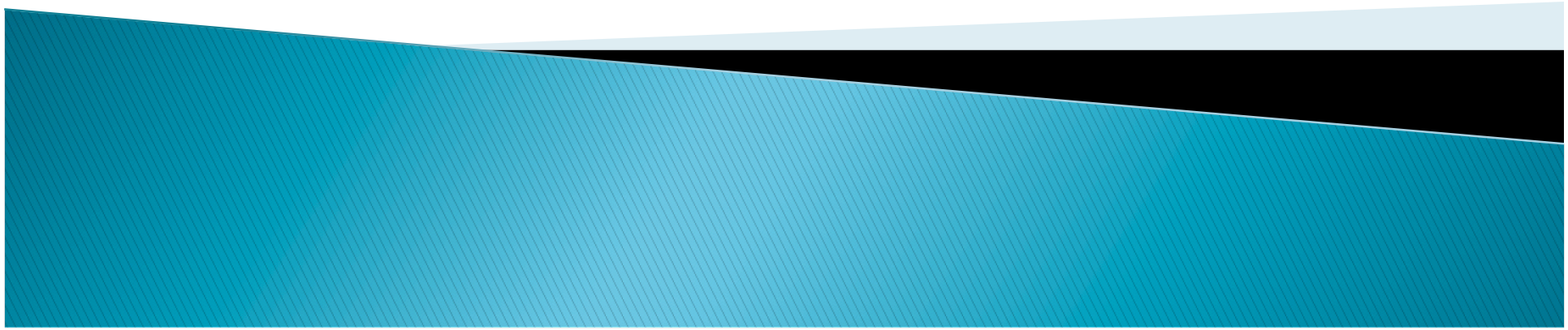
SED

(Stream Editor)

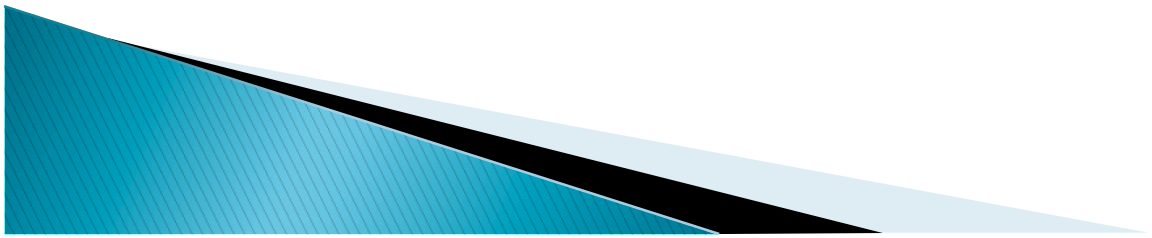
By:
Ross Mills



What is SED?

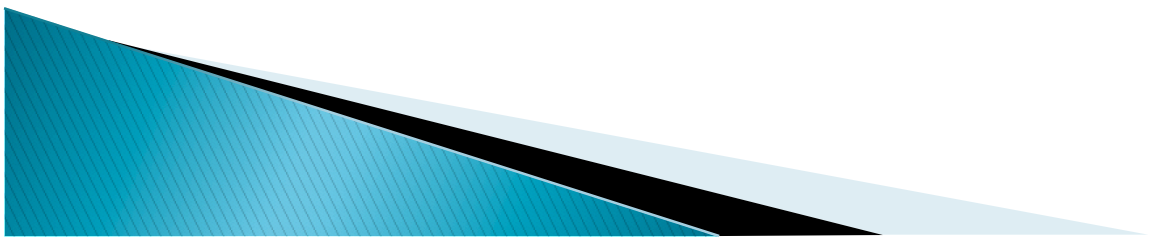
- Sed is an acronym for stream editor
- Instead of altering the original file, sed is used to scan the input file line by line and applies instructions in a script to the file
- There are three options to use for sed:

-n, -f and -e.



The Three SED Options

- The -n flag keeps the computer from automatically outputting the result. This lets the user control what is being printed
- -f indicates that there is a script file to be used
- -e is the default option of Sed. It means that the script is on the command line and not in a specific file. However, you are not required to write -e when using Sed.



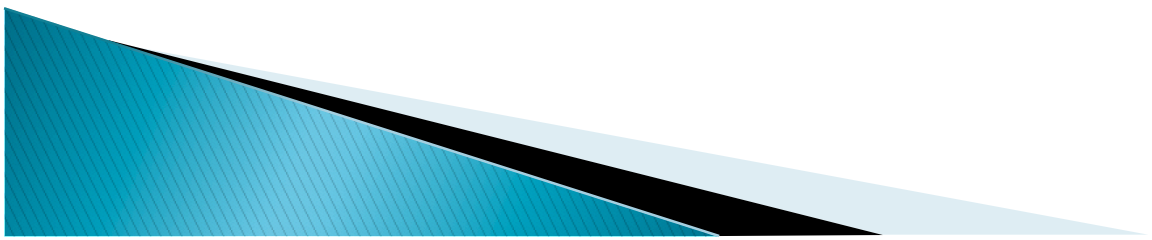
Script Formats

- If the script fits in a couple of lines, then it's instructions can just be included in the command line, but it must be enclosed in single quotes:

  sed –e 'address command' input_file

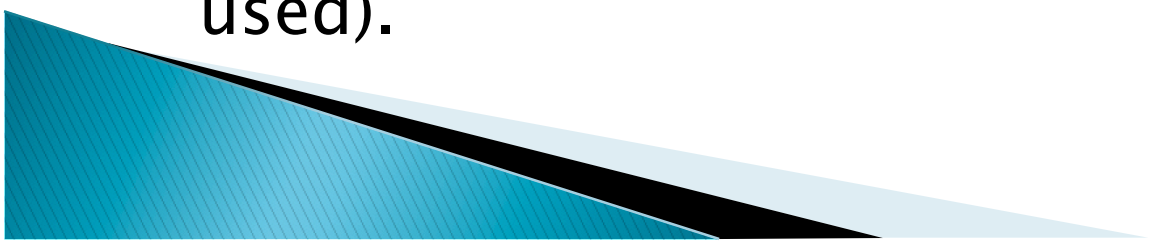
- For scripts that are longer or may be repeated, a text file containing the script should be used, often times named .sed to specify:

  sed –f script.sed input_file



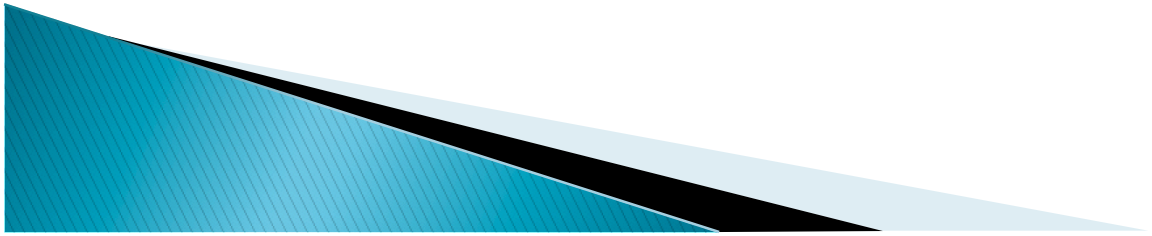
Operation

- Each line in the input file has a line number given to it by sed
- For every line in the file, sed copies an input line to pattern space, which is a buffer that holds one or more text lines for processing.
- Then sed applies the instructions given in the script to all of the lines in the pattern space that match the specified addresses in the instruction.
- After applying the instructions, sed then copies the pattern space to the output file(unless -n was used).



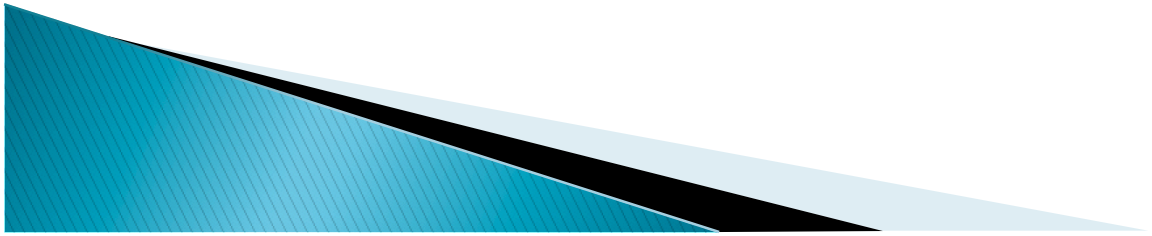
Addresses

- There are four types of addresses in sed:
  - single line
  - set of lines
  - range of lines
  - nested addresses



Single Line

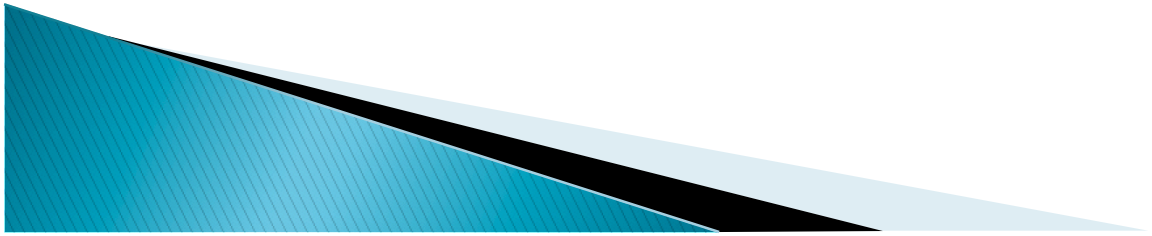
- Single line addresses only specify one line, using either a number or '$' which means the last line of the file
  - Example:
    - sed 5d poem.txt
      → This example would look in poem.txt and delete the line numbered 5



Set of Lines

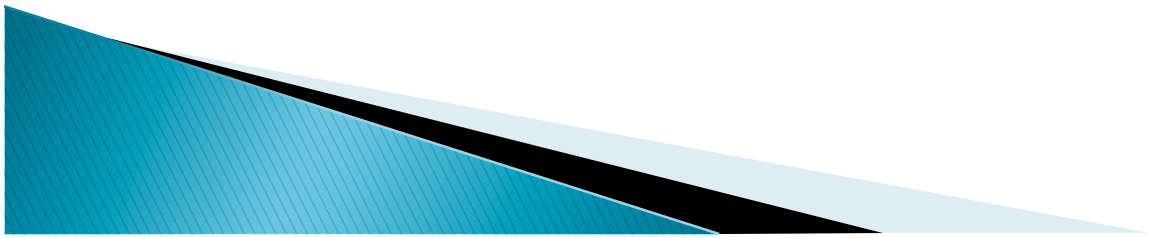
- Don't necessarily have to be consecutive lines
- Use regular expressions written in between two slashes to specify
- Regular expression may match multiple lines
- Even a line that matches might not see an instruction that will effect the line
  - Example: sed '/name/Name/NAME/d' test.txt

    → Deletes any line that contains "name", "Name" or "NAME"



Range of Addresses

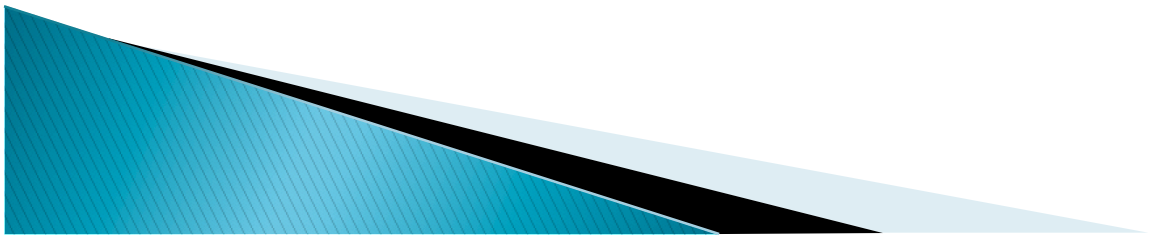
- Defines a set of consecutive lines
- Format is start-address,end-address
- Can be a line number or a regular expression:
  Line-num,/regexp/
- Special case: range of 1,$ which is the first to the last line
  - Example: sed -n '1,1000d' poem.txt
    → Deletes lines 1 to 1000 in poem.txt



Nested Addresses

- An address contained inside another address
- The outer address must be set of lines or an address range
- The inner address may be single line, set of lines or an address range.
- Example: 1,10 {

```
        /begin/,/end/d
}
```

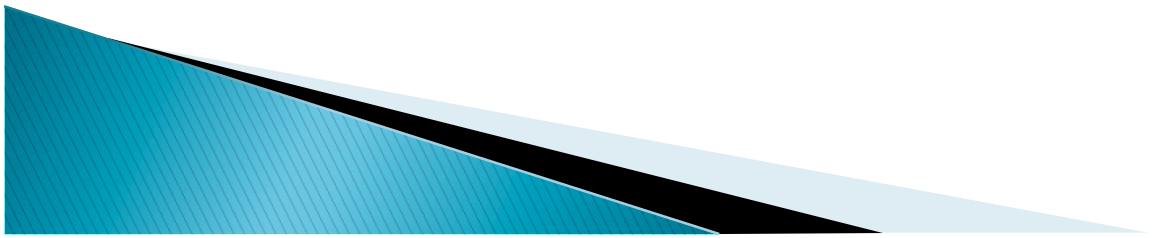


Commands

- Sed has many different commands that may be used but they are grouped into the following categories:
  - Line Number Command
  - Modify Commands
  - Substitute Commands
  - Transform
  - Input/output commands
  - File Commands
  - Branch Commands
  - Hold Space Commands
  - Quit Command

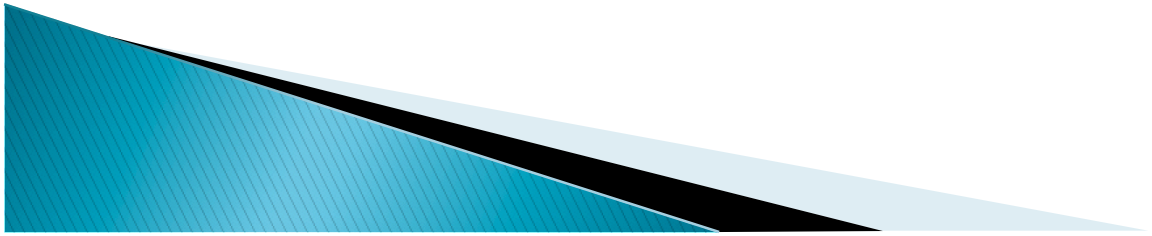
Line Number Command

- The line number command is called by using '='
- This will print the current line number
- Example: sed -n '/name/=' name.txt
  - Looks in the file name.txt to find any line where the word 'name' occurs and then it will print that line to the screen.



Modify Commands

- Insert (i), which inserts a line above every location where the regular expression is found.
  - Sed '/name/i\name2' test.txt : Creates a new line that says 'name2' above any line where 'name' is found
- Append (a), does the same as insert except it adds a line below the found regular expression
- Change (c), replaces the selected lines of text
  - Sed '/name/c\/name2/' test.txt : Replaces all lines that contain 'name' with the line, 'name2'
- Delete (d), deletes the line selected
  - Sed '1d' test.txt : Deletes the first line in the file



Substitute Commands

- Changes all occurrences of the regular expression to whatever is specified.
  - For example take the text:

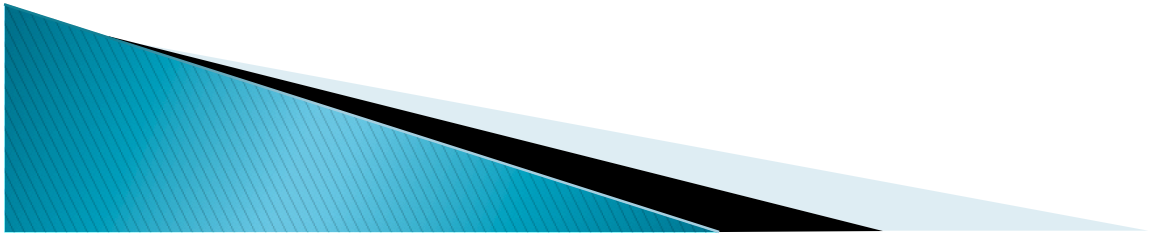
    My name is Ross Mills

    My first name is Ross

If we run : sed ‘s/Ross/John/’ text, the output will be:

My name is John Mills

My first name is John



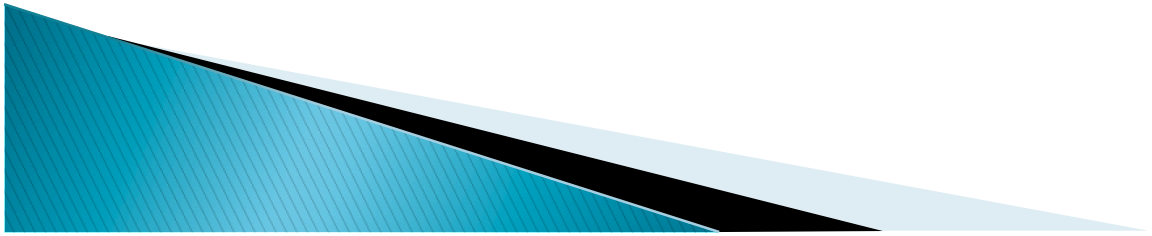
Transform

- The transform command (y) is used for transforming text. Often times it is used to turn letters from lower to uppercase.
  - Example:

    My name is Ross Mills
    My first name is Ross

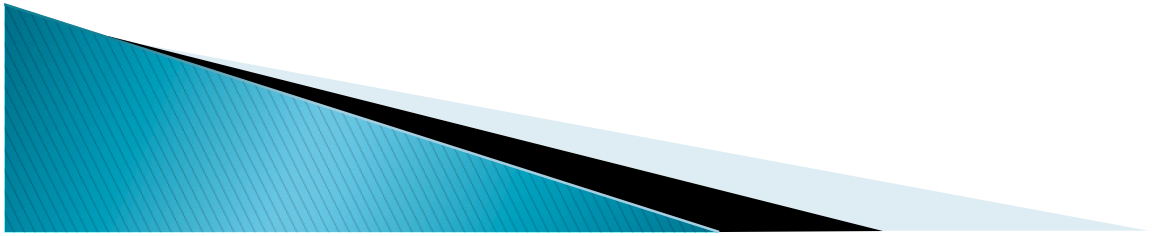
    sed ‘y/abcdef/ABCDEF/’ text would output the following:

    My nAmE is Ross Mills
    My First nAmE is Ross



Input/Output Commands

- Next (n), reads the next input line and starts processing the new line with the command rather than the first command
  - Sed '/Line1/{n; s/Line1/Line2/}' test.txt : If the word Line1 is found, all occurrences of Line1 in the next line are changed to line2
- Append Next (N), appends the next line to pattern space so the previous command would change Line1 to Line2 on the first line that it was found
- Print (p) and Print first line (P), prints the contents of the pattern space or the first line of the pattern space
- List Command (l), prints the characters that are not usually printed such as $ after each line.



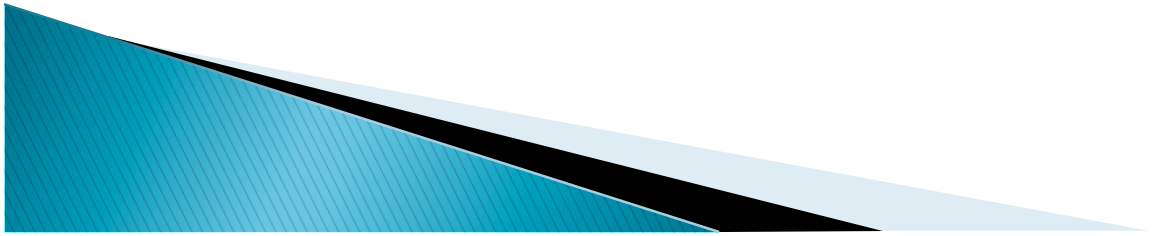
File Commands

- Read file command (r), reads lines from a file and if an expression is found, it is added to the file.
  - Example: Line.txt contains the word line and

    test.txt is:
    My name is Ross Mills
    My first name is Ross
    - Sed '/Ross/r line.txt' test.txt would output the following:

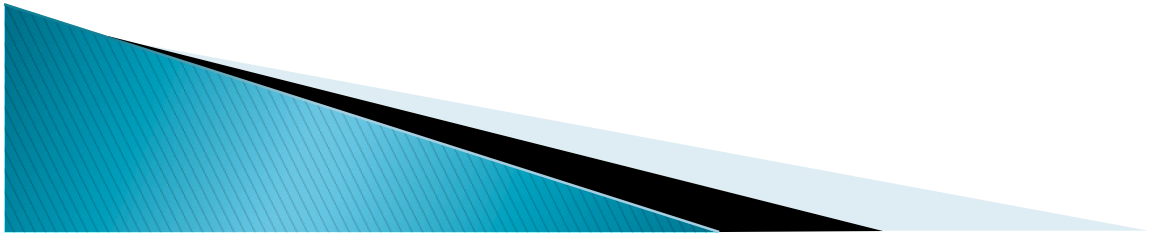
      My name is Ross Mills
      line
      My first name is Ross
      Line
- Write file command (w), writes lines out to a file
  - Example using the same test.txt :
    sed '/Mills/ w test2.txt' test.txt : Creates a new file called test2.txt which contains the lines where "Mills" was found so test2.txt reads:

      My name is Ross Mills



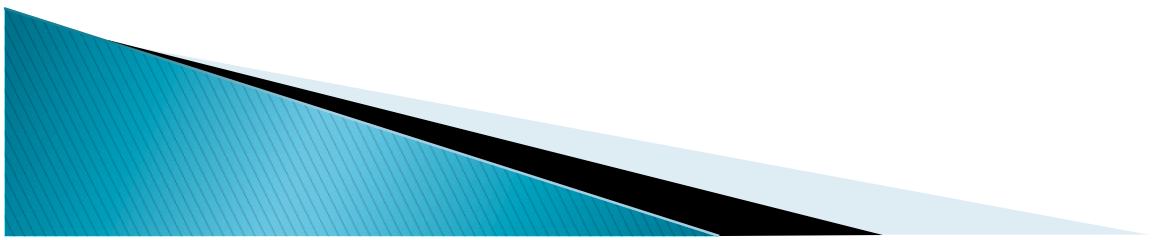
Branch Commands

- Branch (b), takes sed to the label which is specified using ‘:label_name’
- Branch on substitution (t), only takes the branch if the regular expression was found and substituted.



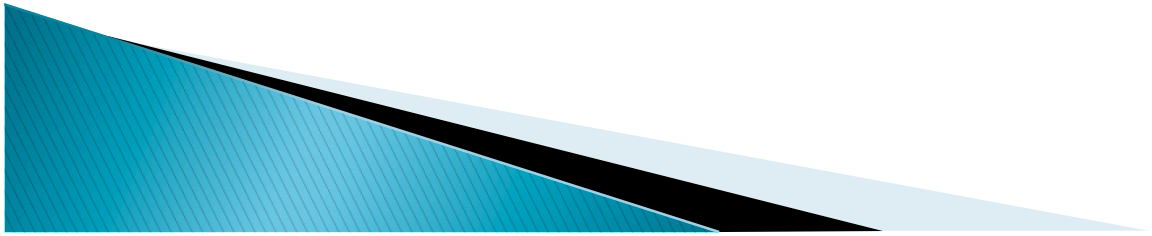
Hold Space Commands

- Hold and Destroy (h) copies the contents of the pattern space to the hold space
- Hold and Append (H) adds the contents of the pattern space to the hold space
- Get and Destroy (g), gets what is in the holding space and overwrites it to the pattern space
- Get and Append (G), gets what is in the holding space and adds it to the pattern space
- Exchange (x), switches the content in the holding space to the pattern space



Quit Command

- Quit (q), prints the content of the pattern space and then exits or quits sed
  - Example: sed ‘2q’ test.txt will print the first two lines of test.txt and then quits the program.



Why Use Sed?

- Sed can allow the user to apply actual text transformations to a file
- Sed may be used instead of Grep, however Grep is much more efficient when you are not trying to alter text
- Sed could be used for many things as seen with the numerous commands available