Version : **2020.01**

Dernière mise-à-jour : 2020/01/30 03:28

SO209 - Gestion du Système de Fichiers UFS

Le système de fichiers d'Unix est organisé autour d'une arborescence unique ayant un point de départ appelé la **racine**, représenté par le caractère **/**. En dessous de cette racine se trouvent des répertoires contenant fichiers et sous-répertoires.

L'arborescence

- **/bin** : est une abréviation de ‘binary’ ou binaires. Sous Solaris /bin est un **lien symbolique** vers **/usr/bin**. Il contient des programmes tels ls.
- **/cdrom** : points de montage pour les CD et DVD.
- **/dev** : contient des **noms logiques** d’unités utilisés pour accéder à tout type de matériel. Ce sont des liens symboliques vers des **fichiers spéciaux** du répertoire **/devices**. Le répertoire /dev est organisé en sous-répertoires, par exemple :
  - **/dev/dsk** : partitions disque en mode **bloc**
  - **/dev/rdsk** : partitions disque en mode **raw bloc**
  - **/dev/printers** : imprimantes
  - **/dev/rmt** : lecteurs de bandes
- **/devices** : contient des **noms physiques** d'unités.
- **/etc** : contient des fichiers de configuration.
- **/export** : contient le répertoire /home. Le répertoire /home contient les répertoires de chaque utilisateur, sauf l’utilisateur root, qui doivent être exportés par le serveur **NFS**.
- **/home** : contient les répertoires des comptes locaux.
- **/kernel** : contient le noyau Solaris (**genunix**) et les fichiers associés.
- **/lib** : contient les bibliothèques communes utilisées par les programmes. Sous Solaris /lib est un lien symbolique vers **/usr/lib**.
- **/lost+found** : répertoire contenant des fichiers à récupérer par la commande **fsck**.
- **/mnt** : répertoire utilisé pour des montages occasionnels.
- **/opt** : répertoire utilisé pour l'installation des logiciels optionnels.
- **/platform** : contient des fichiers spécifiques à la plate-forme matérielle.
- **/proc** : un **pseudo-filesystem** représentant les processus en activité.

- **/sbin** : contient des binaires, donc programmes, pour l'administration du système local.
- **/system** : Point d'attache des pseudo-filesystems suivant :
  - **/system/contract** : système de fichiers au format **CTFS**,
  - **/system/object** : système de fichiers au format **OBJFS**.
- **/tmp** : stocke des fichiers temporaires créés par des programmes. Il est vidé à chaque redémarrage du système.
- **/usr** : contient de nombreux répertoires tels :
  - **/usr/bin** : contient des commandes des utilisateurs,
  - **/usr/sbin** : contient des commandes administratives,
  - **/usr/share/doc** : contient les HOWTO,
  - **/usr/share/man** : contient les manuels.
- **/var** : contient des fichiers de taille variable ainsi que des répertoires tels :
  - **/var/adm** : contient des fichiers journaux et comptabilité,
  - **/var/sadm** : répertoire de travail pour le gestionnaire de paquets logiciels.
- **/vol** : répertoire du service **VOLD** (**VO**lume **M**anagement **D**aemon) utilisé pour monter et démonter automatiquement les périphériques amovibles.

Il existe trois types majeurs de fichier sous le système Solaris:

- les fichiers normaux (ordinary files)
- les répertoires (directories)
- les fichiers spéciaux (special files ou Devices)

Le fichiers normaux sont des fichiers textes, des tableaux ou des exécutables.

La limite de la longueur du nom de fichier est de **255** caractères.

Il y a une distinction entres les majuscules et le minuscules.

Le caractère **/** est interdit dans les noms des fichiers.

L'espace est déconseillé dans les noms des fichiers.

Si le nom d'un fichier commence par un ., le fichier devient caché.

La commande mount

La commande **mount** est utilisée pour monter un filesystem. Monter un filesystem est nécessaire afin d'avoir accès à son contenu.

Les filesystems montés automatiquement par Solaris lors du démarrage sont décrits dans le fichier **/etc/vfstab** :

```
# cat /etc/vfstab
#device             device              mount               FS      fsck    mount   mount
#to mount           to fsck             point               type    pass    at boot options
#
fd                  -                   /dev/fd             fd      -       no      -
/proc               -                   /proc               proc    -       no      -
/dev/dsk/c0t0d0s1   -                   -                   swap    -       no      -
/dev/dsk/c0t0d0s0   /dev/rdsk/c0t0d0s0  /                   ufs     1       no      -
/dev/dsk/c0t0d0s7   /dev/rdsk/c0t0d0s7  /export/home        ufs     2       yes     -
/devices            -                   /devices            devfs   -       no      -
sharefs             -                   /etc/dfs/sharetab   sharefs -       no      -
ctfs                -                   /system/contract    ctfs    -       no      -
objfs               -                   /system/object      objfs   -       no      -
swap                -                   /tmp                tmpfs   -       yes     -
```

Ce fichier comporte 7 colonnes :

- **device to mount**
  - Nom de la partition en mode bloc pour un filesystem UFS ou une zone de swap
  - Nom de la ressource NFS
  - Nom d'un répertoire pour un filesystem virtuel
- **device to fsck**
  - Nom de la partition en mode bloc pour un filesystem UFS
  - Le signe **-** dans les autres cas car non significatif
- **mount point**
  - Nom du répertoire de montage
  - Le signe **-** dans les autres cas car non significatif

- **FS type**
  - Type de filesystem
- **fsck pass**
  - **-** indique que le filesystem ne sera pas vérifié par fsck
  - **0** indique qu'un filesystem UFS ne sera pas vérifié par fsck mais un autre type sera vérifié
  - **1** indique que le filesystem sera vérifié par fsck dans l'ordre de son apparition dans ce fichier
  - **2** indique que le filesystem sera vérifié par fsck en parallèle
- **mount at boot**
  - désigne si oui ou non le filesystem est monté au boot par la commande **mountall**
  - le filesystem racine prend toujours la valeur **no** car il est déjà monté au moment de l'interprétation de ce fichier
- **mount options**
  - la valeur **-** indique toutes les options par défaut
  - ce champs peut comporter une liste d'options séparés par des virgules **sans espaces**
  - dans le cas d'un filesystem UFS, l'option **logging** est activée par défaut

Afin de monter un filesystem manuellement il est nécessaire de connaître :

- le nom da la partition exprimé sous la forme du fichier spécial en mode **bloc**
- le répertoire de montage

Une commande de montage ressemble à celle-ci :

```
mount /dev/dsk/c0t0d1s0 /point_de_montage
```

où **point_de_montage** est un répertoire vide dans lequel est monté le filesystem /dev/dsk/c0d1s0.

Le répertoire de montage doit être dédié à cet usage. Tout fichier dans le répertoire sera caché par la présence du contenu de /dev/dsk/c0d1s0.

Options de la commande

Les options de la commande **mount** sont :

```
# mount --help
```

```
mount: illegal option -- help
Usage:
mount [-v | -p]
mount [-F FSType] [-V] [current_options] [-o specific_options]
        {special | mount_point}
mount [-F FSType] [-V] [current_options] [-o specific_options]
        special mount_point
mount -a [-F FSType ] [-V] [current_options] [-o specific_options]
        [mount_point ...]
```

La commande umount

L'action de démontage est effectuée par l'utilsation de la commande **umount**. Une commande umount ressemble à :

```
umount /point_de_montage
```

ou

```
umount /dev/dsk/c0t0d1s0
```

Si le filesystem est en cours d'utilisation la commande umount échoue.

Pour connaître les processus en cours, on dispose de la commande **fuser**.

Par exemple, dans notre cas, le filesystem **/dev/dsk/c0t0d0s7** est monté sur /export/home :

```
# cat /etc/vfstab | grep home
/dev/dsk/c0t0d0s7       /dev/rdsk/c0t0d0s7      /export/home    ufs     2       yes     -
```

Dans le cas où vous êtes connecté en tant qu'un utilisateur normal et vous agissez en tant que root grâce à la commande su, vous ne pourrez pas démonter /export/home. Par contre, si vous vous connectez directement en tant que root, aucun processus utilise ce montage.

```
# fuser -cu /export/home
/export/home:
```

Les options de cette commande sont :

- -c
  - effectue une recherche récursive
- -u
  - affiche les noms des propriétaires des processus trouvés

Dans ce cas, vous pouvez démonter /export/home :

```
# umount /export/home
```

Options de la commande

Les options de la commande **umount** sont :

```
# umount --help
umount: illegal option -- help
Usage:
umount [-f] [-V] [-o specific_options] {special | mount-point}
umount -a [-f] [-V] [-o specific_options] [mount_point ...]
```

Le fichier /etc/mntab

Les filesystems actuellement montés sont consignés dans le fichier **/etc/mntab**. Ce fichier peut être consulté mais pas modifié :

```
# cat /etc/mnttab
/dev/dsk/c0t0d0s0       /       ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=840000
1575092681
```

```
/devices        /devices        devfs   dev=4b80000     1575092677
ctfs    /system/contract        ctfs    dev=4c00001     1575092677
proc    /proc   proc    dev=4bc0000     1575092677
mnttab  /etc/mnttab     mntfs   dev=4c40001     1575092677
swap    /etc/svc/volatile       tmpfs   xattr,dev=4c80001       1575092677
objfs   /system/object  objfs   dev=4cc0001     1575092677
sharefs /etc/dfs/sharetab       sharefs dev=4d00001     1575092677
/usr/lib/libc/libc_hwcap1.so.1  /lib/libc.so.1  lofs    dev=840000      1575092680
fd      /dev/fd fd      rw,dev=4e80001  1575092681
swap    /tmp    tmpfs   xattr,dev=4c80002       1575092682
swap    /var/run        tmpfs   xattr,dev=4c80003       1575092682
/dev/dsk/c0t0d0s7       /export/home    ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=840007
1575092688
-hosts  /net    autofs  nosuid,indirect,ignore,nobrowse,dev=4f40001     1575092690
auto_home       /home   autofs  indirect,ignore,nobrowse,dev=4f40002    1575092690
solaris.i2tch.loc:vold(pid588)  /vol    nfs     ignore,noquota,dev=4f00001      1575092690
```

Pour remonter /export/home, il convient d'utiliser la commande mount :

```
# mount -F ufs /dev/dsk/c0t0d0s7 /export/home
```

Regardez maintenant de nouveau le fichier /etc/mnttab :

```
# cat /etc/mnttab
/dev/dsk/c0t0d0s0       /       ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=840000
1575092681
/devices        /devices        devfs   dev=4b80000     1575092677
ctfs    /system/contract        ctfs    dev=4c00001     1575092677
proc    /proc   proc    dev=4bc0000     1575092677
mnttab  /etc/mnttab     mntfs   dev=4c40001     1575092677
swap    /etc/svc/volatile       tmpfs   xattr,dev=4c80001       1575092677
objfs   /system/object  objfs   dev=4cc0001     1575092677
sharefs /etc/dfs/sharetab       sharefs dev=4d00001     1575092677
/usr/lib/libc/libc_hwcap1.so.1  /lib/libc.so.1  lofs    dev=840000      1575092680
```

```
fd      /dev/fd fd      rw,dev=4e80001  1575092681
swap    /tmp    tmpfs   xattr,dev=4c80002       1575092682
swap    /var/run        tmpfs   xattr,dev=4c80003       1575092682
-hosts  /net    autofs  nosuid,indirect,ignore,nobrowse,dev=4f40001     1575092690
auto_home       /home   autofs  indirect,ignore,nobrowse,dev=4f40002    1575092690
solaris.i2tch.loc:vold(pid588)  /vol    nfs     ignore,noquota,dev=4f00001      1575092690
/dev/dsk/c0t0d0s7       /export/home    ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=840007
1575094632
```

Options de montage pour un filesystem UFS

Le dernier champs du fichier **/etc/vfstab** contient les options de montage :

Option	Valeur par défaut	Description
rw/ro	rw	lecture/écriture ou lecture seule
largefiles/nolargefiles	largefiles	Création des fichiers de plus de 2Go
logging/nologging	logging	Journalisation ou non
atime/noatime	atime	Mise à jour ou non de la date de dernière consultation des fichiers non modifiés
exec/noexec	exec	Exécution ou non des programmes
devices/nodevices	devices	Accès ou non aux fichiers spéciaux
setuid/nosetuid	setuid	Prise en compte ou non des permissions SUID et SGID
suid/nosuid	suid	Combinaison de nodevices et nosetuid
quota	-	Activation des quotas
rq	-	Combinaison des options rw et quota

Ces options peuvent être aussi introduites sur la ligne de commande lors d'un montage manuel grâce à l'option **-o** de la commande mount.

Les options d'un filesystem monté peuvent être consultées grâce à la commande **mount** :

```
# mount | grep /export/home
/export/home on /dev/dsk/c0t0d0s7
```

```
read/write/setuid/devices/rstchown/intr/largefiles/logging/xattr/onerror=panic/dev=840007 on Sat Nov 30 07:17:12
2019
```

A l'aide du manuel et de l'internet, expliquez l'utilisation de **intr**, **xattr** et **onerror=panic**

Le Filesystem UFS

UFS (**U**nix **F**ile**S**ystem) est un filesystem de type **Berkeley** auquel ont été ajoutés des fonctions de **journalisation** appelées **UFS logging**. L'UFS logging crée un journal dans les blocs libres du filesystem. La taille est de 1Mo par Go de données avec un maximum de 64Mo.

Structure

Chaque système UFS contient des **groupe de cylindres**. Chaque group de cylindres contient un :

- boot block
- superbloc
- inode
- bloc d'indirection
- bloc de données

boot block

Ce bloc est utilisé quand le filesystem sert au démarrage. Il n'apparait donc que dans le premier groupe de cylindres et a une taille de 8Ko.

Superbloc

Le superbloc contient :

- la taille des blocs
- la taille du système de fichiers
- le nombre de montages effectués pour ce système de fichiers
- un pointeur vers la racine du système de fichiers
- les pointeurs vers la liste des inodes libres
- les pointeurs vers la liste des blocs de données libres

Le Superbloc est dupliqué sur le système de fichiers.

Inodes

En tapant la commande **ls -ld** vous obtenez une liste d'objets présents dans le répertoire courant.

Le premier caractère de chaque ligne peut être un des suivants :

- **-** - un fichier
- **d** - un répertoire
- **l** - un lien symbolique
- **b** - un périphérique du type bloc
- **c** - un périphérique du type caractère
- **p** - un tube nommé pour la communication entre processus
- **s** - un socket dans un contexte réseau

Par exemple :

```
# ls -l /dev/*dsk/c0t0d0s0
lrwxrwxrwx   1 root     root          47 Nov 29 13:41 /dev/dsk/c0t0d0s0 ->
../../devices/pci@0,0/pci8086,2829@d/disk@0,0:a
lrwxrwxrwx   1 root     root          51 Nov 29 13:41 /dev/rdsk/c0t0d0s0 ->
```

```
../../devices/pci@0,0/pci8086,2829@d/disk@0,0:a,raw

# ls -l /devices/pci@0,0/pci8086,2829@d/disk@0,0:a*
brw-r-----   1 root     sys       33,  0 Nov 30 06:44 /devices/pci@0,0/pci8086,2829@d/disk@0,0:a
crw-r-----   1 root     sys       33,  0 Nov 30 07:20 /devices/pci@0,0/pci8086,2829@d/disk@0,0:a,raw
```

Chaque fichier est représenté par un **inode**. L'inode, d'une taille de 128 octets contient :

- le type de fichier, soit **-**, **d**, **l**, **b**, **c**, **p**, **s**
- les droits d'accès, par exemple **rwx rw- r–**
- le nombre de liens physiques soit le nombre de noms
- l'UID du créateur ou l'UID affecté par la commande **chown** s'il y a eu une modification
- le GID du processus créateur ou le GID affecté par la commande **chgrp**
- la taille du fichier en octets
- la date de création, soit le **ctime**
- la date de dernière modification, soit le **mtime**
- la date du dernier accès, soit le **atime**
- les adresses qui pointent vers les blocs de données du fichier

Graphiquement, on peut schématiser cette organisation de la façon suivante :

Pour visualiser le numéro d'inode, utilisez l'option **-i** :

```
# ls -ldi /dev/*dsk/c0t0d0s0
    311303 lrwxrwxrwx   1 root     root          47 Nov 29 13:41 /dev/dsk/c0t0d0s0 ->
../../devices/pci@0,0/pci8086,2829@d/disk@0,0:a
    311348 lrwxrwxrwx   1 root     root          51 Nov 29 13:41 /dev/rdsk/c0t0d0s0 ->
../../devices/pci@0,0/pci8086,2829@d/disk@0,0:a,raw

# ls -ldi /devices/pci@0,0/pci8086,2829@d/disk@0,0:a*
  17301507 brw-r-----   1 root     sys       33,  0 Nov 30 06:44 /devices/pci@0,0/pci8086,2829@d/disk@0,0:a
  17301508 crw-r-----   1 root     sys       33,  0 Nov 30 07:20 /devices/pci@0,0/pci8086,2829@d/disk@0,0:a,raw
```

```
# ls -ldi /etc /etc/passwd
       824 drwxr-xr-x  86 root     sys         4608 Nov 30 06:44 /etc
      1301 -rw-r--r--   1 root     sys          710 Jan 11  2013 /etc/passwd
```

Blocs d'Indirection

L'inode contient les adresses des blocs de données du fichier. Par un système de blocs d'indirection, la taille maximale d'un fichier peut être de 1 téraoctet et la taille maximale du filesystem de 16 téraoctets.

Blocs de données

Les données sont stockées dans des blocs de données. Dans le cas d'un répertoire, le bloc de données contient une table qui référence les inodes et les noms des fichiers dans le répertoire.

Le nom d'un fichier est stocké dans le bloc de données et non pas dans l'inode. Cette particularité nous permet de donnéer deux noms différents au même fichier. Pour ajouter un nouveau nom à un fichier, il convient de créer un **lien physique**.

Liens Physiques

Un lien physique se crée en utilisant la commande suivante :

- ln nom_du_fichier nom_supplémentaire

Pour illustrer ce point, tapez la ligne de commande suivante :

```
# cd /tmp; mkdir inode; cd inode; touch fichier1; ls -ali
total 16
4038066373 drwxr-xr-x   2 root     root         182 Nov 30 07:23 .
4031513256 drwxrwxrwt   7 root     sys          524 Nov 30 07:23 ..
4041221744 -rw-r--r--   1 root     root           0 Nov 30 07:23 fichier1
```

Notez bien le numéro de l'inode du fichier **fichier1**. Notez aussi que le numéro dans le troisième champs de la ligne de fichier1 a la valeur **1** :

4041221744 -rw-r–r– **1** root root 0 Nov 30 07:23 fichier1

Créez maintenant un lien physique et visualisez le résultat :

```
# ln fichier1 fichier2; ls -lai
total 16
4038066373 drwxr-xr-x   2 root     root         247 Nov 30 07:25 .
4031513256 drwxrwxrwt   7 root     sys          524 Nov 30 07:23 ..
4041221744 -rw-r--r--   2 root     root           0 Nov 30 07:23 fichier1
4041221744 -rw-r--r--   2 root     root           0 Nov 30 07:23 fichier2
```

Notez les deux lignes suivantes :

4041221744 -rw-r–r– **2** root root 0 Nov 30 07:23 fichier1
4041221744 -rw-r–r– **2** root root 0 Nov 30 07:23 fichier2

Les deux fichiers, fichier1 et fichier2, sont référencés par le même inode. Le nombre de liens est donc augmenté de 1.

Liens Symboliques

Un lien symbolique est un **raccourci** vers un autre fichier ou répertoire. Un lien symbolique se crée en utilisant la commande suivante :

- ln -s nom_du_fichier nom_raccourci

Pour illustrer ce point, tapez les commandes suivantes :

```
# ln -s fichier1 fichier3; ls -lai
total 24
4038066373 drwxr-xr-x   2 root     root         312 Nov 30 07:27 .
4031513256 drwxrwxrwt   7 root     sys          524 Nov 30 07:23 ..
4041221744 -rw-r--r--   2 root     root           0 Nov 30 07:23 fichier1
```

```
4041221744 -rw-r--r--   2 root     root           0 Nov 30 07:23 fichier2
4041221688 lrwxrwxrwx   1 root     root           8 Nov 30 07:27 fichier3 -> fichier1
```

Notez que le lien symbolique est référencé par un autre inode. Le lien symbolique pointe vers le fichier1.

Création

La création d'un système de fichiers UFS se fait grâce à la commande **newfs**. Un exemple d'une telle commande est :

```
newfs -v -b 4096 -m 10 /dev/rdsk/c0t0d1s0
```

Dans cette commande on stipule :

- -v
  - le mode verbose
- -b 4096
  - une taille des blocs de 4096 octets
- -m 10
  - 10% d'espace disque réservé à root

La commande newfs est une commande de confort qui invoque la commande **mkfs**.

Par contre pour connaître les options d'une partition, il convient d'utiliser la commande mkfs avec l'option **-m** :

```
# mkfs -m /dev/dsk/c0t0d0s7
mkfs -F ufs -o
nsect=128,ntrack=48,bsize=8192,fragsize=1024,cgsize=16,free=1,rps=7,nbpi=8155,opt=t,apc=0,gap=0,nrpos=8,maxcontig
=128,mtb=n /dev/dsk/c0t0d0s7 15133230
```

Il est aussi possible lors de la création de spécifier trois autres options :

nbpi

L'option npbi (number of bytes per inode) correspond au nombre maximum de fichiers que peut contenir le filesystem. Les valeurs par défaut sont :

Taile du filesystem	Valeur de nbpi
< 1Go	2048
< 2Go	4096
< 3Go	6144
< 1To	8192
> 1To	1048576

free

Cette option stipule le pourcentage du disque réservé à root.

opt

Cette option désigne la politique de'optimisation :

- **time** - optimisation des performances (option par défaut),
- **space** - optimisation par espace disque.

Seules les options **opt** et **free** peuvent être modifiées après la création du filesystem.

Vérification

Un indicateur stocké dans le superblock permet de connaître l'état du filesystem. Cet indicateur peut prendre plusieurs valeurs :

Valeur	Description
FSACTIVE	Le filesystem est monté sans l'option logging et est actif

Valeur	Description
FSCLEAN	Le filesystem a été démonté correctement
FSSTABLE	Le filesystem est monté sans l'option logging mais est inactif
FSLOG	Le filesystem utilise l'option logging. Il n'est pas précisé si il est monté ou démonté
FSBAD	Le filesystem contient des données incohérentes

Au démarrage du système **fsck** n'est lancé que si l'indicateur est **FSBAD**

La commande fsck peut être lancé manuellement. Il est nécessaire de démonter le filesystem avant son utilisation et de vérifier celui-ci avec fsck et l'option **-m**. La commande fsck attend un argument qui est un nom de partition en mode raw bloc :

```
# umount /export/home
# fsck -m /dev/rdsk/c0t0d0s7
** /dev/rdsk/c0t0d0s7
ufs fsck: sanity check: /dev/rdsk/c0t0d0s7 okay
```

Saisissez maintenant la commande suivante :

```
# fsck /dev/rdsk/c0t0d0s7
** /dev/rdsk/c0t0d0s7
** Last Mounted on /export/home
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3a - Check Connectivity
** Phase 3b - Verify Shadows/ACLs
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cylinder Groups
2 files, 9 used, 7444611 free (19 frags, 930574 blocks, 0.0% fragmentation)
```

La dernière ligne indique :

Mot clé	Description
files	Le nombre d'inodes utilisés

Mot clé	Description
used	Le nombre de fragments utilisés
free	Le nombre de fragments inutilisés
frags	Le nombre de fragments inutilisés dans des blocs utilisés
blocs	Le nombre de blocs complets inutilisés

Si des erreurs sont trouvées, la commande devient interactive.

Il est possible donc de lancer la commande fsck avec une option **-y** ou **-n** pour contourner l'interactivité éventuelle.

Paramétrages

Nous avons vu que l'option free sur **/dev/dsk/c0t0d0s7** a une valeur de **1** :

```
# mkfs -m /dev/dsk/c0t0d0s7
mkfs -F ufs -o
nsect=128,ntrack=48,bsize=8192,fragsize=1024,cgsize=16,free=1,rps=7,nbpi=8155,opt=t,apc=0,gap=0,nrpos=8,maxcontig
=128,mtb=n /dev/dsk/c0t0d0s7 15133230
```

Afin de modifier cette valeur, nous disposons de la commande **tunefs**. L'utilisation de cette commande est limitée aux filesystems **démontés**.

Saisissez la commande suivante :

```
# tunefs -m 5 /dev/dsk/c0t0d0s7
minimum percentage of free space changes from 1% to 5%
```

Vérifiez son application :

```
# mkfs -m /dev/dsk/c0t0d0s7
mkfs -F ufs -o
nsect=128,ntrack=48,bsize=8192,fragsize=1024,cgsize=16,free=5,rps=7,nbpi=8155,opt=t,apc=0,gap=0,nrpos=8,maxcontig
=128,mtb=n /dev/dsk/c0t0d0s7 15133230
```

Concepts RAID

Les solutions RAID ou *Redundant Array of Independent Disks* ou encore *Redundant Array of Inexpensive Disks* permettent la combinaison de plusieurs disques de façon à ce que ceux-ci soient vu comme un seul disque logique.

Les solutions RAID sont issues du travail fourni par l'université de Berkeley en Californie sur un projet de tolérances de pannes. Les systèmes RAID offre maintenant plusieurs avantages :

- Addition des capacités,
- Amélioration des performances,
- Apporter la tolérance de panne.

Deux concepts sont fondamentaux à la compréhension des solutions RAID.

Disques en miroir

La technique des disques en miroir consiste à dupliquer l'écriture des données sur plusieurs disques. Le miroir peut être géré par un logiciel ou par du matériel.

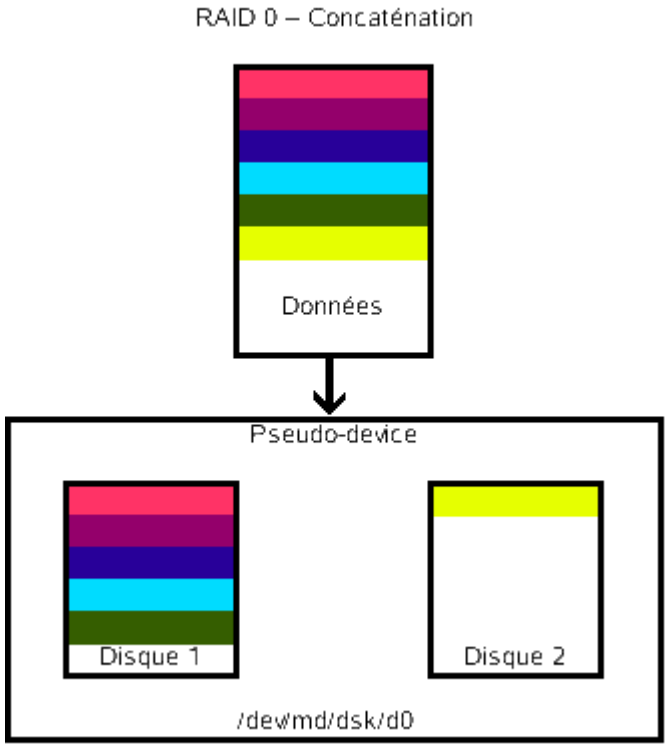
Bandes de données

La technique des bandes de données, autrement appelée *data striping* consiste à couper les données à enregistrer en segments séquentiels et contigus pour les enregistrer sur plusieurs disques physiques. L'ensemble des segments constitue alors un disque logique ou *striped disk*. Cette technique peut être améliorée en déposant une bande de parité, calculée à partir des données des autres bandes, afin de pouvoir reconstituer une bande de données défaillante.

Types de RAID

RAID 0 - Concaténation

Création de volume par récupération de l'espace libre sur un ou plusieurs disques. Le principe de la concaténation est la création d'un volume à bandes où chaque bande est une tranche. Il est similaire au LVM sous Linux et HP-UX.



;#; ;#;

Avantages

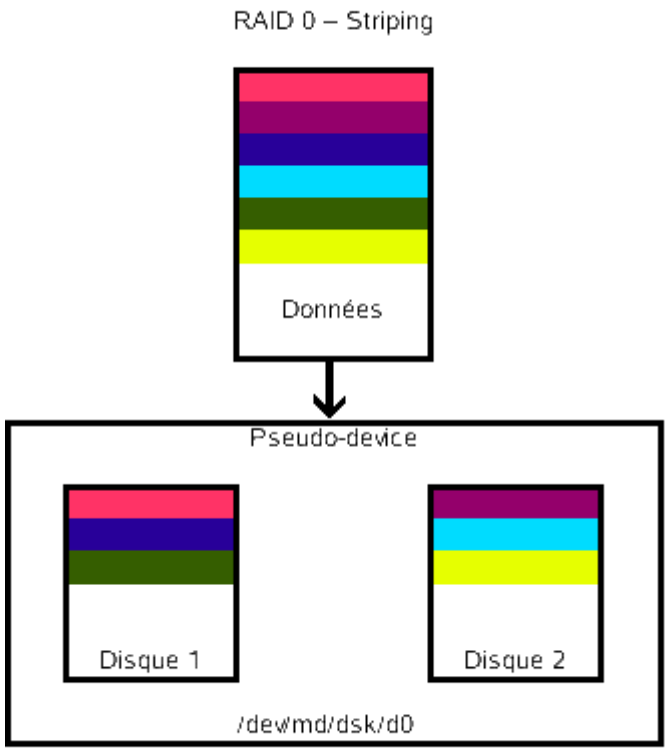
- Récupération de l'espace disque.

Inconvénients

- Pas de protection des données,
- Pas d'augmentation des performances d'E/S.

RAID 0 - Striping

Création de volume sur plusieurs disques afin d'augmenter les performances d'E/S. Le principe du striping est la création d'un volume à bandes réparties sur plusieurs tranche. La taille de la bande doit être fonction des données à écrire sur le volume (16k, 32k, 64k, etc.) Cette taille est choisie à la création du volume.



;#; ;#;

Avantages

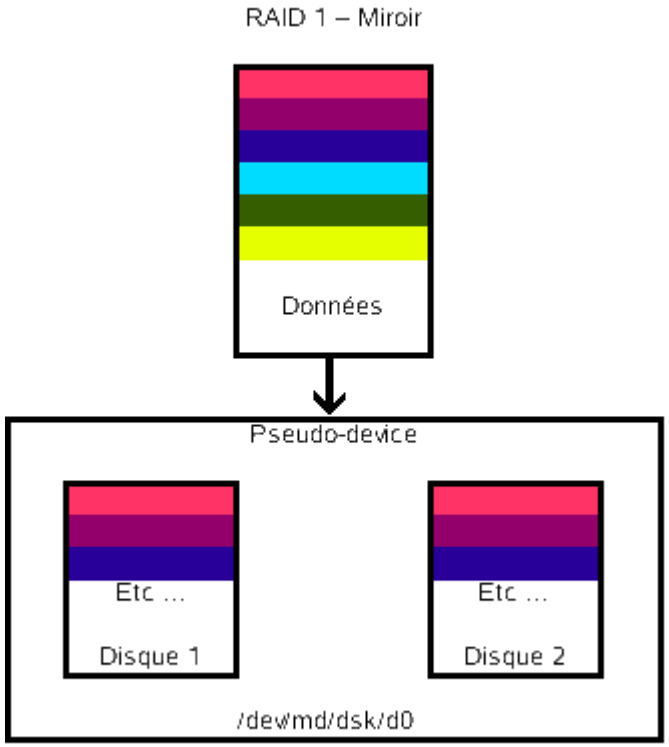
- Augmentation des performances d'E/S par écriture en parallèle sur les disques.

Inconvénients

- Pas de protection des données.

RAID 1 - Miroir

Création d'un volume où les disques sont en miroir. Quand les deux disques sont connectés à des contrôleurs de disques différents, on parle de *duplexing* :



;#; ;#;

Avantages

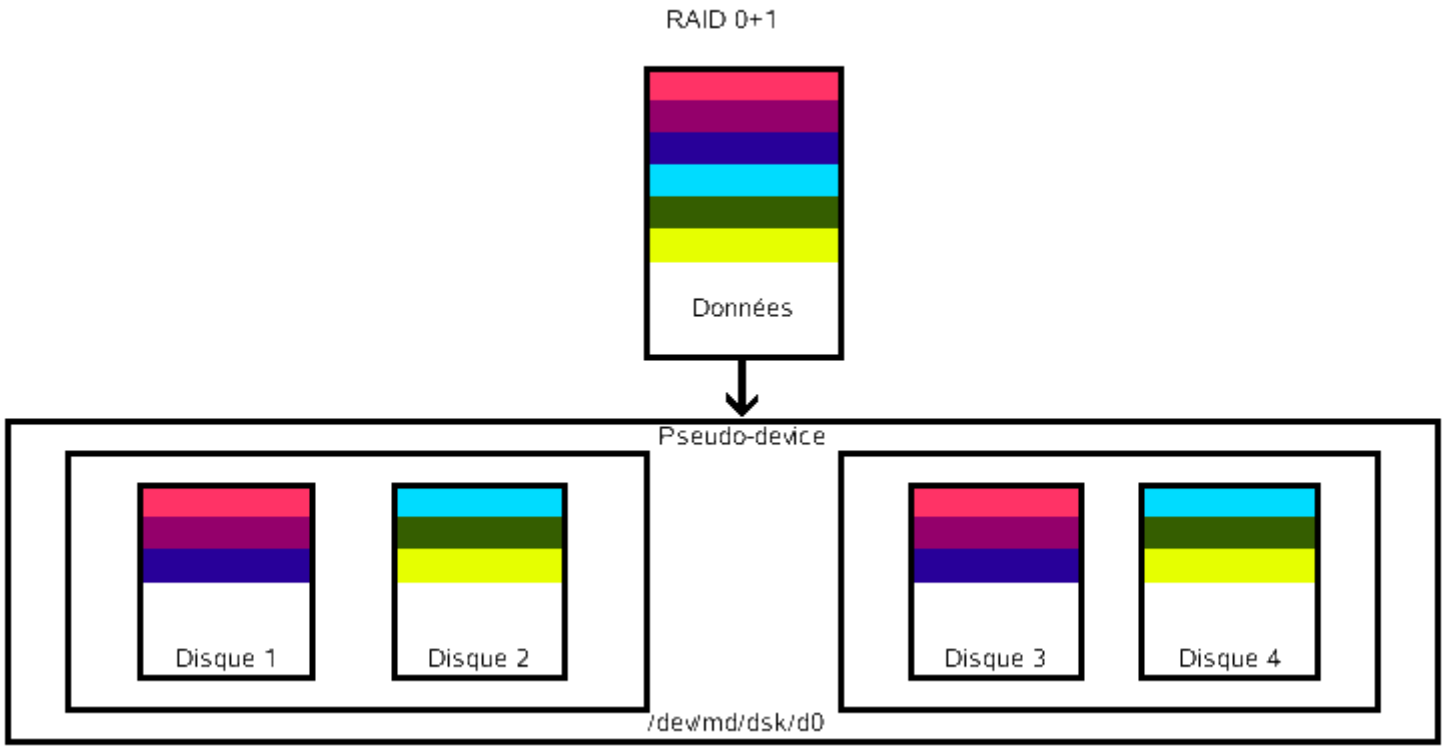
- Protection des données contre une défaillance d'un disque.

Inconvénients

- Coûteux à cause de l'augmentation du nombre de disques.

RAID 1+0 - Striping en Miroir

Le RAID 1+0 ou encore 0+1 est une technique qui réunit le RAID 0 et le RAID 1. On l'appelle aussi un RAID **exotique**:



;#; ;#;

Avantages

- Protection des données contre une défaillance d'un disque.
- Augmentation des performances d'E/S par écriture en parallèle sur les disques.

Inconvénients

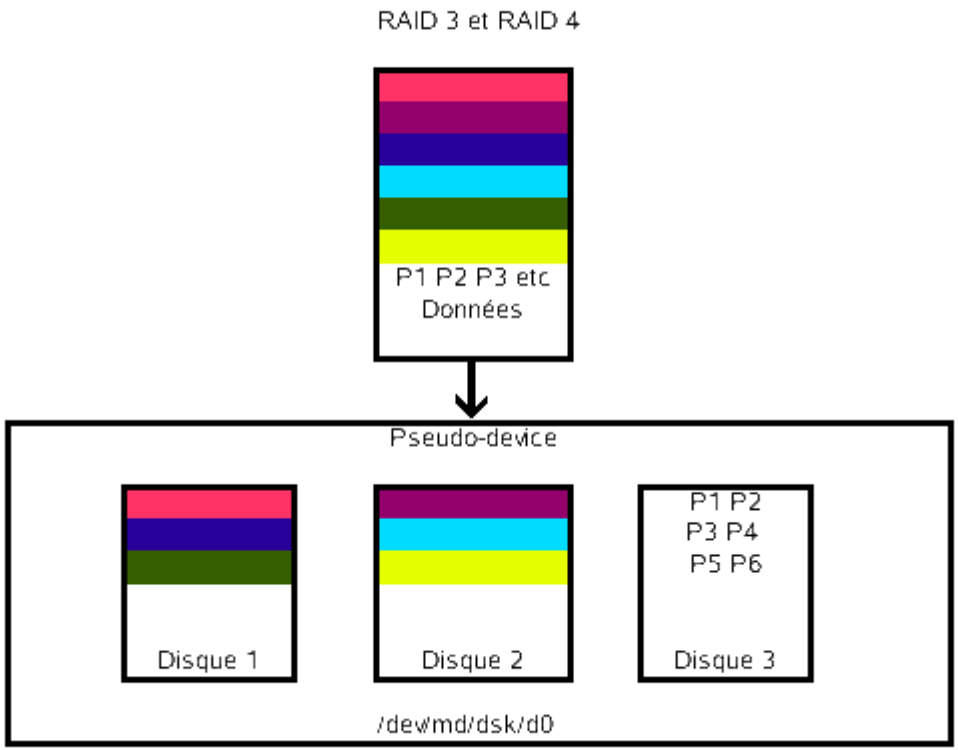
- Coûteux à cause de l'augmentation du nombre de disques.

RAID 2 - Miroir avec Contrôle d'Erreurs

Le RAID 2 est une technique de miroir avec contrôle de correction d'erreurs (EEC). De nos jours cette technique est peu utilisée, ayant été remplacée par les RAID 3, 4 et 5.

RAID 3 et 4 - Striping avec Parité

Les RAID 3 et 4 sont des technologies avec bandes de parité distribuées sur un seul disque :



;#; ;#;

En RAID 3, la taille des segments n’est pas modifiable et est fixée à 512 octets (en RAID 3 : un segment = un secteur de disque dur = 512 octets).

En RAID 4, la taille des segments est variable et se modifie en temps réel. Cela implique que les informations de parité doivent être mise à jour à chaque écriture afin de vérifier si la taille des segments a été modifiée.

Avantages

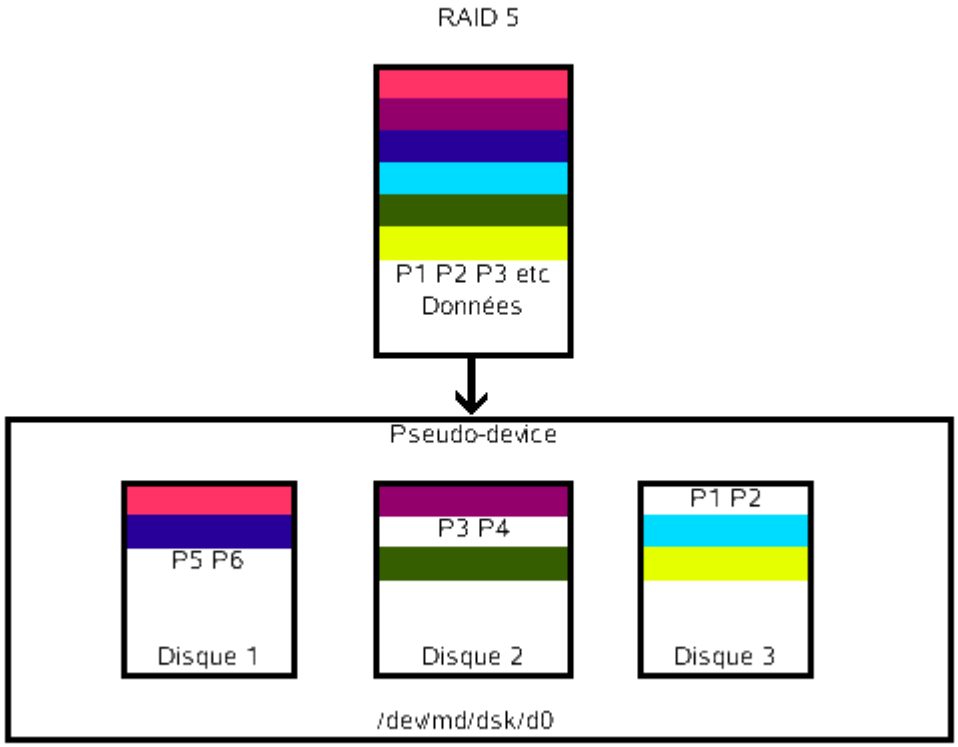
- Protection des données contre une défaillance d'un disque.

Inconvénients

- Création d'un goulot d'étranglement des données à cause de l'écriture des données de parité sur un seul disque.

RAID 5 - Striping avec Parité Distribuée

Le RAID 5 est une technologie avec bandes de parité distribuées sur plusieurs disques :



;#; ;#;

Avantages

- Protection des données contre une défaillance d'un disque,
- Evite le goulot d'étranglement d'un seul disque de parité.

Inconvénients

- Lecture moins performante qu'avec RAID 3 et 4.

Au délà de RAID 5

Il existe aussi deux autres technologies RAID, toute deux issues de la technologie RAID 5 :

- RAID 6
  - *Disk Striping with Double Distributed Parity*
- RAID TP
  - *Disk Striping with Triple Distributed Parity*

Solaris Volume Manager

Introduction

Le Solaris Volume Manager est un service évolué de la gestion de l'espace disque. Il a été introduit avec Solaris 9.

Le Solaris Volume Manager implémente les niveaux :

- RAID 0 - Concaténation
- RAID 0 - Striping
- RAID 1 - Mirroring
- RAID 5 - Striping avec parité distribuée

Sur un système SPARC, il existe 8 tranches tandis que sur un système x64/x86 il en existe 16. Sur un système x64/x86, la 8ième tranche est utilisée pour contenir le code pour le boot tandis que le 9ième est utilisée pour des secteurs alternatives sur certains systèmes. La commande format ne permet pas de manipuler plus de 8 tranches. Pour les systèmes x64/x86, il existe un logiciel sous licence GPL appelé **dskpart** qui permet de manipuler les 16 tranches. Le README se trouve à cette adresse.

Terminologie

Volumes

L'entité de base du Solaris Volume Manager est un **volume**. Un volume est composé d'un ensemble de tranches et apparait comme un disque logique via un **pseudo-device** autrement appelé un **meta-device**.

Noms des Volumes

Les noms des volumes sont générés dans les répertoires **/dev/md/dsk** et **/dev/md/rdsk**. Chaque volume possède un nom unique allant de **d0** jusqu'à **d127**. Il n'y a pas de standardisation au niveau des noms. Par conséquent un RAID 5 peut avoir le nom d5, d0 ou d127 par exemple.

State Database

Les informations concernant la configuration courante du Solaris Volume Manager sont stockées dans la ***State Database***. Il est nécessaire de copier cette base de données en plusieurs exemplaires, appelés des ***replicas*** ou **répliques**. Chaque réplique a une taille de 4 Mo par défaut (8192 blocks).

La création des copies de la *State Database* est la première opération à effectuer sur un serveur. Il est conseillé que les répliques résident dans des tranches dédiées à cet effet. Par convention on utilise la partition S7 des disques pour stocker les répliques.

La *State Database* joue un rôle très important sous Solaris Volume Manager. Quand intervient une désynchronisation entre des sous-miroirs, le système doit savoir où se trouvent les données valides.

Dans ce cas il va avoir une **élection à majorité absolue** entre les copies de la *State Database*. Pour cette raison il est nécessaire de créer au moins trois copie de cette dernière.

Qui plus est au moins la moitié des répliques doit être accessible lorsque le système est en cours de fonctionnement. Dans le cas contraire le système crash.

Au moins la moitié des répliques + 1 doit être accessible pour qu'un système puisse démarrer.

Dans le cas d'un système à 2 disques, si un disque tombe en panne, la règle des 50% +1 ne pourra jamais être respectée et le système crash en boucle.

Il est cependant possible de démarrer un système sans obtenir la majorité absolue. Pour cela on positionne le paramètre md:mirrored_root_flag=1 dans le fichier /etc/system :

```
echo "set md:mirrored_root_flag=1" >>/etc/system
```

Les répliques sont utilisées également comme zone de synchronisation des volume-miroirs (RAID 1). Un nombre insuffisant de répliques provoquerait une baisse significative des performances d'entrée/sortie pour les volumes en miroir. En général, on préconise la création d'au moins 2 répliques par volume RAID 1 jusqu'à une limite de 50 répliques par disque.

Partitions logicielles

Les **Partitions logicielles** ou *soft partitions* en anglais permettent de diviser des devices en plusieurs parties. Cette division permet de créer une multitude de systèmes de fichiers sur un device. Il est possible de créer ces partitions logicielles, soit à partir d'un volume, soit directement à partir d'une partition.

Réserve de Tranches Dynamiques

Une **Réserve de Tranches Dynamiques** ou *hot spare pool* en anglais est un ensemble de tranches disponibles en remplacement en cas d'une

défaillance d'un disque composant un miroir ou un RAID-5. Lorsqu'une défaillance d'un disque survient, SVM va utiliser une partition de la Réserve de Tranches Dynamiques afin de remplacer ce disque. Il est nécessaire bien entendu que les tranches stockées dans la Réserve de Tranches Dynamiques soient d'une taille égale ou supérieure à la tranche à remplacer. Il est impératif d'ajouter les tranches par ordre de taille de la plus petite à la plus grande. En effet lorsque SVM recherche une partition de remplacement, celui-ci lit la liste des tranches dans l'ordre dans lequel celles-ci ont été ajoutées dans la réserve.

Fichiers Importants

- **/etc/lvm/mddb.cf**
  - contient l'emplacement de la metadb et de ses répliques,
- **/kernel/drv/md.conf**
  - reflète les informations du fichier mddb.cf. Il est nécessaire au fonctionnement du driver md.

Gestion des meta devices

LAB#1 - Préparation du disque

Préparez votre disque de la façon suivante :

- tranche 3 de 20Mo minimum
- tranche 4 d'une taille deux fois celle du tranche 5
- tranche 5 et tranche 6 d'une taille égale.

Vous devez obtenir une table des partitions **similaire** à celle-ci :

```
partition> print
Volume:  mydisk
Current partition table (unnamed):
Total disk cylinders available: 2085 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders        Size            Blocks
```

```
  0       root    wm      68 - 1141        8.23GB    (1074/0/0) 17253810
  1       swap    wu       1 -   67      525.56MB    (67/0/0)    1076355
  2     backup    wm       0 - 2084       15.97GB    (2085/0/0) 33495525
  3 unassigned    wm    1563 - 1565       23.53MB    (3/0/0)       48195
  4 unassigned    wm    1566 - 1827        2.01GB    (262/0/0)   4209030
  5 unassigned    wm    1828 - 1955     1004.06MB    (128/0/0)   2056320
  6 unassigned    wm    1956 - 2083     1004.06MB    (128/0/0)   2056320
  7       home    wm    1142 - 1562        3.23GB    (421/0/0)   6763365
  8       boot    wu       0 -    0        7.84MB    (1/0/0)       16065
  9 unassigned    wm       0                0         (0/0/0)           0

partition>
```

Voici quelques indications pour vous faciliter la tâche ci-dessus :

- Démontez **/export/home**,
- Fixez la valeur de la variable **TERM** avec la commande **TERM=vt100; export TERM**,
- Commentez la ligne **/export/home** dans le fichier **/etc/vfstab**,
- Lancez la commande **format** et choisissez le menu **partitions**,
- Diminuez la taille de la tranche 7 de façon à récupérer 4 Go d'espace disque,
- Créez la tranche 3 de 20mo,
- Créez la tranche 4 de 2Go,
- Créez les tranches 5 et 6 d'une taille égale, la somme étant équivalente à l'espace disque restant,
- Nommez la nouvelle table de partitions “My Table” en utilisant les menus **partition > name**,
- Montez au menu supérieur en utilisant la commande **quit**,
- Nommez le volume “My Disk” avec le menu **volname**,
- Sauvez votre table dans l'emplacement par défaut en utilisant la commande **save**,
- Quittez la commande format avec **quit**,
- Créez un nouveau FileSystem sur S7 avec la commande **newfs /dev/dsk/c0t0d0s7**,
- Éditez le fichier **/etc/vfstab** en supprimant le caractère **#** devant la ligne

/export/home,
- Montez /export/home avec la commande **mount /export/home**.

Vérifiez ensuite qu'il n'existe pas de State Database :

```
# metadb -i
metadb: solaris.i2tch.loc: there are no existing databases

#
```

Créez maintenant la *State Database* ainsi que 2 répliques dans votre tranche 3 :

```
# metadb -f -a -c 3 c0t0d0s3
mdmonitord daemon pid 1116 already running
```

Les options de la commande sont :

- **-a** : Permet d'ajouter une réplique.
- **-f** : Permet de forcer un état de la metadb. Cette option est à utiliser lors de la création de la première réplique ou lors de la suppression de la dernière.
- **-c < n >** : Permet de spécifier le nombre de répliques à ajouter sur une partition. Cette option permet de placer plusieurs répliques sur une seule partition.

Constatez le contenu du fichier **/etc/lvm/mddb.cf** :

```
# cat /etc/lvm/mddb.cf
#metadevice database location file do not hand edit
#driver minor_t daddr_t device id       checksum
sd      3       16      id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d  -4246
sd      3       8208    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d  -12438
sd      3       16400   id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d  -20630
```

ainsi que le fichier **/kernel/drv/md.conf** :

```
# cat /kernel/drv/md.conf
#
#pragma ident   "@(#)md.conf    2.2     04/04/02 SMI"
#
# Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# The parameters nmd and md_nsets are obsolete.  The values for these
# parameters no longer have any meaning.
name="md" parent="pseudo" nmd=128 md_nsets=4;
# Begin MDD database info (do not edit)
mddb_bootlist1="sd:3:16:id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d
sd:3:8208:id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d
sd:3:16400:id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430/d";
# End MDD database info (do not edit)
```

Constatez maintenant la création de la metadb :

```
# metadb -i
        flags           first blk       block count
     a        u         16              8192            /dev/dsk/c0t0d0s3
     a        u         8208            8192            /dev/dsk/c0t0d0s3
     a        u         16400           8192            /dev/dsk/c0t0d0s3
 r - replica does not have device relocation information
 o - replica active prior to last mddb configuration change
 u - replica is up to date
 l - locator for this replica was read successfully
 c - replica's location was in /etc/lvm/mddb.cf
 p - replica's location was patched in kernel
 m - replica is master, this is replica selected as input
 W - replica has device write errors
 a - replica is active, commits are occurring to this replica
 M - replica had problem with master blocks
 D - replica had problem with data blocks
```

```
 F - replica had format problems
 S - replica is too small to hold current data base
 R - replica had device read errors
```

Créez ensuite un filesystem UFS sur le tranche 5 :

```
# newfs -v -b 4096 -m 10 /dev/rdsk/c0t0d0s5
newfs: construct a new file system /dev/rdsk/c0t0d0s5: (y/n)? y
pfexec mkfs -F ufs /dev/rdsk/c0t0d0s5 2056320 63 -1 4096 1024 16 10 7 2048 t 0 -1 8 256 n
Warning: 1920 sector(s) in last cylinder unallocated
/dev/rdsk/c0t0d0s5:     2056320 sectors in 335 cylinders of 48 tracks, 128 sectors
        1004.1MB in 56 cyl groups (6 c/g, 18.00MB/g, 8544 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 37024, 74016, 111008, 148000, 184992, 221984, 258976, 295968, 332960,
 1701664, 1738656, 1775648, 1812640, 1849632, 1886624, 1923616, 1960608,
 1997600, 2034592
```

Montez votre tranche 5 à la racine de votre système de fichiers et créez un fichier appelé **SVM test** conteant la chaîne *svm_test* :

```
# mkdir /slice5
# mount /dev/dsk/c0t0d0s5 /slice5
# cd /slice5
# ls
lost+found
# echo "SVM test" > svm_test
```

LAB#2 - RAID 0 - Concaténation

Le Solaris Volume Manager peut être utilisé pour concaténer des tranches afin de créer un volume contigu. Afin d'y parvenir, il convient d'utiliser la commande **metainit** en passant en argument à la commande au moins une tranche **démontée**. Dans notre cas, vous allez concaténer les tranches 5 et 6 de votre disque. Saisissez donc les commandes suivantes :

```
# cd ..
# umount /slice5
# metainit d0 2 1 c0t0d0s5 1 c0t0d0s6
d0: Concat/Stripe is setup
```

Dans ce cas, vous avez créer un volume **d0** de deux tranches qui se suivent physiquement. Le chiffre **2** indique le nombre de **bandes** tandis que les chiffres **1** indiquent le nombre de tranches par bande. Dans le cas d'un volume concaténé il devrait y avoir autant de bandes que de tranches. Constatez ensuite la création du volume en utilisant la commande **metastat** :

```
# metastat -p
d0 2 1 c0t0d0s5 \
         1 c0t0d0s6

# metastat d0
d0: Concat/Stripe
    Size: 4112640 blocks (2.0 GB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
        c0t0d0s5          0     No      Yes
    Stripe 1:
        Device     Start Block  Dbase   Reloc
        c0t0d0s6          0     No      Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Montez ensuite le volume **d0** et constatez l'espace disque libre :

```
# mount /dev/md/dsk/d0 /slice5
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0      8.1G   3.9G   4.1G    49%    /
/devices                 0K     0K     0K     0%    /devices
```

```
ctfs                     0K     0K     0K     0%    /system/contract
proc                     0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
swap                   1.3G   988K   1.3G     1%    /etc/svc/volatile
objfs                    0K     0K     0K     0%    /system/object
sharefs                  0K     0K     0K     0%    /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                       8.1G   3.9G   4.1G    49%    /lib/libc.so.1
fd                       0K     0K     0K     0%    /dev/fd
swap                   1.3G    72K   1.3G     1%    /tmp
swap                   1.3G    32K   1.3G     1%    /var/run
/dev/dsk/c0t0d0s7      3.2G   3.2M   3.1G     1%    /export/home
/dev/md/dsk/d0         945M   1.0M   849M     1%    /slice5
```

Vous noterez que le volume monté ne reflète pas l'addition des deux tranches en termes de taille. En effet, pour l'instant le *file system* précédemment présent sur **c0t0d0s5** n'occupe pas tout le volume d0. Afin d'étendre le *file system* à la taille maximale du volume, il convient d'utiliser la commande **growfs** :

```
# growfs -M /slice5 /dev/md/rdsk/d0
Warning: 3840 sector(s) in last cylinder unallocated
/dev/md/rdsk/d0:        4112640 sectors in 670 cylinders of 48 tracks, 128 sectors
        2008.1MB in 112 cyl groups (6 c/g, 18.00MB/g, 8544 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 37024, 74016, 111008, 148000, 184992, 221984, 258976, 295968, 332960,
 3765024, 3802016, 3839008, 3876000, 3912992, 3949984, 3986976, 4023968,
 4060960, 4097952
```

Constatez ensuite la réussite de la commande :

```
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0      8.1G   3.9G   4.1G    49%    /
/devices                 0K     0K     0K     0%    /devices
ctfs                     0K     0K     0K     0%    /system/contract
```

```
proc                       0K     0K     0K     0%    /proc
mnttab                     0K     0K     0K     0%    /etc/mnttab
swap                     1.3G   988K   1.3G     1%    /etc/svc/volatile
objfs                      0K     0K     0K     0%    /system/object
sharefs                    0K     0K     0K     0%    /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                         8.1G   3.9G   4.1G    49%    /lib/libc.so.1
fd                         0K     0K     0K     0%    /dev/fd
swap                     1.3G    72K   1.3G     1%    /tmp
swap                     1.3G    32K   1.3G     1%    /var/run
/dev/dsk/c0t0d0s7        3.2G   3.2M   3.1G     1%    /export/home
/dev/md/dsk/d0           1.8G   2.0M   1.8G     1%    /slice5
```

Naviguez ensuite au point de montage de votre volume et constatez la présence du fichier précédemment créé :

```
# cd /slice5
# ls
lost+found  svm_test
# cat svm_test
SVM test
```

Créez ensuite un deuxième fichier de contrôle :

```
# echo "svm_test_2" > svm_test_2
# ls
lost+found  svm_test    svm_test_2
```

Constatez l'état du volume avec la commande **metastat** :

```
# metastat
d0: Concat/Stripe
    Size: 4112640 blocks (2.0 GB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
```

```
        c0t0d0s5          0     No      Yes
    Stripe 1:
        Device     Start Block  Dbase   Reloc
        c0t0d0s6          0     No      Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Afin de rajouter une tranche à un volume existant, il convient d'utiliser la commande **metattach**. Ajoutez donc la tranche c0t0d0s4 au volume d0 :

```
# metattach d0 c0t0d0s4
d0: component is attached
```

Augmentez ensuite la taille du *filesystem* :

```
# growfs -M /slice5 /dev/md/rdsk/d0
Warning: 3450 sector(s) in last cylinder unallocated
/dev/md/rdsk/d0:        8321670 sectors in 1355 cylinders of 48 tracks, 128 sectors
        4063.3MB in 226 cyl groups (6 c/g, 18.00MB/g, 8544 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 37024, 74016, 111008, 148000, 184992, 221984, 258976, 295968, 332960,
 7965728, 8002720, 8039712, 8076704, 8113696, 8150688, 8187680, 8224672,
 8261664, 8298656
```

Constatez la taille du volume :

```
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0      8.1G   3.9G   4.1G    49%    /
/devices                 0K     0K     0K     0%    /devices
ctfs                     0K     0K     0K     0%    /system/contract
proc                     0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
```

```
swap                   1.3G   988K   1.3G     1%    /etc/svc/volatile
objfs                    0K     0K     0K     0%    /system/object
sharefs                  0K     0K     0K     0%    /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                       8.1G   3.9G   4.1G    49%    /lib/libc.so.1
fd                       0K     0K     0K     0%    /dev/fd
swap                   1.3G    72K   1.3G     1%    /tmp
swap                   1.3G    32K   1.3G     1%    /var/run
/dev/dsk/c0t0d0s7      3.2G   3.2M   3.1G     1%    /export/home
/dev/md/dsk/d0         3.7G   4.0M   3.6G     1%    /slice5
```

Constatez ensuite la présence de vos fichiers :

```
# pwd
/slice5
# ls
lost+found  svm_test    svm_test_2
# cat svm_test_2
svm_test_2
```

Démontez ensuite votre volume et supprimez-le grâce à la commande **metaclear** :

```
# cd ..
# umount /slice5
# metaclear d0
d0: Concat/Stripe is cleared
```

Montez ensuite la tranche c0t0d0s5 sur slice5 et constatez la présence de vos fichiers :

```
# mount /dev/dsk/c0t0d0s5 /slice5
# cd /slice5
# ls
lost+found  svm_test    svm_test_2
```

Notez que les fichiers et leurs contenus sont intacts.

Dans l'exemple précédent, vous avez créé un volume de tranches qui se suivent physiquement. Cette fois-ci, il convient de récupérer deux tranches *orphelines* sur votre disque. Créez donc un volume contenant les tranches 4 et 6 de votre disque :

```
# cd ..
# umount /slice5
# metainit d0 2 1 c0t0d0s4 1 c0t0d0s6
d0: Concat/Stripe is setup
# newfs -v -b 4096 -m 10 /dev/md/rdsk/d0
newfs: construct a new file system /dev/md/rdsk/d0: (y/n)? y
pfexec mkfs -F ufs /dev/md/rdsk/d0 6265350 63 240 4096 1024 64 10 7 6144 t 0 -1 8 14 n
Warning: inode blocks/cyl group (1222) >= data blocks (708) in last
    cylinder group. This implies 5670 sector(s) cannot be allocated.
/dev/md/rdsk/d0:        6259680 sectors in 414 cylinders of 240 tracks, 63 sectors
        3056.5MB in 138 cyl groups (3 c/g, 22.15MB/g, 3552 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 45456, 90880, 136304, 181728, 227152, 272576, 318000, 363424, 408848,
 5814304, 5859728, 5905152, 5950576, 5996000, 6041424, 6086848, 6132272,
 6177696, 6223120
```

Montez ce volume :

```
# mount /dev/md/dsk/d0 /slice5
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0      8.1G   3.9G   4.1G    49%    /
/devices                 0K     0K     0K     0%    /devices
ctfs                     0K     0K     0K     0%    /system/contract
proc                     0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
```

```
swap                   1.3G   988K   1.3G     1%    /etc/svc/volatile
objfs                    0K     0K     0K     0%    /system/object
sharefs                  0K     0K     0K     0%    /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                       8.1G   3.9G   4.1G    49%    /lib/libc.so.1
fd                       0K     0K     0K     0%    /dev/fd
swap                   1.3G    72K   1.3G     1%    /tmp
swap                   1.3G    32K   1.3G     1%    /var/run
/dev/dsk/c0t0d0s7      3.2G   3.2M   3.1G     1%    /export/home
/dev/md/dsk/d0         2.9G   3.0M   2.6G     1%    /slice5
```

Vérifiez le contenu de votre volume :

```
# cd /slice5
# ls
lost+found
```

Vos fichiers de contrôle ne sont pas présents car ils sont stockés sur le file system sur c0t0d0s5 qui ne fait pas partie de ce d0.

Dernièrement, démontez votre volume et détruisez-le avec la commande **metaclear** :

```
# cd ..
# umount /slice5
# metaclear d0
d0: Concat/Stripe is cleared
```

LAB#3 - RAID 0 - Striping et Partitions Logicielles

Créez maintenant un *volume striping* avec une taille d'inode de 8k :

```
# metainit d0 1 2 c0t0d0s5 c0t0d0s6 -i 8k
d0: Concat/Stripe is setup

# metastat d0
d0: Concat/Stripe
    Size: 4112640 blocks (2.0 GB)
    Stripe 0: (interlace: 16 blocks)
        Device     Start Block  Dbase   Reloc
        c0t0d0s5          0     No      Yes
        c0t0d0s6          0     No      Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Le chiffre **1** représente le nombre de bandes. Le chiffre **2** représente le nombre de tranches sur lesquelles la bande sera répartie.

Avec le Solaris Volume manager, il est aussi possible de créer des partitions logicielles *à l'intérieur d'un volume*. Afin de créer une partition logicielle, il convient d'utiliser la commande **metainit** en passant en arguments :

- le nom de la partition logicielle
- l'option -p
- le nom de volume dans lequel sera insérée la partition logicielle
- la taille de la partition logicielle.

Créez donc la partition logicielle **d5** et constatez sa présence :

```
# metainit d5 -p d0 512m
d5: Soft Partition is setup
# metastat -p d5
d5 -p d0 -o 16 -b 1048576
d0 1 2 c0t0d0s5 c0t0d0s6 -i 16b
```

Créez ensuite un *filesystem* sur d5 :

```
# newfs /dev/md/rdsk/d5
newfs: /dev/md/rdsk/d5 last mounted as /slice5
newfs: construct a new file system /dev/md/rdsk/d5: (y/n)? y
Warning: 9824 sector(s) in last cylinder unallocated
/dev/md/rdsk/d5:        1048576 sectors in 70 cylinders of 240 tracks, 63 sectors
        512.0MB in 14 cyl groups (5 c/g, 36.91MB/g, 17536 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 75696, 151360, 227024, 302688, 378352, 454016, 529680, 605344, 681008,
 756672, 832336, 908000, 983664
```

Montez d5 et constatez sa taille avec la commande **df** :

```
# mount /dev/md/dsk/d5 /slice5
# df -h | grep d5
/dev/md/dsk/d5         482M   1.0M   433M     1%    /slice5
```

Constatez ensuite l'état des volumes avec la commande **metastat** :

```
# metastat
d5: Soft Partition
    Device: d0
    State: Okay
    Size: 1048576 blocks (512 MB)
        Extent              Start Block              Block count
             0                       16                  1048576

d0: Concat/Stripe
    Size: 4112640 blocks (2.0 GB)
    Stripe 0: (interlace: 16 blocks)
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes
        c0t0d0s6          0     No            Okay   Yes

Device Relocation Information:
```

```
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Le Solaris Volume Manager peut également être utilisé pour créer une partition logicielle directement sur une tranche. Créez donc la partition logicielle **d1** sur la tranche 4 de votre disque et constatez le bon déroulement de la commande :

```
# metainit d1 -p c0t0d0s4 1g
d1: Soft Partition is setup
# metastat -p
d1 -p c0t0d0s4 -o 1 -b 2097152
d5 -p d0 -o 16 -b 1048576
d0 1 2 c0t0d0s5 c0t0d0s6 -i 16b
```

Créez ensuite un *file system* sur d1 :

```
# newfs /dev/md/rdsk/d1
newfs: construct a new file system /dev/md/rdsk/d1: (y/n)? y
Warning: 4528 sector(s) in last cylinder unallocated
/dev/md/rdsk/d1:        2097152 sectors in 139 cylinders of 240 tracks, 63 sectors
        1024.0MB in 28 cyl groups (5 c/g, 36.91MB/g, 17536 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 75696, 151360, 227024, 302688, 378352, 454016, 529680, 605344, 681008,
 1361984, 1437648, 1513312, 1588976, 1664640, 1740304, 1815968, 1891632,
 1967296, 2042960
```

Montez la partition logicielle en **/slice6** et constatez sa taille avec la commande **df** :

```
# mkdir /slice6
# mount /dev/md/dsk/d1 /slice6
# df -h /slice6
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d1         964M   1.0M   905M     1%    /slice6
```

Pour augmenter la taille de la partition logicielle, il convient d'utiliser la commande **metattach**. Augmentez donc la taille de d1 de 512m et constatez

le bon déroulement de la commande :

```
# metattach d1 512m
d1: Soft Partition has been grown
# metastat -p
d1 -p c0t0d0s4 -o 1 -b 3145728
d5 -p d0 -o 16 -b 1048576
d0 1 2 c0t0d0s5 c0t0d0s6 -i 16b
```

Comparez les deux lignes d1 avant et après la commande :

- d1 -p c0t0d0s4 -o 1 -b 2097152
- d1 -p c0t0d0s4 -o 1 -b 3145728

Notez que la taille a été augmentée de 1048576.

Constatez ensuite la taille de d1 :

```
# df -h /slice6
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d1         964M   1.0M   905M     1%    /slice6
```

Comme dans le cas précédent, le résultat ne reflète pas la taille réelle du volume. Utilisez donc la commande **growfs** pour aligner la taille du *file system*/ sur celle du volume :

```
# growfs -M /slice6 /dev/md/rdsk/d1
Warning: 14352 sector(s) in last cylinder unallocated
/dev/md/rdsk/d1:        3145728 sectors in 209 cylinders of 240 tracks, 63 sectors
        1536.0MB in 42 cyl groups (5 c/g, 36.91MB/g, 17536 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 75696, 151360, 227024, 302688, 378352, 454016, 529680, 605344, 681008,
```

```
 2421280, 2496944, 2572608, 2648272, 2723936, 2799600, 2875264, 2950928,
 3026592, 3102256
```

Constatez maintenant la taille de d1 :

```
# df -h /slice6
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d1         1.4G   1.5M   1.4G     1%    /slice6
```

Démontez d1 et supprimez-le :

```
# umount /slice6
# metaclear -r d1
d1: Soft Partition is cleared
```

Constatez ensuite le bon déroulement de la commande :

```
# metastat
d5: Soft Partition
    Device: d0
    State: Okay
    Size: 1048576 blocks (512 MB)
        Extent              Start Block              Block count
             0                       16                  1048576

d0: Concat/Stripe
    Size: 4112640 blocks (2.0 GB)
    Stripe 0: (interlace: 16 blocks)
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes
        c0t0d0s6          0     No            Okay   Yes

Device Relocation Information:
Device   Reloc  Device ID
```

```
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Démontez d5 et supprimez **d0** :

```
# umount /slice5
# metaclear -r d0
metaclear: solaris.i2tch.loc: d0: metadevice in use
```

Comme vous pouvez constater, il est impossible de supprimer d0 puisque celui-ci contient d5. Supprimez donc d'abord d5 et d0 en même temps :

```
# metaclear -r d5
d5: Soft Partition is cleared
d0: Concat/Stripe is cleared
```

L'option **-r** de la commande **metaclear** implique une suppression récursive.

LAB#4 - RAID 1 - Miroir

Dans le cas de la mise en place d'un RAID 1, il est **primordial** que les deux disques soient identiques :

- la même capacité,
- la même géométrie.

Dans le cas de la mise en place d'un miroir entre deux disques entiers, la première tâche à effectuer est de copier la structure et le contenu du disque 0 vers le disque 1. Pour accomplir cette tache, utilisez une des deux commandes suivantes :

```
# prtvtoc -h /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c1d0s2
```

```
# dd if=/dev/rdsk/c0t0d0s2 of=/dev/rdsk/c1d0s2 count=16
```

Dans le cas de ce LAB, vous allez *simuler* la mise en place d'un RAID 1 en utilisant deux **tranches identiques**. Pour accomplir cette tâche, il convient de créer d'abord les **sous-miroirs** :

```
# metainit d2 1 1 c0t0d0s5
d2: Concat/Stripe is setup
# metainit d3 1 1 c0t0d0s6
d3: Concat/Stripe is setup
```

Avec SVM on peut utiliser jusqu'à 4 sous-miroirs par volume (4 copies des données).

Ensuite créez le miroir **d4** en y attachant les sous-miroirs **d2** et **d3** :

```
# metainit d4 -m d2
d4: Mirror is setup
# metattach d4 d3
d4: submirror d3 is attached
```

Constatez le résultat de vos commandes :

```
# metastat
d4: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d3
      State: Resyncing
    Resync in progress: 67 % done
    Pass: 1
```

```
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2056320 blocks (1004 MB)

d2: Submirror of d4
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes


d3: Submirror of d4
    State: Resyncing
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s6          0     No            Okay   Yes


Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Vous devez noter dans le résultat de la commande la présence de plusieurs lignes importantes :

- **State: Resyncing**
  - En effet, lors de la mise en place du miroir, les disques se synchronisent. Aucune action n'est alors permise sur le volume tant que la synchronisation ne soit pas effectuée. La ligne **Etat** peut prendre plusieurs valeurs : *Okay*, *Resyncing* , *Resync canceled* ou *Needs Maintenance*,
- **Pass: 1**
  - La valeur d'**Pass** ou *Accès* en français est un chiffre entre **0** et **9**. Ce chiffre conditionne l'ordre dans lequel les miroirs seront synchronisés lors du re-démarrage du système. La valeur par défaut est **1**. Dans le cas où deux miroirs possèdent la même valeur pour l'**Accès**, les deux miroirs sont synchronisés en même temps. Dans le cas contraire les miroirs sont synchronisés dans l'ordre croissant de 1 à 9. Une valeur

de **0** est utilisée pour un miroir en lecture seule.

- **Read option: roundrobin (default)**
  - L'option **roundrobin** concerne l'ordonnancement de la répartition de charge. Dans ce cas les lectures de données sont successivement faites sur chaque sous-miroir, l'un après l'autre.
- **Write option: parallel (default)**
  - L'option **parallel** implique que les données soient répliquées et envoyées d'une manière simultanée aux miroirs du volume.

> Consultez la page Internet suivante pour vous renseigner sur les options de lecture et d'écriture pour des volumes RAID 1 Wikipedia.

A l'issu de la période de synchronisation, l'état devient **OK** :

```
# metastat
d4: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d3
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2056320 blocks (1004 MB)

d2: Submirror of d4
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes
```

```
d3: Submirror of d4
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s6          0     No            Okay   Yes


Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Les options de lecture (**r**) et d'écriture (**w**) peuvent être modifiées grâce à l'utilisation de la commande **metaparam** :

```
# metaparam -r geometric d4
# metaparam d4
d4: Mirror current parameters are:
    Pass: 1
    Read option: geometric (-g)
    Write option: parallel (default)
# metaparam -w serial d4
# metaparam d4
d4: Mirror current parameters are:
    Pass: 1
    Read option: geometric (-g)
    Write option: serial (-S)
```

La même commande peut être utilisée pour modifier la valeur de Pass ou *Accès* :

```
# metaparam -p 5 d4
# metaparam d4
d4: Mirror current parameters are:
    Pass: 5
    Read option: geometric (-g)
```

```
    Write option: serial (-S)
```

Afin de poursuivre, il convient de re-définir l'Accès à **1** :

```
# metaparam -p 1 d4
# metaparam d4
d4: Mirror current parameters are:
    Pass: 1
    Read option: geometric (-g)
    Write option: serial (-S)
```

Créez maintenant un *file system* sur le volume **d4** :

```
# newfs -v -b 4096 -m 10 /dev/md/rdsk/d4
newfs: /dev/md/rdsk/d4 last mounted as /slice5
newfs: construct a new file system /dev/md/rdsk/d4: (y/n)? y
pfexec mkfs -F ufs /dev/md/rdsk/d4 2056320 63 240 4096 1024 16 10 7 2048 t 0 -1 8 14 n
/dev/md/rdsk/d4:        2056320 sectors in 136 cylinders of 240 tracks, 63 sectors
        1004.1MB in 68 cyl groups (2 c/g, 14.77MB/g, 6880 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 30336, 60640, 90944, 121248, 151552, 181856, 212160, 242464, 272768,
 1757664, 1787968, 1818272, 1848576, 1878880, 1909184, 1939488, 1969792,
 2000096, 2030400
```

Montez ensuite votre RAID 1 sur **/slice5** et créez ensuite le fichier **raidtest** sur /slice5 :

```
# mount /dev/md/dsk/d4 /slice5
# echo "Ceci est un test d'un miroir" > /slice5/raidtest
```

Vous allez maintenant renommer le volume RAID 1 en **d40**. Pour accomplir cette tâche, il convient d'abord de démonter le volume et d'utiliser la commande **metarename** :

```
# umount /slice5
# metarename d4 d40
```

```
d4: has been renamed to d40
```

Constatez maintenant vos volumes :

```
# metastat
d40: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d3
      State: Okay
    Pass: 1
    Read option: geometric (-g)
    Write option: serial (-S)
    Size: 2056320 blocks (1004 MB)

d2: Submirror of d40
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes


d3: Submirror of d40
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s6          0     No            Okay   Yes


Device Relocation Information:
Device   Reloc  Device ID
```

```
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Dans le cas d'une défaillance d'un disque du miroir la valeur de **Etat** devient *Needs Maintenance*. Il est donc nécessaire de remplacer celui-ci. Dans ce cas, il convient d'utiliser la commande **metadetach** :

```
# metadetach -f d40 d3
d40: submirror d3 is detached
```

Il est ensuite nécessaire de supprimer le volume du disque défaillant en utilisant la commande **metaclear** :

```
# metaclear -f d3
d3: Concat/Stripe is cleared
```

Après avoir réparé/remplacé le disque défaillant, il est nécessaire de reconstruire le sous-miroir :

```
# metainit d30 1 1 c0t0d0s6
d30: Concat/Stripe is setup
```

Dernièrement, il convient de rattacher le nouveau volume au miroir :

```
# metattach d40 d30
d40: submirror d30 is attached
```

et ensuite de constater l'état des volumes :

```
# metastat
d40: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d30
      State: Resyncing
    Resync in progress: 36 % done
    Pass: 1
```

```
    Read option: geometric (-g)
    Write option: serial (-S)
    Size: 2056320 blocks (1004 MB)

d2: Submirror of d40
    State: Okay
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes


d30: Submirror of d40
    State: Resyncing
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s6          0     No            Okay   Yes


Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Notez que le volume d30 est en cours de synchronisation. Ce processus de synchronisation peut être interrompue grâce à la commande **metasync -c d40**. Dans ce cas la valeur de **Etat** devient *Resync canceled*. La synchronisation peut ensuite est reprise grâce à la commande **metasync d40**.

Il convient maintenant de constater que les données soient présentes :

```
# mount /dev/md/dsk/d40 /slice5
# cd /slice5
# ls
lost+found  raidtest
```

Le volume RAID 1 peut être supprimé à l'aide de la commande **metaclear** :

```
# cd ..
# umount /slice5
# metaclear -r d40
d40: Mirror is cleared
d2: Concat/Stripe is cleared
d30: Concat/Stripe is cleared
```

LAB#5 - Réserve de Tranches Dynamiques

Pour utiliser une Réserve de Tranches Dynamiques il convient d'abord de la créer grâce à la commande **metainit** :

```
# metainit hsp000
hsp000: Hotspare pool is setup
```

A ce stade, votre Réserve de Tranches Dynamiques est vide. Pour visualiser l'état de la Réserve de Tranches Dynamiques, il convient d'utiliser la commande **metahs -i** :

```
# metahs -i
hsp000: is empty

Device Relocation Information:
Device   Reloc  Device ID
```

Ajoutez maintenant la tranche 4 de votre disque à la Réserve de Tranches Dynamiques grâce à la commande **metahs -a** :

```
# metahs -a hsp000 c0t0d0s4
hsp000: Hotspare is added
```

Constatez maintenant l'état de votre Réserve de Tranches Dynamiques :

```
# metahs -i
hsp000: 1 hot spare
        Device     Status      Length           Reloc
        c0t0d0s4   Available    4209030 blocks  Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Bien que votre Réserve de Tranches Dynamiques soit créée, elle est à ce stade inutilisable parce qu'elle n'est attachée à aucun volume. Pour attacher une Réserve de Tranches Dynamiques à un volume, il convient d'utiliser la commande **metaparam**.

A l'aide du manuel de la commande, déterminez les options adéquates pour attacher une Réserve de Tranches Dynamiques à un volume RAID 5.

Pour supprimer une Réserve de Tranches Dynamiques, il convient d'utiliser la commande **metaclear** :

```
# metaclear hsp000
hsp000: Hotspare pool is cleared
```

Il est aussi possible d'attacher une Réserve de Tranches Dynamiques à un volume lors de la création de ce dernier.

Créez donc la Réserve de Tranches Dynamiques **hsp001** :

```
# metainit hsp001
hsp001: Hotspare pool is setup
```

Ajoutez maintenant la tranche 4 de votre disque à la Réserve de Tranches Dynamiques grâce à la commande **metahs -a** :

```
# metahs -a hsp001 c0t0d0s4
hsp001: Hotspare is added
```

Mettez en place un miroir entre les tranches 5 et 6 en indiquant que la Réserve de Tranches Dynamiques hsp001 sera attachée aux volumes :

```
# metainit d10 1 1 c0t0d0s5 -h hsp001
d10: Concat/Stripe is setup

# metainit d11 1 1 c0t0d0s6 -h hsp001
d11: Concat/Stripe is setup

# metainit d20 -m d10 d11
metainit: d20: WARNING: This form of metainit is not recommended.
The submirrors may not have the same data.
Please see ERRORS in metainit(1M) for additional information.
d20: Mirror is setup
```

Notez bien le message d'avertissement lors de la première création du miroir. Bien que possible, cette syntaxe est déconseillée. Il convient donc d'utiliser les **deux** commandes **metainit** et **metattach**.

```
# metaclear d20
d20: Mirror is cleared
# metainit d20 -m d10
```

```
d20: Mirror is setup
# metattach d20 d11
d20: submirror d11 is attached
```

Constatez maintenant le résultat de votre travail :

```
# metahs -i
hsp001: 1 hot spare
        Device     Status      Length           Reloc
        c0t0d0s4   Available    4209030 blocks  Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

```
# metastat
d20: Mirror
    Submirror 0: d10
      State: Okay
    Submirror 1: d11
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2056320 blocks (1004 MB)

d10: Submirror of d20
    State: Okay
    Hot spare pool: hsp001
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s5          0     No            Okay   Yes
```

```
d11: Submirror of d20
    State: Okay
    Hot spare pool: hsp001
    Size: 2056320 blocks (1004 MB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s6          0     No            Okay   Yes


hsp001: 1 hot spare
        Device     Status      Length           Reloc
        c0t0d0s4   Available    4209030 blocks  Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Essayez maintenant de supprimer la Réserve de Tranches Dynamiques hsp001 :

```
# metaclear hsp001
metaclear: solaris.i2tch.loc: hsp001: hotspare pool in use
```

Notez qu'il n'est pas possible de supprimer hsp001 tant que cette dernière soit en cours d'utilisation.

Pour détacher hsp001 des sous-miroirs d10 et d11, il convient d'utiliser la commande **metaparam** avec l'option **-h none** :

```
# metaparam -h none d10
# metaparam -h none d11
```

Il est possible de supprimer des tranches de la Réserve de Tranches Dynamiques grâce à la commande **metahs -d** :

```
# metahs -d hsp001 c0t0d0s4
hsp001: Hotspare is deleted
```

Dernièrement supprimez la Réserve de Tranches Dynamiques hsp001 ainsi que votre miroir :

```
# metaclear hsp001
hsp001: Hotspare pool is cleared

# metaclear -r d20
d20: Mirror is cleared
d10: Concat/Stripe is cleared
d11: Concat/Stripe is cleared
```

LAB#6 - RAID 5 - Striping avec Parité Distribuée

Pour créer un volume RAID 5, il est nécessaire de disposer d'au moins trois disques.

Dans le cas de ce LAB, vous allez *simuler* la mise en place d'un RAID 5 en utilisant trois tranches. Pour accomplir cette tâche, il convient d'utiliser la commande **metainit** :

```
# metainit d5 -r c0t0d0s4 c0t0d0s5 c0t0d0s6
d5: RAID is setup
```

Il convient ensuite de constater l'état du volume :

```
# metastat -p
d5 -r c0t0d0s4 c0t0d0s5 c0t0d0s6 -k -i 32b
# metastat
d5: RAID
    State: Initializing
    Initialization in progress: 41.6% done
    Interlace: 32 blocks
```

```
    Size: 4096575 blocks (2.0 GB)
Original device:
    Size: 4111936 blocks (2.0 GB)
        Device     Start Block  Dbase        State Reloc  Hot Spare
        c0t0d0s4        330        No Initializing   Yes
        c0t0d0s5        330        No Initializing   Yes
        c0t0d0s6        330        No Initializing   Yes

Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Notez que l'état du volume RAID 5 est **Initialisation**. Il existe trois états pour RAID 5, à savoir en anglais *Initializing*, *Okay* et *Maintenance*. Notez aussi qu'il existe un état pour chaque tranche dans le volume. Il existe 5 états pour les tranches, à savoir en anglais, *Initializing*, *Okay*, *Resyncing*, *Maintenance* ou *Maintenance Last Erred*.

A l'issu de la période d'initialisation, l'état devient **Ok** :

```
# metastat
d5: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 4096575 blocks (2.0 GB)
Original device:
    Size: 4111936 blocks (2.0 GB)
        Device     Start Block  Dbase        State Reloc  Hot Spare
        c0t0d0s4        330        No         Okay   Yes
        c0t0d0s5        330        No         Okay   Yes
        c0t0d0s6        330        No         Okay   Yes
```

```
Device Relocation Information:
Device   Reloc  Device ID
c0t0d0   Yes    id1,sd@SATA_____VBOX_HARDDISK____VBc95d9143-c1389430
```

Créez maintenant un *file system* sur le volume RAID 5 :

```
# newfs -v -b 4096 -m 10 /dev/md/rdsk/d5
newfs: construct a new file system /dev/md/rdsk/d5: (y/n)? y
pfexec mkfs -F ufs /dev/md/rdsk/d5 4096575 63 240 4096 1024 32 10 7 4096 t 0 -1 8 14 n
Warning: 946 sector(s) in last cylinder unallocated
/dev/md/rdsk/d5:        4096574 sectors in 271 cylinders of 240 tracks, 63 sectors
        2000.3MB in 91 cyl groups (3 c/g, 22.15MB/g, 5344 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 45456, 90880, 136304, 181728, 227152, 272576, 318000, 363424, 408848,
 3679376, 3724800, 3770224, 3815648, 3861072, 3906496, 3951920, 3997344,
 4042768, 4088192
```

Montez **d5** sur **/slice5** et constatez sa taille :

```
# mount /dev/md/dsk/d5 /slice5
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0      8.1G   3.9G   4.1G    49%    /
/devices                 0K     0K     0K     0%    /devices
ctfs                     0K     0K     0K     0%    /system/contract
proc                     0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
swap                   1.3G   988K   1.3G     1%    /etc/svc/volatile
objfs                    0K     0K     0K     0%    /system/object
sharefs                  0K     0K     0K     0%    /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                       8.1G   3.9G   4.1G    49%    /lib/libc.so.1
fd                       0K     0K     0K     0%    /dev/fd
swap                   1.3G    72K   1.3G     1%    /tmp
```

```
swap                    1.3G    32K   1.3G     1%    /var/run
/dev/dsk/c0t0d0s7       3.2G   3.2M   3.1G     1%    /export/home
/dev/md/dsk/d5          1.9G   2.0M   1.7G     1%    /slice5
```

Quotas

Les quotas sont appliqués sur des *filesystems*. L'administrateur met souvent des quotas en place sur l'arborescence de **/export/home** pour limiter l'espace de stockage occupé par les utilisateurs.

LAB#7 - Quotas sur UFS

Dans le cas de cet exemple, vous allez utiliser le RAID 5 logiciel créé précédemment pour stocker les fichiers de /export/home.

Copiez tous le contenu de /export/home vers /slice5 :

```
# cp -pR /export/home/* /slice5
```

Démontez maintenant /slice5 :

```
# umount /slice5
```

Editez le fichier **/etc/vfstab** en mettant en commentaire la ligne commençant par **/dev/dsk/c0t0d0s7** et en ajoutant la ligne commençant par **/dev/md/dsk/d5** :

```
#/dev/dsk/c0t0d0s7        /dev/rdsk/c0t0d0s7        /export/home    ufs     2       yes     -
/dev/md/dsk/d5 /dev/md/rdsk/d5        /export/home    ufs     2       yes     rq
```

Notez l'utilisation de l'option **rq** qui indique au système la mise en place de quotas pour ce volume.

Démontez ensuite /export/home :

```
# umount /dev/dsk/c0t0d0s7
```

Remontez /export/home et contrôlez le résultat :

```
# mount /export/home
# cat /etc/mnttab
/dev/dsk/c0t0d0s0       /       ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=840000
1575106761
/devices        /devices        devfs   dev=4b80000     1575106757
ctfs    /system/contract        ctfs    dev=4c00001     1575106757
proc    /proc   proc    dev=4bc0000     1575106757
mnttab  /etc/mnttab     mntfs   dev=4c40001     1575106757
swap    /etc/svc/volatile       tmpfs   xattr,dev=4c80001       1575106757
objfs   /system/object  objfs   dev=4cc0001     1575106757
sharefs /etc/dfs/sharetab       sharefs dev=4d00001     1575106757
/usr/lib/libc/libc_hwcap1.so.1  /lib/libc.so.1  lofs    dev=840000      1575106760
fd      /dev/fd fd      rw,dev=4e80001  1575106761
swap    /tmp    tmpfs   xattr,dev=4c80002       1575106762
swap    /var/run        tmpfs   xattr,dev=4c80003       1575106762
-hosts  /net    autofs  nosuid,indirect,ignore,nobrowse,dev=4f40001     1575106769
auto_home       /home   autofs  indirect,ignore,nobrowse,dev=4f40002    1575106769
solaris.i2tch.loc:vold(pid589)  /vol    nfs     ignore,noquota,dev=4f00001      1575106770
/dev/md/dsk/d5  /export/home    ufs     rw,intr,largefiles,logging,xattr,onerror=panic,dev=1540005
1575118335
```

Pour activer les quotas sur /export/home, il convient de créer le fichier normal **quotas** à **la racine du point de montage** :

```
# touch /export/home/quotas
```

Le fichier **quotas** doit appartenir à **root** du groupe **root** et posséder des permissions de 600 :

```
# chmod 600 /export/home/quotas
```

Si besoin est, créez un utilisateur **user1** avec le mot de passe **test1234** :

```
# groupadd groupe1
# useradd -m -g groupe1 -d /export/home/user1 user1
# passwd user1
New Password: test1234
Re-enter new Password: test1234
passwd: password successfully changed for user1
```

Configurez votre éditeur par défaut en tant que vi :

```
# which vi
/usr/bin/vi
# echo $EDITOR

# EDITOR=/usr/bin/vi
# export EDITOR
# echo $EDITOR
/usr/bin/vi
```

Mettez en place maintenant des quotas **soft** de 8 000 Ko et **hard** de 10 000 Ko pour l'utilisateur **user1** :

```
# edquota user1
```

Modifiez ce fichier ainsi :

```
fs /export/home blocks (soft = 8001, hard = 10001) inodes (soft = 0, hard = 0)
```

Notez que les chiffres saisis sont 8001 et 10001. La taille des blocs est de 1 Ko. Il est à noter que vous pouvez soit mettre en place un quota en taille, soit mettre en place un

quota basé sur le nombre d'inodes utilisés par l'utilisateur.

Activez maintenant les quotas sur /export/home :

```
# quotaon /export/home
```

La commande **quotaon** peut prendre comme argument soit un point de montage spécifique soit **-a** au cas où les quotas seront activés sur tous les *filesystems* ayant un fichier **quotas** à la racine de leur point de montage.

De cette manière vous avez mis en place un quota **souple** ou *soft* en anglais de 8 000 Ko et un quota **stricte** ou *hard* en anglais de 10 000 Ko pour l'utilisateur user1.

Quand l'utilisateur user1 aura dépassé le quota **souple**, il recevra un message d'avertissement. L'utilisateur aura ensuite 7 jours de grâce pour redescendre en dessous de sa limite **souple**. A l'issu de la période de grâce, sa consommation en taille ou en inodes devient de facto sa limite **stricte**.

Pour modifier la période de grâce, il convient d'utiliser la commande **edquota** avec l'option **-t**.

Si pendant la période de grâce l'utilisateur dépasse le quota **stricte**, il ne pourra plus enregistrer dans /export/home, sauf dans le cas où il supprime des fichiers pour retomber en dessous de la limite **stricte**.

Pour vérifier la cohérence des informations du fichier quotas, il convient d'utiliser la commande **quotacheck** :

```
# quotacheck /export/home
```

La commande **quotacheck** peut prendre comme argument soit un point de montage spécifique soit **-a** au cas où les cohérences de tous les quotas seront effectués sur tous les *filesystems* ayant un fichier **quotas** à la racine de leur point de montage. Attention, sur un *filesystem* contenant un grand nombre de fichiers la commande **quotacheck** peut

prendre un temps considérable.

Pour visualiser les quotas utilisez la commande **repquota** :

```
# repquota /export/home
                      Block limits                       File limits
User           used   soft   hard    timeleft    used   soft   hard    timeleft
user1     --      5   8001  10001                   5      0      0
```

Vérifiez ensuite la véracité de la colonne **used** :

```
# du -h /export/home/user1
   5K   /export/home/user1
```

Créez un utilisateur **user2** avec le mot de passe **test** :

```
# groupadd groupe2
# useradd -m -g groupe2 -d /export/home/user2 user2
# passwd user2
New Password: test1234
Re-enter new Password: test1234
passwd: password successfully changed for user2
```

Vous pouvez mettre en place les mêmes quotas pour d'autres utilisateurs en utilisant la commande **edquota** avec l'option **-p** suivie par le nom de l'utilisateur *proto-type* :

```
# edquota -p user1 user2
# repquota -v /export/home
/dev/md/dsk/d5 (/export/home):
                      Block limits                       File limits
User           used   soft   hard    timeleft    used   soft   hard    timeleft
user1     --      5   8001  10001                   5      0      0
```

```
user2     --      0   8001  10001                    0      0      0
```

Devenez maintenant l'utilisateur user1 et créez un gros fichier :

```
# su - user1
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user1
$ cd /
$ ls -lRa > /export/home/user1/ls1 2>&1
quota_ufs: Warning: over disk limit (pid 1472, uid 100, inum 15, fs /export/home)
quota_ufs: over hard disk limit (pid 1472, uid 100, inum 15, fs /export/home)
$ exit
# du -h /export/home/user1
 9.8M   /export/home/user1
```

Notez les deux messages lors du dépassement de la limite **souple** et **stricte**.

Revenez à l'utilisation de votre tranche 7 pour le montage de /export/home. Placez-y des quotas pour l'utilisateur user1 et détruisez votre volume RAID 5.

<html> Copyright © 2020 Hugh Norris. </html>