CoCoALib - Design #1699

Shell scripts: Shebang line, etc

11 Oct 2022 21:42 - John Abbott

Status:	In Progress	**Start date:**	11 Oct 2022
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Portability	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	1.00 hour

Description

Ulrich recently provided a nice shell script for checking whether Qt5 is available (see #1393).

The script use a "shebang" line different from what our other shell scripts use.
Should we change Ulrich's script? Should we change all the other scripts?

Discuss!

Related issues:

Related to CoCoA-5 - Support #1393: GUI with Qt5	**Feedback**	**13 Jan 2020**

History

#1 - 11 Oct 2022 21:43 - John Abbott

- Related to Support #1393: GUI with Qt5 added

#2 - 11 Oct 2022 22:00 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have spent some time looking at comments on StackOverflow (or similar sites).

- **(A)** Advantages of /**usr**/**bin**/**env bash** it is portable provided env is indeed in /usr/bin/env (which is not guaranteed, but appears to be almost always the case)
- **(B)** Advantages of /**bin**/**bash** it is fairly portable (except BSD family), and cannot call some "unexpected executable" which happens to be in the user's PATH -- indeed it is independent of the PATH shell variable.

For information: /usr/bin/env looks for the first executable with the given name in the user's path, and uses that.
This should work on any non-ancient linux/unix-like platform. However, if the user has a peculiar PATH then bash may not be found (and the script will exit with an error). A miscreant could also create another executable called bash, and arrange the PATH variable so that this executable is found first -- this means that the script would trigger execution of the "impostor" bash... with uncertain consequences.

Specifying explicitly /**bin**/**bash** is safer, in that it cannot call an "impostor". BUT it will trigger an error if /bin/bash does not exist; apparently this is the situation on some BSD platforms.

What to do? Currently, I am inclined to favour safety over portability.

A secondary note, but with similar underlying cause.
Ulrich perfectly reasonably called cat and rm in the script. I have changed these to /bin/rm and /bin/cat to avoid the possibilities of "impostors". Obviously if /bin/rm or /bin/cat do not exist then an error will be triggered. Is there any modern linux/unix-like platform where such an error would be forthcoming? If so, how do we work around it?

#3 - 12 Oct 2022 19:07 - Ulrich von der Ohe

I agree the CoCoA scripts should consistently use only one of the shebang options. I think either option has a potential disadvantage against the other and by mixing them, one inherits the union of the disadvantages.

(As an advantage of the env option, one might add that it allows the user to use a different version of the shell easily, e.g. should the one in /bin be nonexistent, buggy or otherwise unsuitable. But that may not be a major point here.)

John Abbott wrote:

> Ulrich perfectly reasonably called cat and rm in the script. I have changed these to
> /bin/rm and /bin/cat to avoid the possibilities of "impostors". Obviously if /bin/rm
> or /bin/cat do not exist then an error will be triggered. Is there any modern linux/unix-like
> platform where such an error would be forthcoming? If so, how do we work around it?

I'm not sure I understand the argument for providing full paths to cat and rm in detail. Wouldn't the same argument apply to make and qmake (which are called in the same script)? And also to several other commands, e.g., g++ (see CXX=g++ in the configure script)?

Is there some standard (POSIX ?) you would generally assume a user's system to conform to? In POSIX it seems cat, rm, and make may be expected to behave reasonably, but /bin is not among the directories guaranteed to exist.

#4 - 14 Oct 2022 20:12 - Ulrich von der Ohe

On a more practical note, perhaps the following shebang line provides the behavior that you want (without warranty or encouragement!):

#!/usr/bin/env -S PATH=/what/you:/want /bin/bash

It is however less portable than either of the above options (in particular, env -S doesn't work e.g. with OpenBSD env, but BSDs don't seem to be in your focus anyway).

Also it assumes the script to provide a set of paths that works across all targeted systems.

On the other hand, where it works it allows execution of programs only from a set of prescribed paths while still allowing the system to choose the path from that set for each program.

#5 - 16 Feb 2024 10:01 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880