List of Unix Commands:

Note: Some of these commands may have extensions/options. Use the man command to read more about them.

Commands	**What it does**
bg job	Puts job in the background.
cat file1	Displays the content of file1.
cd dir	Changes directory from the current working directory to dir.
chmod (option) file1	Lets you change the read, write, and execute permissions on file1.
clear	Clears the terminal.
cp file1 file2	Creates a copy of file1 called file2.
cp file1 dir1	Copies file1 into dir1.
cp -r dir1 dir2	Copies dir1 into dir2
echo string	Prints string on the terminal.
fg job	Puts job in the foreground.
grep pattern file1	Searches file1 for pattern, and displays all lines that contain pattern.
head (option) (num) file1	Prints the first 10 lines of file1, by default. If you want to set the number of lines this prints, you need to do: head -n num file1 and it will print the first num lines of file1. E.g. head file1 will print the first 10 lines of file1. E.g. head -n 3 file1 will print the first 3 lines of file1.
history	Prints a list of all past commands typed in the current terminal session.
less file1	Displays the contents of file1. Less is a similar to more, but which allows backward and forward movement in the file. Furthermore, less doesn't have to read the entire input file before starting, so it is faster than some text editors for large input files.
ln target link	This creates a hard link between target and link.
ln -s target link	This creates a soft link between target and link.
ls	Lists all files and directories in the current directory.

ls - R	Lists files in the sub-directories, as well
ls - a	Lists hidden files as well
ls - al	Lists files and directories with detailed information like permissions, size, owner, etc.
jobs	Gives you a list of jobs, each of which is associated with a job number.
man command	Gives help information on command.
mkdir directory	Creates a new directory in the current working directory or at the specified path.
more file1	Displays the contents of file1.
mv file1 file2	Moves file1 into file2. If file2 doesn't exist, then it renames file1 to file2.
ps	Prints the current running processes.
pwd	Prints the absolute path of the current working directory.
rm file1	Deletes file1.
rmdir directory1	Deletes directory1.

sort file1	This rearranges the lines in file1 so that they are sorted, numerically and alphabetically. By default, the rules for sorting are: 1. Lines starting with a number will appear before lines starting with a letter. 2. Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet. 3. If two or more lines, of different lengths, share the first x number of letters, then sort will print the lines from shortest length to longest. 4. Lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase, if they have the same length. **Note:** sort does not affect the original file in any way. E.g. Suppose we have a file called file1 that contains these lines my name is rick My name is rick 1 Then, if we do sort file1, we get the following: 1 my name is rick My name is rick E.g. Suppose we have a file called file2 that contains these lines my name is rick lan My name is rick 1 Then, if we do sort file2, we get the following: 1 My name is rick my name is rick lan
stat file1/dir1	Gives detailed information about file1/dir1
tail (option) (num) file1	Prints the last 10 lines of file1, by default. If you want to set the number of lines this prints, you need to do: last -n num file1 and it will print the last num lines of file1. E.g. last file1 will print the last 10 lines of file1. E.g. last -n 3 file1 will print the last 3 lines of file1

uniq file1	Prints out the contents of file1 but removes all the duplicates in adjacent lines. E.g. Suppose we have a file called file1 and it contains the following: cat cat cat dog dog Then, if we do uniq file1, it would print cat dog E.g. Suppose we have a file called file2 and it contains the following: cat dog cat dog cat Then, if we do uniq file2, it would print cat dog cat dog cat
vi file1	Creates a file called file1.
wc file1	Prints the number of lines, words, and characters file1 has. E.g. Suppose we have a file called file1 and it contains the following: cat dog cat dog cat Then, if we do wc file1, we get: 5 5 20 file1 The first 5 means there are 5 lines in file1. The second 5 means there are 5 words in file1. The 20 means there are 20 characters in file1. Note that because there is a space after each line, there are 20 characters.

Redirection:

1. **Standard Input (stdin):**
   - The file descriptor for stdin is 0.
   - Denoted as <.
   - E.g. cat < file1 works the same as cat file1.
2. **Standard Output (stdout):**
   - The file descriptor for stdout is 1.
   - Denoted as either > or >>.
   - The general form is **command > file** or **command >> file**, where **command** is any command that outputs to stdout.
   - **Note:** If **file** was not created before, then this will create it for you.
   - > will redirect the output to a file. This will overwrite whatever was in the file before.
   - >> will append the output to a file. This will add the output to the end of whatever was in the file before.
   - E.g. Suppose we have 2 files, file1 and file2.
     Suppose the contents of file1 is 12345 and the contents of file2 is abcde.
     If we do cat file1 >> file2 and then do cat file2, the output of cat file2 is:
     abcde
     12345
     If we do cat file2 > file1 and then do cat file1, the output of cat file1 is:
     abcde
     12345
3. **Standard Error (stderr):**
   - The file descriptor for stderr is 2.
   - Denoted as 2> or 2>>.
   - The general form is **command error 2> file** or **command error 2>> file**.
   - **Note:** If **file** was not created before, then this will create it for you.
   - 2> will redirect the error message into the file.
   - 2>> will append the error message into the file.
   - This is useful for shell scripting because you usually do not want error messages cluttering up the normal program output.
   - E.g. Suppose that there is no file in the current working directory called x. Suppose we do cat x 2> file1. This will overwrite the contents of file1 with the error message.
   - E.g. Suppose that there is no file in the current working directory called x. Suppose we do cat x 2>> file1. This will append the contents of file1 with the error message.