runxtb.bash

This script provides a wrapper for the extended tight-binding semi-empirical program package xtb (version 6.0 or later) from Stefan Grimme's group at the University of Bonn (contact: xtb{at}thch.uni-bonn.de). It is available via GitHub.

The runxtb script makes it unnecessary to set environment variables like `OMP_NUM_THREADS`, `MKL_NUM_THREADS`, `OMP_STACKSIZE`, and `XTBPATH` globally, for example via the `.bashrc`. It also provides a mechanism to automatically trap the output that would normally go to standard out (i.e. the terminal). Additionally it can be used to create scripts to submit it to a queueing system.

Installation

A more elaborate guide on how to set up runxtb and xtb can be found in the guides directory: guides/set-up.md.

Briefly:

Obtain the latest version of xtb (and crest) from its GitHub page. Pick a directory, where you would like to install these packages, e.g. a separate user (here: software) and the root directory `/home/software/chemsoft/xtb`. Unpack the downloaded xtb archives (this may create a new directory for the downloaded version, which should be set as `xtb_install_root` for runxtb). Copy crest to the `bin` directory within there.

Get the latest release of this repository from GitHub. You may want to install it alongside the original binary, e.g. `/home/software/chemsoft/xtb`. Unpack the contents, and if necessary rename the directory. Alternatively you can of course also clone this repository.

Change to the runxtb directory. If you have a previous version of the script, it might be possible to import your settings. This is dependent on where your installed them. Now you can simply run the configure script, which will prompt for the values with a short description. If you are using a fresh installation, it will recover the default settings from the supplied `runxtb.rc`. At the end, you will be prompted for a location for the configuration file. Keep in mind that the wrapper script will first look for a file `.runxtbrc`, then for a file `runxtb.rc`, in directories of the following order: `scriptpath`, `/home/$USER`, `/home/$USER/.config`, and `$PWD`. If a `.runxtbrc` is found, it will skip `runxtb.rc`. The location the script has predefined is `$PWD/runxtb.rc`, hence overwriting the provided example file, if it is launched from the installation directory of runxtb. Alternative options (e.g. `'scriptpath'/.runxtbrc`) are also given. After that, the configuration script will ask about creating symbolic links in `~/bin`. If you choose to do that, you can basically access the script from anywhere in your file system.

Updating

If you decided to clone the repository, make sure to stash your changes, then pull the most recent version. If you have stashed any changes, you may want to apply them now. After running the configuration script, everything should be working with the new version.

You can of course also install a fresh version of runxtb. If you want to import older settings, copy your configuration file to the root of runxtb, or supply the absolute path to the settings when prompted by the configuration script.

When downloading a new version, you should always reconfigure the script.

Usage and options

A more detailed description of the usage and the options can be found in the guides directory: guides/usage-opts.md.

Brief overview: If linked (or found) from a directory along the `PATH` environment variable, you can call the runxtb wrapper script as if you would call the original program:

```
runxtb.sh [script options] <coord_file> [xtb options]
```

Any provided options to runxtb used will overwrite the corresponding rc settings, which in turn take precedence over the built-in defaults. If same options or modes are specified multiple times, only the last one will have an effect, e.g. specifying `-sSi` will run interactively (immediately).

The following script options are available:

Option	Description
`-p <ARG>`	Specify the number of processors to be used. (Default: `4`)
`-m <ARG>`	Specify the memory to be used (in megabyte). (Default: `1000`)
`-w <ARG>`	Specify the wall time to be used (`[[HH]:[MM]:]SS`). (Default: `24:00:00`)
`-o <ARG>`	Trap the output (without errors) of `xtb` into a file called `<ARG>`. (Default: `auto`)
`-s`	Write a submit script instead of interactive execution (default: PBS). Resulting file needs to be submitted.
`-S`	Write submit script and directly submit it to the queue, requires setting a queueing system with `-Q`.
`-Q <ARG>`	Set a queueing system for which the submit script should be prepared, supported `pbs-gen`, `slurm-gen`, `bsub-gen`.
`-P <ARG>`	Account to project or account `<ARG>`, only slurm, LSF (bsub).
`-M`	Use pre-installed modules instead of path settings, also needs specified module(s).
`-l <ARG>`	Specify a module to be used, will also invoke `-M`, may be specified multiple times to create a list, `<ARG>=0` clears the list.
`-i`	Execute in interactive mode (default without configuration).
`-B <ARG>`	Set the absolute path to the executable `xtb` to `<ARG>`. The name of the program needs to be included.
`-C <ARG>`	Set the name of the program directly, e.g. to access a different executable like crest (if installed).
`-q`	Suppress any logging messages of the script, specify twice, to also suppress warnings, specify more, also suppress errors.
`-h`	Prints a small help text and current configuration.
`-H`	Retrieve the man page of xtb of the original xtb distribution.
`-X`	Retrieve the man page of xcontrol of the original xtb distribution.

Included files and directories

The following files and/or directories come with this script:

- `runxtb.sh` The main wrapper.
- `runxtb.rc` An example set-up file.
- `xtb.dummy` A tiny bash script only for testing. This will only echo `<coord_file> [options]` verbatim.
- `crest.prepare.sh` A small script that creates a new directory with a suitable `coord` file to start a

`crest` run.
- `README.md` This file.
- `configure` A directory containing a script to configure the wrapper.
- `guides` A directory with markup-formatted file(s), which can be used as tutorials.

Exit status

The script `runxtb` carries over the exit status of its dependencies. In interactive mode that is the exit status of `xtb`. In submission mode it is the exit status of `qsub`, `bsub`, or `sbatch`. In all other cases it will be `0` if everything went according to plan, or `1` if there was a problem within the script.
The script `crest.prepare.sh` will exit with `0` if nothing went wrong and a crest run can be started. If files are not present, or I/O operations fail it will exit with `1`.
The dummy script `xtb.dummy` always exits with `2`.

Debug

Things go wrong.
We need to accept that. To find out more, using `debug` as the very first argument gives more information. For example:

```
runxtb.sh debug -p1 -qq -s  dummy.xyz --opt
```

If you find anything not going as expected, please include the debug output when submitting a bug report to the GitHub issue tracker.

License (GNU General Public License v3.0)

(Martin; 2020-04-09; wrapper version 0.4.0)