

CoCoA-5 - Bug #345

Interpreter interrupt delayed confusingly

24 Apr 2013 16:18 - John Abbott

Status:	Closed	Start date:	24 Apr 2013
Priority:	Normal	Due date:	
Assignee:	John Abbott	% Done:	100%
Category:	Parser/Interpreter	Estimated time:	2.01 hours
Target version:	CoCoA-5.1.3/4 Jan 2016	Spent time:	1.70 hour
Description The following behaved unexpectedly in CoCoA-5 via emacs: <pre>factorial(10000); // any computation with a long output // press C-c C-c while the result is being printed // the output is *not* interrupted 1+1; // the interrupt is recognised *after* sending this command!</pre> Modify interpreter to "flush" any cached interrupts immediately before starting execution of a new command.			
Related issues:			
Related to CoCoA-5 - Bug #713: External libs: interrupting not easy		Closed	18 May 2015
Related to CoCoALib - Feature #714: Interrupt mechanism		Closed	19 May 2015

History

#1 - 24 Apr 2013 16:24 - John Abbott

The interrupt can be delayed longer than I thought!

```
factorial(10000);
// C-c C-c during pinting
factorial(10); // works just fine
1+1; // the interrupt surfaces upon execution of this command!
```

Very strange.

I'll think about what I believe should be the correct behaviour; then I'll have to modify the code...

#2 - 28 Apr 2013 11:14 - John Abbott

- Status changed from New to In Progress
- % Done changed from 0 to 10

Update (2013-04-29) I now think that Giovanni's idea of throwing an exception probably handles correctly all the various cases mentioned here.

Here are some circumstances to consider:

1. normal interactive input
2. interactive input sent via cut-and-paste
3. input from **source** or **SourceRegion**
4. input being read directly from a file

I suspect that cases (1) and (2) cannot be easily distinguished by the interpreter's source code.

What would it mean if CoCoAInterpreter received a C-c C-c interrupt (a SIGTERM signal) when reading directly from a file?

Would it be acceptable for the code to silently ignore an interrupt on rare occasions?

#3 - 29 Apr 2013 09:16 - John Abbott

Some notes about the source code.

An interpreter object contains a (volatile) data member called controlC. This is set obviously in a signal handler. The value is checked by two functions both called checkForInterrupts. I'm not yet sure why there are two of these; perhaps interrupts are allowed during parsing (might be appropriate if an input expression is huge).

#4 - 02 Apr 2014 17:33 - Anna Maria Bigatti

- Target version set to CoCoA-5.1.0 Easter14

#5 - 09 Apr 2014 17:13 - John Abbott

- Target version changed from CoCoA-5.1.0 Easter14 to CoCoA-5.?.?

#6 - 26 Jun 2015 08:23 - Anna Maria Bigatti

Is this a possible solution?

The interpreter checks //before starting a computation// for (un-caught) interruption and gives a suitable error message ("found pending interrupt signal"?).

#7 - 26 Jun 2015 11:01 - John Abbott

I have found this line at Interpreter.C:2740

```
this->controlC = false;
```

It appears to be executed only for the C5 IDE; maybe it should always be executed?

#8 - 26 Jun 2015 11:31 - Anna Maria Bigatti

John Abbott wrote:

I have found this line at Interpreter.C:2740
[...]

It appears to be executed only for the C5 IDE; maybe it should always be executed?

Sounds promising!

#9 - 26 Jun 2015 17:55 - Anna Maria Bigatti

- *Status changed from In Progress to Closed*
- *Assignee set to John Abbott*
- *% Done changed from 10 to 100*
- *Estimated time set to 2.01 h*

moving the line seems to have solved the problem of "pending interrupt".

#10 - 04 Dec 2017 11:45 - Redmine Admin

- *Target version changed from CoCoA-5.?.? to CoCoA-5.1.3/4 Jan 2016*