Part 2: Useful commands

# Master your Command Line

## (Before it masters you)

Tejas Sanap

30th June, 2019

1 Introduction

2 Search
- Text
- grep
- Files

3 Manipulate
- Text
- Files

4 Shell porn

5 Questions

# Introduction

# UNIX Philosophy

Focused on *modularity & reusability*.

It can be summarized as:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

# Basic Operations

All operations performed in the terminal can be categorized as:

- Search for text (in files).
  - `grep, cat`
- Search for files (in directories).
  - `find, locate`
- Manipulate text (in files).
  - `sed, awk, cut`
- Manipulate files (in directories).
  - `cp, scp, rm, mv`
  - `gzip, tar`
- Manipulate file permission and ownership.

# GNU Coreutils

The GNU Core Utilities are the basic file, shell and text manipulation utilities of the GNU operating system.
They are expected to be present on every operating system.
Previously, the core utilities were implemented by the following pacakages:

1. `fileutils`
2. `shellutils`
3. `textutils`

In 2003, these three packages were combined into the current `coreutils` package.

# Search

# Text editors

Do we really need Vim?

Text

# cat

cat

# grep

`grep` prints line that matches a certain pattern.

## Syntax

```
grep OPTIONS PATTERN INPUT_FILE_NAMES
```

## Example

Introduction Search Manipulate Shell porn

=

# grep

The exit status of `grep` when:

- **line is selected** is **0**.
- **no line is selected** is **1**.
- **an error occurs** is **2**.

Useful `grep` options:

| | |
|---|---|
| `-i` | ignore case |
| `-v` | invert matches |
| `-c` | count no. of matching lines |
| `-n` | prefic each line with line number |
| `-l` | print name of the file and suppress all other output |
| `-H` | print filename for each match |
| `-o` | print only the matched parts of a line |
| `-s` | suppress error messages |
| `--color` | color the matching content |
| `-a` | accept binary input |
| `--label=LABEL` | display input actually coming from `stdin` as input from file `LABEL` |

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xf python_code.tar.gz
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep
main'
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a main'
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H main'
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -n main'
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -c main'
```

# grep

## Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

## Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -c -s main'
```

# Files

# find

find

# `locate`

locate

# Manipulate

# File Manager

Do we really need Nautilus, Thunar or Ranger?

Text

# sed

sed

# awk

awk

# cut

cur

# Files

# scp

- download `znc.pem` from server to add to irssi client

# rm, cp & mv

- Text globbing - Use latex compile files and stuff as examples
- Bash Pattern Matching

```
rm pre*.!(tex)
```

# gzip, tar

# Shell porn

# fasd

fasd

# fortune & cowsay

- Let's add some star trek quotes
- Cowthink and cowsay
- Add some bling with pony

# Questions

# Questions?

?

# References

```
bibliography
```