**Computer Lab 1 – Intro to Computing**

**Conservation Genetics (BIOL 4174 / 5174)**

**Part I – Introduction to the Command Line**

In this lab you will learn the basics of navigating through the directory structure of a computer running Mac OS or Linux. You will also learn some basic helpful commands for viewing, searching, and modifying files from the command line. These skills are increasingly important as modern genetic studies operate upon increasingly large datasets. In order to get as much performance as possible from computers, researchers in this field often sacrifice the luxury of a graphical operating system in favor of the command line. The material you learn during this week will be utilized in nearly every Conservation Genetics lab this semester.

Below you will find a list of commands that you will be required to know. A secondary list of helpful commands is also provided.

**Required Commands:**

pwd

mkdir

cd

cp

mv

rm

ls

tar

chmod

exit

**Useful (but not required) Commands:**

which

cat

head

tail

less

clear

man

date

history

The following are explanations of commands with some examples of how they are used. We will go through today’s lab as a group, and most of the remainder of this document can serve as a reference.

**cat: Display or concatenate files**

**cat** takes a copy of a file and sends it to the standard output (i.e. to be displayed on your terminal, unless redirected elsewhere), so it is generally used either to read files, or to string together copies of several files, writing the output to a new file.

**cat** ex

displays the contents of the file ex.

**cat** ex1 ex2 **>** newex

creates a new file newex containing copies of ex1 and ex2, with the contents of ex2 following the contents of ex1.

**cd - change directory**

**cd** is used to change from one directory to another.

**cd** dir1

changes directory so that dir1 is your new current directory. dir1 may be either the full pathname of the directory, or its pathname relative to the current directory.

**cd**

changes directory to your home directory.

**cd ..**

moves to the parent directory of your current directory.

**chmod - change the permissions on a file or directory**

**chmod** alters the permissions on files and directories using either symbolic or octal numeric codes. The symbolic codes are given here:-

**u** user **+** to add a permission **r** read

**g** group **-** to remove a permission **w** write

**o** other **=** to assign a permission explicitly **x** execute (for files),

access (for directories)

The following examples illustrate how these codes are used.

**chmod u=rw** file1

sets the permissions on the file file1 to give the user read and write permission on file1. No other permissions are altered.

**chmod u+x,g+w,o-r** file1

alters the permissions on the file file1 to give the user execute permission on file1, to give members of the user's group write permission on the file, and prevent any users not in this group from reading it.

**chmod u+w,go-x** dir1

gives the user write permission in the directory dir1, and prevents all other users having access to that directory (by using **cd**. They can still list its contents using **ls**.)

**clear - clear old output from your terminal window**

**cp - copy a file**

The command **cp** is used to make copies of files and directories.

**cp** file1 file2

copies the contents of the file file1 into a new file called file2. **cp** cannot copy a file onto itself.

**cp** file3 file4 dir1

creates copies of file3 and file4 (with the same names), within the directory dir1. dir1 must already exist for the copying to succeed.

**cp -r** dir2 dir3

recursively copies the directory dir2, together with its contents and subdirectories, to the directory dir3. If dir3 does not already exist, it is created by **cp**, and the contents and subdirectories of dir2 are recreated within it. If dir3 does exist, a subdirectory called dir2 is created within it, containing a copy of all the contents of the original dir2.

**date - display the current date and time**

**date** returns information on the current date and time in the format shown below:-  
Wed Jan 30 11:27:50 GMT 2008

It is possible to alter the format of the output from date. For example, using the command line  
**date '+The date is %d/%m/%y, and the time is %H:%M:%S.'**

at exactly 11.30am on 30th January 2008, would produce the output  
The date is 30/01/08, and the time is 11:30:00.

**exit - close your current terminal session**

**head and tail - display the first or last ten lines of a file**

**head** file1

prints the first ten lines of file1. The number of lines to be printed can be modified, such as the following command:

**head** -5 file1

which will print the first five lines of file1. The tail command has similar behavior.

**tail** -5 file1

will print the last 5 lines of a file.

**history - display previously used commands**

**less - display contents of a text file in your terminal window**

**less** file1

prints a file to the screen in a way that is easy to scroll through using the arrow keys on the keyboard. By default, this this program wraps long lines which may make some files difficult to read. The command:

**less** -S file1

will display a file without wrapping long lines. This makes many files easier to navigate and read.

**ls - list names of files in a directory**

**ls** lists the contents of a directory, and can be used to obtain information on the files and directories within it.

**ls** dir1

lists the names of the files and directories in the directory dir1, (excluding files whose names begin with . ). If no directory is named, **ls** lists the contents of the current directory.

**ls** -R dir1

also lists the contents of any subdirectories dir1 contains.

**ls** -a dir1

will list the contents of dir1, (including files whose names begin with . ).

**ls** -l file1

gives details of the access permissions for the file file1, its size in kbytes, and the time it was last altered.

**ls** -l dir1

gives such information on the contents of the directory dir1. To obtain the information on dir1 itself, rather than its contents, use

**ls** -ld dir1

**man - display an on-line manual page**

**man** displays on-line reference manual pages.

**man** command1

will display the manual page for command1, e.g **man** cp, **man** man.

**man -k** keyword

lists the manual page subjects that have keyword in their headings. This is useful if you do not yet know the name of a command you are seeking information about.

**mkdir - make a directory**

**mkdir** is used to create new directories. In order to do this you must have write permission in the parent directory of the new directory.

**mkdir** newdir

will make a new directory called newdir.

**mkdir -p** can be used to create a new directory, together with any parent directories required.

**mkdir -p** dir1/dir2/newdir

will create newdir and its parent directories dir1 and dir2, if these do not already exist.

**mv - move or rename files or directories**

**mv** is used to change the name of files or directories, or to move them into other directories.

**mv** file1 file2

changes the name of a file from file1 to file2 unless dir2 already exists, in which case dir1 will be moved into dir2.

**mv** dir1 dir2

changes the name of a directory from dir1 to dir2.

**mv** file1 file2 dir3

moves the files file1 and file2 into the directory dir3.

**pwd - display the name of your current directory**

The command **pwd** gives the full pathname of your current directory.

**rm - remove files or directories**

**rm** is used to remove files. In order to remove a file you must have write permission in its directory, but it is not necessary to have read or write permission on the file itself.

**rm** file1

will delete the file file1. If you use

**rm -i** file1

instead, you will be asked if you wish to delete file1, and the file will not be deleted unless you answer **y**. This is a useful safety check when deleting lots of files.

**rm -r** dir1

recursively deletes the contents of dir1, its subdirectories, and dir1 itself, and should be used with suitable caution.

**tar - create and use archives of files**

**tar** can be used to create and manage an archive of a set of files.

**tar h**

Displayes options for using tar. The **h** option stands for “help”.

**tar cf** archive1.tar

creates an archive file called archive1.tar containing the contents of the current directory (and any subdirectories it contains). The **c** option stands for "create" and the **f** for "filename".

**tar cf** archive2.tar mydir

creates an archive file called archive2.tar containing the contents of the directory mydir.

**tar tvf** archive1.tar

lists the contents of the archive file archive1.tar. The **t** stands for "list" and the **v** for "verbose listing".

**tar xf** archive1.tar

extracts the contents of archive1.tar and copy them into the current directory. The **x** stands for "extract".

**tar xzf** archive1.tar.gz

Does the same as the last command, but the “z” tells tar that the zipped file is also compressed using gzip/bzip2/xz or lzma format

**which - report the full path to an installed program**

**Homework Assignment**

Spend approximately 20 minutes using the terminal window and exploring the commands you have learned in class. You should use each of the “required” commands at least twice (except for the exit command) and each of the “useful” commands at least once. Try using different options to see how these influence the behavior of the commands.

When you are finished, export your history using the following command, where <username> is replaced with your UARK username. Hint: check which directory you are in before executing the following command. You may want to switch to your Desktop/ directory prior to exporting your history so that the output file is easy to find

* history > username\_hist.txt

Submit your history file to the SafeAssign link in the Lab 1 folder on Blackboard.