Xavier Pajani p0601344

Cadre Photo Numérique Debian

I. Création système

Le but de ce projet est de créer une clé usb qui sera bootable et offrira les services d'un cadre photo numérique. Pour ce faire, je suis parti d'une Debian Squeeze à l'aide de debootstrap.
Tout d'abord, formater la clé usb en ext3/4 à l'aide de fdisk et mkfs et créer deux partitions (une système et l'autre pour les photos).

```
fdisk /dev/sdX puis mkfs.ext3 /dev/sdXY
```

Monter la partition système dans un répertoire de travail : `mount /dev/sdb1 /mnt/usb`

Effectuer un debootstrap : `debootstrap squeeze /mnt/usb` (Il n'est pas forcément judicieux de le faire directement sur la clé en cas d'erreur I/O !!!)

Il faut ensuite monter des dossiers utiles pour éviter des messages d'erreurs en chaine et utiles pour Grub par exemple

```
mount --bind /dev /mnt/usb/dev
mount --bind /proc /mnt/usb/proc
mount --bind /sys /mnt/usb/sys
mount --bind /tmp /mnt/usb/tmp
```

Enfin, on change la racine de travail avec chroot.

```
chroot /mnt/usb
```

Le système est donc prêt pour les modifications. Il est temps de faire un Backup !

II. Backup

Je vois deux solutions :

1 – DD
Long et prend autant de place que la taille de la clé, mais facile à utiliser.

```
dd if=/dev/sdX of=/home/sauvegarde_after_debootstrap
```

2 – tar.bz2
Si la clé à un problème, il faut la reformater voir recréer les partitions, mais prend peu de place sur le système

```
cd /mnt/usb
tar cjf /home/sauvegarde_after_debootstrap.tar.gz ./*
```

Il est ensuite possible de lire l'image faite avec dd avec losetup. Pour cela, il faut connaitre la taille des blocs et où commence la partition. Ensuite faire un savant calcul TailleBloc*1erBlocPartition=*indice*

Puis effectuer les commandes suivantes par exemple:

```
losetup –o indice /dev/loopX  /home/sauvegarde_after_debootstrap
mount /dev/loopX /mnt/usb/
chroot /mnt/usb/
```

III. Les outils de travail

Serveur Web : lighttpd (léger, rapide et plus facile à utiliser que httpd de busibox)
SSH : dropbear (un client ssh léger pour prendre la main sur la machie)
Lire des images en Frame Buffer : Frame Buffer Image viewer
Programme d'amorçage : GRUB (car je savais l'utiliser mais je ne le recommande pas, voir avec les alternatives plus légères)

Pour installer ces applications, il suffit de faire :

apt-get update
apt-get install lighttpd dropbear fbi grub2 libjpeg8

IV. Diffuser des images en Frame Buffer

Il faut configurer grub et activer le framebuffer de la manière suivante

grub-install /dev/sdX

éditer le fichier /etc/default/grub :

```
GRUB_GFXPAYLOAD=keep
GRUB_GFXPAYLOAD_LINUX=800x600
```

Lancer un `update-grub`
Il est maintenant possible de tester la clé et de faire une sauvegarde.

V. Adapter FBI

Dans le sujet, il est demandé de faire un diaporama, afficher l'heure sur les images et récupérer des flux de webcam.
S'il est possible de faire un diaporama directement avec fbi, l'affichage de l'heure et la mise à jour du cache pour les webcams ne sont pas inclus. Il faut donc modifier le code source de fbi.

Pour afficher la date :

```
static void show_date(struct flist *f)
{
    char mabuf[256];
    int yt = fb_var.yres + (face->size->metrics.descender >> 6);
    wchar_t str[128];

    status_prepare();

    time_t timestamp = time(NULL);

    strftime(mabuf, sizeof(mabuf), "%Y:%m:%d %X", localtime(&timestamp));

    swprintf(str,ARRAY_SIZE(str), L"%s", mabuf);
    shadow_draw_string(face, 0, yt, str, -1);

    shadow_render();
    //status_error(mabuf);
}
```

Puis, il suffit de fonctionner comme avec les autres touches de fbi et de « binder » la touche O ou o qui inverse le statut de l'affichage de la date. L'avantage de cette technique est que l'on ne perd pas de fonctionnalités de fbi, et qu'en plus, cette option fonctionne même sur la webcam ☺

Pour reload le cache pour la webcam :

```
if (GET_RELOAD()){
    flist_img_free(fcurrent);
    fcurrent->seen = 0;
    snprintf(linebuffer, sizeof(linebuffer), "RELOADING %s", fcurrent->name);
    status_update(linebuffer, NULL);
}
```

Puis il suffit de rajouter une option de lancement :

```
#define GET_RELOAD()        cfg_get_bool(O_RELOAD,        0)
```

```
},{
.cmdline  = "reload",
.option   = { O_RELOAD },
.yesno    = 1,
.desc     = "Reload cache",
},{
```

VI. CGI

Ce script tourne en arrière-plan afin de retenir le choix de l'utilisateur. (mis dans /etc/inittab ligne 1:2345:respawn:/home/scripts/switch.sh à la place de 1:2345:respawn:/sbin/getty 38400 tty1)

```
#! /bin/bash
old="w"

echo "p" > /home/scripts/choix.txt

while true
do

choix=$(cat /home/scripts/choix.txt)
    if [ "$choix" != "$old" ] ; then
        old=$choix
        /home/scripts/kill.sh
        if [ "$choix" = "w" ] ; then
            /home/scripts/webcam_lan.sh
        fi
        if [ "$choix" = "p" ] ; then
            /home/scripts/photos.sh
        fi
    fi
sleep 1
done
```

Le script CGI pour changer de choix.

```
#!/bin/bash

echo "Content-Type: text/html\n"

photo=""
webcam=""
if [ "$QUERY_STRING" = "choix=webcam" ]
then
    webcam="checked=\"checked\""
    if [$(cat /home/scripts/choix.txt) = "p"]
    then
        /home/scripts/kill.bin
        echo "w" > /home/scripts/choix.txt
    fi
fi
if [ "$QUERY_STRING" = "choix=photo" ]
then
    photo="checked=\"checked\""
    if [$(cat /home/scripts/choix.txt) = "w"]
    then
        /home/scripts/kill.bin
        echo "p" > /home/scripts/choix.txt
    fi
fi

echo "<html>"
echo "<form action=\"cgi.sh\" methode=\"GET\">"
echo "<fieldset><legend>Pour changer l'affichage, merci de faire un choix et de valider :)</legend>"
echo "<input type=\"radio\" name=\"choix\" value=\"photo\" $photo > PHOTOS"
echo "<input type=\"radio\" name=\"choix\" value=\"webcam\" $webcam > WEBCAM"
echo "<input type=\"submit\" value=\"Changer!\">"
echo "</fieldset>"
echo "</form>"
echo "</html>"
```

Une modification a dû être apportée pour faire fonctionner la webcam. En effet, le script de lancement ne rend pas la main lorsque l'option –reload est passée à fbi. De ce fait, il faut tuer fbi dans le cgi grâce à un programme C qui contient un appel système kill. Pour pallier le problème de droits, il suffit d'apposer un stickybit au binaire : chmod u+s kill.bin

ATTENTION : L'option –T 1 est une coquille d'une vielle version de fbi, elle ne doit donc plus être utilisée.
Script de lancement de fbi photos :

```
#!/bin/sh

fbi -t 2 -T 1 -a -noverbose /home/photos/*
```

Script de lancement de fbi webcam_lan :

```
#!/bin/sh

fbi -a -noonce -t 2 -T 1 -noverbose -reload http://10.0.0.1/~jpgelas/photo1.jpg
```

Script de lancement de fbi webcam internet :

```
#!/bin/sh

fbi -noonce -t 2 -T 1 -a -noverbose -reload http://metz.fr/webcam_images/image_pa.jpg
```

Script de lancement du kill :

```
#! /bin/bash

pkill fbi
```

VII. Alléger le système

- Compilation d'un nouveau noyau 2.6.32-5
- Suppression de l'ancien noyau
- Amélioration des scripts de lancement de /etc/rcS.d
- Suppression de paquets non utiles :
  apt-get remove --purge
  cpio cpp-4.4 dmidecode exiv2 fim gcc-4.4 info initramfs-tools keyboard-configuration klibc-utils libaa1 libc-dev-bin libc6-dev libcap2 libdjvulibre-text libdjvulibre21 libexiv2-9 libgmp3c2 libgomp1 libjpeg8-dev libklibc libmpfr4 libuuid-perl libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 linux-base linux-image- .6.32-5-686 linux-libc-dev logrotate make man-db manpages manpages-dev netcat-traditional ntpdate rsyslog traceroute vim vim-common vim-runtime xkb-data

  suppression de ce qui prend encore de la place pour rien ☺
- /usr/share/man
- /usr/share/man8
- /usr/share/doc
- /usr/share/locale/.... (pas tout)
- /usr/share/bug/
- /usr/share/i18n/locales/.... (pas tout)
- /usr/share/info/
- /usr/share/zoneinfo/... (pas tout)

Attention, grub2 a besoin de plus que /boot/grub/grub.cfg pour le frame buffer. Il ne faut donc pas le supprimer ou alors changer de bootloader.

Avec tout ça, on obtient un système Debian de 209Mo soit une archive de 65.1Mo