User Interface - Feature #1898

write a replacement for the CHARVA libTerminal.so as an NCURSES-based Linux shared library

11/01/2012 05:21 PM - Greg Shah

Status:	Closed	**Start date:**	05/21/2013
Priority:	Normal	**Due date:**	06/04/2013
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	80.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Blocks User Interface - Feature #2131: implement the Windows equivalent to th...	**Closed**	**06/05/2013**	**07/03/2013**

History

#1 - 11/01/2012 05:24 PM - Greg Shah

- Estimated time changed from 40.00 to 60.00

This implementation must be written from scratch without copying any code from libTerminal.so (really: Toolkit.c). However, it is permissible to read Toolkit.c to understand what has to be done. The currently used code for CHARVA is in p2j/lib/charva_cut_down_20101216.zip.

Make any and all changes to the interface that are needed to make it cleaner and simpler. Plan the code such that it can be compiled for multiple platforms. This means that the JNI interface and as much code as possible should be generic. But of course, the platform-specific backing functions would be in a platform-specific file.

Integrate the build into P2J (this should just be additional JNI methods in libp2j.so/p2j.dll) and make the changes on the Java side to remap the current CHARVA client/dependencies to this new library. Delete the remaining CHARVA code from P2J.

#2 - 04/25/2013 09:54 AM - Greg Shah

- Estimated time changed from 60.00 to 80.00

- Assignee set to Eugenie Lyzenko

- Target version set to Milestone 7

- Start date set to 05/21/2013

- Due date set to 06/04/2013

#3 - 07/15/2013 05:09 PM - Greg Shah

- File Toolkit.c_20101216 added

- File Toolkit.c_20041108 added

- File toolkit_diffs.txt added

Attached are the versions of the Toolkit.c which are important to review (and the diffs between them).

Some important notes:

1. If the code only exists in the 2010 version, then Golden Code owns it and it can be copied directly.
2. Any code that is very heavily modified in 2010 version, such that it is not even close to the 2004 version, can be copied.
3. Any code that is in 2010 version, which is just simply a direct call to an NCURSES API can be copied. This is true even if it is not owned by

Golden Code, because one cannot copyright calls to APIs or other such "purely functional" code that is required to interoperate with a system or interface.
4. Any code in the 2010 version that is never called by P2J can be dropped.

After reviewing the code with the above notes in mind, please list any problem areas which you see, so that I can review them too.

#4 - 07/16/2013 06:07 PM - Eugenie Lyzenko

The function list used in native NCURSES based code from charva.Toolkit.java(Java end, the JNI names to be implemented):

CharvaDriver.java and CharvaPrimitives.java
addArrayNative() - GCD property
addStringNative() - 90% of code has been rewritten by GCD.
beep() - direct call to an NCURSES API
clear() - direct call to an NCURSES API
Static getACSchar() - to rewrite, simple case clause returning NCURSES constants
Static getAttribute() - to rewrite, simple case clause returning NCURSES constants
getScreenColumnsNative() - GCD property
getScreenRowsNative() - GCD property
Static getTermType() - GCD property
getx() - almost direct call to an NCURSES API
gety() - almost direct call to an NCURSES API
hasColors() - direct call to an NCURSES API
Static init() - 50% of code has been rewritten by GCD.
Static isColorEnabled(not JNI) - This flag is true if the system property "charva.color" has been set, no direct call to native lib
readKey() - potentially rewrite, the CHARVA hack has been implemented here
resetMode() - GCD property
resume() - GCD property
setCursorNative() - 50% of code has been rewritten by GCD (almost all are the direct calls to NCURSES API).
setCursorStatus() - GCD property
Static setTermType() - GCD property
suspend() - GCD property
syncNative() - direct call to an NCURSES API

JNI Call that never used by current P2J via charva.Toolkit.java:
getColor(),
addCharNative(),
addHorizontalLineNative(),
addVerticalLineNative(),
blankBoxNative(),
drawBoxNative(),
getMaxColorPairs(),
initColorPair(),
startColors(),
close()

The questions:
1. The calls detected as never used can be just ignored, we do not need to implement them in charva replacement, correct?
2. The main questions are related to mixed functions, where there is original code and GCD improvements:
addStringNative(), init(), setCursorNative(). We need to rewrite "old" code because the risk of property problems?
3. What to do with getACSchar() and getAttribute()? There is really no code logic inside the functions. We can use different approach for example instead of:

```
...
switch (offset)
case 0:
   return result0;
case 1
   return result1;
...
```

we can use just predefined array:

```
...
if (offset within data range)
   return result[offset];
...
```

4. Will our implementation still need the NCURSES to be patched? Or we should fix this bug in our code if this is possible?

#5 - 07/17/2013 07:29 AM - Greg Shah

1. The calls detected as never used can be just ignored, we do not need to implement them in charva replacement, correct?

Correct.

2. The main questions are related to mixed functions, where there is original code and GCD improvements:

addStringNative(), init(), setCursorNative(). We need to rewrite "old" code because the risk of property problems?

Please post these methods here in Redmine. Annotate the code to show which lines are untouched. I will review them.

3. What to do with getACSchar() and getAttribute()? There is really no code logic inside the functions.

Agreed. These are not protectable by copyright because there is nothing expressive here. The code is entirely functional.

With that said, I have no problem with implementing a simpler array-based version. Please go ahead with that idea.

4. Will our implementation still need the NCURSES to be patched? Or we should fix this bug in our code if this is possible?

Yes, we will still need the patch to NCURSES. If there was a better way, I would have already implemented it. The problem is inside NCURSES itself and the patch is required. The readKey() implementation MUST stay that way.

#6 - 07/17/2013 12:16 PM - Eugenie Lyzenko

The code inside EVL<< .. EVL>> "braces" duplicates the original CHARVA source. The other code outside "braces" - GCD property.

```
...
JNIEXPORT void JNICALL Java_charva_awt_Toolkit_addStringNative
  (JNIEnv *env, jobject jo, jbyteArray jstr, jint len, jint attrib, jint colorpair)
{
//EVL<<
    int i;
    int attr = attrib;

    if (colors_started)
       attr |= COLOR_PAIR(colorpair);
//EVL>>

   if (len <= 0)
      return;

   jbyte* chrs = (jbyte*) malloc(len * sizeof(jbyte));

   if (chrs == NULL)
   {
      throwException(env, "java/lang/OutOfMemoryError", "Cannot copy data.");
   }

   (*env)->GetByteArrayRegion(env, jstr, 0, len, chrs);

   for (i = 0; i < len; i++)
   {
      // remove the automatic sign extension (we want to treat this as
      // an unsigned 8-bit number)
      int extended = 0x000000FF & (int) chrs[i];
      my_addch(extended | attr);
   }

   (*env)->DeleteLocalRef(env, jstr);
   free(chrs);
}
...
JNIEXPORT jboolean JNICALL Java_charva_awt_Toolkit_setCursorNative
  (JNIEnv *env, jobject jo, jint x_, jint y_)
{
//EVL<<
    int x, y;
    my_move(y_, x_);
//EVL>>

    // Set current cursor position
    getyx(stdscr, y, x);

    return y == y_ && x == x_;
}
...
JNIEXPORT void JNICALL Java_charva_awt_Toolkit_init(JNIEnv *env, jobject jo)
{
// EVL<<
   char *strcap;
// EVL>>
   char *termname;
// EVL<<
   int i;
// EVL>>

   termname = getenv("TERM");

   if (termname == NULL || *termname == '\0')
   {
      termname = "vt100";
   }

   // use this instead of initscr because there is no way to query the
   // current terminal and when you use multiple terms, newterm overwrites
   // the current terminal... so you can never get it if you don't use
   // newterm directly
   terminal = newterm(termname, stdout, stdin);
```

```
   if (terminal == NULL)
   {
      char* spec   = "Interactive terminal of type %s cannot be initialized.";
      char* errmsg = malloc(strlen(spec) + strlen(termname) + 1);
      sprintf(errmsg, spec, termname);
      throwException(env, "java/lang/IllegalStateException", errmsg);
      return;
   }

// EVL<<
   keypad(stdscr, TRUE);      // enable keyboard mapping
   timeout(500);              // wait up to 500msec for input
   noecho();                  // don't echo input
   raw();
   nonl();
// EVL>>
   auto_getch_refresh(FALSE); // disable sync during getch()
   def_prog_mode();           // save our state (after our mode changes)

// EVL<<
   hascolors = has_colors();

#ifdef NCURSES_MOUSE_VERSION
   /* For some reason, if the mouse_interval is nonzero, the button-presses
    * are reported very erratically. So we disable click resolution by
    * setting mouse_interval to 0.
    */
   mousemask(BUTTON1_RELEASED | BUTTON1_PRESSED |
         BUTTON2_RELEASED | BUTTON2_PRESSED |
         BUTTON3_RELEASED | BUTTON3_PRESSED,
         NULL);
   mouseinterval(0);
#endif
// EVL>>

   getmaxyx(stdscr, base_rows, base_cols);

// EVL<<
   atexit((void (*)(void))endwin);
// EVL>>

}
...
```

> Yes, we will still need the patch to NCURSES. If there was a better way, I would have already implemented it. The problem is inside NCURSES itself and the patch is required. The readKey() implementation MUST stay that way.

OK. Understood.

#7 - 07/17/2013 12:22 PM - Eugenie Lyzenko

Another word we can use readKey() untouched? May be removing try_again: label which is never used now.

#8 - 07/17/2013 04:15 PM - Greg Shah

Some thoughts:

1. I don't see anything in addStringNative() that looks protectible. The entire function can be put into our code. If colors_started is never true, then that portion can be deleted.

2. I don't see anything in setCursorNative() that looks protectible. The entire function can be put into our code. The my_move() is just a remapping of the NCURSES move() so it is not protected.

3. In init() you can use everything except the mouse-related code in the #ifdef. Just delete that portion of the code.

4. readKey() can simply be:

```
{
   return (jint) getch();
}
```

We don't need the label, the local var or the comment.

#9 - 07/22/2013 12:13 PM - Eugenie Lyzenko

The implementation plan:
1. Create two new classes in com/goldencode/p2j/ui/client/chui/driver:
- ncurses/NcursesDriver.java
- ncurses/NcursesPrimitives.java
replacing CharvaDriver and CharvaPrimitives.
2. Move KeyProcessor.java to new com/goldencode/p2j/ui/client/chui/driver/ncurses or com/goldencode/p2j/ui/client/chui.
3. Create in com/goldencode/p2j/ui/client/chui/driver/ncurses NcursesHelper.java class to define JNI instead of charva Toolkit.java declaring only calls we really need.
4. Create native C file terminal.c to implement JNI functions from NcursesHelper.java.
5. Create terminal_win.c and terminal_linux.c to implement OS dependent native C code.
6. Compile and remove all old charva related references

Is this plan and naming conventions OK?

#10 - 07/22/2013 06:18 PM - Eugenie Lyzenko

Another point to discuss is related to the terminal calls in Windows C code. I can see two options here:
1. Use Console Functions API and implement all required functions in P2J.DLL.
2. Use third party libraries for example PDCURSES for Windows(which in turn really use Console Functions API as it can be seen from pdcurses.dll) and call the functions like we do it in Linux.

I'm inclining to choose number 1 above because we can take almost full control on what is happening(until Win32 calls), we can implement only what we need and we can not patch pdcurses.dll so easy as we do with NCURSES sources. And using any additional software can bring potential bugs into our product or become accidentally copyrighted.

What do you think about this point?

#11 - 07/22/2013 07:19 PM - Eugenie Lyzenko

After rethink I would offer to rename:
- ncurses/NcursesDriver.java
to
- console/ConsoleDriver.java
and
- ncurses/NcursesPrimitives.java
to
- console/ConsolePrimitives.java
and
NcursesHelper.java
to
ConsoleHelper.java

This could be more neutral names with reference to the text-based console backend.

#12 - 07/23/2013 07:18 AM - Eugenie Lyzenko

- File evl_upd20130723a.zip added

The drop includes java part of suggested changes for review. The native C code structure will depends on which way we choose. Developing all terminal functions inside P2J.DLL will separate C code in two different parts, one for Linux/Solaris(based on NCURSES) and other - for Windows(based on Console Functions API).

#13 - 07/23/2013 09:30 AM - Eugenie Lyzenko

Another offering is to move the helper class interfacing to JNI - ConsoleHelper to better location for example com/goldencode/p2j/ui/chui or even to com/goldencode/util. In addition this way we can decrease the current long line for JNI prefix:
Java_com_goldencode_p2j_ui_client_chui_driver_console_ConsoleHelper_.

#14 - 07/24/2013 06:29 PM - Eugenie Lyzenko

- File evl_upd20130724a.zip added

This is the first proposal for native implementation for Linux only. The Windows native part is still missing. But it is working without CHARVA and libTerminal.so.

#15 - 07/25/2013 08:48 AM - Greg Shah

I like the overall approach. The changes to make it console/Console* instead of ncurses/Ncurses* is very good. It is more platform independent and that is perfect.

In regard to note 10:

1. Use Console Functions API and implement all required functions in P2J.DLL.

This is the approach. Do NOT use the PDCURSES approach.

> I'm inclining to choose number 1 above because we can take almost full control on what is happening(until Win32 calls), we can implement only what we need and we can not patch pdcurses.dll so easy as we do with NCURSES sources. And using any additional software can bring potential bugs into our product or become accidentally copyrighted.

I completely agree. You are right.

> Another offering is to move the helper class interfacing to JNI - ConsoleHelper to better location for example com/goldencode/p2j/ui/chui or even to com/goldencode/util.

I prefer to leave the helper class in the console directory. It is OK to have the long JNI names.

I'll review the code next.

#16 - 07/25/2013 09:05 AM - Greg Shah

Code Review 0724a

Overall, this is very good. Some minor things:

1. The copyright notice for ConsoleHelper is wrong (it is improperly copied from the Toolkit.java):

```
** Modifications Copyright (c) 2005-2010, Golden Code Development Corporation.
** The original LGPL license terms apply.
```

Please put our normal copyright notice there. Nothing in this file should be owned by anyone other than Golden Code.

2. Please rewrite any javadoc and/or regular comments in ConsoleHelper and terminal_linux.c so that we are not copying any comment text from Toolkit.java or Toolkit.c.

3. Please do remove the // EVL markers and the other commented Charva code that is no longer needed (including in build.xml).

When do you think you will be ready to test? It looks pretty close.

#17 - 07/25/2013 09:06 AM - Greg Shah

To be clear, I want you to get this tested, working and checked in before you do any work on the WIN32 version of the helpers.

#18 - 07/25/2013 11:06 AM - Eugenie Lyzenko

> When do you think you will be ready to test? It looks pretty close.

OK. I'll clean up the code soon and perform some testing. I think during today-tomorrow it will be ready to test on devsrv01.

And another remaining question - what to do with class KeyProcessor.java currently located at com/goldencode/p2j/ui/client/chui/driver/charva? We are going to completely remove this directory, right? Or not? If we remove directory - I can offer to place KeyProcessor.java under com/goldencode/p2j/ui/client/chui/driver/console. Is it OK? Or may be com/goldencode/p2j/ui/client/chui/driver will be better?

#19 - 07/25/2013 01:31 PM - Greg Shah

> I can offer to place KeyProcessor.java under com/goldencode/p2j/ui/client/chui/driver/console. Is it OK?

Yes, it belongs in console/ because it is specific to the NCURSES implementation. When you do the Windows implementation, it may need some modifications.

#20 - 07/25/2013 06:45 PM - Eugenie Lyzenko

- File evl_upd20130725a.zip added

Additional files with references to CHARVA included in this drop to modify:
com/goldencode/p2j/ui/client/driver/ScreenDriver.java
com/goldencode/p2j/ui/Keyboard/java

The new drop uploaded for your review. Hope all javadocs are fixed, some functions simplifications for terminal_linux.c and fixing one bug with getting line drawing constants. I guess it is ready for testing. I've made simple tests with old testcases, working with the Ubuntu terminal with NCURSES patch. Going to test a bit more locally.

#21 - 07/26/2013 07:59 AM - Greg Shah

Code Review 0725a

1. You can remove the javadoc text from ConsoleDriver:

```
 * <p>
 * This replaces CHARVA driver.
```

We don't need to leave any references to CHARVA in the code or docs.

2. As a rule, I would like you to try hard to avoid the need for JNI code (e.g. usage of JNIEnv) in the terminal_linux|win.c files. The idea is that terminal.c should be where all the JNI processing is centralized. The terminal_linux|win.c files would only have minimal helper functions that provide the native calls and whatever limited processing which cannot be centralized. The idea of centralizing this code is to maximize common code between Linux and Windows AND to limit the use of the JNI helpers since that processing can be tricky.

It is OK if you want to do this centralization as part of the work for the #2131. If so then you can go into testing now.

#22 - 07/26/2013 08:16 AM - Eugenie Lyzenko

> 1. You can remove the javadoc text from ConsoleDriver:
>
> - <p>
> - This replaces CHARVA driver.
>
> We don't need to leave any references to CHARVA in the code or docs.

OK.

> 2. As a rule, I would like you to try hard to avoid the need for JNI code (e.g. usage of JNIEnv) in the terminal_linux|win.c files. The idea is that

> terminal.c should be where all the JNI processing is centralized. The terminal_linux|win.c files would only have minimal helper functions that provide the native calls and whatever limited processing which cannot be centralized. The idea of centralizing this code is to maximize common code between Linux and Windows AND to limit the use of the JNI helpers since that processing can be tricky.
>
> It is OK if you want to do this centralization as part of the work for the #2131. If so then you can go into testing now.

Yes, agree. I thought about such separation and this is certainly better than one we currently have. But before I do not have the Windows equivalent I can not say for sure what we can leave in terminal.c as common code and what platform specific code we can put into _linux(win).c versions.

So for now I'm going to test the current version(with modified ConsoleDriver.java).

#23 - 07/26/2013 08:17 AM - Greg Shah

OK, good.

#24 - 07/26/2013 08:30 AM - Eugenie Lyzenko

- File evl_upd20130726a.zip added

The drop with ConsoleDriver.java javadoc change. Going to test in devsrv01.

The question: do I need to manually remove charva package in testing directory before start runtime testing? Or this will happen automatically?

#25 - 07/26/2013 08:38 AM - Greg Shah

> The question: do I need to manually remove charva package in testing directory before start runtime testing? Or this will happen automatically?

The scripts will have no idea about your changes, so they cannot know to remove charva. You will have to remove it yourself (charva.jar and the libTerminal.so).

You also need to make a change to the jar manifests (see p2j/manifest/*.mf for charva.jar references). That must be part of your update.

#26 - 07/26/2013 08:59 AM - Eugenie Lyzenko

- File deleted (evl_upd20130726a.zip)

#27 - 07/26/2013 09:00 AM - Eugenie Lyzenko

- File evl_upd20130726a.zip added

The last drop was replaced with one including manifest changes. The runtime testing has been started.

#28 - 07/31/2013 12:35 PM - Eugenie Lyzenko

Finally the regression testing has been passed with bzr code base 30368. So I guess it is ready to check in and distribute.

#29 - 07/31/2013 12:36 PM - Greg Shah

Yes, go ahead and check-in/distribute.

#30 - 07/31/2013 02:15 PM - Greg Shah

Don't forget to remove the following files:

lib/charva_cut_down_20100816.zip
lib/charva_cut_down_20101216.zip
lib/charva.jar
lib/libTerminal.so.32bit
lib/libTerminal.so.64bit

You will have to do a 2nd check in to bzr since these were left behind in 10369.

#31 - 07/31/2013 02:38 PM - Eugenie Lyzenko

> Don't forget to remove the following files:
>
> lib/charva_cut_down_20100816.zip
> lib/charva_cut_down_20101216.zip
> lib/charva.jar
> lib/libTerminal.so.32bit
> lib/libTerminal.so.64bit
>
> You will have to do a 2nd check in to bzr since these were left behind in 10369.

I need to:
bzr remove lib/files*
and
bzr commit
Correct? This is what I did there, bu the changes was not committed.

#32 - 07/31/2013 02:44 PM - Greg Shah

I just did it here. It worked fine. The revision number is 10370. I just used bzr remove on the specific files and then did a bzr commit.

#33 - 07/31/2013 02:45 PM - Greg Shah

Make sure to list the files that were removed in your distribution email.

#34 - 07/31/2013 02:48 PM - Greg Shah

I forgot: the src/com/goldencode/p2j/ui/client/chui/driver/charva/* has to be removed too. I did that now in bzr revision 10371.

#35 - 09/06/2013 11:29 AM - Greg Shah

- Status changed from New to Closed

- % Done changed from 0 to 100

#36 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

Toolkit.c_20041108	19.8 KB	07/15/2013	Greg Shah
Toolkit.c_20101216	28.5 KB	07/15/2013	Greg Shah
toolkit_diffs.txt	23 KB	07/15/2013	Greg Shah
evl_upd20130723a.zip	17.7 KB	07/23/2013	Eugenie Lyzenko
evl_upd20130724a.zip	32.7 KB	07/24/2013	Eugenie Lyzenko
evl_upd20130725a.zip	52.5 KB	07/25/2013	Eugenie Lyzenko
evl_upd20130726a.zip	53.4 KB	07/26/2013	Eugenie Lyzenko