Et si on écrivait un OS ?

Architecture

- BIOS – EFI etc…
- Secteur de boot
- Bootloader
- Système d’exploitation (OS)

Le bootloader

- Dépendant de l’architecture
- Complexe
- Déjà écrit => Grub

Grub1

- Sur un Linux :
  - /boot/grub/menu.lst :

    title monOs

    kernel (hd0,0) /boot/kernel.bin


mageia
Amorcer Mageia 3
Amorcer Mageia 3 (mode sans échec)
monOS
F1 Aide F2 Langue F3 Option du noyau

La compilation

- Un petit bout d'assembleur : NASM
- Du C : gcc
  - -march=i386 -m32
- Edition des liens : ld
  - -oformat=elf32-i386 -m elf_i386

Charger depuis le bootloader

```
[extern kernel_start]
[global kernel_stack]

[global main]

STACKSIZE equ 0x4000 ; taille de la pile

[section .text]
align 4
main:
 mov esp, kernel_stack + STACKSIZE
 mov eax, kernel_start
 jmp eax

[section .bss]
align 32
kernel_stack:
resb STACKSIZE ; ici on aura la pile
```

```
ENTRY (main)
SECTIONS {
    . = 0x0100000;
    .boot :
    {
        *(.boot);
    LONG(0)
    }
    _start_of_kernel = .;
    .text :
    {
        *(.text)
    }
    .rodata ALIGN (0x1000) :
    {
        *(.rodata)
    }
    .data ALIGN (0x1000) :
    {
        *(.data)
    }
    _end_of_load = .;
    .bss :
    {
        _sbss = .;
        *(COMMON)
        *(.bss)
        _ebss = .;
    }
    _end_of_kernel = .;
}
```

Puis un main !

```
void kernel_start(unsigned long magic, unsigned long addr) {
    while(1);
}
```

L’affichage à l’écran

- On commence à l’adresse 0xB8000
- Aux adresses paires le caractère, à l’adresse suivante les attributs (1 octet) couleur, couleur de fond et clignotement

L'affichage à l'écran

```
#define NOIR  0
#define BLEU  1
#define VERT  2
#define CYAN  2
#define ROUGE 4
#define GRIS 8
#define BLANC 15

#define BG_NOIR  (0<<4)
#define BG_BLEU  (1<<4)
#define BG_VERT  (2<<4)
#define BG_ROUGE (4<<4)
```

```
#define start_video_mem 0xB8000
#define width   80
#define lines    25
//
static short aff= BG_NOIR | BLANC;
static int x, y = 0;
```

```
static volatile video_mem *ecran = (volatile video_mem *)start_video_mem;
```

```
//
// affchar : Display character at current position
//-----------------------------------------------
void affchar(char caractere){
        long offset = x+y*width;
    (*ecran)[offset].caractere =caractere ;
    (*ecran)[offset].attr = aff;

}
```

L’affichage à l’écran

```
//
// Print
//-----------------------------------------------
void print(char* chaine){
    char* pt; /* pointeur sur la chaine de caractere */
    pt = chaine;
    while(*pt!=0x0){ /* tant que le caractere est different de 0x0 */
            if(*pt == '\n'){
            move_to_next_line();
        }else if (*pt == '\t'){
            move_to_next_char();
            move_to_next_char();
            move_to_next_char();
            move_to_next_char();
            move_to_next_char();
        }else if (*pt == '\r'){
            x=0;
        }else{
            affchar(*pt);
            move_to_next_char();
        }
        pt++;
    }
}
```

Du coup…

```
#include "types.h"
#include "video.h"

void kernel_start(unsigned long magic, unsigned long addr) {
    cls();
    changecouleur(BG_NOIR |CYAN);
    print("Mon OS\n");

    changecouleur(BG_NOIR | BLANC);

    print("Entering Infinite loop\n");


    while(1);
}
```

TADA !!