));
813             } else {
814                 kobj_file_err(CE_WARN, file, colon_err, cmd);
815             }
816         }
817     }
818
819     if (sysp->sys_type == MOD_ROOTFS) {
820         if (token == NAME) {
821             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
822                 token = kobj_lexer(file, tok1, sizeof (tok1));
823             } else {
824                 kobj_file_err(CE_WARN, file, colon_err, cmd);
825             }
826         }
827     }
828
829     if (sysp->sys_type == MOD_SWAPFS) {
830         if (token == NAME) {
831             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
832                 token = kobj_lexer(file, tok1, sizeof (tok1));
833             } else {
834                 kobj_file_err(CE_WARN, file, colon_err, cmd);
835             }
836         }
837     }
838
839     if (sysp->sys_type == MOD_FORCELOAD) {
840         if (token == NAME) {
841             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
842                 token = kobj_lexer(file, tok1, sizeof (tok1));
843             } else {
844                 kobj_file_err(CE_WARN, file, colon_err, cmd);
845             }
846         }
847     }
848
849     if (sysp->sys_type == MOD_INCLUDE) {
850         if (token == NAME) {
851             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
852                 token = kobj_lexer(file, tok1, sizeof (tok1));
853             } else {
854                 kobj_file_err(CE_WARN, file, colon_err, cmd);
855             }
856         }
857     }
858
859     if (sysp->sys_type == MOD_EXCLUDE) {
860         if (token == NAME) {
861             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
862                 token = kobj_lexer(file, tok1, sizeof (tok1));
863             } else {
864                 kobj_file_err(CE_WARN, file, colon_err, cmd);
865             }
866         }
867     }
868
869     if (sysp->sys_type == MOD_FORCELOAD) {
870         if (token == NAME) {
871             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
872                 token = kobj_lexer(file, tok1, sizeof (tok1));
873             } else {
874                 kobj_file_err(CE_WARN, file, colon_err, cmd);
875             }
876         }
877     }
878
879     if (sysp->sys_type == MOD_ROOTFS) {
880         if (token == NAME) {
881             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
882                 token = kobj_lexer(file, tok1, sizeof (tok1));
883             } else {
884                 kobj_file_err(CE_WARN, file, colon_err, cmd);
885             }
886         }
887     }
888
889     if (sysp->sys_type == MOD_SWAPFS) {
890         if (token == NAME) {
891             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
892                 token = kobj_lexer(file, tok1, sizeof (tok1));
893             } else {
894                 kobj_file_err(CE_WARN, file, colon_err, cmd);
895             }
896         }
897     }
898
899     if (sysp->sys_type == MOD_FORCELOAD) {
900         if (token == NAME) {
901             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
902                 token = kobj_lexer(file, tok1, sizeof (tok1));
903             } else {
904                 kobj_file_err(CE_WARN, file, colon_err, cmd);
905             }
906         }
907     }
908
909     if (sysp->sys_type == MOD_INCLUDE) {
910         if (token == NAME) {
911             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
912                 token = kobj_lexer(file, tok1, sizeof (tok1));
913             } else {
914                 kobj_file_err(CE_WARN, file, colon_err, cmd);
915             }
916         }
917     }
918
919     if (sysp->sys_type == MOD_EXCLUDE) {
920         if (token == NAME) {
921             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
922                 token = kobj_lexer(file, tok1, sizeof (tok1));
923             } else {
924                 kobj_file_err(CE_WARN, file, colon_err, cmd);
925             }
926         }
927     }
928
929     if (sysp->sys_type == MOD_FORCELOAD) {
930         if (token == NAME) {
931             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
932                 token = kobj_lexer(file, tok1, sizeof (tok1));
933             } else {
934                 kobj_file_err(CE_WARN, file, colon_err, cmd);
935             }
936         }
937     }
938
939     if (sysp->sys_type == MOD_ROOTFS) {
940         if (token == NAME) {
941             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
942                 token = kobj_lexer(file, tok1, sizeof (tok1));
943             } else {
944                 kobj_file_err(CE_WARN, file, colon_err, cmd);
945             }
946         }
947     }
948
949     if (sysp->sys_type == MOD_SWAPFS) {
950         if (token == NAME) {
951             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
952                 token = kobj_lexer(file, tok1, sizeof (tok1));
953             } else {
954                 kobj_file_err(CE_WARN, file, colon_err, cmd);
955             }
956         }
957     }
958
959     if (sysp->sys_type == MOD_FORCELOAD) {
960         if (token == NAME) {
961             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
962                 token = kobj_lexer(file, tok1, sizeof (tok1));
963             } else {
964                 kobj_file_err(CE_WARN, file, colon_err, cmd);
965             }
966         }
967     }
968
969     if (sysp->sys_type == MOD_INCLUDE) {
970         if (token == NAME) {
971             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
972                 token = kobj_lexer(file, tok1, sizeof (tok1));
973             } else {
974                 kobj_file_err(CE_WARN, file, colon_err, cmd);
975             }
976         }
977     }
978
979     if (sysp->sys_type == MOD_EXCLUDE) {
980         if (token == NAME) {
981             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
982                 token = kobj_lexer(file, tok1, sizeof (tok1));
983             } else {
984                 kobj_file_err(CE_WARN, file, colon_err, cmd);
985             }
986         }
987     }
988
989     if (sysp->sys_type == MOD_FORCELOAD) {
990         if (token == NAME) {
991             if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
992                 token = kobj_lexer(file, tok1, sizeof (tok1));
993             } else {
994                 kobj_file_err(CE_WARN, file, colon_err, cmd);
995             }
996         }
997     }
998
999     if (sysp->sys_type == MOD_ROOTFS) {
1000        if (token == NAME) {
1001            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1002                token = kobj_lexer(file, tok1, sizeof (tok1));
1003            } else {
1004                kobj_file_err(CE_WARN, file, colon_err, cmd);
1005            }
1006        }
1007    }
1008
1009    if (sysp->sys_type == MOD_SWAPFS) {
1010        if (token == NAME) {
1011            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1012                token = kobj_lexer(file, tok1, sizeof (tok1));
1013            } else {
1014                kobj_file_err(CE_WARN, file, colon_err, cmd);
1015            }
1016        }
1017    }
1018
1019    if (sysp->sys_type == MOD_FORCELOAD) {
1020        if (token == NAME) {
1021            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1022                token = kobj_lexer(file, tok1, sizeof (tok1));
1023            } else {
1024                kobj_file_err(CE_WARN, file, colon_err, cmd);
1025            }
1026        }
1027    }
1028
1029    if (sysp->sys_type == MOD_INCLUDE) {
1030        if (token == NAME) {
1031            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1032                token = kobj_lexer(file, tok1, sizeof (tok1));
1033            } else {
1034                kobj_file_err(CE_WARN, file, colon_err, cmd);
1035            }
1036        }
1037    }
1038
1039    if (sysp->sys_type == MOD_EXCLUDE) {
1040        if (token == NAME) {
1041            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1042                token = kobj_lexer(file, tok1, sizeof (tok1));
1043            } else {
1044                kobj_file_err(CE_WARN, file, colon_err, cmd);
1045            }
1046        }
1047    }
1048
1049    if (sysp->sys_type == MOD_FORCELOAD) {
1050        if (token == NAME) {
1051            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1052                token = kobj_lexer(file, tok1, sizeof (tok1));
1053            } else {
1054                kobj_file_err(CE_WARN, file, colon_err, cmd);
1055            }
1056        }
1057    }
1058
1059    if (sysp->sys_type == MOD_ROOTFS) {
1060        if (token == NAME) {
1061            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1062                token = kobj_lexer(file, tok1, sizeof (tok1));
1063            } else {
1064                kobj_file_err(CE_WARN, file, colon_err, cmd);
1065            }
1066        }
1067    }
1068
1069    if (sysp->sys_type == MOD_SWAPFS) {
1070        if (token == NAME) {
1071            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1072                token = kobj_lexer(file, tok1, sizeof (tok1));
1073            } else {
1074                kobj_file_err(CE_WARN, file, colon_err, cmd);
1075            }
1076        }
1077    }
1078
1079    if (sysp->sys_type == MOD_FORCELOAD) {
1080        if (token == NAME) {
1081            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1082                token = kobj_lexer(file, tok1, sizeof (tok1));
1083            } else {
1084                kobj_file_err(CE_WARN, file, colon_err, cmd);
1085            }
1086        }
1087    }
1088
1089    if (sysp->sys_type == MOD_INCLUDE) {
1090        if (token == NAME) {
1091            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1092                token = kobj_lexer(file, tok1, sizeof (tok1));
1093            } else {
1094                kobj_file_err(CE_WARN, file, colon_err, cmd);
1095            }
1096        }
1097    }
1098
1099    if (sysp->sys_type == MOD_EXCLUDE) {
1100        if (token == NAME) {
1101            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1102                token = kobj_lexer(file, tok1, sizeof (tok1));
1103            } else {
1104                kobj_file_err(CE_WARN, file, colon_err, cmd);
1105            }
1106        }
1107    }
1108
1109    if (sysp->sys_type == MOD_FORCELOAD) {
1110        if (token == NAME) {
1111            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1112                token = kobj_lexer(file, tok1, sizeof (tok1));
1113            } else {
1114                kobj_file_err(CE_WARN, file, colon_err, cmd);
1115            }
1116        }
1117    }
1118
1119    if (sysp->sys_type == MOD_ROOTFS) {
1120        if (token == NAME) {
1121            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1122                token = kobj_lexer(file, tok1, sizeof (tok1));
1123            } else {
1124                kobj_file_err(CE_WARN, file, colon_err, cmd);
1125            }
1126        }
1127    }
1128
1129    if (sysp->sys_type == MOD_SWAPFS) {
1130        if (token == NAME) {
1131            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1132                token = kobj_lexer(file, tok1, sizeof (tok1));
1133            } else {
1134                kobj_file_err(CE_WARN, file, colon_err, cmd);
1135            }
1136        }
1137    }
1138
1139    if (sysp->sys_type == MOD_FORCELOAD) {
1140        if (token == NAME) {
1141            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1142                token = kobj_lexer(file, tok1, sizeof (tok1));
1143            } else {
1144                kobj_file_err(CE_WARN, file, colon_err, cmd);
1145            }
1146        }
1147    }
1148
1149    if (sysp->sys_type == MOD_INCLUDE) {
1150        if (token == NAME) {
1151            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1152                token = kobj_lexer(file, tok1, sizeof (tok1));
1153            } else {
1154                kobj_file_err(CE_WARN, file, colon_err, cmd);
1155            }
1156        }
1157    }
1158
1159    if (sysp->sys_type == MOD_EXCLUDE) {
1160        if (token == NAME) {
1161            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1162                token = kobj_lexer(file, tok1, sizeof (tok1));
1163            } else {
1164                kobj_file_err(CE_WARN, file, colon_err, cmd);
1165            }
1166        }
1167    }
1168
1169    if (sysp->sys_type == MOD_FORCELOAD) {
1170        if (token == NAME) {
1171            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1172                token = kobj_lexer(file, tok1, sizeof (tok1));
1173            } else {
1174                kobj_file_err(CE_WARN, file, colon_err, cmd);
1175            }
1176        }
1177    }
1178
1179    if (sysp->sys_type == MOD_ROOTFS) {
1180        if (token == NAME) {
1181            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1182                token = kobj_lexer(file, tok1, sizeof (tok1));
1183            } else {
1184                kobj_file_err(CE_WARN, file, colon_err, cmd);
1185            }
1186        }
1187    }
1188
1189    if (sysp->sys_type == MOD_SWAPFS) {
1190        if (token == NAME) {
1191            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1192                token = kobj_lexer(file, tok1, sizeof (tok1));
1193            } else {
1194                kobj_file_err(CE_WARN, file, colon_err, cmd);
1195            }
1196        }
1197    }
1198
1199    if (sysp->sys_type == MOD_FORCELOAD) {
1200        if (token == NAME) {
1201            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1202                token = kobj_lexer(file, tok1, sizeof (tok1));
1203            } else {
1204                kobj_file_err(CE_WARN, file, colon_err, cmd);
1205            }
1206        }
1207    }
1208
1209    if (sysp->sys_type == MOD_INCLUDE) {
1210        if (token == NAME) {
1211            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1212                token = kobj_lexer(file, tok1, sizeof (tok1));
1213            } else {
1214                kobj_file_err(CE_WARN, file, colon_err, cmd);
1215            }
1216        }
1217    }
1218
1219    if (sysp->sys_type == MOD_EXCLUDE) {
1220        if (token == NAME) {
1221            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1222                token = kobj_lexer(file, tok1, sizeof (tok1));
1223            } else {
1224                kobj_file_err(CE_WARN, file, colon_err, cmd);
1225            }
1226        }
1227    }
1228
1229    if (sysp->sys_type == MOD_FORCELOAD) {
1230        if (token == NAME) {
1231            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1232                token = kobj_lexer(file, tok1, sizeof (tok1));
1233            } else {
1234                kobj_file_err(CE_WARN, file, colon_err, cmd);
1235            }
1236        }
1237    }
1238
1239    if (sysp->sys_type == MOD_ROOTFS) {
1240        if (token == NAME) {
1241            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1242                token = kobj_lexer(file, tok1, sizeof (tok1));
1243            } else {
1244                kobj_file_err(CE_WARN, file, colon_err, cmd);
1245            }
1246        }
1247    }
1248
1249    if (sysp->sys_type == MOD_SWAPFS) {
1250        if (token == NAME) {
1251            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1252                token = kobj_lexer(file, tok1, sizeof (tok1));
1253            } else {
1254                kobj_file_err(CE_WARN, file, colon_err, cmd);
1255            }
1256        }
1257    }
1258
1259    if (sysp->sys_type == MOD_FORCELOAD) {
1260        if (token == NAME) {
1261            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1262                token = kobj_lexer(file, tok1, sizeof (tok1));
1263            } else {
1264                kobj_file_err(CE_WARN, file, colon_err, cmd);
1265            }
1266        }
1267    }
1268
1269    if (sysp->sys_type == MOD_INCLUDE) {
1270        if (token == NAME) {
1271            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1272                token = kobj_lexer(file, tok1, sizeof (tok1));
1273            } else {
1274                kobj_file_err(CE_WARN, file, colon_err, cmd);
1275            }
1276        }
1277    }
1278
1279    if (sysp->sys_type == MOD_EXCLUDE) {
1280        if (token == NAME) {
1281            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1282                token = kobj_lexer(file, tok1, sizeof (tok1));
1283            } else {
1284                kobj_file_err(CE_WARN, file, colon_err, cmd);
1285            }
1286        }
1287    }
1288
1289    if (sysp->sys_type == MOD_FORCELOAD) {
1290        if (token == NAME) {
1291            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1292                token = kobj_lexer(file, tok1, sizeof (tok1));
1293            } else {
1294                kobj_file_err(CE_WARN, file, colon_err, cmd);
1295            }
1296        }
1297    }
1298
1299    if (sysp->sys_type == MOD_ROOTFS) {
1300        if (token == NAME) {
1301            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1302                token = kobj_lexer(file, tok1, sizeof (tok1));
1303            } else {
1304                kobj_file_err(CE_WARN, file, colon_err, cmd);
1305            }
1306        }
1307    }
1308
1309    if (sysp->sys_type == MOD_SWAPFS) {
1310        if (token == NAME) {
1311            if (kobj_lexer(file, tok1, sizeof (tok1)) == COLON) {
1312                token = kobj_lexer(file, tok1, sizeof (tok1));
1313
```

new/usr/src/uts/common/os/modsysfile.c

3

```

538
539     cp = tok1 + strlen(tok1);
540     while ((ch = kobj_getc(file)) != -1 && !iswhite(ch) &&
541           !isnewline(ch)) {
542         if (cp - tok1 >= sizeof (tok1) - 1) {
543             kobj_file_err(CE_WARN, file, oversize_err);
544             goto bad;
545         }
546         *cp++ = (char)ch;
547     }
548     *cp = '\0';
549
550     if (ch != -1)
551         (void) kobj_ungetc(file);
552     if (sysp->sys_type == MOD_INCLUDE)
553         return (NULL);
554     sysp->sys_ptr = vmem_alloc(mod_sysfile_arena, strlen(tok1) + 1,
555                                VM_SLEEP);
556     (void) strcpy(sysp->sys_ptr, tok1);
557     break;
558 case MOD_SET:
559 case MOD_SET64:
560 case MOD_SET32:
561 {
562     char *var;
563     token_t tok3;
564
565     if (kobj_lex(file, tok1, sizeof (tok1)) != NAME) {
566         kobj_file_err(CE_WARN, file, "value expected");
567         goto bad;
568     }
569
570     /*
571      * If the next token is a colon (:),
572      * we have the <modname>:<variable> construct.
573      */
574     if ((token = kobj_lex(file, tok2, sizeof (tok2))) == COLON) {
575         if ((token = kobj_lex(file, tok2,
576                               sizeof (tok2))) == NAME) {
577             var = tok2;
578             /*
579              * Save the module name.
580              */
581             sysp->sys_modnam = vmem_alloc(mod_sysfile_arena
582                                           strlen(tok1) + 1, VM_SLEEP);
583             (void) strcpy(sysp->sys_modnam, tok1);
584         } else {
585             op = kobj_lex(file, tok1, sizeof (tok1));
586             kobj_file_err(CE_WARN, file, "value expected");
587             goto bad;
588         }
589     } else {
590         /* otherwise, it was the op */
591         var = tok1;
592         op = token;
593     }
594     /*
595      * kernel param - place variable name in sys_ptr.
596      */
597     sysp->sys_ptr = vmem_alloc(mod_sysfile_arena, strlen(var) + 1,
598                                VM_SLEEP);
599     (void) strcpy(sysp->sys_ptr, var);
600     /* set operation */
601     switch (op) {
602     case EQUALS:

```

new/usr/src/uts/common/os/modsysfile.c

```

/* simple assignment */
sysp->sys_op = SETOP_ASSIGN;
break;
case AMPERSAND:
    /* bitwise AND */
    sysp->sys_op = SETOP_AND;
    break;
case BIT_OR:
    /* bitwise OR */
    sysp->sys_op = SETOP_OR;
    break;
default:
    /* unsupported operation */
    kobj_file_err(CE_WARN, file,
                  "unsupported operator %s", tok2);
    goto bad;
}

switch ((tok3 = kobj_lex(file, tok1, sizeof (tok1)))) {
case STRING:
    /* string variable */
    if (sysp->sys_op != SETOP_ASSIGN) {
        kobj_file_err(CE_WARN, file, bad_op, tok1);
        goto bad;
    }
    if (kobj_get_string(&sysp->sys_info, tok1) == 0) {
        kobj_file_err(CE_WARN, file, "string garbled");
        goto bad;
    }
    /*
     * Set SYSPARAM_STR_TOKEN in sys_flags to notify
     * sysparam_print_warning() that this is a string
     * token.
     */
    sysp->sys_flags |= SYSPARAM_STR_TOKEN;
    break;
case HEXVAL:
case DECVAL:
    if (kobj_getvalue(tok1, &sysp->sys_info) == -1) {
        kobj_file_err(CE_WARN, file,
                      "invalid number '%s'", tok1);
        goto bad;
    }
    /*
     * Set the appropriate flag (hexadecimal or decimal)
     * in sys_flags for sysparam_print_warning() to be
     * able to print the number with the correct format.
     */
    if (tok3 == HEXVAL) {
        sysp->sys_flags |= SYSPARAM_HEX_TOKEN;
    } else {
        sysp->sys_flags |= SYSPARAM_DEC_TOKEN;
    }
    break;
default:
    kobj_file_err(CE_WARN, file, "bad rvalue '%s'", tok1);
    goto bad;
} /* end switch */

/*
 * Now that we've parsed it to check the syntax, consider
 * discarding it (because it -doesn't- apply to this flavor
 * of the kernel)
*/

```

new/usr/src/uts/common/os/modsysfile.c

```

668     if (sysp->sys_type == MOD_SET32)
669         return (NULL);
670 #else
671     if (sysp->sys_type == MOD_SET64)
672         return (NULL);
673 #endif
674     sysp->sys_type = MOD_SET;
675     break;
676 }
677 case MOD_MODDIR:
678     if ((token = kobj_lex(file, tok1, sizeof (tok1))) != COLON) {
679         kobj_file_err(CE_WARN, file, colon_err, cmd);
680         goto bad;
681     }

683     cp = tok1;
684     while ((token = kobj_lex(file, cp,
685         sizeof (tok1) - (cp - tok1))) != NEWLINE && token != EOF) {
686         if (token == -1) {
687             kobj_file_err(CE_WARN, file, oversize_err);
688             goto bad;
689         }
690         cp += strlen(cp);
691         while ((ch = kobj_getc(file)) != -1 && !iswhite(ch) &&
692             !isnewline(ch) && ch != ':') {
693             if (cp - tok1 >= sizeof (tok1) - 1) {
694                 kobj_file_err(CE_WARN, file,
695                             oversize_err);
696                 goto bad;
697             }
698             *cp++ = (char)ch;
699         }
700         *cp++ = ',';
701         *cp++ = ':';
702         if (isnewline(ch)) {
703             cp--;
704             (void) kobj_ungetc(file);
705         }
706     (void) kobj_ungetc(file);
707     *cp = '\0';
708     sysp->sys_ptr = vmem_alloc(mod_sysfile_arena, strlen(tok1) + 1,
709         VM_SLEEP);
710     (void) strcpy(sysp->sys_ptr, tok1);
711     break;

713 case MOD_SWAPDEV:
714 case MOD_ROOTDEV:
715     if ((token = kobj_lex(file, tok1, sizeof (tok1))) != COLON) {
716         kobj_file_err(CE_WARN, file, colon_err, cmd);
717         goto bad;
718     }
719     while ((ch = kobj_getc(file)) == ' ' || ch == '\t')
720         ;
721     cp = tok1;
722     while (!iswhite(ch) && !isnewline(ch) && ch != -1) {
723         if (cp - tok1 >= sizeof (tok1) - 1) {
724             kobj_file_err(CE_WARN, file, oversize_err);
725             goto bad;
726         }

727         *cp++ = (char)ch;
728         ch = kobj_getc(file);
729     }
730     if (ch != -1)
731         (void) kobj_ungetc(file);
732 }
```

new/usr/src/uts/common/os/modsysfile.

```
733         *cp = '\0';
735         sysp->sys_ptr = vmem_alloc(mod_sysfile_arena, strlen(tok1) + 1,
736                                     VM_SLEEP);
737         (void) strcpy(sysp->sys_ptr, tok1);
738         break;
740     case MOD_UNKNOWN:
741     default:
742         kobj_file_err(CE_WARN, file, "unknown command '%s'", cmd);
743         goto bad;
744     }
746     return (sysp);
748 bad:
749     kobj_find_eol(file);
750     return (NULL);
751 }
```

unchanged portion omitted