brainvisa-development - Task #14527

Ensure compatibility with systems that use Python 3 by default

14/03/2016 02:54 PM - Leprince, Yann

Status:	Closed	**Start date:**	14/03/2016
Priority:	Normal	**Due date:**	
Assignee:	Riviere, Denis	**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:	brainvisa-4.6	**Spent time:**	0:00 hour

Description

On some systems (for example Arch Linux since 2010), /usr/bin/python points to a Python 3 installation. Therefore, unqualified calls to python fail when the script uses Python 2 syntax. As Python 3 is becoming mainstream and Python 2 slowly fades out, such systems will become increasingly common. We therefore need to ensure compatibility with such systems.

For now, the only way of running BrainVISA on systems where Python 3 is the default, is by using a workaround linking plain python to /usr/bin/python2:

```
mkdir /tmp/python2-path
ln -s /usr/bin/python2 /tmp/python2-path/python
export PATH=/tmp/python2-path:"$PATH"
```

A proper fix involves replacing all calls to Python with python2:

- The first line (shebang) of Python scripts should be

  ```
  #! /usr/bin/env python2
  ```

- Calls to Python from CMake scripts should use the ${PYTHON_EXECUTABLE} variable;
- Calls to Python in programs need to be replaced, these calls will probably be harder to find.
- We can also expect conflicts with utilities that need to interpret Python code (e.g. Sphinx).

I have pushed a first commit implementing these changes for the brainvisa-cmake project (r72084, to trunk only). With it I can successfully run bv_maker and the CMake configuration step on Arch Linux. **Denis or Yann, can you please test if these changes break anything?** In particular, I am afraid that the python2 executable might not exist on the oldest architectures that are used on your build bots.

If I get your green light, then I will continue to push the necessary changes to trunk, referencing this issue in the commits.

Associated revisions

Revision 72114 - 14/03/2016 05:57 PM - Riviere, Denis

include python2 / python3 command symlink in python packaging
#14527

Revision 72114 - 14/03/2016 05:57 PM - Riviere, Denis

include python2 / python3 command symlink in python packaging
#14527

Revision 72118 - 14/03/2016 07:00 PM - Riviere, Denis

oops, forgot to report a fix for mac case
#14527

Revision 72118 - 14/03/2016 07:00 PM - Riviere, Denis

oops, forgot to report a fix for mac case
#14527

Revision 72120 - 15/03/2016 10:33 AM - Leprince, Yann

Use python2 in the shebang of Python scripts (Task #14527)

The following command has been used to perform the conversion:

sed -i -e '1s/^\(#!.*python\)\($\|[^2].*$\)/\12\2/' script.py [...]

Revision 72120 - 15/03/2016 10:33 AM - Leprince, Yann

Use python2 in the shebang of Python scripts (Task #14527)

The following command has been used to perform the conversion:

sed -i -e '1s/^\(#!.*python\)\($\|[^2].*$\)/\12\2/' script.py [...]

Revision 72121 - 15/03/2016 10:48 AM - Leprince, Yann

Change brainvisa launcher to use python2 if available

Task #14527

Revision 72168 - 16/03/2016 02:04 PM - Leprince, Yann

Replace calls to python with calls to python2 in program code

Task: #14527

Revision 72168 - 16/03/2016 02:04 PM - Leprince, Yann

Replace calls to python with calls to python2 in program code

Task: #14527

Revision 72201 - 18/03/2016 10:56 AM - Leprince, Yann

Replace plain calls to python with python2

Task: #14527

Revision 72202 - 18/03/2016 11:05 AM - Leprince, Yann

Fix mistake in commit r72201

Task #14527

Revision 72203 - 18/03/2016 11:19 AM - Leprince, Yann

Call python2 from axon's ExecutionContext.pythonSystem method

Task #14527

Revision 72204 - 18/03/2016 11:23 AM - Leprince, Yann

Replace remaining calls to python by python2

Task #14527

Revision 72211 - 18/03/2016 01:10 PM - Leprince, Yann

Replace remaining calls to python by python2

Task #14527

History

#1 - 14/03/2016 03:20 PM - Riviere, Denis

Python2 exists as a symlink on most of our linux distributions but I don't know for other ones.
BUT:

- on michael / gabriel (CentOS 5), python2 points to the system python, which is a 2.4, not the one we are using
- on Mac the link does not exist
  We can easily fix those 2 by manually creating the symlink on both machines
- in binary packages we ship "python" and "pyton2.7" as the former is a link to the latter, but not "python2", so we need to change the packaging of python
- on Windows I can't tell right now (can't connect to the Windows VM now - I don't know why) but as symlinks don't exist there I doubt both python and python2 exe are here.
- aren't we going against a kind of standard if we do that, or is it a standard practice ?

#2 - 14/03/2016 05:14 PM - Leprince, Yann

It is indeed recommended practice. From https://www.python.org/dev/peps/pep-0394/:

> In order to tolerate differences across platforms, all new code that needs to invoke the Python interpreter should not specify python , but rather should specify either python2 or python3 [...]. This distinction should be made in shebangs, when invoking from a shell script, when invoking via the system() call, or when invoking in any other context.

The python2 link is actually created when building Python from source (since 2.7.3), so I think that it makes sense to include it in the packaging.

As for Windows, I do not know how the Python scripts are called right now, but a launcher is available that actually interprets the #! /usr/bin/env python2 line and automatically launches the required version (https://docs.python.org/3/using/windows.html#shebang-lines).

#3 - 14/03/2016 05:45 PM - Riviere, Denis

OK let's go for it.
Symlinks have been added on build systems (Linux/Mac).
I'm currently modifying python packaging.
We have also to check what's going on on Windows (I'm not sure how this python launcher is installed/used, if it is: we are using a bash shell most of the time, so it probably works as on unix).

#4 - 18/04/2016 04:00 PM - Leprince, Yann

- File python2refs.patch added

- Status changed from New to In progress

- Assignee set to Cointepas, Yann

Okay, now I think that all the straightforward calls of python have been replaced by python2.

All that remain are calls from BrainVISA / Axon. These are trickier:

- we cannot simply use sys.executable as the Python version needed by the child program does not necessarily match the one used by the GUI;
- remote Python versions (e.g. called through soma-workflow) may not correspond to local versions;
- what version should context.pythonSystem call?

Maybe the safest approach is to have BrainVISA and soma-workflow always run their children processes in a safe environment where python points to python2? This would allow legacy python2 programs to continue working, and any program that wants python3 must mention the version anyway.

For reference, I attach a patch that *shows the places in the code* that still refer to unqualified python. Please do not apply this patch verbatim: this is an old version that changes all the calls to use pythonSystem and sys.executable, which as an afterthought I think is a *bad idea*.

I prefer not to change anything myself in this area, so I will call out for the BrainVISA experts: **Yann, Denis, what do you think?**

#5 - 18/04/2016 05:56 PM - Riviere, Denis

Well, the context.pythonSystem() call runs python as a child process, on the same machine as the caller. Remote execution, at least currently, runs a brainvisa on a remote node, which may spawn subprocesses for system commands (python or other). So in this context I think it's safe to use the same python as the parent brainvisa.
In the code at the moment "python" is used unqualified to let the PATH find it, but (at least most of the time) it will be the same as sys.executable, so maybe it's OK to use sys.executable.

For remote python access, the question would rather be in soma-workflow jobs building from brainvisa processes. Here we may wonder which python should be used (in workflow.py, ProcessToWorkflow.__init__()). Here again, for now we are relying on the PATH, and running the unqualified "python" command. Maybe we could use the basename of sys.executable (because we are running brainvisa, so we are supposed to use the same python version)?

I agree that commands using python3 (or any other python which is not the one used by brainvisa) must use it explicitely. Maybe we could provide a kwarg in pythonSystem() to specify a different python command, but this is not the hot question now.

It should be noted that several old systems do not provide a "python2" command or symlink, we faced a number of such situations, but making a symlink somewhere in the PATH is an easy solution.

#6 - 19/04/2016 11:23 AM - Leprince, Yann

- Assignee changed from Cointepas, Yann to Riviere, Denis

Riviere, Denis a écrit :

> In the code at the moment "python" is used unqualified to let the PATH find it, but (at least most of the time) it will be the same as sys.executable, so maybe it's OK to use sys.executable.

I do not think that we should use sys.executable, because this would basically prevent people from writing Python3 scripts.

The problem with the current interface of context.pythonSystem is that it is one wrapper for two different languages (Python2 and Python3).

I think that the best approach would be to rely on the shebang (initial #!/usr/bin/env python line) for running the correct version. Under Linux/Mac this is straightforward, all that should be done is to make the program executable and run it as /path/to/program.py instead of python /path/to/program.py. Under Windows an official Python launcher exists that reads the shebang line and launches the right version of Python (py.exe / pyw.exe, see https://www.python.org/dev/peps/pep-0397/). We can then use py.exe /path/to/program.py, or maybe with the help of file type associations we do not even need to call py.exe explicitly (I do not know much about Windows).

> For remote python access, the question would rather be in soma-workflow jobs building from brainvisa processes. Here we may wonder which python should be used (in workflow.py, ProcessToWorkflow.__init__()). Here again, for now we are relying on the PATH, and running the unqualified "python" command. Maybe we could use the basename of sys.executable (because we are running brainvisa, so we are supposed to use the same python version)?
>
> It should be noted that several old systems do not provide a "python2" command or symlink, we faced a number of such situations, but making a symlink somewhere in the PATH is an easy solution.

Again, I don't think that we should make the assumption that the script is using the same version as brainvisa/soma-workflow. Disentangling the two would allow the writers of Python child processes to use the version that they want.

As for systems that do not have a python2 executable, we should probably provide a sane PATH that sticks to the recommendations of https://www.python.org/dev/peps/pep-0394/#recommendation, independent of sys.executable:

- python2 -> preferred python2 version
- python3 -> preferred python3 version
- python -> python2

#7 - 19/04/2016 12:06 PM - Riviere, Denis

> I do not think that we should use sys.executable, because this would basically prevent people from writing Python3 scripts.

That's half true, yes. In my last message I was suggesting that scripts using python3 should not use pythonsystem(), or use it with an addditional parameter to specify it (python_command="python3" for instance). But you're right in the way that in this situation the correct python version must be known from "outside" of the script, and it's a "bidouille" anyway. The existance of this pythonSystem() method is, in itself, a poor attempt at fixing something that does not work well by itself...
On Linux we would totally skip it.

> The problem with the current interface of context.pythonSystem is that it is one wrapper for two different languages (Python2 and Python3).
>
> I think that the best approach would be to rely on the shebang (initial #!/usr/bin/env python line) for running the correct version. Under Linux/Mac this is straightforward, all that should be done is to make the program executable and run it as /path/to/program.py instead of python /path/to/program.py. Under Windows an official Python launcher exists that reads the shebang line and launches the right version of Python (py.exe / pyw.exe, see https://www.python.org/dev/peps/pep-0397/). We can then use py.exe /path/to/program.py, or maybe with the help of file type associations we do not even need to call py.exe explicitly (I do not know much about Windows).

Perhape yes, but it would require to install and package correctly this py.exe in binary distributions. It could be done.

Another thing: on the latest Mac systems (10.11), a "funny" "security" feature, called "system integrity protection" prevents shells to run python scripts via the shebang if the python script is not approved by Apple (or something like that, not sure I have really understood) (see http://brainvisa.info/forum/viewtopic.php?f=2&t=1761#p6563). Of course we are not approved by Apple, and BrainVisa is not distributed via the apple app store, so there is no chance we will be one day, and custom processes will not, either. Running python the way we do it in BrainVisa avoids to run a shell, and avoids this problem of system blocking python scripts execution. So it's still useful for this, also.

> > For remote python access,
> > ...

> Again, I don't think that we should make the assumption that the script is using the same version as brainvisa/soma-workflow. Disentangling the two would allow the writers of Python child processes to use the version that they want.

Here it's not really an issue I think, because workflows are generated by brainvisa, and they are always running brainvisa (it's brainvisa processes which are in workflow nodes), so they do need the same python version anyway.

> As for systems that do not have a python2 executable, we should probably provide a sane PATH that sticks to the recommendations of https://www.python.org/dev/peps/pep-0394/#recommendation, independent of sys.executable:
>
> - python2 -> preferred python2 version
> - python3 -> preferred python3 version

- python -> python2

In binary distributions we ship a python2 symlink along with the python command. The problem is only for people who develop by taking the sources of brainvisa, on an old system (but it's a common situation on clusters, for instance). And again it's a very small problem that is easily overcome.

#8 - 26/09/2017 02:47 PM - Riviere, Denis

- Target version set to brainvisa-4.6

#9 - 26/09/2017 02:53 PM - Riviere, Denis

- Status changed from In progress to Closed

Files

python2refs.patch	7.82 KB	18/04/2016	Leprince, Yann