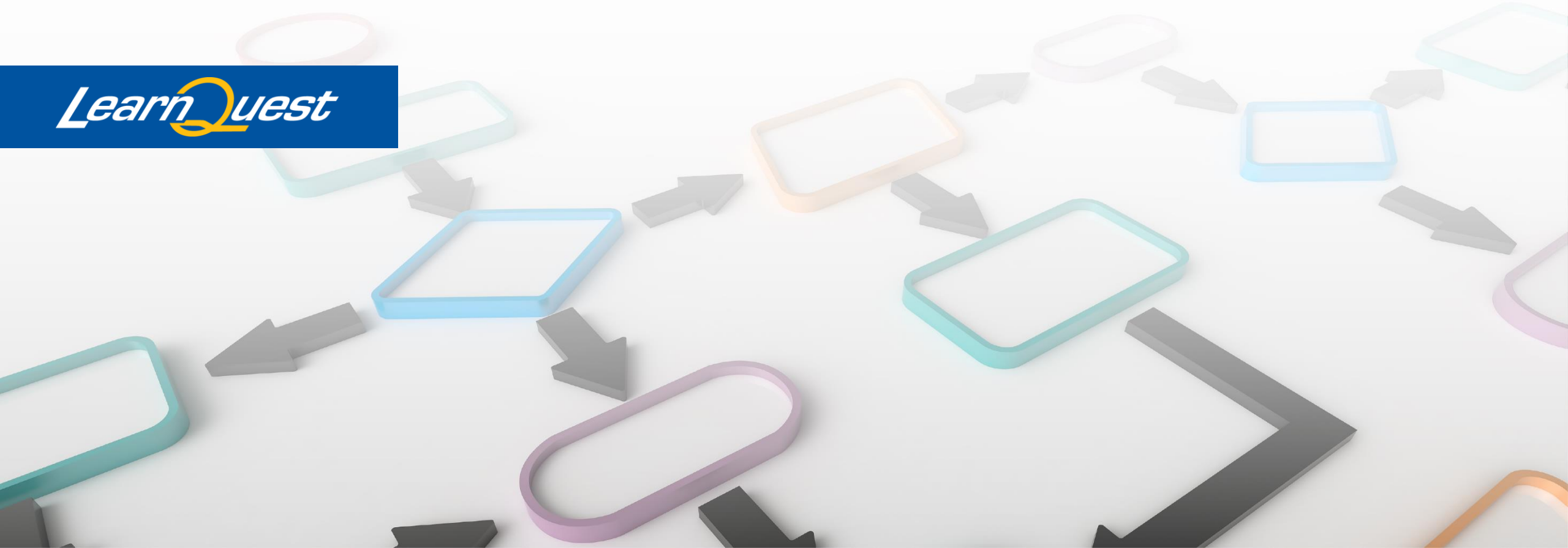
Linux Fundamentals

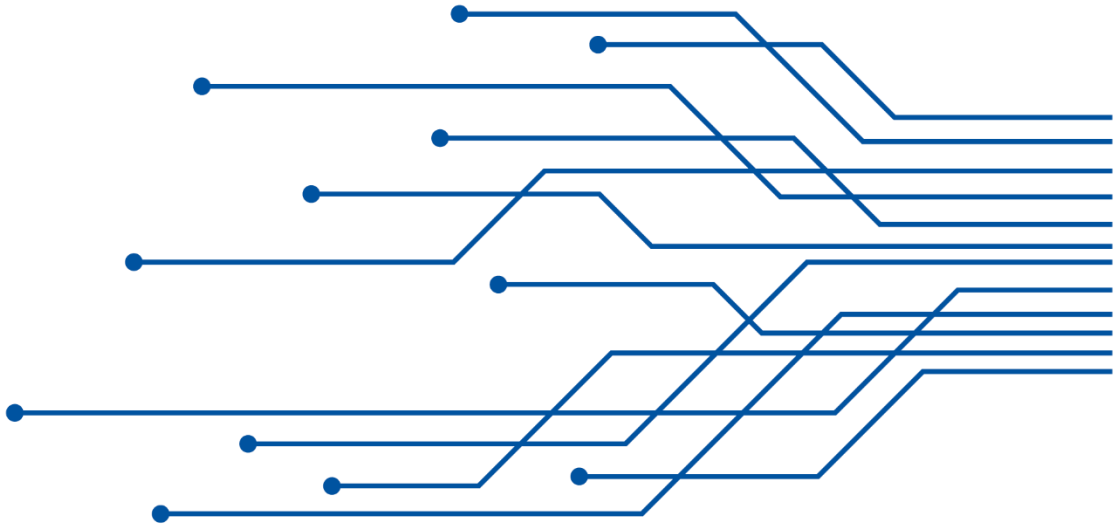
- 1st Course in Linux Foundations Specialization



Search and Analyze Text

In the fourth module of this course, we will touch on how we can combine commands together in Linux to create automations and build new tools.

4



Learning Objectives

Search and Analyze Text

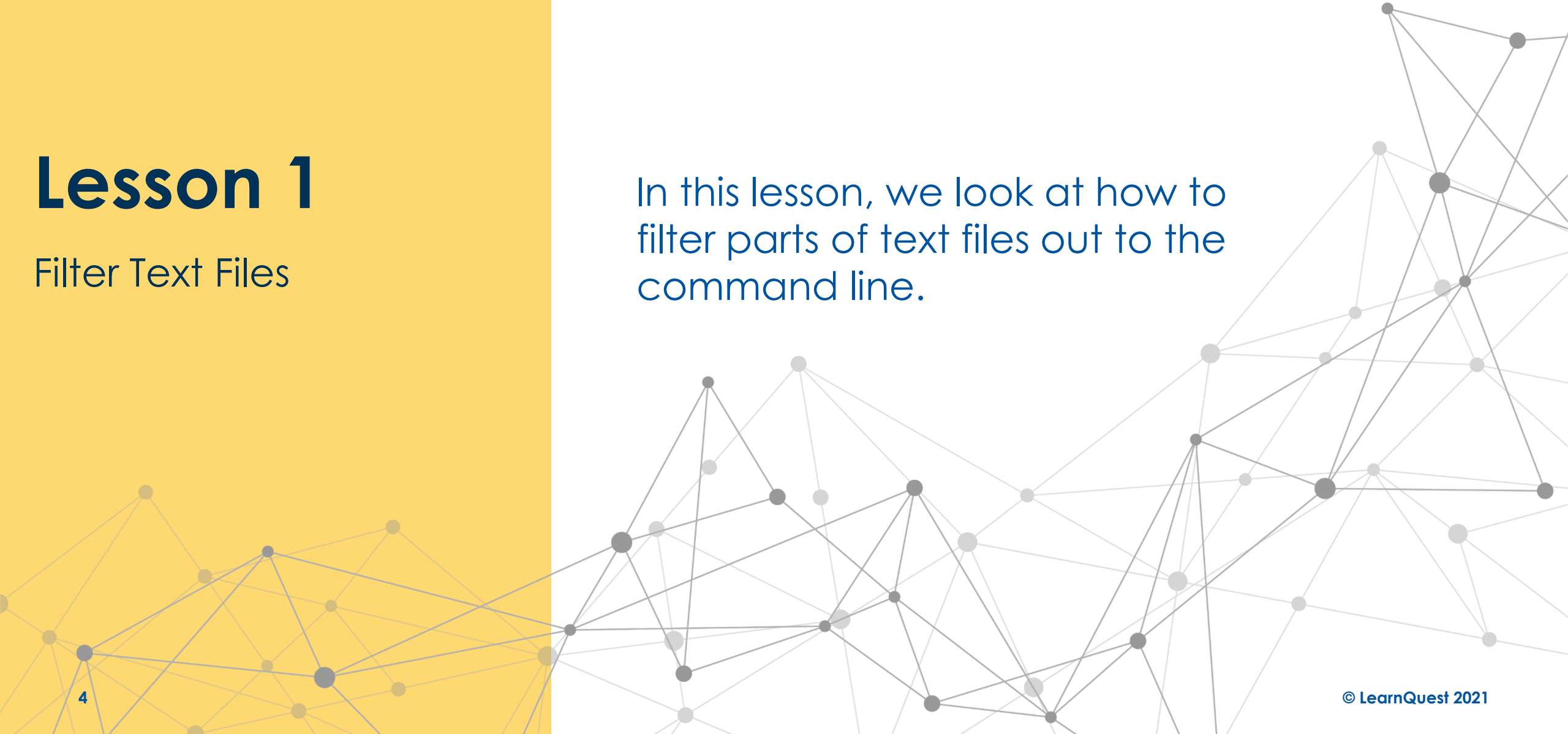
Upon completion of this module, learners will be able to:

- Filter Text Files
- Redirect Standard Output
- Redirect Standard Input
- Redirect Standard Error
- Pipe Command Together
- Edit Text Files

Lesson 1

Filter Text Files

In this lesson, we look at how to filter parts of text files out to the command line.



Cut Command

Cuts out sections from each line of files and writes the result to standard output

Example Usage:

- cut -d ":" -f 1,7 /etc/passwd
- cut -c 1-5 /etc/passwd

Options:

- -c nlist: Display only the record characters in the nlist.
- -d d: Designate the record's field delimiter as d. This overrides the Tab default delimiter. Put d within quotation marks to avoid unexpected results.
- -f flist: Display only the record's fields denoted by flist

Grep Command

Uses regular expressions to filter text from files.

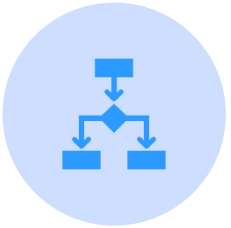
Example Usage:

- grep daemon.*nologin /etc/passwd
- grep ^root /etc/passwd
- grep -v nologin$ /etc/passwd

Options:

- -E : Designate the PATTERN as an extended regular expression.
- -i : Ignore case.
- -r : Search a directory's contents, and for any subdirectory within the original directory tree, consecutively search its contents as well.
- -v : Display only text files records that do not contain a PATTERN match.

Basic Regular Expressions (BREs)



Represent multiple characters: .*

Signify one character: single dot (.)

Represent diverse characters with brackets
Example: [a,e,i,o,u]

Signify range of characters with brackets and dash
Example: [A-z]

Represent characters to find at line's beginning: ^

Signify characters to find at line's end: $

Extended Regular Expressions (EREs)

Allow more complex patterns than BREs

Specify two or more possible character sets by: |

To designate additional subexpressions: ()

Wc Command

Displays number of lines, words, bytes in file

Example Usage:

- wc -lL /proc/cpuinfo

Options:

- -l : Display the file's line count.
- -L : Display the byte count of the file's longest line.
- -m : Display the file's character count.

Lesson 1 Review

Grep can use both basic regular expressions and extended regular expressions

The * means 0 or more characters in basic regular expressions

The . means 1 character in basic regular expressions

Lesson 2

Redirect Standard In, Out and Error

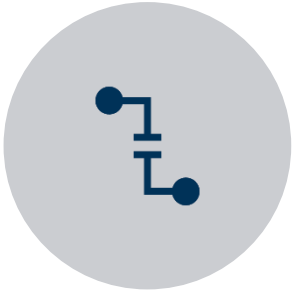
In this lesson, we the look at redirecting Standard Input, Standard Output and Standard Error

Standard Output

File descriptor that identifies output is 1 or STDOUT

STDOUT, by default, sent to current terminal (/dev/tty)

Redirection operators allow you to redirect STDOUT



Example: > redirection operator

Standard Error

File descriptor that identifies error is 2 or STDERR

STDERR, by default, sent to current terminal (/dev/tty)

Redirection operators allow you to redirect STDERR

Example: 2> redirection operator

Standard Input

File descriptor that identifies input is 0 or STDIN

STDIN, by default, received from keyboard or other input devices

Redirection operators allow you to redirect STDIN

Example: < redirection operator

Common Redirection Operators

Operator	Description
>	Redirect STDOUT to specified file. If file exists, overwrite it. If it does not exist, create it.
>>	Redirect STDOUT to specified file. If file exists, append to it. If it does not exist, create it.
2>	Redirect STDERR to specified file. If file exists, overwrite it. If it does not exist, create it.
2>>	Redirect STDERR to specified file. If file exists, append to it. If it does not exist, create it.
&>	Redirect STDOUT and STDERR to specified file. If file exists, overwrite it. If it does not exist, create it.
&>>	Redirect STDOUT and STDERR to specified file. If file exists, append to it. If it does not exist, create it.
<	Redirect STDIN from specified file into command.

Lesson 2 Review

STDIN is typically the keyboard and can be redirected

STDOUT is typically the console and can be redirected

STDERR is where error messages are sent

Lesson 3

Pipe & Filter

In this lesson, we look at how to pipe and filter output together to create advanced automations.

Piping Commands

Redirect STDOUT, STDIN, and STDERR between multiple commands on one command line

Syntax:

```
Command1 | Command2 [| ...]
```

Tee Command

Redirects STDOUT to current terminal and a file

Example Usage:

- wc -l file1.txt | tee -a file2.txt

Options:

- -a : Do not overwrite the files instead append to the given files (append).
- -i: Ignore interrupt signals.

Creating Here Documents

Allows the redirection of multiple items into a command

Redirection operator:
<< followed by end-of-data keyword

Example:

```
-cat <<ADDTEXT
This text is
added by Here Document
ADDTEXT
```

Lesson 3 Review

Piping takes the output of one command and sends as input to another command

The Tee command sends STDOUT to both terminal and a file

A Here Document allows redirection of multiple items into a command

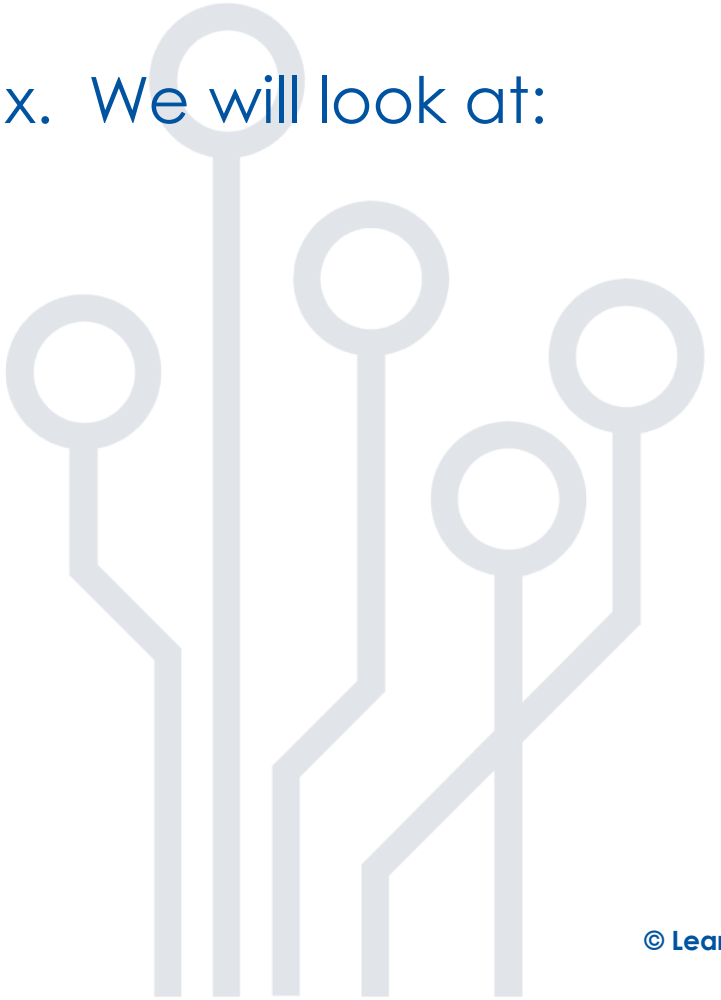
Lesson 4

Editing Text Files

In this lesson, we look at how to use a text editor to modify a file

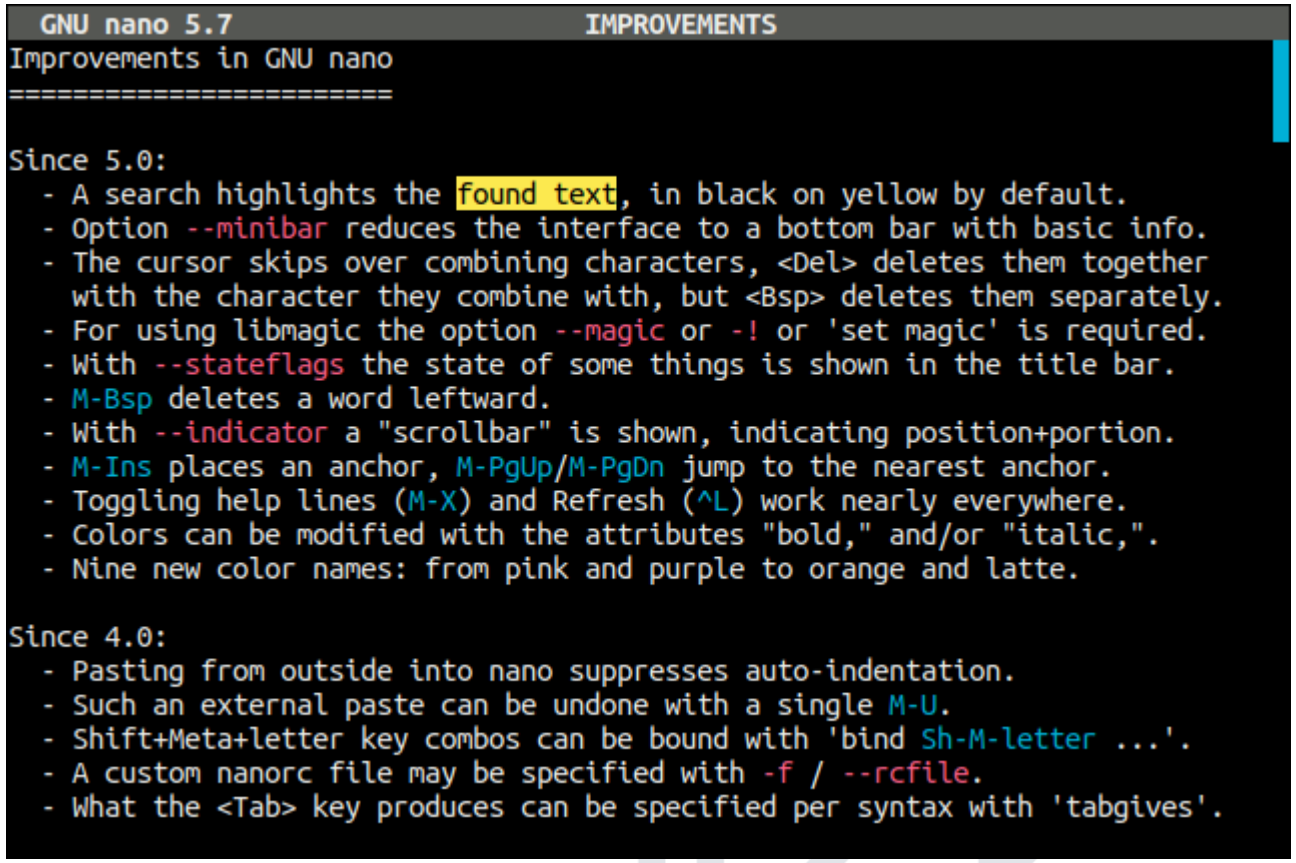
Text Editors

- Text Editors allow you to edit a text file without storing formatting inside the text file.
- There are many text editors available for Linux. We will look at:
  - Nano
  - Vim



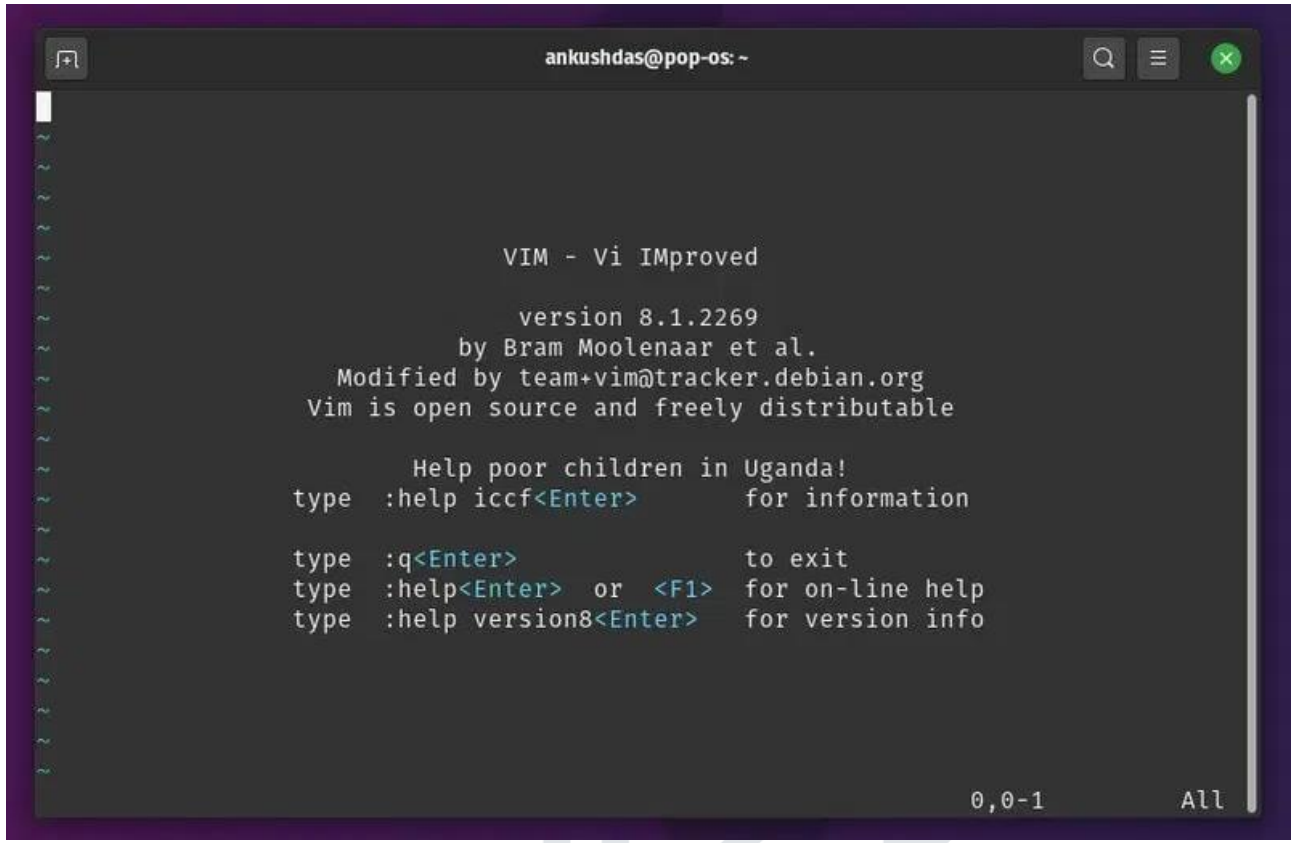
Nano

- GNU nano was designed to be a free replacement for the Pico text editor
- Usage:
  - nano file-name
- Four main sections:
  - Title Bar
  - Main Body
  - Status Bar
  - Shortcut List



VIM

- Vim is a highly configurable text editor. It is included as "vi" with most Linux distros.
- Vim is stable and is continuously being developed. Features include:
  - persistent, multi-level undo tree
  - extensive plugin system
  - support for hundreds of programming languages and file formats
  - powerful search and replace
  - integrates with many tools



Vim Usage

Two ways to start:

vim file-name
vi file-name

Three modes:

Command
Insert
Ex

Stream Editors

Allows you to edit text files without having to use a text editor.

A stream editor modifies text that is passed to it via a file or output from a pipeline.

The editor uses special commands to make text changes as the text "streams" through the editor utility.

SED

AWK or GAWK

Sed Command

Reads one text line at a time from input stream, matches text with supplied editor commands, modifies text as specified in commands and then outputs modified text to STDOUT

Example Usage:

- sed -e 's/aspen/seamus/ ; s/olmsted/moran-olmsted/' infile.txt

Options:

- -e : Add commands in script to text processing. The script is written as part of the sed command.
- -f : Add commands in script to text processing. The script is a file.

Gawk Command

Stands for GNU Awk. Four main features:

- Define variables to store data.
- Use arithmetic and string operators to work on data.
- Use programming structures, such as loops, to add logic to your processing.
- Create formatted reports from data.

Variables

- The $0 variable represents the entire text line.
- The $1 variable represents the text line's first data field.
- The $2 variable represents the text line's second data field.
- The $n variable represents the text line's nth data field.

Example Usage:

- gawk '{if ($4 == "aspen.") {$4="seamus"; print $0}}' cake.txt

Lesson 4 Review

Vim is the vi implementation on most Linux Distros

Sed process lines of text and runs commands

The Gawk Command has many features of programming languages