CoCoA-5 - Bug #672

Emacs UI: strange string literal causes crash

09 Mar 2015 21:35 - John Abbott

Status:	Closed	**Start date:**	09 Mar 2015
Priority:	Low	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	EmacsUI	**Estimated time:**	4.80 hours
Target version:	CoCoA-5.3.0	**Spent time:**	4.65 hours

Description

I created a file (CRASH.cocoa5) containing the following:

```
"^\";
```

Note that the string is just 1 character long: with ASCII code 28, but Emacs prints it as ^\.

Sending this line to CoCoA-5 (via C-c C-RET) for evaluation causes a crash (after a few seconds).

Reading the file directly into CoCoA-5 seems to work OK (well, it does not crash).

Related issues:

Related to CoCoA-5 - Bug #1448: EmacsUI: sending line with "7" calls SourceRe...	**Closed**	**17 Apr 2020**

History

#1 - 10 Mar 2015 00:45 - John Abbott

Other chars which give trouble include those with ASCII code 3, ...
[to be completed]

#2 - 10 Mar 2015 11:38 - John Abbott

The EmacsUI really seems to do strange things when the input contains unprintable chars. I am guessing that it is the comint function which actually sends the input that is causing the trouble, yet the documentation for comint-send-input says nothing about "mangling".

As a workaround the user can use source and SourceRegion (*e.g.* via C-c C-f and C-c C-r).

#3 - 10 Mar 2015 11:40 - John Abbott

- Description updated

#4 - 09 Nov 2017 14:43 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have confirmed that a crash occurs even if you type as input to an interactive CoCoA-5 session the string "\^" where the single character in the string has ASCII code 28.

NOTE the same string literal does strange things when sent to /bin/bash running inside a comint buffer. I have not yet found any useful hints on the internet.

NOTE2 Whatever Emacs does to the string, it should not cause the CoCoA-5 interpreter to crash; so there is a CoCoA-5 bug somewhere.

#5 - 09 Nov 2017 16:14 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 20

I have no idea what Emacs does with such an input string. Searching on internet yielded nothing useful.

I have recompiled CoCoAInterpreter without READLINE, and added (temporary) debugging code. Whatever Emacs does, it causes getline to crash; so it is not really a CoCoA-5 problem... but still annoying :-(

Do we just have to close this as an unresolvable issue?

#6 - 25 Mar 2019 13:52 - John Abbott

This is still a bug (in emacs 26.1).

A possible solution could be to print "weird characters" using backslash+octal; then we would also have to change the code which reads string literals so that it can read a printed string correctly. An advantage of handling "weird characters" ourselves is that the appearance of the output should be independent of the user interface (*e.g.* emacs, gui, shell, other?)

Or do we need two ways to print strings? One more-or-less as it is currently, and one which prints a string so it can be read in correctly?

#7 - 02 Mar 2020 20:46 - John Abbott

The bug/crash still happens.

Note that by default the CoCoA-5 interpreter is started in Emacs with the option **--no-readline**; so it cannot be readline's fault.

We could make C-c RET scan the line for "funny chars"; if there are any, then the line is sent via SourceRegion.
Note that long lines are already automatically sent via SourceRegion.

#8 - 03 Mar 2020 13:56 - John Abbott

- Target version changed from CoCoA-5.?.? to CoCoA-5.3.0

- % Done changed from 20 to 70

As far as I can tell the problem lies inside the std c++ function getline and the way it interacts with cin (which is **not** a binary stream). There seems to be no portable way to "reopen" cin as a binary stream.

Note that the problem "largely goes away" if we use readline because readline itself "eats" the weird characters.

Source and SourceRegion seem to work OK; I suppose the ifstream associated to a FileLineProvider is automatically in binary mode?

I suggest closing (rejecting?) this issue. Perhaps the best approach would be as in comment 7: scan the input line, and if it contains "unprintable" chars then read the line using SourceRegion.

#9 - 03 Mar 2020 14:10 - John Abbott

The relevant part of emacs code is probably the function **cocoa5-send-line** around line 1400 in file cocoa5.el.

Looking into altering it...

#10 - 03 Mar 2020 14:48 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 70 to 90

I have revised **cocoa5-send-line**, and it now seems to work more or less as hoped.

The only remaining problem is reading a string literal containing a NUL character, but that bug lies elsewhere...
BUG seems to be in line 449 of Lexer.C. How to fix??? Changed **if (!ch)** into **if (cp != CharPointer::Null)**. Seems to work!

#11 - 03 Mar 2020 20:56 - John Abbott

The current impl thinks that a TAB character is a "funny" character; so sending a line containing a TAB will automatically use SourceRegion. Should TABs be allowed as "normal" characters?

Perhaps it is not so important; and KISS suggests that I should not make TAB an exception...

#12 - 04 Mar 2020 15:31 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 4.80 h

I have now modified the regexp so that TAB does not match (and this is considered a "normal" rather than a "funny" character). This is useful when a line contains a TAB to achieve the correct amount of indentation.

I believe the original problem is due a shell sitting in between Emacs and the running CoCoAInterpreter. The shell sees CTRL-\ which it interprets as a request to send a signal (SIGQUIT) to the underlying process (namely CoCoAInterpreter).

By sending lines which contain "funny" characters via SourceRegion the shell is bypassed, and no strange interpretations are made (except perhaps translation of newlines(?)).

There was also a bug in the lexer which interpreted a NUL inside a string literal as an end-of-input. I have corrected that.

#13 - 17 Apr 2020 16:07 - John Abbott

- Related to Bug #1448: EmacsUI: sending line with "7" calls SourceRegion (!?!?!) added