****
79278 Sun Dec 29 18:32:46 2013
new/usr/src/cmd/mdb/common/mdb/mdb_cmds.c
4436 ::dis -b fails to separate address columns
****
_____unchanged_portion_omitted_

```
1911 static int
1912 cmd_dis(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
1913 {
1914         mdb_tgt_t *tgt = mdb.m_target;
1915         mdb_disasm_t *dis = mdb.m_disasm;

1917         uintptr_t oaddr, naddr;
1918         mdb_tgt_as_t as;
1919         mdb_tgt_status_t st;
1920         char buf[BUFSIZ];
1921         GElf_Sym sym;
1922         int i;

1924         uint_t opt_f = FALSE;           /* File-mode off by default */
1925         uint_t opt_w = FALSE;           /* Window mode off by default */
1926         uint_t opt_a = FALSE;           /* Raw-address mode off by default */
1927         uint_t opt_b = FALSE;           /* Address & symbols off by default */
1928         uintptr_t n = -1UL;             /* Length of window in instructions */
1929         uintptr_t eaddr = 0;            /* Ending address; 0 if limited by n */

1931         i = mdb_getopts(argc, argv,
1932             'f', MDB_OPT_SETBITS, TRUE, &opt_f,
1933             'w', MDB_OPT_SETBITS, TRUE, &opt_w,
1934             'a', MDB_OPT_SETBITS, TRUE, &opt_a,
1935             'b', MDB_OPT_SETBITS, TRUE, &opt_b,
1936             'n', MDB_OPT_UINTPTR, &n, NULL);

1938         /*
1939          * Disgusting argument post-processing ... basically the idea is to get
1940          * the target address into addr, which we do by using the specified
1941          * expression value, looking up a string as a symbol name, or by
1942          * using the address specified as dot.
1943          */
1944         if (i != argc) {
1945                 if (argc != 0 && (argc - i) == 1) {
1946                         if (argv[i].a_type == MDB_TYPE_STRING) {
1947                                 if (argv[i].a_un.a_str[0] == '-')
1948                                         return (DCMD_USAGE);

1950                                 if (dis_str2addr(argv[i].a_un.a_str, &addr))
1951                                         return (DCMD_ERR);
1952                         } else
1953                                 addr = argv[i].a_un.a_val;
1954                 } else
1955                         return (DCMD_USAGE);
1956         }

1958         /*
1959          * If we're not in window mode yet, and some type of arguments were
1960          * specified, see if the address corresponds nicely to a function.
1961          * If not, turn on window mode; otherwise disassemble the function.
1962          */
1963         if (opt_w == FALSE && (argc != i || (flags & DCMD_ADDRSPEC))) {
1964                 if (mdb_tgt_lookup_by_addr(tgt, addr,
1965                     MDB_TGT_SYM_EXACT, buf, sizeof (buf), &sym, NULL) == 0 &&
1966                     GELF_ST_TYPE(sym.st_info) == STT_FUNC) {
1967                         /*
1968                          * If the symbol has a size then set our end address to
1969                          * be the end of the function symbol we just located.
```



```
1970                          */
1971                         if (sym.st_size != 0)
1972                                 eaddr = addr + (uintptr_t)sym.st_size;
1973                 } else
1974                         opt_w = TRUE;
1975         }

1977         /*
1978          * Window-mode doesn't make sense in a loop.
1979          */
1980         if (flags & DCMD_LOOP)
1981                 opt_w = FALSE;

1983         /*
1984          * If -n was explicit, limit output to n instructions;
1985          * otherwise set n to some reasonable default
1986          */
1987         if (n != -1UL)
1988                 eaddr = 0;
1989         else
1990                 n = 10;

1992         /*
1993          * If the state is IDLE (i.e. no address space), turn on -f.
1994          */
1995         if (mdb_tgt_status(tgt, &st) == 0 && st.st_state == MDB_TGT_IDLE)
1996                 opt_f = TRUE;

1998         if (opt_f)
1999                 as = MDB_TGT_AS_FILE;
2000         else
2001                 as = MDB_TGT_AS_VIRT;

2003         if (opt_w == FALSE) {
2004                 n++;
2005                 while ((eaddr == 0 && n-- != 0) || (addr < eaddr)) {
2006                         naddr = mdb_dis_ins2str(dis, tgt, as,
2007                             buf, sizeof (buf), addr);
2008                         if (naddr == addr)
2009                                 return (DCMD_ERR);
2010                         if (opt_a)
2011                                 mdb_printf("%-#32p%8T%s\n", addr, buf);
2012                         else if (opt_b)
2013                                 mdb_printf("%-#?p  %-#32a%8T%s\n",
2013                                 mdb_printf("%-#10p%-#32a%8T%s\n",
2014                                     addr, addr, buf);
2015                         else
2016                                 mdb_printf("%-#32a%8T%s\n", addr, buf);
2017                         addr = naddr;
2018                 }

2020         } else {
2021 #ifdef __sparc
2022                 if (addr & 0x3) {
2023                         mdb_warn("address is not properly aligned\n");
2024                         return (DCMD_ERR);
2025                 }
2026 #endif

2028                 for (oaddr = mdb_dis_previns(dis, tgt, as, addr, n);
2029                     oaddr < addr; oaddr = naddr) {
2030                         naddr = mdb_dis_ins2str(dis, tgt, as,
2031                             buf, sizeof (buf), oaddr);
2032                         if (naddr == oaddr)
2033                                 return (DCMD_ERR);
2034                         if (opt_a)
```

```
2035                                    mdb_printf("%-#32p%8T%s\n", oaddr, buf);
2036                            else if (opt_b)
2037                                    mdb_printf("%-#?p  %-#32a%8T%s\n",
2037                                    mdb_printf("%-#10p%-#32a%8T%s\n",
2038                                        oaddr, oaddr, buf);
2039                            else
2040                                    mdb_printf("%-#32a%8T%s\n", oaddr, buf);
2041                    }

2043                    if ((naddr = mdb_dis_ins2str(dis, tgt, as,
2044                        buf, sizeof (buf), addr)) == addr)
2045                            return (DCMD_ERR);

2047                    mdb_printf("%<b>");
2048                    mdb_flush();
2049                    if (opt_a)
2050                            mdb_printf("%-#32p%8T%s%", addr, buf);
2051                    else if (opt_b)
2052                            mdb_printf("%-#?p  %-#32a%8T%s", addr, addr, buf);
2052                            mdb_printf("%-#10p%-#32a%8T%s", addr, addr, buf);
2053                    else
2054                            mdb_printf("%-#32a%8T%s%", addr, buf);
2055                    mdb_printf("%</b>\n");

2057                    for (addr = naddr; n-- != 0; addr = naddr) {
2058                            naddr = mdb_dis_ins2str(dis, tgt, as,
2059                                buf, sizeof (buf), addr);
2060                            if (naddr == addr)
2061                                    return (DCMD_ERR);
2062                            if (opt_a)
2063                                    mdb_printf("%-#32p%8T%s\n", addr, buf);
2064                            else if (opt_b)
2065                                    mdb_printf("%-#?p  %-#32a%8T%s\n",
2065                                    mdb_printf("%-#10p%-#32a%8T%s\n",
2066                                        addr, addr, buf);
2067                            else
2068                                    mdb_printf("%-#32a%8T%s\n", addr, buf);
2069                    }
2070            }

2072            mdb_set_dot(addr);
2073            return (DCMD_OK);
2074    }
```

_____***unchanged_portion_omitted_***