```
#This is a presentation using the suckless sent tool
#You can find information on the tool and how to view this file at
# https://tools.suckless.org/sent/
```

SEd & AWK

An Introduction to the Power Tools of Text Processing

Tools with a similar purpose

To edit text

Like a text editor?

Perform more editing than writing

Perform scripts on one or multiple text files

or stdin

Allowing you to edit absurd amounts of text quickly

Similarities

Invoked Similarly

```
command [options] script input
        $ awk -f script.awk input
        $ sed -i "s/sllug/sluug/g" input
```

Instructions can be specified in a script or on the command line

Great filters for your pipes

```
df /| sed -n 2p | awk '{print $5 " "  $1}' | sort -n | awk '{print $2 " " $1}'
```

Both use RegEx

A note on regular expressions

Like algebra except more so

Lots of simple rules

That come together to ruin your GPA freshman year

Things to remember

They are expressions

"The cat's out of the bag"

```
/^"([a-zA-Z]{3}) [a-z]*'s [^asd]{3} [o|f]. (\1) [^h-z]+"$/
```

The expressions are not literal they need to be evaluated

Take practice

Many different versions

Check which one you are using

Safety

Avoid working directly on your input

Never pipe the output to the input

You can and will null it out

Scripting

Two parts to each instruction

Patterns & procedures

Each line is read

Whenever a pattern is matched the procedure is performed

Each instruction is compared against each input line

Once all instructions are compared move on to the next line

Sed prints the output automatically

AWK needs to be instructed to do so

SEd – The Stream Editor

Descendant of Ed(1) the Standard Text Editor

Command Syntax is similar

```
        * g/re/p
```
Prints all the lines containing the regular expression re

I wonder where I've seen that before?

```
        $ sed –n '/re/p'
```
Prints all the lines containing the regular expression re

SEd operates on all lines by default

SEd can be restricted by providing addresses to operate on

5 Basic SEd commands

Flags:
- –n – suppress automatic printing
- –i – in place, overwrite input file
- –e – execute the following command
- –f – execute commands from this script file

d – delete

i – insert (before a matching line)

a – append (after a matching line)

c – change

s – substitute

[address]s/RegEx/replacement/flags

Address – checks all lines matching the address

Can be a RegEx

Replacement

Special Characters
- * & – the string matched by the RexEx
- * \# – back reference to capture group #
- * \n – newline
- * \ – escape character

Flags:

```
        * g - global, change all occurrences of a regex on a line
                        not just the first
        * n - number, replace only nth occurrence of the regex
        * p - print
        * w - write, write to file

AWK -  Aho, Weinberger, Kernighan

Programming Language built to manipulate structured data

View text files as databases of records and fields

Arithmetic and string operations

Loops and conditionals
        * for loops
        * while loops
        * if statements

Define functions

function name ( arguments ) {
        statments
        return expression
}

Variables to store and manipulate data

You don't need to initialize before assigning or worry about type

Splits input lines into variables based on white space

Use $ to access

$127 - is the 127 field in the input

$0 is the whole input

Structure of an AWK Script

BEGIN { performed before input }

RegEx { performed on each matching line }

End { after input is read }

This structure lets you ignore the messy boiler plate of most other languages

Flags
        * -f - script file
        * -F - field delimiter
        * -v - assign variable a value

Further Reading
        * sed & awk By Dale Dougberty & Arnold Robbins, O'Reilly
        * regexcrossword.com
        * regexr.com


Any Questions
```