Lab# 03

Grep, Redirection, and Piping

CS211L

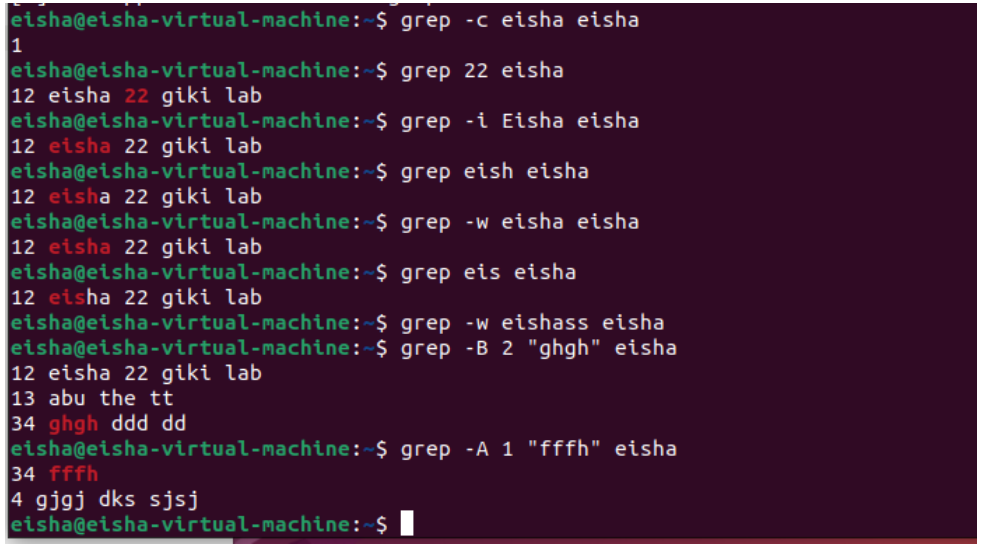
Lab Engr. Eisha ter raazia Mir

Learning Objectives

- To use the grep command
- To know the standard input and output
- To know about the I/O redirection operators
- To know how the output of the command will be the input for others using the pipe operator

Linux Commands

- **Global Regular Expression Parser (grep):**
  - ***grep*** The grep command is used to search for strings in a text file
  - **grep [options] pattern [files]:**
    - **-c** Print the number of matches only
    - **-i** Ignore the case
    - **-l** Print only filenames containing the pattern
    - **-n** Precede each matched line with its line number in the file
    - **-v** Print line NOT containing the pattern
    - **-B** 5 Display the context - i.e., 5 lines of context before a match
    - **-A** 3 Display the context - 3 lines of context after a match
    - **-w** Select only those lines containing matches that form whole words



Regular Expressions

- Regular expressions match a character string within the text (e.g. a file) you want to match
- i.e, a character string '**cool**' matches within the text '**in the cool nights of winter**
- The character string to be matched is called a **pattern**
- Associating (matching) this pattern with a text is called pattern matching
- Pattern matching - through regular expressions
- To see the power of the grep command, you shall use what we call "Regular Expressions"

Regular Expressions

- **.**
  - Matches any **single** text character
- ***
  - Matches **zero or more** preceding characters
- ****
  - Turns off any special meaning of the character following
- **^**
  - It matches the beginning of a line if used at the beginning of a regular expression
- **$**
  - It matches the end of the line; it is used at the end of a regular expression
- **[***str***]**
  - Matches any single character in *str*
- **[^***str* **]**
  - Matches any single character not in *str*
- **[**a - b**]**
  - Matches any character between *a* and *b*

Regular Expressions

- **‘h.t’** Example matches: "hat", "hot", "h&t", etc.
- **ho*t** Example matches: "ht", "hot", "hoot", "hooot", etc.
- **^The** Example matches: "The", "There", "They", etc.
- **ed$** Example matches: "liked", "jumped", "played", etc.
- **grep '3\.14' file** Example matches: 3.14 literally, not “3 followed by any characters 14
- **^The** Example matches: The, There, They, etc.
- **ed$** Example matches: liked, jumped, played, etc.
- **b[aeiou]t** Example matches: bat, bet, bit, bot, but.
- **b[^aeiou]t** Example matches: bxt, b1t, b$t (but not bat, bet, etc.).

grep -n '^[a-f]' test

```
eisha@eisha-virtual-machine:~$ grep -n '^[a-f]' file
1:aple
2:ball
4:fish
eisha@eisha-virtual-machine:~$
```

Difference b/w Wildcards and Regular Expressions

- Regular expressions and wildcards are both used for ***pattern matching***
- ***Wildcards*** are used for matching ***file names*** with file commands like **ls** etc
- ***Regular expressions*** are used for matching patterns ***within text_files***

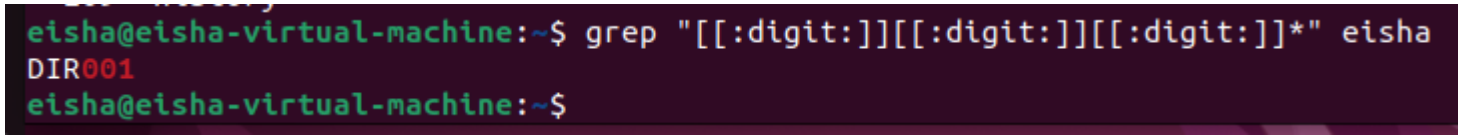
Character Classes

- Characters may be letters, numbers, punctuation marks, etc
- Use with ***grep*** command
- The general form of character classes:
  - **[: classname:]**
    - Following are the character classes:
    - [:alnum:] letters and digits
    - [:alpha:] letters
    - [:blank:] space and TAB
    - [:digit:] digits
    - [:cntrl:] all control letters
    - [:lower:] lower case letters
    - [:punct:] punctuation marks
    - [:upper:] upper-case letters
    - [:xdigit:] hexadecimal digits (0-9,A-F, a-f)

Character Classes

- ***grep “[[:alpha:]] ..” filename (means any two characters)***

- ***grep "[[:digit:]][[:digit:]][[:digit:]]*” eisha***



Streams

- Streams are associated with the flow of data
- Input flowing from an input device, like a keyboard to memory, is referred to as an input stream
- Output flowing from memory to an output device is referred to as an output stream
- When a Linux command executes, it may take input before execution and gives some results after execution
- Therefore, each command in Linux is associated with streams

Streams

- Streams associated with the execution of Linux commands are:
  - Standard Input Stream (Stdin)
  - Standard Output Stream (Stdout)
  - Standard error Stream (Stderr)
- Standard input flows from the keyboard to the memory
- Standard output and standard error flow to the terminal
- In case of errors, error messages flow as output from the computer to the terminal, called the **standard error stream**

I/O Redirection

- Redirection changes the assignments for standard input and standard output
- Usually, standard input comes from the keyboard, and standard output goes to the screen
- The term **redirection** – when standard input not coming from keyboard but from other sources like a file
- Standard output will not be going to the screen but to other sources like a file or other commands
- In the case of I/O redirection, the shell should be informed


stdout
stdin

I/O Redirection

- The following characters are used to notify the shell to redirect the input or output of any command:
  - >
    - Redirects output of a command to a file or device (e.g., printer). It overwrites the existing file. E.g **ls > file.txt**
  - >>
    - It is similar to **>**, except if the target file already exists, the new output is appended to its end. E.g **echo "Hello" >> file.txt**
  - <
    - Redirects input of command from a file or device. **sort < file.txt**
  - |
    - Sends the output of one command to become the input of another command.
    - E.g **ls | grep “eisha"**

Output Redirection

- Usually, the output is sent to the screen
- Output redirection can also be sent to other sources, like a file
- The symbol ***'>'*** is used with a command to redirect output
- We can ***overwrite*** the standard output using the ***'>'*** symbol

command > filename

date > Ali

cat Ali

Input Redirection

- Usually, the input is taken from the keyboard
- Input can be taken from another source, like a file
- The symbol ***'<'*** is used for redirecting input
- The standard input may be received from a file rather than the keyboard

command < filename

wc -l < users

Pipe Operator (|)

- The standard input and output are redirected to some sources other than default sources
- The **pipe** command sends the output of one Linux command to another Linux command
- The **pipe (|) operator** (vertical bar) is placed between the two commands and forms a connection between them
- The pipe operation receives output from the command placed before the pipe operator and sends this output as an input command placed after the pipe operator

ls | grep “eisha"

cat file.txt|sort -rn | head -3

File Descriptors (FD)

- In Linux/Unix, everything is a file. Regular files, Directories, and even Devices are files
- Every File has an associated number called File Descriptor (FD)

File Descriptors (FD)

- Standard Input Stream (Stdin)
  - 0
- Standard Output Stream (Stdout)
  - 1
- Standard error Stream (Stderr)
  - 2

Process Management

- The process is a program in execution
- The process is created when a command is to be executed, so it can be called a running instance of a program in execution
- Tuning or controlling a process is called Process Management
- ***ps***
  - To show the current running processes in a system

Terminating a Process

- Linux provides a ***kill*** command to terminate the unwanted process
- The ***kill*** command sends a signal to the specified process
- A signal is an integer number; the process is identified by the process ID number (***PID***)

Terminating a Processes

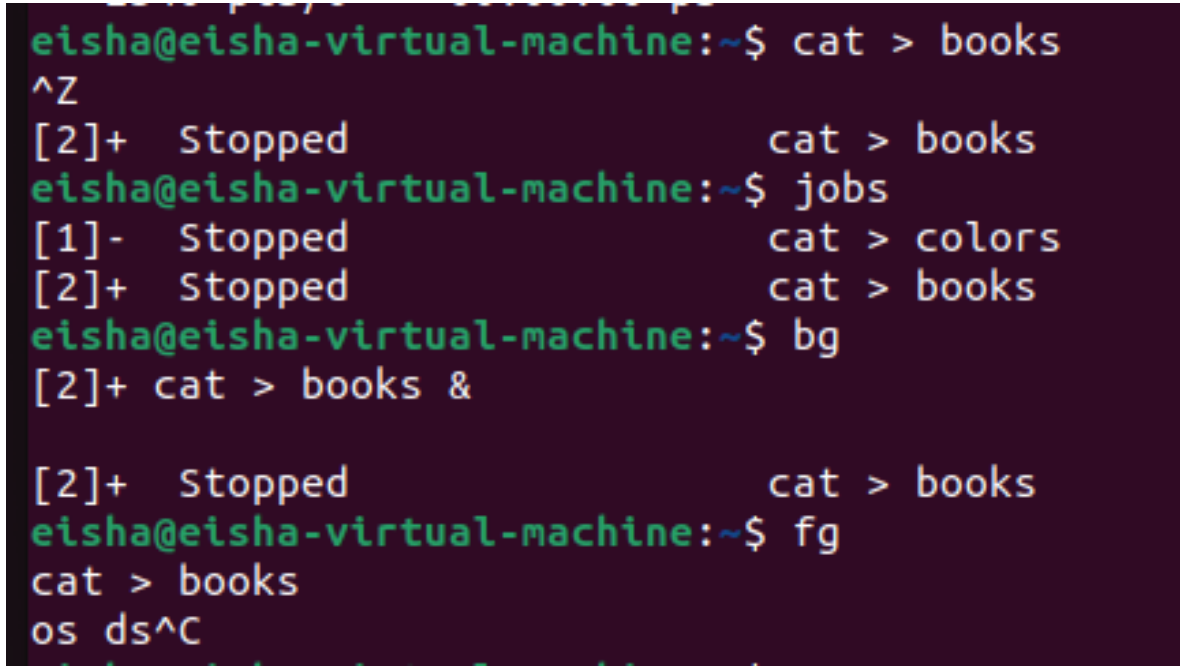
- ***top***
  - List of all the running processes on your Linux machine
- **$ kill [PID]**
- **$ kill 5432**
- **$ kill -9 5432**
- Kill the process having a specific PID number

Foreground and Background Processes

- Foreground processes:
  - Processes running in front
  - Can only be one
- Background processes:
  - Processes running in background
  - Several processes
- When to run a process in the background:
  - If a process takes a bit longer in its execution and does not require any user/interactive input

Moving Processes

- It is possible to move some of the programs to the background so that you can work on something else
  - ***cat>file_name***
  - ***Ctrl+z***
- Background processes
  - ***Jobs***
- Foreground to background
  - ***bg***
- Back to foreground
  - ***fg***



Ready

Files:

OS -> (‘write your name here’), lab3, lab1, hello, Lab3 is in progress, I am a student, 2024) (Hint: Each word with separated commas is on new line)

file1 -> (Hello ‘write your name’)

file2 -> (The only way to do great work is to love what you do)

file3 -> (Be kind)

Tasks # 1

a) Find the number of lines that start with either integers, capital or small letters in the file ***OS***
b) Find the lines that contain spaces in the file ***OS***
c) Make a file name ***lab3*** and put some information in it. Find all the lines that start with vowels
d) Find all the lines not containing the word ***kind*** case insensitively
e) Display 2 lines just before the statement "***I am a student***"

Task # 2

a. Store the data of ***file1, file2, file3*** in a file name ***Newfile*** containing nothing without overwritten
b. Echo ***Hello World*** in a file name ***newfile1***
c. Count the number of times your name is shown in a file name ***Newfile*** just created in part a of this task which is used as input for a file named ***redirection_Example***
d. Find your name in file file1, file3 and file4 and put the valid output in output.txt and error messages in error.txt