

## CoCoA-5 - Feature #1231

### system command

26 Oct 2018 16:04 - John Abbott

<b>Status:</b>	Closed	<b>Start date:</b>	26 Oct 2018
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	John Abbott	<b>% Done:</b>	100%
<b>Category:</b>	CoCoA-4 function to be added	<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>	CoCoA-5.3.0	<b>Spent time:</b>	5.65 hours
<b>Description</b> CoCoA-4 had a <b>system</b> command which would pass its string argument to a shell for execution.  There is no such command in CoCoA-5 since it is obviously unsafe <i>e.g.</i> the string could be "cd; rm -rf *"  Martin says they need the system command for various interactions between ApCoCoA and other programs.  A possible approach might be to have a system command which is activated only if CoCoA is started with a suitable argument; so you have to start CoCoA in a "special" way to be able to use the system command.  Is this a reasonable compromise? Are there any other ideas?			
<b>Related issues:</b>			
Related to CoCoA-5 - Support #1311: THINGS TO DO IN GENOVA September 2019		<b>Closed</b>	<b>11 Sep 2019</b>
Related to CoCoA-5 - Support #1222: Release CoCoA-5.3.0		<b>Closed</b>	<b>28 Aug 2018</b>
Related to CoCoA-5 - Bug #1502: SystemCommand in Microsoft version		<b>In Progress</b>	<b>08 Oct 2020</b>
Related to CoCoA-5 - Bug #1524: wrong SystemCommand exit value		<b>Closed</b>	<b>28 Oct 2020</b>

### History

#### #1 - 26 Oct 2018 18:41 - Anna Maria Bigatti

- % Done changed from 0 to 10

I believe that Macaulay and Singular have such a command, and they use it for their interfaces.

So I do agree that we should add it ourselves, and with an explicit flag.

Maybe we could also forbid the substring **rm** in system, and make a **system\_rm** function, so it is easy to spot call to **rm**?

#### #2 - 09 Nov 2018 14:58 - John Abbott

- Status changed from New to In Progress

I have made a first attempt at implementing a "system call" function, but I am not really comprehending the structure of the interpreter :-(

I want there to be a "global" flag which is set at initial start up (and should never be changed thereafter), and then the interpreter should immediately give an error if someone attempts to execute a system command when this is not allowed by the "global" flag.

So far I have been copying blindly code used for the describe command, but that seems to have been a bad idea. Maybe return is better?

#### #3 - 16 Nov 2018 16:18 - John Abbott

I am now thinking that making **system** a new interpreter command was not a clever idea. It can simply be a built-in function; this should be much easier to implement.

Note that there is no corresponding function to add to CoCoALib since the C++ standard function **system** is always available.

#### #4 - 07 Jan 2019 17:56 - John Abbott

- Status changed from *In Progress* to *Resolved*
- Assignee set to *John Abbott*
- % Done changed from 10 to 80

This seems to be (almost) done, but I cannot yet access CVS...

#### #5 - 18 Feb 2019 09:50 - John Abbott

I repeat here something I wrote on "Telegram" (because redmine was down).

The name **system** causes a clash with an example from the CoCoAManual (where a top-level variable called `system` is used). Indeed, it seems natural for people to want to use the name `system`.

What should the name of the function for making "system calls" be?

Other possible choices are **SystemCall** or **SysCall**, **SystemCommand** or **SystemCmd** or **SysCmd**, perhaps also **ShellCommand** or **CLICCommand** (but maybe this is Unix/Linux-specific?).

Note that Wikipedia makes a clear distinction between "system call" and "system command". In our case, only "system command" is correct.

My current preference is with **SystemCmd** (or perhaps **SystemCommand**); for some reason **SysCmd** seems too short/cryptic.

#### #6 - 18 Feb 2019 10:37 - Anna Maria Bigatti

Note that Wikipedia makes a clear distinction between "system call" and "system command". In our case, only "system command" is correct.

My current preference is with **SystemCmd** (or perhaps **SystemCommand**); for some reason **SysCmd** seems too short/cryptic.

I'll check the meaning, but I still think that **SystemCall** would be more self-explanatory.

Otherwise **SystemCommand**, with no abbreviations. Or, better, **SystemCommand\_Beware\_AtYourRisk** ;-)

#### #7 - 18 Feb 2019 11:11 - John Abbott

OK, the longer name **SystemCommand** is clearer, and anyway the function should only very rarely be called, so succinctness is unimportant (and it is nice that the name is long enough to be easily visible).

#### #8 - 20 Sep 2019 16:10 - John Abbott

There is currently no manual entry for **SystemCommand**.

**#9 - 20 Sep 2019 16:10 - John Abbott**

- Related to Support #1311: *THINGS TO DO IN GENOVA* September 2019 added

**#10 - 20 Sep 2019 16:10 - John Abbott**

- Related to Support #1222: Release CoCoA-5.3.0 added

**#11 - 25 Sep 2019 11:54 - John Abbott**

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

There is a manual page now.

**#12 - 25 Sep 2019 11:58 - John Abbott**

Currently, any output from the system command is sent to stdout (so might be lost if a GUI is being used).

Perhaps it would be better to have the result being a record of the form **record[ExitCode:=0, output:="abc"]**

Comments, ideas, suggestions?

KISS?

**#13 - 27 Sep 2019 17:25 - Anna Maria Bigatti**

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

Tested on MacOSX.

Works.

**#14 - 08 Oct 2020 12:29 - John Abbott**

- Related to Bug #1502: SystemCommand in Microsoft version added

**#15 - 30 Oct 2021 19:50 - John Abbott**

- Related to Bug #1524: wrong SystemCommand exit value added