Refaire votre monde (avec “make world”)

Nik Clayton

Nik.Clayton@nothing-going-on.demon.co.uk

10 Juillet 1997

Ce document présuppose que vous ayez installé une version du code source de FreeBSD dans le répertoire `/usr/src`. Ce peut être la dernière version de la branche en cours de développement -current, à moins que vous ne restiez simplement à niveau sur la branche -stable. Vous pouvez avoir téléchargé l’instantané le plus récent, ou bien rester à jour en utilisant les mécanismes fournis par CVSup ou CTM.

Dans les deux cas, vous disposez du code source et voulez maintenant mettre à jour votre système.

Il y a un certain nombre d’étapes à effectuer pour y arriver, et quelques pièges à éviter en cours de route. Ce document vous guide pas à pas à chacune de ces étapes.

Il existe aussi des Traductions de ce document dans d’autres langues.

Version française de Frédéric Haby <`frederic.haby@mail.dotcom.fr`>.

1. Avertissements préalables

Faites une sauvegarde : Je n’insisterai jamais assez sur l’importance de faire un sauvegarde de votre système *avant* toute autre chose. Bien qu’il soit facile de “refaire le monde” (N.d.T.: recompiler FreeBSD) - si vous suivez ces instructions, vous ferez forcément des erreurs à un moment ou à un autre, sans compter les erreurs des autres dans l’arborescence des sources qui empêcheraient votre système de redémarrer.

Assurez-vous que vous avez bien fait une sauvegarde. Ayez une disquette de maintenance à portée de la main. Je n’ai jamais eu à les utiliser, et, je touche du bois, espère ne jamais devoir m’en servir, mais prudence est mère de sûreté.

Abonnez-vous à la bonne liste de diffusion : Les branches -stable et -current du code de FreeBSD sont, par nature, *en développement*. Les gens qui participent au projet FreeBSD sont humains, et des erreurs se produisent parfois.

Ces erreurs sont parfois bénignes. Votre système affiche simplement un nouveau message d’avertissement. Elles peuvent aussi être catastrophiques, et empêcher votre système de redémarrer, détruire vos systèmes de fichiers (ou pire).

Quand de tels problèmes se produisent, un avertissement “heads up” est posté sur la liste de diffusion appropriée, décrivant le problème et les machines concernées. Un message “all clear” est diffusé quand le problème est résolu.

Si vous restez à niveau sur -stable ou -current et ne lisez pas <FreeBSD-stable@FreeBSD.ORG> ou <FreeBSD-current@FreeBSD.ORG>, vous allez au devant d’ennuis.

Important : S’il vous plaît, ne me posez pas de questions que vous devriez poster sur les listes -questions, -current, ou -stable. Je n’ai ni le temps ni l’environnement nécessaire pour diagnostiquer des problèmes spécifiques, et n’ai probablement pas votre réponse. Les membres de ces listes de diffusion sont nombreux, expérimentés et serviables, aussi, posez leur vos questions. Adressez-moi cependant vos commentaires, réponses et corrections[1]. Si vous vous adressez à moi au sujet de ce document, *merci* de m’indiquer le numéro de version qui se trouve en haut de cette page, que je sache à quelle version vous faites référence.

Ce document est un effort collectif. De nombreuses personnes y ont participé, elles sont mentionnées dans les Contributions en fin de document.

2. Examinez `/etc/make.conf`

Reportez-vous au fichier `/etc/make.conf`. Il contient les valeurs par défaut utilisées par la commande `make`, qui sera employée pour la recompilation. Elles sont aussi utilisées toutes les fois que vous invoquez `make`, il est donc bon

de vous assurer qu’elles comportent les valeurs appropriées à votre système.

Tout y est, par défaut, en commentaire. Activez les options qui vous paraissent utiles. Pour un utilisateur normal (qui ne développe pas), il faut probablement utiliser les définitions de CFLAGS et NOPROFILE.

> **Version 2.1.7 et antérieures :** Si votre machine dispose d’une unité de calcul en virgule flottante (386DX, 486DX, Pentium et ultérieurs) vous pouvez aussi activer l’option HAVE_FPU.
>
> Cette définition a disparu depuis la version 2.2.2 de FreeBSD.

Examinez les autres définitions (COPTFLAGS, NOPORTDOCS et ainsi de suite) et décidez de celles qui vous conviennent.

3. Mettez à jour le fichier `/etc/group`

Le répertoire `/etc` contient la plupart des informations de configuration de votre système, ainsi que ses procédures de démarrage. Certaines de ces procédures changent d’une version à l’autre de FreeBSD.

Certains fichiers de configuration sont aussi utilisés en permanence par le système. En particulier, `/etc/group`.

Il est arrivé que la phase d’installation de “make world” ait besoin que certains utilisateurs et groupes existent. Il y a de fortes chances qu’ils n’aient pas été définis avant la mise à jour. C’est une source de problèmes.

L’exemple le plus récent concerne l’ajout du groupe “ppp” (renommé par la suite “network”). Les utilisateurs ont vu leur mise à jour avorter à l’installation du sous-sytème `ppp` sous un groupe inexistant (chez eux).

La solution consiste à examiner le fichier `/usr/src/etc/group` et à le comparer à votre propre liste de groupes. S’il y a des groupes dans le nouveau fichier qui ne sont pas dans votre fichier, copiez-les. De même, vous devez renommer tout groupe de votre fichier `/etc/group` qui a le même GID, mais un nom différent, qu’un groupe du fichier `/usr/src/etc/group`.

> **Astuce :** Si vous êtes particulièrement paranoïaque, vous pouvez contrôler votre système pour trouver les fichiers appartenant au groupe que vous renommez ou supprimez. Une commande du type:
>
> ```
> # find / -group GID -print
> ```
>
> vous donnera la liste des fichiers appartenant au groupe *GID* (qui peut être un nom de groupe ou un identifiant numérique).

4. Passez en mode mono-utilisateur

Il vaut mieux recompiler le système en mode mono-utilisateur. En dehors du fait que cela ira un peu plus vite, la réinstallation va modifier un grand nombre de fichiers systèmes importants, tous les binaires de base, les bibliothèques, les fichiers inclus et ainsi de suite. Les modifier quand le système est en service (en particulier s’il y a des utilisateurs connectés à ce moment là), c’est aller au devant de problèmes.

Cela dit, si vous avez confiance en vous, vous pouvez vous en passer.

> **Version 2.2.5 et ultérieure :** Comme décrit plus bas, la version 2.2.5 et les suivantes de FreeBSD séparent la recompilation et l’installation. Vous pouvez dès lors *compiler* le nouveau système en mode multi-utilisateurs et passer en mode mono-utilisateur juste pour l’installer.

En tant que super-utilisateur, vous pouvez passer la commande:

```
# shutdown now
```

sur un système en fonctionnement, pour passer en mode mono-utilisateur.

Ou bien, redémarrez le système, et, à l’invite de démarrage, entrez l’indicateur `-s`. Le système redémarrera en mode mono-utilisateur. A l’invite de l’interpréteur de commandes, exécutez:

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

pour effectuer la vérification des systèmes de fichiers, remontez `/` en mode lecture/écriture, et monter tous les autres systèmes de fichiers UFS listés dans le fichier `/etc/fstab`, puis activez la pagination.

5. Effacez `/usr/obj`

Les composants du système reconstruit sont au fur et à mesure placés dans les sous-répertoires de `/usr/obj` (par défaut). Ces répertoires masquent ceux de `/usr/src`.

Vous pouvez accélérer le travail de “make world”, et peut-être vous éviter quelques maux de tête en supprimant aussi le répertoire `/usr/obj`.

Certains fichiers dans `/usr/obj` sont marqués immuables (reportez-vous aux pages de manuel de `chflags(1)` pour plus de détails). Il faut d’abord supprimer cet indicateur.

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

6. Recompilez les sources et installez le nouveau

système

6.1. Toutes versions

Vous devez être dans le répertoire `/usr/src`, donc:

```
# cd /usr/src
```

(à moins, bien sûr, que votre code source soit ailleurs, auquel cas vous devez aller dans le répertoire correspondant).

Pour recompiler le système, vous utilisez la commande make(1). Cette commande lit ses instructions dans le fichier `Makefile`, qui décrit comment reconstruire les modules qui constituent FreeBSD, dans quel ordre, et ainsi de suite.

Le format général de la commande que vous taperez sera:

```
# make -x -DVARIABLE cible
```

Dans cet exemple, `-x` est une option que vous donnez à make(1). Reportez-vous aux pages de manuel pour connaître les options disponibles.

`-DVARIABLE` transmet une variable au fichier `Makefile`. Le comportement de `Makefile` est défini par ces variables. Ce sont les mêmes variables que l’on trouve dans `/etc/make.conf`, et c’est un autre moyen de les positionner.

Par exemple:

```
# make -DNOPROFILE=true cible
```

est une autre manière de dire qu’il ne faut pas compiler les bibliothèques profilées et correspond aux lignes:

```
NOPROFILE=    true
#   Avoid compiling profiled librairies
#   Ne pas compiler les bibliothèques profilées
```

du fichier `/etc/make.conf`.

`cible` dit à make(1) ce que vous voulez faire. Chaque `Makefile` définit un certains nombre de “cibles” différentes, et votre choix de cibles détermine ce qui se passe.

Il y a des cibles définies dans le fichier `Makefile` que vous n’avez pas à employer. Ce sont des étapes intermédiaires utilisées par le processus de recompilation pour décomposer les étapes en sous-étapes.

La plupart du temps, vous n’aurez pas besoin de donner d’options à make(1), et votre commande sera simplement:

```
# make cible
```

6.2. Enregistrez le résultat

C’est une bonne idée d’enregistrer le résultat de make(1) dans un ficher. Si quelque chose se passe mal, vous aurez une trace des messages d’erreur, et la liste complète de ce qui a été fait. Même si cela ne vous aide pas à diagnostiquer ce qui n’a pas marché, cela peut aider les autres si vous soumettez votre problème sur une des listes de diffusion de FreeBSD.

La meilleure façon de faire cela est d’utiliser la commande script(1), avec en paramètre le nom du fichier où enregistrer les résultats. Vous devez faire cela juste avant de recompiler le système, et taper **`exit`** une fois que c’est terminé.

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make world
... compile, compile, compile ...
# exit
Script done, ...
```

Si vous le faites, *n’enregistrez pas* les résultats dans `/tmp`. Ce répertoire peut être vidé au prochain redémarrage du sytème. Il vaut mieux les mettre dans `/var/tmp` (comme dans l’exemple précédent) ou dans le répertoire utilisateur de root.

6.3. Version 2.2.2 et antérieure

`/usr/src/Makefile` contient la cible “world”, qui recompile tout le système et l’installe.

Utilisez donc:

```
# make world
```

6.4. Version 2.2.5 et ultérieure

A partir de la version 2.2.5 de FreeBSD (de fait, c’est la première version à avoir été créée sur la branche -current, puis rapatriée dans la branche -stable entre les versions 2.2.2 et 2.2.5) la cible “world” a été décomposée en deux: “buildworld” et “installworld”.

Comme leurs noms l’indiquent, “buildworld” reconstruit la nouvelle arborescence dans `/usr/obj`, et “installworld” l’installe sur la machine.

C’est très utile pour deux raisons. Tout d’abord, vous pouvez recompiler en toute sûreté, sans toucher aux composants du système actuel. Le processus est “autonome”. Vous pouvez donc exécuter “buildworld” sur une machine en mode multi-utilisateurs sans redouter d’effets fâcheux. Je vous recommande néanmoins de toujours exécuter l’étape “installworld” en mode mono-utilisateur.

En second lieu, cela vous permet d’utiliser des systèmes de fichiers montés par NFS pour mettre à jour les autres machines de votre réseau. Si vous avez trois machines, A, B, et C que vous voulez mettre à jour, exécutez `make`

`buildworld` et `make installworld` sur A. B et C doivent alors monter par NFS `/usr/src` et `/usr/obj` depuis A, et vous pouvez alors exécuter `make installworld` pour installer le système recompilé sur B et C.

La cible “world” existe toujours et vous pouvez l’utiliser exactement comme avec la version 2.2.2. `make world` exécute `make buildworld` suivi de `make installworld`.

> **Note :** Si vous utilisez séparement `make buildworld` et `make installworld`, vous devez donner à chaque fois les mêmes paramètres à make(1).
>
> Par exemple, si vous exécutez:
>
> ```
> # make -DNOPROFILE=true buildworld
> ```
>
> vous devrez ensuite installer les résultats avec:
>
> ```
> # make -DNOPROFILE=true installworld
> ```
>
> sinon il essayera d’installer les bibliothèques profilées qui n’ont pas été recompilées à l’étape `make buildworld`.

6.5. -current et ultérieure

Si vous êtes sur la branche -current, vous pouvez aussi donner l’option `-j` à `make`. Cela permet à `make` d’exécuter plusieurs programmes simultanément.

C’est particulièrement utile sur une machine avec plusieurs processeurs. Néanmoins, comme la compilation est plus gourmande en Entrées/Sorties qu’en CPU, c’est aussi utile sur une machine mono-processeur.

Typiquement, sur une machine mono-processeur, vous exécuteriez:

```
# make -j4 cible
```

pour autoriser make(1) à exécuter 4 programmes simultanément. Les constatations empiriques postées sur les listes de diffusion montrent que c’est en général ce qui apporte le plus de gain en performances.

Si vous avez une machine multi-processeurs et que vous avez configuré un noyau SMP, essayez des valeurs entre 6 et 10 et voyez quel bénéfice vous en tirez.

N’oubliez pas que c’est toujours expérimental (au moment où j’écris ceci), et que des modifications de l’arborescence des sources rendent parfois cette possibilité inutilisable. Si vous n’arrivez pas à recompiler avec ce paramètre, essayez sans avant de signaler votre problème.

6.6. Durée

En supposant que tout ce passe bien, il vous faudra attendre entre une heure et demie et une demi-journée.

En règle générale, un P6 200MHz avec plus de 32MB de RAM et des disques SCSI corrects exécutera `make world` en environ une heure et demie. Un P133 32MB prendra 5 à 6 heures. Revoyez ces chiffres à la baisse si vos machines sont plus lentes…

7. Mettez à jour `/etc`

Recompiler le système ne met pas à jour certains répertoires (en particulier, `/etc`, `/var` et `/usr`) pour y installer des fichiers de configuration nouveaux ou modifiés. Il vous faudra le faire à la main, à vue, et en utilisant à bon escient la commande `diff`.

Vous ne pouvez pas vous contenter de copier les fichiers de `/usr/src/etc` dans `/etc` pour que cela marche. Certains de ces fichiers doivent d'abord être "installés". En effet le répertoire `/usr/src/etc` *n'est pas* une simple copie de ce que devrait contenir votre répertoire`/etc`. De plus, il y a des fichiers qui doivent être dans `/etc` et ne sont pas dans `/usr/src/etc`.

La façon la plus simple de procéder est d'installer les fichiers dans un nouveau répertoire, puis de passer en revue les différences.

> **Sauvegardez vos fichiers actuels dans `/etc` :** Bien qu'en principe rien ne sera modifié automatiquement dans ce répertoire, prudence est mère de sûreté. Copiez donc votre répertoire `/etc` dans un endroit sûr. Quelque chose du genre:
>
> ```
> # cp -Rp /etc /etc.old
> ```
>
> fera l'affaire (`-R` fait une copie récursive, `-p` conserve la date, les autorisations des fichiers et ainsi de suite).

Vous devez créer un jeu de répertoires provisoires pour y installer les fichiers de `/etc` et autres. En général, je les mets dans `/var/tmp/root`; il y a un certain nombre de sous-répertoires à créer. Pour ce faire, exécutez:

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

qui va créer l'arborescence nécessaire et y installera les fichiers. Un grand nombre des sous-répertoires créés dans `/var/tmp/root` seront vides et devront être supprimés. La façon la plus simple de le faire est:

```
# cd /var/tmp/root
# find -d . -type d | /usr/bin/perl -lne \
  'opendir(D,$_);@f=readdir(D);rmdir if $#f == 1;closedir(D);'
```

qui fait une recherche en profondeur, examine chaque répertoire, et s'il ne contient que 2 fichiers ("1" n'est pas une faute de frappe dans la procédure), i.e. "`.`" et "`..`" supprime le répertoire.

`/var/tmp/root` contient maintenant tous les fichiers à installer à l'endroit requis sous `/`. Vous devez ensuite examiner chacun de ces fichiers pour voir en quoi ils diffèrent de vos propres fichiers.

Notez que certains des fichiers qui ont été installés dans `/var/tmp/root` commencent par un "." Au moment où j'écris ceci, les seuls fichiers concernés sont les fichiers d'initialisation des interpréteurs de commandes dans `/var/tmp/root/` et `/var/tmp/root/root/`, mais il pourrait y en avoir d'autres (cela dépend de quand vous lirez ces lignes). Assurez-vous d'utiliser `ls -a` pour ne pas les oublier.

La manière la plus simple de procéder est d'utiliser la commande `diff` pour comparer deux fichiers.

Par exemple:

```
# diff /etc/shells /var/tmp/root/etc/shells
```

vous indiquera les différences entre votre fichier `/etc/shells` et le nouveau fichier `/etc/shells`. A partir de là, décidez si vous allez reporter les modifications que vous y avez apportées ou si vous allez simplement recopier le nouveau fichier.

`http://www.nothing-going-on.demon.co.uk/FreeBSD/make-world/dircmp.pl` est une petite procédure Perl (Perl 4.036, qui est installé par défaut à partir de la version 2.0 de FreeBSD) qui compare les fichiers de deux répertoires (`/etc` et `/var/tmp/root/etc` par défaut) et liste les fichiers absents ou différents dans les deux répertoires.

Donnez au nouveau répertoire (`/var/tmp/root`) un nom qui inclue une date, pour pouvoir facilement comparer différentes versions : Si vous recompilez fréquemment votre système, vous devrez aussi souvent mettre à jour `/etc`, ce qui peut devenir une vrai corvée.

Vous pouvez accélérer le processus en gardant une copie du dernier jeu de fichiers reportés dans `/etc`. La procédure suivante vous suggère comment faire.

1. Recompilez le système comme d'habitude. Au moment de mettre à jour `/etc` et les autres répertoires, donnez au répertoire cible un nom basé sur la date du jour. Si vous faisiez cela le 14 Février 1998, vous pouviez procéder comme suit:

   ```
   # mkdir /var/tmp/root-980214
   # cd /usr/src/etc
   # make DESTDIR=/var/tmp/root-980214 \
       distrib-dirs distribution
   ```

2. Reportez les modifications depuis ce répertoire comme décrit plus haut.

   Ne supprimez pas le répertoire `/var/tmp/root-980214` quand vous aurez terminé.

3. Quand vous chargerez la version la plus récente des sources et la recompilerez, faites de même. Vous aurez alors un nouveau répertoire, `/var/tmp/root-980221` par exemple (si vous faites une mise à jour chaque semaine).

4. Vous pouvez maintenant voir les modifications intervenues d'une semaine à l'autre avec quelque chose comme:

   ```
   # cd /var/tmp
   # diff -r root-980214 root-980221
   ```

   qui vous donnera les différences entre tous les fichiers des deux répertoires.

   Typiquement, il y aura beaucoup moins de différences qu'entre `/var/tmp/root-980221/etc` et `/etc`. Comme il y a beaucoup moins de différences, il est beaucoup plus facile de les reporter dans le répertoire `/etc`.

5. Vous pouvez maintenant supprimer le plus ancien des deux répertoires `/var/tmp/root-*`:

   ```
   # rm -rf /var/tmp/root-980214
   ```

6. Répétez l'opération chaque fois que vous devez reporter des modifications dans `/etc`.

Vous pouvez utiliser la commande `date(1)` pour automatiser la génération des noms de répertoires. Par exemple:

```
# mkdir /var/tmp/root-`date "+%y%m%d"`
```

crée un répertoire dont le nom dépend de l’année, du mois et du jour.

8. Mettez à jour `/dev`

DEVFS : Si vous utilisez DEVFS ce qui suit ne vous concerne probalement pas.

Pour des raisons de sécurité, cette mise à jour se fait en plusieurs étapes.

Copiez tout d’abord `/var/tmp/root/dev/MAKEDEV` dans `/dev`.

```
# cp /var/tmp/root/dev/MAKEDEV /dev
```

Prenez maintenant un instantané de l’état de votre répertoire `/dev`. Il doit indiquer les propriétaires, les droits et les codes majeur et mineur de chaque fichier spécial de périphérique, mais pas leur date de dernière mise à jour. La façon la plus facile de procéder est d’utiliser la commande `awk` pour éliminer les informations inutiles:

```
# cd /dev
# ls -l | awk '{print $1, $2, $3, $4, $5, $6, $NF}' > /var/tmp/dev.out
```

Ensuite, recréez tous les fichiers spéciaux de périphériques:.

```
# sh MAKEDEV all
```

Reprenez un instantané de l’état de votre répertoire, cette fois-ci dans `/var/tmp/dev2.out`. Comparez maintenant ces deux instantanés pour voir si certains fichiers spéciaux de périphériques n’ont pas été recréés. Il ne devrait pas en manquer, mais prudence est mère de sûreté.

```
# diff /var/tmp/dev.out /var/tmp/dev2.out
```

Il manquera peut-être des descripteurs de partitions, il vous faudra alors exécuter des commandes du type :

```
# sh MAKEDEV sd0s1
```

pour les recréer. Les détails dépendent de votre installation.

9. Mettez à jour `/stand`

> **Note :** Cette étape n’est décrite que pour être exhaustif, elle peut être omise sans danger.

Pour être exhaustif, vous pouvez aussi mettre à jour les fichiers de `/stand`. Ces fichiers sont des liens physiques sur le programme `/stand/sysinstall`. L’édition de liens de cet exécutable doit être statique, pour qu’on puisse l’utiliser sans qu’aucun autre système de fichiers (et en particulier `/usr`) ne soit monté.

```
# cd /usr/src/release/sysinstall
# make all install
```

> **Sources antérieurs au 2 Avril 1998 :** Si votre code source date d’avant le 2 Avril 1998, ou que la version de votre `Makefile` est inférieure à 1.68 (pour FreeBSD-current et les systèmes 3.x) ou à 1.48.2.21 (pour les systèmes 2.2.x), vous devrez ajouter l’option **`NOSHARED=yes`** comme suit:
>
> ```
> # make NOSHARED=yes all install
> ```

10. Compilez et installez un nouveau noyau

Pour tirer pleinement parti de votre nouveau système, vous devez recompiler le noyau. C’est pratiquement indispensable, parce que des structures de données peuvent avoir changé, et des programmes comme `ps` et `top` ne marcheront pas tant que le système et le noyau ne seront pas au même niveau de version.

Suivez les instructions du "manuel" pour compiler un nouveau noyau. Si vous avez déjà recompilé un noyau personnalisé examinez en détail le fichier de configuration `LINT` pour voir s’il y a de nouvelles options dont vous pourriez tirer parti.

Une version précédente de ce document suggérait de redémarrer le système avant de recompiler le noyau. C’est un erreur parce que :

- Des commandes comme `ps`, `ifconfig` and `sysctl` peuvent ne plus fonctionner. Dans ce cas, votre machine ne peut plus se connecter au réseau.
- De même, des utilitaires essentiels comme `mount` peuvent aussi être inutilisables, auquel cas `/`, `/usr` et ainsi de suite, ne peuvent plus être montés. Il y a peu de chances que cela arrive si vous êtes sur la branche -stable, mais c’est plus probable sur la branche -current après des modifications importantes.
- Les LKMs ("Loadable Kernel Modules" - modules du noyau à chargement dynamique) reconstruits en même temps que “world” peuvent "planter" un noyau plus ancien.

Pour toutes ces raisons, il vaut mieux recompiler et installer un nouveau noyau avant de redémarrer.

Vous devez recompiler le noyau après avoir terminé **`make world`** (ou **`make installworld`**). Si vous ne le faites pas (peut-être voulez-vous vous assurer que le noyau compile avant de mettre à jour le système), vous pourriez avoir des problèmes. Cela parce que votre commande `config` n’est pas à niveau avec les sources du noyau.

Dans ce cas, exécutez:

```
# /usr/obj/usr/src/usr.sbin/config NOM_DU_NOYAU
```

pour recompiler le noyau avec la nouvelle version de `config`. Cela ne marchera peut-être pas à tous les coups. Il est recommandé d’en finir avec **`make world`** (ou **`make installworld`**) avant de compiler un nouveau noyau.

11. Redémarrez

Vous en avez fini. Après avoir vérifié que tout semble être en place, vous pouvez maintenant redémarrez votre système. Un simple fastboot(8) devrait suffire.

```
# fastboot
```

12. C’est fini

Vous devriez maintenant avoir mis à jour avec succès votre système FreeBSD. Félicitations.

Vous aurez peut-être de petits problèmes si des détails vous ont échappés. Par exemple, il m’est arrivé d’effacer le fichier `/etc/magic` au moment de la mise à jour de `/etc`, de ce fait, la commande `file` ne marchait plus. Un petit moment de réflexion et j’ai trouvé que:

```
# cd /usr/src/usr.bin/file
# make all install
```

suffisait à régler ce problème.

13. Questions?

13.1. Dois-je refaire le monde à chaque évolution?

Il n’y a pas de réponse toute faite à cette question, tout dépend de la nature des évolutions. Je viens juste, par exemple, d’exécuter CVSup, et les fichiers suivants ont été modifiés depuis ma dernière recompilation:

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/share/mk/bsd.port.mk
```

Il n’y a pas là matière à ce que je recompile mon système. Je vais simplement aller dans les bons sous-répertoires et exécuter `make all install`, et c’est à peu près tout. Mais s’il y a des évolutions importantes, par exemple sur

`src/lib/libc/stdlib` alors ou je referais, le monde, ou je recompilerais au moins toutes les parties du système qui sont liées statiquement (de même que tout ce que je pourrais avoir ajouté qui serait lié statiquement).

En fin de journée, c’est à vous de voir. Vous préférerez peut-être recompiler votre système tous les quinze jours, et laisser les modifications s’empiler pendant ces quinze jours. Ou bien vous préférez ne recompiler que ce qui a changé et vous faire confiance pour repérer ce qui en dépend.

Et, bien sûr, cela dépend de la fréquence avec laquelle vous voulez faire vos mises à jour, et de si vous êtes sur la branche -stable ou sur la branche -current.

13.2. Ma compilation échoue avec de nombreuses erreurs "signal 12" (ou tout autre numéro de signal)

Cela indique généralement un problème matériel. (Re)faire le monde est un bon moyen de mettre votre matériel sous pression, et mettra souvent en évidence des défaillances de la mémoire vive. Cela se manifeste normalement de soi-même: le compilation échoue en recevant de mystérieux signaux.

Vous pouvez vous en assurer si vous relancer la compilation et qu’elle échoue en un endroit différent.

Dans ce cas, vous ne pouvez guère faire autre chose que d’intervertir les différents composants de votre matériel pour déterminer lequel est en cause.

13.3. Puis-je détruire `/usr/obj` après avoir fini?

Tout dépend de comment vous voulez refaire le monde par la suite.

`/usr/obj` contient tous les fichiers objets générés à la compilation. Normalement, une des premières étapes de “make world” est de supprimer ce répertoire et de repartir à zéro. Dans ce cas, conserver ce répertoire `/usr/obj` après en avoir terminé ne sert pas à grand chose, alors que vous économiseriez pas mal d’espace disque (au jour d’aujourd’hui environ 150Mo).

Néanmoins, si vous savez ce que vous faites, vous pouvez faire en sorte que “make world” saute cette étape. Les reconstructions ultérieures seront beaucoup plus rapides, car la plupart des sources n’auront pas besoin d’être recompilés. Le revers de la médaille est que des problèmes de dépendance subtils peuvent se manifester, provoquant l’échec de votre recompilation de manière étrange. Cela génère fréquemment du bruit sur les listes de diffusion de FreeBSD, quand quelqu’un se plaint que sa mise à jour a échoué, sans réaliser qu’il a tenté de brûler les étapes.

Si vous aimez vivre dangereusement, passez le paramètre “NOCLEAN” à `make`, comme suit:

```
# make -DNOCLEAN world
```

13.4. Une recompilation interrompue peut-elle être reprise?

Tout dépend de jusqu’où vous êtes allé avant de rencontrer un problème.

En général (mais ce n’est pas une règle absolue) “make world” crée de nouveaux exemplaires des utilitaires de base (comme `gcc`, et `make`) et des bibliothèques système. Ces outils et bibliothèques sont ensuite installés. Ils sont ensuite

utilisés pour se reconstruire eux-mêmes, et installés de nouveau. Le système entier (y compris maintenant les outils usuels, comme `ls` ou `grep`) est ensuite recompilé avec les nouveaux outils et bibliothèques de base.

Si vous en êtes à cette dernière étape, et que vous le savez (parce que vous avez consulté les résultats que vous avez enregistrés) alors vous pouvez (avec une bonne chance de réussite) faire:

```
... régler le problème ...
# cd /usr/src
# make -DNOCLEAN all
```

qui ne détruira pas les résultats du travail qu’a déjà effectué “make world”.

Si vous voyez le message :

```
--------------------------------------------------------------
 Building everything..
--------------------------------------------------------------
```

dans les comptes-rendus de “make world”, cette façon de procéder est probablement sûre.

Si vous ne voyez pas ce message, ou doutez de vous, alors prudence est mère de sûreté, et il vaut mieux tout reprendre depuis le début.

13.5. Puis-je utiliser une seule machine de *référence* pour mettre à jour plusieurs machines (NFS)?

On pose souvent la question sur les listes de diffusion de FreeBSD de savoir s’il est possible de tout compiler sur une seule machine puis d’installer les résultats de cette compilation sur d’autres machines du réseau avec `make install`.

C’est quelque chose que je n’ai jamais fait, aussi les indications qui suivent m’ont-elles été données par d’autres ou déduites des `Makefile`s.

La marche exacte à suivre dépend de votre version de FreeBSD.

> **Note :** Vous devrez encore mettre à jour `/etc` et `/dev` sur les machines cibles après cette étape.

13.5.1. Version 2.1.7 et antérieures

Dans un message adressé à questions@freebsd.org, Antonio Bemfica a suggéré la méthode suivante:

```
Date: Thu, 20 Feb 1997 14:05:01 -0400 (AST)
From: Antonio Bemfica <bemfica@militzer.me.tuns.ca>
To: freebsd-questions@freebsd.org
Message-ID: <Pine.BSI.3.94.970220135725.245C-100000@militzer.me.tuns.ca>

Josef Karthauser a demandé:

> Quelqu’un a-t-il la bonne méthode pour mettre à jour
```

```
> les machines d'un réseau?

D'abord make world, etc... sur votre machine de référence
Ensuite, montez / and /usr sur la machine distante:

machine_de_référence% mount machine_distante:/    /mnt
machine_de_référence% mount machine_distante:/usr /mnt/usr

Ensuite, faites make install avec /mnt comme cible:

machine_de_référence% make install DESTDIR=/mnt

Répétez cela pour chaque machine de votre réseau.
Cela marche très bien dans mon cas.

Antonio
```

Ce mécanisme ne fonctionne (autant que je sache) que si vous pouvez écrire sur `/usr/src` sur le serveur NFS, car ce devait être la cible d’“install” avec la version 2.1.7 et les précédentes.

13.5.2. Version 2.2.0 and ultérieures

Entre les versions 2.1.7 et 2.2.0 la cible “reinstall” a été introduite. Vous pouvez utiliser la méthode décrite ci-dessus pour la version 2.1.7, en remplaçant “install” par “reinstall”.

Cela *ne demande plus* de droits d’écriture sur le répertoire `/usr/src` du serveur NFS.

> **Note :** Un bogue est apparu avec cette cible entre les versions 1.68 et 1.107 du `Makefile`, qui impliquait qu’il *fallait* avoir les droits d’écriture. Ce bogue a été corrigé avant la diffusion de la version 2.2.0 de FreeBSD, mais peut encore poser problème si vous avez un vieux serveur sous -stable de cette époque.

13.5.3. Version 2.2.5 et ultérieure

Comme décrit plus haut, les cibles “buildworld” et “installworld” peuvent être employées pour recompiler sur une machine, puis monter par NFS `/usr/src` et `/usr/obj` sur la machine distante et y installer le nouveau système.

13.6. Comment puis-je accélérer `make world`?

- Passez en mode mono-utilisateur.
- Mettez les répertoires `/usr/src` et `/usr/obj` sur des systèmes de fichiers et des disques différents. Si possible, installez ces disques sur des contrôleurs différents.
- Mieux encore, mettez ces systèmes de fichiers sur plusieurs disques et utilisez “ccd” ("concatenated disk driver" = pilote de disques concaténés).

- Ne compilez pas les bibliothèques profilées (mettez “NOPROFILE=true” dans `/etc/make.conf`). Vous n’en avez certainement pas besoin.
- Dans `/etc/make.conf`, positionnez aussi “CFLAGS” à quelque chose comme “-O -pipe”. L’optimisation “-O2” est bien plus lente, et la différence d’optimisation entre “-O” et “-O2” est en général négligeable. “-pipe” dit au compilateur d’utiliser des tuyaux (“pipes”) à la place de fichiers, ce qui économise des accès disque (mais utilise plus de mémoire).
- Donnez l’option `-j<n>` au compilateur (Si vous avez une version suffisamment récente de FreeBSD) pour exécuter plusieurs programmes en parallèle. Cela améliore les choses, que vous ayez une machine mono- ou multi-processeurs.
- Le système de fichiers qui contient `/usr/src` peut être monté (ou remonté) avec l’option “noatime”. De cette manière, les dates de dernier accès aux fichiers ne sont pas enregistrées sur disque. Vous n’avez de toute façon probablement pas besoin de cette information.

  Note : “noatime” existe à partir de la version 2.2.0.

```
# mount -u -o noatime /usr/src
```

  Note : Cet exemple suppose que `/usr/src` constitue à lui seul un système de fichiers. Si ce n’est pas le cas (s’il fait partie de `/usr` par exemple) vous devez indiquez le point de montage de ce système de fichiers, et non `/usr/src`.

- Le système de fichiers où se trouve `/usr/obj` peut être monté (ou remonté) avec l’option “async”. Les écritures sur disque se font alors de façon asynchrone. En d’autres termes, le programme reprend immédiatement la main, mais l’écriture se fait quelques secondes après. Les accès disque sont ainsi groupés, et le gain en performances est spectaculaire.

  Note : Rappelez-vous que cette option rend votre système de fichiers plus fragile. Avec cette option, les risques sont accrus qu’en cas de coupure d’alimentation, le système de fichiers soit irrécupérable quand la machine redémarrera.

  S’il n’y a que `/usr/obj` sur ce système de fichiers, ce n’est pas un problème. S’il contient des informations plus sensibles, assurez-vous que vos sauvegardes soient à jour avant d’activer cette option.

```
# mount -u -o async /usr/obj
```

  Note : Comme auparavant, si `/usr/obj` ne constitue pas un système de fichiers en soit, remplacez-le dans l’exemple par le nom du point de montage qui convient.

14. Traductions

14.1. Document original

L’original de ce document se trouve sur
http://www.nothing-going-on.demon.co.uk/FreeBSD/make-world/make-world.html.

14.2. Japonais

MAEKAWA Masahide a traduit ce document en Japonais. Sa traduction est disponible à l’adresse
http://www.rr.iij4u.or.jp/~bishop/FreeBSD/mw.html.

15. Contributions

Les personnes suivantes ont contribué d’une façon ou d’une autre à la rédaction de ce document. Soit directement en suggérant des modifications ou des améliorations ou en signalant des erreurs, soit par leurs messages sur les listes de diffusion de FreeBSD, où j’ai puisé sans scrupule de l’information. Mes remerciements à tous.

- Antonio Bemfica, bemfica@militzer.me.tuns.ca (mailto:bemfica@militzer.me.tuns.ca)
- Sue Blake, sue@welearn.com.au (mailto:sue@welearn.com.au)
- Brian Haskin, haskin@ptway.com (mailto:haskin@ptway.com)
- Kees Jan Koster, kjkoster@kjkoster.org (mailto:kjkoster@kjkoster.org)
- A Joseph Kosy, koshy@india.hp.com (mailto:koshy@india.hp.com)
- Greg Lehey, grog@lemis.com (mailto:grog@lemis.com)
- Wes Peters, softweyr@xmission.com (mailto:softweyr@xmission.com)
- Joseph Stein, joes@wstein.com (mailto:joes@wstein.com)
- Studded, studded@dal.net (mailto:studded@dal.net)
- Axel Thimm Axel.Thimm@physik.fu-berlin.de (mailto:Axel.Thimm@physik.fu-berlin.de)
- Matthew Thyer Matthew.Thyer@dsto.defence.gov.au (mailto:Matthew.Thyer@dsto.defence.gov.au)

Notes

1. En anglais !