CoCoA-5 - Bug #713

External libs: interrupting not easy

18 May 2015 15:35 - John Abbott

Status:	Closed	**Start date:**	18 May 2015
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	External Libs	**Estimated time:**	2.66 hours
Target version:	CoCoA-5.4.0	**Spent time:**	2.65 hours

Description

CoCoAInterpreter intercepts SIGTERM signals; this means that a long computation happening inside an external lib cannot be interrupted with a normal ctrl-C (since the signal is caught by CoCoAInterpreter, but is not considered until CoCoAInterpreter regains control).

Is this a bug?

Related issues:

Related to CoCoALib - Feature #714: Interrupt mechanism	**Closed**	**19 May 2015**
Related to CoCoA-5 - Bug #345: Interpreter interrupt delayed confusingly	**Closed**	**24 Apr 2013**

History

#1 - 18 May 2015 15:42 - John Abbott

Remember that a (portable) signal handler cannot do much.

Christof was puzzled when running some Normaliz tests through CoCoA-5: he wanted to stop the (long) computation, but found that ctrl-C was just ignored.

One possibility is to restore the default signal handler when calling functions from an external library; this means that the external library decides how to handle the signal.

Upon returning from the external library the CoCoA-5 signal handler should be reinstated.

How does a call to an external lib know that it is inside CoCoA interpreter?

#2 - 18 May 2015 15:44 - John Abbott

Perhaps the best solution would be to leave the code as it is, and to write better documentation. After all a call to GMP is a call to an external lib; and some calls can be quite long.

#3 - 18 May 2015 16:25 - John Abbott

Another possible (simple) approach is to offer a command line flag which stops CoCoAInterpreter from intercepting signals; this means that a ctrl-C (equiv. SIGINT) would immediately kill the process, losing all data computed so far.

Actually, it could make sense not to intercept signals if the interpreter is in "batch mode" (*i.e.* taking input from a file rather than a terminal).

#4 - 18 May 2015 16:31 - John Abbott

Christof has tried some experiments with GAP, and he noticed that a single ctrl-C appeared to do nothing, but 2 ctrl-C within a second caused the program to abort (with a message explaining that it had received 2 ctrl-C within a second).

We must look up to see what a (portable) signal handler can do.

I recall also that a long G-basis computation cannot be interrupted with ctrl-C (equiv. SIGINT); instead you have to use a stronger signal such as SIGKILL.

#5 - 18 May 2015 21:01 - John Abbott

I have found this web page which gives some advice:
http://stackoverflow.com/questions/103280/portable-way-to-catch-signals-and-report-problem-to-the-user

Here is another web site:
http://www.cplusplus.com/forum/unices/16430/

This second site suggests that sigaction is better than signal for specifying the handler; if so, we will have to make some minore changes to src/CoCoA-5/Main.C.

#6 - 18 May 2015 21:09 - John Abbott

I am considering the following behaviour:

- each time ctrl-C is received a counter is increased
- if the CoCoA interpreter handles the interrupt, it resets the counter to zero
- if the counter is increased to 2, it triggers printing a message saying that one more ctrl-C will kill the process
- if the counter is increased to 3 or greater than the process terminates (perhaps by calling _Exit()?)

This is really rather complicated for a signal handler, so may not be portable (but it probably is).

Do you think this is a reasonable solution?

I could try mimicking GAP but taking note of when the last ctrl-C was received, and allowing increasing beyond 2 only if the last ctrl-C was not long ago (*e.g.* 1-2secs)

#7 - 18 May 2015 21:35 - John Abbott

GNU has some useful looking documentation:
http://www.gnu.org/software/libc/manual/html_node/Signal-Sets.html#Signal-Sets
http://www.gnu.org/software/libc/manual/html_node/Sigaction-Function-Example.html

I noticed some example using write instead of cerr << ...; supposedly it is safer, but it looks to be a real hassle :-(

#8 - 21 May 2015 15:23 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 30

It is probably best to KISS: *i.e.* just have a straightforward signal handler which simply sets the flag. If this proves to be inadequate, we can "improve"

it later on.

I am not even sure what the advantages of GAP's sophisticated approach are.

#9 - 28 Jan 2020 10:54 - John Abbott

- Status changed from In Progress to Feedback

- Assignee set to John Abbott

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

- % Done changed from 30 to 90

- Estimated time set to 2.44 h

Since this issue was created we have also added SignalWatcher to CoCoALib; so this issue is just as relevant to CoCoALib as it is to CoCoA-5.

The current impl is fairly KISS. Right now the user can choose between sending a SIGINT (usu. ctrl-C on Linux boxes) and then possibly waiting for an uninterruptible step of the computation to finish, or sending a SIGQUIT (usu. ctrl-\ on Linux boxes) which normally terminates the process abruptly.

In theory, a function from an external library may want to monitor for interrupts, and potentially exit prematurely. But how would it exit prematurely in a way that CoCoALib can handle/comprehend what is going on? This is not clear to me.

I suggest accepting the current arrangement, and if later a real situation arises where it is unacceptable, then we can use that experience to help guide us to a better solution.

#10 - 14 Feb 2020 08:59 - John Abbott

- Target version changed from CoCoA-5.3.0 to CoCoA-5.4.0

#11 - 24 Sep 2021 21:58 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time changed from 2.44 h to 2.66 h

CoCoALib and CoCoA-5 do often monitor for ctrl-C. Inside CoCoA-5 the situation seems to be reasonably good.

The discussion about external libraries did not reach a conclusion: it seems to be a tricky issue.

Closing after more than 12 months in feedback. Current code is acceptable; maybe return to the matter at some later point (if problems are encountered).