Our Workshop Environment

John Urbanic
Parallel Computing Scientist
Pittsburgh Supercomputing Center

Our Environment Today

Your laptops or workstations: only used for portal access.

Bridges-2 is our HPC platform.

We will here briefly go through the steps to login, edit, compile and run before we get into the real materials.

We want to get all of the distractions and local trivia out of the way here. Everything *after* this talk applies to any HPC environment you will encounter.

Getting Connected

- The first time you use your account sheet, you must go to apr.psc.edu to set a password. You may already have done so, if not, we will take a minute to do this shortly.

- We will be working on bridges2.psc.edu. Use an ssh client (a Putty terminal, for example), to ssh to the machine.

- At this point you are on a login node. It will have a name like “bridges2-login011”. This is a fine place to edit and compile codes. However we must be on compute nodes to do actual computing. We have designed Bridges to be the world’s most interactive supercomputer. We generally only require you to use the batch system when you want to. Otherwise, you get your own personal piece of the machine. To get a 8 processors on a node use “interact”:

```
[urbanic@bridges2-login011]$ interact -n 8
[urbanic@r590 ~]$
```

- You can tell you are on a regular memory compute node because it has a name like “r590” or “r101”.

- Do make sure you are working on a compute node. Otherwise your results will be confusing.

Editors

For editors, we have several options:

- emacs
- vim
- nano: use this if you aren’t familiar with the others

Compiling

We will be using standard Fortran and C compilers this week. They should look familiar.

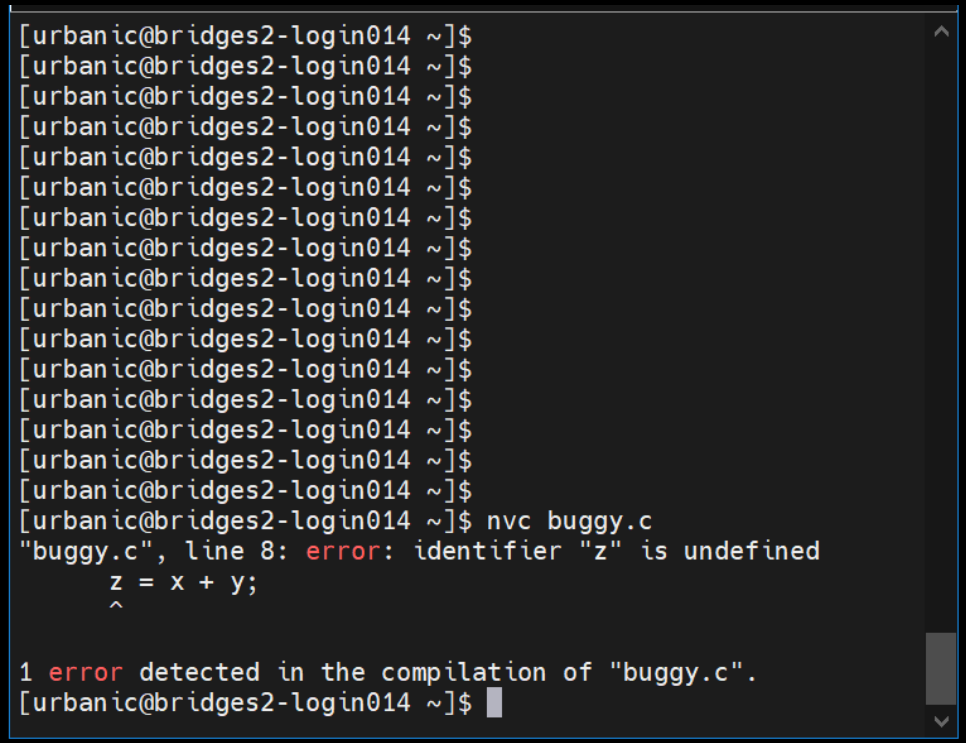
- nvc for C
- nvfortran for Fortran

We will slightly prefer the NVIDIA/PGI compilers (the Intel or gcc or AMD ones would also be fine). Note that on Bridges you would normally have to enable this compiler with

```
module load nvhpc
```

I have put that in the .bashrc file that we will all use.

I like two windows. One on login and one on compute.



Undefined y on line 8!

Poor man's IDE.

```
 GNU nano 2.9.8                  buggy.c

#include <stdio.h>

int main( ) {

    int x = 5;
    int y = 10;

    int z = x + y;

    printf("x + y = %d\n", z);

}




                [ Wrote 12 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text
```

Don't forget to save. Very common source of confusion. You keep fixing things to no effect?!?.

```
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$
[urbanic@bridges2-login014 ~]$ nvc buggy.c
"buggy.c", line 8: error: identifier "z" is undefined
      z = x + y;
      ^

1 error detected in the compilation of "buggy.c".
[urbanic@bridges2-login014 ~]$ nvc buggy.c
[urbanic@bridges2-login014 ~]$ a.out
x + y = 15
[urbanic@bridges2-login014 ~]$
```

Recompile, and run.

Our Setup For This Workshop

After you copy the files from the training directory, you will have:

```
/Exercises
        /Test
        /OpenMP
                laplace_serial.f90/c
                /Solutions
                /Examples
                /Prime
        /OpenACC
        /MPI
```

Preliminary Exercise

Let's get the boring stuff out of the way now.

- Log on to apr.psc.edu and set an initial password.
- Log on to Bridges2.

```
ssh username@bridges2.psc.edu
```

- Run the setup script that will copy over the Exercises directory we will all use. It will also automatically load the right compiler using your .bashrc script whenever you login.

```
~training/Setup
```

- As told, logout and log back on again to complete the setup. You won't need to do that in the future.
- User interact to grab some compute cores.

```
interact -n 32
```

- Edit a file to make sure you can do so. Use emacs, vi or nano (if the first two don't sound familiar).
- cd into your exercises/test directory and compile (C or Fortran)

```
cd Exercises/Test
nvc test.c
nvfortran test.f90
```

- Run your program

```
a.out
```

It should say *"Congratulations!"*