QUICK GUIDE TO UNIX COMMANDS

Command	Purpose	Useful Flags	Example	Action
cd	change working direc-tory		1. cd /usr/homedir/proj1 2. cd .. 3. cd	1. makes proj1 the current work directory 2. moves up one level in the directory structure 3. make your homedir the current work directory
pwd	print path to current work directory		1. pwd	1. displays the absolute path to the current working directory
ls	list the contents of a directory	-l -R -a	1. ls 2. ls proj 3. ls -l 4. ls -R 5. ls -a 6. ls -lR	1. displays a short listing of the contents of the current directory 2. displays a short listing of the contents of the proj directory 3. displays a long listing showing protection, size, dates, etc. 4. recursively lists the contents of the current directory and all subdirectories 5. lists all files including hidden configuration (dot) files 6. displays a long listing recursively
mkdir	create a directory		1. mkdir sub1	1. creates subdirectory called sub1 below the current working directory
rmdir	remove directory		1. rmdir sub1	1. removes a subdirectory called sub1 from the current directory. The directory must be empty or the operation will fail.
cp	copy file	-r	1. cp f1 f2 2. cp -r proj1 proj2	1. copies the contents of file f1 to f2, if f2 exists it is overwritten 2. copy the proj1 directory and its contents to a directory named proj2
mv	rename (move) file		1. mv f1 f2	1. renames a file (or directory) f1 as f2, if f2 exists it is overwritten
rm	remove file	-r -f	1. rm f1 2. rm -r sub1 3. rm -f f1 4. rm -rf sub1	1. removes file f1, prompts the user for confirmation 2. removes the directory sub1 and all its subdirectories and files with confirmation 3. remove the file f1 without confirmation 4. removes the directory sub1 and all its subdirectories and files without confirmation
cat	concatentate files		1. cat f1 2. cat f1 f2	1. lists the contents of file f1 on the screen 2. lists files f1 and f2 on the screen
more	list files		1. more f1	1. lists the contents of the file one screen at a time and prompts for continuation
man	access a manual page	-k	1. man ls 2. man -k C++	1. displays the manual page for the ls command 2. finds all manual pages that refer to C++ in their NAME section

Command	Purpose	Useful Flags	Example	Action
g++	invoke the GNU C++ compiler/linker	-g -o -c	1. g++ file.cpp 2. g++ -o file file.cpp 3. g++ -c file.cpp 4. g++ -g file.cpp 5. g++ -g -o f f1.cpp f2.cpp	1. compiles and links file.cpp (file.cpp must contain a main) and produces an execuatble file named a.out 2. compiles and links file.cpp (file.cpp must contain a main) and produces an execuatble file named file (-o indicates the executable file name) 3. compiles and does not link file.cpp (file.cpp may contain a main) and produces an object file named file.o 4. compiles and links file.cpp (file.cpp must contain a main) and produces an execuatble file named a.out that contain symbol table information for a debugger 5. compiles and links the files f1.cpp and f2.cpp (one of the file must contain a main) and produces an execuatble file named f along with debugger information

The output of most Unix commands can be piped to other commands using the pipe operation (|). For example, **ls -l | more**, will perform a long directory listing and the **more** command will display the listing one screen at a time. In addition, any command that takes input from the keyboard or puts output to the screen can receive input or send output to a file using the redirection operators (< and >). For example, to capture the output of ls in a file named filename, type: **ls > filename**. If you have a C++ executable program called **hw1** that takes input from the keyboard, you can use redirection to make hw1 take input from a file. For example, to make hw1 take input from file the file **hw1.dat** use the command: **hw1 < hw1.dat.**