Run a Linux application on WSL from a Windows python script

Asked 1 month ago Modified 1 month ago Viewed 139 times

1

In my case the application is OpenFoam, but this happens also with others Linux programs.

I have a py script (run_test.py) wrote with Windows VS Code:

```
import subprocess

command = "wsl python3 /mnt/wslg/distro/home/j/test.py"
subprocess.Popen(command, shell=True)
```

that run another script py (test.py) wrote in Linux VS Code in my WSL:

```
import subprocess

command = "cd $FOAM_RUN/airfopt && foamRun"
process=subprocess.Popen(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
output, error = process.communicate()

if output:
    print("Output:", output.decode())
if error:
    print("Error:", error.decode())
```

When i start test.py through Linux VS Code everything works fine, but when i run "run_test.py" i get the error `/bin/sh: 1: cd: can't cd to /airfopt`

Instead `$FOAM_RUN/airfopt` if i write the full path `/mnt/wslg/distro/root/OpenFOAM/root-11/run/airfopt` the new error is `/bin/sh: 1: foamRun: not found` .

I think because run_test.py run test.py under Windows enviroment and not Linux one. I want run test.py from run_test.py but under Linux env.

The same error running an test.sh file too.

It's my first time with WSL because I need OpenFoam Linux (not blueCFD etc) and connect it to a Windows python script. I search on Google, here... and I also asked ChatGPT without getting a good answer.

python **windows-subsystem-for-linux** **openfoam**

Share Improve this question edited Apr 8 at 10:27 asked Mar 25 at 13:59

Follow


Chenmunka
811 7 24 26


Dio Bello
17 5

I'm really glad you changed the title on this. I never bothered reading the first version you posted a few days ago since I don't know anything about OpenFOAM, but you are correct that this isn't really an OpenFOAM-related question. – NotTheDr01ds Mar 29 at 11:35

I realized that the title was too specific, as the solutions can be applied to many programs between wsl and windows. Before no one answered after reading "OpenFOAM" in the title. Like "what is OpenFOAM?" XD – Dio Bello Mar 30 at 0:56

1 Answer

Sorted by: Highest score (default)

2

Short version

- Examine your `~/.bashrc`, `~/.bash_profile`, and `~/.profile` to determine where `$FOAM_RUN` is being set.
- If it's defined in `~/.bashrc`, then move its definition to `~/.profile`
- Change your command definition in the first script to:

```
command = "wsl -e bash -lc 'python3 /mnt/wslg/distro/home/j/test.py'"
```

Also see the bottom of this answer for a better method, but it may be overkill here.

Side-note: `/mnt/wslg/distro/home/j/test.py` is a bit of an odd path. While it works, it should be the same as the shorter `/home/j/test.py`. The `/mnt/wslg` mount is used *internally* by WSLg and may change in the future (it has already changed many times in the past). There is no guarantee that `/mnt/wslg/distro` will continue to point to root of the distro in future WSL releases.

More detail and other options

Most likely, `$FOAM_RUN` is defined in your `~/.bashrc` or equivalent. This startup config is only sourced when you are starting an *interactive* shell. When the first Python script runs:

```
wsl python3 /mnt/wslg/distro/home/j/test.py
```

There's an implicit call to `bash -c` first (or whatever your default shell is):

```
wsl -e bash -c 'python3 /mnt/wslg/distro/home/j/test.py'"
```

By default, when executing a commandline with `-c`, Bash does *not* run interactively, and so `~/.bashrc` is never read. On some distributions, the stock `~/.bashrc` even goes further to try to prevent interactive use.

You don't mention if running the:

```
wsl python3 /mnt/wslg/distro/home/j/test.py
```

... works from PowerShell or CMD. If my answer is correct, this should fail with the same message.

You have a few options:

- First, and probably the right way to do this if you need `$FOAM_RUN` to be available in a non-interactive shell, is to move its definition from `~/.bashrc` to `~/.profile`. The later is read for *login* shells. Then have WSL tell `bash` to run as a login shell so that file will be processed. The command line in the "short version" above:
  - Uses `wsl -e bash` to start Bash explicitly. This allows us to pass along the options ...
  - `-lc`: The `l` forces a login shell, which forces `.profile` to be sourced when starting.
  - The `-c` is used to execute the next command (your Python script invocation) in the shell.

- Another option would leaving the variable definition in `~/.bashrc` and then changing your commandline to:

  ```
  command = "wsl -e bash -ic 'python3 /mnt/wslg/distro/home/j/test.py'"
  ```

  The only difference being that we now start Bash with `-i` instead of `-l`.

  This tells Bash that you are starting an "interactive" shell and sources `~.bashrc`. I generally don't think this is a great idea, because you *should* be able to assume that an interactive shell can process (obviously) interactive commands - I.e., those that require input.

  If you do happen to have commands that require input in your `~/.bashrc`, then your Python script is going to pause and wait for input. That may not work in a `subprocess` call - I can't remember for certain. Regardless, it's probably not what you want to happen in a test script.

- Finally, the "true and proper" best method is likely to:
  - Define the `$FOAM_RUN` in the *Windows* (calling) Python environment
  - Also define (or modify) the `WSLENV` variable to include `FOAM_RUN` as an environment variable that you want to pass from the Windows process to the WSL environment.

  See Share Environment Vars between WSL and Windows and the corresponding WSL Doc Page for more info.

Share Improve this answer Follow

answered Mar 29 at 11:33

NotTheDr01ds
18.5k 7 55 81

Thank you very much, after reading your solutions I couldn't get the program to run, however I managed to understand that the problem started from bashrc. When I install OpenFOAM for the first time one thing I did is to define the openfoam bashrc in the bashrc "HOME" with "source /opt/openfoam11/etc/bashrc". So I tried to add it: "wsl "source /opt/openfoam11/etc/bashrc && python3 /home/j/test.py" and it works! I know, probably isn't the most efficient solution, in the next few days I will try to work one of your recommended methods. – Dio Bello Mar 30 at 0:53

@DioBello The workaround you have is perfect as well. Regardless of the method used, yes, the ultimate goal is to make sure that the code that normally runs in `~/.bashrc` still gets run when launching via `wsl ...` . Good to hear you have it running! – NotTheDr01ds Mar 30 at 16:01