February 2005

AN INTRODUCTION TO UNIX
Wharton Research Data Services

WRDS' UNIX server, ***wrds.wharton.upenn.edu***, is a SUN Ultra Enterprise 15000 system running Solaris 9. The principal applications supported are SAS, FORTRAN (with IMSL subroutines) and C.
Remember that UNIX is case-sensitive; a lower case letter is not the same character as its upper case counterpart. In general, UNIX commands should be entered entirely in lower case, but there are a few exceptions.

Login to the computer

To login, direct your **ssh** client to **wrds.wharton.upenn.edu**. After you connect to the server, you will be prompted for your username and password. (Note: **WRDS** no longer supports **telnet** due to security issues).

Logout

Logout by entering <**Ctrl-d**>, or simply click the **exit** icon.

On-line documentation

UNIX computers have on-line documentation that is accessed by invoking the "**man**" (short for manual) command:

man *command_name* = look up "*command_name*" in the on-line manual.

For example, to get more information on the "**passwd**" command, enter: **man passwd**

You can also search for information based on a keyword:

man -k *keyword* = search for commands that contain the word "keyword" in their descriptions.

File specification / Directory structure

“Everything in the UNIX system is a file.”[1] UNIX filenames may be comprised of up to 256 characters, and the beginner is advised to use only letters, numbers, the period, the hyphen and the underscore when naming a file. For example, the following are all valid, and *different,* UNIX filenames:

myfile Myfile MYFILE my-file myfile

Remember that “myfile”, “Myfile” and “MYFILE” are three different filenames, representing three distinct files. Also, all filenames that begin with a “.” are known as “hidden files”. For example, the files **.profile** and **.login** are hidden files.

Files are organized into groups called directories. Each directory may in turn have subgroups within it, called subdirectories. The resulting structure of directories, subdirectories and other files is often likened to a tree, with subdirectories “branching out” from their parent directories. All files and subdirectories are members of the group known as the "root directory."

The UNIX root directory is named “/”. The “/” character is also used to separate directory names from subdirectory names, and directory names from filenames. For example, if the file “myfile” is contained in a subdirectory of the root directory named “subdir1”, its complete file specification would be:

```
wrds(~)% cd subdir1/myfile
```

You will be prompt:

```
wrds(~/subdir1)%
```

When you login to the computer you will be in your *home directory*, also known as your *login directory*. The directory you are currently in is referred to as the *current* or *working directory*. If your username were “smith”, your home directory would be:

/home/school/smith

Therefore, the file named “myfile” located in Smith's home directory is completely specified by:

/home/school/smith/myfile

If Smith is in his/her home directory, the file could simply be referred to as **myfile**, but if Smith is in the directory **/home** , the file would be referenced as **smith/myfile** . (Note that

[1] Kernighan, Brian W. and Rob Pike, The UNIX Programming Environment, Prentice-Hall, 1984, pg. 41.

there is no leading "/" in this specification.) If Smith is in the root directory, it would be referred to as:

home/school/smith/myfile.

Finally, no matter which directory is current for user Smith, the file can always be referenced as **home/school/smith/myfile**. The complete file specification may always be used to unambiguously reference a file, no matter where you are in the directory structure.

Managing Directories

(a) Identify which directory is "*working*"

To find out which directory is "working", use the **pwd** command.

```
wrds(~)% pwd              (print current directory)
/home/school/smith        (output)
```

(b) Change the "current" directory

The command that allows you to move around within the directory structure (i.e., change the "current" or "working directory") is **cd**.

cd *dir_path_name* = change directory to "*dir_path_name*"

```
wrds(~)% cd /wrds/compustat     (go to subdirectory compustat in wrds)
wrds(~/wrds/compustat)%         (output)
```

If the **cd** command is used without an argument (i.e., without specifying the "*dir_path_name*", it will make your home directory the current directory. Therefore, to return to your home directory, simply enter:

```
wrds(~/wrds/compustat)% cd      (to return to home directory)
wrds(~)%                        (output)
```

To make a parent directory the current directory - that is, to "back up" the directory tree one level - enter: **cd ..**

```
wrds(~/wrds/compustat)% cd ..   (to "back up" one level)
wrds(~/wrds)%                   (output)
```

Notice that there is a space between the "d" and the first "." in the above command - it is necessary in UNIX (unlike DOS).

(c) List file and directory names

To list the names of files and subdirectories that are contained in the current directory, enter **ls** command:

```
wrds(~/wrds/compustat)% ls          (list filenames in compustat directory)
samples/ sasdata/ seqdata/          (output)
```

To list the names of files and subdirectories that are contained in the current directory, along with additional descriptive information, type: **ls -l**

The "**ls**" command will not necessarily list all of the filenames in your home directory - some files may remain hidden unless you use the "**-a**" switch with the "**ls**" command as follows:

```
wrds(~)% ls -a       (list all files and subdirectories in the current directory)
```

More than one option at a time may be used on the command line. For example, to get a "long listing" of all files in the current directory, including the hidden files, enter:

```
wrds(~)% ls -al      (long listing of all file and directory names)
```

The **ls** command may be used to list the files in other directories (the same applies to "**ls -l**" and the "**ls -al**"):

```
wrds(~/wrds/compustat)% ls /wrds/crsp   (list files in crsp directory)
samples/ sasdata/ seqdata/              (output)
```

For example, if the directory "/home" were the current directory of user "smith", a listing of the files in the directory "/home/smith/data" may be obtained by entering:

```
wrds(~)% ls smith/data
```

(d) Make a new directory

To create a new directory which is a subdirectory of the current directory, enter:

wrds(~)% mkdir *dir_name* (make a new directory named "*dir_name*")

(e) Remove (delete) a directory

To remove an empty directory (i.e., a directory that contains no files), enter the following command from the parent directory of the subdirectory you want removed:

wrds(~)% rmdir *dir_name* (remove the empty directory "*dir_name*")

To remove a directory that is not empty - i.e., a directory that contains files or other subdirectories - use the "**-R**" switch with the **rm** command:

> **wrds(~)% rm -R *dir_name*** (remove (erase) the directory "dir_name,")

This command and option deletes the entire directory structure beneath "dir_name".

(f) Rename a directory

To rename a directory, use the **mv** command (short for "move"):

> **wrds(~)% mv *oldname_dir newname_dir*** (change name of directory "*oldname_dir*" to "*newname_dir*")

Managing files

(a) Copy a file

To copy a file (that is, to make another copy of a file, as opposed to renaming it) use the **cp** command:

> **wrds(~)% cp *myfile new_file*** (make a copy of file "*myfile*", named "*new_file*", in the current directory)
> **wrds(~)% cp *myfile dest_dir*** (copy the file "*myfile*" to the directory "*dest_dir*")

For example, to copy a sample program from compustat/samples subdirectory to your home directory, use the following syntax:

> **wrds(/wrds/compustat/samples)% cp ina.sas ~**

or

> **wrds(/wrds/compustat/samples)% cp ina.sas /home/school/smith/ina.sas**

In UNIX, as in many other operating systems, the lone character "**.**" is used to represent the current directory, while "**..**" represents the parent directory.

> **wrds(~)% cp *myfile* .** (copy the file "*myfile*" to the current directory)

(b) Move or rename a file

The "**mv**" command (short for "move") is used to either relocate a file to a different directory or to rename a file. Note that, unlike the "**cp**" command, the "**mv**" command will not create another copy of a file. To move a file to a different directory:

wrds(~)% mv -i *myfile dest_dir* (move the file "*myfile*" to the directory "*dest_dir*")

The "**-I**" switch will cause you to be prompted if you are about to overwrite an existing file of the same name in the destination directory. Also, "**.**" may be used in the "*dest_dir*" field to represent the current directory.

To rename a file:

wrds(~)% mv -i *filename newfile* (rename the file "*filename*" to "*newfile*")

(c) Delete, or remove a file

To delete, or remove a file:

wrds(~)% rm *filename* (remove the file "*filename*")

(d) Display a file's contents on your screen

To display the contents of a text file on your monitor's screen, use the "**more**" command:

wrds(~)% more *myfile* (display the contents of the file "*myfile*")

The "**more"** command will display the contents of a text file on your screen one page at a time. The "**more**" command will not attempt to send the contents of a non-text file to your screen. If you want to look at only the last few lines of a large file, the following command is useful:

wrds(~)% tail -10 *myfile* (display the last 10 lines "*myfile*")

Of course, you can change the parameter "10" in the above command to view as many lines as you wish (up to a maximum - check the manual pages for details).

(e) File comparison

If you want to compare two files to see if they are identical, and if they are not, display their differences, enter:

wrds(~)% diff *filespec1 filespec2*

If the output of the "diff" command is null, i.e., nothing is displayed on your screen, then the files are identical.

(f) Find a file

The "**find**" command is useful for locating files in a labyrinthine directory structure. An example of its use:

```
wrds(~)% find dir_name -name myfile
```

The above command will search the directory "*dir_name*" (e.g., "/home/smith") and all of its subdirectories for the file named "*myfile*", and if it is found it will display the complete file specification on your screen.
To search for all files in the directory "/home/smith" with filenames beginning with the letter "t", you can use the following syntax:

```
wrds(~)% find /home/smith -name 't*'
```

The asterisk is referred to as a wild card (discussed below.) Note that **t*** is enclosed in single (' ') quotes.

Extensions and Utilities

(a) Wildcards

You can refer to a set of files in UNIX by using wildcards; the ***** character represents any sequence of characters and the **?** character represents any single character. To list all files in the current directory beginning with the letter "f", you could enter the command:

```
wrds(~)% ls f*
```

To list all files with four-letter filenames ending in "f":

```
wrds(~)% ls ???f
```

To list all files that have filenames beginning with "s" and ending in "t":

```
wrds(~)% ls s*t
```

(b) Concatenate files

To concatenate two files and put the result in a third file use the "**cat**" command, which stands for "concatenate".

```
wrds(~)% cat file1 file2 > file3
```
(concatenate "*file1*" and "*file2*"; put result in "*file3*")

(c) Pipes

If a command sends more than one screen of output to your terminal, you can view it one page at a time by “piping” the command into the “**more”** command. For example:

wrds(~)% ls -l | more

The “|” symbol is called a “pipe”. The command to the left of a pipe must produce standard output and the command to the right of a pipe must accept standard input. A command that may have pipes on its left and right must do both, and is known as a “filter”. Several commands can be strung together on the command line by using pipes.

(d) Search file contents for a pattern

To determine if a text file contains a certain character string that you are looking for, use the “**grep**” command. The “**grep”** command will print out (to the standard output) any lines in the indicated file which contain the character string (or “regular expression”) you are looking for. The syntax of the “**grep**” command is:

wrds(~)% grep *pattern myfile* (search the file “*myfile*” and print out any lines containing the character string *pattern*)

The “**grep**” command accepts wildcards in the file field; this is useful for searching entire directories for a file containing a certain character string. For example:

wrds(~)% grep alias * (search all files in home directory for the pattern “alias”)

(e) Disk space quotas

You will have a disk space quota on the WRDS server. To see what your quota is, and how much of it you have used, enter:

wrds(~)% quota -v (display disk space statistics (in K-byte units)

(f) Command line history

To recall a previously executed command, press the up-arrow key. Continuing to hit the up-arrow key will recall more of the commands you recently executed (beginning with the most recent and progressing backward). The down-arrow key displays the commands in the opposite order. When you find the command you want to execute, simply hit the <Enter> key.

(g) Viewing system activity (“load”)

To find out how busy the system is, use the **top** command. This utility will display system resource utilization. Hit the “q” key to exit the **top** utility.

Managing processes

(a) Listing processes

A program running on a UNIX system is referred to as a *process*. All of the commands described above, when executed, will spawn a new process on the workstation. A process is uniquely identified by its *process I.D.* number (PID), which is assigned by the operating system when it is spawned. To list all processes that are currently running on the workstation, enter:

```
wrds(~)% ps -ef        (list every process running, in full format)
```

(You may want to pipe the output of the above command into the "**more**" command). The most important information gained from the output of the "**ps –ef**" command is contained in the first two and last two fields - UID, PID, TIME and COMMAND. UID is the user ID, or name of the owner of the process, PID is the process I.D. number, TIME is how long the process has been running, and COMMAND is the name and syntax of the command that spawned the process. To list only the processes that are yours, enter:

```
wrds(~)% ps -ef | grep username    (list all processes of user "username" in full
                                   format)
```

(b) Background processes

It is possible to run a process in the "background", so that you can enter other commands while the first process is running (i.e., the command prompt will be returned to you). A process that is running in the background is often referred to as a "job". To run a process in the background, simply append an ampersand (&) to the end of the command line:

```
wrds(~)% sas ina.sas &        (the file ina.sas will be executed in the background)
```

A job run in the background will terminate when you logoff the computer unless you instruct it not to. To run a job in the background, and also be able to logoff without terminating it, submit it as follows:

```
wrds(~)% nohup jobname &           (submit job "jobname" in the background
                                   and don't terminate ("hangup") upon logout)
```

(c) Terminating processes

Foreground processes are terminated by issuing an "interrupt": **<Ctrl-c>**
Background processes can be terminated, or "killed" as follows:

```
wrds(~)% ps –fu your_user _name
wrds(~)% kill your_process_PID    (terminate process identified by the PID)
```