Projet Rock'n Roll : *MarioBrOS*

Florentin GUTH   Lionel ZOUBRITZKY

13 mai 2017

Table des matières

1 Truc

1.1 Truc2

On a du code `int main() { return 0; }`

ata_pio.c[0-20]

```c
#include "ata_pio.h"


unsigned char disk_id = 8;


/**
 *  @name poll - Waits for the drive to be ready to transfer data
 *  @return  0 - No error
 *           1 - ERR (Error) bit set
 *           2 - DF (Drive Fault) bit set
 */
char poll()
{
  u_int8 status = inb(ATA_COMMAND);
  while(status & ATA_BSY) {
    status = inb(ATA_COMMAND);
  }
  if(status & ATA_ERR) { return 1; }
  if(status & ATA_DF ) { return 2; }
```

Listing 1 – Disque

loader.s[0-50]

```
MBOOT_PAGE_ALIGN    equ 1<<0     ; Load kernel and modules on a page boundary
MBOOT_MEM_INFO      equ 1<<1     ; Provide your kernel with memory info
MBOOT_HEADER_MAGIC  equ 0x1BADB002 ; Multiboot Magic value
MBOOT_HEADER_FLAGS  equ MBOOT_PAGE_ALIGN | MBOOT_MEM_INFO
MBOOT_CHECKSUM      equ -(MBOOT_HEADER_MAGIC + MBOOT_HEADER_FLAGS)

KERNEL_STACK_SIZE   equ 0x1000   ; Define a stack of one page (4KB)


section .bss                     ; Uninitialized data section
align 4                          ; Align at 4 bytes
kernel_stack:                    ; Label points to beginning of memory
  resb KERNEL_STACK_SIZE         ; Reserve stack for the kernel

KERNEL_STACK_START  equ kernel_stack + KERNEL_STACK_SIZE

section .text                    ; Code section
align 4                          ; 4 byte-aligned code

global mboot                     ; The multi-boot header is accessible from C
extern ld_code
extern ld_bss
extern ld_end

mboot:
  dd MBOOT_HEADER_MAGIC          ; Write the magic number to the machine code,
  dd MBOOT_HEADER_FLAGS          ; The flags,
  dd MBOOT_CHECKSUM              ; And the checksum

  dd  mboot                      ; Location of this descriptor
  dd  ld_code                    ; Start of kernel '.text' (code) section.
  dd  ld_bss                     ; End of kernel '.data' section.
  dd  ld_end                     ; End of kernel.
  dd  loader                     ; Kernel entry point (initial EIP).


global loader                    ; The entry symbol for ELF (Executable and Linkable Format)
extern kmain                     ; The kmain function is not defined her (in kmain.c)

loader:                          ; The loader label (defined as entry point in the linker script)
  mov eax, 0xDEADBEEF            ; Place whatever we want in the register eax
  mov esp, KERNEL_STACK_START    ; Points esp to the start of the stack (end of memory area)

  ; Push eventual arguments to the stack, from last to first
  push KERNEL_STACK_SIZE
  push KERNEL_STACK_START
  push ebx                       ; Load multi-boot header location, which have been set up by GRUB
  call kmain                     ; Call the kmain function from kmain.c (return in eax)
  cli                            ; Prevents further interruptions

```

Listing 2 – Loader