Enhance (prev. program)

Linux

WebLinux

History of Linux & Command line:

window, macos, ios, Unix, Linux etc.

* Linux - open source - Linux kernel (created: Linus Torvalds) - developed by thousand of programmers

OS - Hardware & Apps (Intermediate)

provides services to apps

Services:

1) file management (layout) - Harddrive - manage local tree structure
2) Memory management
3) Management of I/P & O/P S
4) Manage of running apps

Operating system

1940's (mid) - first computers - Vaccum tubes - evacuated glass containers - control - Electric Current → (AS on/OFF switches) - Large

* Programming - done manually - by moving around tubes
* Limited → I/P & O/P
* Programmer - also in charge of the Operating the machine via direct interaction → "Heavy lifting"!

Designer - builder too!

Programmer as well as operator!

"Invention of Translators" → Beginning of O.S.

IBM punched Cards: Program cloth making machinery - holds data - represented by presence/absence of holes - in predefined positions
Card reader → Reads (Top - left) - vertically.

'Info - o/p as punch cards'

5Mb - 62500 - punched cards

Operating system - manage th is → memory, processes, Programs, i/p, o/p - reading etc..

mid-1960 - OS invented!

Unix genesis

* most adv. OS of its time - many modern Concepts still used
* IC, magnetic disks, Compatible Computers - IBM

IBM - 1964 - Announced clear distinction b/w architecture & implementation.

Started!

MAC project on mathematics & computation founded by MIT funded by US military research funding agency (ARPA) & the US National Science foundation.

Aim: Realize - time sharing system - allow wide community of users to simultaneously access hardware & software! resources of a single Computer from multiple locations.

MAC - Bell labs & General Electric

↓

Developed Multics

(multiplexed - info & computing service)

↓

No longer - just computer resource timesharing but evolved to also incorporate features such as

* File sharing
* file management
* Security.

Multiplexing:

Technique of sending multiple pieces of info over a comm. channel - same time - in the form of single complex signal.

↓

made possible - to share the same resource b/w several users.

---

* General electrics - (MULTICS project) - took - design & build the underlying machine.
* Bell Labs - design & writing of OS

Bell labs resigned - some group (Dennis Ritchie - left different path)

↓

Build minimal path System

"Single user OS - Bell labs doesn't backed!

↓

A pun of MULTICS called: UNICS

---

* 1970: Changed from single user to multiple users (Unix) morphed

* System: written in B language (Ken Thompson) - changes B way to C - Dennis Ritchie (Improved C - Unix upgraded)

* Opensource - sent to Universities

↓

Various versions of Unix!

(various machine built by others)

* Ken & Dennis: received Dennis (Turning Award)!
* Mac, Android, even Linux — "Basics of Unix"

---

* 1980 - Unix - widely adopted by univ & research - many startup & companies.
* 1983: Turing award: Ken thompson & Dennis Ritchie

Same year: Richard stallman — MIT launched

"GNU project"!

* GNU - Recursive acronym - referenced to itself
* GNU - open source, free, collaborative - provide software compatible with Unix.

* Unix - owned by Bell (sells license)
* GNU - Free - Anyone can modify

1990 - GNU already - lot eg software → Text editors, graphical user interface (windows)
GCC (C compilers)

Problem: use the free software on the Unix OS - (Unix - proprietary not open source)

Change! → Linus Torvalds - 21 years old Finnish CS Student
↓
Frustrated - (Felt limited)
↓
messaged community - free OS (he is dev)
↓
Hobby - Not so professional
(Later become the Linux Kernel)
(Independent eg OS)

* Meant to run on Linus Torvalds - PC - 83.86 processor.
* Development - done using GNU C compiler - main choice (still) compiling Linux.

'Community formed'
↓
* Free Software (dev) to run on Linux Kernal

1992 (First Linux distribution) → OS, Collection eg free GNU (editor, compiler, library etc..)

90s → Dell, IBM, HP - Announced compatibility with Linux - Runs web servers - (70%. eg market share) - 90%. (Android OS based on Linux).

* 'Dominating super computers'
* Film & special effects - 99% share

'Evolved!'

CLI - Command Line Interface

* Text mode: Communication b/w user & computer
* Execute - other apps.

me@linuxbox → user / Computer / current directory name

---

* Command options files_or_data
* os command / app command.

  * Cal → Calendar in text model
  * Cal -3 → In julian days.

Commands

* whoami → username
* whoami --help → gives help
* id → userid (UID), (GID), groups
* logname --help → gives help

---

* logname → prints current user
* echo hello → print hello
* echo $0 → print sh (Not: $0)

  Why? → Name: sh (Shell) → Interpreting Commands on linux (App)

* hostname --help → who is the host

  → $ hostname → Name of the Computer

* uname → Name of the system (-a → all info about system)
* history
* (uparrow) → Previous Commands.

---

* clear → clear the cmd

* uptime → (when system booted)!

* cal → displays calendar (-3 → Julian, -y → entire year)

* date → today date. [can change formats]

---

date + "%A %d %B %Y"

↓ Thursday | ↓ 13 | ↓ September | ↓ 2018

Real Linux

man - manual of command

* man cal → long manual.

---

*

Interactive Commands

top, htop, nano, vim, ge

* top → CPU usage (memory usage, load) - whats happening in system!

*

Back to prompt: Q (or) ctrl+c

---

* htop → more desined (graphical) [F10: quit]

* nano → file editor

---

* vim → dificult to exit often!

q <Enter> exit

esc → (go to menu)

↓

:q!

to exit

Play with Commands - Not in Real linux

hello → hello world

worm → (simulation in text : game) - worms moving.

fireworks → (simulation - fire works).

rain → (simulation)

hanoi → (towers of hanoi) - we can play.

→ some moves forbidden.

from 1 to 2
from 1 to 3
) play.

Knight → logistery solitaire (game).

filesystem	/bin	/root	/tmp	/dev
history	/sbin	/etc	/var	
FSSTND	/home	/lib	/usr	

File systems (management)

each cmd: 80 characters

'tree' - directories (Functions & Functions)

Hierarchy

Root → /

/folder/subfolder/file

1993 - standardized - Filesystem hierarchy to GNU Linux (

* /bin → directory having basic commands - to start & use a minimal system. (Binary → .exe file)

* /sbin → root user (administrator)

* /home → All eg users directory

* /root → home of the root user.

* /etc → config files (editable text configurable)

* /lib → shares software libraries - prev dos .exe files.

* /tmp → temp files (/var/tmp or /run/sub)

↳ emptied (restart) → Be careful!

* /var → various files used by OS (DB, email, history), logs

*/usr → unix system resources → subfolders, (extend system operation.

* /usr/bin → .exe files - not already present in /bin

eNot must for minimal system

/dev → files - corresponds directly/indirectly to a physical device

(dev - device)

Strength of Linux

* Considers everything as files - including devices.

/dev/printers
/dev/audio
/dev/mem
/dev/networkCard

Navigate

* pwd → print working directory. /home/users
* whoami → users
* ls → list of files in working directory.
*

Change directory

* cd /
(root directory)
* pwd
/ → Root

---

* cd /home
/home
* /home $ ls
users
* /home $ cd users
$ pwd
/home/users.

To home directory

* cd
- $ pwd
/home/users

Relative path

* ls -a (Show hidden files)
* .. → parent directory
* cd .. → go to //
* cd ../..
Parent of parent!
* cd ./sys (or) * cd sys
(same directory)

Absolute path

cd /sys
↳ Absolute path
(start with root) - not with ..

cd ../home/user → Relative

↓ Relative — ↓ Then go to user.

→ Absolute.

cd /sys

cd../../.. → parent of Parent of parent

* ls -al → detailed view

pwd, cd, ls, absolute path, relative path

Type of a file

* file program.c

c: Source, ASCII text

* nano f → create a new file (edit)

* ^O → Save file (See menu): write out

↳ In current directory.

* file f

f: ASCII text

* realpath f

/home/user/f

* cd /

* file bin

↳ directory.

'Some of the files: link to other files' → See path (gives original path: pointing)

* which realpath [path of realpath]

/usr/bin/realpath

* which /cat

/bin/cat

↓

path of binaries (use which)

* Torch

* rm

* names with spaces

Create & delete files

* ~~touch~~ → Create empty file (touch file)
  ↳ Name
* rm f → delete

files name with space

$ touch my file
2 files created

'Use backslash'

my\ file → my file created

rm my\ file

---

or

* touch 'my file'
* rm 'my file'

$ rm '
>
>
> → Stucked!

'started to write a name'

Anything - past of the name! (even Enter)

Finish: ' close single quotes.

{ | put ' (or) } what
> | you started!
> |
} | (or)
Ctrl + D (If no! → Enter then ^D)

Cat less - Read files

* nano 'file.txt'

» 'write something'

Read - cat command

>> cat file.txt

>> echo "hello world" → hello world

↓ redirect to a file.

>> Echo "hello world" > file2.txt → saved to this file

>> cat file2.txt → read.

>> cat file2.txt > file3.txt → o/p of cat to file3.txt

↓

"copy files"

---

cat > another file.txt (No file as i/p) → Editing mode

this is me!

ctrl + D → written.

---

* cat /etc/services → has all the services of the OS

Linux: no scroll : redirect to a file → then see

* cat /etc/services > list.txt

In command line

$ cat /etc/services | more

(give something to i/p) → more (display more)

↳ long file (ls)

---

* cat /etc/services | less → std. i/p → we can now scroll.

Advanced command

Space bar - next page.

Lowercase g → beginning
G → end
) of long file in cmd.

---

display using more (instead of cat)

* more another file.txt (we can use space).

* less text.txt → display file.

search

$ ls /usr/bin | less (we can scroll all the files)

$ /cut (to search)
pattern

: n (next match

/ pattern
: n → next match

prev match: N (uppercase)

Backward Search

? pattern

: n → next match

N → prev match

$ mkdir folder → make directory

$ cd folder → go to folder

remove folder

$ rm folder → 'folder has files'

Need to do it recursively.

$ rm -r folder → delete directory!

$ rm -ri folder (ASK whether to delete? each file)

$ yes (Put yes until stopped)

y
y
y
:
:

(o/p yes all the time)

ctrl + c

$ yes | rm -ri folder

'yes to all queries'

pipeline

* mkdir d1 d2 d3 → 3 folders

* mkdir -p D1/D2/D3/D4/D5

'we can't create directories with \ command'

↳ Use of -p make D5 inside D4
D4 // D3
:
D2 // D1

$rm -rf D1 → 'del d1 means del d2, d3, d4 & d5'

'careful: using rm'

Kill the program

* htop → all the programmes running
* F9 (Kill)

---

mv (move)

$ mv f1 d1 (move f1 to d1)

Rename

$ mv bigdir smalldir (Renamed! If file not found)

Rename (already exist)

$ mv f5 f6 'f5 → lost' forever.

∴ Caution: If same name — lost!

replace existing files.

-f → Don't prompt (warn)

-i → Interactive

-n → Don't overwrite

CP, CP -r — copy

$ cp f1 f2 (Copy f1 to f2)

$ cat f1 > f2 (Same as copy)

Copy all files of one folder to other

$ cp -r d1 d2 (do recursively)

Already existed:

create d1 inside (copy) d2

Again doing

d1
d2
d1
f1
f1

∴ d1 inside d2 → replaced.

Real: locate (uses big database) | Find locate

updatedb → update database.

use echo

$ echo p* (expand all files starting with p)

$ echo /usr/bin/a*

In bin, show all files with a (starting).

Starting with a & 3 letters

a??

??? → 3 letters long

$ find / → list all directories & files under it

search for a file

$ find . -name "program.c"

space

$ find . -name "prog"

'starting with prog' (use double quotes)

Get rid of error messages

$ find / -name "hello" 2> file.txt

↓ errors messages

↓ write to a file.

/dev/null – blackhole / bin – lost dogs even

$ find / -name "hello" 2 > /dev/null

---

joe → Search before & after any characters -joe

find / -ipath "*joe*" 2 > dev/null

↓

ignore lower case

Interpreteur - read a line → executes

*Slow - wait for each command to be interpreted (machine)

* we can correct - each line instantly. (Stay with you all the time)

Compiler - whole (translates) - No waiting!

Interpreteur	Compiler
Runs slowly	* Extra preparation time
Skips KB sights away	* Then runs program very quickly
Lets you see results	

---

* Inter: (B/w) - Translates line by line.

* Compile: Together.

'compile : C program on web linux'

program.c

Compile → gcc -std=c11 -Wall -fmax-errors=10 -Wextra program.c -o program

exec ./program

Type of file cp → copy.

* file program.c

Compiling using Gcc

* gcc program.c → creates a.out

* ./a.out

o/p name - as per my request

* gcc program.c -o program
* ./program

Hello world!

---

* gcc -std=c11 -Wall

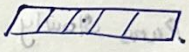
Memory

RAM (during execution) ← → ROM

* word - several bits
* Byte = 8 bits. (mine: 64 bits: For 1 word)

why group? - Address!

Say: 8bit: (4 words: 32bits) - 4 address



In c: we can access address. - low level access

'memory optimization'

---

manage memory

* free [-b/k/m/g]
  - b → byte
  - g → gigabyte.
  - → shows memory usage
* top → All running programs (we can q it, sort etc..)
* htop → visual representation
  - F6 - sort → many options.
  - M → sort
  - St → showing memory - alone.

---

printf("Enter any key to exit: ");
scanf("%c", &c);

Virtual memory: different from physical memory.

'Computer - used', not physically used

eg: char arr[100000]; → mostly empty (not used).

'System Optimizing memory' → (Virtual memory (not directly impacted by physical memory)

Rebook

'Locked in cmd - (without i/p) - tried to run' → Refresh (web linux)

```
gcc program.c | program
./program
```

'Run c program'

Scanf, &

Problem: Need to display printf → In new lines \n before using scanf → else

remi
18
Sheldon
What is family? Name: ? Age: ? → printed at the same time

Reason!

Buffered - Buffer ends - when newline \n in printf.

Force to print - even no \n → Flush the buffer

fflush(stdout); → After each printf

Some linux: no fflush needed: default (Behaviour). - Flush automatically

---

Redirect to other file (or) command

* echo 'hello' > file.txt
* echo 'hello' | less

$ cat > answers

Shaarnock
rem?
18

$ cat answers
Shaarnock
rem?
18

• ./program < answers
(i/p gn as file)

Don't → use scanf { use fgets, get line or readline }

* Scanf : dangerous → whenever space → Split the i/p
*

Name : Shaarnock rem? 18
Age :
Family :
Shaarnock rem? 18

---

use: fgets → all spaces & newline characters.
getline → (not compatible on all systems)
gnureadline → only for linux

---

libraries in c

declare & define functions:

* define function (whole code)
* declare: prototype (just).

math library

* cd /usr/lib
* ls (many libraries) | less

libm.a

$ cd /usr/include
$ ls | less

} all libraries include

* math.h → headers

```
cp math.h /home/user
```

copied

---

modularity

touch weatherstats.c

program code

⋮

Source code

weatherstats.c → file

```
#include <stdio.h>
int main () {
    double temp [7] = {6, 9, 12.3, 9, 5.3,
                       9.8, 1.8,
                       0.3};
    double average = averageTemp (temp, 7);
    printf ("Average : %.2lf \n", average);
    return 0;
}
```

program.c

header file

```
double averageTemp(
    double *temps, int
    numofTemps);
```

program.c

```
#include <stdio.h>
int main (void) {
    do.
```

---

see-hub

'modularity' – more detail.

```
# define the C compiler to use
CC = gcc
⋮
```

see hub.

```
# define the C compiler to use
CC = gcc
# define compiler flags
CFLAGS = -std=c11 -Wall -fmax-errors=10 -Wextra
# define library paths in addition to /usr/lib
LFLAGS =                    → Now (nothing: no libraries used)
# define libraries to use
LIBS =
# define the object files that this project needs
OBJFILES = program.o weatherStats.o
# define the name of the exe file
MAIN = program
# all of the below is generic - one typically only adjust the above
all: $(MAIN)                → target
$(MAIN): $(OBJFILES)
        $(CC) $(CFLAGS) -o $(MAIN) $(OBJFILES)
%.o: %.c
        $(CC) $(CFLAGS) -c -o $@ $<
clean:
        rm -f $(OBJFILES) $(MAIN)
```

$ → refer to variable in makefile

program depends on object file.

"more generic" - we can use this by simple modifications.

(get rid of .o file) - c file alone enough.

* Library - has no. of object file — Archive.

* gcc weatherstats.c -o weatherstat.o

$ ar rcs libweather.a weatherstats.o

↓

replace
Creative
Index create for faster access

$ gcc -o program program.o libweather.a

Get arguments

```
printf("Argument %d : %s \n", i, argv[i]);
```

↳ only in command line