Installing ChaNGa

Lucas Caudill
(Dated: June 17, 2020)

This document outlines the process used to install ChaNGa on a Ubuntu machine as performed by the author in May of 2020. This process is derived from the information outlined in Section I of *Summer 2019 N-body Astrophysics Simulation Research Overview* by W. Lum, B. Cummings, and J. Powell. It is likely that future versions of ChaNGa and/or Charm++ will render the instructions within this document obsolete. Given the often confusing nature of installation, however, the reader is encouraged to attempt to replicate this process if deemed appropriate.

I. CHARM++

A. Installation and Setup

The first step to installing ChaNGa, as described on the github wiki [1] is to install Charm++ from the University of Illinois at [2]. Note that, as per the recommendation of [3], version 6.8.2 of Charm++ should be installed rather than the latest release to avoid issues when compiling ChaNGa. As of May 2020, this can be done by clicking the "Download Source Code" button on the left side of the page (see Fig. 1) and selecting "Charm 6.8.2 Source Code". This will download a tarball file containing the desired version of Charm++ to your machine.

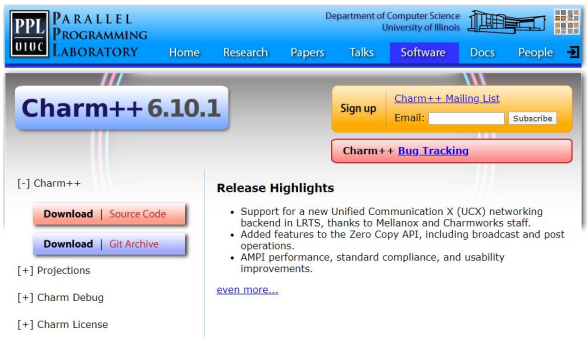

FIG. 1. The Charm++ webpage, as of May 2020.

Once installed, the tarball file will need to be extracted, and the resulting directory will need to be moved to the desired location within the machine's file system. The easiest way to extract the tarball is to use your machine's user interface to navigate to the Downloads folder and right-click on the file, selecting the option to extract. Alternatively, the tarball can be extracted from the Linux terminal if the user navigates to the `Downloads` directory and inputs the following command:

```
$ tar -xvf [filename]
```

Extracting the tarball should add a directory named `charm-[version]` (in this case, `charm-6.8.2`) to the `Downloads` directory. At this point, the user needs to rename this directory to "charm" and move it elsewhere within the directory tree. I recommend doing both simultaneously using the `mv` command, for instance, the command

```
$ mv charm-[version] ~/charm
```

will rename the directory and move it to `/home/[user]/`. As described in [1], the user should then navigate to the charm directory and enter the command

```
$ ./build ChaNGa netlrts-linux-x86_64 --with-production
```

to build charm with the appropriate ChaNGa libraries.

B. Issues

The charm github page [4] includes details about a test program within the charm directory tree called `hello` located in `/netlrts-linux-x86_64/tests/charm++/simplearrayhello/`. To run this test program with charm++6.8.2, you must first navigate to the `simplearrayhello` directory and run the `make` command to compile the source code into the `hello` and `charmrun` executables. To run the test, you can enter the command

```
$ ./charmrun +p1 ./hello
```

This command runs the test on a single processor. To run with an alternate number of processors, substitute `1` with the desired number

The expected output of the `hello` test program is documented on [4]. When attempting to run this test, however, I encountered an error which I had trouble finding documentation for online (see Fig. 2).

The solution to this error (for reasons I do not fully understand) is to generate an ssh key on the user's machine and to authorize that key on the machine. To do this, you must first go to the `.ssh` directory using the following terminal command

```
$ cd ~/.ssh
```

Note that this command can be used from within any directory.

If you are encountering the error in Fig. 2, it is likely that you will find the `.ssh` directory to be empty.

```
Charmrun> scalable start enabled.
lucas@localhost: Permission denied (publickey,password).
Charmrun> Error 255 returned from remote shell (localhost:0)
Charmrun> Reconnection attempt 1 of 3
lucas@localhost: Permission denied (publickey,password).
Charmrun> Error 255 returned from remote shell (localhost:0)
Charmrun> Reconnection attempt 2 of 3
lucas@localhost: Permission denied (publickey,password).
Charmrun> Error 255 returned from remote shell (localhost:0)
Charmrun> Reconnection attempt 3 of 3
lucas@localhost: Permission denied (publickey,password).
Charmrun> Error 255 returned from remote shell (localhost:0)
Charmrun> Too many reconnection attempts; bailing out
```

FIG. 2. The terminal output resulting from my initial attempts at running the `hello` program. Note that `lucas` (seen in output line 2) is the name of the user.

The first step to resolving this issue is to generate an `id_rsa.pub` file with the following terminal command

```
$ ssh-keygen
```

After inputting this command, the user will be prompted to enter a file in which the key will be stored. This option can be ignored by simply pressing the `[ENTER]` key. The user will then be prompted to provide a password. The user can again press `[ENTER]` to generate a key with no password, or can set a password if so desired.

After doing this, the user should find a `id_rsa.pub` file in the directory. The user should then read the file with the command

```
$ cat id_rsa.pub
```

and copy the resulting terminal output. The user should then look for a file named `authorized_keys` in the `.ssh` directory. If the file exists, simply paste the key you just copied into a new line in the file. If no such file exists, you'll need to make a blank `authorized_keys` file using your preferred text editor (I like emacs personally) and paste the key into that file instead. Once you've added the key to the `authorized_keys` file, you should no longer encounter the error in question.

II. CHANGA

If you've made it this far, you likely have a working version of charm++. Congratulations! The next step is obtaining and setting up ChaNGa.

First, you should navigate to the directory in your file system that contains the `charm` directory that was installed in Section I. This means that when you enter the terminal command `ls`, `charm` should be listed as a subdirectory. Starting from directory that contains `changa` ensures that you need not set the environment variable `CHARM_DIR`, as described in [1]. It should be noted that if you are comfortable with setting the environmental variable, then you can put ChaNGa anywhere in your file system, regardless of the location of `charm`.

Once you've navigated to the directory where you intend to install ChaNGa, enter the following commands (as per the instructions on [1])

```
$ git clone https://github.com/N-BodyShop/changa.git
$ git clone https://github.com/N-BodyShop/utility.git
```

These commands will create two new subdirectories, `changa` and `utility`, in your current directory. You will now need to switch to ChaNGa version 3.3 [3]. This can be done by navigating to the `changa` directory and entering the command

```
$ git checkout v3.3
```

You may consider running the `configure` file with the command

```
$ ./configure
```

To ensure that the change in version is applied. You can verify that you've successfully switched versions by entering

```
$ ./configure -V
```

Additionally, you can use the command

```
$ git branch
```

to check that you've successfully switched to the v3.3 GitHub branch of ChaNGa. These two commands should produce terminal outputs similar to those displayed in Fig. 3.

```
(base) lucas@ubuntu-MS-7A54:~/changa$ git branch
* (HEAD detached at v3.3)
  master
(base) lucas@ubuntu-MS-7A54:~/changa$ ./configure -V
ChaNGa configure 3.3
generated by GNU Autoconf 2.69

Copyright (C) 2012 Free Software Foundation, Inc.
This configure script is free software; the Free Software Foundation
gives unlimited permission to copy, distribute and modify it.
```

FIG. 3. The terminal outputs of `git branch` and `./configure -V` after successfully switching to ChaNGa version 3.3

At this point, attempting to compile ChaNGa with the `make` command will likely produce a compiler error reporting that the variable `HUGE` in the file `InOutput.C` is undefined. In ChaNGa v3.3, this variable should be present in lines 55 and 568 of `InOutput.C` (though it may be worth double checking the compiler error to make sure this is the case).

The solution to this compiler error, as documented in [3], is to edit `InOutput.C` using your preferred text editor, going to each line containing the variable `HUGE` and replacing it with `FLT_MAX`. Doing so should resolve the compiler error. Enter the command

```
$ make
```

If successful, this will add the executables `charmrun` and `ChaNGa` to the `changa` directory, and you should have a working version of ChaNGa. You can test your version of ChaNGa by navigating to the `testcosmo` directory and running the tests documented on the "Running ChaNGa" section of the ChaNGa wiki at [1].

[1] "N-BodyShop/changa,", URL https://github.com/N-BodyShop/changa/wiki/Building-ChaNGa.
[2] "Charm++,", URL http://charm.cs.uiuc.edu/software.
[3] W. Lum, B. Cummings, and J. Powell, "Summer 2019 N-body Astrophysics Simulation Research Overview," (2019).
[4] "UIUC-PPL/charm,", URL https://github.com/UIUC-PPL/charm/wiki/.Quickstart.