

Notes for Users of the Department of Economics SUN Workstations¹

Mark J. Roberts
Brad Winters
Updated March , 2008

I. General Description

The department has five SUN workstations. All of them run the Solaris operating system. The names of the machines and their specifications are:

kodiak.econ.psu.edu

This is a SUN V445 workstation with four 1.6 ghz CPUs and 16GB of RAM. This is currently our fastest machine and is meant for batch processing of computationally intensive programs. Smaller programs or programs that are run interactively should be run on one of the other machines.

beaver.econ.psu.edu

This is a SUN V440 workstation with four 1.6 ghz CPU's and 16GB of RAM.

grizzly.econ.psu.edu, wildcat.econ.psu.edu

These are SUN Blade 2000 workstations with two 900 mhz CPUs and 2GB of RAM.

bighorn.econ.psu.edu

This is a SUN Ultra 80 Model 4450 workstation with four 450 mhz processors and 2GB of RAM.

We also have two IBM workstations with a total of four processors available for econ faculty and grad students. See the description at <http://www.econ.psu.edu/~mroberts/sp2notes.pdf>

II. Login Procedures

The department's SUN workstations are open to use by the faculty, staff, and graduate students in the economics department. To use the workstations you will need your Penn State accessid and password. To be added as a user send an e-mail to econ@aset.psu.edu and request to be added to the system.. In your request specify whether you are faculty, staff, or grad student and your intended use of the workstations.

You will log into the workstations using SSH., a secure shell program that functions like telnet but with a higher level of security. This lets you set up a simple line-by-line interface with the workstation. SSH is installed in all department lab machines and can be installed on any faculty pc, on or off campus. (The SSH software can be downloaded from the university web site <https://downloads.its.psu.edu/> . After logging in choose "file transfer" and one of the software options is SSH.)

¹The latest version of these notes is available as a pdf file at <http://econ.la.psu.edu/~mroberts/sunnotes.pdf>.

Within SSH you will establish a profile containing the log in information for each of machines (grizzly, wildcat, beaver, kodiak, or bighorn).

Once logged in you will need to know basic UNIX commands to move between directories, save files, copy etc. The following Web sites have introductions to UNIX and summaries of the basic file commands:

<http://theory.uwinnipeg.ca/UNIXhelp/>
<http://cac.psu.edu/~leous/classes/utilities/>
<http://unixhelp.ed.ac.uk/>

Customizing your login files

When you are added as a user on the network you will be given a userid, a home directory called either /home/faculty/userid or /home/students/userid, and a couple of generic login files that will be placed in your home directory. These files, 2 of which are called .login and .cshrc, are run every time you logon and make certain that you have access to all the directories you will need to run the software packages. You can add additional customizing commands to your login sequence by editing these two files. Notice in these file that there are some clearly-marked sections that you should not alter. A number of possible customizing features are also discussed. If you corrupt your startup files you can start over with the initial generic files for a new user by replacing your files with the generic versions stored as /apps/skel/econ.login and /apps/skel/econ.cshrc. There is a README file that explains how to copy them.

Checking for system workload

Because the workstations are multiuser machines, their speed will depend upon how many users are logged on and how many jobs are running. After logging in use the “who” command to check how many users are logged on the machine and the “/usr/ucb/ps -au” command to check the number of jobs running and the amount of memory each is using. (A slightly less informative version of the ps command is available as “ps -ef”). Use both commands to check the workload on the machine. It is not necessary to be logged on to run jobs so a machine can be fully utilized with very few users. You can also use the “top” command which will give you the status of all jobs being processed. (To exit from the top command type CNTRL-c). If the machine you are on is busy try one of the others. The workstations with two processors (grizzly and wildcat) can each have two jobs running, one on each processor, without slowing them both down. Similarly, bighorn and beaver can run four simultaneous jobs without interference.

III. File Structure:

Home Directory

Every user has a home directory which they can use to store files. When you login you are automatically placed in your home directory which is /home/faculty/userid or /home/students/userid depending on whether you are faculty or student. This home directory is located on the central file server but is accessed when you log into any of the machines. You will always be put into your home directory regardless of which machine you use and most users will not need to be concerned about the exact location of the home directories. The default

specification for your home directory is that anyone will be able to read the files in your directory, but only you will be able to write files to the directory. Other users will be able to access or copy files from your directory, which makes it easy to share programs or data sets, but cannot store anything in your directory. If you do not like this arrangement you can change your home directory to be read and write protected using the UNIX “chmod” command. To make your home directory only accessible to yourself, for example, use the command “chmod 700 /home/faculty/userid” or “chmod 700 /home/student/userid” depending on where your home directory is located. You should be particularly careful to check the permissions assigned to any subdirectories that you set up within your home directory to make sure that they are not writeable by users who should not have access.

It is very important that you keep track of the amount of disk space you are using for file storage. This resource is shared by all users and excessive use by one person restricts what is available for all other users. Under no circumstances should your home directory exceed 500mb in size. The “du” command allows you to check how much storage space you are using. If you begin in your home directory, the command “du -sk /* | sort -n” will report the size (in kilobytes) of all your files and subdirectories sorted in order from smallest to largest. The command “du -sk” will report the total size of your home directory. If your storage requirements exceed 500mb contact the system administrative who will allocate you storage space in a different directory. Also see the discussion of compressing files in the last section of these notes.

DCE/DFS Home Directory

Students and faculty can also get storage space on the university’s distributed file system. This is convenient because this file system is linked from our department unix workstations to other unix systems around the campus, including our IBM SP nodes, and can be mapped as a separate drive on your desktop pc. This will allow you to store your programs and data in one place and access them from your pc or any of the unix workstations around campus. The name of your dfs home directory is based on your access account user id (not your department id). Suppose your access account id is abc123. Your dfs home directory will be /.../dce.psu.edu/fs/users/a/b/abc123. To access this directory from our department workstations you need to log into the dfs system by typing: dce_login abc123. You will be prompted for your access account password. You can then access your dfs directory as you would any other unix directory using the cd command, copy files to it, make subdirectories, etc. Setting up an alias for the full directory name will save you a lot of typing

Data Set Storage

Each of the computing workstations, has additional hard disk space which can be used to store data sets, programs, or outputs. This workspace on each machine is in the directory /animal/w0 (where “animal” is the machine name grizzly, wildcat, beaver, or bighorn). This workspace is specific to each machine but is linked automatically across machines. This allows you to be logged onto grizzly, for example, but read or write files on the other machines. Users that want to store data or programs in this space will be assigned a subdirectory with their userid. If your program either reads large data sets or writes a lot of output to a file then using this workspace will speed up the execution of programs. In particular, writing your program’s output to the harddisk on the machine your program is running on is substantially faster than writing directly to your home directory since that requires moving information over the network from one machine

to another.

Temporary Work Space

Each of the computing workstations also has temporary workspace in the directory /tmp. The /tmp directory on SUN machines is also the system swap space. Programs like Gauss, SAS, and elm (mail program) use it to store temporary files and the system uses it as paging space. Unfortunately /tmp is a finite resource. If too many programs use the space at once, others may not run or processes that are currently running may crash. Since this disk space is shared by all users it is going to fill up and when it does files will be deleted by the system administrator. It is not a secure place to save files you want to keep. Rather, when your work is finished move the files you want to save to your home or data directory and delete the stuff you don't need to keep. Several programs, particularly SAS and GAUSS, will leave files behind in the tmp directory if they end because of an error. Each user needs to check this directory when their session is ending and delete any unnecessary files that belong to them. See the discussion below for how individual packages use the /tmp directory.

IV. Unix Editors

All are available in the standard path. Type the name of the editor, followed by the file to edit.

vi - this is a standard line editor. For some help in learning the basic vi commands (you will need it) see an introductory Unix book or the Web sites listed above.

Pico, emacs, nroff, and troff are other editors that are available. See a Unix book or the Web sites listed above.

Alternatively, you can use an editor on the pc and then move the file to the workstations using the File Transfer option in SSH. One nice editor for the pc is available free from <http://www.crimsoneditor.com/>. This editor recognizes the syntax of many programs, including matlab and stata, and uses color to distinguish commands, comments, if statements, etc. It is handy if you want to write your programs on a pc and run them on the workstation because it helps you avoid coding errors.

V. Statistical Software

Most users on the workstations will be using one of the statistical packages, MATLAB, GAUSS, STATA, SAS, or programming languages, Fortran and C. All of these packages can be used in **BATCH MODE** or **INTERACTIVE MODE**. Batch mode should be used for all programs that require substantial time to execute (more than 30 minutes). With batch mode you submit the program and it will run in the background and you can log off the workstation. Shorter runs or sessions where you are debugging programs can be run in interactive mode. This requires you to stay logged into the machine. **You should not run long programs in interactive mode, particularly from the PC's in the department because this ties up the PC and makes it impossible for others to use it.**

For each of the software packages the following pages describe how to submit jobs in both

interactive and batch mode.

1. SAS (version 8)

When using an SSH connection, SAS can only be used in batch mode. To run a SAS program in batch mode, first store your program in a file that uses the 3letter qualifier .sas. For example, filename.sas. At the command line enter: `sas filename.sas &` (The & puts the run in background and frees your terminal so you can do other things). If your program file had .sas at the end then SAS will create log and output files named filename.log and filename.out. If your program did not end with .sas then SAS will simply append .log and .out to whatever the file name was and use these files to store the log and output results. When using SAS in batch mode you do not need to include RUN as the last line of the program.

SAS is one of those programs that uses the directory /tmp on each machine to store its temporary files. Usually it clears those files when exited correctly. However, if it fails to exit properly (due to a power failure or the process being terminated with 'kill') it leaves behind a directory with a name like /tmp/SAS_worka000006A8. The user whose run created the directory must go in and delete it to free up the space in /tmp. To check if you have left anything behind follow these steps:

1. Switch to the /tmp directory. Unix command is: `cd /tmp`
2. List the files and directories in temp. The unix command is: `ls -l`
If SAS has left directories behind you will see one or more lines in the listing like:
`drwx----- 2 <username> other 109 May 18 1242 SAS_worka000006A8`
3. If one of these SAS directories is owned by you, delete it. The unix command to delete this particular directory is: `rm -rf SAS_worka000006A8`

2. STATA (version 9.1)

To use STATA in an interactive session type “stata” . You can either enter commands at this point and stata will execute them one at a time or call in an existing program file to run. If the program is stored in the file prog.do you can execute it by typing “do prog.do”. In order to have a log of your run you will need to start a log file at the beginning of your session or in the beginning of your prog file. Use the command “log using prog.log” to create an ASCII output file called prog.log.

To run a batch program in STATA use the command “stata -b do prog.do & “ where prog.do is the name of your program file. This command will run your program in background (that’s what the & does), create a log file called prog.log for your run, and output everything to that file. It is important to use this method of running batch programs if you use the #delimit command in your program. The #delimit command lets you set the character you want to use at the end of each command in your program file (such as the semicolon). The alternative way to run a batch program is to use “stata <prog.do> prog.log”. If you use this method the #delimit command will not work in your program so you are stuck using the carriage return to end each command in your program. When running batch programs you can change the amount of memory allocated with the -k option when you call stata.

One general point to keep in mind when using STATA is that it is important to allocate sufficient memory to your program so that all your data can be held in memory. If you do not allocate enough stata will use the hard drive as additional RAM but this will slow down execution of the program substantially. In order to allocate more memory to stata when you start an interactive session use the -k option. For example, type “stata -k50000” if you want to allocate 50mb of memory to the program. When running batch programs you also change the amount of memory allocated with the -k option when you call stata.

3. GAUSS (version 7.0)

Our current GAUSS license is a floating license which allows GAUSS to be run on any of the machines in our network but limits us to eight simultaneous users across all machines. If you attempt to use gauss when there are 8 active users you will not be allowed in and will have to wait until one of the users ends their gauss session. In addition to the basic GAUSS program we have five of the application modules - optmum, maxlik, conoptmum, fanpac and count. If you use one of the applications modules you must enter their names in your program in lower case letters. If you use upper case letters to refer to them you will get an error message that the application cannot be found.

To use GAUSS in an interactive session, type “gauss -v” to start an interactive session.

To run GAUSS in batch mode type “gauss -B prog.gau &” where prog.gau is the file containing the gauss program. Make sure you use a capital B in the option. As with the other software packages, the & puts the run in the background and returns control of the keyboard to you.

When you run gauss the default memory size is 50MB. To alter this you need to create your own configuration file. To do this copy the file /apps/gauss/gauss.cfg to your home directory. About halfway through this file is a line workspace=50 that sets the mb of workspace. Using an editor change this number and save the file (if you set it larger than the free space in the /tmp directory you will get an error message and gauss will not start). You then need to tell gauss to use the configuration file in your home directory, rather than the default one, when it starts up. To do this you edit a line in your .login file. The line is “setenv GAUSS_CFG /apps/gauss” and you need to replace the /apps/gauss path name with the path of your home directory (/home/students/userid or /home/faculty/userid).

When a gauss program ends due to an error it will not free up the workspace that was allocated to the program when it started. You need to go to the /tmp directory, do a long listing (Unix command “ls -l”), and delete any files that belong to you (Unix command “rm filename”). The files that gauss creates will have names like wksp_AAAA12345. If you do not delete these files the /tmp directory will become full and this will prevent anyone (including you) from running a gauss program.

There are two error messages that commonly occur when calling gauss. The first error message is that there is insufficient workspace to allocate for your program. This occurs because the amount of workspace requested in the gauss.cfg file (50 mb is the default) is larger than the available space in the /tmp directory. This can occur when there are many users on the machine or users fail to clean up after themselves. One option is to modify the gauss.cfg file to request less space (see the discussion above). A second option is to use a different machine which has

sufficient space in the /tmp directory. A third option is to check where the /tmp space is allocated and ask the file owners to delete unnecessary files. A second error is that the gauss license cannot be located. This happens if the central file server is shutdown, either accidentally or for planned maintenance. Send an email to econ@aset.psu.edu and ask them to restart the gauss license server.

4. FORTRAN

The workstations have Fortran77 and Fortran90, the IMSL sub-routines are not available anymore.

You compile a fortran program called myprog.f by going to the directory containing the program and typing: f77 myprog.f for the Fortran77 compiler and f90 myprog.f for the Fortran 90 compiler.

This will create an executable element called a.out which will be stored in the same directory. You can then run the executable by typing ./a.out

If you would like to redirect any errors generated when you compile the program to an output file add the following at the end of the compiler command >& outfilename. For example, to compile the program above and send the compilation errors to an output file called cerror use the command: f77 myprog.f >& cerror

If you prefer a different name for the executable element use the -o option when compiling: f77 myprog.f -o myprog.exe. This will compile the program myprog.f and create the executable element named myprog.exe. Whatever name appears immediately after the -o will be used for the executable. You can run it by typing ./myprog.exe

5 S-PLUS (version 5.0) - To access it type: /apps/bin/Splus5 . You must use the upper and lower case letters as given. This is all the information we have for now.

6. MATLAB (version 7.1).

We have the OPTIMIZATION, STATISTICS, SYMBOLIC MATH, and GENETIC ALGORITHMS toolboxes available. To check the version of the toolbox we have type "vers" from the command line inside matlab. Matlab can be run interactively or in batch mode.

Program Files: These are ascii files defined by the extension ".m", for example myprog.m. Any line that begins with a % is treated as a comment card and ignored in the program execution.

Interactive Programs: To begin an interactive session type "/usr/ladmin/bin/matlab" to end one type "exit". You can execute a program named myprog.m by just typing "myprog". If you want to create an output file called result.txt add the command "diary result.txt" at the beginning of the program and output will be directed there. To turn off the output file add the line "diary off" at the end of the program. If you use the same file name (e.g. result.txt) repeatedly during a session, Matlab will add the results from the latest run at the end of the file rather than erasing it and beginning with a new version. If you are running matlab interactively there are a few tricks that will save you time. Adding ; at the end of a command line will prevent the results of the command from being sent to the terminal screen and this can speed up execution time

substantially. The up and down arrow keys are shortcut keys. The up key repeats the last command and will cycle through the past commands as you press it. If you type a letter e.g. “d” and the up arrow it will find the latest command beginning with that letter.

Batch Programs: To run the program myprog.m in the background (if you have specified an output file internally in the program) type:

```
/usr/ladmin/bin/matlab myprog.m &
```

If you have not specified an output file internally in the program and want to output any results to result.txt then type:

```
/usr/ladmin/bin/matlab myprog.m result.txt &
```

The & puts the run in the background and returns the keyboard to your control. As with any process, you can use the “ps” command to check the execution status of the job.

Matlab Files: Matlab will search for data files or program files in the local directory it is running from and in directories which contain the matlab toolboxes. To use other directories you need to add the path names to the program. For example if you are running from a directory /home/students/jsmith/matlabprog but your data is stored in /home/students/jsmith/matlabdata you add the command “addpath /home/students/jsmith/matlabdata” to your program file.

Data Files: The default extension for matlab generated data sets is “.mat”. These datafiles can store several different types of data (variables and matrices). To load a data set called “mydata.mat” use the command “load mydata”. To see the contents of the data set type “whos” after loading the data set. If you have an ascii file called mydata.asc load it with the command “load mydata.asc -ascii”.

7. MATHEMATICA. This software is currently being installed and tested. Contact econ@aset.psu.edu with questions about its status.

VI. System Etiquette

- 1 Running jobs in the background and the NICE command.

When you have long runs they should be submitted to run in the background, otherwise they will tie up the terminal until completed. You do this by including an & at the end of the run command, for example, stata -b do myprog.do &. If you are using the C shell, which is the default on our system, the & will allow you to logoff the computer but your program will continue to run in the background. (With other shells use “nohup command &” to run jobs in the background and keep them from being terminated when you logoff).

If your runs are very long they should be submitted with a lower priority by using the NICE command. You add the expression “nice +n” to the start of your command where n is a number from 1 to 20. For example, “nice +10 stata -b do prog.do &” will run a stata program in background with a priority of +10. The command “nice +20 gauss -B prog.gau &” will run a gauss program in background with a priority of +20. The higher the number after the plus sign

the lower the priority of the job. Not using the nice command is equivalent to a priority of 0. If you use the nice command and your program is the only job running on the machine you will get the full cpu and the program will run in the same amount of time as without the nice command. When there are other jobs running your job will get a lower priority and receive a smaller fraction of the cpu time than the other jobs. Some rough tests that I did showed that, when two jobs were running without the nice command, each received approximately 50% of the cpu time. Submitting one of the jobs with a priority of +4 reduced its share of cpu time to approximately 35%. A priority of +10 reduced its share to about 25% and a priority of +20 reduced it to 15%. Any job which is going to take more than 2 hours to complete should be submitted with a priority of at least +10 so that users with smaller jobs will be able to get quicker turnaround during the day.

2. Compressing files.

If you are storing files that you do not need for a while you should compress them to save disk space. Specify "compress filename" to compress a file. The compressed version will have the same name as the original file with a .Z at the end. Most programs will not be able to use a data file that has been compressed. To return it to its original size specify "uncompress filename.Z" and the process is reversed. There are other commands that can be used to compress files. "gzip" and "gunzip" can be used in tandem or "pack" and "unpack".

3. Priorities for the different SUN workstations.

The newest machine, kodiak, is meant for large programs that are submitted to run in the background and not for interactive sessions. The other machines are available for interactive sessions, debugging programs, and smaller jobs.

4. Killing runaway jobs and cleaning up after yourself

Every process (SSH session, unix command, running program, open editor, print job etc.) on a unix machine is assigned a process id number (pid). To see the pid number and the user responsible for every process running on a machine use the command: ps -ef |more. If you have a background job which is not performing properly you can kill it with the unix command: kill -9 pid, where pid is the number for the particular process (SAS job, gauss program etc.). You should also check if you have processes left running from past sessions (telnet sessions that were not ended properly are a common one) and kill those as well.

5. If you have questions or problems please send an email to econ@aset.psu.edu.