

# *Voyage au centre du noyau Linux*

## *organisation, méthodes, outils, licences*

Stelian Pop <stelian.pop@fr.alcove.com>



Retour
Fermer

# Introduction

Le noyau Linux est un des fleurons des logiciels libres.

Ceci est du à un modèle de développement très efficace favorisant à la fois :
- ✓ la stabilité ;
- ✓ les performances ;
- ✓ l'évolutivité.

Comment ça marche ?
- ✓ organisation des développeurs basée sur le mérite ;
- ✓ méthodes de travail efficaces ;
- ✓ outils adaptés ;
- ✓ licences favorisant le progrès.



# Organisation - le code est le roi

L'organisation des développeurs est une méritocratie :

- ✓ personne n'est irremplaçable (même Linus) ;
- ✓ celui qui a le meilleur code a le dernier mot ;
- ✓ ceux qui ont souvent le meilleur code "montent" en grade ;
- ✓ seul le code compte, pas les (mauvaises) raisons commerciales.

☛ voir `Documentation/ManagementStyle`



# Organisation - organigramme

L'organigramme est complexe et mobile, aperçu actuel :

- ✓ chef suprême : *Linus Torvalds*
- ✓ adjoint : *Andrew Morton*
- ✓ responsables par architecture :
  - ARM : *Russell King*
  - x86-64 : *Andi Kleen*
  - SPARC32 : *William L. Irwin*
  - IA64 : *David Mosberger-Tang*
  - ...

# Organisation - organigramme suite

✓ lieutenants (responsables des sous-ensembles noyau) :
- ordonnanceur : *Ingo Molnar, Robert Love*
- infrastructure pilotes : *Greg Kroah-Hartman*
- systèmes de fichiers : *Alexander Viro*
- modules : *Rusty Russell*
- input : *Vojtech Pavlik*
- réseau : *David Miller, Alexey Kuznetsov, ...*
- SCSI : *James Bottomley*
- USB : *Greg Kroah-Hartman*
- ...

✓ responsables des pilotes :
- environ 300-500 développeurs...

☛ voir `MAINTAINERS/CREDITS`.

# Organisation - financement

Financement :

- ✓ sociétés distributrices de *GNU/Linux* (*Red Hat*, *SuSE* etc) ;
- ✓ sociétés assurant le support (*Red Hat*, *Alcôve* etc) ;
- ✓ constructeurs de matériel (*Intel*, *IBM* etc) ;
- ✓ organisations multi-entreprises (*OSDL* etc) ;
- ✓ l'état (chercheurs utilisant Linux pour la recherche) ;
- ✓ bénévoles (étudiants, temps libre).



# Méthodes - développement distribué

Le développement se fait d'une façon distribuée.

Le moyen de coordination est le courrier électronique : liste de diffusion *LKML* (environ 10000 mails/mois).

- FAQ de LKML : http://www.tux.org/lkml/
- Archives de LKML : http://www.lkml.org/
- Résumés hebdomadaires de LKML : http://www.kerneltraffic.org/




# Méthodes - revue de pairs

Tout développement noyau doit passer (et être discuté) sur *LKML* avant acceptation (revue de pairs).

La barre d'acceptation de code est placée très haut, obtenant ainsi :
– un code très propre ;
– un code très performant ;
– une véritable "oeuvre d'art".

☛ voir `Documentation/CodingStyle` ;
☛ voir `Documentation/SubmittingDrivers` ;
☛ voir `Documentation/SubmittingPatches`.

# Méthodes - branches

Il n’existe pas **une** version de Linux, mais une **multitude** de versions.

Linus maintient une branche (que l’on appelle la branche officielle), mais encourage la création de *forks* :

- ✓ les *forks* se basent sur la branches officielle ;
- ✓ les *forks* visent à réintégrer la branche officielle.

# Méthodes - exemples de branches

✓ branches officielles (branches *Linus* ou *kernel.org*) :
- branches stable : *1.0*, *1.2*, *2.0*, *2.2*, *2.4*, *2.6*
- branches de développement : *1.1*, *1.3*, *2.1*, *2.3*, *2.5*
- cas spécial : pas de branche *2.7* pour l'instant

✓ branches pour les architectures non x86 :
- *arm* : http://www.arm-linux.org/
- *m64k* : http://www.linux-m68k.org/
- *itanium* : http://www.ia64-linux.org/
- etc..

# Méthodes - exemples de branches - suite

✓ branches alternatives :
- *-mm* (*Andrew Morton*) :
  http://www.kernel.org/pub/linux/kernel/people/akpm/
- *-ac* (*Alan Cox*) :
  http://www.kernel.org/pub/linux/kernel/people/alan/
- *-aa* (*Andrea Arcangeli*) :
  http://www.kernel.org/pub/linux/kernel/people/andrea/
- *-wolk* (*Working Overloaded Linux Kernel*) :
  http://sourceforge.net/projects/wolk/
- etc...

✓ branches des distributions:
- *Red Hat*: basée sur une branche *-ac*;
- *SuSE*: basée sur une branche *-aa*;
- etc...

# Méthodes - API/ABI en constante évolution

L'ABI (*Application Binary Interface*) n'est pas stable entre deux versions du noyau (même mineures).

L'API (*Application Programming Interface*) est généralement stable entre deux versions mineures du noyau, mais pas toujours.

Cela permet d'améliorer continuellement les interfaces sans devoir traîner le poids de la compatibilité.

De toute façon, la plupart des pilotes Linux sont présents directement dans l'arborescence du noyau et sont mis à jour immédiatement.

Mais les pilotes externes doivent suivre manuellement...

# Outils - patches

Un *patch* est à la fois :

- ✓ une modification de source utilisable par des outils automatiques ;
- ✓ un support de discussion entre développeurs.

La maîtrise des patches est indispensable pour tout développeur noyau (`diff, patch, lsdiff, filterdiff`, etc.).

Outils de gestion de patches :

- ✓ *quilt* : http://savannah.nongnu.org/projects/quilt/
- ✓ *patch-scripts* : http://developer.osdl.org/rddunlap/scripts/

# Outils - gestion de sources

Jusqu'au noyau 2.4, Linus n'utilisait aucun système de gestion de sources.

Depuis, un certain nombre de développeurs noyau (y compris Linus) utilisent *BitKeeper* (http://www.bitmover.com/).

- historique complet
- adapté au mode de développement distribué du noyau
- merges entre branches facilitées
- fournit une passerelle *CVS/SVN*

- logiciel propriétaire
- licence d'utilisation controversée

# Outils - kbuild

Outil de configuration et de compilation du noyau développé sur mesure :

- ✓ langage de définition d'options noyau (sélection du processeur, compilation de pilotes, etc) gérant les dépendances ;
- ✓ IHM permettant de modifier les options ;
- ✓ ensemble de *Makefiles* permettant la compilation du noyau.

*kbuild* permet d'intégrer du code facilement dans le noyau :

- ✓ 3-10 lignes pour définir l'option (fichier *Kconfig*) ;
- ✓ 1 ligne pour compiler le source (fichier *Makefile*).

# Outils - sparse

*sparse* est un outil d'analyse du code source C permettant de détecter des erreurs de programmation.

Le code C du noyau a été enrichi avec des annotations spéciales permettant de distinguer par exemple :
– un pointeur noyau d'un pointeur utilisateur (et détecter les erreurs) ;
– un entier big-endian d'un entier little-endian (et contrôler les opérations) ;
– etc.

➶ Il a permis de détecter de nombreux bogues dans le code noyau.

☛ voir http://www.kernel.org/pub/software/devel/sparse/

# Outils - bugzilla

Bugzilla est une base de données de rapports et suivi de bogues.

➶ Outil mature, utilisé par des projets importants (*Mozilla*, *Red Hat* etc.)

➴ Pas très adapté au multi-branche du noyau.
➴ Pas adapté non plus à la vitesse de développement du noyau.

☛ voir http://bugzilla.kernel.org/

# Outils - LTP

LTP (*Linux Test Project*, http://ltp.sourceforge.net/) est un ensemble de tests automatiques qui permettent de trouver les régressions d'une nouvelle version du noyau.

LTP n'est pas automatiquement utilisé avant une release (mais pourrait l'être dans le futur).

Les résultats sont dépendants de la configuration matérielle et logicielle de l'environnment de test (architecture, nombre de processeurs, quantité de mémoire, périphériques, etc.).

Résultats pour les noyaux récents : http://developer.osdl.org/bryce/ltp/

# Licences - auteurs

Le noyau Linux est la propriété (*droit d'auteur*) de centaines de développeurs.

Le *droit d'auteur* est automatique (*Convention de Berne*).

L'appartenance multiple est une garantie contre l'appropriation du code par qui que se soit (changement de licence, etc).

Il peut être difficile de tracer l'origine d'une modification, donc retrouver l'auteur d'un morceau de code.

Les développeurs noyau signent maintenant toute modification (via une ligne attachée à chaque patch) afin de garder l'historique :

```
Signed-off-by : Foo Bar <foo@bar.net>
```

# Licenses - distribution

Le noyau Linux est distribué sous la licence *GPL v2* :
✓ donne le droit d'avoir le code source, le modifier et le redistribuer ;
✓ oblige de distribuer les travaux dérivés sous la même licence, au format source.

Les applications (utilisant les appels système du noyau) ne sont pas concernés par la licence.

Les modules noyau peuvent ne pas être *GPL* si et seulement si :
– utilisant une API restreinte (`EXPORT_SYMBOL`) ;
– ne sont pas un travail dérivé (mais qu'est-ce qu'un travail dérivé ?) ;
– déconseillé, aucun support de la communauté.

# Conclusion

Le noyau Linux est une preuve concrète de l'efficacité du modèle de développement du *bazar*.

Les développeurs noyau sont allés au delà du simple *bazar* en construisant une organisation, des méthodes et des outils adaptés (modèle de la *cathédrale*).

Le modèle de développement du noyau combine le *bazar* avec la *cathédrale* afin de tirer le meilleur des deux mondes.

A vous d'en faire partie !

# Liens

- Le site officiel de distribution du noyau
  http://www.kernel.org/

- Site consacré au développeurs noyau débutants
  http://www.kernelnewbies.org/

- Site de nouvelles concernant les noyaux Linux, BSD, autres
  http://www.kerneltrap.org/

- La cathédrale et le bazar (par Eric Raymond)
  http://http://www.catb.org/~esr/writings/cathedral-bazaar/



# Liens - suite

- Alcôve
  http://www.alcove.com/

- Stelian Pop
  http://popies.net/
  <stelian.pop@fr.alcove.com>
  <stelian@popies.net>