**1516 L2 SEPS TDTP 1 - Boot Secteur**

1. **Connexion**
2. **Installation**
   1. **Installation Bochs:**

₪ - Qu'est-ce la commande **"sudo"** ?

Elle sert à acquérir momentanément (le temps d’une commande) en mode administrateur.

₪ - Qu'est-ce la commande **"apt-get"** ? Pourquoi l'option **install** est-elle nécessaire ?

C’est un gestionnaire de paquet et avec l’option install, il installe les dépendances et l’élément choisis par lui-même sans avoir besoin d’une intervention utilisateur.

₪ - Expliquez chaque ligne du fichier de paramétrage de **Bochs** ci-dessus.

|  |  |
| --- | --- |
| megs: 32 | Quantité de par défaut de mémoire. |
| romimage: file="/usr/share/bochs/BIOS-bochs-latest" | Le chemin d’accès à l’image du BIOS, Boschs en fourni un de base.  (On utilise l’image du BIOS de Boschs) |
| vgaromimage: file ="/usr/share/bochs/VGABIOS-lgpl-latest" | Le chemin d’accès à l’image d’un BIOS VGA, Boschs en fourni un de base.  (On utilise l’image du BIOS de Boschs) |
| floppya: 1\_44=JmOsFloppy.img, status=inserted | Le chemin de l’image d’une disquetteou d’un périphérique. |
| display\_library: sdl | Libraire d’affichage. |
| boot: a | Périphérique de démarage. (a ou c) |
| log: bochsout.txt | Le chemin d’accès du fichier de Log. |
| mouse: enabled=0 | Activation de la souris **(on = 1 & off = 0)** |
| clock: sync=realtime | Synchronisation de la clock du BIOS. |
| cpu: ips=500000 | Nombre d’instruction seconde. |

* 1. **Installez NASM:**

Exécutez la commande:

**sudo apt-get install nasm**

₪ - Qu'est-ce que nasm ?

Il sert à « traduire » un fichier .asm en un autre format.

1. **TD/TP**
   1. Ouvrez le code le code du boot secteur qui vous est donné sur Espadon (fichier "bootSect.asm")

₪ - Modifiez les noms du SE (iOEM et acVolLabel) à votre nom. Attention à respecter le nombre de caractères.

***iOEM: db "Stegt " ; OEM String***

₪ - Modifiez également le message de boot loadmsg à votre nom.

***loadmsg: db "Texte A Modifier",0Dh,0Ah,0***

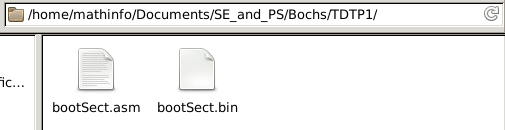
* 1. Compiler le code du boot secteur qui vous est donné sur Espadon (fichier "bootSect.asm")   
       
     Exécutez la commande:

**nsam –f bin –o bootSect.bin bootSect.asm**

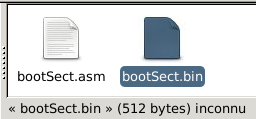
₪ - Qu'est-ce que l'option "–f" et la "–o" ?

* + 1. –f désigne le format
    2. –o désigne le nom du fichier

₪ - Quel est le format de sortie du fichier après compilation ?



₪ - Vérifiez que votre fichier binaire fait bien 512 octets exactement.



* 1. Créez une disquette virtuelle avec votre secteur de boot:

Exécutez la commande:

cat bootsect.bin /dev/zero | dd of=JmOsFloppy.imgbs=512 count=2880

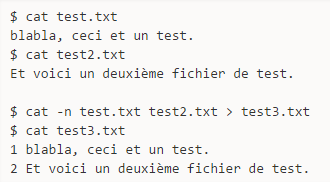
₪ - Qu'est-ce que la fonction dd ?

La commande dd permet de copier tout ou partie d'un disque par blocs d'octets, indépendamment de la structure du contenu du disque en fichiers et en répertoires.

₪ - Qu'est-ce que la fonction "cat" ?

Elle sert le plus souvent à lire un contenu textuel, mais elle peut faire plus. Son nom est dérivé de l'Anglais “concatenate”, “concaténer” en Français. (Linuxpédia) La commande cat permet aussi de concaténer deux fichiers en un seul comme utilisé plus bas.

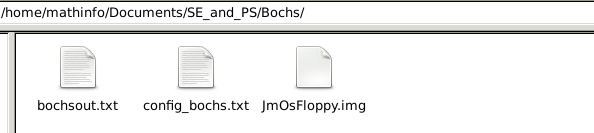
Par exemple :



₪ - A quoi sert le symbole "|"?

Ce symbole permet de rediriger la sortie d’une commande vers une autre commande sous cette forme : commande |commande.

₪ - Vérifiez que l'image disquette a été créée.



* 1. Modifiez "config\_bochs.txt" pour booter sur votre image disquette.

Modifiez la ligne de paramètres adéquate:

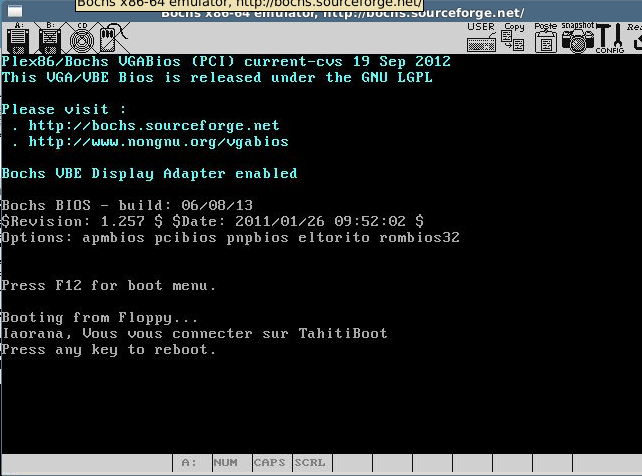
floppya: 1\_44=JmOsFloppy.img, status=inserted

* 1. Lancer Bochs :

Exécutez la commande:

bochs -f config\_bochs.txt

₪ - Produisez une sortie d'écran du résultat du boot:



PS : Je m’escuse pour la faute (connecter ⬄ connectez) …

₪ - Pourquoi le code utilisé et tiré de la page

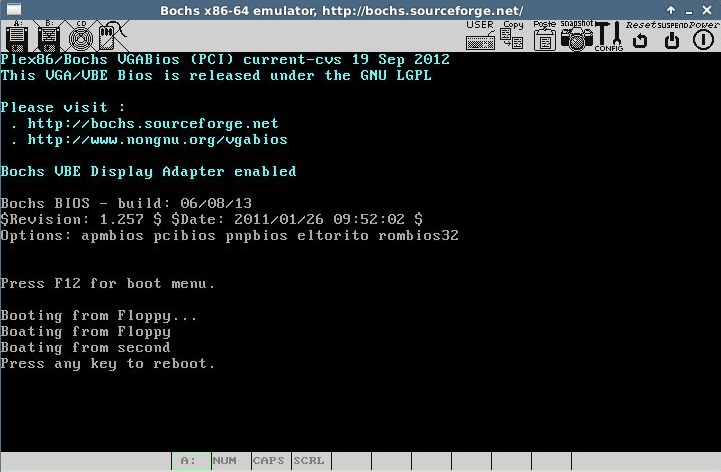
« http://www.websofia.com/2011/10/writing-yourown-bootloader-for-a-toy-operating-system-2/ » est-il plus "intéressant" en premier lieu que le code proposé dans la page "http://a.michelizza.free.fr/pmwiki.php?n=TutoOS.Intro" ?

Le code est surement plus proche de ce que l’on connait au point de vue programmation.

₪ - Pourquoi ces deux codes Assembleur (sur les pages citées) sont-ils différents ?

Un est écrit dans un langage différent avec la même base (assembleur), il se nomme GNU assembleur. La principale différence est que le langage Gnu asm est un langage qui doit être préalablement traduit / compiler avant de pouvoir être utilisé.

₪ - Essayez de modifier le code pour afficher un second message de boot (de votre choix).



₪ - À quoi servent les nombres "0Dh, 0Ah, 0" placés à la fin des chaines de caractères?

les valeurs ‘0Dh’=13, ‘0Ah’=10 correspondent aux codes ASCII commandant le saut a la ligne ( Carriage return et Line Feed) et le ‘0’ marque la fin de la chaine de caractère nécessaire pour la fonction **WriteLine**.

1. **Suite**

J’ai utilisé le code source de la page fournie mais en le commentant et en essayant de le personnalisé pour le comprendre.

Désolé de ne pas avoir pris de screenshot.