

Lab 1

Programming Environment, File
Systems, Modules, Batch System

LAB I: Program Environments

- login to the Sun Constellation system (ranger) using your train## login:

```
% ssh -X train##@ranger.tacc.utexas.edu
```



- Untar the file lab1.tar file (in ~train00) into your directory:

```
% tar xvf ~train00/lab1.tar
```

- Change into the lab1 directory:

```
% cd lab1
```

```
% ls
```

Programming Environment

- Try out a few of the following commands:

- `env`
- `df -h`
- `ls -la`
- `uname -a`
- `cdw`
- `echo $WORK`

File Systems – scp vs. bbcp

- Let's compare transfer speeds for encrypted (scp) and unencrypted (bbcp) transfer to *ranch.tacc.utexas.edu*
- Create a 512 MB file using the **dd** command

```
% dd if=/dev/zero of=zero_512 bs=1M count=512
```
- Transfer this file to ranch twice, noting the transfer speeds reported by each method. (You will need to type your password each time.)

```
% bbcp -V zero_512 ${ARCHIVER}:${ARCHIVE}
% scp zero_512 ${ARCHIVER}:${ARCHIVE}
```
- After completing the transfers, remove the data files!

```
% rm zero_512
% ssh ranch rm zero_512
```

Modules

- List the arguments available in the module command
`% module`
- List the modules that are presently loaded
`% module list`
- List the modules that are available
`% module avail`

- Determine which mpicc is being used and switch compiler/MPI stacks
`% which mpicc`
`% module swap pgi intel`
`% which mpicc`
`% module swap mvapich mvapich2`
`% which mpicc`

Module spider

- “module avail” tells what packages are available with current compiler and mpi implementation
- “module spider” lists all packages independent of compiler or mpi stack
- Try:
 - % module spider
 - % module spider petsc
 - % module spider petsc/3.1-cxx

More Modules

- Test Compiler Family

```
% module purge; module load TACC
```

```
% module load intel
```

→ You will get an error

```
% module swap pgi intel
```

→ Message about reloaded modules

```
% echo $TACC_FAMILY_COMPILER
```

Creating your own default setup

- Create your own initial modules for login (test on ranger)

```
% module purge; module load TACC
```

```
% module load git boost petsc
```

```
% module setdefault
```

SGE Batch

- `cd` to `$HOME/lab1/batch`.

- Compile the simple “hello world” fortran or C MPI code

```
% mpif90 -O3 mpihello.f90 -o mpihello
```

OR

```
% mpicc -O3 mpihello.c -o mpihello
```

- Look over the “job” script, and submit the program to SGE

```
% qsub job
```

- Watch the status of your job (you will have to be quick to see something!)

```
% watch showq -u -l (Ctrl-C to quit watching)
```

- Now, put a “sleep 60” statement in the jobs script, resubmit it, & delete the job. (Edit the job file with emacs or vi, see provided reference if needed)

```
% qsub job (observe the returned jobid or determine it from bjobs)
```

```
% qdel jobid
```

Precision

- Look over the precision.f program in the **precision** directory.
% `cd $HOME/lab1/precision`
- The program computes prints $\sin(\pi)$. The π constant uses “E” (double precision) format in one case and “D” (single) in the other.

```
% ifort -FR precision.f  
(or)  
% ifort precision.f90  
% ./a.out
```

(The ifc compiler regards “.f” files as F77 fixed format programs. The -FR option specifies that the file is free format.)

Makefiles

- **cd** to the using_makefiles directory.
- Read over the Makefile.
- Compile the program (this generates a.out)
% **make**
- Simulate a change to one of the source files.
% **touch suba.f**
- Execute make again. Only suba.f should be recompiled!

Library Generation

- Generate a library (a collection of object files) with the **ar** command.

```
% ar rv libsub.a suba.o subb.o
```

- Now use the library when building a new a.out.

```
% ifort -FR -O3 prog.f libsub.a
```

- All system software uses (.a or .so) libraries.
- You can link system libraries into your personal code in the same manner.