```
**********************************************************
   75162 Tue Jan 14 16:48:02 2014
new/usr/src/lib/libdtrace/common/dt_consume.c
4159 dtrace ignores all ustack helper frames from dead processes
Reviewed by: Bryan Cantrill <bryan@joyent.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

1318 int
1319 dt_print_ustack(dtrace_hdl_t *dtp, FILE *fp, const char *format,
1320     caddr_t addr, uint64_t arg)
1321 {
1322         /* LINTED - alignment */
1323         uint64_t *pc = (uint64_t *)addr;
1324         uint32_t depth = DTRACE_USTACK_NFRAMES(arg);
1325         uint32_t strsize = DTRACE_USTACK_STRSIZE(arg);
1326         const char *strbase = addr + (depth + 1) * sizeof (uint64_t);
1327         const char *str = strsize ? strbase : NULL;
1328         int err = 0;

1330         char name[PATH_MAX], objname[PATH_MAX], c[PATH_MAX * 2];
1331         struct ps_prochandle *P;
1332         GElf_Sym sym;
1333         int i, indent;
1334         pid_t pid;

1336         if (depth == 0)
1337                 return (0);

1339         pid = (pid_t)*pc++;

1341         if (dt_printf(dtp, fp, "\n") < 0)
1342                 return (-1);

1344         if (format == NULL)
1345                 format = "%s";

1347         if (dtp->dt_options[DTRACEOPT_STACKINDENT] != DTRACEOPT_UNSET)
1348                 indent = (int)dtp->dt_options[DTRACEOPT_STACKINDENT];
1349         else
1350                 indent = _dtrace_stkindent;

1352         /*
1353          * Ultimately, we need to add an entry point in the library vector for
1354          * determining <symbol, offset> from <pid, address>.  For now, if
1355          * this is a vector open, we just print the raw address or string.
1356          */
1357         if (dtp->dt_vector == NULL)
1358                 P = dt_proc_grab(dtp, pid, PGRAB_RDONLY | PGRAB_FORCE, 0);
1359         else
1360                 P = NULL;

1362         if (P != NULL)
1363                 dt_proc_lock(dtp, P); /* lock handle while we perform lookups */

1365         for (i = 0; i < depth && pc[i] != NULL; i++) {
1366                 const prmap_t *map;

1368                 if ((err = dt_printf(dtp, fp, "%*s", indent, "")) < 0)
1369                         break;

1371                 if (P != NULL && Plookup_by_addr(P, pc[i],
1372                     name, sizeof (name), &sym) == 0) {
1373                         (void) Pobjname(P, pc[i], objname, sizeof (objname));
```



```
1375                         if (pc[i] > sym.st_value) {
1376                                 (void) snprintf(c, sizeof (c),
1377                                     "%s`%s+0x%llx", dt_basename(objname), name,
1378                                     (u_longlong_t)(pc[i] - sym.st_value));
1379                         } else {
1380                                 (void) snprintf(c, sizeof (c),
1381                                     "%s`%s", dt_basename(objname), name);
1382                         }
1383                 } else if (str != NULL && str[0] != '\0' && str[0] != '@' &&
1384                     (P == NULL || (map = Paddr_to_map(P, pc[i])) == NULL ||
1385                     map->pr_mflags & MA_WRITE)) {
1384                     (P != NULL && ((map = Paddr_to_map(P, pc[i])) == NULL ||
1385                     (map->pr_mflags & MA_WRITE))) {
1386                         /*
1387                          * If the current string pointer in the string table
1388                          * does not point to an empty string _and_ the program
1389                          * counter falls in a writable region, we'll use the
1390                          * string from the string table instead of the raw
1391                          * address.  This last condition is necessary because
1392                          * some (broken) ustack helpers will return a string
1393                          * even for a program counter that they can't
1394                          * identify.  If we have a string for a program
1395                          * counter that falls in a segment that isn't
1396                          * writable, we assume that we have fallen into this
1397                          * case and we refuse to use the string.  Finally,
1398                          * note that if we could not grab the process (e.g.,
1399                          * because it exited), the information from the helper
1400                          * is better than nothing.
1397                          * case and we refuse to use the string.
1401                          */
1402                         (void) snprintf(c, sizeof (c), "%s", str);
1403                 } else {
1404                         if (P != NULL && Pobjname(P, pc[i], objname,
1405                             sizeof (objname)) != NULL) {
1406                                 (void) snprintf(c, sizeof (c), "%s`0x%llx",
1407                                     dt_basename(objname), (u_longlong_t)pc[i]);
1408                         } else {
1409                                 (void) snprintf(c, sizeof (c), "0x%llx",
1410                                     (u_longlong_t)pc[i]);
1411                         }
1412                 }

1414                 if ((err = dt_printf(dtp, fp, format, c)) < 0)
1415                         break;

1417                 if ((err = dt_printf(dtp, fp, "\n")) < 0)
1418                         break;

1420                 if (str != NULL && str[0] == '@') {
1421                         /*
1422                          * If the first character of the string is an "at" sign,
1423                          * then the string is inferred to be an annotation --
1424                          * and it is printed out beneath the frame and offset
1425                          * with brackets.
1426                          */
1427                         if ((err = dt_printf(dtp, fp, "%*s", indent, "")) < 0)
1428                                 break;

1430                         (void) snprintf(c, sizeof (c), "  [ %s ]", &str[1]);

1432                         if ((err = dt_printf(dtp, fp, format, c)) < 0)
1433                                 break;

1435                         if ((err = dt_printf(dtp, fp, "\n")) < 0)
1436                                 break;
1437                 }
```

```
1439                 if (str != NULL) {
1440                         str += strlen(str) + 1;
1441                         if (str - strbase >= strsize)
1442                                 str = NULL;
1443                 }
1444         }

1446         if (P != NULL) {
1447                 dt_proc_unlock(dtp, P);
1448                 dt_proc_release(dtp, P);
1449         }

1451         return (err);
1452 }
```

unchanged_portion_omitted_