```
**********************************************************
   50647 Fri Jul 22 11:11:26 2016
new/src/java.desktop/unix/native/common/awt/fontpath.c
**********************************************************
```

```
   1 /*
   2  * Copyright (c) 1998, 2016, Oracle and/or its affiliates. All rights reserved.
   2  * Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved.
   3  * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
   4  *
   5  * This code is free software; you can redistribute it and/or modify it
   6  * under the terms of the GNU General Public License version 2 only, as
   7  * published by the Free Software Foundation.  Oracle designates this
   8  * particular file as subject to the "Classpath" exception as provided
   9  * by Oracle in the LICENSE file that accompanied this code.
  10  *
  11  * This code is distributed in the hope that it will be useful, but WITHOUT
  12  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
  13  * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
  14  * version 2 for more details (a copy is included in the LICENSE file that
  15  * accompanied this code).
  16  *
  17  * You should have received a copy of the GNU General Public License version
  18  * 2 along with this work; if not, write to the Free Software Foundation,
  19  * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
  20  *
  21  * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
  22  * or visit www.oracle.com if you need additional information or have any
  23  * questions.
  24  */

  26 #if defined(__linux__)
  27 #include <string.h>
  28 #endif /* __linux__ */
  29 #include <stdio.h>
  30 #include <stdlib.h>
  31 #include <strings.h>
  32 #include <sys/types.h>
  33 #include <sys/stat.h>
  34 #include <sys/mman.h>
  35 #include <fcntl.h>
  36 #include <unistd.h>
  37 #ifdef __solaris__
  38 #include <sys/systeminfo.h>
  39 #endif

  41 #include <jni.h>
  42 #include <jni_util.h>
  43 #include <jvm_md.h>
  44 #include <sizecalc.h>
  45 #ifndef HEADLESS
  46 #include <X11/Xlib.h>
  47 #include <awt.h>
  48 #else
  49 /* locks ought to be included from awt.h */
  50 #define AWT_LOCK()
  51 #define AWT_UNLOCK()
  52 #endif /* !HEADLESS */

  54 #if defined(__linux__) && !defined(MAP_FAILED)
  55 #define MAP_FAILED ((caddr_t)-1)
  56 #endif

  58 #ifndef HEADLESS
  59 extern Display *awt_display;
  60 #endif /* !HEADLESS */
```



```
  62 #define FONTCONFIG_DLL_VERSIONED VERSIONED_JNI_LIB_NAME("fontconfig", "1")
  63 #define FONTCONFIG_DLL JNI_LIB_NAME("fontconfig")

  65 #define MAXFDIRS 512    /* Max number of directories that contain fonts */

  67 #if defined(__solaris__)
  68 /*
  69  * This can be set in the makefile to "/usr/X11" if so desired.
  70  */
  71 #ifndef OPENWINHOMELIB
  72 #define OPENWINHOMELIB "/usr/openwin/lib/"
  73 #endif

  75 /* This is all known Solaris X11 directories on Solaris 8, 9 and 10.
  76  * It is ordered to give precedence to TrueType directories.
  77  * It is needed if fontconfig is not installed or configured properly.
  78  */
  79 static char *fullSolarisFontPath[] = {
  80     OPENWINHOMELIB "X11/fonts/TrueType",
  81     OPENWINHOMELIB "locale/euro_fonts/X11/fonts/TrueType",
  82     OPENWINHOMELIB "locale/iso_8859_2/X11/fonts/TrueType",
  83     OPENWINHOMELIB "locale/iso_8859_5/X11/fonts/TrueType",
  84     OPENWINHOMELIB "locale/iso_8859_7/X11/fonts/TrueType",
  85     OPENWINHOMELIB "locale/iso_8859_8/X11/fonts/TrueType",
  86     OPENWINHOMELIB "locale/iso_8859_9/X11/fonts/TrueType",
  87     OPENWINHOMELIB "locale/iso_8859_13/X11/fonts/TrueType",
  88     OPENWINHOMELIB "locale/iso_8859_15/X11/fonts/TrueType",
  89     OPENWINHOMELIB "locale/ar/X11/fonts/TrueType",
  90     OPENWINHOMELIB "locale/hi_IN.UTF-8/X11/fonts/TrueType",
  91     OPENWINHOMELIB "locale/ja/X11/fonts/TT",
  92     OPENWINHOMELIB "locale/ko/X11/fonts/TrueType",
  93     OPENWINHOMELIB "locale/ko.UTF-8/X11/fonts/TrueType",
  94     OPENWINHOMELIB "locale/KOI8-R/X11/fonts/TrueType",
  95     OPENWINHOMELIB "locale/ru.ansi-1251/X11/fonts/TrueType",
  96     OPENWINHOMELIB "locale/th_TH/X11/fonts/TrueType",
  97     OPENWINHOMELIB "locale/zh_TW/X11/fonts/TrueType",
  98     OPENWINHOMELIB "locale/zh_TW.BIG5/X11/fonts/TT",
  99     OPENWINHOMELIB "locale/zh_HK.BIG5HK/X11/fonts/TT",
 100     OPENWINHOMELIB "locale/zh_CN.GB18030/X11/fonts/TrueType",
 101     OPENWINHOMELIB "locale/zh/X11/fonts/TrueType",
 102     OPENWINHOMELIB "locale/zh.GBK/X11/fonts/TrueType",
 103     OPENWINHOMELIB "X11/fonts/Type1",
 104     OPENWINHOMELIB "X11/fonts/Type1/sun",
 105     OPENWINHOMELIB "X11/fonts/Type1/sun/outline",
 106     OPENWINHOMELIB "locale/iso_8859_2/X11/fonts/Type1",
 107     OPENWINHOMELIB "locale/iso_8859_4/X11/fonts/Type1",
 108     OPENWINHOMELIB "locale/iso_8859_5/X11/fonts/Type1",
 109     OPENWINHOMELIB "locale/iso_8859_7/X11/fonts/Type1",
 110     OPENWINHOMELIB "locale/iso_8859_8/X11/fonts/Type1",
 111     OPENWINHOMELIB "locale/iso_8859_9/X11/fonts/Type1",
 112     OPENWINHOMELIB "locale/iso_8859_13/X11/fonts/Type1",
 113     OPENWINHOMELIB "locale/ar/X11/fonts/Type1",
 114     NULL, /* terminates the list */
 115 };
_____unchanged_portion_omitted_

 683 typedef FcConfig* (*FcInitLoadConfigFuncType)();
 684 typedef FcPattern* (*FcPatternBuildFuncType)(FcPattern *orig, ...);
 685 typedef FcObjectSet* (*FcObjectSetFuncType)(const char *first, ...);
 686 typedef FcFontSet* (*FcFontListFuncType)(FcConfig *config,
 687                                          FcPattern *p,
 688                                          FcObjectSet *os);
 689 typedef FcResult (*FcPatternGetBoolFuncType)(const FcPattern *p,
 690                                              const char *object,
 691                                              int n,
 692                                              FcBool *b);
```

```
693 typedef FcResult (*FcPatternGetIntegerFuncType)(const FcPattern *p,
694                                                 const char *object,
695                                                 int n,
696                                                 int *i);
697 typedef FcResult (*FcPatternGetStringFuncType)(const FcPattern *p,
698                                                const char *object,
699                                                int n,
700                                                FcChar8 ** s);
701 typedef FcChar8* (*FcStrDirnameFuncType)(const FcChar8 *file);
702 typedef void (*FcPatternDestroyFuncType)(FcPattern *p);
703 typedef void (*FcFontSetDestroyFuncType)(FcFontSet *s);
704 typedef FcPattern* (*FcNameParseFuncType)(const FcChar8 *name);
705 typedef FcBool (*FcPatternAddStringFuncType)(FcPattern *p,
706                                              const char *object,
707                                              const FcChar8 *s);
708 typedef void (*FcDefaultSubstituteFuncType)(FcPattern *p);
709 typedef FcBool (*FcConfigSubstituteFuncType)(FcConfig *config,
710                                              FcPattern *p,
711                                              FcMatchKind kind);
712 typedef FcPattern* (*FcFontMatchFuncType)(FcConfig *config,
713                                           FcPattern *p,
714                                           FcResult *result);
715 typedef FcFontSet* (*FcFontSetCreateFuncType)();
716 typedef FcBool (*FcFontSetAddFuncType)(FcFontSet *s, FcPattern *font);

718 typedef FcResult (*FcPatternGetCharSetFuncType)(FcPattern *p,
719                                                 const char *object,
720                                                 int n,
721                                                 FcCharSet **c);
722 typedef FcFontSet* (*FcFontSortFuncType)(FcConfig *config,
723                                          FcPattern *p,
724                                          FcBool trim,
725                                          FcCharSet **csp,
726                                          FcResult *result);
727 typedef FcCharSet* (*FcCharSetUnionFuncType)(const FcCharSet *a,
728                                              const FcCharSet *b);
729 typedef FcChar32 (*FcCharSetSubtractCountFuncType)(const FcCharSet *a,
730                                                    const FcCharSet *b);

732 typedef int (*FcGetVersionFuncType)();

734 typedef FcStrList* (*FcConfigGetCacheDirsFuncType)(FcConfig *config);
735 typedef FcChar8* (*FcStrListNextFuncType)(FcStrList *list);
736 typedef FcChar8* (*FcStrListDoneFuncType)(FcStrList *list);

738 static char **getFontConfigLocations() {

740     char **fontdirs;
741     int numdirs = 0;
742     FcInitLoadConfigFuncType FcInitLoadConfig;
743     FcPatternBuildFuncType FcPatternBuild;
744     FcObjectSetFuncType FcObjectSetBuild;
745     FcFontListFuncType FcFontList;
746     FcPatternGetStringFuncType FcPatternGetString;
747     FcStrDirnameFuncType FcStrDirname;
748     FcPatternDestroyFuncType FcPatternDestroy;
749     FcFontSetDestroyFuncType FcFontSetDestroy;

751     FcConfig *fontconfig;
752     FcPattern *pattern;
753     FcObjectSet *objset;
754     FcFontSet *fontSet;
755     FcStrList *strList;
756     FcChar8 *str;
757     int i, f, found, len=0;
758     char **fontPath;
```



```
760     void* libfontconfig = openFontConfig();

762     if (libfontconfig == NULL) {
763         return NULL;
764     }

766     FcPatternBuild     =
767         (FcPatternBuildFuncType)dlsym(libfontconfig, "FcPatternBuild");
768     FcObjectSetBuild   =
769         (FcObjectSetFuncType)dlsym(libfontconfig, "FcObjectSetBuild");
770     FcFontList         =
771         (FcFontListFuncType)dlsym(libfontconfig, "FcFontList");
772     FcPatternGetString =
773         (FcPatternGetStringFuncType)dlsym(libfontconfig, "FcPatternGetString");
774     FcStrDirname       =
775         (FcStrDirnameFuncType)dlsym(libfontconfig, "FcStrDirname");
776     FcPatternDestroy   =
777         (FcPatternDestroyFuncType)dlsym(libfontconfig, "FcPatternDestroy");
778     FcFontSetDestroy   =
779         (FcFontSetDestroyFuncType)dlsym(libfontconfig, "FcFontSetDestroy");

781     if (FcPatternBuild     == NULL ||
782         FcObjectSetBuild   == NULL ||
783         FcPatternGetString == NULL ||
784         FcFontList         == NULL ||
785         FcStrDirname       == NULL ||
786         FcPatternDestroy   == NULL ||
787         FcFontSetDestroy   == NULL) { /* problem with the library: return. */
788         closeFontConfig(libfontconfig, JNI_FALSE);
789         return NULL;
790     }

792     /* Make calls into the fontconfig library to build a search for
793      * outline fonts, and to get the set of full file paths from the matches.
794      * This set is returned from the call to FcFontList(..)
795      * We allocate an array of char* pointers sufficient to hold all
796      * the matches + 1 extra which ensures there will be a NULL after all
797      * valid entries.
798      * We call FcStrDirname strip the file name from the path, and
799      * check if we have yet seen this directory. If not we add a pointer to
800      * it into our array of char*. Note that FcStrDirname returns newly
801      * allocated storage so we can use this in the return char** value.
802      * Finally we clean up, freeing allocated resources, and return the
803      * array of unique directories.
804      */
805     pattern = (*FcPatternBuild)(NULL, FC_OUTLINE, FcTypeBool, FcTrue, NULL);
806     objset = (*FcObjectSetBuild)(FC_FILE, NULL);
807     fontSet = (*FcFontList)(NULL, pattern, objset);
808     if (fontSet == NULL) {
809         /* FcFontList() may return NULL if fonts are not installed. */
810         fontdirs = NULL;
811     } else {
812         fontdirs = (char**)calloc(fontSet->nfont+1, sizeof(char*));
813         for (f=0; f < fontSet->nfont; f++) {
814             FcChar8 *file;
815             FcChar8 *dir;
816             if ((*FcPatternGetString)(fontSet->fonts[f], FC_FILE, 0, &file) ==
817                                       FcResultMatch) {
818                 dir = (*FcStrDirname)(file);
819                 found = 0;
820                 for (i=0;i<numdirs; i++) {
821                     if (strcmp(fontdirs[i], (char*)dir) == 0) {
822                         found = 1;
823                         break;
824                     }
```

```
825                 }
826                 if (!found) {
827                     fontdirs[numdirs++] = (char*)dir;
828                 } else {
829                     free((char*)dir);
830                 }
831             }
832         }
833         /* Free fontset if one was returned */
834         (*FcFontSetDestroy)(fontSet);
835     }

837     /* Free memory and close the ".so" */
831     (*FcFontSetDestroy)(fontSet);
838     (*FcPatternDestroy)(pattern);
839     closeFontConfig(libfontconfig, JNI_TRUE);
840     return fontdirs;
841 }
```

*unchanged_portion_omitted_*