Bash Cheat Sheet

Presenter

Duy-Thanh Nguyen

30/05/2024

Bash Cheat Sheet

- Linux command line
- Bash Scripts
- SED and AWK

Linux command line

	Class	Example	Description
	Command History	history	Display the commands history list with line numbers
Directories	Creating Directories	mkdir *dir_name*	Create a directory
Directories	Navigating Directories	cd ~	Go to home directory
Directories	Moving Directories	mv *source destination*	Move directory
Directories	Deleting Directories	rm -r *dir_name*	Delete directory including contents
Files	Creating Files	touch *{file1,file2}.txt*	Create multiple files
Files	Moving Files	mv *file1.txt file2.txt*	Move file
Files	Deleting Files	rm *file.txt*	Delete file
Files	Read Files	cat *file.txt*	Print all contents

Bash Scripts

Terminology	Definition	Example
Variables	An element used to store values (which can be strings, numbers, or other types of data)	**#!/bin/bash** var=123
Environment variables	These are key-value pairs stored in the operating system environment and accessed by shells or running programs.	**#!/bin/bash** echo $USER

Functions	Functions"are blocks of code that can be called multiple times within a script. They allow for code reuse and for organizing code in a structured and tidy manner	**#!/bin/bash**
		greet() {
		local world="World"
		echo "$1 $world"
		}
		
		greet "Hello"

Exit Codes	Exit codes in a bash script are numerical values returned by a command or script when it finishes	#!/bin/bash exit 0 exit 1

Bash Scripts

Terminology	Definition	Example

Conditional Statements	Conditional Statements are commands used to execute different actions based on given conditions, typically employed to check the values of variables, the results of commands, or any other logical conditions	#!/bin/bash
		
		if [[$var = "version"]]; then
		  echo "one"
		elif [[$var = "vision"]] || [[$var = "baz"]]; then
		  echo "two"
		elif [[$var = "ban"]] && [[$USER = "thanhnguyen"]];
		then
		  echo "three"
		else
		  echo "four"
		fi

SED and AWK *(AWK)*

1. Strong with structured data: awk allows you to handle structured data, such as CSV files or log files with columns separated by commas or spaces.

2. Easy access to columns and rows: awk provides easy-to-use syntax for accessing and manipulating columns and rows in text files.

Basic Format

```
awk 'pattern {action}' file
```

Print & Search a specific pattern

```
# Written as one line.
awk 'BEGIN { print "Beginning this command"; } { print $2, $NF; } END { print "End of this command"; }' file_name.
txt
```

Numbering & Calculations

```
# Print the total number of lines that contain "SRR2326647"
awk '/SRR2326647/{n++}; END {print n+0}' file_name
```

Delete And Substitution

```
# Use "tr" to delete character "S"
tr -d \S < file_name

# Substitute (find and replace) "SRR" with "SRA" on each line
awk '{sub(/SRR/,"SRA")}; 1' file_name          # replace only 1st instance
```

SED and AWK *(SED)*

1. Simpler tasks: sed is often used for simpler tasks such as text replacement or line-by-line processing.

2. Shorter syntax: For basic tasks like text replacement or simple processing, sed can provide shorter syntax compared to awk, especially in cases where no complex processing or parsing of each line is required.

```
# SYNTAX
sed [options] command [input-file]
```

Sed substitute command and flags

```
# g flag - Global substitution
sed 's/Windows/Linux/g' file_name

# 1,2.. flag Substitute the nth (2th) occurrence
sed 's/locate/find/2' file_name
```

Sed commands

```
# p Print pattern space
sed -n '1,4 p' file_name

# d Delete lines
sed -n '1,4 d' file_name
```