Le Guide du Nouveau “*Committer*”

Projet de Documentation de FreeBSD

$FreeBSD: head/fr_FR.ISO8859-1/articles/committers-guide/article.xml 41645 2013-05-17 18:49:52Z gabor $

Septembre 1999

Nouveau “*committer*”, bienvenue dans l’équipe de développement de FreeBSD !

L’objectif de cette documentation est de vous orienter sur la façon d’utiliser CVS sur la machine d’archive centrale de FreeBSD. Il est présumé que vous avez déjà une connaissance de base de CVS, quoique des informations de référence, des guides et Questions Fréquemment Posées soient disponibles à l’adresse : http://www.cyclic.com/cyclic-pages/books.html

Bonne chance, et bienvenue à bord !

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

1. Détails d’organisation

Machine d’archive principale	`freefall.FreeBSD.org`
Machine d’archive internationale pour les codes de cryptographie	`internat.FreeBSD.org`
Méthode de connexion	ssh(1)
Répertoire CVSROOT	/home/ncvs
Répertoire CVSROOT pour la version internationale des codes de cryptographie	/home/cvs.crypt
Administrateurs des archives CVS principales	John Polstra <`jdp@FreeBSD.org`> et Peter Wemm <`peter@FreeBSD.org`> ainsi que Satoshi Asami <`asami@FreeBSD.org`> pour `ports/`
Administrateur des archives CVS pour la version internationale des codes de cryptographie	Mark Murray <`markm@FreeBSD.org`>
Liste de diffusion	<`cvs-committers@FreeBSD.org`>
Etiquettes CVS importantes	RELENG_3 (3.x-STABLE), HEAD (-CURRENT)

Il vous est demandé d’utiliser ssh(1) ou telnet(1) et Kerberos 5 pour vous connecter aux machines d’archive. Ces méthodes sont globalement plus sûres qu’un simple telnet(1) ou rlogin(1) parce que la négociation de l’authentification est cryptée. Par défaut ssh(1) crypte toute la session. Les utilitaires disponibles ssh-agent(1) et scp(1) sont aussi bien plus pratiques. Si vous ne connaissez pas ssh(1), reportez-vous à la Section 7.

2. Opérations CVS

Les opérations CVS se font habituellement en se connectant à `freefall`, vérifiant que votre variable d’environnement `CVSROOT` est bien positionnée à `/home/ncvs`, et en effectuant les opérations d’extraction (*check-out*) et de mise à jour (*check-in*) nécessaires. Si vous avez quelque chose de entièrement nouveau à ajouter (un nouveau logiciel porté, du source d’origine externe, etc.), il existe une procédure appelée « easy-import » qui facilite cette opération. Elle ajoute automagiquement une entrée pour le nouveau module, fait ce qu’il faut via `cvs import`, etc. – exécutez-la sans arguments et elle vous demandera tout ce qu’elle a besoin de savoir.

Si vous avez la pratique de CVS à distance et vous considérez relativement opérationnel sur CVS en général, vous pouvez aussi effectuer les opérations CVS directement depuis votre machine avec une copie local à jour des sources. N’oubliez cependant pas de positionner `CVS_RSH` à *ssh* de façon à utiliser un moyen de transmission sécurisé et fiable. D’une autre côté, si vous ne savez pas ce que cela veut dire, tenez-vous en s’il vous plaît à la méthode qui consiste à vous connecter à `freefall` et mettre en place vos modifications avec patch(1).

Si vous avez à utiliser les opérations `add` et `delete` pour faire en fait une opération « mv », il faut une copie sur l’archive plutôt que votre commande CVS `add` suivie d’un `delete`. Dans ce cas, un Administrateur CVS copiera le(s) fichier(s) là où il(s) doi(ven)t aller et vous avertira une fois qu’il l’aura fait. Le but de la copie dans les archives est de conserver l’historique des modifications, la journalisation. Le Projet FreeBSD accorde une grande importance à l’historique du projet que CVS nous permet de conserver.

3. Conventions et Traditions

Les Administrateurs CVS (Peter Wemm et John Polstra) sont les « propriétaires » des archives CVS et sont responsables de chaque et de *toute* modification directe de celles-ci pour mise au propre ou rectification d’erreur CVS dûe à un *committer*. Personne d’autre ne doit intervenir directement sur les archives. Si vous faites un fausse manipulation, une importation incorrecte ou vous trompez d’étiquette par exemple, n’essayez **pas** de la rectifier vous-même ! Envoyez immédiatement un courrier électronique ou téléphonez à John ou Peter et expliquez leur le problème. Satoshi Asami est aussi Administrateur CVS pour la partie `ports/` de l’arborescence. Mark Murray est l’administrateur des archives internationales pour les logiciels de cryptographie, en Afrique du Sud.

Si vous êtes nouveau *committer*, la première chose à faire est de vous ajouter vous-même à la liste des développeurs (section 28.2) du Manuel de Référence. Extraire le manuel de référence et ajouter une entrée à la liste est relativement facile, mais c’est néanmoins un bon test initial de vos compétences CVS. Si vous pouvez le faire, vous n’aurez probablement pas de problèmes par la suite.

L’étape suivante consiste à vous présenter aux autres *committers*, sans quoi ils n’auront aucune idée de qui vous êtes et à quoi vous travaillez. Il n’est pas nécessaire de rédiger une biographie exhaustive, un paragraphe ou deux suffiront, pour dire qui vous êtes et à quoi vous comptez travailler sur FreeBSD. Envoyez-les par courrier électronique à <`cvs-committers@FreeBSD.org`> et vous serez prêt à commencer à travailler !

N’oubliez pas aussi de vous connecter à `hub.FreeBSD.org` et de vous y créez un fichier `/var/forward/`*`utilisateur`* (où *`utilisateur`* est votre nom d’utilisateur), qui contienne votre adresse de courrier électronique principale où vous souhaitez que le courrier électronique adressé à *`votre_nom_d_utilisateur`*@FreeBSD.org vous soit redirigé. Les boîtes aux lettres vraiment volumineuses qui demeurent en en permanence sur `hub` sont souvent « accidentellement » tronquées sans avertissement, redirigez donc votre courrier, ou lisez-le, et vous ne le perdrez pas.

Tous les nouveaux *committers* ont un mentor qui leur est assigné les premiers mois. Votre mentor est plus ou moins chargé de vous expliquer tout ce que vous ne comprenez pas bien et est aussi responsable de ce que vous faites à vos

débuts. Si vous faites une soumission erronée, c'est votre mentor qui sera ennuyé et vous devriez probablement vous fixer comme ligne de conduite de faire passer vos premières soumissions par lui avant de les intégrer aux archives.

Toutes les soumissions doivent être intégrées d'abord à `-CURRENT`, avant d'aller dans `-STABLE`. Aucune nouvelle fonctionnalité ou modification à haut risque ne devrait être intégrée à la branche `-STABLE`.

4. Relations entre développeurs

Si vous travaillez directement sur votre propre code ou sur du code dont il est bien établi que vous avez la responsabilité, il n'est probablement pas nécessaire de valider ce que vous allez faire avec d'autres développeurs avant de soumettre du code. Si vous trouvez un bogue dans un module qui est manifestement orphelin (il y en a malheureusement quelques uns), cela s'y applique aussi. Si, au contraire, vous vous apprêtez à modifier quelque chose qui est activement maintenu par quelqu'un d'autre (ce n'est qu'en surveillant la liste de diffusion des messages de modification CVS de FreeBSD (http://lists.FreeBSD.org/mailman/listinfo/cvs-all) que vous pourrez vous faire une idée de ce qu'il l'est et de ce qui ne l'est pas), envisagez alors de lui envoyer vos modifications, tout comme vous l'auriez fait quand vous n'étiez pas *committer*. Pour les logiciels portés, vous devriez contacter la personne listée comme `MAINTAINER` dans le `Makefile`. Pour le reste des archives, si vous n'êtes pas sûr de qui maintient effectivement tel ou tel module, il peut être utile de passer en revue le résultat de `cvs log` pour voir qui a soumis des modifications dans le passé. Si vous ne trouvez personne, ou si la personne en charge montre un désintérêt pour la partie en question, allez-y et faites vos modifications.

Si vous avez pour une raison ou une autre des doutes à propos d'une soumission que vous envisagez, faites-la d'abord examiner par `-hackers` avant de l'intégrer. Il vaut mieux que l'on vous fasse des remarques alors, qu'une fois qu'elle fera partie des archives CVS. S'il vous arrive de soumettre quelque chose qui soulève une controverse, envisagez éventuellement de faire marche arrière en attendant que la question soit réglée. N'oubliez pas – avec CVS, vous pourrez toujours remettre votre modification en service.

5. GNATS

Le Projet FreeBSD utilise **GNATS** pour enregistrer les rapports de bogues et les demandes de modification. Si vous effectuez une correction ou une modification décrite dans un PR - *Problem Report*, rapport d'anomalie - **GNATS**, veillez à utiliser `edit-pr` *`numéro_de_pr`* sur `freefall` pour le fermer. L'usage veut aussi que vous preniez le temps de fermer les rapports ayant trait à vos soumission, le cas échéant. Vous pouvez aussi utiliser vous-même send-pr(1) pour proposer les modifications dont vous pensez qu'il faut les probablement les faire, après une revue plus extensive par les autres participants.

Vous trouverez plus d'informations sur **GNATS** aux adresses suivantes :

- http://www.cs.utah.edu/csinfo/texinfo/gnats/gnats.html
- http://www.FreeBSD.org/support.html
- http://www.FreeBSD.org/send-pr.html
- send-pr(1)

6. Who’s Who

En dehors de Peter Wemm et John Polstra, les administrateurs des archives, il y a d’autres membres du Projet FreeBSD que vous rencontrerez probablement dans votre nouvelle fonction de *committer*. Rapidement, et en aucun cas exhaustivement, ce sont :

Satoshi Asami <asami@FreeBSD.org>

> Est le responsable du catalogue des logiciels portés, ce qui signifie qu’il a le pouvoir de décision en ce qui concerne toute modification aux logiciels portés et à leurs macros-instructions de compilation. Il est aussi responsable la gestion des gels du code entre deux versions.

Bruce Evans <bde@FreeBSD.org>

> Est l’*Obersturmbahnfuhrer* de la Police du Style. Quand vous soumettez quelque chose que vous auriez pu faire mieux, Bruce sera là pour vous le signaler. Remerciez-le qu’il y ait quelqu’un pour le faire.

David Greenman <dg@FreeBSD.org>

> Est notre architecte principal et superviseur du système de gestion de la mémoire virtuelle. Si vous envisagez une modification de ce système, voyez cela avec David. Si vous êtes pris dans une discussion âpre et insoluble avec un autre participant à propos d’une modification envisagée (ce qui, heureusement, ne se produit pas souvent), il peut aussi occasionnellement être nécessaire de demander alors à David de mettre sa casquette d’Architecte Principal et de prendre la décision finale.

Jordan K. Hubbard <jkh@FreeBSD.org>

> Est le responsable des versions. Il a la charge de définir les dates butées et de superviser le processus de mise en place de la nouvelle version. Pendant les gels du code, il a aussi le pouvoir de décision sur toutes les modifications sur la branche de code qui est en cours de finalisation. S’il y a quelque chose que vous voudriez voir reporter de `-CURRENT` dans `-STABLE` (quelqu’intérêt que cela puisse avoir à un moment donné), c’est aussi la personne à qui il faut en parler.

Mark Murray <markm@FreeBSD.org>

> Mark est le responsable des archives CVS internationales pour le code de cryptographie, sur `internat.FreeBSD.org` en Afrique du Sud.
>
> Mark supervise la plupart du code de cryptographie ; si vous vous y envisagez des mises à jour, parlez-en s’il vous plaît d’abord à Mark.

Steve Price <steve@FreeBSD.org>

> Steve est le responsable non officiel de `/usr/src/bin`. S’il y a quelque chose d’important que vous voudriez y voir, vous devriez probablement envisager d’abord cela avec Steve. Il est aussi administrateur des “*Problem Report*”, en coopération avec Poul-Henning Kamp <phk@FreeBSD.org>.

Brian Somers <brian@FreeBSD.org>

> Maintient officiellement `/usr/bin/ppp` et **LPD**.

Garrett Wollman <wollman@FreeBSD.org>

> Si vous avez besoin d’un conseil sur des points obscurs du code réseau ou n’êtes pas certain d’une modification que vous envisagez à ce sous-système, c’est avec Garrett qu’il faut en discuter.

7. Introduction rapide à SSH

1. Mettez à jour et installez le logiciel porté **ssh** dans `/usr/ports/security/ssh` (il faut une version 1.2.25 ou postérieure).
2. Veillez à exécuter ssh-agent(1) avant toute autre application. Les utilisateurs de **X**, par exemple, le font habituellement depuis leur fichier `.xsession` ou `.xinitrc`. Reportez-vous à ssh-agent(1) pour plus de détails.
3. Générez une paire de clés avec ssh-keygen(1). Ces clés seront créées dans le répertoire `$HOME/.ssh`.
4. Copiez votre clé publique (`$HOME/.ssh/identity.pub`) dans le fichier `authorized_keys` de votre répertoire utilisateur sur `freefall` (i.e. `$HOME/.ssh/authorized_keys`).

Vous devriez maintenant pouvoir utiliser ssh-add(1) pour vous authentifier à chaque début de session. Il vous demandera la phrase clé pour votre clé privée, et l’enregistrera via votre agent d’authentification (ssh-agent(1)) de façon à ce que vous n’ayez plus à la retaper à chaque fois.

Testez en faisant quelque chose du style : `ssh freefall.FreeBSD.org ls /usr`.

Pour plus d’informations, reportez-vous à `/usr/ports/security/ssh`, ssh(1), ssh-agent(1), scp(1), et ssh-keygen(1).

8. Régles à Suivre par les *Committers* FreeBSD

1. Respectez les autres *committers*.
2. Discutez de toute modification importante *avant* intégration.
3. Respectez les responsables de la maintenance s’il y en a de définis par la variable `MAINTAINER` du `Makefile` ou dans le fichier `MAINTAINER` au premier niveau de l’arborescence.
4. N’intervenez jamais directement sur les archives. Demandez au reponsable de ces archives de le faire.
5. Toute modification controversée doit, si le responsable de la maintenance ou l’Architecte Principal le demande, être annulée jusqu’à ce que la discussion soit terminée. Les modifications pour des questions de sécurité peuvent être effectuées par l’Officier de Sécurité, malgré les souhaits d’un responsable de la maintenance.
6. Les modifications doivent être faites dans `-current` avant d’être reportées dans `-stable` sauf autorisation expresse du responsable des versions ou si elles ne s’appliquent pas à `-current`. Toute modification non triviale ni urgente doit rester au moins trois jours dans `-current` pour être testée suffisamment avant d’être reportée. Le responsable des versions a les mêmes prérogatives sur la branche `-stable` que celles décrites, pour ce qui concerne l’Architecte Principal, par le règle #5.
7. Ne vous disputez pas publiquement avec les autres *committers* ; cela fait mauvais effet. Si vous êtes en “profond” désaccord sur un point, n’en discutez qu’en privé.
8. Respectez tous les gels du code et lisez régulièrement la liste de diffusion pour les *committers* pour savoir quand il y en a.
9. En cas de doute sur une procédure, renseignez-vous d’abord !
10. Testez vos modifications avant de les intégrer.

Comme indiqué, enfreindre l’un de ces règles peut entraîner une suspension provisoire, et, en cas de récidive, une suppression permanente des privilèges de *committers*. Trois membres ou plus de l’équipe de base, ou l’Architecte

Principal et un autre membre de l'équipe de base, peuvent, s'ils en sont d'accord, suspendre temporairement ces privilèges jusqu'à ce que l'ensemble de `-core` examine la question. En cas d'« urgence » (un *committer* endommageant les archives), une suspension provisoire peut aussi être décidée par l'un des administrateurs des archives ou tout autre membre de l'équipe de base qui se trouve être réveillé à ce moment-là. Seule la totalité de l'équipe de base peut suspendre pour une durée importante les droits d'un *committer*, ou les retirer définitivement, cette dernière mesure n'étant en général prise qu'après consultation avec les *committers*. Le but de cette règle n'est pas de faire de l'équipe de base une bande de dictateurs cruels qui puissent disposer des *committers* comme de cannettes vides, mais d'avoir une sorte de fusible pour le projet. Si quelqu'un est sévèrement incontrôlable, il est important de pouvoir réagir immédiatement, au lieu d'être paralysé par la discussion. Dans tous les cas, le *committers* dont les privilèges sont suspendus a le droit d'être "entendu", c'est à ce moment-là qu'il est décidé de la durée totale de la suspension. Il peut aussi demander un révision de la décision après 30 jours et tous les 30 jours ensuite (à moins que la durée totale de la suspension soit inférieure à 30 jours). Quelqu'un à qui les privilèges ont été définitivement retiré peut demander que son cas soit revu après 6 mois. La procédure de révision est *strictement informelle*, et, dans tous les cas, l'équipe de base se réserve le droit de prendre en compte ou d'ignorer les demandes de révision, si elle pense que sa décision initiale était la bonne.

Pour toutes les autres aspects du fonctionnement du projet, l'équipe de base est un sous-ensemble des *committers* et est soumise aux *même* règles. Ce n'est pas parce que quelqu'un appartient à l'équipe de base qu'il est dispensé de suivre les instructions que l'on vient de donner, les "pouvoirs spéciaux" de l'équipe de base ne sont effectifs que lorsqu'elle agit en tant que groupe, pas individuellement. Individuellement, nous sommes tous d'abord des *committers* et ensuite seulement membres de l'équipe de base.

8.1. Détails

1. Respectez les autres *committers*.

   Cela signifie que vous devez traiter les autres *committers* en tant que groupe de co-développeurs qu'ils sont en fait. Malgré nos tentatives occasionnelles pour prouver le contraire, on ne devient pas *committer* en étant stupide et rien n'est plus irritant que d'être traité comme tel par un de vos collaborateurs. Que nous apprécions toujours quelqu'un d'autre ou pas (chacun a ses jours sans), nous devons malgré tout toujours *traiter* les autres avec respect, sans quoi c'est toute l'organisation de l'équipe qui se désagrège rapidement.

   Etre capable de travailler ensemble à long terme est le plus grand atout du projet, bien plus important que n'importe quel série de modifications du code, et transformer les discussions à propos du code en disputes qui affectent notre capacité à travailler harmonieusement ensemble à long terme n'en vaut vraiment pas la peine, quelque justification que l'on puisse imaginer.

   Pour respecter cette règle, n'envoyez pas de courrier électronique quand vous êtes en colère et ne vous comportez en outre pas de façon à paraître inutilement agressif aux autres. Commencez par vous calmer et réfléchissez à la manière la plus efficace de convaincre l(es) autre(s) personne(s) de la justesse de votre point de vue. Ne partez pas sur les chapeaux de roues pour vous sentir simplement immédiatement mieux au prix d'une dispute à long terme. Non seulement c'est une mauvaise "gestion des ressources", mais les responsables du projet sanctionneront sévèrement les manifestations d'agressivité publiques et répétées, jusqu'à suspendre ou vous retirer définitivement vos privilèges de *committer*. Ce n'est pas une chose qu'ils aiment le moins du monde faire, mais l'unité est la priorité. Aucune dose de code ou de judicieux conseils ne s'y mesure.

2. Discutez de toute modification importante *avant* intégration.

   Ce n'est pas dans les archives CVS que les modifications doivent être intégrées pour validation ou discussion, cela doit se faire d'abord sur les listes de discussion et être intégré ensuite lorsqu'on est arrivé à quelque chose

qui approche du consensus. Cela ne signifie pas que vous deviez demander la permission avant de corriger chaque erreur évidente de syntaxe ou d'orthographe dans une page de manuel, mais simplement que vous devriez essayer de sentir quand vous envisagez une modification qui n'est pas aussi triviale et qui demande à être discutée au préalable. Les gens n'ont rien contre les modifications d'envergure si le résultat en est quelque chose de nettement meilleur que ce qu'ils avaient auparavant, mais ils n'aiment pas être *surpris* par ces modifications. La meilleure façon de vous assurer que vous allez dans le bon sens et de faire valider votre code par un ou plusieurs autres *committers*.

En cas de doute, demandez une validation !

3. Respectez les responsables de la maintenance, s'il y en a.

   De nombreuses parties de FreeBSD n'"appartiennent" à personne, c'est-à-dire qu'il n'y aura personne pour pousser de hauts cris si vous faites des modifications sur "leur" terrain, mais il vaut mieux s'en assurer d'abord. Une de nos convention est de mettre une ligne indiquant qui maintient dans le `Makefile` du paquetage ou de la sous-arborescence activement maintenue par une ou plusieurs personnes voyez http://www.FreeBSD.org/handbook/policies.html pour plus d'information à ce sujet. Quand il y a plusieurs personnes qui maintiennent une même section de code, les soumissions d'une de ces personnes sur ces sections doivent être revues par au moins une des autres personnes qui la maintiennent. Dans le cas où l'« attribution » n'est pas claire, vous pouvez aussi consulter les messages de CVS pour les fichiers concernés, pour voir si quelqu'un a travaillé dessus récemment ou travaille de façon prédominante sur ce domaine.

   Il y a d'autres parties de FreeBSD qui sont contrôlées par quelqu'un qui gère tout un domaine de l'évolution de FreeBSD, l'internationalisation ou le réseau par exemple. Reportez-vous à http://www.FreeBSD.org/handbook/staff-who.html (http://www.FreeBSD-fr.org/handbook/staff-who.html) pour avoir plus d'informations à ce sujet.

4. N'intervenez jamais directement sur les archives. Demandez à un responsable des archives de le faire.

   C'est assez clair - vous n'avez pas le droit de faire de modifications directement sur les archives, point. En cas de difficultés, adressez-vous à l'un des responsables des archives en envoyant un courrier électronique à <`cvs@FreeBSD.org`> et attendez qu'ils corrigent le problème et vous relancent. N'essayez pas de régler le problème vous-même !

   Si vous envisagez de supprimer un étiquette ou d'en mettre une nouvelle, ou bien d'importer du code sur nouvelle branche, il vous sera peut-être utile de demander d'abord un avis. Nombreux sont ceux qui se trompent en faisant cela les premières fois et cela aboutit à la modification de nombreux fichiers et irrite les utilisateurs de **CVSup/CTM** qui recoivent tout à coup de nombreuses mises à jour inutiles.

5. Toute modification controversée doit, si le responsable de la maintenance ou l'Architecte Principal le demande, être annulée jusqu'à ce que la discussion soit terminée. Les modifications pour des questions de sécurité peuvent être effectuées par l'Officier de Sécurité, malgré les souhaits d'un responsable de la maintenance.

   Ce peut être dur à avaler en cas de conflit (quand chaque partie est bien sûr convaincue qu'elle a raison) mais CVS permet d'éviter de prolonger la dispute, il est bien plus facile de revenir sur les modifications, d'attendre que tout le monde se calme et d'essayer de voir quelle est la meilleure solution. S'il s'avère que la modification était la bonne chose à faire, elle peut-être facilement remise en service. Dans le cas contraire, les utilisateurs n'auront pas eu à subir l'évolution erronée le temps que tout le monde ait débattu de sa pertinence. Il est très rare que l'on ait à revenir sur des modifications archivées, parce que la discussion met la plupart du temps en évidence les interventions controversés ou non justifiées avant même qu'elles n'aient été intégrées, mais dans les rares cas où cela se produit, il faut revenir en arrière sans discuter de façon à ce que l'on puisse immédiatement examiner s'il y avait erreur ou non.

6. Les modifications doivent être faites dans `-current` avant d’être reportées dans `-stable` sauf autorisation expresse du responsable des versions ou si elles ne s’appliquent pas à `-current`. Toute modification non triviale ni urgente doit rester au moins trois jours dans `-current` pour être testée suffisamment avant d’être reportée. Le responsable des versions a les mêmes prérogatives sur la branche `-stable` que celles décrites, pour ce qui concerne l’Architecte Principal, par le règle #5

   C’est un autre point « sans appel » parce que c’est l’ingénieur de version qui est en dernier lieu responsable (et encaisse les coups) si une modification s’avère mal fondée. Respectez s’il vous plaît cette règle et coopérez totalement avec le responsable des versions pour ce qui concerne la branche `-stable`. La gestion de la branche `-stable` peut parfois paraître excessivement conservatrice à un observateur occasionnel, mais rappelez vous que c’est le principe même de `-stable` et que `-current` suit d’autres règles. Il n’y a aucune raison d’avoir une branche `-current` si toutes les modifications vont immédiatement dans `-stable`, sans pouvoir d’abord être testées par les développeurs de `-current`, laissez donc passer un peu de temps avant de les reporter dans `-stable`, à moins que la modification ne soit critique, urgente, ou suffisamment triviale pour rendre tout test ultérieur superflu (correction d’orthographe dans les pages de manuel, de bogue flagrant ou de faute de frappe, etc.) En d’autres termes, faites preuve de bon sens.

7. Ne vous disputez pas publiquement avec les autres *committers* ; cela fait mauvais effet. Si vous êtes en “profond” désaccord sur un point, n’en discutez qu’en privé.

   Le projet a une image publique à conserver et cette image est très importante pour nous tous, en particulier si nous voulons continuer à attirer de nouveaux membres. Il y aura des situations où, malgré tous les efforts de chacun pour rester mesurés, certains perdront leur calme et laisserons leur colère s’exprimer, et le mieux que nous puissions faire est d’essayer d’en minimiser les effets jusqu’à ce que chacun se soit de nouveau calmé. Cela signifie que vous ne devez ni laisser exprimer votre colère en public, ni faire suivre de courriers privés sur des listes ou des alias publics. Ce que les gens se disent entre eux est souvent moins édulcoré que ce qu’ils disent en public, et ce type d’échange n’y a donc pas sa place - cela ne peut qu’envenimer une situation déjà regrettable. Si la personne qui vous adresse des reproches prend au moins la précaution de le faire en privé, ayez vous aussi la correction de le garder pour vous. Si vous estimez avoir été injustement traité par un autre développeur et que cela vous soucie, parlez-en à l’équipe de base plutôt qu’en public. Nous ferons de notre mieux pour jouer les médiateurs et ramener les choses au raisonnable. Quand la discussion a trait à une modifications de code et que les participants n’arrivent apparemment pas à se mettre d’accord, l’équipe de base peut désigner une troisième partie ayant l’accord mutuel pour résoudre le problème. Les autres personnes impliquées doivent alors accepter de se plier aux décisions de cette troisième partie.

8. Respectez tous les gels du code et lisez régulièrement la liste de diffusion pour les *committers* pour savoir quand il y en a.

   Soumettre des modifications pendant un gel du code est vraiment une grave erreur et l’on attend des *committers* qu’ils se tiennent au courant de ce qui se passe avant de se remanifester après une longue absence et soumettre 10 Mo de code accumulés pendant ce temps. Les gens qui se comportent régulièrement de cette façon verront leurs privilèges de *committers* suspendus jusqu’à leur retour du Joyeux Camp de Rééducation de FreeBSD que nous gérons au Gröenland.

9. En cas de doute sur une procédure, renseignez-vous d’abord !

   De nombreuses erreurs sont commises parce que quelqu’un est pressé et estime qu’il sait quelle est la meilleure façon de faire quelque chose. Il y a des bonnes chances que vous ne sachiez en fait pas comment faire ce que vous n’avez encore jamais fait et que vous ayez vraiment besoin de demander d’abord sans quoi vous allez vous mettre publiquement dans l’embarras. Il n’y a aucune honte à demander “Comment diable fait-on cela ?”, nous savons déjà que vous êtes quelqu’un d’intelligent, sans quoi vous ne seriez pas *committer*.

10. Testez vos modifications avant de les intégrer.

Cela peut paraître évident, mais si c'était vraiment le cas, nous ne verrions probablement pas autant de cas où les gens ne le font manifestement pas. Si vos modifications touchent le noyau, vérifiez que vous pouvez toujours compiler et `GENERIC` et `LINT`. Si vos modifications s'appliquent ailleurs, assurez-vous que vous pouvez toujours compiler l'ensemble du système - `make world`. Si vous faites vos modifications sur une branche donnée, veillez à tester vos modifications sur une machine qui utilise cette version du système. Si votre modifications risque de poser des problèmes sur une autre architecture matérielle, veillez à tester sur toutes les architectures supportées. Nous n'avons actuellement qu'x86 et Alpha, c'est donc assez facile à faire. Si vous avez besoin de tester sur l'AXP, votre compte sur `beast.FreeBSD.org` vous permet de compiler et tester des binaires/noyaux/etc. sur Alpha. Quand d'autres architectures seront ajoutées à la liste des plates-formes supportées par FreeBSD, des ressources partagées de test seront disponibles.

8.2. Autres suggestions

Quand vous intégrez des modifications de la documentation, utilisez un correcteur orthographique avant de soumettre. Pour toutes les documentations en SGML, vous devriez aussi vérifier que vos directives de formatage sont valides, avec un `make lint`.

Pour toutes les pages de manuel en ligne, servez-vous de `manck` (au catalogue des logiciels portés) sur la page pour vérifier que toutes les références croisées et noms de fichiers sont corrects et que les `MKLINK`s appropriés sont installés.

9. Questions Fréquemment Posées propres aux logiciels portés

1. Importer un nouveau logiciel

1.1. Comment faire pour importer un nouveau logiciel ?

Lisez s'il vous plaît d'abord la section sur la copie des archives.

Pour importer un nouveau logiciel, le plus facile est d'utiliser la procédure `easy-import` sur `freefall`. Elle vous posera quelques questions et importera directement le logiciel dans le répertoire que vous aurez indiqué. Elle a été écrite par Jörg Wunsch <`joerg@FreeBSD.org`>, envoyez-lui s'il vous plaît un courrier électronique si vous avez des questions à propos de `easy-import`.

Il y a une chose qu'elle ne fera pas à votre place : ajouter le logiciel au `Makefile` du répertoire de niveau supérieur (catégorie). Il faudra le faire vous-même.

1.2. Y'a-t-il d'autres choses qu'il faut que je sache quand j'importe un nouveau logiciel ?

Vérifiez votre portage, pour vous assurez qu'il compile et que le paquetage est correctement construit. Voici ce qu'il est recommandé de faire :

```
% make install
% make package
% make deinstall
% pkg_add le_paquetage_que_vous_venez_de_compiler
% make deinstall
```

```
% make reinstall
% make package
```

Le chapitre faire vous-même un portage (../handbook/porting.html) du Manuel de Référence vous donnera des instructions plus détaillées.

Utilisez portlint(1) pour vérifier la syntaxe du portage. Il n'est pas indispensable d'éliminer la totalité des messages d'avertissement, mais veillez à régler les problèmes les plus évidents.

Si le logiciel porté a été soumis par quelqu'un qui n'a jamais collaboré au projet auparavant, ajoutez le nom de cette personne à la section *Autres Collaborateurs* du Manuel de Référence.

Fermez le PR, si le portage résulte d'un PR. Pour fermer un PR, il suffit d'exécuter **`edit-pr`** ***`PR#`*** sur `freefall` et de modifier la valeur de la variable `state` de `open` en `closed`. On vous demandera d'entrer un commentaire, et c'est tout.

2. Copies des archives

2.1. Quand avons-nous besoin qu'une opération de copie soit faite sur les archives ?

Quand vous voulez importer un logiciel en rapport avec un autre logiciel déjà archivé dans un autre répertoire, envoyez s'il vous plaît un courrier électronique au responsable des logiciels portés pour lui demander son avis. *En rapport* désigne ici une version différente ou une version légèrement modifiée. En exemple, on peut citer `print/ghostscript*` (versions différentes) et `x11-wm/windowmaker*` (version Anglaise et version internationalisée).

Comme autre exemple, on peut citer le cas d'un logiciel porté déplacé d'un sous-répertoire à un autre, ou d'une modification du nom d'un répertoire parce que l'auteur a changé le nom de son logiciel, bien qu'il dérive d'un logiciel déjà importé.

2.2. Quand n'avons-nous *pas* besoin q'une opération de copie soit faite sur les archives ?

Quand il n'y a pas d'historique à conserver. Si un logiciel est importé dans une catégorie erronée et immédiatement déplacé, il suffit d'un simple `cvs remove` de l'ancien suivi d'un `cvs import` du nouveau.

2.3. Que faut-il que je fasse ?

Envoyez un courrier électronique au responsable des logiciels portés, qui fera la copie de l'ancien emplacement vers le nouveau. Vous en serez averti, et l'on attendra alors de vous que vous exécutiez les opérations suivantes :

1. `cvs remove` de l'ancien logiciel (si besoin est),
2. Correction du `Makefile` de niveau supérieur (catégorie),
3. Mise à jour de `CVSROOT/modules`
4. Si d'autres logiciels dépendent de celui qui vient d'être mis à jour, correction des lignes décrivant leurs dépendances dans leurs `Makefile`s,

5. Si le logiciel a changé de catégories, modification en conséquence de la ligne `CATEGORIES` du `Makefile` du logiciel.

3. Gel des logiciels portés

3.1. Qu'est-ce qu'un « gel des logiciels portés » ?

Avant livraison d'une nouvelle version, il est nécessaire de limiter les interventions sur les archives des logiciels portés pendant une courte période, le temps que les paquetages et la version elle-même soient compilés. Cela pour garantir la cohérence entre les différents composants de la version, c'est cela que l'on appelle le « gel des logiciels portés ».

3.2. Combien de temps dure ce gel ?

Habituellement deux à trois jours.

3.3. Qu'est-ce que cela signifie pour moi ?

Pendant le gel des logiciels portés, vous ne devez pas soumettre quoi que ce soit dans l'arborescence des logiciels portés, sauf autorisation explicite du responsable des logiciels. « Autorisation explicite » correspond ici à l'un des deux cas de figure suivants :

- Vous avez posé la question au responsable des logiciels, et il vous a répondu : « Allez-y, intégrez ».
- Le responsable des ports vous a envoyé un courrier électronique, soit directement, soit à la liste de diffusion, pour signaler un problème à corriger sur le logiciel.

Notez bien que vous n'êtes pas implicitement autorisé à corriger un logiciel pendant un gel simplement parce qu'il ne compile plus.

3.4. Comment suis-je averti du début du gel des logiciels ?

Le responsable des logiciels portés enverra des messages d'avertissement sur la liste de diffusion à propos du catalogue des logiciels portés de FreeBSD (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports) et la liste de diffusion pour les *committers* de FreeBSD pour annoncer la mise en oeuvre prochaine d'une nouvelle version, habituellement deux à trois semaines à l'avance. La date exacte ne sera définitivement fixée que quelques jours avant. Cela parce que le gel des logiciels doit être synchronisé avec la mise en oeuvre de la version elle-même, et que ce n'est qu'à ce moment-là que l'on sait exactement quand cette opération aura lieu.

Quand le gel commencera, il y aura bien sûr une nouvelle annonce sur la liste de diffusion pour les *committers* de FreeBSD.

3.5. Comment suis-je averti de la fin du gel des logiciels ?

Quelques heures après la mise en place de la nouvelle version, le responsable des logiciels enverra un courrier électronique à la liste de diffusion à propos du catalogue des logiciels portés de FreeBSD (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports) et à la liste de diffusion pour les *committers* de FreeBSD pour annoncer la fin du gel des logiciels. Remarquez que la finalisation de la version n'implique pas

automatiquement la fin du gel. Nous devons nous assurer qu’un problème de dernière minute ne demande pas de reconstruction immédiate de la version.

4. Questions diverses

4.1. Comment sais-je si un logiciel porté compile correctement ou non ?

Commencez par consulter http://bento.FreeBSD.org/~asami/errorlogs/. Vous y trouverez les messages d’erreurs des dernières compilations des logiciels portés sous `3-stable` et `4-current`.

Néanmoins, il ne suffit pas qu’un logiciel n’y apparaisse pas pour pouvoir dire qu’il compile correctement. (Une de ses dépendances, par exemple, peut ne pas avoir compilé.) Voici les répertoires de `bento`, n’hésitez pas à aller y voir :

```
/a/asami/portbuild/3/errors        messages d’erreur de la dernière compilation de 3-stable
                    /logs          tous les messages de la dernière compilation de 3-stable
                    /packages      messages d’erreur sur les paquetages de la dernière compil
                    /bak/errors    messages d’erreur de la dernière compilation intégrale de
                    /bak/logs      tous les messages de la dernière compilation de l’intégral
                    /bak/packages  messages d’erreur sur les paquetages de la dernière compil
                  /4/errors        messages d’erreur de la dernière compilation de 4-current
                    /logs          tous les messages de la dernière compilation de 4-current
                    /packages      messages d’erreur sur les paquetages de la dernière compil
                    /bak/errors    messages d’erreur de la dernière compilation intégrale de
                    /bak/logs      tous les messages de la dernière compilation de l’intégral
                    /bak/packages  messages d’erreur sur les paquetages de la dernière compil
```

Essentiellement, si le logiciel apparaît dans `packages`, ou dans `logs` mais pas dans `errors`, il compile correctement. (Les répertoires `errors` contiennent ce que vous voyez sur la page Web.)

4.2. J’ai importé un nouveau logiciel. Dois-je l’ajouter au fichier `INDEX` ?

Non. Le responsable des logiciels portés regénère l’`INDEX` et l’intègre régulièrement aux archives.

4.3. Y’a-t-il d’autres fichiers auxquels je ne dois pas toucher ?

Tous les fichiers immédiatement dans `ports/`, et tous les fichiers des sous-répertoires dont le nom commence par une majuscule (`Mk`, `Tools`, etc.). Le responsable des logiciels est particulièrement susceptible pour ce qui touche à `ports/Mk/bsd.port.mk`, n’y touchez donc pas à moins que vous ne vouliez affronter son courroux.