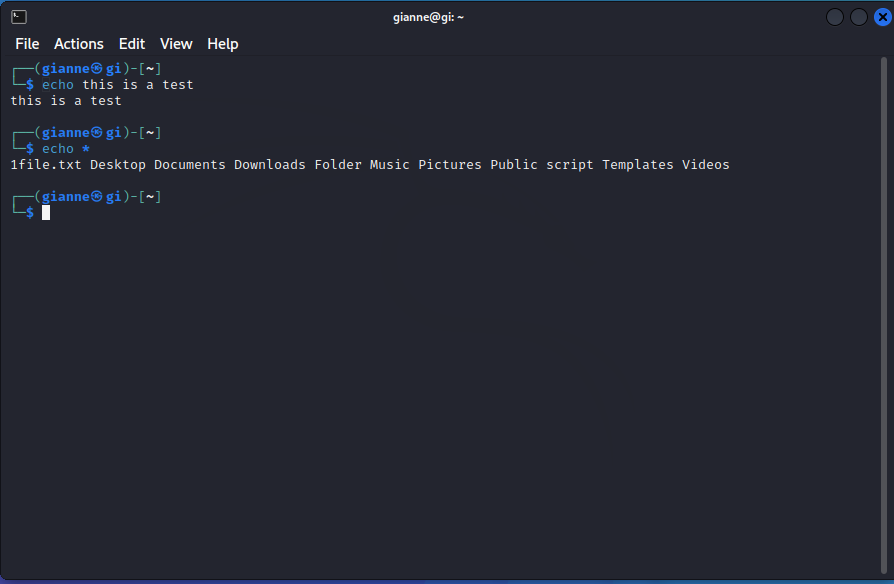
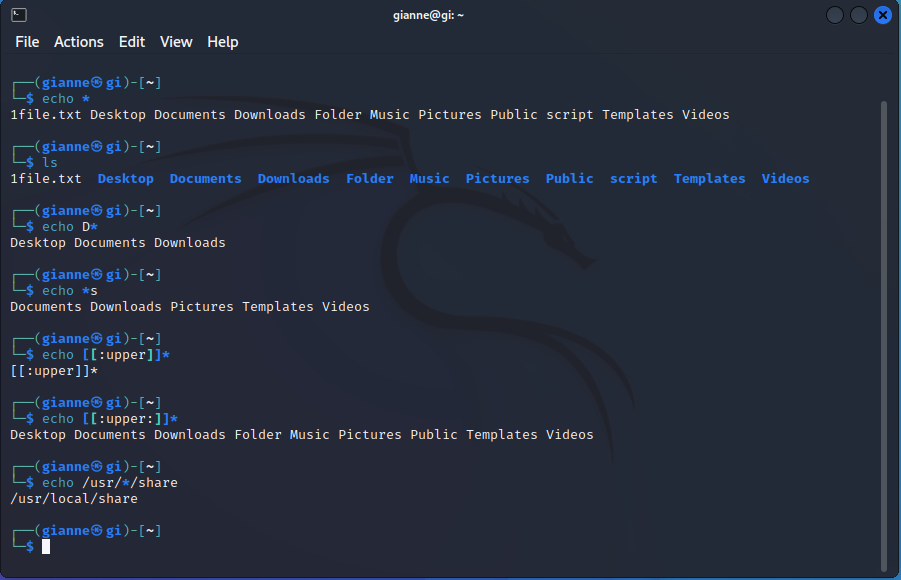
**Activity 7 - Seeing The World As The Shell Sees It**

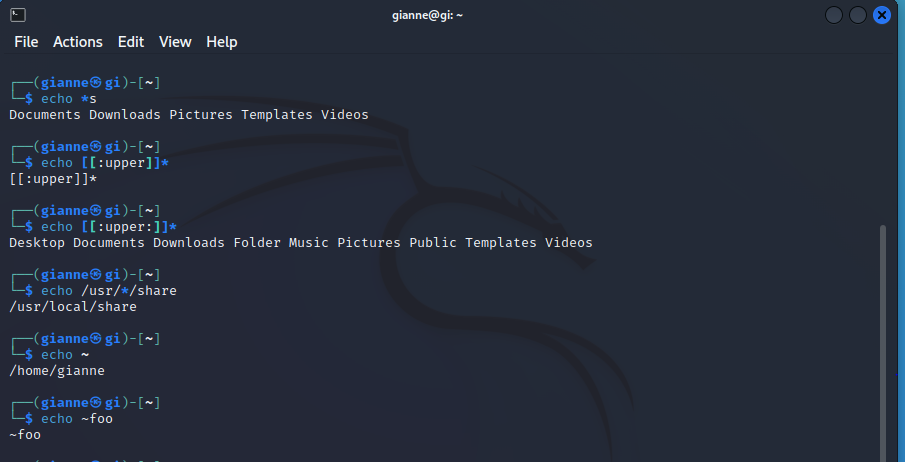
**Expansion -** With expansion, you enter something and it is expanded into something else before the shell acts upon it. To demon- strate what we mean by this, let's take a look at the echo command. echo is a shell builtin that performs a very simple task. It prints out its text arguments on standard out- put:

****

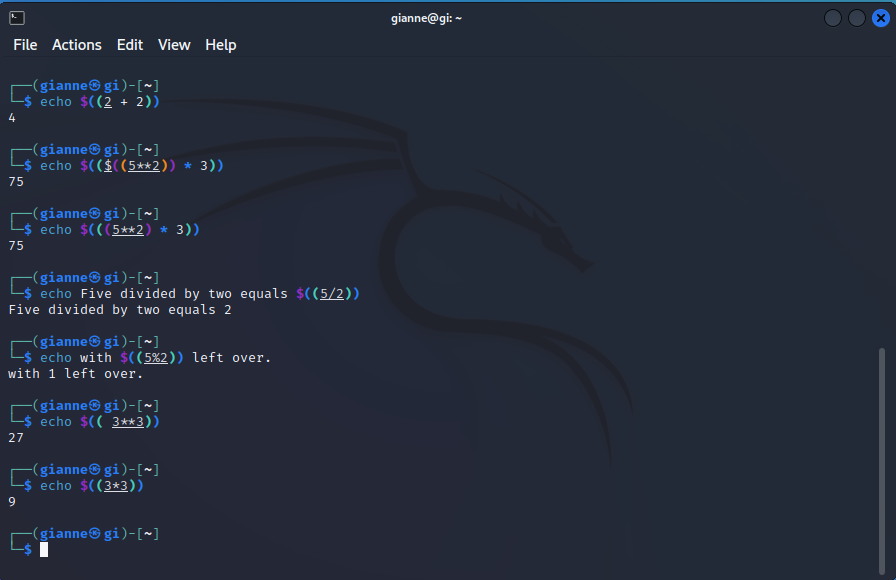
**Pathname Expansion -** The mechanism by which wildcards work is called *pathname expansion*. If we try some of the techniques that we employed in our earlier chapters, we will see that they are really expansions. Given a home directory that looks like this:



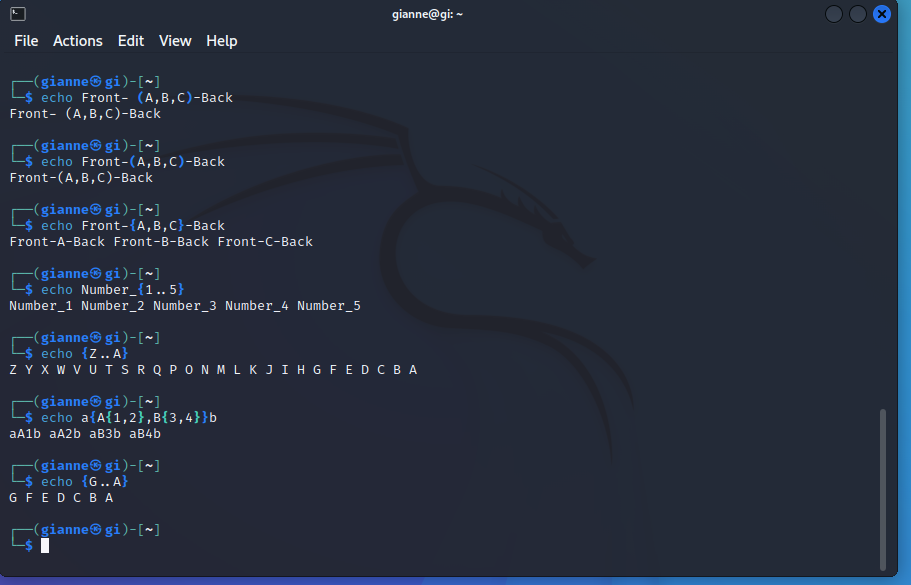
**Tilde Expansion -** As you may recall from our introduction to the cd command, the tilde character (“~”) has a special meaning. When used at the beginning of a word, it expands into the name of the home directory of the named user, or if no user is named, the home directory of the cur- rent user:

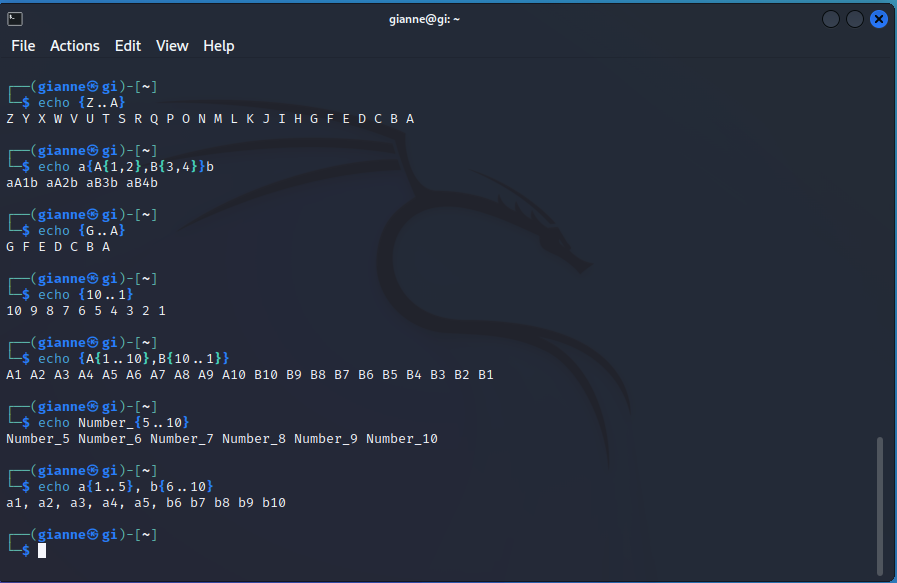
****

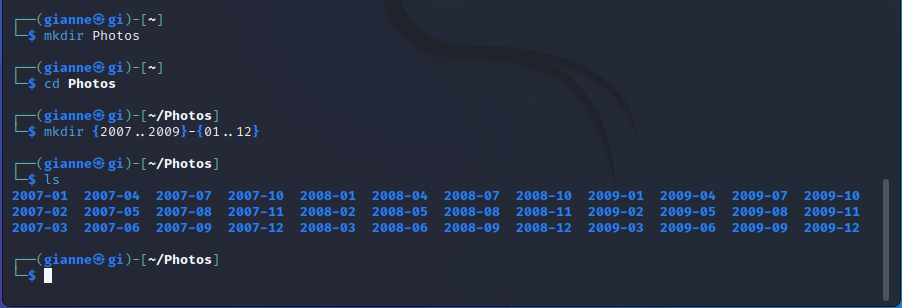
**Arithmetic Expansion -** The shell allows arithmetic to be performed by expansion. This allow us to use the shell prompt as a calculator:

****

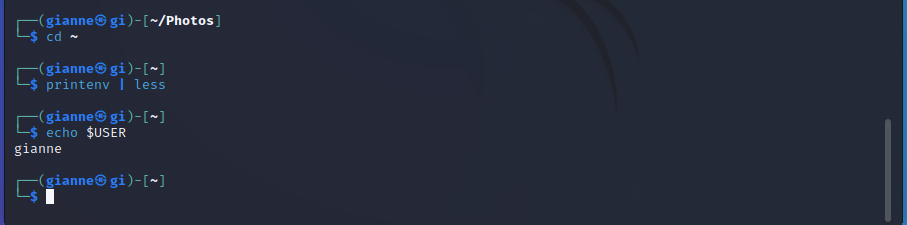
**Brace Expansion -** the strangest expansion is called *brace expansion*. With it, you can create multi- ple text strings from a pattern containing braces

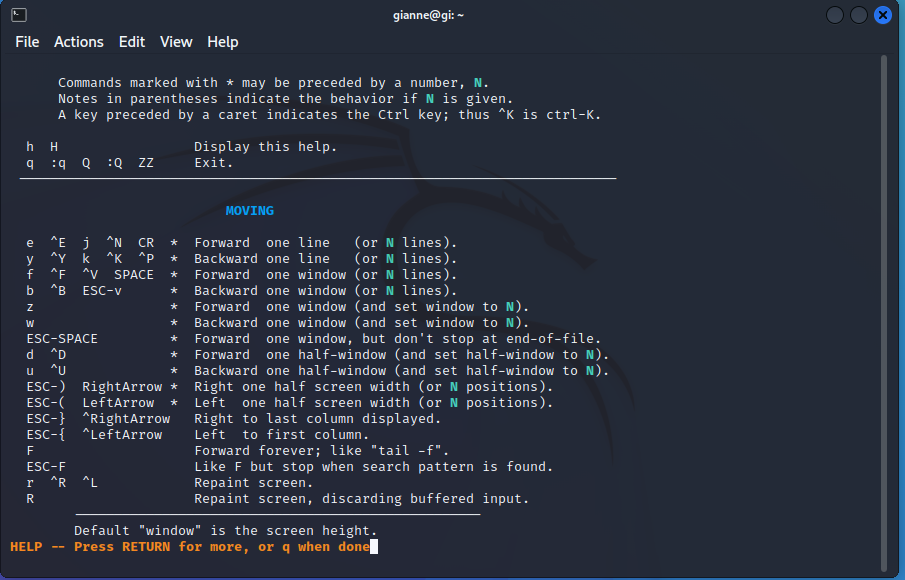
****

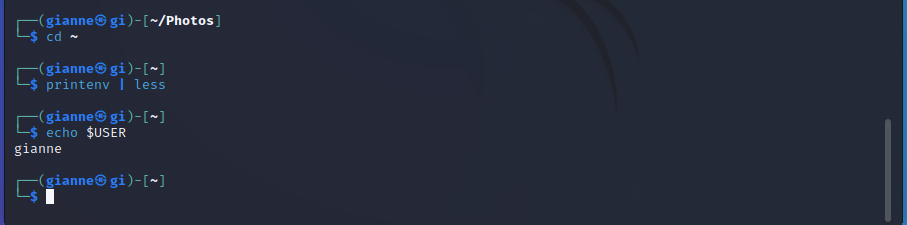
****

****

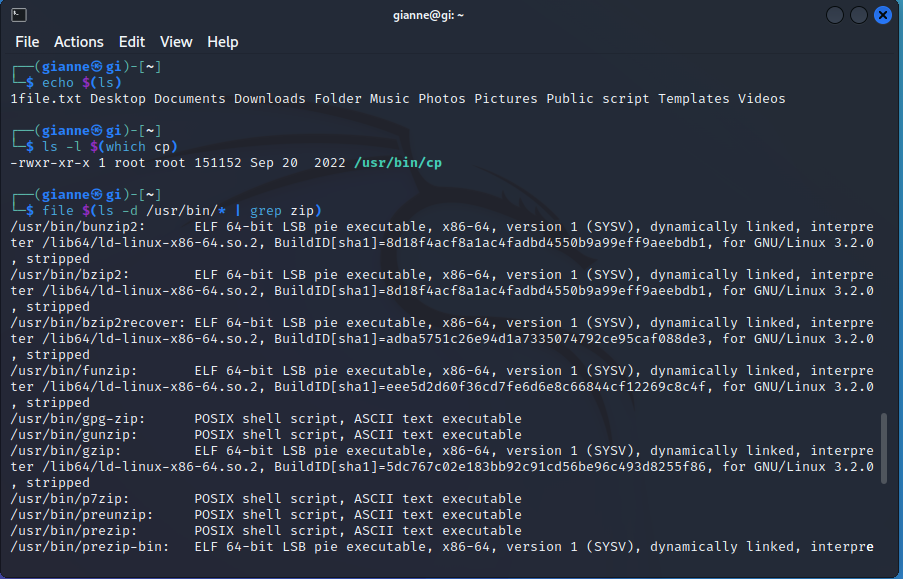
**Parameter Expansion -** We're only going to touch briefly on parameter expansion in this chapter, but we'll be covering it extensively later. It's a feature that is more useful in shell scripts than directly on the command line. Many of its capabilities have to do with the system's ability to store small chunks of data and to give each chunk a name

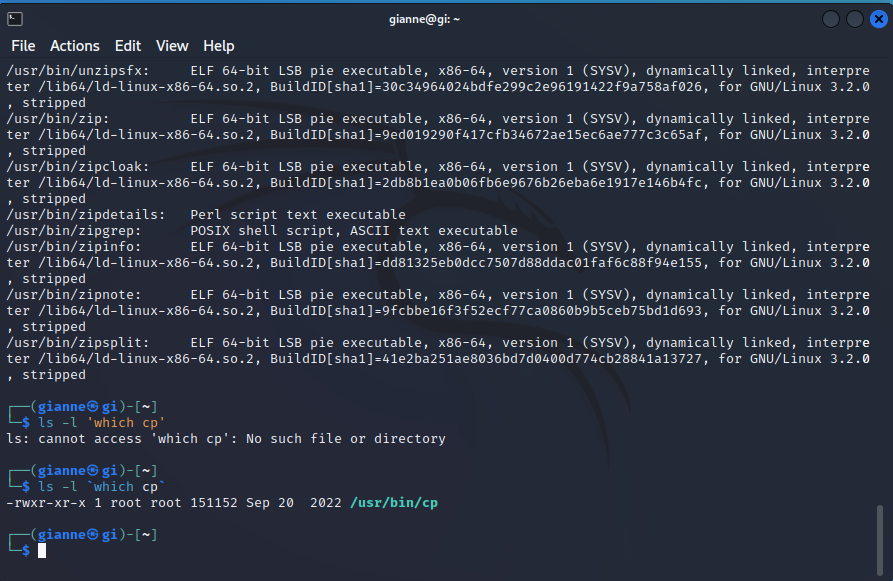
****

****

****

**Command Substitution -** Command substitution allows us to use the output of a command as an expansion:

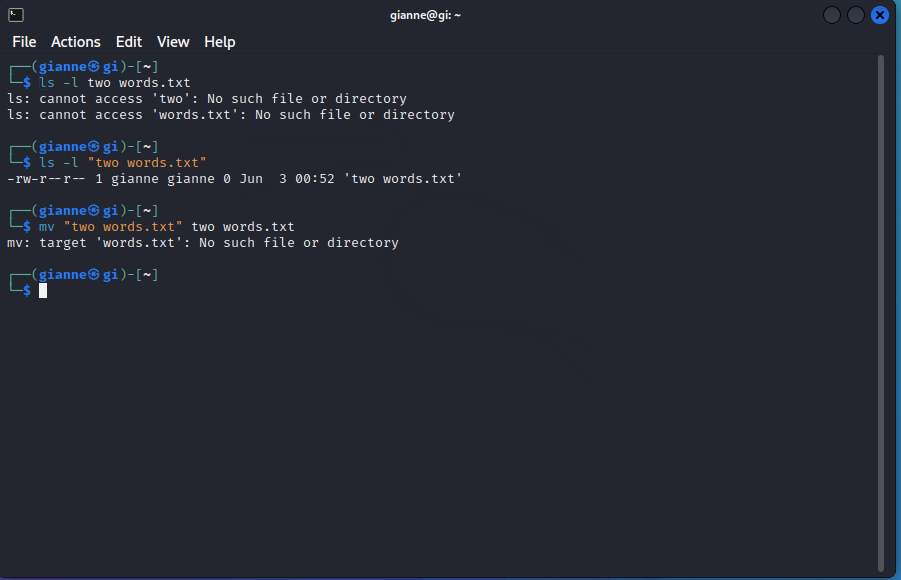
****

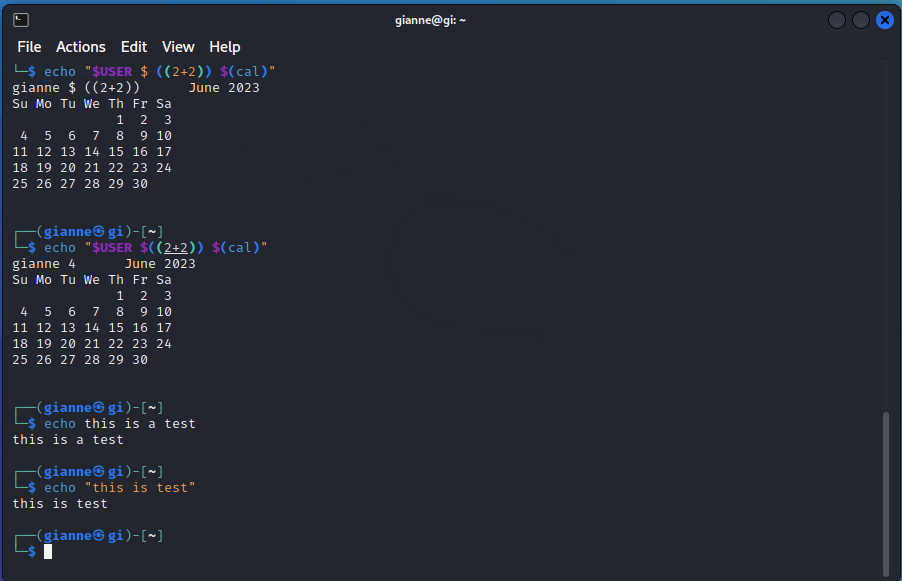
****

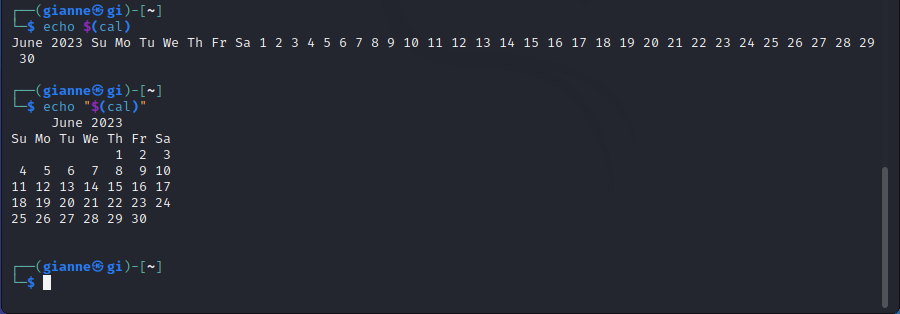
**Quoting -** In the first example, *word-splitting* by the shell removed extra whitespace from the echo command's list of arguments. In the second example, parameter expansion substituted an empty string for the value of “$1” because it was an undefined variable. The shell pro- vides a mechanism called *quoting* to selectively suppress unwanted expansions.

****

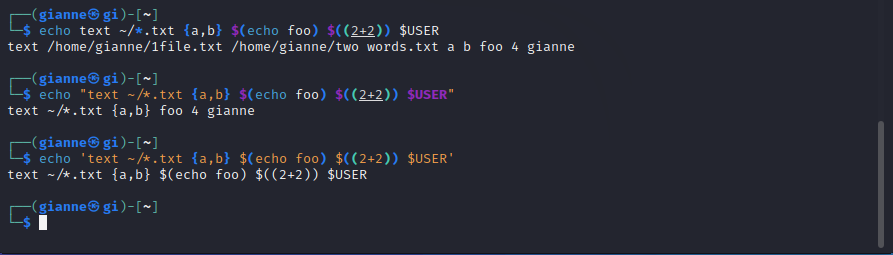
**Double Quotas -** Using double quotes, we can cope with filenames con- taining embedded spaces. Say we were the unfortunate victim of a file called two words.txt. If we tried to use this on the command line, word-splitting would cause this to be treated as two separate arguments rather than the desired single argument:

****

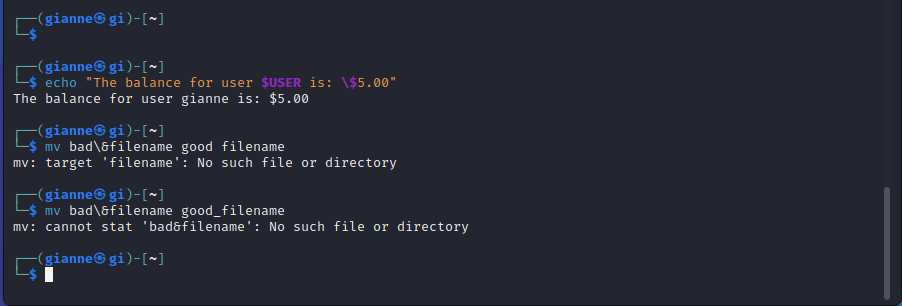
****

****

**Single Quotas -** If we need to suppress *all* expansions, we use *single quotes*. Here is a comparison of un- quoted, double quotes, and single quotes:

****

**Escaping Characters -** To do this, we can precede a charac- ter with a backslash, which in this context is called the *escape character*. Often this is done inside double quotes to selectively prevent an expansion:

****