User Interface - Support #2660

evaluate if the NCURSES 5.7 "threading improvements" can be made to work for P2J such that auto_getch_refresh() is no longer needed

08/25/2015 08:50 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Runtime Infrastructure - Support #4549: reduce/eliminate installat...	**New**
Related to Build and Source Control - Support #5167: using static linking to ...	**Closed**

History

#1 - 08/25/2015 08:53 AM - Greg Shah

http://invisible-island.net/ncurses/ncurses.faq.html#multithread

```
Why does (fill in the blank) happen when I use two threads?

If you have a program which uses curses in more than one thread, you will almost certainly see odd behavior. T
hat is because curses relies upon static variables for both input and output. Using one thread for input and o
ther(s) for output cannot solve the problem, nor can extra screen updates help. This FAQ is not a tutorial on
threaded programming.

Starting with ncurses 5.7, this implementation of curses provides the ability to configure and compile the lib
rary to help solve the problem by:

    reducing the use of static variables (see curs_sp_funcs(3x)),

    adding mutexes around the low-level terminal I/O,

    changing global variables such as LINES to "getter" functions (see curs_opaque(3x), and

    adding functions which an application can use to avoid those which rely upon global (or static) variables
(see curs_threads(3x).

Almost all programs can be recompiled to use the "ncursest" or "ncursestw" libraries. Just recompiling is not
enough to make a program thread-safe. As usual, some (re)design effort is probably needed.

The test-programs provided with ncurses (ncurses-examples) include a few which demonstrate this alternate conf
iguration of the library: ditto, rain, worm.
```

#2 - 08/25/2015 08:53 AM - Greg Shah

-------- Forwarded Message --------
Subject: proposed extension API addition for ncurses 5.5
Date: Tue, 29 Aug 2006 12:30:37 -0400
From: Greg Shah <ges@goldencode.com>
To: dickey@invisible-island.net

Mr. Dickey,

I have an application which requires that a user can generate an asynchronous notification (via CTRL-C) at any time, no matter what the state of the input or output processing of the application. This notification must be honored immediately by the application. Due to this design requirement, we have implemented our application in 2 threads.

I fully understand that NCURSES is not thread safe, but I have found a way to make it safe-enough in this particular case.

I use 1 thread for reading the keyboard via getch() and another thread for all output processing. Any time the keyboard reading thread (which is simply in an infinite loop of calling getch() with timeout set to 500 ms) happens to call into getch() at the same moment that the output thread is processing a screen update there will be a problem. In particular, getch() forces a refresh() before blocking on reading the FIFO or terminal. Since access to the internal data structures of ncurses is not protected in any way, calling a refresh() at (essentially) random times will sometimes force the output buffer to the terminal at a moment when it is in an inconsistent state (because it is in the middle of being updated by another thread). This causes unfinished escape sequences etc... to be pushed out, leaving the terminal looking corrupted.

My solution is to add a simple extension to NCURSES 5.5:

void auto_getch_refresh(bool)

This simply sets a static flag inside lib_getch.c. This flag is checked inside the wgetch_should_refresh macro to allow the disabling of refresh() during getch(). This getch_refresh flag defaults to TRUE which is completely compatible with the current NCURSES implementation. So in all current usage, there is no difference in behavior. In the case where one wishes to disable refresh during getch(), this is now possible. In particular, we found that our 2 threaded case where 1 thread is dedicated to getch() is now safe.

Some thoughts:

1. This is very low risk.
2. In the case where this new function is never called, this doesn't change the behavior in regards to standards or interface compliance.
3. The performance cost is very low (on an Intel CPU, only 3 additional instructions per getch()).
4. The memory footprint of the data is only an extra 4 bytes per process.

I am hoping this will be acceptable for inclusion in the next release of NCURSES.

Thank you for your time. Please let me know if you have any questions or comments.

Greg Shah

#3 - 08/25/2015 08:54 AM - Greg Shah

-------- Forwarded Message --------
Subject: Re: proposed extension API addition for ncurses 5.5
Date: Tue, 29 Aug 2006 20:10:06 -0400 (EDT)
From: Thomas Dickey <dickey@his.com>
To: Greg Shah <ges@goldencode.com>
CC: dickey@invisible-island.net

On Tue, 29 Aug 2006, Greg Shah wrote:

> My solution is to add a simple extension to NCURSES 5.5:

I'd suppose it would be more straightforward to add mutexes and measure
the performance than to add a special case. What you're describing is
a non-threadsafe version of one of the likely places for adding a mutex.

> void auto_getch_refresh(bool)
>
> This simply sets a static flag inside lib_getch.c. This flag is checked
> inside the wgetch_should_refresh macro to allow the disabling of refresh()
> during getch(). This getch_refresh flag defaults to TRUE which is completely
> compatible with the current NCURSES implementation. So in all current usage,
> there is no difference in behavior. In the case where one wishes to disable
> refresh during getch(), this is now possible. In particular, we found that
> our 2 threaded case where 1 thread is dedicated to getch() is now safe.

But if two threads were to call wgetch(), that wouldn't work.

--
Thomas E. Dickey
http://invisible-island.net
ftp://invisible-island.net

#4 - 08/25/2015 08:55 AM - Greg Shah

-------- Forwarded Message --------
Subject: Re: proposed extension API addition for ncurses 5.5
Date: Wed, 30 Aug 2006 07:22:19 -0400
From: Greg Shah <ges@goldencode.com>
To: Thomas Dickey <dickey@his.com>
CC: dickey@invisible-island.net

Yes, this is all true. And having multiple threads in output operations
is similarly problematic. But a real thread-safe implementation is
quite a large undertaking. Every shared resource and data structure
would have to be protected. To do it right, a great deal of analysis is
needed as to the granularity of the mutexes, possible deadlock/race
conditions... and access to most data structures would best be hidden
behind new internal interfaces (getters/setters or the like) to allow
the mutex usage to be consistent without hard coding the mutex access
everywhere. For example, even reading a flag from the WINDOW structure
would require protection.

I don't have the time (does anyone?) to do a full thread-safe
implementation. So my approach is a compromise to be safe-enough while
minimizing the effort and risk.

Thanks,
Greg

#5 - 08/25/2015 08:59 AM - Greg Shah

I never heard back from that reply. BUT then they added some dual threading support in 5.7. NCURSES 5.7 came out on November 2, 2008, 2 years after my email conversation with Thomas Dickey. It is possible that the new approach handles our dual threading needs as well as our approach. The mutexes are certainly slower than my approach, but if the difference can't be seen by the user AND it is just as safe/reliable, then we should move to the new approach and eliminate the need for patching NCURSES.

This task is intended to test out the viability of this approach and implement the change if it is OK.

#6 - 08/25/2015 09:01 AM - Greg Shah

- Target version set to Deployment and Management Improvements

#7 - 08/25/2015 09:08 AM - Greg Shah

NCURSES 5.9 release notes includes:

> This extends support for threaded applications by providing a new API which eliminates the need for a global screen-pointer.

#8 - 11/16/2016 01:08 PM - Greg Shah

- Target version deleted (Deployment and Management Improvements)

#9 - 02/19/2020 02:12 PM - Greg Shah

- Related to Support #4549: reduce/eliminate installation dependencies added

#10 - 02/26/2021 08:05 AM - Greg Shah

- Related to Support #5167: using static linking to eliminate the need to patch the system-wide ncurses added