Norme de hiérarchie du système de fichiers — Version 2.0

Groupe pour la norme de hiérarchie du système de fichiers

éditée par Daniel Quinlan

ABSTRACT

Cette norme consiste en un ensemble d'exigences et de suggestions concernant la disposition des fichiers et des répertoires dans un système d'exploitation de type UNIX. Les suggestions servent à faciliter l'interopérabilité des applications, des outils d'administration système, des outils de développement et des scripts, ainsi qu'à avoir une documentation plus uniforme pour ces systèmes.

January 16, 19100

1. Introduction

1.1 État de la norme

Voici la version 2.0 de la norme de hiérarchie du système de fichiers (FHS 2.0).

Les commentaires sur cette norme sont les bienvenus de la part des personnes intéressées. Les suggestions de modifications devraient être faites sous la forme d'une proposition de changement du texte, accompagnée de commentaires explicatifs appropriés.

Les suggestions de cette norme sont sujettes à modification. L'utilisation des informations incluses dans ce document se fait à vos risques et périls.

1.2 Organisation de la norme

Cette norme est divisée entre les sections suivantes :

1. introduction ;
2. le système de fichiers : établissement de quelques principes clés ;
3. le répertoire racine ;
4. la hiérarchie /usr ;
5. la hiérarchie /var ;
6. annexe spécifique au système d'exploitation.

1.3 Conventions

Une police de taille fixe est utilisée pour l'affichage des noms de fichiers et de répertoires.

Les parties variables des noms de fichiers sont représentées par une description de leur contenu à l'intérieur des caractères chevrons "<" et ">", <ainsi>. Les adresses de courrier électronique sont aussi entourées de chevrons "<" et ">" mais sont indiquées dans la police habituelle.

Les parties optionnelles des noms de fichiers sont entourées des caractères crochet "[" et "]" et peuvent être combinées avec la convention "<" et ">". Par exemple, si on pouvait trouver un fichier existant avec ou sans extension, on pourrait le représenter par <nom de fichier>[.<extension>].

Les parties de chaînes variables des noms de répertoires et de fichiers sont indiquées par une étoile : "*".

1.4 Historique de la FHS

Le processus de développement d'une hiérarchie de système de fichiers standard a débuté en août 1993 dans un effort de restructuration de la structure de fichiers et de répertoires de Linux. La FSSTND, norme pour une hiérarchie du système de fichiers spécifique au système d'exploitation Linux, est sortie le 14 février 1994. Des versions successives sont sorties le 9 octobre 1994 et le 28 mars 1995.

Au début de 1995, avec l'aide de membres de la communauté de développement BSD, il a été décidé de développer une version de FSSTND plus complète pour englober non seulement Linux mais aussi les autres systèmes de type UNIX. En définitive, nous avons fait un effort concerté pour nous concentrer sur des problèmes généraux aux systèmes de type UNIX. En reconnaissance de cette ouverture, le nom de la norme a été modifié pour devenir Norme de hiérarchie du système de fichiers ou FHS en abrégé. (NDT : en anglais, FHS veut dire Filesystem Hierarchy Standard.)

Les noms des volontaires qui ont contribué activement à cette norme sont indiqués à la fin de ce document. Cette norme représente un consensus entre les points de vue de ceux-ci et d'autres contributeurs.

1.5 Étendue

Ce document spécifie une hiérarchie de système de fichiers standard pour les systèmes de fichiers FHS en indiquant l'emplacement des fichiers et répertoires, et le contenu de certains fichiers système.

Cette norme a été faite pour être utilisée par les intégrateurs de systèmes, les développeurs de paquets et les administrateurs système dans la construction et la maintenance de systèmes de fichiers se conformant à FHS. Elle est tout d'abord destinée à servir de référence et n'est pas un tutoriel sur la manière de gérer une hiérarchie de système de fichiers conforme.

La FHS est basée sur des travaux préliminaires sur FSSTND, une norme d'organisation du système de fichiers pour le système d'exploitation Linux. Elle est basée sur la FSSTND pour pallier à des problèmes d'interopérabilité non seulement dans la communauté Linux mais dans un horizon plus vaste comprenant les systèmes d'exploitation basés sur 4.4BSD. Elle incorpore les leçons concernant le support de plusieurs architectures et les demandes en matière de réseaux hétérogènes, leçons apprises dans le monde BSD ou ailleurs.

Bien que cette norme soit plus complète que les tentatives précédentes sur la normalisation de la hiérarchie de systèmes de fichiers, des mises à jour périodiques peuvent s'avérer nécessaires à mesure que les demandes changent par rapport à la technologie émergeante. Il est aussi possible que de meilleures solutions aux problèmes évoqués ici soient découvertes ou que nos solutions ne soient plus les meilleures possibles. Des brouillons supplémentaires pourront être apportés en plus des mises à jour périodiques de ce document. Cependant, l'un des buts suivis est la compatibilité ascendante entre une version de ce document et la suivante.

Les commentaires relatifs à cette norme sont les bienvenus. Tout commentaire ou suggestion de modification devraient être adressés à l'éditeur de la FHS (Daniel Quinlan <quinlan@pathname.com>), ou si vous préférez, à la liste de diffusion FHS. Les commentaires de nature typographique ou grammaticale doivent être adressés directement à l'éditeur de la FHS.

Nous vous demandons de contacter en premier l'éditeur de la FHS avant d'envoyer un courrier à la liste de diffusion afin d'éviter un nouveau débat sur des sujets anciens. Les messages mal conçus ne seront pas bien vus sur la liste de diffusion.

Des questions concernant l'interprétation des objets de ce document peuvent se poser de temps en temps. Si vous avez besoin de précisions, veuillez contacter l'éditeur de la FHS. Puisque cette norme représente le consensus de nombreux participants, il est important de s'assurer que toute interprétation représente aussi l'opinion collective. Pour cette raison, il peut ne pas être possible de fournir une réponse immédiate sauf si la demande a déjà fait l'objet d'une discussion.

1.6 Indications générales

Voici quelques-unes des suggestions qui ont été utilisées dans le développement de cette norme :

- résoudre des problèmes techniques tout en limitant les difficultés de la transition ;
- faire une spécification suffisamment stable ;
- obtenir l'approbation des distributeurs, des développeurs, et autres décideurs dans les groupes de développement adéquats et encourager leur participation ;
- fournir une norme attractive pour les implémenteurs des différents systèmes de type UNIX.

1.7 Audience visée

L'audience visée par cette norme comprend, mais n'est pas limitée aux groupes de personnes suivants :

- développeurs de systèmes ;
- intégrateurs et distributeurs de systèmes ;
- développeurs d'applications ;
- auteurs de documentations ;
- administrateurs système et autres personnes intéressées (à des fins d'information).

1.8 Conformité avec ce document

Cette section définit la signification des termes "conforme" et "compatible" en ce qui concerne cette norme, et de conformité et compatibilité "partielle".

Une "implémentation" fait ici référence à une distribution, un système installé, un programme, un paquet (ou toute partie similaire d'un logiciel ou de données), ou tout composant de ceux-ci.

Une implémentation est totalement conforme à cette norme si chaque exigence de cette norme est satisfaite. Chaque fichier ou répertoire faisant partie de l'implémentation doit être situé comme il est spécifié dans ce document. Si le contenu d'un fichier est décrit ici, le contenu véritable doit correspondre à la description. L'implémentation doit aussi tenter de trouver tout fichier ou répertoire (extérieur à elle-même) en premier lieu ou exclusivement à l'endroit spécifié dans cette norme.

Une implémentation est totalement compatible avec cette norme si chaque fichier ou répertoire qu'elle contient peut être trouvé en regardant à l'endroit spécifié ici et sera trouvé avec un contenu identique à ce qui est spécifié ici, même si ce n'est pas l'emplacement de base ou physique du fichier ou du répertoire en question. L'implémentation doit, quand elle essaie de trouver un fichier ou un répertoire n'en faisant pas partie, le faire à l'endroit spécifié dans cette norme, bien qu'elle puisse aussi tenter de le trouver à d'autres endroits (non standards).

Une implémentation est partiellement conforme ou compatible si elle est conforme à ou compatible avec une partie significative de ce document. La conformité ou compatibilité partielle n'est faite pour s'appliquer qu'aux distributions et non à des programmes séparés. L'expression "une partie significative" est effectivement subjective, et dans les cas limites, la personne concernée devrait contacter l'éditeur de la FHS. Nous avons anticipé le fait que des dérapages soient tolérés dans les cas limites.

Afin de se définir comme partiellement conforme à la FHS ou partiellement compatible avec la FHS, une implémentation doit fournir une liste de tous les endroits auxquels elle et le document FHS diffèrent, en plus d'une courte explication de la raison de cette différence. Cette liste sera fournie avec l'implémentation en question, et aussi mise à disposition de la liste de diffusion FHS ou de l'éditeur de la FHS.

Les termes "doit", "devrait", "contient", "est" et ainsi de suite doivent être lus comme des exigences pour la conformité ou la compatibilité.

Notez qu'une implémentation n'a pas besoin de contenir tous les fichiers et répertoires spécifiés dans cette norme pour être conforme ou compatible. Il est simplement nécessaire que les fichiers qu'elle contient soient placés correctement. Par exemple, si le système de fichiers minix n'est pas supporté par une distribution, les outils minix n'ont pas besoin d'être inclus, même s'ils sont mentionnés explicitement dans la section sur `/sbin`.

De plus, certaines parties de ce document sont optionnelles. Dans ce cas, ceci sera dit explicitement, ou indiqué à l'aide d'un ou plusieurs mots parmi "peut", "recommande" ou "suggère". Les objets indiqués comme optionnels n'ont pas de portée sur la conformité ou la compatibilité d'une implémentation ; ce sont des suggestions faites pour encourager la pratique courante, mais ils peuvent être situés n'importe où au gré de l'implémenteur.

2. Le système de fichiers

Le système de fichiers UNIX est caractérisé par :

- une structure hiérarchique ;
- le traitement uniforme des fichiers de données ;
- la protection des fichiers de données.

Cette norme suppose que le système d'exploitation sous-jacent à un système de fichiers conforme à la FHS supporte les mêmes possibilités de sécurité de base que l'on trouve dans la plupart des systèmes de fichiers UNIX. Notez que cette norme n'essaie pas d'être en accord au mieux possible avec une implémentation quelconque d'un système UNIX. Cependant, beaucoup d'aspects de cette norme sont basés sur des idées que l'on trouve dans UNIX et autres systèmes de type UNIX.

Ceci après une considération attentive d'autres facteurs, comprenant :

- des pratiques courantes et saines dans les systèmes de type UNIX ;
- l'implémentation d'autres structures de systèmes de fichiers ;
- les normes applicables.

On peut définir deux catégories orthogonales de fichiers : partageables contre non partageables, et variables contre statiques.

Les données partageables sont celles que l'on peut partager entre plusieurs machines différentes ; non partageables, celles spécifiques à une machine particulière. Par exemple, les répertoires personnels des utilisateurs sont des données partageables, mais les fichiers de verrou de périphériques (locks), ne le sont pas.

Les données statiques comprennent les binaires, les bibliothèques, la documentation, et tout ce qui ne change pas sans l'intervention de l'administrateur système ; les données variables représentent tout le reste qui change sans l'intervention de l'administrateur système.

Pour faciliter la sauvegarde, l'administration et le partage de fichiers sur des réseaux de systèmes hétérogènes, il est préférable d'établir une correspondance simple et aisément compréhensible entre les répertoires (surtout les répertoires considérés comment des points de montage potentiels) et le type de données qu'ils contiennent.

À travers ce document, et dans tout système de fichiers bien organisé, la compréhension de ce principe de base aidera à diriger la structure et lui apporter une cohérence supplémentaire.

La distinction entre données partageables et non partageables est nécessaire pour plusieurs raisons :

- dans un environnement en réseau (par exemple, plus d'un hôte par site), une bonne partie des données peuvent être partagées entre différentes machines pour économiser de la place et faciliter la tâche de maintenance ;
- dans un environnement en réseau, certains fichiers contiennent des informations spécifiques à une seule machine. Par conséquent ces systèmes de fichiers ne peuvent être partagés (sans prendre des mesures spéciales) ;

- historiquement, certaines implémentations des systèmes de fichiers de type UNIX ont mélangé des données partageables et non partageables dans la même hiérarchie, rendant difficile le partage de grandes parties du système de fichiers.

La distinction "partageable" permet de supporter, par exemple :

- une partition `/usr` (ou des composants de `/usr`) montés (en lecture seule) à travers le réseau (en utilisant NFS) ;
- une partition `/usr` (ou des composants de `/usr`) montés à partir d'un support en lecture seule. Un CD-ROM peut être considéré comme un système de fichiers en lecture seule partagé avec d'autres systèmes conformes à la FHS, en utilisant le système de courrier comme un "réseau".

La distinction "statique" contre "variable" affecte le système de fichiers de deux manières principales :

- puisque `/` contient à la fois des données statiques et variables, il doit être monté en lecture-écriture ;
- puisque le traditionnel `/usr` contient à la fois des données variables et statiques, et puisque nous voudrons le monter en lecture seule (voir ci-dessus), il est nécessaire de fournir une méthode pour monter `/usr` en lecture seule. Ceci est obtenu par la création d'une hiérarchie `/var` qui est montée en lecture-écriture (ou qui fait partie d'une autre partition en lecture-écriture, telle que `/`) , qui remplace bien des fonctions traditionnelles de la partition `/usr`.

Voici un tableau pour résumer le tout. Puisque ce graphique contient des exemples généralisés, il peut ne pas s'appliquer à chaque implémentation possible d'un système conforme à la FHS.

	partageable	non partageable
statique	/usr /opt	/etc /boot
variable	/var/mail /var/spool/news	/var/run /var/lock

3. Le répertoire racine

Cette section décrit la structure du répertoire racine (root). Le contenu du système de fichiers racine doit être adéquat pour démarrer, reconstituer, rétablir et/ou réparer le système :

- pour démarrer un système, il doit y avoir suffisamment de choses sur la partition racine pour monter d'autres systèmes de fichiers. Ceci comprend les utilitaires, la configuration, les informations du chargeur de démarrage, et d'autres données de démarrage importantes. `/usr`, `/opt` et `/var` sont faits pour pouvoir être situés sur d'autres systèmes de fichiers ;
- pour permettre le rétablissement et/ou la réparation d'un système, les utilitaires nécessaires au mainteneur expérimenté pour diagnostiquer et reconstruire un système endommagé doivent être présents sur le système de fichiers racine ;
- pour reconstituer un système, les utilitaires nécessaires à la reconstitution à partir des sauvegardes système (sur disque, bande, etc.) doivent être présents sur le système de fichiers racine.

Le principal argument utilisé pour contrer ces considérations, qui tendent à mettre beaucoup de choses sur le système de fichiers racine, est le but de garder la racine aussi petite que possible dans les limites du raisonnable. Pour plusieurs raisons, il est souhaitable de limiter la taille du système de fichiers racine :

- il est monté de temps en temps à partir d'un moyen de stockage très petit ;
- le système de fichiers racine contient beaucoup de fichiers de configuration spécifiques au système. Les exemples possibles comprennent un noyau spécifique au système, un nom d'hôte différent, etc. Ceci veut dire que le système de fichiers racine n'est pas toujours partageable entre des systèmes en réseau. Limiter sa taille sur des systèmes en réseau minimise l'espace perdu en fichiers non-partageables sur les serveurs. Cela permet aussi d'avoir des stations de travail avec des disques durs locaux plus petits ;
- bien que vous puissiez avoir un système de fichiers racine sur une grande partition, et pouvez le remplir à votre aise, il y aura des gens avec des partitions plus petites. Si vous avez plus de fichiers installés, vous pourrez trouver des incompatibilités avec d'autres systèmes qui utilisent des systèmes de fichiers racine sur des partitions plus petites. Si vous êtes développeur vous risquez de changer votre hypothèse en un problème pour un grand nombre d'utilisateurs ;
- les erreurs de disque qui corrompent les données sur le système de fichiers racine posent un problème plus important que les erreurs sur tout autre partition. Un système de fichiers racine petit est moins sujet à la corruption à la suite d'un plantage système ;

les logiciels ne doivent jamais créer ou demander des fichiers spéciaux ou des sous-répertoires dans le répertoire racine. D'autres emplacements dans la hiérarchie FHS fournissent plus de flexibilité qu'il n'en faut pour tout paquet.

Raison d'être

Il y a plusieurs raisons pour lesquelles l'introduction d'un nouveau répertoire dans le système de fichiers racine est interdit :

- ceci demande de la place sur une partition racine que l'administrateur système veut garder petite et simple pour des raisons de performance ou de sécurité ;
- ceci contredit toute logique que l'administrateur système a pu mettre en place pour distribuer des hiérarchies de fichiers standards sur des volumes montables.

/ — le répertoire racine

```
├── bin     binaires des commandes importantes
├── boot    fichiers statiques du chargeur de démarrage
├── dev     fichiers de périphériques
├── etc     configuration système spécifique à la machine
├── home    répertoires personnels des utilisateurs
├── lib     bibliothèques partagées importantes et modules du noyau
├── mnt     point de montage des partitions temporaires
├── opt     paquets de logiciels applicatifs supplémentaires
├── root    répertoire personnel de l'utilisateur root
├── sbin    binaires systèmes importants
├── tmp     fichiers temporaires
├── usr     hiérarchie secondaire
└── var     données variables
```

Chaque répertoire listé ci-dessus est spécifié en détail dans des sous-sections séparées ci-dessous. `/usr` et `/var` ont chacun une section complète dans ce document à cause de la complexité de ces répertoires.

L'image du noyau du système d'exploitation doit être situé soit dans `/`, soit dans `/boot`. Les informations supplémentaires sur l'emplacement du noyau se trouvent dans la section concernant `/boot` ci-dessous.

3.1 /bin : binaires de commandes utilisateurs importantes (pour tous les utilisateurs)

`/bin` contient des commandes à l'usage à la fois de l'administrateur système et des utilisateurs, mais qui sont obligatoires en mode utilisateur simple. Il peut aussi contenir des commandes utilisées indirectement par des scripts.

Il ne devrait pas y avoir de sous-répertoires à l'intérieur de `/bin`.

Les binaires des commandes qui ne sont pas suffisamment importantes pour rester dans `/bin` doivent être mises dans `/usr/bin`, à la place. Les objets qui ne sont utilisés que par des utilisateurs non root (`mail`, `chsh`, etc.) ne sont en général pas assez importants pour être placés dans la partition racine.

Fichiers obligatoires pour /bin :

- commandes générales :

  Les commandes suivantes ont été incluses parce qu'elles sont importantes. Certaines sont présentes à cause de leur emplacement traditionnel dans `/bin`.

  ```
  { cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, ed,
    false, kill, ln, login, ls, mkdir, mknod, more, mount, mv, ps,
    pwd, rm, rmdir, sed, setserial, sh, stty, su, sync, true, umount,
    uname }
  ```

  Si `/bin/sh` est le Bash, alors `/bin/sh` devrait être un lien symbolique ou en dur vers `/bin/bash` puisque Bash se comporte de manière différente quand il est appelé en tant que `sh` ou `bash`. `pdksh`, qui peut être `/bin/sh` sur certains disques d'installation, devrait être arrangé de la sorte, `/bin/sh` pointant vers `/bin/ksh`. L'utilisation d'un lien symbolique dans ces cas permet aux utilisateurs de voir aisément que `/bin/sh` n'est pas un vrai shell de Bourne.

Puisque l'emplacement standard de facto du shell C est `/bin/csh`, si et seulement si un shell C ou équivalent (comme `tcsh`) est disponible sur le système, il devrait être disponible par le nom `/bin/csh`. `/bin/csh` peut être un lien symbolique vers `/bin/tcsh` ou `/usr/bin/tcsh`.

Note : les commandes `[` *et* `test` *sont intégrées dans les remplacements du shell de Bourne (*`/bin/sh`*) les plus couramment utilisés. Ces deux commandes ne doivent pas être placées dans* `/bin` *; elles peuvent être placées dans* `/usr/bin`*. Elles doivent être incluses comme binaires séparés sur tout système UNIX ou de type UNIX essayant de se conformer à la norme POSIX.2.*

- commandes de remise en état :

  Ces commandes ont été ajoutées pour permettre la remise en état d'un système (en supposant que `/` soit intact).

  `{ tar, gzip, gunzip` (lien vers gzip), `zcat` (lien vers gzip) `}`

  Si les sauvegardes système sont faites en utilisant des programmes autres que `gzip` et `tar`, alors la partition racine devrait contenir le minimum requis par les composants de remise en état. Par exemple, beaucoup de systèmes devraient inclure `cpio` puisque c'est l'utilitaire de sauvegarde le plus couramment utilisé après `tar`. Inversement, si l'on est sûr de ne jamais faire aucune remise en état de la partition racine, on peut alors omettre ces binaires (par exemple, une partition racine en ROM, montant `/usr` en NFS). Si la remise en état d'un système est prévue à travers le réseau, alors `ftp` ou `tftp` (avec tout ce qui est nécessaire à l'établissement d'une connexion ftp) doit être disponible sur la partition racine.

  Les commandes de remise en état peuvent apparaître soit dans `/bin`, soit dans `/usr/bin` sur des systèmes différents.

- commandes réseau :

  Voici les seuls binaires nécessaires pour le réseau, autres que ceux de `/usr/bin` ou `/usr/local/bin`, et que à la fois root et les utilisateurs voudront ou auront besoin d'exécuter.

  `{ domainname, hostname, netstat, ping }`

3.2 /boot : fichiers statiques du chargeur de démarrage

Ce répertoire contient tout ce qu'il faut pour le processus de démarrage à part les fichiers de configuration et l'installeur de carte. Ainsi, `/boot` stocke les données utilisées avant que le noyau ne commence à exécuter des programmes en mode utilisateur. Ceci peut comprendre les secteurs de démarrage principaux sauvegardés, les fichiers de cartes des secteurs, et tout autre donnée qui n'est pas directement éditée à la main. Les programmes nécessaires à ce que le chargeur de démarrage soit capable de démarrer sur un fichier doivent être placés dans `/sbin`. Les fichiers de configuration pour les chargeurs de démarrage doivent être placés dans `/etc`.

Le noyau du système d'exploitation doit être situé soit dans `/`, soit dans `/boot`.

Note : sur certaines machines i386, il peut être nécessaire que `/boot` *soit situé sur une partition séparée* *située complètement en-dessous du cylindre 1024 du périphérique de démarrage à cause de contraintes matérielles.*

3.3 /dev : fichiers de périphériques

Le répertoire `/dev` est l'emplacement des fichiers spéciaux ou de périphériques.

S'il est possible que des périphériques dans `/dev` aient besoin d'être créés à la main, `/dev` contiendra une commande nommée `MAKEDEV`, qui pourra créer les périphériques au besoin. Il peut aussi disposer d'une commande `MAKEDEV.local` pour tout périphérique local.

Au besoin, `MAKEDEV` doit avoir de quoi créer n'importe quel périphérique qu'on pourrait trouver dans le système, et non pas simplement ceux qu'une implémentation particulière installe.

3.4 /etc : configuration système spécifique à la machine

`/etc` contient les fichiers et les répertoires de configuration spécifiques au système en cours.

Aucun binaire ne devrait être situé dans `/etc`.

/etc — configuration système spécifique à la machine

```
├── X11      configuration pour le système X Window
└── opt      configuration pour /opt
```

La section suivante sert surtout à illustrer la description du contenu de `/etc` avec un certain nombre d'exemples ; ce n'est surtout pas une liste exhaustive.

Fichiers obligatoires pour /etc :

- fichiers généraux :

  `{ adjtime, csh.login, disktab, fdprm, fstab, gettydefs, group, inittab, confissue, ld.so.conf, lilo.conf, motd, mtab, mtools, passwd, profile, securetty, shells, syslog.conf, ttytype }`

- fichiers de réseau :

  `{ exports, ftpusers, gateways, host.conf, hosts, hosts.allow, hosts.deny, hosts.equiv, hosts.lpd, inetd.conf, networks, printcap, protocols, resolv.conf, rpc, services }`

Notes :

La mise en place des scripts de commandes invoqués au démarrage peut ressembler aux modèles System V ou BSD. Des spécifications supplémentaires dans ce domaine pourront être ajoutées à une version future de la norme.

Les systèmes qui utilisent la suite shadow password auront des fichiers de configuration supplémentaires *dans* `/etc` *(*`/etc/shadow` *et autres) et des programmes dans* `/usr/sbin` *(*`useradd`*,* `usermod`*, et autres).*

3.4.1 /etc/X11 : configuration du système X Window

/etc/X11 est l'emplacement recommandé pour toute configuration X11 spécifique à la machine. Ce répertoire est nécessaire pour permettre un contrôle local si /usr est monté en lecture seule. Les fichiers qui devraient être dans ce répertoire comprennent Xconfig (et/ou XF86Config) et Xmodmap.

Les sous-répertoires de /etc/X11 peuvent inclure ceux pour xdm et pour tout autre programme (certains gestionnaires de fenêtres, par exemple) qui en a besoin. Nous recommandons que les gestionnaires de fenêtres qui n'ont qu'un fichier de configuration par défaut .*wmrc le nomment system.*wmrc (sauf s'il existe un nom de rechange largement reconnu) et n'utilisent pas de sous-répertoire. Tout sous-répertoire de gestionnaire de fenêtres devrait être nommé du même nom que le binaire réel du gestionnaire de fenêtres.

/etc/X11/xdm contient les fichiers de configuration de xdm. Ce sont la plupart des fichiers que l'on trouve habituellement dans /usr/lib/X11/xdm. Certaines données variables et locales pour xdm sont stockées dans /var/state/xdm.

3.4.2 /etc/opt : fichiers de configuration pour /opt

Les fichiers de configuration spécifiques à la machine pour les paquets des logiciels applicatifs supplémentaires seront installés dans le répertoire /etc/opt/<paquet>, où <paquet> est le nom du sous-répertoire de /opt où sont stockées les données statiques de ce paquet. Aucune structure n'est imposée sur la disposition interne de /etc/opt/<paquet>.

Si un fichier de configuration doit résider dans un endroit différent pour que le paquet ou le système fonctionne correctement, on peut le placer dans un endroit différent de /etc/opt/<paquet>.

Raison d'être :

Voir la raison d'être pour /opt.

3.5 /home : répertoires personnels des utilisateurs (optionnel)

/home est un concept très standard, mais c'est clairement un système de fichiers spécifique au site. Sa mise en place sera différente d'une machine à l'autre. Cette section ne décrit qu'un ordonnancement suggéré des répertoires personnels des utilisateurs ; néanmoins nous recommandons que toutes les distributions conformes à la FHS l'utilisent comme emplacement par défaut des répertoires utilisateurs.

Sur les petits systèmes, chaque répertoire utilisateur est en général l'un des nombreux sous-répertoires de /home comme /home/dupont, /home/torvalds, /home/admin, etc.

Sur des grands systèmes (surtout quand les répertoires /home sont partagés entre beaucoup d'hôtes par NFS) il est utile de subdiviser les répertoires personnels des utilisateurs. Le partage peut se faire en utilisant des sous-répertoires comme /home/equipe, /home/invites, /home/eleves, etc.

D'autres personnes préfèrent placer les comptes utilisateurs à divers autres endroits. Par conséquent, aucun programme ne devrait se fier à cet emplacement. Si vous voulez trouver le répertoire personnel d'un utilisateur, vous devriez utiliser la fonction de bibliothèque getpwent(3) plutôt que de compter sur /etc/passwd parce que les informations sur les utilisateurs peuvent être stockées à distance en utilisant des systèmes tels que NIS.

3.6 /lib : bibliothèques partagées importantes et modules du noyau

Le répertoire /lib contient les images des bibliothèques partagées nécessaires au démarrage du système et au lancement des commandes du système de fichiers racine.

/lib — bibliothèques partagées importantes et modules du noyau

└─ modules modules chargeables du noyau

Ceci comprend /lib/libc.so.*, /lib/libm.so.*, l'éditeur de liens dynamiques partagés /lib/ld.so, et d'autres bibliothèques partagées nécessaires aux binaires de /bin et /sbin.

Les bibliothèques partagées nécessaires uniquement aux binaires de /usr (comme n'importe quel binaire X Window) n'appartiennent pas à /lib. Seules les bibliothèques partagées nécessaires au fonctionnement des binaires de /bin et /sbin doivent se trouver ici. La bibliothèque libm.so.* peut aussi se trouver dans /usr/lib si elle n'est pas nécessaire dans /bin ou /sbin.

Pour des raisons de compatibilité, /lib/cpp doit exister et se référer au pré-processeur C installé sur le système. L'emplacement traditionnel de ce binaire est /usr/lib/gcc-lib/<cible>/<version>/cpp. /lib/cpp peut pointer soit vers ce binaire, soit vers toute référence à ce binaire qui existe dans le système de fichiers. (Par exemple, on utilise souvent /usr/bin/cpp.)

La spécification pour /lib/modules est en cours d'élaboration.

3.7 /mnt : point de montage pour les systèmes de fichiers montés temporairement

Ce répertoire est fourni pour que l'administrateur système puisse monter de manière temporaire et au besoin des systèmes de fichiers. Le contenu de ce répertoire est une affaire locale et ne devrait pas affecter la manière dont n'importe quel programme est lancé.

Nous sommes opposés à l'utilisation de ce répertoire par les programmes d'installation, et nous suggérons qu'un répertoire temporaire convenable non utilisé par le système soit utilisé à la place.

3.8 /opt : paquets de logiciels applicatifs supplémentaires

/opt — paquets de logiciels d'applications supplémentaires

└─ <paquet> objets statiques du paquet

/opt est réservé à l'installation de paquets de logiciels applicatifs supplémentaires.

Un paquet qui s'installe dans /opt devra mettre ses fichiers statiques dans une arborescence /opt/<paquet> séparée, où <paquet> est un nom décrivant le paquet logiciel.

Les programmes devant être lancés par les utilisateurs seront situés dans le répertoire `/opt/<paquet>/bin`. Si le paquet comprend des pages de manuel UNIX, elle seront situées dans `/opt/<paquet>/man` et la même structure que `/usr/share/man` sera utilisée.

Les répertoires `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib` et `/opt/man` sont réservés à l'usage local de l'administrateur système. Les paquets peuvent fournir des fichiers de "lancement" (front-end) faits pour qu'un administrateur système local les place (en faisant un lien ou en les copiant) dans ces répertoires réservés, mais ils devront fonctionner normalement en l'absence de ces répertoires réservés.

Les fichiers variables d'un paquet (qui changent pendant l'utilisation normale) devraient être installés dans `/var/opt`. Voyez la section sur `/var/opt` pour plus d'informations.

Les fichiers de configuration spécifiques à la machine devraient être installés dans `/etc/opt`. Voyez la section sur `/etc/opt` pour plus d'informations.

Aucun autre fichier de paquet ne devrait exister en dehors des hiérarchies `/opt`, `/var/opt` et `/etc/opt` sauf pour les fichiers de paquet qui doivent résider dans des endroits spécifiques à l'intérieur de l'arborescence du système de fichiers afin de fonctionner correctement. Par exemple, les fichiers de verrou des périphériques doivent être placés dans `/var/lock` et les périphériques dans `/dev`.

Raison d'être

L'utilisation de `/opt` pour les logiciels supplémentaires est une pratique bien établie dans la communauté UNIX. L'interface Binaire d'Applications (ABI) System V [AT&T 1990], basée sur la Définition d'Interface System V (troisième édition) fournit une structure `/opt` très similaire à celle décrite ici.

La Norme de Compatibilité Binaire Intel version 2 (iBCS2) fournit aussi une structure similaire pour `/opt`.

En général, toutes les données nécessaires au support d'un paquet sur un système doivent être présentes dans `/opt/<paquet>`, y compris les fichiers destinés à être copiés dans `/etc/opt/<paquet>` et `/var/opt/<paquet>` ainsi que dans les répertoires réservés de `/opt`.

3.9 /root : répertoire personnel de l'utilisateur root (optionnel)

`/` est traditionnellement le répertoire personnel du compte root sur les systèmes UNIX. `/root` est utilisé sur de nombreux systèmes Linux et sur certains systèmes UNIX (afin de réduire l'encombrement du répertoire `/`). Le répertoire personnel du compte root peut être déterminé par une préférence au niveau du développeur ou locale. Les possibilités évidentes comprennent `/`, `/root` et `/home/root`.

Si le répertoire personnel du compte root n'est pas stocké sur la partition racine il sera nécessaire de s'assurer qu'il prendra la valeur `/` par défaut si on ne peut le trouver.

Note : nous nous opposons à l'utilisation du compte root pour des choses simples comme le courrier électronique ou les forums de discussion, et recommandons qu'il ne soit utilisé qu'au titre de l'administration système. Pour cette raison, nous recommandons que les sous-répertoires tels que `Mail` *et* `News` *n'apparaissent pas dans le répertoire personnel du compte root, et que le courrier à destination des rôles administratifs comme root, postmaster ou webmaster soient redirigés vers un utilisateur approprié.*

3.10 /sbin : binaires systèmes (qui se trouvaient autrefois dans /etc)

Les utilitaires utilisés pour l'administration système (et autres commandes faites uniquement pour le super-utilisateur) sont stockés dans `/sbin`, `/usr/sbin` et `/usr/local/sbin`. `/sbin` contient typiquement des binaires importants au démarrage du système en plus des binaires de `/bin`. Tout ce qui est exécuté après qu'il soit sûr que `/usr` soit monté (quand il n'y a pas de problèmes) devrait aller dans `/usr/sbin`. Les binaires d'administration système spécifiques au système devraient être installés dans `/usr/local/sbin`.

Décider de ce qui va dans les répertoires `"sbin"` est simple : si un utilisateur normal (pas un administrateur système) doit le lancer directement, il devrait alors aller dans l'un des répertoires `"bin"`. Les utilisateurs ordinaires ne devraient mettre aucun des répertoires `sbin` dans leur chemin d'accès (path).

Note : par exemple, les fichiers tels que `chfn` *que les utilisateurs n'utilisent que de temps en temps devraient quand même être placés dans* `/usr/bin`. `ping`*, bien qu'il ne soit absolument nécessaire que pour root (remise en état et diagnostic réseau) est souvent utilisé par les utilisateurs et pour cette raison devrait exister dans* `/bin`.

Nous recommandons que les utilisateurs aient les permissions de lecture et d'exécution pour tout ce qui se trouve dans `/sbin` mis à part, peut-être, certains programmes setuid et setgid. La division entre `/bin` et `/sbin` n'a pas été créée pour des raisons de sécurité ou pour empêcher les utilisateurs de voir le système d'exploitation, mais pour fournir une bonne séparation entre les binaires que tout le monde utilise et ceux qui sont tout d'abord utilisés pour des tâches d'administration. Il n'y a pas d'avantage spécifique pour la sécurité à rendre `/sbin` inaccessible aux utilisateurs.

Fichiers obligatoires pour /sbin :

- commandes générales :

  `{ clock, getty, init, update, mkswap, swapon, swapoff, telinit }`

- commandes d'extinction :

  `{ fastboot, fasthalt, halt, reboot, shutdown }`

  (ou toute combinaison des fichiers ci-dessus, pourvu que `shutdown` soit inclus.)

- commandes de gestion du système de fichiers :

  `{ fdisk, fsck, fsck.*, mkfs, mkfs.* }`

  `*` = un ou plus parmi `ext, ext2, minix, msdos, xia` et peut-être d'autres

- commandes réseau :

  `{ ifconfig, route }`

3.11 /tmp : fichiers temporaires

Le répertoire `/tmp` devra êre mis à la disposition des programmes qui ont besoin de créer des fichiers temporaires.

Bien que les données stockées dans `/tmp` puissent être effacées d'une manière spécifique à chaque site, il est recommandé que les fichiers et répertoires situés dans `/tmp` soient effacés à chaque démarrage du système.

Les programmes ne devront pas supposer que tout fichier ou répertoire de `/tmp` est préservé entre deux lancements de ces programmes.

Raison d'être :

La norme IEEE P1003.2 (POSIX, partie 2) donne des obligations similaires à la section ci-dessus.

FHS a ajouté la recommandation du nettoyage de `/tmp` au démarrage sur la base de précédents historiques et de pratique commune, mais n'en a pas fait une obligation parce que l'administration système n'entre pas dans l'objectif de cette norme.

4. La hiérarchie /usr

`/usr` est la deuxième grande partie du système de fichiers. `/usr` contient des données partageables, en lecture seule. Ceci veut dire qu'on devrait pouvoir partager `/usr` entre plusieurs machines différentes se conformant à la FHS et on ne devrait pas y écrire. Toute information spécifique à la machine ou qui varie avec le temps est stockée autre part.

Aucun grand paquet logiciel ne devrait utiliser un sous-répertoire direct sous la hiérarchie `/usr`. Exception est faite du système X Window à cause d'un énorme précédent et d'une pratique largement suivie. Cette section de la norme spécifie l'emplacement de la plupart de ces paquets.

/usr — hiérarchie secondaire

├── `X11R6`	système X Window, version 11 release 6
├── `X386`	système X Window, version 11 release 5 sur des plates-formes x86
├── `bin`	la plupart des commandes utilisateurs
├── `games`	binaires de jeux et d'apprentissage
├── `include`	fichiers d'en-têtes inclus par les programmes C
├── `lib`	bibliothèques
├── `local`	hiérarchie locale (vide après l'installation principale)
├── `sbin`	binaires systèmes non importants
├── `share`	données indépendantes de l'architecture
└── `src`	code source

Les liens symboliques vers les répertoires suivants peuvent exister. Cette possibilité est basée sur le besoin de préserver la compatibilité avec les vieux systèmes jusqu'à ce qu'on soit sûr que toutes les implémentations utilisent la hiérarchie `/var`.

```
/usr/spool -> /var/spool
/usr/tmp -> /var/tmp
/usr/spool/locks -> /var/lock
```

Une fois qu'un système n'a plus besoin de l'un des liens symboliques ci-dessus, le lien peut être enlevé si désiré.

4.1 /usr/X11R6 : système X Window, version 11 release 6

Cette hiérarchie est réservée au système X Window, version 11 release 6, et aux fichiers apparentés.

Pour simplifier les choses et rendre XFree86 plus compatible avec le système X Window sur les autres systèmes, les liens symboliques suivants doivent être présents :

```
/usr/bin/X11 -> /usr/X11R6/bin
/usr/lib/X11 -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

En général, les logiciels ne devraient pas être installés ou gérés par l'intermédiaire de ces liens symboliques. Ils ne sont destinés qu'à une utilisation par les utilisateurs. La difficulté est liée à la version de sortie du système X Window — dans les périodes de transition, il est impossible de savoir quelle version de X11 est en cours d'utilisation.

Les données spécifiques à la machine dans `/usr/X11R6/lib/X11` devraient être considérées comme des fichiers de démonstration. Les applications ayant besoin d'informations sur la machine courante (grâce à des fichiers comme `Xconfig`, `XF86Config` ou `system.twmrc`) doivent se référer à un fichier de configuration dans `/etc/X11`, qui peut être un lien vers un fichier de `/usr/X11R6/lib`.

4.2 /usr/X386 : système X Window, version 11 release 5, sur les plates-formes x86

Cette hiérarchie est en général identique à `/usr/X11R6` ; les liens symboliques de `/usr` pour X11 devraient pointer vers la version du système X Window désirée.

4.3 /usr/bin : la plupart des commandes utilisateurs

Voici le répertoire principal des commandes exécutables sur le système.

/usr/bin — binaires non nécessaires en mode utilisateur unique

```
├─ mh        commandes pour le système de gestion de courrier MH
└─ X11       lien symbolique vers /usr/X11R6/bin
```

Les interpréteurs de scripts shell (qu'on lance avec `#!<chemin>` sur la première ligne d'un script shell) ne pouvant pas compter sur un chemin, il est avantageux de normaliser leur emplacement. Les interpréteurs du shell de Bourne et du shell C sont déjà fixés dans `/bin`, mais on trouve souvent Perl, Python et Tcl dans maints endroits différents. `/usr/bin/perl`, `/usr/bin/python` et `/usr/bin/tcl` devraient faire référence respectivement aux interpréteurs de shell `perl`, `python` et `tcl`. Il peut y avoir des liens symboliques vers l'emplacement véritable des interpréteurs shell.

4.4 /usr/include : répertoire pour les fichiers d'en-têtes standards.

Voici l'endroit où tous les fichiers d'en-têtes à usage général pour les langages de programmation C et C++ devraient être placés.

/usr/include — fichiers d'en-têtes

```
├── X11        lien symbolique vers /usr/X11R6/include/X11
├── bsd        fichiers d'en-têtes de compatibilité BSD (si nécessaire)
└── g++        fichiers d'en-têtes pour le GNU C++
```

4.5 /usr/lib : bibliothèques pour la programmation et les paquets

`/usr/lib` contient des fichiers objets, des bibliothèques et des binaires internes qui ne sont pas destinés à être exécutés directement par les utilisateurs ou les scripts shell.

Les applications peuvent utiliser un sous-répertoire unique sous `/usr/lib`. Si une application utilise un sous-répertoire, toutes les données dépendantes de l'architecture utilisées exclusivement par cette application devraient être placées dans ce sous-répertoire. Par exemple, le sous-répertoire `perl5` pour les modules et les bibliothèques de Perl 5.

Les fichiers et répertoires divers, statiques, indépendants de l'architecure et spécifiques à une application devraient aller dans `/usr/share`.

Certaines commandes exécutables telles que `makewhatis` et `sendmail` ont aussi été placées de manière traditionnelle dans `/usr/lib`. `makewhatis` est un binaire interne et devrait être placé dans un répertoire de binaires ; les utilisateurs n'utilisent que `catman`. Les nouveaux binaires `sendmail` sont maintenant placés par défaut dans `/usr/sbin` ; un lien symbolique devrait rester depuis `/usr/lib`. En plus, les systèmes qui utilisent Smail devraient placer Smail dans `/usr/sbin/smail` et `/usr/sbin/sendmail` devrait être un lien symbolique vers celui-ci.

Un lien symbolique `/usr/lib/X11` pointant vers le répertoire `lib/X11` de la distribution X par défaut est nécessaire si X est installé.

Note : aucune donnée spécifique à la machine pour le système X Window ne devrait être stocké dans `/usr/lib/X11`. *Les fichiers de configuration spécifiques à la machine tels que* `Xconfig` *ou* `XF86Config` *devraient se trouver dans* `/etc/X11`. *Ceci devrait comprendre les données de configuration tels que* `system.twmrc` *même si l'on n'en fait qu'un lien symbolique vers un fichier de configuration plus global (probablement dans* `/usr/X11R6/lib/X11`*).*

4.6 /usr/local : hiérarchie locale

La hiérarchie `/usr/local` est destinée à l'usage de l'administrateur système quand il installe des logiciels localement. Il doit être mis à l'abri de tout écrasement lors de la mise à jour du logiciel système. On peut l'utiliser pour des programmes et des données qu'on peut partager parmi un groupe de machines, mais qu'on ne trouve pas dans `/usr`.

/usr/local — hiérarchie locale

```
├── bin        binaires locaux
├── games      binaires de jeux locaux
├── include    fichiers d'en-têtes C locaux
├── lib        bibliothèques locales
├── sbin       binaires système locaux
├── share      hiérarchie indépendante de la machine locale
└── src        code source local
```

Ce répertoire devrait toujours être vide après la première installation d'un système se conformant à la FHS. Aucune exception à cette règle ne devrait être faite à part les morceaux de répertoires listés.

Un logiciel installé localement devrait être placé dans `/usr/local` plutôt que dans `/usr` sauf si on l'installe pour remplacer ou mettre à jour un logiciel de `/usr`.

Notez que les logiciels placés dans `/` ou `/usr` peuvent être écrasés par les mises à jour systèmes (bien que nous recommandons que les distributions n'écrasent pas les données de `/etc` dans ces circonstances). Pour cette raison, les logiciels locaux ne devraient pas se trouver en dehors de `/usr/local` sans une bonne raison.

4.7 /usr/sbin : binaires systèmes normaux non importants

Ce répertoire contient tous les binaires non importants utilisés exclusivement par l'administrateur système. Les programmes d'administration système nécessaires à la réparation du système, sa remise en route, le montage de `/usr` ou d'autres fonctions importantes devraient plutôt être placés dans `/sbin`.

Typiquement, `/usr/sbin` contient les démons réseau, tous les outils d'administration non importants et les binaires pour les programmes serveurs non critiques.

Ces programmes serveurs sont utilisés pendant le passage à l'état System V connu sous le nom de "run level 2" (niveau d'exécution, état multi-utilisateurs) et "run level 3" (niveau d'exécution, état en réseau) ou dans l'état BSD connu sous le nom de "mode multi-utilisateurs". À ce point, le système met certains services à la disposition des utilisateurs (par exemple, les impressions) et à d'autres machines (par exemple, l'exportation NFS).

Les programmes d'administration système installés en local devraient se trouver dans `/usr/local/sbin`.

4.8 /usr/share : données indépendantes de l'architecture

/usr/share — données indépendantes de l'architecture

```
├── dict        listes de mots
├── doc         documentations diverses
├── games       fichiers de données statiques pour /usr/games
├── info        répertoire principal du système Info de GNU
├── locale      informations pour Locale
├── man         pages de manuel en ligne
├── nls         support pour la langue natale
├── misc        données diverses indépendantes de la machine
├── terminfo    répertoires pour la base de données terminfo
├── tmac        macros troff non distribuées avec groff
└── zoneinfo    information et configuration pour le fuseau horaire
```

La hiérarchie `/usr/share` contient tous les fichiers de données indépendantes de l'architecture en lecture seule. La plupart de ces données se trouvaient à l'origine dans `/usr` (`man`, `doc`) ou `/usr/lib` (`dict`, `terminfo`, `zoneinfo`). Cette hiérarchie est destinée à être partagée entre toutes les plates-formes et architectures d'un système d'exploitation donné ; ainsi, par exemple, un site avec des plates-formes i386, Alpha et PPC peut maintenir un seul répertoire `/usr/share` monté de manière centrale. Notez, cependant, que `/usr/share` n'est en général pas fait pour être partagé par des systèmes d'exploitation différents ou par différentes versions du même système d'exploitation.

Tout programme ou paquet qui contient ou nécessite des données ne nécessitant pas de modification devrait stocker ces données dans `/usr/share` (ou `/usr/local/share` en cas d'installation locale). Il est recommandé d'utiliser un sous-répertoire de `/usr/share` à cet effet.

Notez que Linux utilise pour le moment des fichiers de bases de données au format DBM. Bien que ceux-ci ne soient pas indépendants de l'architecture, ils sont autorisés dans `/usr/share` en anticipation d'un passage au format DB 2.0 indépendant de l'architecture. (NdT : il semble que la plupart des distributions aient maintenant -- janvier 2000 -- inclus le format DB 2.0.)

Les données de jeux stockées dans `/usr/share/games` devraient être des données purement statiques. Tout fichier modifiable, comme les fichiers de scores, les enregistrements de parties et ainsi de suite, devraient se trouver dans `/var/games`.

Il est recommandé que les répertoire spécifiques à une application, indépendants de l'architecture soient placés ici. De tels répertoires comprennent `groff`, `perl`, `ghostscript`, `texmf` et `kbd` (Linux) ou `syscons` (BSD). On peut, cependant, les placer dans `/usr/lib` pour des raisons de compatibilité ascendante, à la discrétion du distributeur. De même, on peut utiliser une hiérarchie `/usr/lib/games` en plus de la hiérarchie `/usr/share/games` si le distributeur désire placer quelques données de jeux à cet endroit.

4.8.1 /usr/share/dict : listes de mots

Fichiers recommandés pour /usr/share/dict :

`{ words }`

Traditionnellement, ce répertoire ne contient que le fichier anglais `words`, utilisé par `look(1)` et divers programmes de correction orthographique. `words` peut utiliser l'orthographe américaine ou britannique. Les sites qui veulent les deux peuvent faire un lien `words` vers `/usr/share/dict/american-english` ou `/usr/share/dict/british-english`.

On peut ajouter des listes de mots pour d'autres langues en utilisant le nom anglais de cette langue, par exemple `/usr/share/dict/french`, `/usr/share/dict/danish`, etc. Ils devraient, si possible, utiliser un jeu de caractères ISO 8859 approprié à la langue en question ; si possible en utilisant le jeu de caractères Latin1 (ISO 8859-1) -- ce n'est souvent pas possible.

D'autres listes de mots, comme le "dictionnaire" `web2` devraient y être inclus, s'ils existent.

Raison d'être :

La raison pour laquelle seules les listes de mots sont situées ici est que ce sont les seuls fichiers communs à tous les correcteurs d'orthographe.

4.8.2 /usr/share/man : pages de manuel

Cette section parcourt en détail l'organisation des pages de manuel pour le système entier, en incluant `/usr/share/man`. Reportez-vous aussi à la section sur `/var/cache/man`.

Les pages de manuel sont stockées dans `<manrep>/<locale>/man<section>/<arch>`. `<manrep>`, `<locale>`, `<section>` et `<arch>` sont expliqués ci-dessous.

<manrep>/<locale> — hiérarchie de pages de manuel

```
├── man1    programmes utilisateurs
├── man2    appels système
├── man3    appels de bibliothèque
├── man4    fichiers spéciaux
├── man5    formats de fichiers
├── man6    jeux
├── man7    divers
└── man8    administration système
```

Le `<manrep>` principal du système est `/usr/share/man`. `/usr/share/man` contient des informations du manuel pour les commandes et les données des systèmes de fichiers `/` et `/usr`. Évidemment, il n'y a pas de pages de manuel dans `/` parce qu'elles ne sont pas utiles au démarrage, ni dans les situations d'urgence.

La `<section>` décrit la section du manuel.

Il faut s'assurer de faire de la place dans la structure de `/usr/share/man` pour supporter les pages de manuel écrites en des langues différentes (ou multiples). Cette place doit prendre en compte le stockage et la référence à ces pages de manuel. Les facteurs pertinents comprennent la langue (avec les différences géographiques), et le codage des caractères.

Le nommage des sous-répertoires spécifiques à la langue dans `/usr/share/man` est basé sur l'annexe E de la norme POSIX 1003.1 qui décrit la chaîne d'identification locale — la méthode la mieux acceptée pour décrire un environnement culturel. La chaîne `<locale>` est :

```
<langue>[_<territoire>][.<code_caractère>][,<version>]
```

Le champ `<langue>` sera tiré d'ISO 639 (un code de représentation des noms de langues). Il sera constitué de deux caractères et spécifié en lettres minuscules uniquement.

Le champ `<territoire>` sera le code à deux lettres d'ISO 3166 (spécification des représentations de pays) si possible. (La plupart des personnes sont familières avec les codes à deux lettres utilisés pour les codes de pays des adresses électroniques[1].) Il sera constitué de deux caractères spécifiés en lettres majuscules uniquement.

Le champ `<code_caractère>` devrait représenter la norme décrivant le code de caractères. Si le champ `<code_caractère>` est une simple indication numérique, le nombre représente le numéro de la norme internationale décrivant le code de caractères. Il est recommandé que ce soit une représentation numérique si possible (surtout pour les normes ISO), qu'il n'inclue pas de symboles de ponctuation supplémentaires et que toute lettre soit en minuscule.

Un paramètre spécifiant la `<version>` du profil peut être placé après le champ `<code_caractère>`, séparé par une virgule. Ceci peut servir à différencier plusieurs besoins culturels ; par exemple, l'ordre du dictionnaire comparé à un ordre d'assemblage plus orienté système. Cette norme recommande de ne pas utiliser le champ `<version>`, sauf si c'est nécessaire.

Les systèmes utilisant une langue et un code de caractères uniques pour toutes les pages de manuel peuvent omettre la sous-chaîne `<locale>` et stocker toutes les pages de manuel dans `<manrep>`. Par exemple, les systèmes qui n'ont que les pages de manuel en anglais codées en ASCII peuvent stocker ces pages (les répertoires `man<section>`) directement dans `/usr/share/man`. (Ce qui représente, en fait, la disposition et les circonstances habituelles.)

Les pays pour lesquels existe un ensemble de codes de caractères standard bien accepté peuvent omettre le champ `<code_caractère>`, mais il est fortement recommandé de l'inclure, surtout pour les pays ayant plusieurs normes en compétition.

Quelques exemples :

langue	territoire	code de caractère	répertoire
Anglais	—	ASCII	`/usr/share/man/en`
Anglais	Royaume Uni	ASCII	`/usr/share/man/en_GB`
Anglais	États-Unis	ASCII	`/usr/share/man/en_US`
Français	Canada	ISO 8859-1	`/usr/share/man/fr_CA`
Français	France	ISO 8859-1	`/usr/share/man/fr_FR`
Allemand	Allemagne	ISO 646	`/usr/share/man/de_DE.646`
Allemand	Allemagne	ISO 6937	`/usr/share/man/de_DE.6937`
Allemand	Allemagne	ISO 8859-1	`/usr/share/man/de_DE.88591`
Allemand	Suisse	ISO 646	`/usr/share/man/de_CH.646`
Japonais	Japon	JIS	`/usr/share/man/ja_JP.jis`
Japonais	Japon	SJIS	`/usr/share/man/ja_JP.sjis`
Japonais	Japon	UJIS (ou EUC-J)	`/usr/share/man/ja_JP.ujis`

1. Une exception majeure à cette règle est le Royaume-Uni, qui est 'GB' dans ISO 3166, mais 'UK' pour la plupart des adresses électroniques.

De même, il faut faire de la place pour les pages de manuel dépendant de l'architecture, comme la documentation sur les pilotes de périphériques ou les commandes d'administration système de bas niveau. Elles devraient être placées dans un répertoire `<arch>` dans le répertoire `man<section>` approprié ; par exemple, une page de manuel pour la commande i386 ctrlaltdel(8) peut être placée dans `/usr/share/man/<locale>/man8/i386/ctrlaltdel.8`.

Les pages de manuel pour les commandes et les données de `/usr/local` sont stockées dans `/usr/local/man`. Les pages de manuel pour X11R6 sont stockées dans `/usr/X11R6/man`. Il s'ensuit que toutes les hiérarchies de pages de manuel du système devraient avoir la même structure que `/usr/share/man`. Les répertoires vides peuvent être exclus d'une hiérarchie de pages de manuel. Par exemple, si `/usr/local/man` n'a pas de page de manuel pour la section 4 (périphériques), alors on peut omettre `/usr/local/man/man4`.

Les sections de pages cat (`cat<section>`) contenant des entrées de pages de manuel formatées se trouvent aussi dans les sous-répertoires de `<manrep>/<locale>`, mais ne sont pas obligatoires ni ne devraient être distribuées à la place des pages de manuel en source nroff.

Les sections numérotées "1" à "8" sont définies de manière traditionnelle. En général, le nom de fichier des pages de manuel situées dans une section particulière se termine par `.<section>`.

De plus, certains grands ensembles de pages de manuel spécifiques à une application possèdent un suffixe supplémentaire accolé au nom de fichier de la page de manuel. Par exemple, les pages de manuel du système de gestion de courrier MH devraient avoir `mh` accolé à toutes les pages de manuel de MH. Toutes les pages de manuel du système X Window devraient avoir un `x` accolé au nom de fichier.

La pratique de placer les pages de manuel de diverses langues dans les sous-répertoires appropriés de `/usr/share/man` s'applique aussi aux autres hiérarchies de pages de manuel, comme `/usr/local/man` et `/usr/X11R6/man`. (Cette partie de la norme s'appliquera aussi plus loin dans la section sur la structure optionnelle `/var/cache/man`.)

Voici une description de chaque section :

- `man1` : programmes utilisateurs
  Les pages de manuel décrivant les commandes accessibles à tous se trouvent dans ce chapitre. La majeure partie de la documentation sur les programmes dont aura jamais besoin un utilisateur se trouve ici.
- `man2` : appels système
  Cette section décrit tous les appels systèmes (requêtes au noyau pour effectuer des opérations).
- `man3` : fonctions et routines de la bibliothèque
  La section 3 décrit les routines de bibliothèque des programmes qui ne sont pas des appels directs aux services du noyau. Ceci ainsi que le chapitre 2 n'intéressent vraiment que les programmeurs.
- `man4` : fichiers spéciaux
  La section 4 décrit les fichiers spéciaux, les fonctions spécifiques aux pilotes et le support réseau disponible sur le système. On y trouve typiquement les fichiers de périphériques de `/dev` et l'interface noyau pour le support des protocoles réseau.
- `man5` : formats de fichiers
  Le format de nombreux fichiers de données non intuitifs est documenté dans la section 5. Ceci comprend divers fichiers d'en-têtes, les fichiers de résultats de programmes et les fichiers systèmes.

- `man6` : jeux
  Ce chapitre documente les jeux, les démonstrations et des programmes en général triviaux. Des personnes différentes ont des notions variées sur l'opportunité de cette partie.
- `man7` : divers
  Les pages de manuel difficiles à classer sont placées dans la section 7. Les paquets de macros pour troff et autres processeurs de texte se trouvent ici.
- `man8` : administration système
  La documentation pour les programmes utilisés par les administrateurs système pour la bonne marche du système et sa maintenance se trouve ici. Certains de ces programmes sont aussi utiles de temps à autre pour les utilisateurs normaux.

4.8.3 /usr/share/misc : diverses données indépendantes de l'architecture

Ce répertoire contient divers fichiers indépendants de l'architecture qui ne nécessitent pas un sous-répertoire séparé dans `/usr/share`. C'est un répertoire obligatoire de `/usr/share`.

Les fichiers suivants, s'ils sont présents, devraient se trouver dans `/usr/share/misc` :

```
{ ascii, magic, termcap, termcap.db }
```

D'autres fichiers (spécifiques à une application) peuvent apparaître ici, mais un intégrateur peut les placer dans `/usr/lib` à sa discrétion. De tels fichiers comprennent :

```
{ airport, birthtoken, eqnchar, getopt, gprof.callg, gprof.flat,
  inter.phone, ipfw.samp.filters, ipfw.samp.scripts, keycap.pcvt,
  mail.help, mail.tildehelp, man.template, map3270, mdoc.template,
  more.help, na.phone, nslookup.help, operator, scsi_modes, sendmail.hf,
  style, units.lib, vgrindefs, vgrindefs.db, zipcodes }
```

4.9 /usr/src : code source

Tout code source non local devrait être placé dans ce sous-répertoire.

5. La hiérarchie /var

/var — données variables

```
├── account   historique de la comptabilité des processus (si supporté)
├── cache     données de cache des applications
├── crash     données brutes des plantages système (si supporté)
├── games     données variables des jeux
├── lock      fichiers de verrous
├── log       fichiers et répertoires d'historique
├── mail      fichiers de boîtes aux lettres des utilisateurs
├── opt       données variables de /opt
├── run       fichiers relatifs aux processus en cours
├── spool     données en attente pour les applications
├── state     informations variables d'état
├── tmp       fichiers temporaires préservés entre les redémarrages du système
└── yp        fichiers de base de données de NIS (Network Information Service)
```

`/var` contient des fichiers de données variables. Ceci comprend les répertoires et fichiers en attente (spool), les données administratives et d'historique, et les fichiers transitoires et temporaires.

Certaines parties de `/var` ne sont pas partageables entre des systèmes différents. Par exemple, `/var/log`, `/var/lock` et `/var/run`. D'autres parties peuvent être partagées, comme notamment `/var/mail`, `/var/cache/man`, `/var/cache/fonts` et `/var/spool/news`.

`/var` est ici spécifié afin de rendre possible le montage de `/usr` en lecture seule. Tout ce qui allait auparavant dans `/usr` et sur lequel on écrivait pendant la marche du système (au contraire de l'installation et de la maintenance logicielle) doit aller dans `/var`.

Si on ne peut pas faire de `/var` une partition séparée, il est souvent préférable de déplacer `/var` hors de la partition racine à l'intérieur de la partition `/usr`. (On fait parfois ceci pour réduire la taille de la partition racine ou quand l'espace se réduit dans la partition racine.) Cependant, `/var` ne devrait pas être un lien vers `/usr` parce que cela rend la séparation de `/usr` et `/var` plus difficile et rend plus probable la création d'un conflit de noms. À la place, faites un lien de `/var` vers `/usr/var`.

Les applications ne devraient en général pas ajouter de répertoire au niveau supérieur de `/var`. De tels répertoires ne devraient être ajoutés que s'ils concernent le système en entier, et en consultant la liste de diffusion FHS.

Les répertoires `cache`, `lock`, `log`, `run`, `spool`, `state` et `tmp` doivent être inclus et utilisés dans toutes les distributions ; les répertoires `account`, `crash`, `games`, `mail` et `yp` doivent être inclus et utilisés si les applications ou possibilités correspondantes sont fournies dans la distribution.

Les versions précédentes de la FSSTND incluaient une hiérarchie `/var/lib`. Pour plus d'informations, voir la section sur `/var/state`.

Plusieurs répertoires sont 'réservés' dans le sens où une application nouvelle ne devrait pas les utiliser de façon arbitraire, puisqu'ils entreraient en conflit avec une pratique historique et/ou locale. Ce sont :

```
/var/backups
/var/cron
/var/lib
```

```
/var/local
/var/msgs
/var/preserve
```

5.1 /var/account : historique de la comptabilité des processus (si supporté)

Ce répertoire contient l'historique en cours de la comptabilité des processus et les données consolidées d'utilisation des processus (utilisés dans certains systèmes de type UNIX par `lastcomm` et `sa`).

5.2 /var/cache : données de cache des applications

/var/cache — répertoires de cache

```
├── fonts       fontes générées en local
├── man         pages de manuel formatées en local
├── www         données de cache ou de proxy WWW
└── <paquet>    données de cache spécifiques à paquet
```

`/var/cache` est fait pour les données de cache des applications. De telles données sont générées localement comme résultat d'entrées-sorties ou de calculs qui prennent du temps. L'application doit être capable de regénérer ou de retrouver les données. À la différence de `/var/spool`, les fichiers cachés peuvent être effacés sans perte de données. Les données doivent rester valides entre deux lancements de l'application et après le redémarrage du système.

Les fichiers situés dans /var/cache peuvent expirer d'une manière spécifique à l'application, par l'administrateur système ou des deux manières. L'application devrait toujours être capable de s'en remettre suite à un effacement manuel des fichiers (généralement à cause d'un manque d'espace disque). Aucune autre obligation n'est faite concernant le format des données dans les répertoires de cache.

Raison d'être :

L'existence d'un répertoire séparé pour les données de cache permet aux administrateurs système d'attribuer des politiques de disque et de sauvegarde différentes des autres répertoires de `/var`.

5.2.1 /var/cache/fonts : fontes générées en local

Le répertoire `/var/cache/fonts` devrait êre utilisé pour stocker toute fonte créée de manière dynamique. En particulier, `/var/cache/fonts/pk` stockera toutes les fontes automatiquement générées par `MakeTeXPK`.

Il devrait y avoir un lien de `/usr/lib/texmf/fonts/tmp` vers `/var/cache/fonts`. Ce lien permet aux utilisateurs d'utiliser le chemin unique `/usr/lib/texmf/fonts/tfm` quand ils font des modifications à leur variable d'environnement TEXFONTS. (Ceci est le chemin par défaut pour les outils TEX de Karl Berry, distribués sur `ftp.cs.umb.edu:/pub/tex`[2]. Si on utilise une autre distribution TEX, on devrait faire un lien du répertoire de fontes approprié vers `/var/cache/fonts`.)

---

2. La raison pour laquelle les outils de Karl Berry sont mentionnés est qu'ils sont le standard de-facto pour les installations UNIX de TEX. Ces outils sont largement utilisés dans la communauté UNIX libre.

Le MakeTeXPK distribué avec dvipsk placera les fichiers exemple, fonts/pk/CanonCX/cmr10.300pk). Les fichiers .pk peuvent être purgés régulièrement de l'arborescence /var/cache/fonts ou peuvent être déplacés dans l'arborescence /usr/lib/texmf. Si on utilise des générateurs automatiques de sous-répertoires mf ou tfm de /var/cache/fonts.

D'autres fontes créées dynamiquement peuvent aussi se trouver dans cette arborescence, dans des sous-répertoires de /var/cache/fonts nommés de manière adéquate.

5.2.2 /var/cache/man : pages de manuel formatées en local (optionnel)

Ce répertoire fournit un emplacement standard pour les sites qui possèdent une partition /usr en lecture seule, mais qui veulent permettre le cache des pages de manuel formatées en local. Les sites qui montent /usr en écriture (par exemple, les installations en utilisateur unique) peuvent choisir de ne pas utiliser /var/cache/man et peuvent écrire les pages de manuel formatées dans les répertoires cat<section> directement dans /usr/share/man. Nous recommandons que la plupart des sites utilisent plutôt l'une des options suivantes :

- préformater toutes les pages de manuel à côté des versions non formatées ;
- ne permettre aucun cache des pages de manuel formatées et obliger à refaire le formatage à chaque utilisation d'une page de manuel ;
- permettre le cache local des pages de manuel formatées dans /var/cache/man.

La structure de /var/cache/man nécessite de refléter à la fois les hiérarchies multiples de pages de manuel et la possibilité d'un support multilingue.

Étant donnée une page de manuel non formatée qui apparaît normalement dans <chemin>/man/<locale>/man<section>, le répertoire pour placer les pages de manuel formatées s'appelle /var/cache/man/<chemin_cat>/<locale>/cat<section>, où <chemin_cat> s'inspire de <chemin> en enlevant toute composante de nom de chemin usr au début et/ou share à la fin. (Notez que la composante <locale> peut ne pas être présente.)

Par exemple, /usr/share/man/man1/ls.1 sera formatée en /var/cache/man/cat1/ls.1 et /usr/X11R6/man/<locale>/man3/XtClass.3x en /var/cache/man/X11R6/<locale>/cat3/XtClass.3x.

Les pages de manuel écrites dans /var/cache/man peuvent à la fin être transférées vers les répertoires préformatés appropriés dans la hiérarchie source man ou bien expirer ; de même, les pages de manuel formatées dans la hiérarchie des sources man peuvent être expirées si personne n'y a accédé pendant une certaine période de temps.

Si les pages de manuel préformatées sont distribuées avec un système sur des supports en lecture seule (un CD-ROM, par exemple), elles seront installées dans la hiérarchie des sources man (par exemple /usr/share/man/cat<section>). /var/cache/man est réservé comme cache dans lequel on peut écrire les pages de manuel formatées.

Raison d'être :

La version 1.2 de la norme spécifiait `/var/catman` pour cette hiérarchie. Le chemin a été modifié en `/var/cache` pour mieux refléter la nature dynamique des pages de manuel formatées. Le nom du répertoire a été modifié en `man` pour permettre l'agrandissement de la hiérarchie et inclure des formats traités autres que "cat", comme PostScript, HTML ou DVI.

5.3 /var/crash : données brutes des plantages système (si supporté)

Ce répertoire contient des données brutes (dump) des plantages système. À la date de la sortie de cette norme, les données brutes de plantage système n'étaient pas supportées par Linux.

5.4 /var/games : données variables des jeux

Toute donnée variable concernant les jeux de `/usr` devrait être placée ici. `/var/games` devrait contenir les données variables qu'on trouvait auparavant dans `/usr` ; Les données statiques telles que les textes d'aide, les descriptions de niveaux et ainsi de suite devraient rester autre part, comme dans `/usr/share/games`.

Comme pour `/var/state`, les données variables des jeux peuvent être placées dans `/var/lib` en tant que mesure de transition obsolète. Cependant, cette permission sera levée dans une version future de la norme.

Raison d'être :

On a donné une hiérarchie `/var/games` à part entière, plutôt que de le laisser mélangé avec l'ancien `/var/lib` comme dans la version 1.2. La séparation permet un contrôle local des stratégies de sauvegarde, permissions et utilisation des disques, ainsi que de permettre un partage inter-machines et de réduire l'encombrement de `/var/state`. De plus, `/var/games` est le chemin utilisé traditionnellement par BSD.

5.5 /var/lock : fichiers de verrous

Les fichiers de verrous devraient être stockés dans la structure de répertoires `/var/lock`.

Les fichiers de verrous de périphériques, tels les fichiers de verrous du périphérique série qu'on trouvait d'habitude, soit dans `/usr/spool/locks`, soit dans `/usr/spool/uucp` doivent maintenant être stockés dans `/var/lock`. La convention de nommage à utiliser est `LCK..` suivi du nom de base du périphérique. Par exemple, pour verrouiller `/dev/cua0` le fichier `LCK..cua0` serait créé.

Le format utilisé pour les fichiers de verrous de périphérique doit être le format de fichier de verrou HDB UUCP. Le format HDB permet de stocker l'identificateur du processus (PID) comme un nombre décimal ASCII sur dix octets, avec un signe nouvelle ligne à la fin. Par exemple, si le processus 1230 tenait un fichier de verrou, ce dernier contiendrait les onze caractères : espace, espace, espace, espace, espace, espace, un, deux, trois, zéro et nouvelle ligne.

5.6 /var/log : fichiers et répertoires d'historique

Le répertoire contient divers fichiers d'historique (log). La plupart des historiques devraient être écrits dans ce répertoire ou dans un sous-répertoire approprié.

`lastlog`	enregistrement du dernier login de chaque utilisateur
`messages`	messages système de `syslogd`
`wtmp`	enregistrement de chaque login et logout

5.7 /var/mail : fichiers de boîtes aux lettres utilisateurs

Ce répertoire contient les fichiers de boîtes aux lettres utilisateurs stockés dans le format de boîtes aux lettres UNIX standard.

Raison d'être :

Ce répertoire a été relogé de `/var/spool/mail` pour amener FHS en accord avec la plupart des implémentations UNIX. Ce changement est important pour l'interopérabilité puisqu'un `/var/mail` unique est souvent partagé entre plusieurs machines et plusieurs implémentations d'UNIX (malgré les problèmes de verrouillage NFS).

5.8 /var/opt : données variables de /opt

Les données variables devraient être installées dans `/var/opt/<paquet>`, où `<paquet>` est le nom de la sous-arborescence de `/opt` où sont stockées les données statiques de tout paquet logiciel supplémentaire, sauf quand elles sont supplantées par un autre fichier dans `/etc`. Aucune structure n'est imposée sur la disposition interne de `/var/opt/<paquet>`.

Raison d'être :

Voir la raison d'être pour `/opt`.

5.9 /var/run : fichiers variables d'exécution

Ce répertoire contient des fichiers d'information système décrivant le système depuis qu'il a démarré. Les fichiers de ce répertoire devraient être nettoyés (enlevés ou tronqués selon le cas) au début du processus de démarrage.

Les fichiers d'identification de processus (PID), qui étaient placés à l'origine dans `/etc`, devraient être placés dans `/var/run`. La convention de nommage des fichiers PID est `<nom_programme>.pid`. Par exemple, le fichier PID de `crond` est nommé `/var/run/crond.pid`.

Le format interne des fichiers PID reste inchangé. Le fichier consiste en un identificateur de processus en décimal codé ASCII, suivi d'un caractère nouvelle ligne. Par exemple, si `crond` était le processus numéro 25, `/var/run/crond.pid` contiendrait trois caractères : deux, cinq et nouvelle ligne.

Les programmes qui lisent les fichiers PID devraient être assez souples dans ce qu'ils acceptent ; par exemple, ils devraient ignorer les espaces blancs supplémentaires, les zéros au début, l'absence d'une nouvelle ligne à la fin ou les lignes supplémentaires dans le fichier PID. Les programmes qui créent des fichiers PID devraient utiliser la spécification simple située dans le paragraphe ci-dessus.

Le fichier `utmp`, qui stocke les informations sur qui est en train d'utiliser le système, est placé dans ce répertoire.

Les programmes qui gardent des sockets du domaine UNIX temporaires devraient les placer dans ce répertoire.

5.10 /var/spool : données en attente pour les applications

/var/spool — répertoires d'attente

├── `lpd`	répertoire d'attente de l'imprimante
├── `mqueue`	file d'attente du courrier à l'expédition
├── `news`	répertoire d'attente des news
├── `rwho`	fichiers rwhod
├── `smail`	répertoire d'attente pour smail
└── `uucp`	répertoire d'attente pour UUCP

`/var/spool` contient des données en attente d'un traitement ultérieur. Les données de `/var/spool` représentent un travail à faire dans le futur (par un programme, un utilisateur ou un administrateur) ; les données sont souvent effacées après avoir été traitées.

Les fichiers de verrou UUCP doivent être placés dans `/var/lock`. Voir la section ci-dessus à propos de `/var/lock`.

5.10.1 /var/spool/lpd : file d'impression du démon de l'imprimante ligne

/var/spool/lpd — répertoire d'attente de l'imprimante

└── `<imprimante>`	file d'attente d'une imprimante spécifique (optionnel)

Le fichier de verrou de `lpd`, `lpd.lock`, devrait être placé dans `/var/spool/lpd`. Nous suggérons que le fichier de verrou de chaque imprimante soit placé dans le répertoire d'attente spécifique à cette imprimante et soit nommé `lock`.

5.10.2 /var/spool/rwho : fichiers rwhod

Ce répertoire contient les informations rwhod d'autres systèmes sur le réseau local.

Raison d'être :

Certaines versions de BSD utilisent `/var/rwho` pour ces données ; étant donné leur emplacement historique dans `/var/spool` sur d'autres systèmes et sa convenance approximative à la définition de données 'en attente', cet emplacement a été estimé plus approprié.

5.11 /var/state : informations variables d'état

/var/state — informations variables d'état

├── `<éditeur>`	fichiers de sauvegarde et état d'éditeurs
├── `misc`	diverses données d'état
├── `xdm`	données variables du gestionnaire d'affichage X xdm
├── `<pkgtool>`	fichiers d'aide au paquet
└── `<paquet>`	données d'état pour les paquets et les sous-systèmes

Cette hiérarchie contient des informations d'état se rapportant à une application ou au système. Les informations d'état sont des données que les programmes modifient au cours de leur exécution, relatives à une machine spécifique. Les utilisateurs ne devraient jamais avoir besoin de modifier des fichiers dans `/var/state` pour configurer la bonne marche d'un paquet.

Les informations d'état sont utilisées en général pour préserver l'environnement d'une application (ou d'un groupe d'applications liées entre elles) entre plusieurs lancements et entre plusieurs instances de la même application. Les informations d'état devraient en général rester valides après un redémarrage, ne devraient pas garder l'historique de sortie d'un programme et ne devraient pas être des données en attente.

Une application (ou un groupe d'applications liées) devrait utiliser un sous-répertoire de `/var/state` pour ses données. Il y a un sous-répertoire obligatoire, `/var/state/misc`, fait pour les fichiers d'état qui n'ont pas besoin de sous-répertoire ; les autres sous-répertoires ne devraient être présents que si l'application en question est incluse dans la distribution.

`/var/state/<nom>` est l'endroit à utiliser pour tout le support de paquet de la distribution. Des distributions différentes peuvent évidemment utiliser des noms différents.

Les versions précédentes de cette norme utilisaient le nom `/var/lib` pour cette hiérarchie. `/var/lib` est obsolète, mais peut être utilisé en parallèle avec la hiérarchie obligatoire `/var/state`, comme mesure transitoire pour les données spécifiques aux applications. Notez cependant que cette permission sera retirée dans une version future de la norme. Par contre, vous pouvez faire un lien symbolique de `/var/lib` vers `/var/state`.

Raison d'être :

`/usr/lib` est de plus en plus utilisé uniquement pour les fichiers objets ou leurs archives ; ceci est vrai pour les variantes BSD UNIX actuelles ainsi que les paquets GNU actuels. Dans le même ordre d'idées, le nom `/var/lib` semblait inapproprié.

BSD utilise le nom `/var/db` pour un répertoire similaire. Ce nom semblait trop restrictif, puisqu'il impliquait une structure de répertoires faite tout d'abord pour les fichiers de base de données (`.db`).

5.11.1 /var/state/<éditeur> : fichiers de sauvegarde et état d'un éditeur

Ces répertoires contiennent des fichiers sauvegardés par toute fin inattendue d'un éditeur (par exemple elvis, jove, nvi).

D'autres éditeurs peuvent ne pas nécessiter de répertoire pour les fichiers de sauvegarde de plantage, mais peuvent nécessiter un endroit bien défini pour stocker d'autres informations pendant que l'éditeur fonctionne. Ces informations devraient être stockées dans un sous-répertoire de `/var/state` (par exemple, GNU Emacs placerait ses fichiers de verrou dans `/var/state/emacs/lock`).

Des éditeurs futurs pourront avoir besoin d'informations d'état supplémentaires au-delà des fichiers de sauvegarde de plantage et des fichiers de verrou -- ces informations devraient aussi être placées dans `/var/state/<éditeur>`.

Raison d'être :

Les versions précédentes de Linux, ainsi que tous les distributeurs du commerce, utilisaient `/var/preserve` pour vi et ses clones. Cependant, chaque éditeur utilise son propre format pour ces fichiers de sauvegarde de plantage, c'est pourquoi un répertoire séparé est nécessaire à chaque éditeur.

Les fichiers de verrous spécifiques à chaque éditeur sont en général assez différents des fichiers de verrous de périphérique ou de ressources stockés dans `/var/lock` et de ce fait sont stockés dans `/var/state`.

5.11.2 /var/state/misc : diverses données variables

Ce répertoire contient des données variables qui ne sont pas placées dans un sous-répertoire de `/var/state`. Il serait judicieux d'utiliser dans la mesure du possible des noms uniques dans ce répertoire pour éviter les conflits de noms.

Notez que cette hiérarchie devrait contenir les fichiers de `/var/db` des versions actuelles de BSD. Celles-ci comprennent `locate.database` et `mountdtab`, et la (les) base(s) de données des symboles du noyau.

5.12 /var/tmp : fichiers temporaires préservés entre les redémarrages du système

Le répertoire `/var/tmp` est mis à la disposition des programmes qui ont besoin de fichiers ou de répertoires temporaires préservés entre les redémarrages du système. Par conséquent, les données stockées dans `/var/tmp` restent plus longtemps que les données de `/tmp`.

Les fichiers et répertoires situés dans `/var/tmp` ne doivent pas être effacés quand le système démarre. Bien que les données stockées dans `/var/tmp` soient typiquement effacées d'une manière spécifique au site, il est recommandé que l'effacement se fasse dans un intervalle plus long que pour `/tmp`.

5.13 /var/yp : fichiers de base de données NIS (Network Information Service)

Les données variables du Service d'Information Réseau (NIS ou Network Information Service), qu'on appelait auparavant Pages Jaunes Sun (YP ou Sun Yellow Pages), seront placées dans ce répertoire.

Raison d'être :

`/var/yp` est le répertoire normal des données NIS (YP) et est utilisé presque exclusivement dans la documentation et les systèmes NIS.

Il ne faut pas confondre NIS et Sun NIS+, qui utilise un répertoire différent, `/var/nis`.

6. Annexe spécifique aux systèmes d'exploitation

Cette section contient des obligations et recommandations supplémentaires qui ne s'appliquent qu'à un système d'exploitation spécifique. Les principes de cette section ne devraient jamais entrer en conflit avec la norme de base.

6.1 Linux

Voici l'annexe pour le système d'exploitation Linux.

6.1.1 / : répertoire racine

Sur les systèmes Linux, si le noyau est situé dans `/`, nous recommandons d'utiliser les noms `vmlinux` ou `vmlinuz`, qui sont utilisés dans les paquets récents de sources du noyau Linux.

6.1.2 /dev : fichiers de périphériques et fichiers spéciaux

Tous les fichiers de périphériques et fichiers spéciaux de `/dev` devraient se conformer au document *Linux Allocated Devices* (périphériques alloués dans Linux), que l'on trouve dans les sources du noyau Linux. Il est maintenu par H. Peter Anvin <hpa@zytor.com>.

Les liens symboliques de `/dev` ne devraient pas être distribués avec des systèmes Linux sauf s'ils sont fournis dans le document *Linux Allocated Devices*.

Raison d'être :

L'obligation de ne pas faire de liens symboliques au hasard est donnée parce que la configuration locale diffère souvent de celle de la machine de développement du distributeur. De plus, si un script d'installation de distribution configure les liens symboliques au moment de l'installation, ces liens symboliques ne seront souvent pas mis à jour si des changements locaux ont lieu sur le matériel. De manière responsable et à un niveau local, cependant, on peut les utiliser à bon escient.

6.1.3 /proc : système de fichiers virtuel d'informations sur le noyau et les processus

Le système de fichiers proc devient la méthode standard de-facto sur Linux pour manipuler les processus et les informations du système, plutôt que par `/dev/kmem` et autres méthodes similaires. Nous encourageons fortement ce principe pour le stockage et l'obtention d'informations sur les processus ainsi que d'autres informations sur le noyau et la mémoire.

6.1.4 /sbin : binaires systèmes importants

Les systèmes Linux placent ces fichiers supplémentaires dans `/sbin` :

- commandes du système de fichiers étendu deuxième version (ext2 -- optionnel) :

  `{ badblocks, dumpe2fs, e2fsck, mke2fs, mklost+found, tune2fs }`

- installateur de carte du chargeur de démarrage :

  `{ lilo }`

Fichiers optionnels de /sbin :

- binaires statiques :

  `{ ldconfig, sln, ssync }`

  Les binaires statiques `ln` (`sln`) et `sync` (`ssync`) sont utiles quand quelque chose va mal. L'utilisation principale de `sln` (pour réparer les liens symboliques incorrects dans `/lib` après une mise à jour mal faite) n'est plus d'une importance cruciale maintenant que le programme `ldconfig` (situé généralement dans `/usr/sbin`) existe et peut agir comme guide pendant la mise à jour des bibliothèques. sync en statique est utile dans certaines situations d'urgence. Notez qu'ils ne doivent pas forcément être des versions liées en statique des commandes normales `ln` et `sync`, mais elle peuvent l'être.

  Le binaire `ldconfig` est optionnel dans `/sbin` puisqu'un site peut choisir de lancer `ldconfig` au démarrage plutôt qu'à la mise à jour des bibliothèques partagées. (Il n'est pas clair qu'il soit avantageux de lancer `ldconfig` à chaque démarrage.) Même ainsi, certaines personnes aiment avoir `ldconfig` à portée de clavier pour les situations suivantes (toutes trop fréquentes) :

  (1) je viens d'enlever `/lib/<fichier>` ;

  (2) je ne peux pas trouver le nom de la bibliothèque parce que `ls` est lié en dynamique, j'utilise un shell qui n'a pas `ls` intégré et je ne sais pas utiliser "`echo *`" à la place ;

  (3) j'ai un `sln` en statique, mais je ne sais pas comment nommer le lien.

- divers :

  `{ ctrlaltdel, kbdrate }`

  Pour pallier au fait que certains claviers sont livrés avec une fréquence de répétition si grande qu'ils en sont inutilisables, `kbdrate` peut être installé dans `/sbin` sur certains systèmes.

  Puisque l'action par défaut dans le noyau pour la combinaison de touches Ctrl-Alt-Del est un redémarrage brutal instantané, il est recommandable de désactiver ce comportement avant de monter le système de fichiers racine en mode lecture/écriture. Certaines versions d'`init` sont capables de désactiver Ctrl-Alt-Del, mais d'autres nécessitent le programme `ctrlaltdel` qui peut être installé dans `/sbin` sur ces systèmes.

6.1.5 /usr/include : fichiers d'en-tête inclus par les programmes C

Les liens symboliques suivants sont nécessaires si un compilateur C ou C++ est installé.

```
/usr/include/asm -> /usr/src/linux/include/asm-<arch>
/usr/include/linux -> /usr/src/linux/include/linux
```

6.1.6 /usr/src : code source

Le seul code source qui doit être placé dans un endroit spécifique est le code source du noyau Linux. Il est situé dans `/usr/src/linux`.

Si un compilateur C ou C++ est installé, mais que le code source complet du noyau Linux n'est pas installé, les fichiers d'en-tête du code source du noyau devront être situés dans ces répertoires :

```
/usr/src/linux/include/asm-<arch>
/usr/src/linux/include/linux
```

`<arch>` est le nom de l'architecture du système.

Note : `/usr/src/linux` *peut être un lien symbolique vers l'arborescence réelle du code source du noyau.*

Raison d'être :

Il est important que les fichiers d'en-têtes du noyau soient situés dans `/usr/src/linux` et non dans `/usr/include` pour qu'il n'y ait pas de problemes quand les administrateurs système mettent à jour la version du noyau pour la première fois.

6.1.7 /var/spool/cron : travaux cron et at

Ce répertoire contient les données variables pour les programmes cron et at.

La liste de diffusion FHS La liste de diffusion FHS est située à <fhs-discuss@ucsd.edu>. Pour vous abonner à la liste envoyez un courrier à <listserv@ucsd.edu> avec dans le corps du message "`ADD fhs-discuss`".

Merci au service réseau de l'université de Californie à San Diego qui nous a autorisés à utiliser leur super serveur de listes de distribution.

Comme il est indiqué dans l'introduction, veuillez ne pas envoyer de courrier à la liste de diffusion sans d'abord contacter l'éditeur de la FHS ou un contributeur listé.

Remerciements

Les développeurs de la FHS souhaitent remercier les développeurs, administrateurs système et utilisateurs dont l'avis a été essentiel à cette norme. Nous souhaitons remercier chacun des contributeurs qui ont aidé à écrire, compiler et composer cette norme.

Le groupe FHS souhaite aussi remercier les développeurs Linux qui ont supporté la FSSTND, prédécesseur de cette norme. S'ils n'avaient pas démontré le bénéfice apporté par la FSSTND, la FHS n'aurait jamais pu évoluer.

Contributeurs

Brandon S. Allbery	<bsa@kf8nh.wariat.org>
Keith Bostic	<bostic@cs.berkeley.edu>
Drew Eckhardt	<drew@colorado.edu>
Rik Faith	<faith@cs.unc.edu>
Stephen Harris	<sweh@spuddy.mew.co.uk>
Ian Jackson	<ijackson@cus.cam.ac.uk>
John A. Martin	<jmartin@acm.org>
Ian McCloghrie	<ian@ucsd.edu>
Chris Metcalf	<metcalf@lcs.mit.edu>
Ian Murdock	<imurdock@debian.org>
David C. Niemi	<niemidc@clark.net>
Daniel Quinlan	<quinlan@pathname.com>
Eric S. Raymond	<esr@thyrsus.com>
Mike Sangrey	<mike@sojurn.lns.pa.us>
David H. Silber	<dhs@glowworm.firefly.com>
Theodore Ts'o	<tytso@athena.mit.edu>
Stephen Tweedie	<sct@dcs.ed.ac.uk>
Fred N. van Kempen	<waltje@infomagic.com>

Traduction :

La traduction française a été réalisée par Olivier Tharan, <olive@minet.net>. Tous les commentaires sont acceptés. La relecture a été effectuée le 16/01/2000.

CONTENTS