

```

*****
4265 Tue Oct 2 19:27:28 2012
new/usr/src/cmd/cmd-inet/usr.lib/in.ndpd/defs.h
3245 in.ndpd daemon should not be session leader
Reviewed by: Sebastien Roy <sebastien.roy@delphix.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
*****
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 */

26 #ifndef _NDPD_DEFS_H
27 #define _NDPD_DEFS_H

29 #include <stdio.h>
30 #include <stdlib.h>
31 #include <strings.h>
32 #include <errno.h>
33 #include <sys/types.h>
34 #include <fcntl.h>
35 #include <signal.h>
36 #include <poll.h>
37 #include <unistd.h>
38 #include <time.h>
39 #include <stdarg.h>
40 #include <syslog.h>

42 #include <sys/param.h>
43 #include <sys/socket.h>
44 #include <arpa/inet.h>
45 #include <netdb.h>

47 #include <sys/ioctl.h>
48 #include <sys/socketio.h>
49 #include <net/if.h>
50 #include <sys/stropts.h>

52 #include <string.h>
53 #include <ctype.h>

55 #include <netinet/in_sysm.h>
56 #include <netinet/in.h>
57 #include <netinet/ip.h>
58 #include <netinet/ip_icmp.h>
59 #include <netinet/if_ether.h>

```

```

60 #include <netinet/ip6.h>
61 #include <netinet/icmp6.h>
62 #include <net/route.h>
63 #include <libipadm.h>
64 #include <ipadm_ndpd.h>

66 #include "tables.h"

68 #ifdef __cplusplus
69 extern "C" {
70 #endif

72 #define CURHOP_UNSPECIFIED 0
73 #define PATH_NDPD_CONF "/etc/inet/ndpd.conf"
74 #define PATH_PID "/var/run/in.ndpd.pid"

75 extern int debug, no_loopback;

77 extern struct in6_addr all_nodes_mcast;
78 extern struct in6_addr all_routers_mcast;

80 extern int rtsock;
81 extern struct rt_msghdr *rt_msg;
82 extern struct sockaddr_in6 *rta_gateway;
83 extern struct sockaddr_dl *rta_ifp;

85 /* Debug flags */
86 #define D_ALL 0xffff
87 #define D_DEFAULTS 0x0001 /* Default values in config file */
88 #define D_CONFIG 0x0002 /* Config file */
89 #define D_PHYINT 0x0004 /* phyint table */
90 #define D_PREFIX 0x0008 /* prefix table */
91 #define D_ROUTER 0x0010 /* router table */
92 #define D_STATE 0x0020 /* RS/RA state machine */
93 #define D_IFSCAN 0x0040 /* Scan of kernel interfaces */
94 #define D_TIMER 0x0080 /* Timer mechanism */
95 #define D_PARSE 0x0100 /* config file parser */
96 #define D_PKTIN 0x0200 /* Received packet */
97 #define D_PKTBAD 0x0400 /* Malformed packet */
98 #define D_PKTOUT 0x0800 /* Sent packet */
99 #define D_TMP 0x1000 /* RFC3041 mechanism */
100 #define D_DHCP 0x2000 /* RFC3315 DHCPv6 (stateful addr) */

102 #define IF_SEPARATOR ':'
103 #define IPV6_MAX_HOPS 255
104 #define IPV6_MIN_MTU (1024+256)
105 #define IPV6_ABITS 128
106 #define TMP_TOKEN_BITS 64
107 #define TMP_TOKEN_BYTES (TMP_TOKEN_BITS / 8)
108 #define MAX_DAD_FAILURES 5

110 /* Return a random number from a an range inclusive of the endpoints */
111 #define GET_RANDOM(LOW, HIGH) (random() % ((HIGH) - (LOW) + 1) + (LOW))

113 #define TIMER_INFINITY 0xFFFFFFFF /* Never time out */
114 #define PREFIX_INFINITY 0xFFFFFFFF /* A "forever" prefix lifetime */

116 /*
117  * Used by 2 hour rule for stateless addrconf
118  */
119 #define MIN_VALID_LIFETIME (2*60*60) /* In seconds */

121 /*
122  * Control how often pi_ReachableTime gets re-randomized
123  */
124 #define MIN_REACH_RANDOM_INTERVAL (60*1000) /* 1 minute in ms */

```

```
125 #define MAX_REACH_RANDOM_INTERVAL      (60*60*1000)    /* 1 hour in ms */
127 /*
128  * Parsing constants
129  */
130 #define MAXLINELEN      4096
131 #define MAXARGSPERLINE  128

133 void          timer_schedule(uint_t delay);
134 extern void    logmsg(int level, const char *fmt, ...);
135 extern void    logperror(const char *str);
136 extern void    logperror_pi(const struct phyint *pi, const char *str);
137 extern void    logperror_pr(const struct prefix *pr, const char *str);
138 extern int     parse_config(char *config_file, boolean_t file_required);

140 extern int     poll_add(int fd);
141 extern int     poll_remove(int fd);

143 extern char    *fmt_lla(char *llabuf, int bufsize, uchar_t *lla, int llalen);

145 extern int     do_dad(char *ifname, struct sockaddr_in6 *testaddr);

147 #ifdef __cplusplus
148 }
unchanged_portion_omitted
```

```

*****
66314 Tue Oct  2 19:27:28 2012
new/usr/src/cmd/cmd-inet/usr.lib/in.ndpd/main.c
3245 in.ndpd daemon should not be session leader
Reviewed by: Sebastien Roy <sebastien.roy@delphix.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
*****
unchanged_portion_omitted_

908 static void
909 daemonize_ndpd(void)
910 {
911     FILE *pidfp;
912     mode_t pidmode = (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH); /* 0644 */
913     struct itimerval it;
914     boolean_t timerval = _B_TRUE;

915     /*
916      * Need to get current timer settings so they can be restored
917      * after the fork(), as the it_value and it_interval values for
918      * the ITIMER_REAL timer are reset to 0 in the child process.
919      */
920     if (getitimer(ITIMER_REAL, &it) < 0) {
921         if (debug & D_TIMER)
922             logmsg(LOG_DEBUG, "daemonize_ndpd: failed to get itimerval\n");
923         timerval = _B_FALSE;
924     }

925     /* Daemonize. */
926     if (daemon(0, 0) == -1) {
927         switch (fork()) {
928             case 0:
929                 /* Child */
930                 break;
931             case -1:
932                 logperror("fork");
933                 exit(1);
934             default:
935                 /* Parent */
936                 _exit(0);
937         }

938         /* Store our process id, blow away any existing file if it exists. */
939         if ((pidfp = fopen(PATH_PID, "w")) == NULL) {
940             (void) fprintf(stderr, "%s: unable to open " PATH_PID ": %s\n",
941                 argv[0], strerror(errno));
942         } else {
943             (void) fprintf(pidfp, "%ld\n", getpid());
944             (void) fclose(pidfp);
945             (void) chmod(PATH_PID, pidmode);
946         }

947         (void) close(0);
948         (void) close(1);
949         (void) close(2);

950         (void) chdir("/");
951         (void) open("/dev/null", O_RDWR);
952         (void) dup2(0, 1);
953         (void) dup2(0, 2);
954         (void) setsid();

955         already_daemonized = _B_TRUE;

956     }
957 }

```

```

935     * Restore timer values, if we were able to save them; if not,
936     * check and set the right value by calling run_timeouts().
937     */
938     if (timerval) {
939         if (setitimer(ITIMER_REAL, &it, NULL) < 0) {
940             logperror("daemonize_ndpd: setitimer");
941             exit(2);
942         }
943     } else {
944         run_timeouts();
945     }
946 }
unchanged_portion_omitted_

1324 /*
1325  * Pick up a signal "byte" from the pipe and process it.
1326  */
1327 static void
1328 in_signal(int fd)
1329 {
1330     uchar_t buf;
1331     struct phyint *pi;
1332     struct phyint *next_pi;

1333     switch (read(fd, &buf, sizeof (buf))) {
1334     case -1:
1335         logperror("in_signal: read");
1336         exit(1);
1337         /* NOTREACHED */
1338     case 1:
1339         break;
1340     case 0:
1341         logmsg(LOG_ERR, "in_signal: read eof\n");
1342         exit(1);
1343         /* NOTREACHED */
1344     default:
1345         logmsg(LOG_ERR, "in_signal: read > 1\n");
1346         exit(1);
1347     }

1348     if (debug & D_TIMER)
1349         logmsg(LOG_DEBUG, "in_signal() got %d\n", buf);

1350     switch (buf) {
1351     case SIGALRM:
1352         if (debug & D_TIMER) {
1353             uint_t now = getcurrenttime();
1354             logmsg(LOG_DEBUG, "in_signal(SIGALRM) delta %u\n",
1355                 now - timer_next);
1356         }
1357         timer_next = TIMER_INFINITY;
1358         run_timeouts();
1359         break;
1360     case SIGHUP:
1361         /* Re-read config file by exec'ing ourselves */
1362         for (pi = phyints; pi != NULL; pi = next_pi) {
1363             next_pi = pi->pi_next;
1364             if (pi->pi_AdvSendAdvertisements)
1365                 check_to_advertise(pi, START_FINAL_ADV);
1366         }

1367         /*
1368          * Remove all the configured addresses.
1369          * Remove the addrobj names created with ipmgmt.
1370          * Release the dhcpv6 addresses if any.
1371          * Cleanup the phyints.
1372          */
1373     }
1374 }

```

```
1376             */
1377             phyint_delete(pi);
1378         }
1380     /*
1381     * Prevent fd leaks. Everything gets re-opened at start-up
1382     * time. 0, 1, and 2 are closed and re-opened as
1383     * /dev/null, so we'll leave those open.
1384     */
1385     closefrom(3);
1387     logmsg(LOG_ERR, "SIGHUP: restart and reread config file\n");
1388     (void) execv(argv0[0], argv0);
1389     (void) unlink(PATH_PID);
1390     _exit(0177);
1391     /* NOTREACHED */
1392 case SIGUSR1:
1393     logmsg(LOG_DEBUG, "Printing configuration:\n");
1394     phyint_print_all();
1395     break;
1396 case SIGINT:
1397 case SIGTERM:
1398 case SIGQUIT:
1399     for (pi = phyints; pi != NULL; pi = next_pi) {
1400         next_pi = pi->pi_next;
1401         if (pi->pi_AdvSendAdvertisements)
1402             check_to_advertise(pi, START_FINAL_ADV);
1403     }
1404     phyint_delete(pi);
1405     (void) unlink(NDPD_SNMP_SOCKET);
1406     (void) unlink(PATH_PID);
1407     exit(0);
1408     /* NOTREACHED */
1409 case 255:
1410     /*
1411     * Special "signal" from loopback_ra_enqueue.
1412     * Handle any queued loopback router advertisements.
1413     */
1414     loopback_ra_dequeue();
1415     break;
1416 default:
1417     logmsg(LOG_ERR, "in_signal: unknown signal: %d\n", buf);
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 }
```