

new/usr/src/cmd/hal/hald/solaris/devinfo\_storage.c

1

```
*****
51356 Mon May  5 22:34:22 2014
new/usr/src/cmd/hal/hald/solaris/devinfo_storage.c
4846 HAL partition names don't match real partition names
*****
1 / ****
2 *
3 * devinfo_storage.c : storage devices
4 *
5 * Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
6 * Copyright 2013 Garrett D'Amore <garrett@damore.org>
7 * Copyright 2014 Andrew Stormont.
8 #endif /* ! codereview */
9 *
10 * Licensed under the Academic Free License version 2.1
11 *
12 ****
14 #ifdef HAVE_CONFIG_H
15 # include <config.h>
16 #endif

18 #include <stdio.h>
19 #include <string.h>
20 #include <strings.h>
21 #include <ctype.h>
22 #include <libdevinfo.h>
23 #include <sys/types.h>
24 #include <sys/mkdev.h>
25 #include <sys/stat.h>
26 #include <sys/mntent.h>
27 #include <sys/mnttab.h>

29 #include "../osspec.h"
30 #include "../logger.h"
31 #include "../hald.h"
32 #include "../hald_dbus.h"
33 #include "../device_info.h"
34 #include "../util.h"
35 #include "../hald_runner.h"
36 #include "hotplug.h"
37 #include "devinfo.h"
38 #include "devinfo_misc.h"
39 #include "devinfo_storage.h"
40 #include "osspec_solaris.h"

42 #ifdef sparc
43 #define WHOLE_DISK      "s2"
44 #else
45 #define WHOLE_DISK      "p0"
46 #endif

48 /* some devices,especially CDROMs, may take a while to be probed (values in ms)
49 #define DEVINFO_PROBE_STORAGE_TIMEOUT    60000
50 #define DEVINFO_PROBE_VOLUME_TIMEOUT     60000

52 typedef struct devinfo_storage_minor {
53     char      *devpath;
54     char      *devlink;
55     char      *slice;
56     dev_t     dev;
57     int       dosnum; /* dos disk number or -1 */
58 } devinfo_storage_minor_t;

60 HalDevice *devinfo_ide_add(HalDevice *parent, di_node_t node, char *devfs_path,
61 static HalDevice *devinfo_ide_host_add(HalDevice *parent, di_node_t node, char *
```

new/usr/src/cmd/hal/hald/solaris/devinfo\_storage.c

2

```
62 static HalDevice *devinfo_ide_device_add(HalDevice *parent, di_node_t node, char
63 static HalDevice *devinfo_ide_storage_add(HalDevice *parent, di_node_t node, cha
64 HalDevice *devinfo_scsi_add(HalDevice *parent, di_node_t node, char *devfs_path,
65 static HalDevice *devinfo_scsi_storage_add(HalDevice *parent, di_node_t node, ch
66 HalDevice *devinfo_blkdev_add(HalDevice *parent, di_node_t node, char *devfs_pat
67 static HalDevice *devinfo_blkdev_storage_add(HalDevice *parent, di_node_t node,
68 HalDevice *devinfo_floppy_add(HalDevice *parent, di_node_t node, char *devfs_pat
69 static void devinfo_floppy_add_volume(HalDevice *parent, di_node_t node);
70 static HalDevice *devinfo_lofi_add(HalDevice *parent, di_node_t node, char *devf
71 static void devinfo_lofi_add_minor(HalDevice *parent, di_node_t node, char *mino
72 static void devinfo_storage_minors(HalDevice *parent, di_node_t node, gchar *dev
73 static struct devinfo_storage_minor *devinfo_storage_new_minor(char *maindev_pat
74     char *devlink, dev_t dev, int dosnum);
75 static void devinfo_storage_free_minor(struct devinfo_storage_minor *m);
76 HalDevice *devinfo_volume_add(HalDevice *parent, di_node_t node, devinfo_storage
77 static void devinfo_volume_preprobing_done(HalDevice *d, gpointer userdata1, gpo
78 static void devinfo_volume_hotplug_begin_add (HalDevice *d, HalDevice *parent, D
79 static void devinfo_storage_hotplug_begin_add (HalDevice *d, HalDevice *parent,
80 static void devinfo_storage_probing_done (HalDevice *d, guint32 exit_type, gint
81 const gchar *devinfo_volume_get_prober (HalDevice *d, int *timeout);
82 const gchar *devinfo_storage_get_prober (HalDevice *d, int *timeout);

84 static char *devinfo_scsi_dtype2str(int dtype);
85 static char *devinfo_volume_get_slice_name (char *devlink);
86 static boolean_t is_dos_slice(const char *slice, int *partnum);
7 static gboolean dos_to_dev(char *path, char **devpath, int *partnum);
8 static gboolean is_dos_path(char *path, int *partnum);

88 static void devinfo_storage_set_nicknames (HalDevice *d);

90 DevinfoDevHandler devinfo_ide_handler = {
91     devinfo_ide_add,
92     NULL,
93     NULL,
94     NULL,
95     NULL,
96     NULL
97 };
unchanged_portion_omitted

796 /*
797 * Storage minor nodes are potential "volume" objects.
798 * This function also completes building the parent object (main storage device)
799 */
800 static void
801 devinfo_storage_minors(HalDevice *parent, di_node_t node, gchar *devfs_path, gbo
802 {
803     di_devlink_handle_t devlink_hdl;
804     gboolean is_cdrom;
805     const char *whole_disk;
806     int major;
807     di_minor_t minor;
808     dev_t dev;
809     char *minor_path = NULL;
810     char *maindev_path = NULL;
811     char *devpath, *devlink;
812     int doslink_len;
813     char *doslink;
814     char dospath[64];
815     char *slice;
816     int pathlen;
817     int i;
818     char *raw;
819     boolean_t maindev_is_d0;
820     GQueue *mq;
821     HalDevice *volume;
```

new/usr/src/cmd/hal/hald/solaris/devinfo\_storage.c

3

```

822 struct devinfo_storage_minor *m;
823 struct devinfo_storage_minor *maindev = NULL;
824
825 /* for cdroms whole disk is always s2 */
826 is_cdrom = hal_device_has_capability (parent, "storage.cdrom");
827 whole_disk = is_cdrom ? "s2" : WHOLE_DISK;
828
829 major = di_driver_major(node);
830
831 /* the "whole disk" p0/s2/d0 node must come first in the hotplug queue
832 * so we put other minor nodes on the local queue and move to the
833 * hotplug queue up in the end
834 */
835 if ((mq = g_queue_new()) == NULL) {
836     goto err;
837 }
838 if ((devlink_hdl = di_devlink_init(NULL, 0)) == NULL) {
839     g_queue_free (mq);
840     goto err;
841 }
842 minor = DI_MINOR_NIL;
843 while ((minor = di_minor_next(node, minor)) != DI_MINOR_NIL) {
844     dev = di_minor_devt(minor);
845     if ((major != major(dev)) ||
846         (di_minor_type(minor) != DDM_MINOR) ||
847         (di_minor_spectype(minor) != S_IFBLK) ||
848         ((minor_path = di_devfs_minor_path(minor)) == NULL)) {
849         continue;
850     }
851     if ((devlink = get_devlink(devlink_hdl, NULL, minor_path)) == NULL)
852         di_devfs_path_free (minor_path);
853         continue;
854 }
855
856 slice = devinfo_volume_get_slice_name (devlink);
857 if (strlen (slice) < 2) {
858     free (devlink);
859     di_devfs_path_free (minor_path);
860     continue;
861 }
862
863 /* ignore p1..N - we'll use p0:N instead */
864 if ((strlen (slice) > 1) && (slice[0] == 'p') && isdigit(slice[1])
865     ((atol(&slice[1])) > 0)) {
866     free (devlink);
867     di_devfs_path_free (minor_path);
868     continue;
869 }
870
871 m = devinfo_storage_new_minor(minor_path, slice, devlink, dev, -
872 if (m == NULL) {
873     free (devlink);
874     di_devfs_path_free (minor_path);
875     continue;
876 }
877
878 /* main device is either s2/p0 or d0, the latter taking precedence
879 if ((strcmp (slice, "d0") == 0) ||
880     (((strcmp (slice, whole_disk) == 0) && (maindev == NULL))) {
881     if (maindev_path != NULL) {
882         di_devfs_path_free (maindev_path);
883     }
884     maindev_path = minor_path;
885     maindev = m;
886     g_queue_push_head (mq, maindev);
887 }
888 }
```

[new/usr/src/cmd/hal/hald/solaris/devinfo\\_storage.c](#)

```

932             } else {
933                 HAL_INFO(("rescan volume exists %s", m->devpath)
934             }
935         } else {
936             devinfo_volume_add (parent, node, m);
937         }
938         devinfo_storage_free_minor (m);
939     }
940     if (maindev_path != NULL) {
941         di_devfs_path_free (maindev_path);
942     }
943
944     return;
945
946 err:
947     if (maindev_path != NULL) {
948         di_devfs_path_free (maindev_path);
949     }
950     if (!rescan) {
951         devinfo_add_enqueue (parent, devfs_path, &devinfo_storage_handle
952     }
953
954 }
```

unchanged\_portion\_omitted

```

1023 static void
1024 devinfo_volume_preprobe_done (HalDevice *d, gpointer userdatal1, gpointer userd
1025 {
1026     void *end_token = (void *) userdatal1;
1027     char *whole_disk;
1028     char *block_device;
1029     const char *storage_ud1;
1030     HalDevice *storage_d;
1031     const char *slice;
1032     int dos_num;
1033
1034     if (hal_device_property_get_bool (d, "info.ignore")) {
1035         HAL_INFO(("Preprobe merged info.ignore==TRUE %s", hal_device_
1036         goto skip;
1037     }
1038
1039     /*
1040      * Optimizations: only probe if there's a chance to find something
1041      */
1042     block_device = (char *)hal_device_property_get_string (d, "block.device"
1043     storage_ud1 = hal_device_property_get_string (d, "block.storage_device")
1044     slice = hal_device_property_get_string(d, "block.solaris.slice");
1045     if ((block_device == NULL) || (storage_ud1 == NULL) ||
1046         (slice == NULL) || (strlen (slice) < 2)) {
1047         HAL_INFO(("Malformed volume properties %s", hal_device_get_udi
1048         goto skip;
1049     }
1050     storage_d = hal_device_store_match_key_value_string (hal_device_get_gdl (), "i
1051     if (storage_d == NULL) {
1052         HAL_INFO(("Storage device not found %s", hal_device_get_udi (d)
1053         goto skip;
1054     }
1055
1056     whole_disk = hal_device_has_capability (storage_d,
1057         "storage.cdrom") ? "s2" : WHOLE_DISK;
1058
1059     if (is_dos_slice(slice, &dos_num)) {
1060         if (is_dos_path(block_device, &dos_num)) {
1061             /* don't probe more dos volumes than probe-storage found */
1062             if ((hal_device_property_get_bool (storage_d, "storage.no_partit
1063                 (dos_num > hal_device_property_get_int (storage_d, "storage
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
20100
20101
20102
20103
20104
20105
20106
20107
20108
20109
20110
20111
20112
20113
20114
20115
20116
20117
20118
20119
20120
20121
20122
20123
20124
20125
20126
20127
20128
20129
20130
20131
20132
20133
20134
20135
20136
20137
20138
20139
20140
20141
20142
20143
20144
20145
20146
20147
20148
20149
20150
20151
20152
20153
20154
20155
20156
20157
20158
20159
20160
20161
20162
20163
20164
20165
20166
20167
20168
20169
20170
20171
20172
20173
20174
20175
20176
20177
20178
20179
20180
20181
20182
20183
20184
20185
20186
20187
20188
20189
20190
20191
20192
20193
20194
20195
20196
20197
20198
20199
20200
20201
20202
20203
20204
20205
20206
20207
20208
20209
20210
20211
20212
20213
20214
20215
20216
20217
20218
20219
20220
20221
20222
20223
20224
20225
20226
20227
20228
20229
20230
20231
20232
20233
20234
20235
20236
20237
20238
20239
20240
20241
20242
20243
20244
20245
20246
20247
20248
20249
20250
20251
20252
20253
20254
20255
20256
20257
20258
20259
20259
20260
20261
20262
20263
20264
20265
20266
20267
20268
20269
20270
20271
20272
20273
20274
20275
20276
20277
20278
20279
20280
20281
20282
20283
20284
20285
20286
20287
20288
20289
20290
20291
20292
20293
20294
20295
20296
20297
20298
20299
20299
20300
20301
20302
20303
20304
20305
20306
20307
20308
20309
20309
20310
20311
20312
20313
20314
20315
20316
20317
20318
20319
20319
20320
20321
20322
20323
20324
20325
20326
20327
20328
20329
20329
20330
20331
20332
20333
20334
20335
20336
20337
20338
20339
20339
20340
20341
20342
20343
20344
20345
20346
20347
20348
20349
20349
20350
20351
20352
20353
20354
20355
20356
20357
20358
20359
20359
20360
20361
20362
20363
20364
20365
20366
20367
20368
20369
20369
20370
20371
20372
20373
20374
20375
20376
20377
20378
20379
20379
20380
20381
20382
20383
20384
20385
20386
20387
20388
20389
20389
20390
20391
20392
20393
20394
20395
20396
20397
20398
20399
20399
20400
20401
20402
20403
20404
20405
20406
20407
20408
20409
20409
20410
20411
20412
20413
20414
20415
20416
20417
20418
20419
20419
20420
20421
20422
20423
20424
20425
20426
20427
20428
20429
20429
20430
20431
20432
20433
20434
20435
20436
20437
20438
20439
20439
20440
20441
20442
20443
20444
20445
20446
20447
20448
20449
20449
20450
20451
20452
20453
20454
20455
20456
20457
20458
20459
20459
20460
20461
20462
20463
20464
20465
20466
20467
20468
20469
20469
20470
20471
20472
20473
20474
20475
20476
20477
20478
20479
20479
20480
20481
20482
20483
20484
20485
20486
20487
20488
20489
20489
20490
20491
20492
20493
20494
20495
20496
20497
20498
20499
20499
20500
20501
20502
20503
20504
20505
20506
20507
20508
20509
20509
20510
20511
20512
20513
20514
20515
20516
20517
20518
20519
20519
20520
20521
20522
20523
20524
20525
20526
20527
20528
20529
20529
20530
20531
20532
20533
20534
20535
20536
20537
20538
20539
20539
20540
20541
20542
20543
20544
20545
20546
20547
20548
20549
20549
20550
20551
20552
20553
20554
20555
20556
20557
20558
20559
20559
20560
20561
20562
20563
20564
20565
20566
20567
20568
20569
20569
20570
20571
20572
20573
20574
20575
20576
20577
20578
20579
20579
20580
20581
20582
20583
20584
20585
20586
20587
20588
20589
20589
20590
20591
20592
20593
20594
20595
20596
20597
20598
20599
20599
20600
20601
20602
20603
20604
20605
20606
20607
20608
20609
20609
20610
20611
20612
20613
20614
20615
20616
20617
20618
20619
20619
20620
20621
20622
20623
20624
20625
20626
20627
20628
20629
20629
20630
20631
20632
20633
20634
20635
20636
20637
20638
20639
20639
20640
20641
20642
20643
20644
20645
20646
20647
20648
20649
20649
20650
20651
20652
20653
20654
20655
20656
20657
20658
20659
20659
20660
20661
20662
20663
20664
20665
20666
20667
20668
20669
20669
20670
20671
20672
20673
20674
20675
20676
20677
20678
20679
20679
20680
20681
20682
20683
20684
20685
20686
20687
20688
20689
20689
20690
20691
20692
20693
20694
20695
20696
20697
20698
20699
20699
20700
20701
20702
20703
20704
20705
20706
20707
20708
20709
20709
20710
20711
20712
20713
20714
20715
20716
20717
20718
20719
20719
20720
20721
20722
20723
20724
20725
20726
20727
20728
20729
20729
20730
20731
20732
20733
20734
20735
20736
20737
20738
20739
20739
20740
20741
20742
20743
20744
20745
20746
20747
20748
20749
20749
20750
20751
20752
20753
20754
20755
20756
20757
20758
20759
20759
20760
20761
20762
20763
20764
20765
20766
20767
20768
20769
20769
20770
20771
20772
20773
20774
20775
20776
20777
20778
20779
20779
20780
20781
20782
20783
20784
20785
20786
20787
20788
20789
20789
20790
20791
20792
20793
20794
20795
20796
20797
20798
20799
20799
20800
20801
20802
20803
20804
20805
20806
20807
20808
20809
20809
20810
20811
20812
20813
20814
20815
20816
20817
20818
20819
20819
20820
20821
20822
20823
20824
20825
20826
20827
20828
20829
20829
20830
20831
20832
20833
20834
20835
20836
20837
20838
20839
20839
20840
20841
20842
20843
20844
20845
20846
20847
20848
20849
20849
20850
20851
20852
20853
20854
20855
20856
20857
20858
20859
20859
20860
20861
20862
20863
20864
20865
20866
20867
20868
20869
20869
20870
20871
20872
20873
20874
20875
20876
20877
20878
20879
20879
20880
20881
20882
20883
20884
20885
20886
20887
20888
20889
20889
20890
20891
20892
20893
20894
20895
20896
20897
20898
20899
20899
20900
20901
20902
20903
20904
20905
20906
20907
20908
20909
20909
20910
20911
20912
20913
20914
20915
20916
20917
20918
20919
20
```

```
1396             gint return_code, gchar **error,
1397             gpointer data1, gpointer data2)
1398 {
1399     char *mount_point = (char *) data1;
1401     HAL_INFO (("Cleaned up mount point '%s'", mount_point));
1402     g_free (mount_point);
1403 }
```

unchanged portion omitted

new/usr/src/cmd/hal/utils/fsutils.c

```
*****
5151 Mon May  5 22:34:22 2014
new/usr/src/cmd/hal/utils/fsutils.c
4846 HAL partition names don't match real partition names
*****
1 /*
2 *
3 * fsutils.c : filesystem utilities
4 *
5 * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
6 * Use is subject to license terms.
7 *
8 * Copyright 2014 Andrew Stormont.
9 *
10 #endif /* ! codereview */
11 * Licensed under the Academic Free License version 2.1
12 *
13 */

15 #ifdef HAVE_CONFIG_H
16 #include <config.h>
17 #endif

19 #include <stdio.h>
20 #include <sys/types.h>
21 #include <sys/scsi/impl/uscsi.h>
22 #include <string.h>
23 #include <strings.h>
24 #include <ctype.h>
25 #include <unistd.h>
26 #include <stdlib.h>
27 #include <errno.h>
28 #include <fcntl.h>
29 #include <sys/dkio.h>
30 #include <libintl.h>
31 #include <sys/dktp/fdisk.h>
32 #include <sys/fs/pc_label.h>

34 #include <libhal.h>
35 #include "fsutils.h"

37 /*
38 * Separates dos notation device spec into device and drive number
39 *   pN partition names are rewritten to point to p0
40 *   :N partition names are dropped
41 #endif /* ! codereview */
42 */
43 boolean_t
44 dos_to_dev(char *path, char **devpath, int *num)
45 {
46     int i;
47     char *buf;
48     boolean_t found = B_FALSE;
49     char *p;

50     for (i = strlen(path); i > 0; i--) {
51         if (path[i] == 'p' || path[i] == ':') {
52             found = B_TRUE;
53             break;
54         }
55     }
56     if ((p = strrchr(path, ':')) == NULL) {
57         return (B_FALSE);
58     }

59     if ((*num = atoi(path + i + 1)) == 0 ||
60         (buf = strdup(path)) == NULL) {
```

1

new/usr/src/cmd/hal/utils/fsutils.c

```
13     if ((*num = atoi(p + 1)) == 0) {
59         return (B_FALSE);
60     }

62     (void) strcpy(buf + i, path[i] == 'p' ? "p0" : "");
63     *devpath = buf;
64     return (B_TRUE);
65     p[0] = '\0';
66     *devpath = strdup(path);
67     p[0] = ':';
68     return (*devpath != NULL);
69 }

_____unchanged_portion_omitted_____
```

2