Bash meta characters

We can use special characters called `metacharacters` in a Bash that the shell interprets rather than passing to the command. See the examples below:

Symbol	Example	How it works?
`<`	`sort < filename.txt`	Get input for the command to the left from the file listed to the right of this symbol.
`>`	`echo "BASH" > filename.txt`	Send the output of the command on the left into the file named on the right of this symbol. If the file does not exist, create it. If it does exist, overwrite it (erase everything in it and put this output as the new content).
`>>`	`date >> filename.txt`	Send the output of the command on the left to the end of the file named on the right of this symbol. If the file does not exist, it will

		be created. If it does exist, it will be appended.
`?`	`ls e?.txt`	This symbol matches any single character.
`*`	`ls e*.txt`	This symbol matches any number of any characters
`[ ]`	`ls -l e[abc].txt`	listing. One of the characters within the square brackets must be matched.
`$`	`my_variable="this string"` and `echo $my_variable`	Denotes that a string is a variable name. Not used when assigning a variable.
\	`echo "I have \$10.00"`	Escape. Ignore the shell's special meaning for the character after this symbol. Treat it as though it is just an ordinary character.
`( )`	`(cal; date) > filename.txt`	If you want to run two commands and send their output to the same place, put them in a group together.
`{ }`	`numbers=( 5 2 7 )`	Used in special cases

	and `echo ${numbers[1]}`	for variables with the `$`. One use is in parameter substitution (see the `$` definition, above), the other is in arrays. Example:
`"`	`my_variable="This is a test."`	Used to group strings that contain spaces and other special characters.
`'`	`my_variable='This is a backslash: \'`	Used to prevent the shell from interpretting special characters within the quoted string.
`` ` ``	``my_variable="This is the date: `date`"``	Unquoting. Used within a quoted string to force the shell to interpret and run the command between the backticks.
`&&`	`mkdir stuff && echo "Made the directory"`	Run the command to the right of the double-ampersand **only if** the command on the left succeeded in running.
`&`	`cat /etc/passwd &`	Run the process in the background, allowing you to continue your work on the

		command line.
`;`	`date;cat passwd; date`	Allows you to list multiple commands on a single line, separated by this character.
`=`	`my_variable="Hello World!"`	Assignment. Set the variable named on the left to the value presented on the right.