A Blue Waters Usage Guide

Aaron Weeden, Shodor
2015

Table of Contents

I. Logging in

A. Windows

1. Download, install, and open PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
2. For the Host Name, enter **bwbay.ncsa.illinois.edu**.
3. Click the **Open** button.
4. If a **PuTTY Security Alert** window appears, click **Yes**.
5. Enter your username.
6. Enter your password.
7. The first time you connect, type **y** to accept the terms of use, and fill in your personal information.

B. *Mac*

1. Open the **Terminal** application under Applications -> Utilities.
2. Type in the following command, replacing **<your username>** with your actual username on Blue Waters (spaces are highlighted), and hit enter:

```
ssh <your username>@bwbay.ncsa.illinois.edu<ENTER>
```

3. Enter your password.
4. The first time you connect, type **y** to accept the terms of use, and fill in your personal information.

II. Working with directories and files

Blue Waters uses a Linux operating system with a command-line interface. This means that the primary way of interacting with Blue Waters is by typing in commands. Here are a few notes to remember:

1. Any time you wish to cancel a command that is currently running, use **Control-C** (hold the 'Control' key and press the 'C' key).
2. There is a history of commands you have already typed, which you can access in two ways:
    - Use the up-arrow and down-arrow keys to fill the command line with a previous command you entered. To run one of these commands again, just hit enter.
    - Type the following command to see a list of all the commands you have entered:

```
history<ENTER>
```

3. As you type the name of a file, if you have typed enough letters to distinguish it from the names of other files in the same directory, you can type **<TAB>** and it will complete the name for you.

Like any Linux operating system, Blue Waters has **directories** (folders) and **files**. At any given time there is a directory called the **working directory**, in which you can think of yourself as being physically located. You can view the name of the working directory at any time using the following command:

```
pwd<ENTER>
```

This will print out the **path** to the directory from the **root** directory of the filesystem (the filesystem is stored as a tree, with the root at the top). Each directory in the path is separated by slashes. For example, when I log in and enter the **pwd** command, I get back the following path:

```
/u/training/aweeden
```

This means that in the root directory, there is a **u** directory, inside that is a **training** directory, and inside that is an **aweeden** directory, which is also my username. I can see what files and directories exist in the working directory by typing this command:

```
ls<ENTER>
```

This will list out the names of all the directories and files in the working directory. If no list appears, it means there are no directories or files in the working directory.

You can create a directory using the following command (make sure to choose a name that does not have spaces in it - this will make it easier to work with):

```
mkdir <directory name><ENTER>
```

After you make a directory, when you enter the **ls** command from above, you should see the name of the directory you just created in the list.

If you want to change the working directory, you can do so using the following command (e.g. you could try this on the directory you just created):

```
cd <directory name><ENTER>
```

Now the **pwd** command will show the name of the directory that you changed into. **ls** will show you what is in that directory.

If you want to change to the directory that contains the working directory (e.g. **training** contains **aweeden**), you can use the following command:

```
cd ..<ENTER>
```

The two dots refer to one directory “up” or “back” in the filesystem (towards the root). If you wanted to go up two directories, it would instead be this command:

```
cd ../..<ENTER>
```

If you ever get lost in the filesystem and want to return to your home directory, you can simply type this command:

```
cd<ENTER>
```

III. Editing files

To edit a file (including one that does not exist yet), use the vi text editor:

```
vi <file name><ENTER>
```

vi has two modes: **command mode**, which is the default when you open the file, and **insert mode**, which you can enter by pressing the 'i' key. You can verify you are in insert mode if at the bottom of the window it says **-- INSERT --** .

To go back to command mode from insert mode, press the 'Esc' key; at the bottom of the window it will no longer say **-- INSERT --**.

Try this now; keep switching between insert and command mode until you feel like you have the hang of it.

To scroll through the file, you can use the up-arrow and down-arrow keys or **Command-F** to go forward one page and **Command-B** to go backward one page. In command mode, you can type **gg** to jump to the top of the file and capital **G** to jump to the bottom of the file. You can search in command mode by pressing slash (*/*), typing a search term, and pressing enter.

If you are in insert mode, you can type things to edit the file. Once you are done editing, press **Esc** to enter command mode, then type the following command to save. If you are in command mode, you will see the letters at the bottom of the window as you type:

```
:w<ENTER>
```

When you are ready to quit the file, enter the following command:

```
:q<ENTER>
```

There are two other useful vi commands for quitting. The following command will both save and quit:

```
:wq<ENTER>
```

The following command will quit without saving:

```
:q!<ENTER>
```

IV. Requesting a job

Batch job - requires a script file

General form:

```
qsub <script file><ENTER>
```

Example:

With tabs:	`qsub ~awee<TAB>f<TAB>/o<TAB>s<TAB>p<TAB><ENTER>`
Without tabs:	`qsub ~aweeden/fire/omp/scale-omp.pbs<ENTER>`

You can use **vi** to view a sample batch script:

With tabs:	`vi ~awee<TAB>f<TAB>/o<TAB>s<TAB>p<TAB><ENTER>`
Without tabs:	`vi ~aweeden/fire/omp/scale-omp.pbs<ENTER>`

Interactive job - allows you to type in commands yourself

General form:

```
qsub <parameters><ENTER>
```

Example:

```
qsub -I -l nodes=1:ppn=32:xe -l walltime=01:00:00<ENTER>
```

Once you request an interactive job, it may take a few minutes, since you are competing with other users of Blue Waters to be automatically scheduled for time on the cluster.

When the job starts, you will see a “Torque Prologue”, followed by a new prompt on which you can enter commands again, including running programs (see the section *Running Programs* below). After you have run some programs and are ready to terminate the interactive job, type **Control-D** (hold down ‘Control’ key and press the ‘D’ key).

Whether you are requesting a batch or interactive job, certain parameters can be used to control the job. To specify parameters in a batch script, use lines that start with **#PBS**. To specify parameters for an interactive job, type them directly on the command line after the word **qsub**, separated by spaces. Here are some useful parameters:

-l walltime=<hours>:<minutes>:<seconds>
Maximum amount of time granted to the job.

-l nodes=<number of nodes>:ppn=<number of cores>:<either>xe<or>xk
How many nodes you want to run on, how many cores you want to use per node, and whether you want to use XE nodes (32 CPU cores, no GPGPU) or XK nodes (16 CPU cores and 1 GPGPU).

-o <file>
Store the output of the job in a file.

-e <file>
Store the errors of the job in a file.

V. Monitoring job status

To see all jobs that you currently have running, use the following command:

```
qstat -u <your username><ENTER>
```

This prints out a table of information, with one row for each job. The second-to-last column tells you the status of the job: **Q** means waiting to run, **R** means currently running, and **C** means complete. To the left of the status is the maximum amount of time the job can run, and to the right of the status is the amount of time the job has already been running.

In the far left column is the numerical ID of the job. You can use this ID to cancel the job by entering the following command:

```
qdel <job ID><ENTER>
```

When the job finishes, two files will be created: **<script file>.o<job ID>**, which contains the output of the job, and **<script file>.e<job ID>**, which contains the errors of the job. The names of these files can be overridden when submitting the job using the **-o** and **-e** parameters, respectively, as mentioned in the explanations above for requesting a job.

VI. Running programs

Once an interactive job has started, you can use the following commands to run a program. Note that this command should only be run as part of a job, so either in a batch script or after obtaining an interactive job:

```
export OMP_NUM_THREADS=<# of threads><ENTER>
aprun -n <# of processes> -d <# of threads> <executable><ENTER>
```

A program can execute in parallel using multiple **processes** and/or multiple **threads**. To run a program without any parallelism (that is, a **serial** program) or a program that uses a **GPGPU** (general-purpose graphics processing unit), use the following command:

```
aprun -n 1 <executable><ENTER>
```

There is an example program you can run below:

With tabs:	`export OMP_NUM_THREADS=1<ENTER>` `aprun -n 1 -d 1 ~awee<TAB>f<TAB>/m<TAB>-<TAB>f<TAB><ENTER>`
Without tabs:	`export OMP_NUM_THREADS=1<ENTER>` `aprun -n 1 -d 1 ~aweeden/fire/mpi-omp/fire-mpi-omp<ENTER>`

This will run a forest fire model similar to the model at http://shodor.org/interactivate/activities/Fire. To change how many **processes** are used, change the value after **-n**. To change how many **threads** are used, change the value after **OMP_NUM_THREADS=** and the value after **-d**.

To time how long it takes a command to run, use the word **time** at the beginning of the command, as in the example below:

```
time aprun -n 1 <executable><ENTER>
```

The **real** time that is reported is a measure of the wall clock time from the time you press enter to the time the program finishes running.

VII. Compiling programs

Use the following command to copy the source code for the forest fire program to your home directory:

With tabs:	`cp -r ~awee<TAB>f<TAB> ~<ENTER>`
Without tabs:	`cp -r ~aweeden/fire ~<ENTER>`

You can now change to the directory you copied using this command:

```
cd ~/fire<ENTER>
```

Inside this directory are different directories for each version of the program: **`omp`**, **`mpi`**, **`acc`**, **`cuda`**, and **`mpi-omp`**. You should see them all if you run this command:

```
ls<ENTER>
```

You can use **`cd`** to change into one of the directories. Inside each directory is a file called **`Makefile`**, which makes it possible to run the source code through a **compiler** and generate an executable program. To make this happen, change into one of the directories using **`cd`** and enter the following command:

```
make<ENTER>
```

If this gives the error "**`execvp: nvcc: Permission denied`**", then try running these commands instead:

```
module load cudatoolkit<ENTER>
make<ENTER>
```

This will take the source code file (e.g. **`fire-omp.c`**) and generate an executable file (e.g. **`fire-omp`**) which can be used for running a program (see the *Running Programs* section above).

If the source code has not been changed since the last time you compiled, this will probably give the message "**`make: `<executable>' is up to date`**".

To remove the executable file, type the following command:

```
make clean<ENTER>
```

If you have your own source code files, you can use the following commands to compile them into executable files:

Language	Command
C	**cc -o <executable file> <source file><ENTER>**
C++	**CC -o <executable file> <source file><ENTER>**
Fortran	**ftn -o <executable file> <source file><ENTER>**
C with OpenACC	**module load cudatoolkit<ENTER>** **module load craype-accel-nvidia35<ENTER>** **cc -o <executable file> <source file><ENTER>**
CUDA C	**module load cudatoolkit<ENTER>** **nvcc -o <executable file> <source file><ENTER>**

VIII. Viewing/editing source code

The sample source code for the forest fire model consists of 5 files written in the C language within the **fire** directory:

omp/fire-omp.c (OpenMP version)
mpi/fire-mpi.c (MPI+OpenMP version)
acc/fire-acc.c (OpenACC version)
cuda/fire-cuda.cu (CUDA version)
mpi-omp/fire-mpi-omp.c (MPI+OpenMP version)

Each of these can be viewed and edited using vi (see the explanations above for using vi). Once changes are made to one of the files, a new executable can be generated (see explanations above for compiling).

These programs can each generate an ASCII visualization of the model by adding **-o** to the end of the run command and providing an output file name, as shown below:

aprun -n 1 <executable> -o <output file name><ENTER>

Inside each of the **omp**, **mpi**, **acc**, **cuda**, and **mpi-omp** directories is a sample visualization output file that you can view using the **less** command, for example:

With tabs:	**less** **~awee<TAB>f<TAB>/o<TAB>f<TAB>.o<TAB><ENTER>**
Without tabs:	**less ~aweeden/fire/omp/fire-omp.out<ENTER>**

To advance the visualization, hold down the spacebar. You can move backwards using **Control-B**, jump to the top of the file with lower-case **g**, jump to the bottom of the file with capital **G**, and quit with **q.**

-o is not the only run-time parameter. Use **-?** to show the full list, as in the command below:

```
aprun -n 1 <executable> -?<ENTER>
```

IX. Additional Resources

- Blue Waters documentation (https://bluewaters.ncsa.illinois.edu/documentation)
- Resources from the 2015 Blue Waters Petascale Institute (http://shodor.org/petascale/workshops/bw2015/)