(Traduction française : Nicolas Jadot (njadot@etu.info.unicaen.fr))

SRM Firmware Howto

v0.5, 17 August 1996

Ce document décrit la manière de démarrer une station Linux/Alpha utilisant le *firmware* SRM, lequel est normalement utilisé pour démarrer Digital Unix. (NDT : le *firmware* est un micro-code embarqué sur une puce, en quelque sorte l'équivalent du BIOS des PC)En général, il est préférable d'utiliser MILO à la place du programme `aboot` sachant que MILO est parfaitement adapté aux besoins de Linux. Cependant, MILO n'existe pas sur tous les systèmes et ne permet pas encore de démarrer sur un réseau. Dans ces cas là, utiliser la console SRM est peut-être la bonne solution.

Contents

A moins que vous soyiez intéressés par les détails techniques, vous pouvez passer directement à la section 3 ()

1 Comment SRM démarre-t-il un système d'exploitation ?

Toutes les versions de SRM peuvent démarrer à partir d'un disque SCSI et les versions destinées aux plates-formes récentes, comme les Noname et AlphaStations, peuvent de plus démarrer depuis une disquette. Le démarrage réseau via `bootp` est également supporté. Notez que les anciennes versions de SRM (notamment celles pour Jensen) *ne pouvaient pas* démarrer depuis une disquette. Le démarrage depuis un disque IDE n'est pas supporté.

Le démarrage de Linux avec SRM s'effectue en deux étapes : d'abord, SRM charge et transfère le controle à un chargeur secondaire. Ensuite, ce chargeur secondaire met en place l'environnement de Linux, lit l'image du noyau depuis un système de fichiers sur disque et donne la main à Linux.

A l'heure actuelle il existe deux chargeurs secondaires pour Linux : le chargeur *brut* fourni avec le noyau Linux et `aboot`, distribué séparement. Ces deux chargeurs sont décrits en détails ci-dessous.

1.1 Charger le chargeur secondaire

SRM est ignorant des systèmes de fichiers et des partitions des disques. Il s'attend simplement à ce que le chargeur secondaire occupe un certain nombre de secteurs physiques consécutifs, commençant à une adresse donnée. L'information concernant la taille du chargeur secondaire et l'adresse de son premier secteur est stockée dans les 512 premiers octets. Plus précisement , l'entier long occupant l'adresse 480 contient la *taille* du chargeur secondaire et celui occupant l'adresse 488 donne le *nombre de secteurs* de décalage jusqu'au début du chargeur. Le premier secteur contient également un drapeau à l'adresse 496 qui vaut toujours 0 et une somme de contrôle à l'adresse 504. Cette somme de contrôle est simplement la somme des 63 premiers entiers longs du premier secteur.

Si cette somme est correcte, SRM continue et lit le nombre de secteurs indiqué par la *taille* en commençant par celui indiqué dans le champ *numéro de secteur* et les place en mémoire *virtuelle* à l'adresse `0x20000000`. Si la lecture se termine avec succés, SRM effectue un saut à l'adresse `0x20000000`.

2 Le chargeur brut

Les sources de ce chargeur peuvent être trouvés dans le répertoire

```
linux/arch/alpha/boot
```

des sources du noyau Linux. Ce programme charge le noyau Linux en lisant `START_SIZE` octets en commençant à l'adresse `BOOT_SIZE+512` (également en octets). Les constantes `START_SIZE` et `BOOT_SIZE` sont définies dans le fichier d'en-tete `linux/include/asm-alpha/system.h`. `START_SIZE` doit être au moins aussi élévé que la taille de l'image du noyau (i.e, la somme des tailles des segments `.text`, `.data`, et `.bss`). De même, `BOOT_SIZE` doit être au moins aussi élevé que l'image du chargeur. Ces deux constantes doivent avoir comme valeur un multiple entier de la taille d'un secteur, soit 512 octets. Les valeurs par défaut sont 2Mo pour `START_SIZE` et 16Ko pour `BOOT_SIZE`. Notez que si vous voulez démarrer depuis une disquette de 1.44Mo, vous devez réduire `START_SIZE` à 1400Ko et vous assurer que la taille du noyau que vous voulez démarrer ne dépasse pas cette valeur.

Pour construire un chargeur brut, tapez simplement `make rawboot` dans `/usr/src/linux`. Ceci devrait produire dans `arch/alpha/boot` les fichiers suivants :

`tools/lxboot:`

Le premier secteur du disque. Il contient l'adresse et la taille du prochain fichier au format décrit ci-dessus.

`tools/bootlx:`

Le lanceur brut qui chargera le fichier ci-dessous

`vmlinux.nh:`

L'image brute du noyau constituée des segments `.text`, `.data` et `.bss` du fichier objet `/usr/src/linux/vmlinux`. L'extension `.nh` indique que ce fichier n'a pas l'entête d'un fichier objet.

La concaténation de ces trois fichiers devrait être écrite sur le disque à partir duquel vous voulez démarrer. Par exemple, pour démarrer depuis une disquette, insérez une disquette vierge dans le lecteur, soit `/dev/fd0` et ensuite tapez :

```
cat tools/lxboot tools/bootlx vmlinux >/dev/fd0
```

Vous pouvez maintenant arréter le système et démarrer depuis une disquette en utilisant la commande `boot dva0`.

3 Le chargeur `aboot`

Si vous utilisez le *firmware* SRM, `aboot` est la meilleure façon de démarrer Linux. Il supporte :

- démarrage direct depuis divers systèmes de fichiers (`ext2`, `ISO9660`, et `UFS`, le système de fichiers de Digital Unix),
- lancement de fichiers objets exécutables (ELF et ECOFF),
- lancement de noyaux compressés,
- démarrage par réseau (en utilisant le protocole `bootp`),

- table de partitions au format Digital Unix (compatible avec les tables de partitions de BSD),
- démarrage interactif et configurations par défaut des consoles SRM qui n'acceptent pas les longues chaînes d'option.

3.1 Se procurer et installer aboot

Les codes sources les plus récents d'`aboot` sont disponibles à l'adresse *ftp://ftp.azstarnet.com/pub/linux/axp/aboot* `ftp://ftp.azstarnet.com/pub/linux/axp/aboot` . La description de ce manuel s'applique à `aboot` pour les versions 0.5 et suivantes.

Une fois que vous avez téléchargé et extrait l'archive `tar`, jetez un oeil aux fichiers `README` et `INSTALL` pour lire les directives d'installation. En particulier, assurez vous que les variables, dans les fichiers `Makefile` et `include/config.h` sont correctes vis-à-vis de votre environnement . Normalement, vous ne devriez pas avoir à changer quoi que ce soit pour compiler sous Linux, mais c'est toujours une bonne chose de vérifier. Si la configuration vous convient, tapez simplement `make` pour lancer la compilation (si vous n'effectuez pas cette opération sous Linux, sachez que `aboot` requiert GNU `make`).

Après l'exécution de `make`, le répertoire `aboot` devrait contenir les fichiers suivants :

aboot

L'exécutable réel (fichier objet ECOFF ou ELF),

bootlx

Comme ci-dessus, mais ce fichier ne contient que les segments text, data et bss (ce fichier n'est pas un fichier objet),

sdisklabel/writeboot

Un utilitaire pour installer `aboot` sur un disque dur,

tools/e2writeboot

Un utilitaire pour installer `aboot` sur un système de fichiers ext2 (n'est en général utilisé que pour les disquettes),

tools/isomarkboot

Un utilitaire pour installer `aboot` sur un système de fichiers iso9660 (utilisé par les distributeurs de CD-ROM),

tools/abootconf

Un utilitaire pour configurer `aboot` s'il est installé.

3.2 Installation sur disquette

Le lanceur peut être installé sur une disquette en utilisant la commande `e2writeboot` (note : ceci ne peut se faire sur un Jensen car son *firmware* n'implante pas le démarrage depuis une disquette). Cette commande nécessite que le disque ne soit pas trop fragmenté car elle a besoin de trouver suffisament de secteurs contigus pour stocker l'image entière de `aboot` (actuellement, environ 90Ko). Si `e2writeboot` échoue à cause de ça, reformatez la disquette et réessayez (par ex., avec `fdformat(1)`). Par exemple, la procédure suivante installe `aboot` sur une disquette en supposant que la disquette est dans le lecteur correspondant à `/dev/fd0` :

```
fdformat /dev/fd0
mke2fs /dev/fd0
e2writeboot /dev/fd0 bootlx
```

3.3 Installation sur disque dur

Sachant que la commande `e2writeboot` peut échouer sur un disque hautement fragmenté et comme le reformattage d'un disque dur ne se fait pas sans peine, il est généralement plus sûr d'installer `aboot` sur un disque dur en utilisant la commande `swriteboot`. `swriteboot` nécessite que les premiers secteurs soient réservés aux procédures de démarrage. Nous suggérons que le disque soit partitionné de manière à ce que la première partition commence à une adresse correspondant à 2048 secteurs. Cela laisse 1Mo d'espace ibre pour stocker `aboot`. Sur un disque partitionné de cette façon , il est alors possible d'installer `aboot` comme décrit ci-dessous (en supposant que le disque correspond à `/dev/sda`.) :

```
swriteboot /dev/sda bootlx
```

Sur un Jensen, vous devrez laisser un peu plus d'espace, sachant que vous devrez également stocker le noyau à cet endroit - 2Mo devraient suffire en utilisant une image compressée. Utilisez `swriteboot` comme décrit à la section 3.6 () pour écrire `bootlx` avec le noyau Linux.

3.4 Installation sur CD-ROM

Pour construire un CD-ROM amorçable avec SRM, construisez simplement `aboot` comme décrit ci-dessus. Assurez-vous ensuite que le fichier `bootlx` est présent sur le système de fichiers iso9660 (e.g., copiez `bootlx` dans le répertoire où est monté le système de fichiers) , et lancez `mkisofs` sur ce répertoire). Après cela, la seule chose restant à faire est de marquer le système de fichiers comme amorçable avec SRM. Cela est réalisé grâce à une commande de la forme :

```
isomarkboot filesystem bootlx
```

La commande ci-dessus nécessite que `filesystem` est un fichier contenant le système de fichiers iso9660 et que `bootlx` a été copié dans la racine de ce système de fichiers. C'est tout !

3.5 Construire un noyau Linux

Un noyau Linux amorçable peut être construit par les étapes suivantes. Durant le `make config`, assurez-vous de répondre "oui" (*"yes"*) à la question concernant le lancement du noyau par SRM.

```
cd /usr/src/linux
make config
make dep
make boot
```

La dernière commande construira le fichier `arch/alpha/boot/vmlinux.gz` qui peut alors être copié sur le disque à partir duquel vous désirez démarrer. Dans notre exemple précédent concernant la disquette, cela donnerai :

```
mount /dev/fd0 /mnt
cp arch/alpha/boot/vmlinux.gz /mnt
umount /mnt
```

3.6 Démarrer Linux

Avec le *firmware* SRM et `aboot` installé, le démarrage de Linux s'effectue généralement avec une commande de la forme :

```
boot devicename -fi filename -fl flags
```

Les arguments *filename* et *flags* sont optionels. S'ils ne sont pas spécifiés, SRM utilise les valeurs par défaut contenues dans les variables d'environnement `BOOT_OSFILE` et `BOOT_OSFLAGS`. La syntaxe et la signification de ces deux arguments est décrite plus en détail ci-dessous.

3.6.1 Nom du fichier boot

L'argument *filename* est de la forme :

[*n*/]*filename*

n est un simple nombre dans l'intervalle 1..8 qui donne le numéro de la partition de démarrage. *filename* est le chemin d'accés au fichier à lancer. Par exemple, pour démarrer depuis la deuxième partition du sixième disque SCSI, vous entreriez :

```
boot dka600 -file 2/vmlinux.gz
```

Ou, pour démarrer depuis le premier lecteur de disquette :

```
boot dva0 -file vmlinux.gz
```

Si un disque n'a pas de table des partitions, `aboot` considère que le disque contient une partition `ext2` commençant au premier bloc du disque. Cela permet de démarrer depuis une disquette.

Le numéro de partition 0 est utilisé pour demander le démarrage depuis un disque qui ne contient pas (encore) de système de fichiers. Si l'on spécifie le numéro de "partition" 0, `aboot` considère que le noyau Linux suit directement l'image de `aboot`. Une telle chose peut être réalisée avec la commande `swriteboot`. Par exemple, pour configurer un démarrage sans système de fichiers depuis `/dev/sda`, on pourrait utiliser la commande :

```
swriteboot /dev/sda bootlx vmlinux.gz
```

Démarrer un système de cette façon n'est pas obligatoirement nécessaire. La raison d'être de cette fonctionnalité est de permettre l'installation de Linux sur un système qui ne peut démarrer depuis une disquette (e.g., le Jensen).

3.6.2 Drapeaux de démarrage

Plusieurs drapeaux de démarrage peuvent être spécifiés. La syntaxe en est :

```
-flags "options..."
```

Où "options..." est une combinaison des options suivantes (séparées par des espace). Il y a encore plus d'options, en fonction des pilotes que le noyau a installé. Les options listées ci-après ne sont là que pour illustrer l'idée générale :

load_ramdisk=1

Copie le système de fichiers racine depuis une disquette vers un disque virtuel en mémoire avant de lancer le système. Ce disque virtuel sera utilisé en lieu et place du périphérique racine. Ceci est utile pour démarrer Linux sur une machine qui ne possède qu'un lecteur de disquettes.

floppy=*str*

root=*dev*

Sélectionne le périphérique *dev* comme système de fichiers racine. Le périphérique peut être spécifié comme la combinaison des numéros *major/minor* du fichier de périphérique en hexadécimal (e.g., 0x802 pour /dev/sda2) ou un nom de fichier de périphérique (e.g.,`/dev/fd0`, `/dev/sda2`).

single

Lance le système en mode mono-utilisateur.

kgdb

Autorise *kernel-gdb* (ne fonctionne que si `CONFIG_KGDB` est activé ; un deuxième système Alpha doit être connecté par voie série pour que cela fonctionne).

Quelques implémentations de SRM (e.g., celle du Jensen) sont limitées et n'autorisent que les chaînes d'options de courte longueur (e.g., au plus 8 caractères). Dans ce cas là, `aboot` peut être démarré avec le drapeau de démarrage "i". Avec ce drapeau, `aboot` demandera à l'utilisateur d'entrer une chaîne d'options pouvant atteindre 256 caractères. Par exemple :

```
boot dka0 -fl i
aboot> 3/vmlinux.gz root=/dev/sda3 single
```

Comme démarrer de cette façon devient rapidement pénible , `aboot` autorise l'utilisateur à définir des raccourcis pour les lignes de commande fréquemment utilisées. En particulier, une option donnée par un chiffre – option –> (0-9) demande à `aboot` d'utiliser l'option correspondante dans le fichier `/etc/aboot.conf`. Un exemple de fichier `aboot.conf` est donné ci-dessous :

```
#
# aboot default configurations
#
0:3/vmlinux.gz root=/dev/sda3
1:3/vmlinux.gz root=/dev/sda3 single
2:3/vmlinux.new.gz root=/dev/sda3
3:3/vmlinux root=/dev/sda3
8:- root=/dev/sda3            # fs-less boot of raw kernel
9:0/vmlinux.gz root=/dev/sda3 # fs-less boot of (compressed) ECOFF kernel
-
```

Avec ce fichier, la commande

```
boot dka0 -fl 1
```

correspond exactement à la commande de démarrage donnée ci-dessus. Il est cependant facile d'oublier la correspondance entre les numéros et les chaînes d'options. Pour éviter ce problème, démarrez avec l'option "h" et `aboot` affichera le contenu de `/etc/aboot.conf` avant d'afficher l'invite demandant la chaîne d'option entière.

En conclusion, même si aboot demande l'entrée d'une chaîne d'options, il est possible d'entrer un simple caractère ("i", "h", ou "0"-"9") pour obtenir le même résultat que si le drapeau avait été spécifié sur la ligne de commande de démarrage. Par exemple, vous pouvez démarrer avec le drapeau "i", taper ensuite "h" (suivi par entrée) pour vous rappeler le contenu de /etc/aboot.conf

Sélectionner la partition de /etc/aboot.conf Quand aboot est installé sur un disque dur, il a besoin de savoir sur quel partition il lui faut chercher le fichier /etc/aboot.conf. Nouvellement compilé, aboot cherchera sur la deuxième partition (/dev/sda2). Comme il serait contraignant d'avoir à recompiler aboot uniquement pour changer le numéro de la partition, abootconf autorise à directement modifier aboot déjà installé. Par exemple, si vous désiriez changer aboot afin qu'il utilise la *troisième* partition du disque /dev/sda, vous utiliseriez la commande :

```
abootconf /dev/sda 3
```

Vous pouvez vérifier le réglage courant simplement en omettant le numéro de partition. Alors, abootconf /dev/sda affichera la partition actuellement sélectionnée. Notez que aboot être déjà installé pour que cette commande réussisse. Aussi, lors de l'installation d'un nouvel aboot, le numéro de partition redeviendra celui par défaut (i.e., il sera nécessaire de relancer abootconf).

Depuis la version 0.5 de aboot, il est également possible de sélectionner la partition contenant le fichier aboot.conf depuis la ligne de commande de démarrage. Cela peut être fait avec une ligne de commande de la forme *a*:*b* où *a* est le numéro de la partition contenant /etc/aboot.conf et *b* est une option d'une lettre comme décrit plus haut (0-9, i, ou h). Par exemple, si vous tapez boot -fl "3:h" dka100 le système démarre depuis SCSI ID 1, charge /etc/aboot.conf depuis la troisième partition, affiche son contenu à l'écran et attend que vous entriez les options de démarrage.

3.7 Démarrage réseau

Deux étapes préliminaires sont nécessaires avant que Linux puisse démarrer par un réseau. Premièrement, vous devrez positionner les variables d'environnement de SRM pour permettre le démarrage *via* le protocole bootp et deuxièmement vous devrez configurer une autre machine comme serveur de démarrage. Reportez-vous à la documentation de SRM fournie avec votre machine pour toute information sur la mise en place de bootp. Configurer le serveur de démarrage dépend étroitement du système d'exploitation de cette machine, mais typiquement cela nécessite de lancer le programme bootpd en tâche de fond après avoir configuré le fichier /etc/bootptab. Le fichier bootptab possède une entrée par machine cliente autorisée à démarrer depuis le serveur. Par exemple, si vous voulez démarrer la machine myhost.cs.arizona.edu, une entrée de la forme suivante serait nécessaire :

```
myhost.cs.arizona.edu:\
        :hd=/remote/:bf=vmlinux.bootp:\
        :ht=ethernet:ha=08012B1C51F8:hn:vm=rfc1048:\
        :ip=192.12.69.254:bs=auto:
```

Cette entrée considère que l'adresse Ethernet de la machine est 08012B1C51F8 et que son adresse IP est 192.12.69.254. L'adresse Ethernet peut être trouvée grâce à la commande show device de la console SRM ou, si Linux est lancé, avec la commande ifconfig. L'entrée précise également que si le client ne déclare pas le contraire, le fichier qui sera lancé sera le fichier vmlinux.bootp du répertoire /remote. Pour plus d'informations sur la configuration de bootpd, reportez-vous à sa page de manuel.

Ensuite, construiser aboot grâce à la commande make netboot. Assurez-vous que le noyau que vous désirez lancer a déjà été construit. Par défaut, le Makefile du programme aboot utilise le noyau

`/usr/src/linux/arch/alpha/boot/vmlinux.gz` (éditez le `Makefile` si vous désirez utiliser un autre chemin d'accés). Le résultat de `make netboot` est un fichier nommé `vmlinux.bootp` contenant `aboot` *et* le noyau Linux, prêt pour le démarrage par réseau.

Enfin, copiez `vmlinux.bootp` dans le répertoire du serveur de démarrage. Dans l'exemple plus haut, vous l'auriez copié dans le répertoire `/remote/`. Ensuite, allumez la machine client et démarrez la, en spécifiant l'adaptateur Ethernet comme périphérique de démarrage. SRM nomme typiquement le premier adaptateur Ethernet `ewa0`, donc, pour démarrer depuis ce périphérique, vous utiliserez la commande :

```
boot ewa0
```

Les options `-fi` et `-fl` sont utilisable comme d'habitude. En particulier, vous pouvez demander à `aboot` d'attendre l'entrée d'arguments pour le noyau Linux en spécifiant l'option `-fl i`.

4 Partager un disque avec Digital Unix

Malheureusement, Digital Unix ne sait rien de Linux, aussi, partager un disque unique entre les deux systèmes n'est pas totalement simple. Cependant, ce n'est pas une tâche difficile si vous suivez les conseils prodigués dans cette section. Nous considérerons que vous utilisez la version 0.5 ou postérieure de `aboot`.

4.1 Partitionner le disque

Premièrement et avant tout : n'utilisez *jamais* les programmes de partitionnement de Linux (`minlabel` ou `fdisk`) sur un disque également utilisé par Digital Unix. Le programme Linux `minlabel` utilise le même format de table de partitions que le programmme `disklabel` de Digital Unix, mais il existe des incompatibilités avec les données écrites par `minlabel`, alors Digital Unix refusera tout simplement la table de partitions engendrée par `minlabel`. Pour configurer une partition Linux `ext2` sous Digital Unix, vous allez devoir changer l'entrée *disktab* de votre disque. Pour illustrer notre propos, supposons que vous avez un disque rz26 (un disque de 1Go) sur lequel vous voulez installer Linux. L'entrée *disktab* sous Digital Unix v3.2 ressemble à (voyez le fichier `/etc/disktab`) :

```
rz26|RZ26|DEC RZ26 Winchester:\
        :ty=winchester:dt=SCSI:ns#57:nt#14:nc#2570:\
        :oa#0:pa#131072:ba#8192:fa#1024:\
        :ob#131072:pb#262144:bb#8192:fb#1024:\
        :oc#0:pc#2050860:bc#8192:fc#1024:\
        :od#393216:pd#552548:bd#8192:fd#1024:\
        :oe#945764:pe#552548:be#8192:fe#1024:\
        :of#1498312:pf#552548:bf#8192:ff#1024:\
        :og#393216:pg#819200:bg#8192:fg#1024:\
        :oh#1212416:ph#838444:bh#8192:fh#1024:
```

Les champs intéressants ici sont oit/?/; et `p`*?*, où *?* désigne une lettre de l'intervalle `a-h` (les huit premières partitions). La valeur `o` indique l'adresse du début de la partition (en nombre de secteurs) et la valeur `p` donne la taille de la partition (également en nombre de secteurs). Reportez-vous à `disktab(4)` pour plus d'informations. Notez que Digital Unix *aime* définir des partitions qui se chevauchent. Pour les entrées ci-dessus, l'organisation des partitions ressemble à cela (vous pouvez vérifier en ajoutant les diverses valeurs `o` et `p`) :

```
   a    b        d          e          f
```

```
|---|-------|-----------|-----------|-----------|

                      c
|---------------------------------------------|

                   g             h
           |----------------|----------------|
```

Digital Unix insiste pour que la partition `a` commence à l'adresse 0 et que la partition `c` couvre l'étendue du disque. A part cela, vous pouvez organiser la table des partitions comme bon vous semble.

Supposons que vous avez Digital Unix utilisant la partition `g` et que vous voulez installer Linux sur la partition `h` avec la partition `b` comme partition de swap. Pour obtenir cette organisation sans détruire la partition Digital Unix existante, vous devez configurer explicitement les types des partitions. Vous pouvez réaliser ceci en ajoutant un champ `t` pour chaque partition. Dans notre cas, nous ajoutons la ligne suivante à l'entrée *disktab*.

```
            :ta=unused:tb=swap:tg=4.2BSD:th=reservd8:
```

Pourquoi avons-nous marqué la partition `h` comme "reservd8" plutôt que comme "ext2" ? Bon, Digital Unix ne connait rien de Linux. Une partition de type "ext2" correspond à une valeur numérique de 8, et Digital Unix utilise la châne "reservd8" pour cette valeur. Donc, dans le langage de Digital Unix, "reservd8" signifie "ext2". Ceci était la partie hardue. Maintenant, il ne nous reste plus qu'à installer la nouvelle entrée *disktab* sur le disque. Considérons que le disque à l'ID SCSI 5. Dans ce cas, nous faisons :

```
disklabel -rw /dev/rrz5c rz26
```

Vous pouvez vérifier que tout va bien en lisant le *disklabel* grâce à la commande `disklabel -r /dev/rrz5c`. A ce point, vous pouvez vouloir redémarrer Digital Unix et vous assurer que la partition Digital Unix est encore présente et en bon état. Si c'est le cas, vous pouvez arréter la machine et commencer l'installation de LInux. Prenez soin de sauter l'étape de partitionnement du disque lors de la procédure d'installation. Sachant que nous avons déjà installé une table de partitions correcte, vous devriez être capable de procéder à cette opération et de sélectionner la huitième partition comme partition racine de Linux et la deuxième comme partition de swap. Si le disque est le deuxième disque SCSI de la machine, les noms de périphériques pour ces deux partitions seront `/dev/sdb8` et `/dev/sdb2`, respectivement (notez que Linux utilise des lettre pour désigner les disques et des numéros pour désigner les partitions, exactement à l'inverse de Digital Unix ; le schéma de Linux a plus de sens bien sûr ;-).

4.2 Installer `aboot`

Premier obstacle : avec le *firmware* –> – SRM, vous ne pouvez démarrer qu'un et un seul système d'exploitation par disque. Pour cette raison, il est généralement préférable de disposer d'au moins deux disques SCSI dans une machine sur laquelle vous désirez utiliser aussi bien Linux que Digital Unix. Bien sûr vous pouvez aussi démarrer Linux depuis une disquette si la vitesse importe peu, ou par un réseau, si vous disposez d'un serveur `bootp`. Mais dans cette partie, nous considérerons que vous souhaitez démarrer Linux depuis un disque contenant une ou plusieurs partitions Digital Unix.

Deuxième obstacle : installer `aboot` sur un disque partagé avec Digital Unix rend les première et troisième partitions inutilisables (sachant qu'elles doivent commencer à l'adresse 0). Pour cette raison, nous vous recommandons de changer la taille de la partition `a` à une valeur juste suffisament élevée pour contenir `aboot` (1Mo devrait convenir).

Une fois que ces deux obstacles sont surmontés, installer `aboot` est aussi simple que d'habitude : comme les partitions `a` et `c` vont recouvrir `aboot` , nous devons spécifier à `swriteboot` que ceci est intentionnel . Nous pouvons le faire sous Linux avec une ligne de commande de la forme suivante (de nouveau, nous supposerons que l'on veut installer `aboot` sur le deuxième disque SCSI) :

```
swriteboot -f1 -f3 /dev/sdb bootlx
```

Le paramètre `-f1` signifie que nous voulons forcer l'écriture de `bootlx` même s'il recouvre la première partition. La même chose s'applique à la troisième partition.

C'est tout. Vous devriez désormais pouvoir arrêter le système et lancer Linux depuis le disque dur. Dans notre exemple, la ligne de commande SRM pour le faire serait :

```
boot dka5 -fi 8/vmlinux.gz -fl root=/dev/sdb8
```