Unix, Linux & Virtual Machines

UNIX History

Unix was originally developed for internal use @ AT&T by Ken Thompson and Dennis Ritchie

First version created in Bell Labs – 1969

Unix flavors are	AIX from IBM
	HP-UX from Hewlett Packard
	SunOs from Sun
	IRIX from SGI

UNIX Principles

- **Everything is a file-including hardware**

  Secure access to hardware as secure access to docs

- **Configuration data stored in text**

  Admins can easily move configurations to other machines

- **Small, single-purpose programs**

  Many small utilities that perform one task very well

- **Avoid captive UI**

  Options and arguments typed on the command line

- **Ability to chain programs for complex tasks**

  Output of a program can be input for another

GNU Project/GPL

- GNU project started in 1984
- GOAL: **CREATE a FREE clone of UNIX**
- **Free Software Foundation**
  Does not refer to the cost of the software, but the fact that the end user has the free to modify and change the program
- **GPL-GNU General Public License**
  Primary License for Open Source software
  Encourage free software

Linux Origins

- **Linus Torvalds**
  - Finnish college student in 1991
  - Created Linux Kernel

- **Linux kernel + GNU applications = complete free UNIX - OS**

Linux Principles

- **Fresh implementation of UNIX APIs**
- **Open source development model**
- **Multi-user and Multi-tasking**

  Many users can be logged on to the same Linux computer at the same time,

  and can have more than one process at the same time.
- **Supports wide variety of hardware**

  Supports most piece of modern x86-Compatible PC hardware

Date Time and Calendar

- **date**

  display date and time

- **cal**

  prints an ASCII character of the current month

- **man <command>**

  displays pages from reference manual

File Information

- File names may be up to 255 characters
- File names are case-sensitive
- Files and directories on Linux system can be named by any combination of letters, digits and (most) punctuation symbols
- **pwd** displays the absolute path to the current directory
- Locations can be specified in ways:
  - **ABSOLUTE PATH:** absolute path starts with **/**

/home/pippo/Desktop/file.txt complete road map to a location

  - **RELATIVE PATH:** relative path do not begin with **/**

~/Desktop/file.txt, ./../Desktop/file.txt location relative to the current directory

Changing & Listing Directories

- **cd** change directory

cd /to/absolute/path	To absolute path
cd ..	One level up
cd or cd ~	To home directory
cd -	To your previous working directory

- **ls** listing directory contents

ls	list the contents of the directory
ls -l	long listing
ls -a	listing also hidden directories
ls -R	recursive through subdirectories

System Directories

/bin, /usr/bin	User commands
/sbin, /usr/sbin	Administrator commands
/var	Logs, PID files, mail
/proc	“Virtual window” into the kernel
/etc	Configuration files
/lib	Shared libraries
/dev	Device files
/boot	Linux kernel and boot files
/home	User’s home directories
/opt	Third party packages

Checking Free Space

- **df** Reports filesystem disk space usage

df -h	displays filesystem information in human readable format

- **du** Estimated file space usage

du -h	displays file space usage in human readable format
du -s	summarizes the space in a directory

Copy & Move Files and Directories

- **cp** Copy files and directories

  Syntax:

  ```
  cp [options] source_file destination_file
  cp [options] source_1 source_2 … source_N destination_path
  ```

cp -r	Recursive copy
cp -p	preserve time and date information when making a copy
cp -f	forceful copy of file to destination file

- **mv** Move files and directories

- mv and cp are identical, but cp results in matching identical files, while with mv the source disappears, leaving only the destination files

Create & Remove Files and Directories

touch	create an empty file or update file with timestamp
mkdir	create a directory
mkdir –p	creates the full path with the intermediate directories
rmdir	remove an empty directory
rm	remove files
rm -i	interactive
rm -r	recursive
rm -f	force

View a File

• **cat**	contents are displayed sequentially with no break
• **less**	displays the content of a text file one screen at a time
• **tail**	displays last few lines of text in a file
• **head**	displays first few lines of text in a file

Redirecting Input and Output

- Standard output, usually displayed on the terminal, can be redirected into a file or into another command
- Standard error, usually displayed on the terminal, can be redirected to a file
- Standard input, ordinarily coming from the keyboard, can be redirected from a file

• **command > file**	Directs standard output of command to file
• **command >> file**	Appends standard output of command to file
• **command < file**	Command receives its input from file
• **command 2> file**	Command errors are redirected to file
• **command 2>> file**	Command errors are appended to file
• **command1 | command2**	Pipes the standard output of command1 into the standard input of command2

String Processing Commands

wc	Word count: count lines and characters
sort	Sorts data from a file or from the output of another command
uniq	Removes duplicate adjacent lines from a file
cut	Cut fields or columns of text from a file and display them to standard output

String Processing Commands

- **grep** (General Regular Expression Processor)

  displays the lines in a file that match a pattern

  Also used as filter in pipelines

  ex:
  ```
  ls -l /usr/bin/ | grep java
  ```

- **sed** Stream Editor

  Reads a file or stream of data, performing search and replace instructions

  ex:
  ```
  sed s/dog/cat/g pets.txt
  ```

- **awk** Manipulates text, can be programmed

  ex:
  ```
  awk -F '\t' '{print $1}' data.tsv
  ```

Exercise

- Go to https://simple.wikipedia.org/wiki/List_of_fruits
- Copy it into a file named fruits.txt
- Count the number of lines and output the value into lines.txt
- Sort the file lines randomly (option -R) and redirect the output into random.txt
- Filter (using grep) the fruits that contain the word 'berry', sort them, and save the output into berry.txt

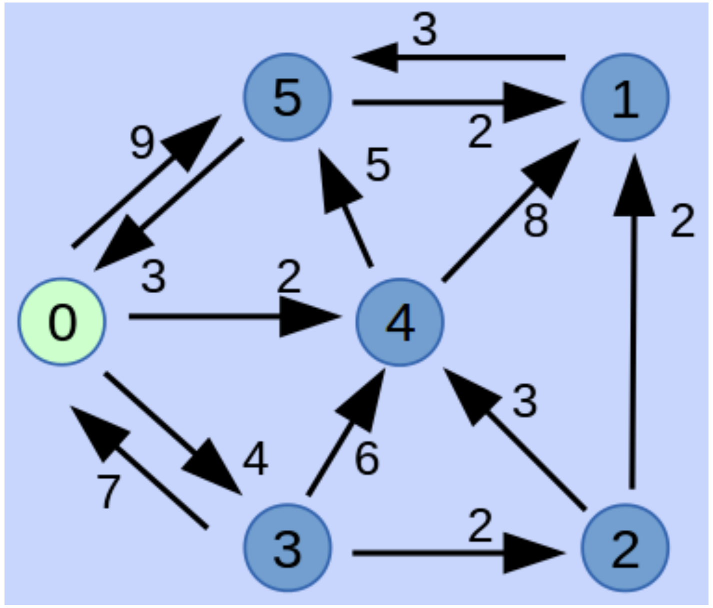
```
wc -l fruits.txt > lines.txt
sort -R fruits.txt > random.txt
cat random.txt | grep berry | sort > berry.txt
```

Exercise

- Goal: # of outgoing edges for each node

  wget disi.unitn.it/~foroni/graph.txt

  each line is an edge represented as:
  vertex_from vertex_to



- Select the first column (use awk) and count how many times there is an edge for each node (use a combination of sort and uniq)

```
awk -F' ' '{print $1}' graph.txt | sort |
uniq -c
```

Exercise

- Goal: Word Count

```
wget https://www.gutenberg.org/cache/epub/1112/pg1112.txt
```

- Replace with sed the spaces with a new line (-e $'s/ /\\n/g'), order the words, count the number of times it appears (with uniq), and sort them in decreasing order

```
sed -e $'s/ /\\n/g' romeo.txt | sort |
    uniq -c | sort -r | head -n 10
```

Contacts

For any problem, write me a mail:

daniele.foroni@unitn.it