

MPI How-To

This document is not intended to teach anyone how to program using the MPI interface. This is also not the place to be if you are wishing to install a parallel programming environment of your own. For help on these two topics, try the [MPI homepage](#), the [Lam-Mpi homepage](#) (this is the software used in the lab), the [MPI forum](#), or this [really good tutorial](#).

The intention of this document is to teach the steps necessary to boot a parallel programming environment already configured in the College of Wooster Computer Science Lab.

Lab rules prohibit occupying machines in a manner that prevents other students from using them. The exception to this rule is if a student has permission from an instructor to do so. Any student wishing to use the parallel computing environment should contact the lab intern to reserve a specific number of computers for an evening. Any complaints received regarding a student occupying multiple machines during an unscheduled time will be regarded as a violation of lab rules and may result in a loss of privilege for the person using the lab at that time.

Getting Started

- In Taylor 200, boot the desired number of parallel systems into the Linux Operating system. Choose any one computer as your master computer.
- The best place to start is in /home/cslab/Documents/lamWkDir/. There are a few useful files in this directory.
- mass_copy_files: This file is an extremely simple shell script that will synchronize any set of files across all the systems you are using. The only requirement is that the destination folder already exist on each system, and that you only copy files--no directories. This file will be used at a later point in this document.
- hostfile: this file is used by LAM to start the parallel environment. It is merely a list of IP addresses, so you can create a new one of your own, but what's the point? There is more to hostfiles than just listing IP addresses. Other options, such as the number of processes to be run on a certain machine, can be set in this file. For more information regarding this technique, refer to the LAM user's guide.
- user.pdf: This is the LAM user's guide. It is a very thorough document and if you ever wish to do something more advanced, you should read this first.
- install.pdf: This is the LAM installation guide. It is probably not as useful, but may answer some questions you have about how LAM works.
- README: The readme document contains a smaller portion of what is already in this document. It serves as a good quick reference if you forget how to boot the environment, stop the environment, etc.
- The remainder of the files in the directory were used for testing and are not important.

Compilation

lamWkDir is nice to work out of, but you should create your own so that the lamWkDir does not get too messy. Either create a new directory in the Documents folder, or you can use a USB flash key instead. Please note that there are several flash keys that for whatever reason are not compatible with SuSE Linux 9.1. This is a known issue, and there is very little that can be done about it. But if you are able to get to the files on your flash drive, you may work off of that as well.

Copy the following files into your new working folder:

```
hostfile
mass_copy_files
```

Now get some source to work with. One good place to look is /usr/src/packages/SOURCES/lam-7.1.1/examples/. For example, in that folder, copy hello.c (the C version) from the hello directory to your working folder.

Compile the file, using:

```
> mpicc -o myHello hello.c
```

LAM configures a series of system variables to aid in the compilation process. These variables are 'wrapper' commands. Basically, there are several libraries and packages that must be linked in order for your MPI code to work correctly. LAM provides the convenience of feeling like you are just using gcc, g++ and F77 to compile. mpicc will compile your C applications, mpiCC will compile your C++ applications, and mpif77 will compile your fortran applications. Note that you can use any typical gcc flags with these commands. Also note that unlike gcc, mpicc will not autodetect file extensions because different languages require different libraries, so you must be explicit as to what compiler you wish to use.

Booting LAM

Now that you have compiled your code, it is time to boot the LAM environment. Open your hostfile. Here is what you will see inside:

```
#My boot schema
#cs-lab-1 through cs-lab-14
140.103.108.219 cpu=1
140.103.108.231 cpu=1
140.103.108.212 cpu=1
140.103.108.232 cpu=1
140.103.108.242 cpu=1
140.103.108.234 cpu=1
140.103.108.237 cpu=1
```

```
140.103.108.235 cpu=1
140.103.108.201 cpu=1
140.103.108.204 cpu=1
140.103.108.236 cpu=1
140.103.108.233 cpu=1
140.103.108.239 cpu=1
140.103.108.247 cpu=1
```

If you number these from 1 to 14, they correspond to the computers in the lab labeled 1 through 14. Note that line 4 has a '#' comment in front of it, causing LAM to skip this machine when booting the environment. This is legal, while placing comments after the IP address seems to cause problems. Note the `cpu=1` after each IP address. This does not refer to the number of processors on that machine, but rather the desired number of processes on that machine. This can be a number greater than 1, even if the machine has only one processor. Having more than one process on a node is a more advanced topic than this tutorial covers, and you should refer to `user.pdf` in the `lamWkDir` to learn more.

Finally, you can use `mass_copy_files` to send your executable out to each machine you need.

1. Open the file with a text editor and change `file` and `file2` to be the full path to the file on the machine you are using, and the full path where the file will be copied on all the destination machines. Here is an example, excluding the series of 'scp' commands:

```
#number these 1-14 and you will know which machine you are
working with
```

```
file=~/Desktop/myWkDir/myHello
```

```
file2=/home/cslab/bin/myHello
```

2. Again, imagine the lines of the file are numbered 1-14 and comment out the machines you are not using. Note that the machine you are sitting at is also present in this file. This ensures that the executable is truly in the same place on every machine in the cluster.

3. Exit the text editor, and on the command line, type:

```
> ./mass_copy_files
```

to execute the script and synchronize the files across all the machines. By the example above, this command will copy `myHello` to `/home/cslab/bin/` on every workstation that is not commented out in `mass_copy_files`.

4. Finally, start the lam environment at the command line by typing:

```
> lamboot -v hostfile
```

where 'hostfile' is your list of [ip adress][cpu #] pairs.

Running an MPI program in LAM

Finally, your code is compiled, lam is booted, and your executable is copied to a common location on every machine you are using. To run the myHello program, assuming it is in /home/cslab/bin on each computer, type:

```
> mpirun C /home/cslab/bin/myHello
```

The C argument says to run one copy of myHello on every cpu in the LAM environment. Other arguments exist, though this is the simplest and most-used argument. The reader should refer to the mpirun manpage, or consult user.pdf for more information on other flags that can be used.

Shutting Down LAM

Shutting down LAM gets its own section because it is very important. Failure to shut down LAM will leave zombie processes running on every machine you used for your program. To shutdown normally, type:

```
lamhalt
```

if that should fail, type:

```
> lamclean -v hostfile
```

to force a shutdown.

Summary

Compiling and running MPI applications can be a complex process. To summarize the steps explained in this document:

1. Write your MPI code.
2. Compile your code using mpicc for C applications, mpiCC for c++ applications and mpif77 for fortran-77 applications.
3. Boot the lam environment, using a hostfile listing the IP addresses, one per line, for each machine in your cluster.
4. Copy the file to a common directory, such as /home/cslab/bin/ using mass_copy_keys or any other method you find convenient.
5. Run the program using mpirun.
6. Shut down the LAM environment using lamhalt or lamclean, if lamhalt fails.

For more information

The user's guide (user.pdf in /home/cslab/Documents/lamWkDir) is a very thorough document explaining how to use the system. Most of your questions will have answers there. If not, refer to lam-mpi.org where you can find additional documentation, discussion boards, and useful tutorials.