DE VINCI
INNOVATION
CENTER

BIENVENUE!

Introduction

2020
2021

LÉONARD DE VINCI
PARIS-LA DÉFENSE

ESILV LÉONARD DE VINCI
ÉCOLE D'INGÉNIEURS
PARIS-LA DÉFENSE

iiM LÉONARD DE VINCI
INSTITUT DE L'INTERNET ET DU MULTIMEDIA
PARIS-LA DÉFENSE

EMLV LÉONARD DE VINCI
ÉCOLE DE MANAGEMENT
PARIS-LA DÉFENSE

DE VINCI
INNOVATION
CENTER

Au programme

1 Qui ?

2 Philosophie

3 Organisation

4 Notation

5 Rythme

6 Des pépins

7 Pré requis

DE VINCI INNOVATION CENTER

1 Bienvenue

Group Meeting Vendredi

Mais en attendant, on a du travail

1 Qui?

Bonjour

C’est moi

1 Qui?

Grégor JOUET
PhD Student

Réseau
Sysadmin
Élec
Deep Learning
Machine Learning

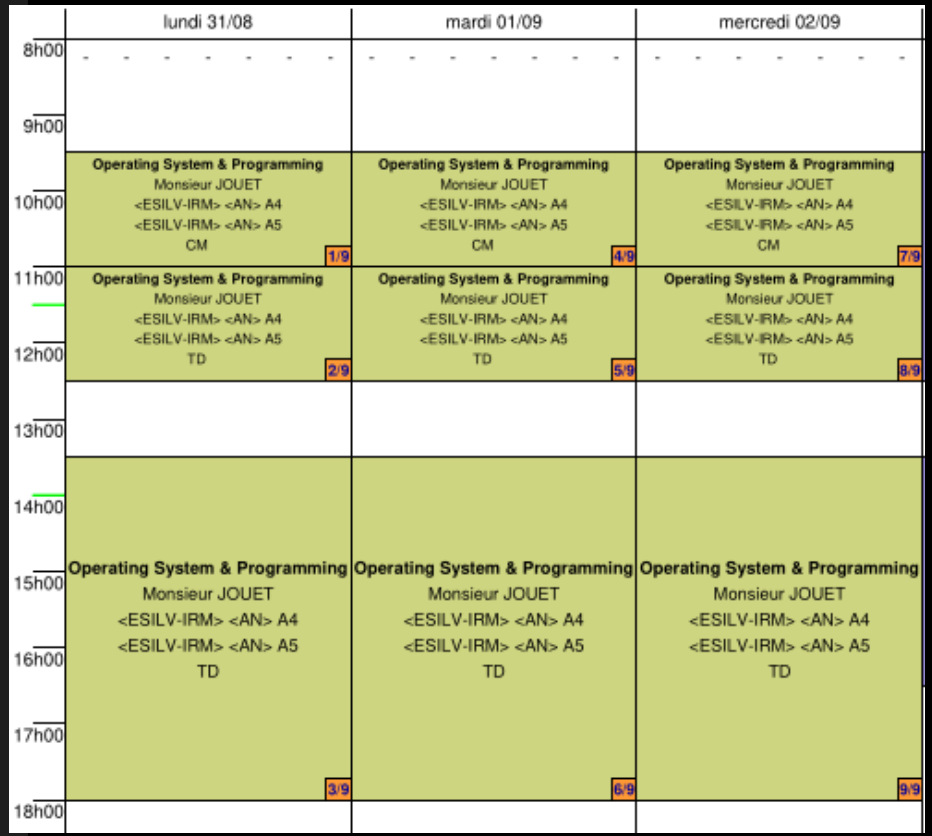
2 Philosophie.. des cours

Cours “fondamentaux”

- Connaissances techniques théoriques et pratiques indispensables
- Vous montrer l’étendu de ce qu’il est possible d’apprendre. (Parce qu’en 3 jours on pourra pas tout voir)
- Vous montrer les ressources et où chercher pour en savoir plus

3 Organisation

- Traditionnellement
  - Cours
  - Puis TD
  - repeat

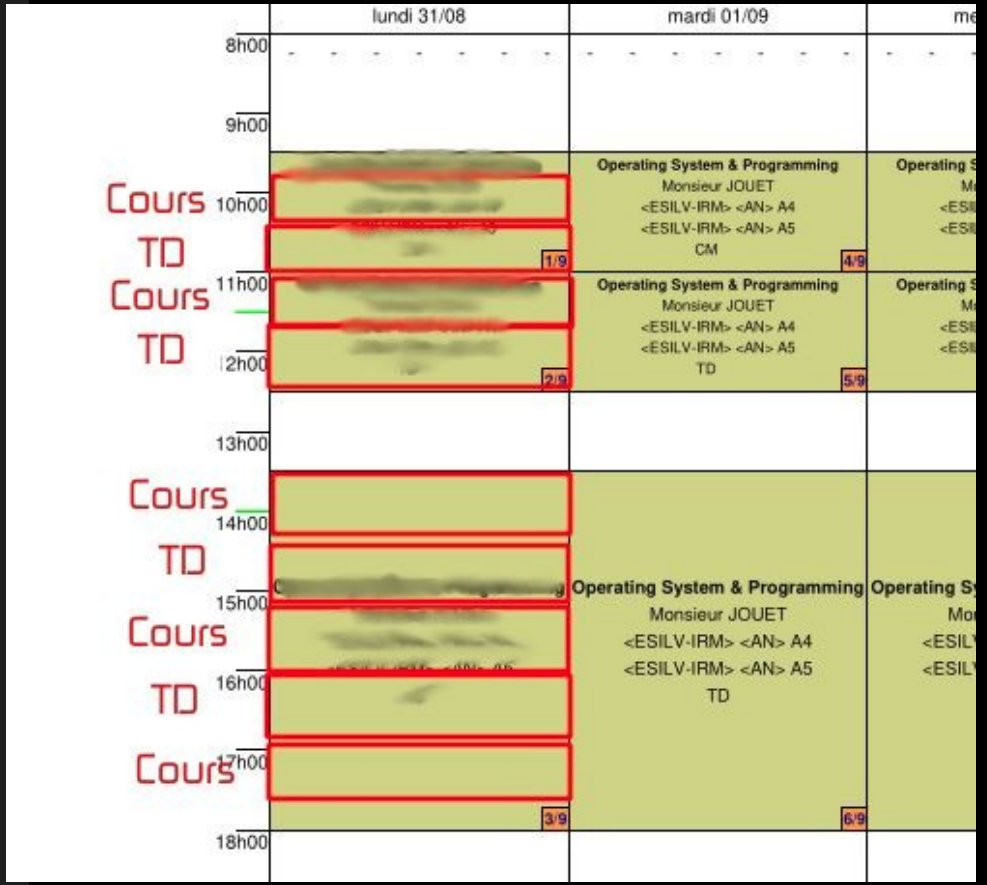


3 Organisation

On va essayer autre chose

- Alternance Cours/TD
- Pauses régulières

- Questions à la volée, et non pas à la fin d'un bloc de cours.



4 Notation

- **QCM à la fin des cours**

Pas de points négatifs, pas de pièges, simple restitution

- Projet sur le cours, à rendre dans 2 semaines.

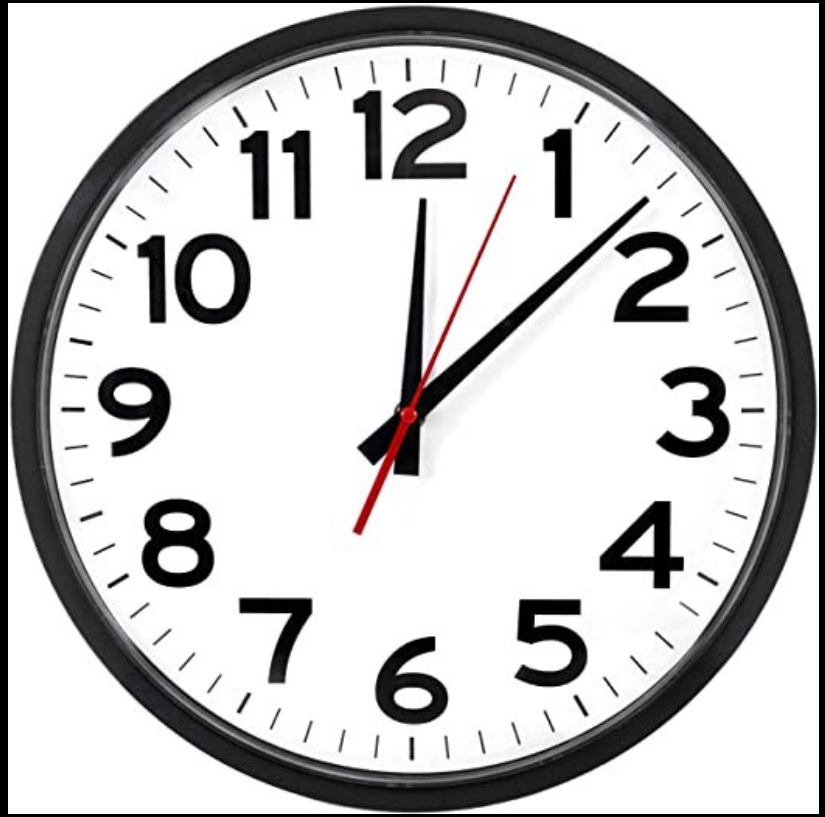
Sujet sous forme de cahier des charges, vous devez présenter un projet qui y répond. (Mais qui peut répondre à plusieurs projets)

- Examen en fin de semestre (on en reparlera)

5 Rythme

Le rythme des cours est volontairement soutenu

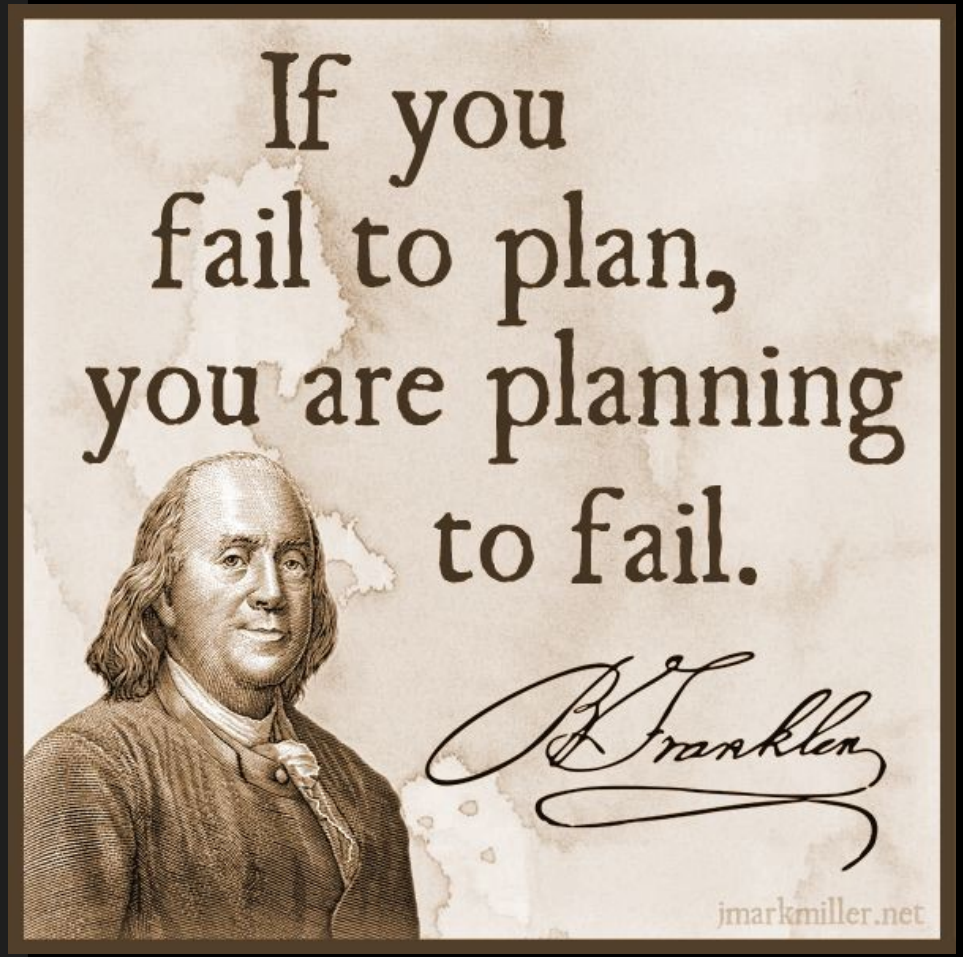
Votre principal challenge au DVIC sera d’avoir une bonne gestion de votre temps.
Vous ne devez pas faire d’impasse, réservez-vous du temps pour chaque projet et **n’hésitez pas à demander de l’aide de méthodologie.**



6 Des pépins

- Cursus nouveau
- Corona
- Nouvel environnement
- Nouveau groupe
- Nouvelles méthodes et philosophie

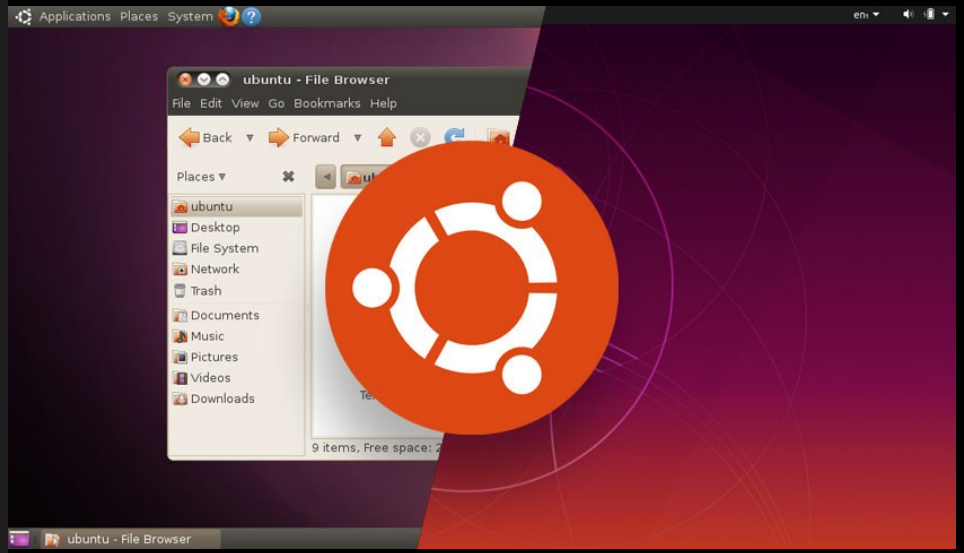
⇒ Il est certain qu'il y aura des difficultés.
Il est absolument primordiale d'en discuter le plus tôt possible.



7 Pré requis

- Tous les TPs sont prévus pour Ubuntu
- Compatibilité pas garantie pour Mac mais devrait marcher
- Windows, pas la peine

- Zoom marche sur linux au fait
- Trouvez quelqu’un qui a un linux pour pour les TP.



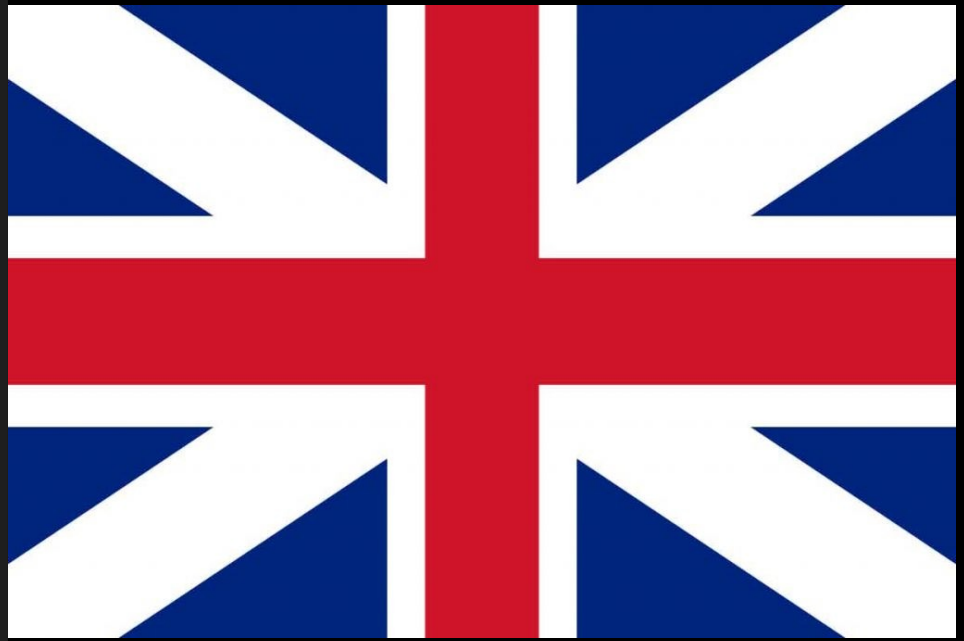
? Next

- Aujourd’hui: GNU / Linux
- Demain: Langage C
- Mercredi: Python

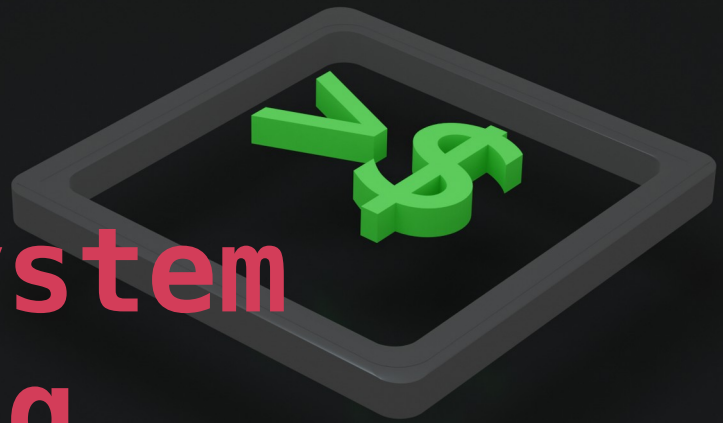
- Semaine Prochaine: Data Structures & Algorithms

Get Ready

Slides are now in English



DE VINCI INNOVATION CENTER



Operating System & Programming

Lesson 1 GNU / Linux

Grégor JOUET
PhD Student
gregor.jouet@ext.devinci.fr

LÉONARD DE VINCI PARIS-LA DÉFENSE
ESILV LÉONARD DE VINCI ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE
iiM LÉONARD DE VINCI INSTITUT DE L'INTERNET ET DU MULTIMEDIA PARIS-LA DÉFENSE
EMLV LÉONARD DE VINCI ÉCOLE DE MANAGEMENT PARIS-LA DÉFENSE

DE VINCI INNOVATION CENTER

Course Overview

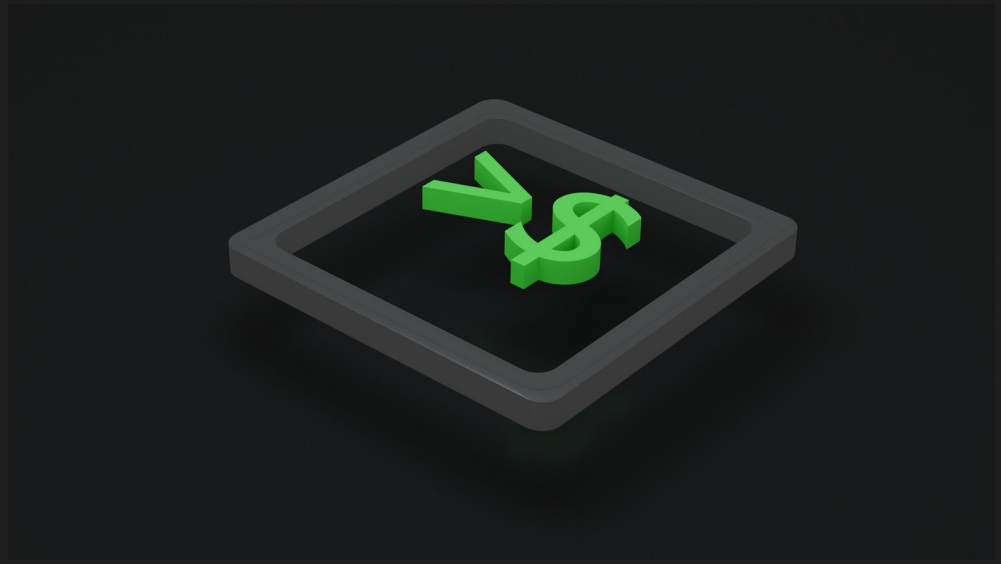
1 Introduction

2 Hardware

3 OS Basics

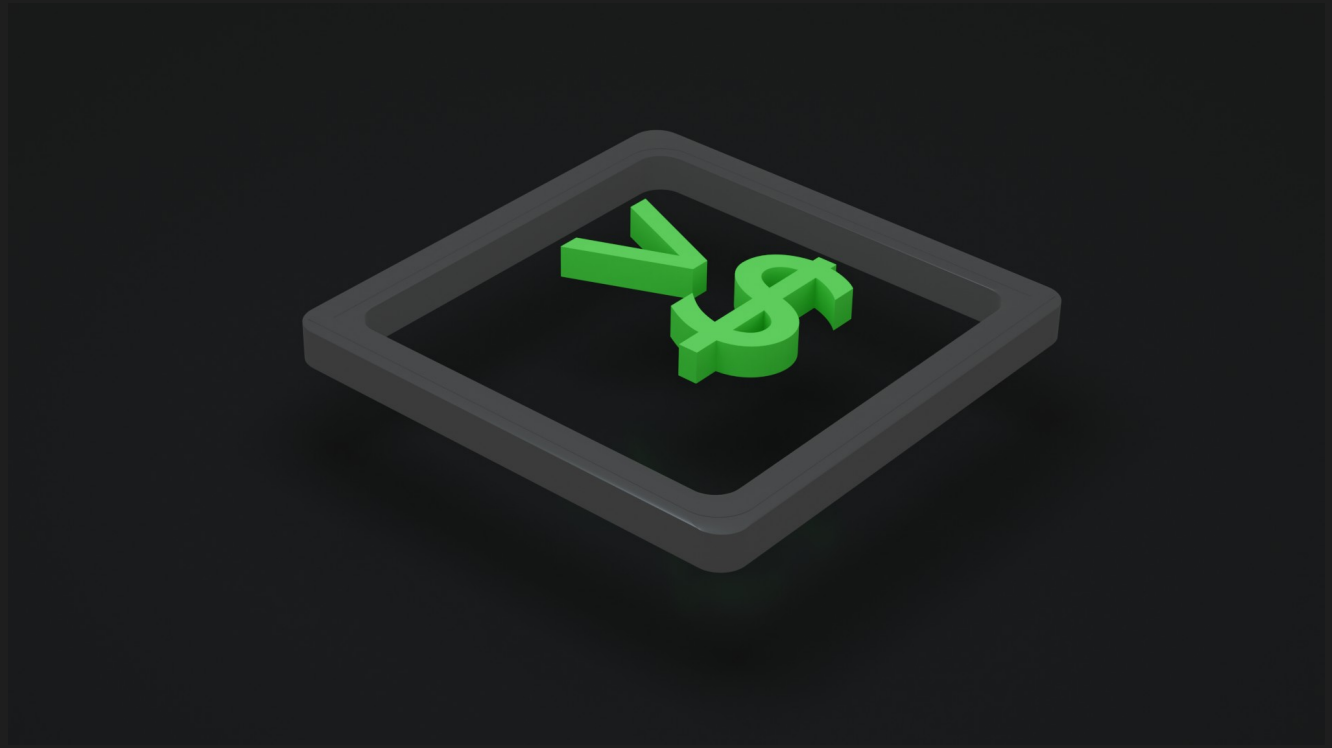
4 About Linux

5 Daily Usage



DE VINCI INNOVATION CENTER

1| Intro & History



DE VINCI
INNOVATION
CENTER

1| Intro & History

Dennis Ritchie

- Invented C
- Invented UNIX
- Paved the way for BSD, Linux, Apple etc... In 1970

DE VINCI INNOVATION CENTER

1| Intro & History

Linus Torvalds

- Initiated Linux
- Linux today:
  - 80% of the cloud
  - Android
  - Robotics
  - Etc...

DE VINCI
INNOVATION
CENTER

1| Intro & History

Meanwhile

- MS DOS
- Soon to be Windows NT

DE VINCI
INNOVATION
CENTER

1 Intro

What is an OS for ?

Share resources

Share process time
Memory, GPU among users

Access Control

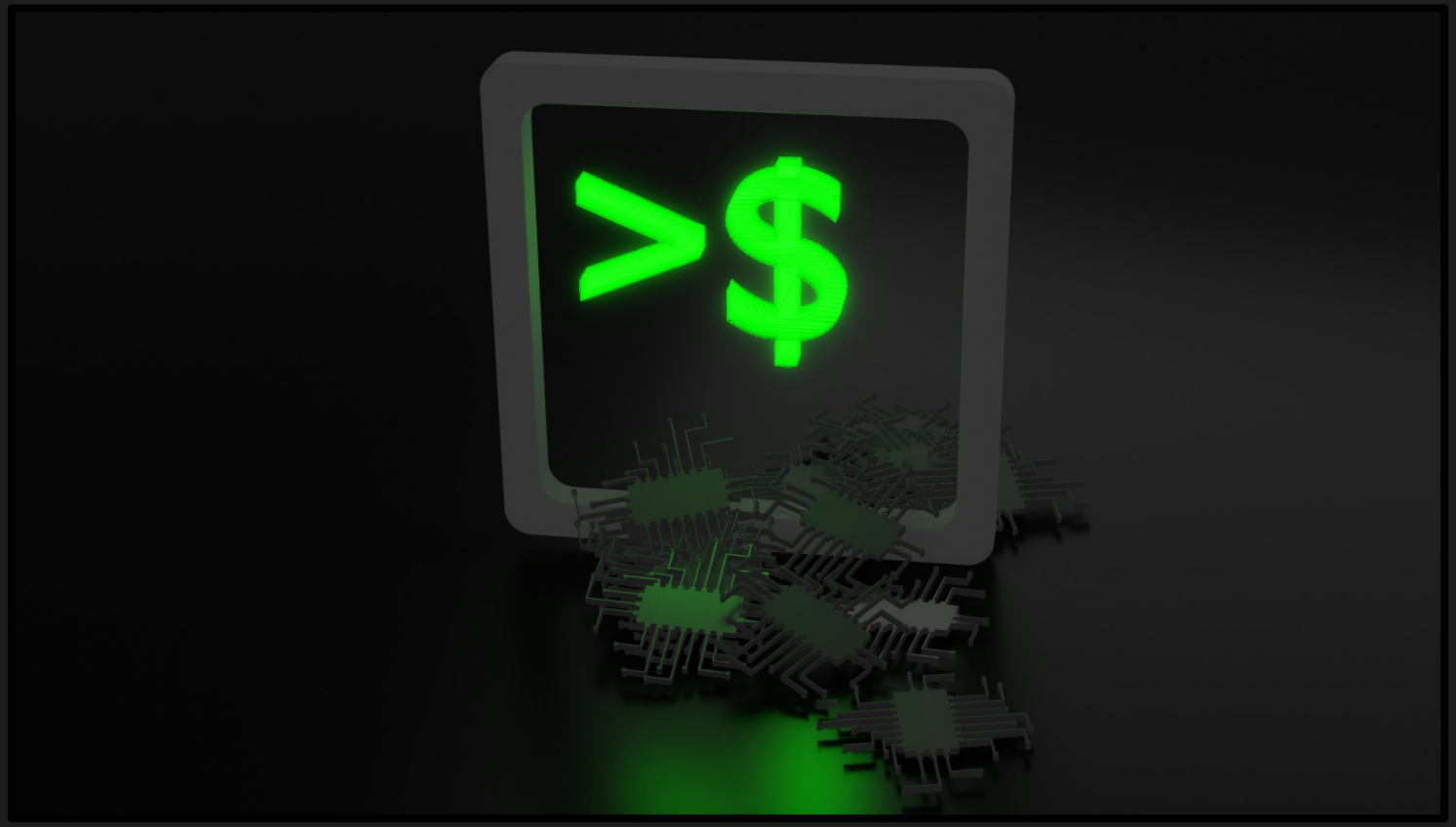
Decide who can access to which resource

Hardware abstraction

Mask the complexity of hardware handling

DE VINCI INNOVATION CENTER

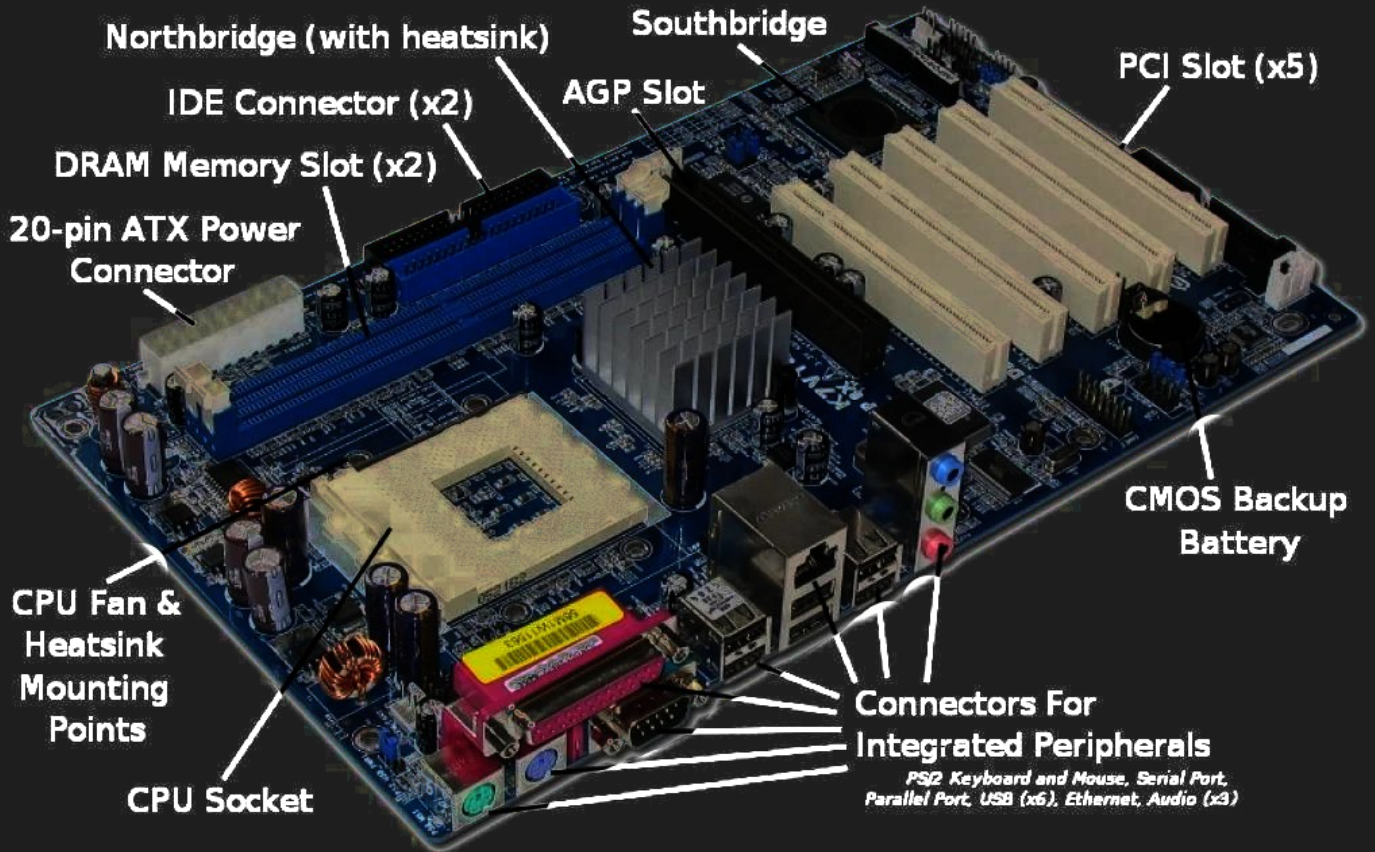
2 | Hardware



DE VINCI
INNOVATION
CENTER

2

Hardware



DE VINCI INNOVATION CENTER

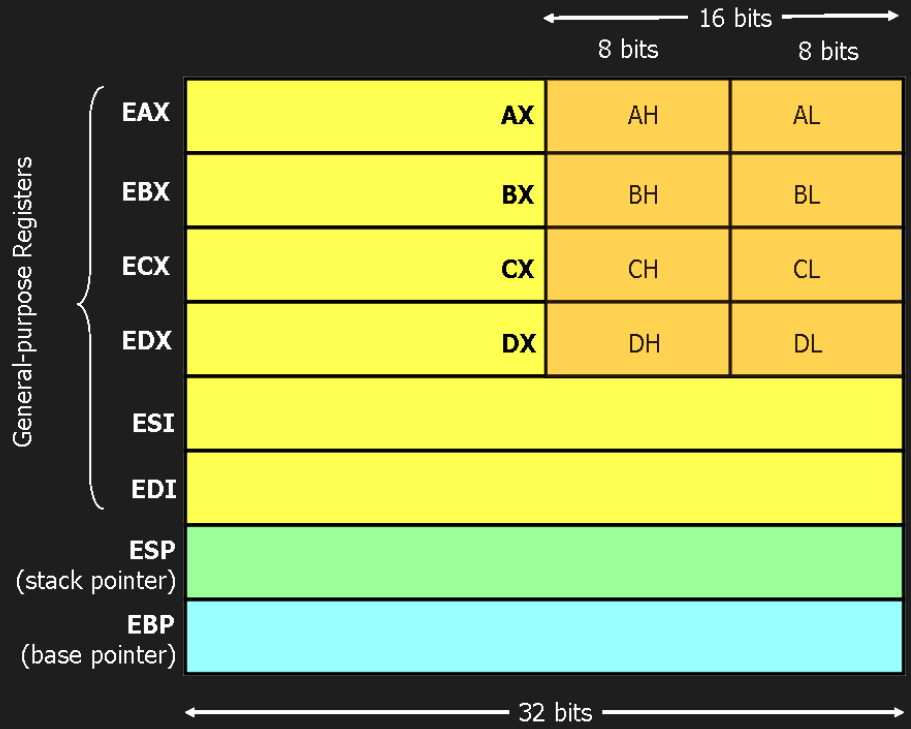
2| Hardwar e

- CPU executes instructions 1 by 1
- “Instructions” are built from a hardware specific language: **Assembly** (asm)
- Assembly is formed by an Instruction Set Architecture (ISA)

DE VINCI INNOVATION CENTER

2| Hardwar e

- CPU has “mini memories” called **Registers**
- Registers can be used for calculation, program flow control, cpu features control, argument passing...



DE VINCI INNOVATION CENTER

2| Hardware

- **ASM** allows direct control over memory and registers
- **ASM** is executed as binary

```
section .text
   global _start     ;must be declared for linker (ld)

_start:              ;tells linker entry point
   mov  edx,len      ;message length
   mov  ecx,msg      ;message to write
   mov  ebx,1        ;file descriptor (stdout)
   mov  eax,4        ;system call number (sys_write)
   int  0x80         ;call kernel

   mov  eax,1        ;system call number (sys_exit)
   int  0x80         ;call kernel

section .data
msg d 'Hello, world!', 0xa  ;string to be printed
len equ $ - msg     ;length of the string
```

DE VINCI INNOVATION CENTER

2| Hardwar

- Assembly code is separated in **segments**
- ***.text*** is the actual program
- ***.data*** is the program static data

```
1   section .text          Text segment
2      global _start       ;must be declared for linker (ld)
3
4   _start:                ;tells linker entry point
5      mov  edx,len        ;message length
6      mov  ecx,msg        ;message to write
7      mov  ebx,1          ;file descriptor (stdout)
8      mov  eax,4          ;system call number (sys_write)
9      int  0x80           ;call kernel
10
11     mov  eax,1          ;system call number (sys_exit)
12     int  0x80           ;call kernel
13
14  section .data          Data Segment
15  msg d 'Hello, world!', 0xa  ;string to be printed
16  len equ $ - msg        ;length of the string
```

DE VINCI INNOVATION CENTER

2| Hardwar**e**

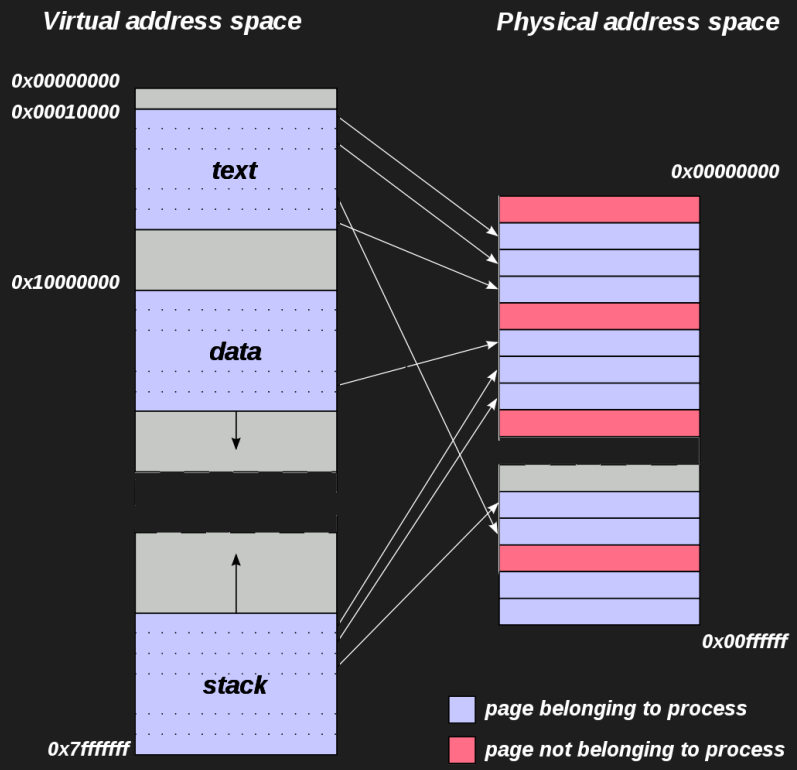
- Hardware (or software) can required immediate attention from the system
- An **Interrupt Request** (IRQ) is fired to handle this. The function executed is an **interrupt handler**

IRQ	Usage
0	system timer (cannot be changed)
1	keyboard controller (cannot be changed)
2	cascaded signals from IRQs 8–15
3	second RS-232 serial port (COM2: in Windows)
4	first RS-232 serial port (COM1: in Windows)
5	parallel port 2 and 3 or sound card
6	floppy disk controller
7	first parallel port
8	real-time clock
9	open interrupt
10	open interrupt
11	open interrupt
12	PS/2 mouse
13	math coprocessor
14	primary ATA channel
15	secondary ATA channel

DE VINCI INNOVATION CENTER

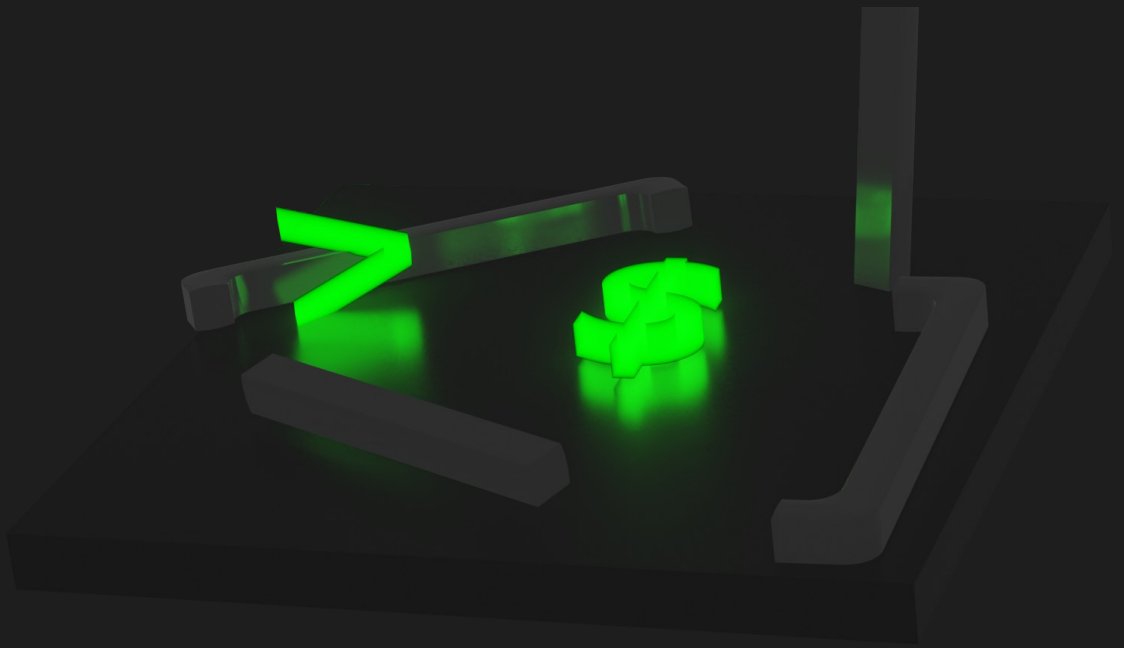
2| Hardware

- Modern CPU can enter a **Protected Mode** where they can see a virtual memory and not have access outside of it.
- This is used with **processes** to separate memory for different applications



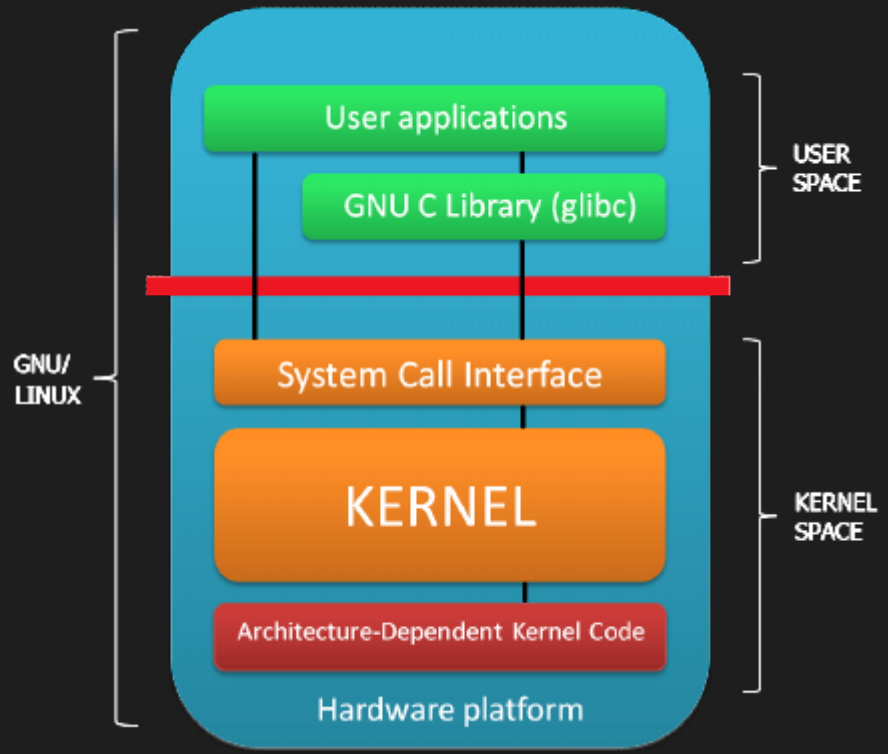
DE VINCI INNOVATION CENTER

3|OS Basics



DE VINCI
INNOVATION
CENTER

3| OS Basics



DE VINCI INNOVATION CENTER

3|OS Basics

Process

The base unit for resource sharing

File

The base unit of data organisation

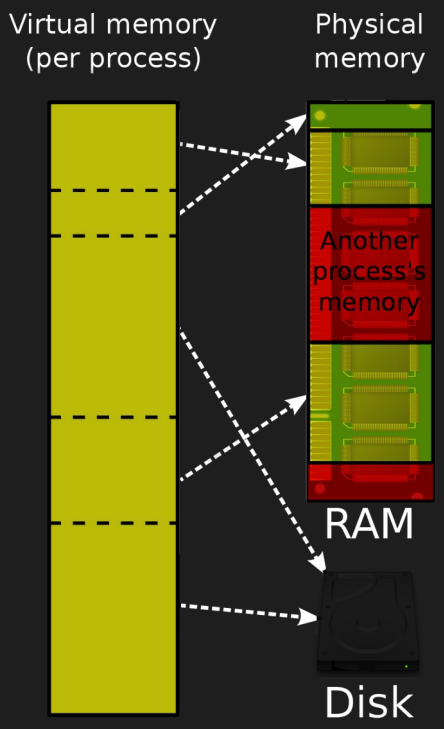
Syscall

The only way programs interact with hardware

3 OS Basics

Process

- Has a unique identifier (**PID**)
- Has its own **virtual memory**
- Starts with an **executable**
- Has a parent process



DE VINCI
INNOVATION
CENTER

3 OS Basics

Process

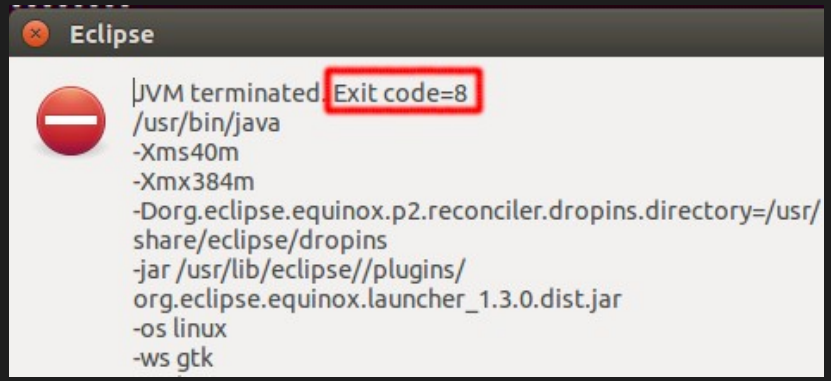
- Can receive **Signals**
- Signals are used by the OS to communicate essential informations

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

3 OS Basics

Process

- When process exit, they give a **return code.** An integer indicating why they stopped.
- Usually
  - **0:** Normal exit
  - >0 Error
- Can help troubleshooting



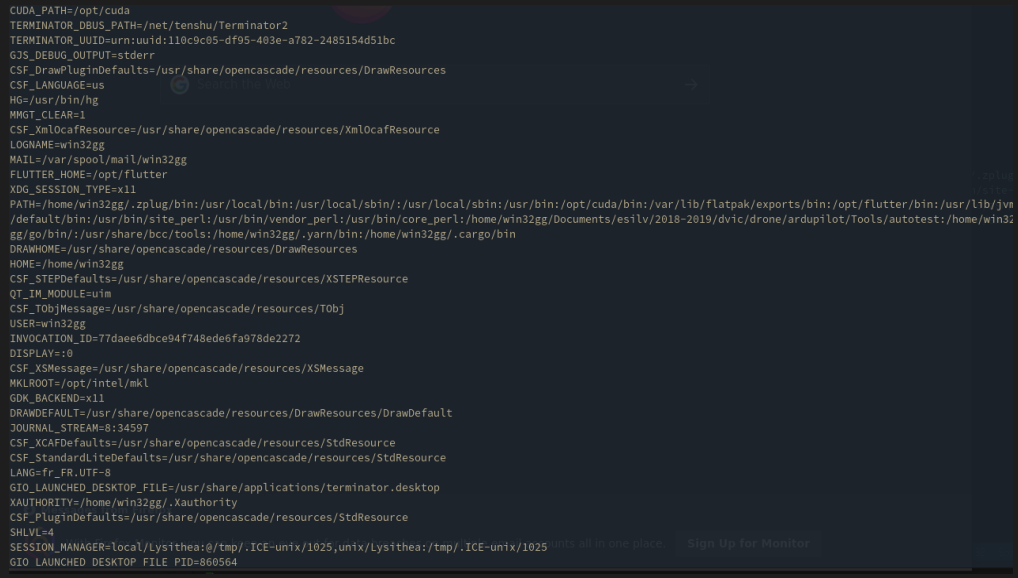
DE VINCI INNOVATION CENTER

3 OS Basics

Process

- Has **Environment Variables**
- Has a **Current Working Directory**

See chdir(2), getcwd(2) for linux

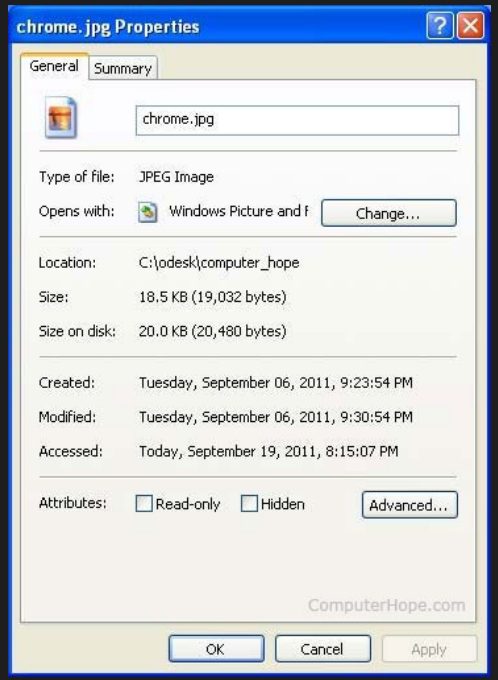


DE VINCI INNOVATION CENTER

3 OS Basics

File

- Base unit to access and organise documents
- Files are organized in a Tree.
- Files have a **Owner, Permissions, Name, Access Time**

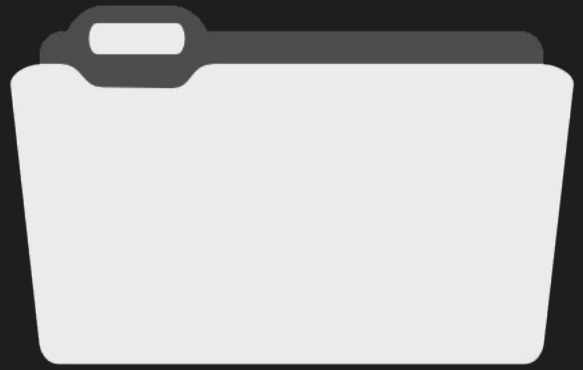


DE VINCI INNOVATION CENTER

3 OS Basics

File

- Files can be contained in **Folders** technically considered a File as well.



DE VINCI INNOVATION CENTER

3 OS Basics

File

Files can be accessed with an **Absolute Path**
C:\Users\me\file.txt
/home/me/file.txt

Or can be accessed with a **Relative Path**
Relative to the process current directory:
../file1.txt

```
[win32gg@Lysithea:examples/relative]$ pwd                    (08-20 10:43)
/home/win32gg/Documents/DVIC/cours/OS/examples/relative
[win32gg@Lysithea:examples/relative]$ cat file1.txt          (08-20 10:43)
hello
[win32gg@Lysithea:examples/relative]$ cd sub                 (08-20 10:43)
[win32gg@Lysithea:relative/sub]$ pwd                         (08-20 10:43)
/home/win32gg/Documents/DVIC/cours/OS/examples/relative/sub
[win32gg@Lysithea:relative/sub]$ cat ../file1.txt            (08-20 10:43)
hello
[win32gg@Lysithea:relative/sub]$ cat /home/win32gg/Documents/DVIC/cours/
OS/examples/relative/file1.txt
hello
[win32gg@Lysithea:relative/sub]$                             (08-20 10:44)
```

DE VINCI INNOVATION CENTER

3 OS Basics

Syscalls

- **Syscalls** are special functions defined by the OS kernel.
- They serve as base abstractions for a program to interact with the hardware or the OS itself.

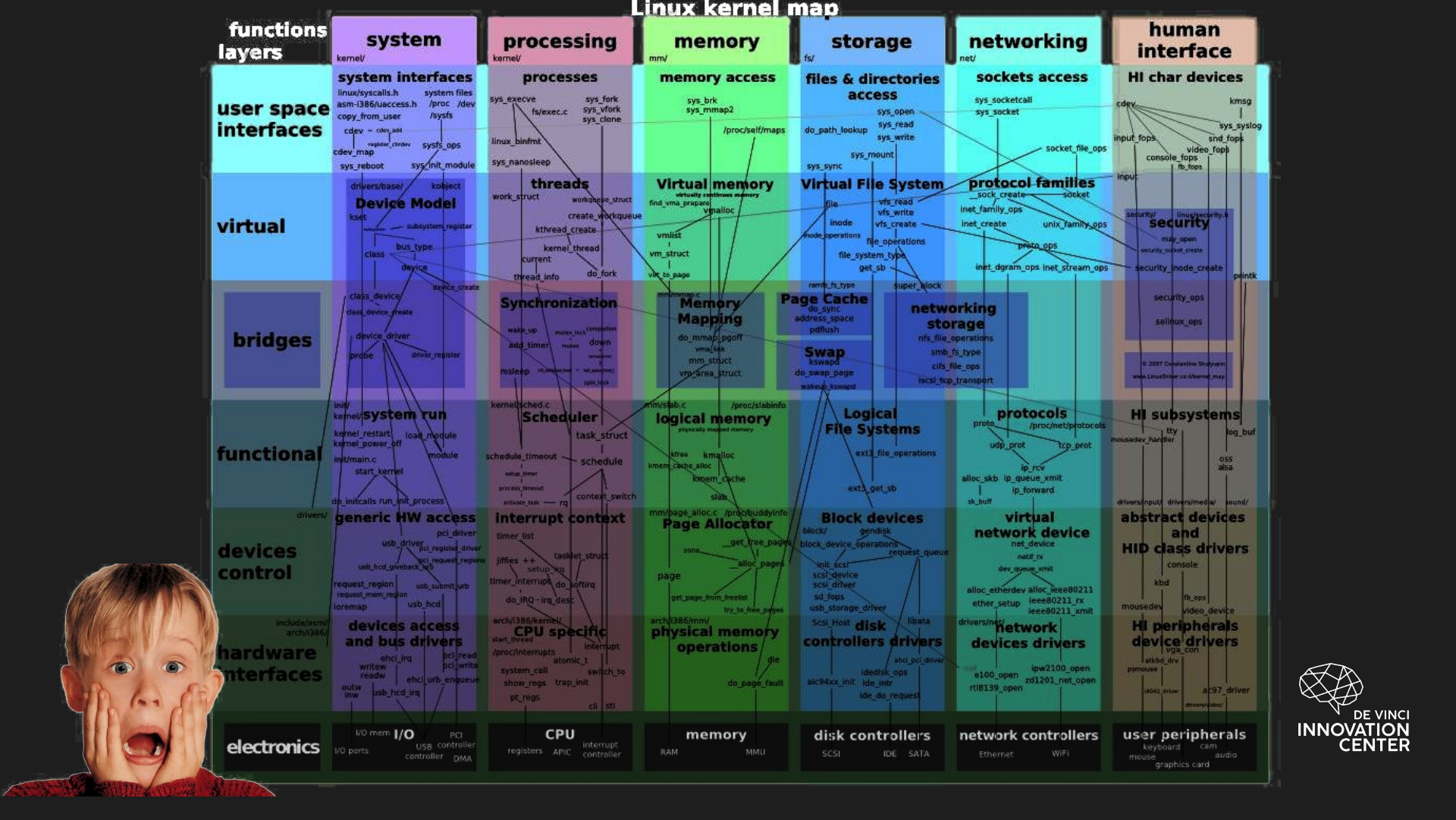
%rax	Name	Entry point
0	read	sys_read
1	write	sys_write
2	open	sys_open
3	close	sys_close
4	stat	sys_newstat
5	fstat	sys_newfstat
6	lstat	sys_newlstat
7	poll	sys_poll
8	lseek	sys_lseek
9	mmap	sys_mmap
10	mprotect	sys_mprotect
11	munmap	sys_munmap
12	brk	sys_brk

NtCancelTimer
NtCancelWaitCompletionPacket
NtClearAllSavepointsTransaction
NtClearEvent
NtClearMUILicenseInfo
NtClearSavepointTransaction
NtClose
NtCloseObjectAuditAlarm
NtCommitComplete
NtCommitEnlistment
NtCommitTransaction
NtCompactKeys
NtCompareTokens
NtCompleteConnectPort
NtCompressKey
NtConnectPort
NtContinue

DE VINCI INNOVATION CENTER

4 | About Linux

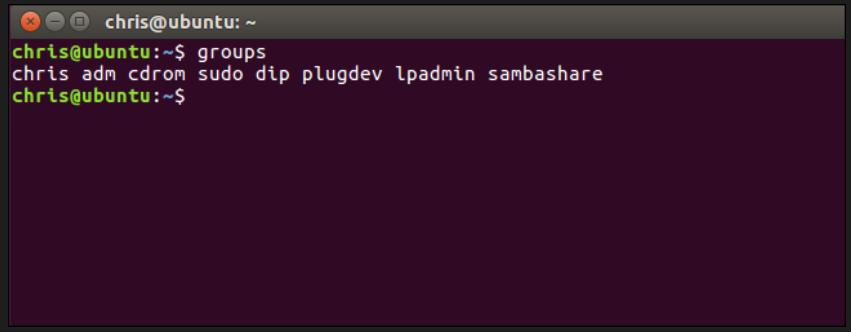
DE VINCI
INNOVATION
CENTER


Linux kernel map
functions
layers
system
kernel/
processing
kernel/
memory
mm/
storage
fs/
networking
net/
human interface
user space interfaces
virtual
bridges
functional
devices control
hardware interfaces
electronics
system interfaces
processes
memory access
files & directories access
sockets access
HI char devices
Device Model
threads
Virtual memory
Virtual File System
protocol families
security
Synchronization
Memory Mapping
Page Cache
networking storage
Swap
system run
Scheduler
logical memory
Logical File Systems
protocols
HI subsystems
generic HW access
interrupt context
Page Allocator
Block devices
virtual network device
abstract devices and HID class drivers
devices access and bus drivers
CPU specific
physical memory operations
disk controllers drivers
network devices drivers
HI peripherals device drivers
I/O
CPU
memory
disk controllers
network controllers
user peripherals
DE VINCI INNOVATION CENTER

3 About Linux

Users

- On linux the administrator is **root**
- Each user has a unique id. Root id is 0
- Users can be in groups

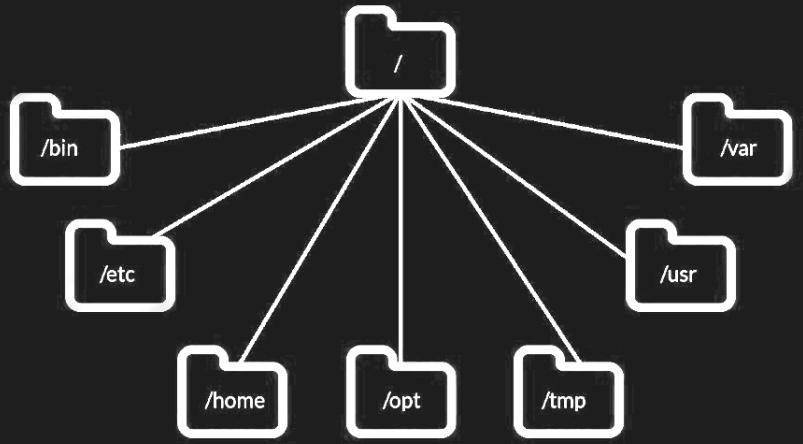


DE VINCI INNOVATION CENTER

3 About Linux

Files & dirs
Examples

- “/etc/fstab” is the file “fstab” in the directory “/etc”
- “/usr/lib/libssl3.so” is the file libssl3.so in the directory /usr/lib

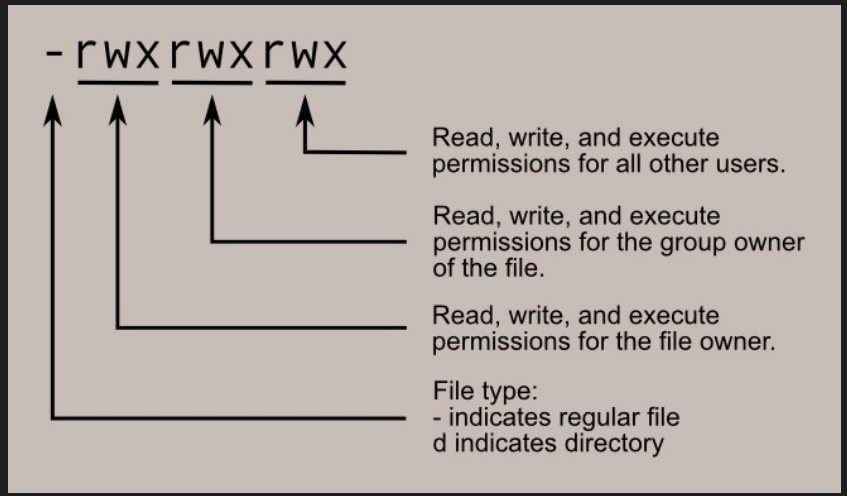


DE VINCI INNOVATION CENTER

3 About Linux

Access control

- File can be **Readable, Writable, Executable** by **The Owner, The Group, Other users**
- -rw-r--r-- is readable by everyone, writable by the owner, executable by no one



DE VINCI INNOVATION CENTER

3 About Linux

Access control

- **BONUS :D**
  Some files like /bin/passwd have an extra attribute: **The Sticky Bit**
- You can execute this file has its owner even if its root

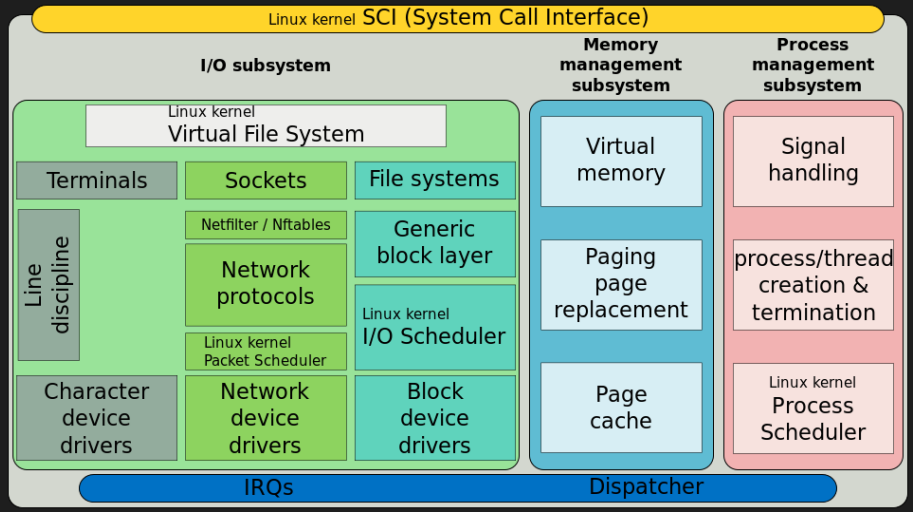
```
-rwsr-xr-x 1 root root 63640  4 févr.  2020 /bin/passwd
```

DE VINCI INNOVATION CENTER

3. About Linux

Files
Filesystems
About Syscalls

- All files do not come from a disk !
- The difference cannot be seen when

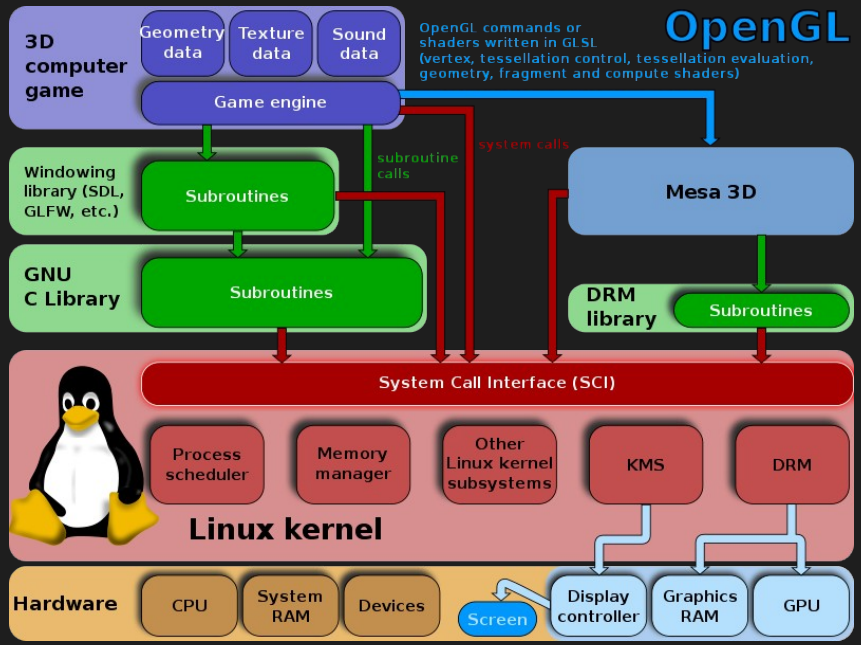


DE VINCI
INNOVATION
CENTER

3 About Linux

Syscalls everywhere

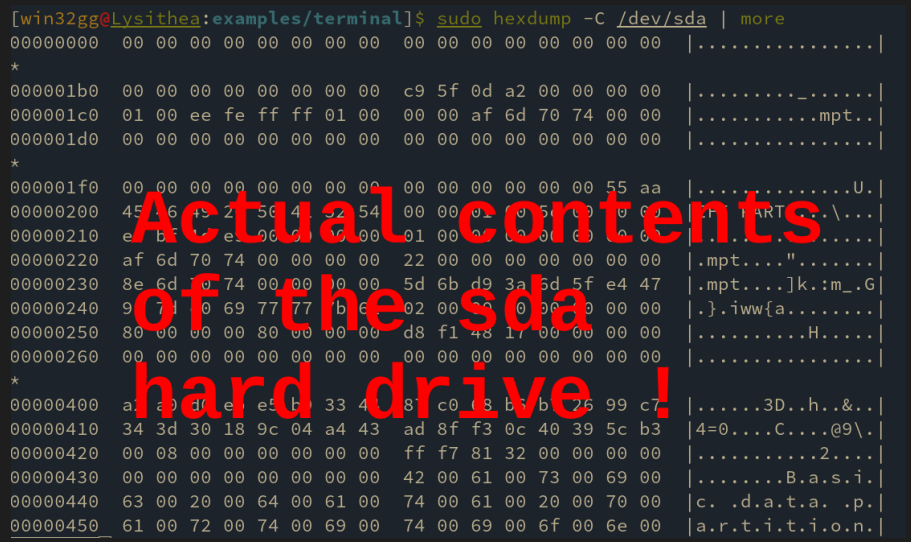
- All programs use syscalls to interact with OS
- Syscall are used to interact with all of the devices



DE VINCI INNOVATION CENTER

3 About Special FIles Linux

- In linux **Everything is a file**
- Special files give you access to useful infos about the OS, or are direct access to the hardware



Actual contents of the sda hard drive !

DE VINCI INNOVATION CENTER

3 About Linux

Files Descriptors

- When a process wants to read/write something, it uses a **File Descriptor**
- A file descriptor is an Integer, managed by the OS
- File descriptors 0, 1, 2 are reserved for **stdin**, **stdout** and **stderr**

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fd = open("./test", O_CREAT | O_RDWR);
    write(fd, "hi!", 3);
    close(fd);
    return 0;
}
```

DE VINCI INNOVATION CENTER

5| Daily Usage

DE VINCI
INNOVATION
CENTER

3 Daily Usage

You live in the Shell

- Essential commands

ls cd sudo cat

- Terminal uses **sh** (Bourne Shell) or **bash**
- **Be careful, most commands don’t ask for confirmation**
- More in the workshops

```
[win32gg@Lysithea:examples/terminal][130]$ ls -ls
total 0
0 -rw-r--r-- 1 win32gg users 0 18 août  18:36 file1
0 -rw-r--r-- 1 win32gg users 0 18 août  18:36 file2
[win32gg@Lysithea:examples/terminal]$ sudo su
[root@Lysithea:examples/terminal]#
[win32gg@Lysithea:examples/terminal]$ echo "ok" > file1
[win32gg@Lysithea:examples/terminal]$ cat file1
ok
[win32gg@Lysithea:examples/terminal]$ cat --help
Utilisation : cat [OPTION]... [FICHIER]...
Concaténer les FICHIERs vers la sortie standard.

Sans FICHIER ou quand FICHIER est -, lire l'entrée standard.

  -A, --show-all           équivalent à -vET
  -b, --number-nonblank    numéroter les lignes non vides en sortie, annule -n
  -e                       équivalent à -vE
  -E, --show-ends          afficher $ à la fin de chaque ligne
  -n, --number             numéroter toutes les lignes en sortie
  -s, --squeeze-blank      supprimer les lignes vides qui se répètent en sortie
  -t                       équivalent à -vT
```

DE VINCI INNOVATION CENTER

3 Daily Usage

Files to configure

- Linux system configuration is mainly done through files.
- /etc/fstab
- /etc/hosts
- /etc/resolv.conf
- /etc/passwd

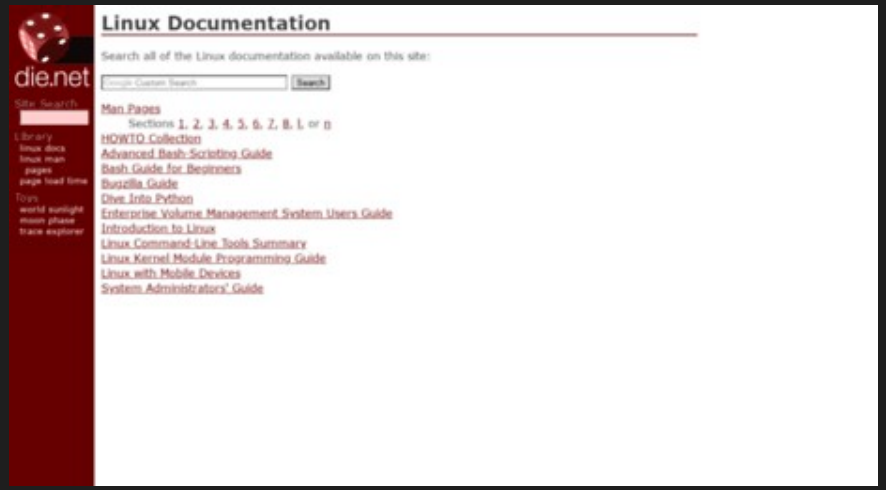
```
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/mapper/vgnv-root
UUID=7b2386c3-bf5b-48c6-8050-e96f660cae4d       /               ext4
/dev/nvme0n1p6                                  /boot           ext2
/dev/nvme0n1p1                                  /boot/efi       vfat
#/dev/mapper/vgnv-swap                          none            swap
#/dev/mapper/vgsd-swap                          none            swap
/dev/mapper/vgsd-home                           /home           ext4
/dev/mapper/vgsd-opt                            /opt            ext4
#/dev/nvme0n1p5                                 /efi-backup     vfat
/dev/disk/by-uuid/0620EE9220EE87CF              /windows/data   ntfs-3g
UUID=04F058A2F0589C2C                           /windows/exchange       ntfs-3g
```

DE VINCI INNOVATION CENTER

3 Daily Usage

Getting Help

- Use the man pages !
- Use linux.die
- Use the **man** command
- Use --help after a command



DE VINCI INNOVATION CENTER

3 Daily Usage

The Warning

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

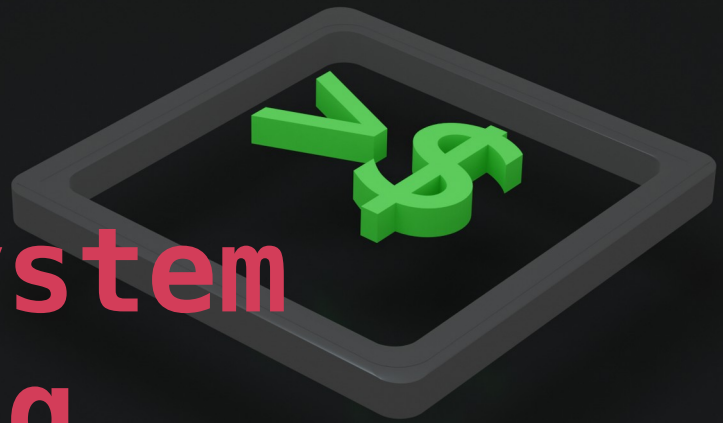
1) Respect the privacy of others.

2) Think before you type.

3) With great power comes great responsibility.

DE VINCI INNOVATION CENTER

DE VINCI INNOVATION CENTER



Operating System & Programming

Lesson 2 C / Python Part #1

Grégor JOUET
PhD Student
gregor.jouet@ext.devinci.fr

LÉONARD DE VINCI PARIS-LA DÉFENSE

ESILV LÉONARD DE VINCI ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

iiM LÉONARD DE VINCI INSTITUT DE L'INTERNET ET DU MULTIMEDIA PARIS-LA DÉFENSE

EMLV LÉONARD DE VINCI ÉCOLE DE MANAGEMENT PARIS-LA DÉFENSE

DE VINCI INNOVATION CENTER

C Programming Language

1 Introduction & Threads

2 Practical Compilation Guide

3 C Syntax Overview

4 About Memory

5 Patterns

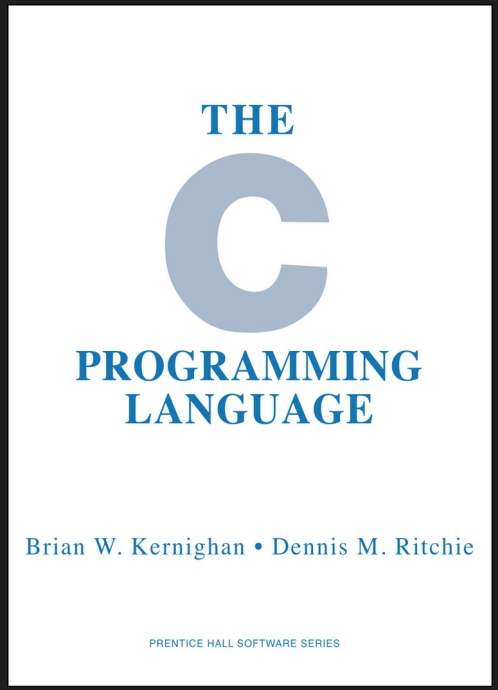
6 Errors

DE VINCI INNOVATION CENTER

1| Introduction

C Language

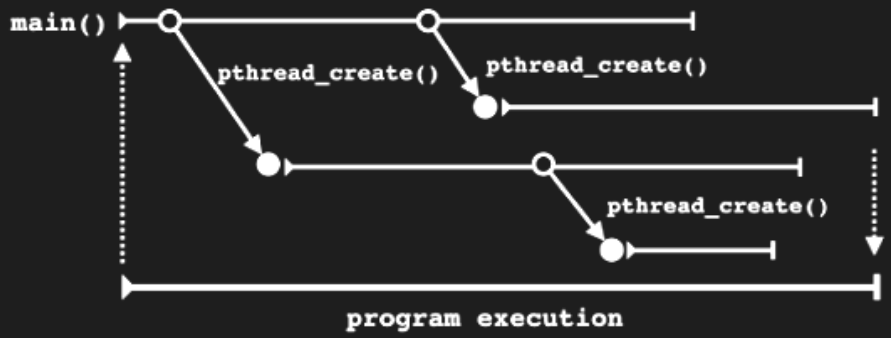
- Dennis Ritchie
- General purpose language
- Close to memory
- Compiled language



DE VINCI
INNOVATION
CENTER

1| Introduction
Threads

- Multiple concurrent execution in the same process
- Using the pthread library



DE VINCI INNOVATION CENTER

2| Practical compilation guide

Process

- C is a compiled language. It is converted to machine code to be executed.
- Compilation is the action of converting human readable code to machine code

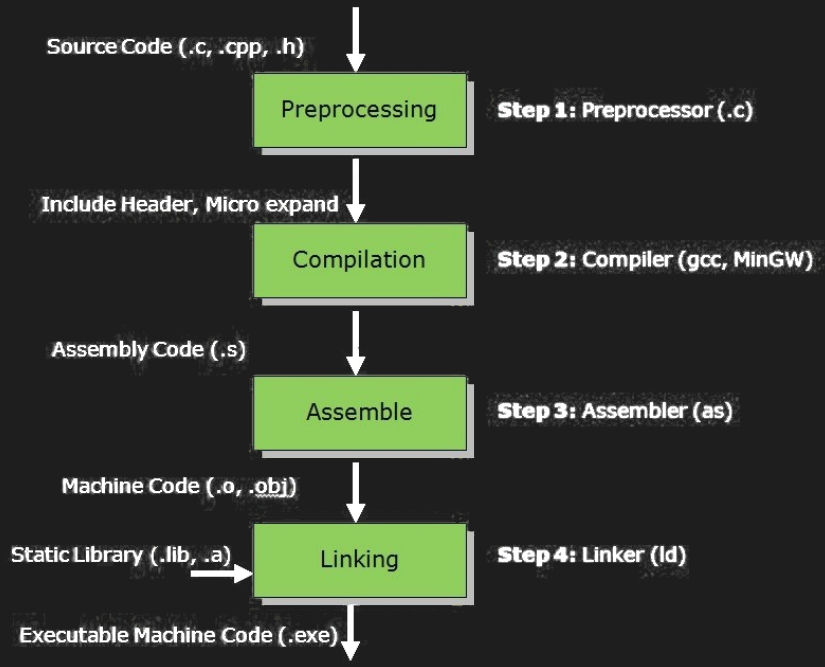
```
GCC(1)

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-c|-S|-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-Wpedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] [@file] infile...
```

DE VINCI INNOVATION CENTER

2| Practical compilation guide

Linking

- The compilation can be split into 3 steps
  - Pre-processing
  - Assembly
  - Linking
- Assembly changes C code to assembler & machine code
- Linking sets up dynamic libraries and startup up code

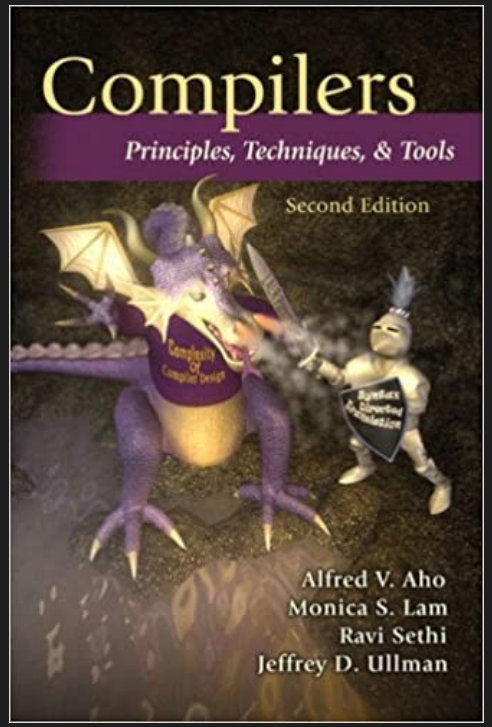


DE VINCI INNOVATION CENTER

2| Practical compilation guide

Going Further

- Compilers are cool, but hard, but cool.
- Sorry, not time
- Read the book (~900 pages good luck)
- ISBN-10: 0321486811



DE VINCI INNOVATION CENTER

3| C Syntax Overview

Variables

<type> <name> = <value>;

<type> can be raw type, modified raw types eg. “unsigned” or more complex structure

Type	Explanation
`char`	Smallest addressable unit of the machine that can contain basic character set. It is an inte type can be either signed or unsigned. It contains CHAR_BIT bits.[3]
`signed char`	Of the same size as *char*, but guaranteed to be signed. Capable of containing at least the range.[3][note 1]
`unsigned char`	Of the same size as *char*, but guaranteed to be unsigned. Contains at least the [0, 255] ra

`short`
`short int`
`signed short`
`signed short int`
	Short signed integer type. Capable of containing at least the [−32,767, +32,767] range.[3

`unsigned short` `unsigned short int`	*Short* unsigned integer type. Contains at least the [0, 65,535] range.[3]

`int`
`signed`
`signed int`
	Basic signed integer type. Capable of containing at least the [−32,767, +32,767] range.[3

DE VINCI INNOVATION CENTER

3| C Syntax Overview

#include

#include is a pre-processing directive to include another .c file or a header (.h) file.

#include “file.c” includes file.c from the current directory
#include <stdio.h> includes stdio.h from the system include dir (/usr/include)

```c
#include <stdio.h>   // import stdio from /usr/include
#include <stdbool.h> // import stdbool from /usr/include
#include "test/myh.h" // import myh from the test/ directory
// myh defines a int myvar

int main() {
    printf("%i\n", myvar);
    return 0;
}
```

DE VINCI INNOVATION CENTER

3| C Syntax Overview

Booleans

No boolean in C by default
Use #include<bool.h>
or use a 1 byte variable to store your boolean and create your type with **typedef**

```c
#include <stdbool.h> // import bool

typedef char mybool; // define my bool type as a char
#define mytrue 1
#define myfalse 0

int main() {
    bool a = true;
    mybool b = mytrue;
    return 0;
}
```

DE VINCI INNOVATION CENTER

3| C Syntax Overview

Headers as API

Developers give function information and access point as header files. The linker will make the link with the actual library function.

```
[win32gg@Enceladeus:~/test]$ cat /usr/include/stdio.h | g
extern int printf (const char *__restrict __format, ...);
[win32gg@Enceladeus:~/test]$ ldd ./main
        linux-vdso.so.1 (0x00007fff90dc0000)
        libc.so.6 => /usr/lib/libc.so.6 (0x00007f550eba00
        /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linu
[win32gg@Enceladeus:~/test]$ nm -g -C /usr/lib/libc.so.6
0000000000057980 T printf
```

DE VINCI INNOVATION CENTER

3| C Syntax Overview

#define

#define is a pre-processing directive to define a variable. Useful to declare constants or not to include the same library twice when using it with #if / #endif

```c
#include <stdio.h>

#define MYVAR 10 // MYVAR is not compiled
// it is replaced in the source code

int main() {
    printf("%i\n", MYVAR);
    return 0;
}
```

DE VINCI INNOVATION CENTER

3| C Syntax & Memory

sizeof

sizeof is a very useful function to know the length of any type

```
[win32gg@Enceladeus:~/test][1]$ bat test.c
       File: test.c
   1   #include <stdio.h>
   2
   3   int main() {
   4       printf("type\tsize\n");
   5       printf("int\t%i\n", sizeof(int));
   6       printf("long\t%i\n", sizeof(long));
   7       return 0;
   8   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_size
[win32gg@Enceladeus:~/test]$ ./test_size
type    size
int     4
long    8
```

DE VINCI INNOVATION CENTER

4| About Memory

Pointers

A pointer is a variable containing the memory address of another variable.

Pointers declare the type of the variable they point to.

```
[win32gg@Enceladeus:~/test]$ bat test.c
       File: test.c
   1   #include <stdio.h>
   2
   3   int main() {
   4       int a; // an integer
   5       int *b; // a pointer to an integer
   6
   7       a  = 43; // we access the variable directly
   8       *b = 42; // we access the memory the variable points to
   9       b = 42; // CAREFUL, this make the variable point to memory
  10           // adress 42, which will cause a problem.
  11
  12       // b is the pointer, *b is the int at the assiciated address.
  13       printf("%i\n", *b);
  14       return 0;
  15   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_segv
test.c: Dans la fonction « main »:
test.c:9:4: attention: l'affectation à « int * » depuis « int » transforme un entier en pointeur
    9 |   b = 42; // CAREFUL, this make the variable point to memory
      |   ^
[win32gg@Enceladeus:~/test]$ ./test_segv
[1]    800759 segmentation fault (core dumped)  ./test_segv
[win32gg@Enceladeus:~/test][139]$
```

DE VINCI INNOVATION CENTER

3| C Syntax & Memory

struct

struct defines contiguous a memory layout of different elements.
structs can be passed in sizeof to know their total size and can be accessed via pointers

```
[win32gg@Enceladeus:~/test]$ bat test.c
File: test.c
1   #include <stdio.h>
2
3   struct mystruct {
4       int a_int;
5       int another_int;
6   };
7
8   int main() {
9       struct mystruct s; // struct is in the stack
10      s.a_int = 42;
11      s.another_int = 43;
12
13      printf("%i\n", s.a_int);
14      printf("%i\n", sizeof(struct mystruct));
15      return 0;
16  }
[win32gg@Enceladeus:~/test]$ ./test_struct
42
8
```

DE VINCI INNOVATION CENTER

3| C Syntax & Memory

struct access

Use “.” to access a struct member

Use ‘->’ when accessing a struct pointer

```
[win32gg@Enceladeus:~/test]$ bat test.c
       File: test.c
   1   #include <stdio.h>
   2
   3   struct mystruct {
   4       int a;
   5       int *b;
   6   };
   7
   8   int main() {
   9       int c = 42;
  10       struct mystruct thestruct; // declare a struct in the stack
  11       thestruct.a = 12;
  12       thestruct.b = &c;   // we set the b pointer to the c address
  13       *thestruct.b = 43;  // we dereference the b pointer and set it
  14
  15       // thestruct_ptr is a pointer to the struct we created at line 10
  16       struct mystruct *thestruct_ptr = &thestruct;
  17       thestruct_ptr->a = 14;  // we use -> for a struct pointer
  18
  19       printf("%i\n%i\n", c, thestruct.a);
  20       return 0;
  21   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_struct
[win32gg@Enceladeus:~/test]$ ./test_struct
43
14
```

DE VINCI INNOVATION CENTER

4| About Memory

Where are my variables ?

- Variables declared outside of any function are in the **.data** segment and have fixed sizes
- Variables declared in a function are in the **stack**
- Using “malloc” from libc you can allocate memory on the **heap**

File: **test.c**

```c
#include <stdio.h>
#include <stdlib.h>

int b; // integer in bss
int data = 1; // integer in .data

int main() {
    int st = 42; // integer in the stack
    int *c = malloc(4); // allocate an integer in heap
    *c = 43; // set the integer in the heap
    return 0;
}
```

DE VINCI INNOVATION CENTER

4| About Memory

Where are my variables ?

- All variables not declared in the stack must have a pre-allocated memory space. (e.g: with malloc)
- **Don’t forget to free your memory**

File: **test.c**

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *c = malloc(4); // allocate an integer in heap
    *c = 43; // set the integer in the heap
    free(c); // IM FREEEEEEEEE
    return 0;
}
```

DE VINCI INNOVATION CENTER

4| About Memory

About the stack

Space is automatically allocated on the stack but

- **it does not grow forever**
- **you data will be out of scope when the function returns.**

```
[win32gg@Enceladeus:~/test]$ bat test.c
       File: test.c
   1   #include <stdio.h>
   2   #include <stdlib.h>
   3
   4   int *func() {
   5       int c = 42;
   6       return &c;
   7   }
   8
   9   int main() {
  10       int *myint = func();
  11
  12       printf("%i\n", *myint);
  13       return 0;
  14   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_scope
test.c: Dans la fonction « func »:
test.c:6:9: attention: la fonction retourne l'adresse d'une variable locale
    6 |   return &c;
      |          ^~
[win32gg@Enceladeus:~/test]$ ./test_scope
[1]    822430 segmentation fault (core dumped)  ./test_scope
[win32gg@Enceladeus:~/test][139]$
```

DE VINCI INNOVATION CENTER

Argument passing

When calling a function, the arguments will be:

- Copied in the stack for primitive types
- Their address are copied in the stack for others

When passing a pointer to a function, it has access to memory targeted by that pointer.

```
[win32gg@Enceladeus:~/test]$ bat test.c
       File: test.c
   1   #include <stdio.h>
   2   #include <stdlib.h>
   3
   4   void func(int a, int *b) {
   5       a = 42;
   6       *b = 43;
   7   }
   8
   9   int main() {
  10       int int1 = 0;
  11       int int2 = 0;
  12
  13       func(int1, &int2);
  14
  15       printf("%i %i\n", int1, int2);
  16       return 0;
  17   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_scope
[win32gg@Enceladeus:~/test]$ ./test_scope
0 43
[win32gg@Enceladeus:~/test]$
```

5| Patterns

Buffering

Allocating a memory space to store arbitrary data (usually before outputting to a file descriptor)
write(2) and **read(2)** work very well with this concept

File: **test.c**

```c
#include <stdio.h>
#include <stdlib.h>

#define BUFFER_SIZE 1024

int main() {
    void *buffer = malloc(BUFFER_SIZE);
    // buffer is ready to be used as a temporary memory location
    // read(2), write(2) requires void* to write the result of the call
    return 0;
}
```

DE VINCI INNOVATION CENTER

Return value via argument pointer

Instead of returning a value, the function directly modifies the data at a given pointer.
This pattern is omnipresent.
“real” return value is used for error handling

NAME top

read - read from a file descriptor

SYNOPSIS top

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

RETURN VALUE top

On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number. It is not an error if this number is smaller than the number of bytes requested; this may happen for example because fewer bytes are actually available right now (maybe because we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or because **read**() was interrupted by a signal. See also NOTES.

On error, -1 is returned, and *errno* is set appropriately. In this case, it is left unspecified whether the file position (if any) changes.

6| Errors

Return value

Generally, functions that handle errors return an **int**

If the return value is
- >= 0 : No error
- -1 : Error, see errno

```
[win32gg@Enceladeus:~/test][130]$ bat test.c
   File: test.c
   1   #include <stdio.h>
   2   #include <stdlib.h>
   3   #include <unistd.h>
   4   #include <sys/types.h>
   5   #include <sys/stat.h>
   6   #include <fcntl.h>
   7
   8   int main() {
   9       int fd = open("thefile", O_RDWR); // open the file with read and write access
  10       if(fd == -1) {
  11           perror("open");
  12           return 1; // return error code 1
  13       }
  14
  15       return 0;
  16   }
[win32gg@Enceladeus:~/test]$ gcc test.c -o test_code
[win32gg@Enceladeus:~/test]$ ./test_code
open: No such file or directory
[win32gg@Enceladeus:~/test][1]$ touch thefile
[win32gg@Enceladeus:~/test]$ ./test_code
[win32gg@Enceladeus:~/test]$ chmod -r thefile
[win32gg@Enceladeus:~/test]$ ./test_code
open: Permission denied
[win32gg@Enceladeus:~/test][1]$
```

6| Errors

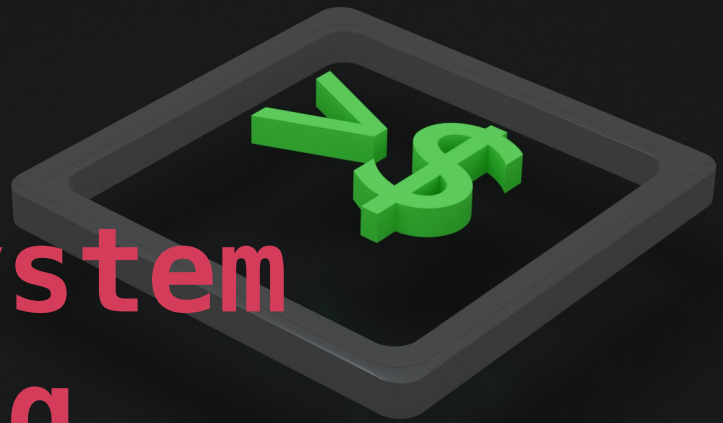
errno

System errors are transmitted by the global variable **errno**
You can #include <errno.h>
to have access to it.
You can print error information vie **perror(3)**

```
ERRORS      top

       EAGAIN The file descriptor fd refers to a file other than a socket
              and has been marked nonblocking (O_NONBLOCK), and the read
              would block.  See open(2) for further details on the
              O_NONBLOCK flag.

       EAGAIN or EWOULDBLOCK
              The file descriptor fd refers to a socket and has been marked
              nonblocking (O_NONBLOCK), and the read would block.
              POSIX.1-2001 allows either error to be returned for this case,
              and does not require these constants to have the same value,
              so a portable application should check for both possibilities.

       EBADF  fd is not a valid file descriptor or is not open for reading.

       EFAULT buf is outside your accessible address space.
```

```
[win32gg@Enceladeus:~/test]$ errno -l
EPERM 1 Opération non permise
ENOENT 2 Aucun fichier ou dossier de ce type
ESRCH 3 Aucun processus de ce type
EINTR 4 Appel système interrompu
EIO 5 Erreur d'entrée/sortie
ENXIO 6 Aucun périphérique ou adresse
E2BIG 7 Liste d'arguments trop longue
ENOEXEC 8 Erreur de format pour exec()
EBADF 9 Mauvais descripteur de fichier
ECHILD 10 Aucun processus enfant
EAGAIN 11 Ressource temporairement non disponible
ENOMEM 12 Ne peut allouer de la mémoire
EACCES 13 Permission non accordée
```

DE VINCI INNOVATION CENTER



Operating System & Programming

Lesson 2 C / **Python** Part #2

Grégor JOUET
PhD Student
gregor.jouet@ext.devinci.fr

LÉONARD DE VINCI PARIS-LA DÉFENSE
ESILV LÉONARD DE VINCI ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE
iiM LÉONARD DE VINCI INSTITUT DE L'INTERNET ET DU MULTIMEDIA PARIS-LA DÉFENSE
EMLV LÉONARD DE VINCI ÉCOLE DE MANAGEMENT PARIS-LA DÉFENSE

DE VINCI INNOVATION CENTER

Python Programming Language

1 Introduction

2 Interpreted language

3 Python Syntax

4 Modules & pip

5 Pythonic way

6 Errors

DE VINCI INNOVATION CENTER

1| Introduction

Python Language

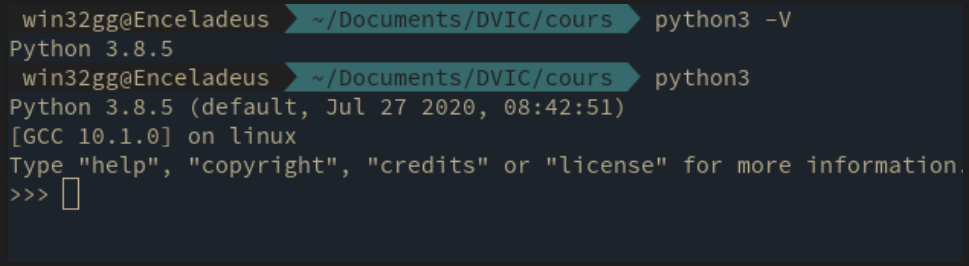
- Created in 1991 by Guido Van Rossum
- Object Oriented, programming language
- Lots of applications
- Can be used as scripting language
- Written in C

DE VINCI INNOVATION CENTER

2| Interpreted language

Python Interpreter

- Python executable reads the .py file and interprets them.
- **.py files never become assembly or machine code.**



DE VINCI INNOVATION CENTER

About Memory

- Automatic typing
- Garbage collected

```
>>> a = 1
>>> a = 'ok'
>>> a = 1.42
>>>
```

```
>>> gc.get_stats()
[{'collections': 13, 'collected': 66, 'uncollectable': 0},
'collected': 0, 'uncollectable': 0}]
>>>
```

3| Python Syntax

Functions

- Use **def**
- Arguments and return are not typed
- Return value can be inconsistent
- Arguments other than raw types are passed by reference (pointer)

File: **a.py**

```
1  def func():
2   return 1 # space is used to put
3            # the statement in the function
4
5  print(func())
```

DE VINCI INNOVATION CENTER

3| Python Syntax

Lists & Dicts

- Very useful data structures
- Lists: Order objects
- Dictionary: Key-Value

File: a.py

```
1  a = []
2  a.append(1)
3  print(a[0])
4
5  a = {}
6  a['a'] = 1
7  print(a['a'])
```

DE VINCI INNOVATION CENTER

3| Python Syntax

Files

- Use **open(filename, flags)**
- Use the **with** statement to automatically close the file.

File: **a.py**

```
1  fh = open("a.py")
2  print(fh)
3  print(fh.read())
```

File: **a.py**

```
1  with open("a.py") as fh:
2      print(fh.read())
```

DE VINCI INNOVATION CENTER

4| Modules and pip

Package manager

- Use **pip** to install new packages
- Visit PyPi or github to find the package you need for you project

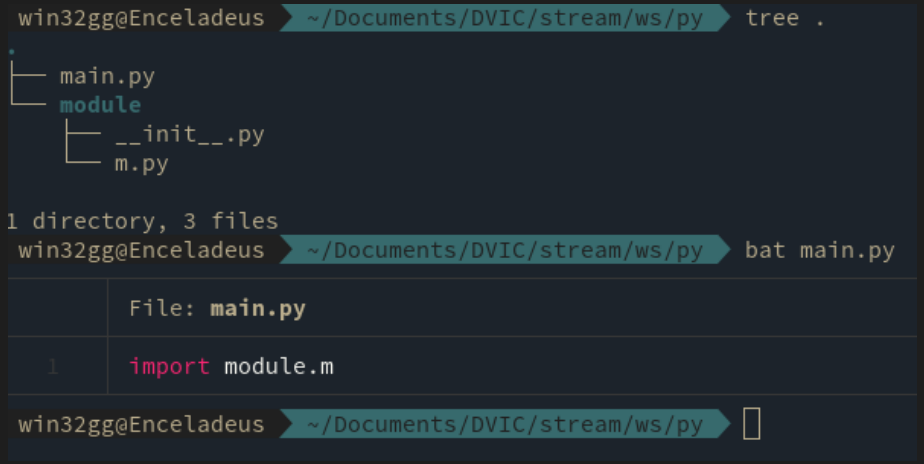
```
 win32gg@Enceladeus  ~/Documents/DVIC/stream/ws  sudo pip3 install numpy
Collecting numpy
  Downloading numpy-1.19.1-cp38-cp38-manylinux2010_x86_64.whl (14.5 MB)
     |████████████████████████████████| 14.5 MB 9.7 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.1
WARNING: You are using pip version 20.1.1; however, version 20.2.2 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
 win32gg@Enceladeus  ~/Documents/DVIC/stream/ws
```

```
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
 win32gg@Enceladeus  ~/Documents/DVIC/stream/ws  sudo pip3 install torch
Requirement already satisfied: torch in /usr/lib/python3.8/site-packages (1.6.0)
Requirement already satisfied: future in /usr/lib/python3.8/site-packages (from torch) (0.18.2)
Requirement already satisfied: numpy in /usr/lib/python3.8/site-packages (from torch) (1.19.1)
WARNING: You are using pip version 20.1.1; however, version 20.2.2 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
 win32gg@Enceladeus  ~/Documents/DVIC/stream/ws  sudo pip3 install keras
Collecting keras
  Downloading Keras-2.4.3-py2.py3-none-any.whl (36 kB)
Requirement already satisfied: numpy>=1.9.1 in /usr/lib/python3.8/site-packages (from keras) (1.19.1)
Requirement already satisfied: pyyaml in /usr/lib/python3.8/site-packages (from keras) (5.3.1)
Requirement already satisfied: scipy>=0.14 in /usr/lib/python3.8/site-packages (from keras) (1.5.2)
Collecting h5py
  Downloading h5py-2.10.0-cp38-cp38-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 13.2 MB/s
Requirement already satisfied: six in /usr/lib/python3.8/site-packages (from h5py->keras) (1.15.0)
Installing collected packages: h5py, keras
Successfully installed h5py-2.10.0 keras-2.4.3
WARNING: You are using pip version 20.1.1; however, version 20.2.2 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
```

DE VINCI INNOVATION CENTER

Modules

- Modules can be imported using **import**
- In your projet, the import is relative to where your python is launched.



DE VINCI INNOVATION CENTER

5| Pythonic Way

Generator

- A generator is a function that keeps its internal state and can return multiple times.
- The next value in a generator is obtained with the **next** function

File: main.py

```python
def gen():
    i = 0
    while True:
        yield i**0.5
        i += 1
```

DE VINCI INNOVATION CENTER

List comprehension

- A Pythonic way to create a List

File: main.py

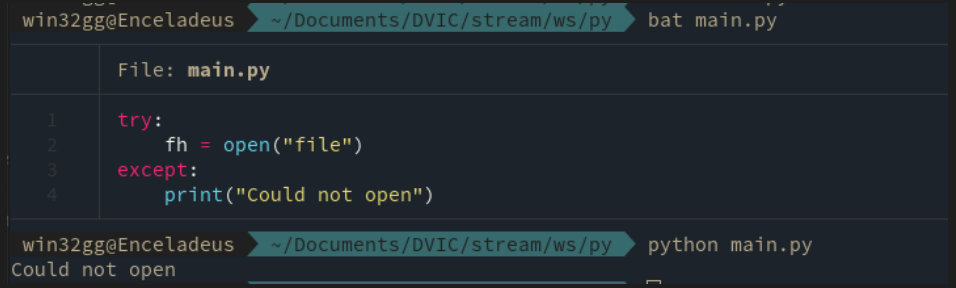
```python
l = [i**2 for i in range(10)]
```

DE VINCI INNOVATION CENTER

6| Errors

Traceback

- Error Handling
- Can be intercepted and cancelled



DE VINCI INNOVATION CENTER