Linux : histoire d'un noyau

Christophe BLAESS – Août 2003*

Comme c'est souvent le cas dans les disciplines scientifiques, l'informatique est dotée d'une histoire dont l'étude est intéressante à bien des titres. Outre l'aspect anecdotique des aventures humaines (ou commerciales), il est possible de retracer la piste de concepts, de langages, de systèmes d'exploitation et de logiciels d'application à travers un héritage souvent disparate. Tout naturellement, il en va de même pour le noyau Linux, qui bien que relativement jeune, est déjà riche d'une histoire assez conséquente, comme nous le verrons dans cet article.

Ancêtres et influences

Si je devais définir en une phrase le système Linux, je dirais qu'il s'agit, dans cet ordre :

- d'un système d'exploitation compatible Unix,
- libre,
- développé à l'origine, à titre de passe-temps par un étudiant finlandais et de nombreux bénévoles passionnés d'informatique.

Cette définition, qui n'est pas particulièrement innovante, j'en conviens, est intéressante car elle contient les trois influences majeures qui ont permis la naissance de Linux.

AT&T Bell Labs - 1969

La première influence de Linux est évidemment le système d'exploitation Unix. Mais, plus que le système Unix BSD que nous verrons plus bas, j'estime que l'héritage essentiel pour l'esprit Linux est celui du premier Unix né dans les laboratoires Bell AT&T en 1969. Il s'agit d'un système d'exploitation rudimentaire dont les pères - Dennis Ritchie et Ken Thompson - sont de véritables hackers au sens noble du terme. L'esprit de partage des connaissances et de performance intellectuelle pure est véritablement au coeur de la communauté informatique de cette époque.

L'anecdote est bien connue du petit ordinateur PDP-7 inutilisé dans les laboratoires Bell, sur lequel Thompson désirait porter un jeu de sa conception intitulé "Space Travel". Le système d'exploitation étant indisponible, il fut contraint de reprendre la programmation de son environnement de travail à zéro. Aidé de son confrère Ritchie, il réalisa le tour de force de programmer un noyau de système d'exploitation multitâche, un système de fichiers, un mécanisme de gestion mémoire et un squelette de gestion des périphériques en quelques semaines.

Si le système Unix original appartenait par licence aux laboratoires AT&T Bell, le peu de cas qu'ils en faisaient permettait à Thompson de le distribuer gratuitement aux utilisateurs intéressés sous forme de code source. Chose que l'on retrouvera naturellement dans les logiciels libres et avec Linux. Même lorsque la distribution commerciale du système Unix commença, des licences peu coûteuses étaient réservées aux universités, permettant aux étudiants de se familiariser avec les détails internes du système.

La situation évoluera au début des années 80 quand AT&T réalisant le potentiel commercial d'Unix décidera d'en fermer l'accès aux sources, et de limiter la diffusion à un système de licences payantes classiques. Par réaction, ceci donnera naissance à l'Unix BSD qui verra le jour sous le soleil de la Californie à l'Université de Berkeley, où Thompson avait enseigné quelques années auparavant, essaimant des concepts qui seront mis en oeuvre par les étudiants et chercheurs.

La concurrence entre les deux branches d'Unix, sera d'ailleurs profitable à l'ensemble des systèmes

* Cet article a été publié dans le numéro Hors-Série 16 de Linux Magazine France

puisque de cette émulation naîtront de nombreuses innovations (sockets d'un côté, IPC de l'autre par exemple). D'autre part, les descendants libres de l'Unix BSD (FreeBSD, NetBSD, etc.) donneront naissance à de nombreux outils qui seront réutilisés dans les distributions Linux.

MIT AI Lab - 1984

L'évolution des systèmes d'exploitation vers des modèles de plus en plus fermés rendait impossible l'examen et l'amélioration du code source par l'utilisateur curieux désireux de l'adapter à son besoin personnel. Si cette situation s'avérait supportable pour beaucoup d'utilisateurs habitués aux systèmes propriétaires, elle l'était nettement moins pour les passionnés qui aimaient se plonger dans le coeur du système pour en comprendre le fonctionnement.

En 1984, un jeune chercheur du laboratoire d'Intelligence Artificielle du MIT décida de lancer un projet de substitut libre pour le système Unix, où tous les outils seraient disponibles gratuitement et sous forme de code source. Richard Stallman fonda à cette occasion le projet GNU (Gnu is Not Unix) et surtout lui donna un cadre formel important : la GPL (General Public License) qui permet de garantir la liberté de modification et redistribution, ainsi que la pérennité de l'accès aux sources d'un logiciel

En quelques années, le système GNU s'enrichit d'un grand nombre d'utilitaires simples (ls, cp, mv, etc.) ou complexes (éditeur Emacs, shell Bash, compilateur gcc, etc.) mais il manquait encore le noyau, le coeur du système d'exploitation, chargé de réguler les accès aux ressources des différentes applications. Pour réaliser ce travail, un projet nommé HURD démarra ; toutefois, plutôt que de "copier" le comportement d'un système existant (comme Linux avec Unix), un véritable travail de conception indépendant fut entrepris. Malheureusement, en raison de l'ampleur de la tâche, le développement du noyau Hurd fut long, et pendant de nombreuses années les outils GNU existaient sans qu'aucun noyau ne soit présent dans ce futur système d'exploitation. C'est naturellement dans cette niche que viendra se glisser le noyau Linux.

Amsterdam 1987

Le troisième point que j'ai mentionné pour définir le noyau Linux insiste sur l'aspect hobby de sa création. Il s'agissait véritablement d'un passe-temps d'étudiant, destiné à mettre en pratique les connaissances théoriques apprises dans le domaine des systèmes d'exploitation. En cela, Linux avait déjà un célèbre prédécesseur : le système Minix.

En 1987, Andrew Tanenbaum, chercheur et professeur à la *Vrije Universiteit* d'Amsterdam, créa un mini-système Unix simplifié, capable de fonctionner sur des ordinateurs de type PC-XT (processeurs 8088) auquel il donna le nom de Minix. Utilisant son code source pour enseigner la théorie des systèmes d'exploitation, il le publia dans un ouvrage célèbre "Operating Systems: Design and Implementation".

Tanenbaum n'a jamais désiré transformer Minix en un système véritablement utilisable en production, mais au contraire le conserver comme outil d'étude et d'enseignement. En conséquence, il refusait la plupart des extensions proposées par les utilisateurs. De plus la licence d'utilisation de Minix était très contraignante, puisqu'elle ne permettait pas la redistribution de versions modifiées du code source.

Toutefois, en 1991, Linus Torvalds lut avec grand intérêt le livre de Tanenbaum, expérimenta sur son PC flambant neuf le système Minix, et décida de l'utiliser comme source d'inspiration pour réécrire un nouveau système d'exploitation, sans néanmoins réutiliser le code source de Minix.

Naissance

```
From:   torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject:  What would you like to see most in minix?
Date:   25 Aug 91 20:57:08 GMT
Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
```

```
among other things).
I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want. Any suggestions
are welcome, but I won't promise I'll implement them:-)
Linus (torvalds@kruuna.helsinki.fi)
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have:-(.
```

Traduction :

```
Salut à tous les utilisateurs de Minix,
Je suis en train de réaliser un système d'exploitation (gratuit), (c'est juste un
passe-temps, il ne sera pas important et professionnel comme le Gnu) pour les
clones d'AT 386 (ou 486). Il mijote depuis le mois d'avril, et commence à être au
point. J'aimerais des remarques sur ce que les gens aiment ou non dans Minix, car
mon système lui ressemble un peu (même organisation du système de fichiers, pour
des raisons pratiques, entre autres).
J'y ai porté Bash (1.08) et Gcc (1.40), et tout semble fonctionner. Je devrais
donc disposer de quelque chose d'utilisable dans les mois à venir, et j'aimerais
connaître les fonctionnalités que la plupart d'entre vous aimeraient. Toutes les
suggestions sont les bienvenues, mais je ne promets pas de toutes les
implémenter ;-)
Linus (torvalds@kruuna.helsinki.fi)
PS : Oui, il est indépendant de tout code provenant de Minix, et il dispose d'un
système de fichiers multi-thread. Il n'est PAS portable (utilisation de la
commutation de tâches 386, etc.) et ne supportera probablement jamais rien
d'autres que les disques durs AT car ce sont les seuls dont je dispose :-(.
```

C'est par ce message dans un groupe Usenet consacré au système Minix, que Linus Torvalds, étudiant en seconde année d'informatique de l'Université d'Helsinki, a fait connaître publiquement son "passe-temps sans prétention". En quelques semaines des passionnés ont harcelé Linus de questions, si bien qu'il a rapidement mis les sources de son projet en accès libre. C'est à ce moment qu'Ari Lemke, le responsable informatique gérant le serveur ftp.funet.fi, dut créer le répertoire de stockage des sources, et choisit le nom "Linux" en guise de plaisanterie sur "Linus's Unix".

Le noyau Linux est officiellement né avec une version 0.0.1 publiée le 17 septembre 1991. La taille totale des sources représentait moins de 64 Ko avec compression (les versions actuelles compressées pèsent au moins 20 Mo). Il disposait toutefois de cinquante appels-système, c'est-à-dire l'essentiel des fonctionnalités de base de gestion des processus, de la mémoire et des fichiers :

```
fork(), exit(), waitpid(), execve(), getpid(), getppid(), setuid(),
getuid(), geteuid(), setgid(), getgid(), getegid(), setpgid(), getpgrp(),
setsid(), open(), creat(), close(), read(), write(), lseek(), access(),
ioctl(), fcntl(), mkdir(), rmdir(), dup(), dup2(), pipe(), link(),
unlink(), stat(), fstat(), chmod(), chown(), utime(), chdir(), chroot(),
sync(), stime(), time(), kill(), signal(), alarm(), pause(), brk(),
umask(), nice(), times(), uname().
```

Les développeurs apprécieront l'importance du travail de Linus Torvalds pour implémenter seul toutes ces primitives système dans un projet "amateur". A ceci s'ajoutent les points d'entrée pour les seize appels-système suivants, qui n'étaient pas encore implémentés, et renvoyaient toujours l'erreur ENOSYS :

```
mknod(), mount(), umount(),rename(), ftime(), ulimit(), ustat(), ptrace(),
acct(), phys(), lock(), mpx(), stty(), gtty(), prof(), break().
```

Au début, les passionnés de Linux communiquaient par l'intermédiaire du groupe Usenet comp.os.minix. Cela agaça rapidement les supporters ardents de Minix qui voyait d'un mauvais oeil ce nouveau système prendre chaque jour plus d'importance, et fin janvier 1992, une discussion enflammée entre Linus Torvalds et Andrew Tanenbaum eut lieu. Ce dernier, à vrai dire d'assez mauvaise foi, déclara dans un moment d'irritation que Linux était un système déjà obsolète du fait de sa mauvaise conception (Linux était basé sur un noyau monolithique, d'un seul bloc, alors que Minix était conçu comme un ensemble de micro-noyaux dialoguant entre eux). Piqué au vif, Linus Torvalds répliqua en mettant en avant tous les défauts de Minix, et une conversation haute en couleurs s'ensuivit. Quelques extraits :

- Andrew Tanenbaum : «Les micro-noyaux ont d'ors et déjà gagné […] Linux est un système monolithique, c'est un bond en arrière dans les années soixante-dix. C'est comme prendre un programme écrit en langage C et qui fonctionne, et le réécrire en Basic.»
- Linus Torvalds : «Vous utilisez [le fait d'être professeur] comme une excuse pour les limitations de Minix ? Désolé mais j'ai encore plus d'excuses et Linux bat déjà allégrement Minix. Sans compter que les meilleures portions de code de Minix semblent avoir été écrites par Bruce Evans.»
- Andrew Tanenbaum : «Je maintiens que la conception d'un noyau monolithique en 1991 est une erreur fondamentale. Soyez content de ne pas être mon étudiant ; vous n'auriez pas eu une bonne note pour un tel travail :-) […] Écrire un système d'exploitation limité au 386 en 1991 ne vous aurait rapporté qu'un 'F'.»
- Linus Torvalds : «Vous prétendez écrire Minix en guise de hobby ? Qui fait de l'argent avec Minix ? Qui donne Linux gratuitement ?»

La conséquence de cette guérilla virtuelle fut la création en mars 1992 du groupe Usenet comp.os.linux.

Enfance

Le développement du noyau Linux a progressé assez lentement au départ, puisque ce n'est qu'en mars 1994 que Linus Torvalds le jugea suffisamment complet et stable pour être doté d'un numéro 1.0. Ce noyau était assez limité, mais contenait déjà les éléments essentiels d'une station de travail Unix (réseau, gestion de la mémoire, IPC, divers systèmes de fichiers, pilotes de périphériques divers…) Certaines étapes importantes du développement sont résumées dans le tableau suivant :

Noyau	Fonctionnalité ajoutée
0.10	Appel-système mknod() pour créer des noeuds génériques du système de fichiers, et symlink()/readlink() pour créer et suivre des liens symboliques.
	Appels-système mount()/umount() pour attacher des périphériques blocs dans le système de fichiers.
	Gestion complète des signaux (blocage, interception, attente, etc.)
0.11	Chargement à la demande des pages des fichiers exécutables.
0.12	Multiples consoles virtuelles en mode texte.
	Mécanisme de mémoire virtuelle utilisant une partition de swap.
	Appel-système select() permettant le multiplexage d'entrées-sorties.
0.95	Le noyau lance au démarrage un processus /bin/init en mode utilisateur qui sera chargé de toute l'initialisation du système.
0.96a	Apparition du répertoire net/ dans les sources du noyau, et implémentation des sockets BSD.
	Accès aux ports d'entrées-sorties depuis l'espace utilisateur après autorisation avec ioperm()/iopl().
	Appel-système mmap() permettant la projection de fichiers ou de périphériques dans l'espace d'adressage.
0.96c	Système de fichiers extfs (conçu par Rémy Card à l'Université de Jussieu).
0.96c-2	Système de fichiers msdos.
0.97.5	Pseudo système de fichiers /proc permettant de configurer le noyau depuis l'espace utilisateur.
0.99	Apparition des systèmes de fichiers iso9660 (CD-Rom) et NFS.

Dans le noyau 1.0, se trouvent déjà de nombreux pilotes de périphériques, des systèmes de fichiers divers, etc. On peut considérer que la structure générale des noyaux actuels était déjà présente dans le noyau 1.0. La plupart des fonctionnalités - vues depuis l'espace utilisateur - étaient présentes, même si certaines étaient implémentées de manière encore relativement sommaire.

On peut noter également la parution en mars 1994 du premier numéro de la revue américaine "Linux Journal", qui deviendra rapidement incontournable dans la communauté de passionnés du monde entier. Ce premier numéro, qui annonce en première page la sortie imminente du noyau 1.0 contient outre un comparatif - déjà ! - entre Linux et NT, une interview de Linus Torvalds par Robert Young (le futur

fondateur de la compagnie RedHat).

Adolescence

Au bout de quelques semaines, Linus initia une nouvelle branche de développement avec un noyau 1.1.0. À cette occasion, il fut décider de laisser simultanément en circulation deux versions du noyau : une version stable sur laquelle ne seront apportées que les corrections de bogues éventuels, et une version de développement sur laquelle les nouveaux algorithmes, pilotes, fonctionnalités, etc. seront testés. Pour les différencier, la numérotation des versions doit être explicite. Le numéro d'un noyau est composé de trois chiffres successifs :

- un numéro majeur de version
- un numéro mineur de version
- un numéro de mise à jour.

Voici quelques exemples.

Numéro	Majeur	Mineur	Mise à jour
1.0.5	1	0	5
1.1.65	1	1	65
1.2.13	1	2	13
1.3.29	1	3	29
2.0.6	2	0	6
2.1.67	2	1	67
2.2.19	2	2	19
2.3.55	2.	3	55
2.4.21	2	4	21
2.5.51	2	5	51

La règle établie précise que les noyaux stables, ceux que l'on peut se permettre d'utiliser pour des applications de production ont un numéro mineur pair : 1.0.x, 1.2.x, 2.0.x, 2.2.x et 2.4.x. Les noyaux ayant un numéro mineur impair sont uniquement utilisés à des fins de test et de développement. Certaines mises à jour ne sont même pas compilables pour toutes les options disponibles.

Pour un système en production, on n'utilisera donc qu'un noyau stable. Mais les versions stables continuent d'évoluer en parallèle des versions de développement au gré des bogues découverts et corrigés. Il est donc bon pour l'administrateur de suivre l'évolution des noyaux stables pour vérifier si ses machines en utilisation courante ont besoin d'une mise à jour ou non (correction de failles de sécurité par exemple).

Durant le développement du noyau 1.1 au cours de l'année 1994, apparut une grande nouveauté : le support d'architectures autres que le processeur i386 du PC. Les processeurs MIPS d'abord, puis Alpha, 68000, et Sparc ont été peu à peu intégrés dans le noyau Linux.

Une seconde nouveauté importante mise en chantier dans le noyau 1.1 fut l'utilisation de modules chargeables dans le noyau. Il devint alors possible de compiler des pilotes de périphériques ou des systèmes de fichiers, sous forme de fichiers objet externes, et de ne les insérer dans la mémoire du noyau en fonctionnement que lorsque le besoin s'en fait vraiment sentir. Inutile dorénavant de recompiler systématiquement le noyau après chaque installation pour disposer du support pour les périphériques présents, ou d'utiliser un énorme noyau monolithique intégrant tous les pilotes possibles (avec les risques de conflits que cela implique lors des détections du matériel). Les distributions peuvent installer un noyau minimal et y insérer uniquement les modules correspondants aux périphériques détectés.

Le noyau 1.2, version stable intégrant ces évolutions apparut en mars 1995. Jusqu'à présent les seuls utilisateurs réguliers de Linux étaient essentiellement des étudiants, des passionnés d'Unix, des développeurs… mais avec le noyau 1.2 un nouveau public commença à apparaître : celui des

professionnels qui appréciaient le côté libre et gratuit de ce système Unix fonctionnant sur du matériel peu onéreux. Bien entendu, la diffusion de Linux dans les entreprises était encore très confidentielle, mais déjà des administrateurs système installaient "discrètement" des serveurs Mail, Web ou NFS basé sur un noyau Linux. Des projets industriels complets commençaient déjà (surtout aux États-Unis) à se bâtir autour de stations Linux réputées pour leur stabilité et l'accessibilité de tout le système.

Le développement continua sur une version 1.3, puis un an plus tard environ, un nouveau noyau stable fut publié. Ce nouveau noyau avait été considérablement modifié dans sa conception, au point que Linus considéra qu'il était temps de changer de numéro majeur de version.

Maturité

En juin 1996, Linux 2.0 vit le jour. Il supportait entre autres nouveautés l'architecture Power PC, le support des systèmes multi-processeurs (SMP), un nombre de périphériques beaucoup plus étendu, des fonctionnalités réseau très complètes, et le point le plus important : une mascotte ! Le célèbre pingouin Tux dessiné par Larry Ewing fut intégré dans les sources du noyau sous forme de fichier GIF.

Le développement aboutissant au noyau 2.2 fut plutôt long, durant plus de deux ans et demi. Linux 2.2.0 ne vit le jour qu'en janvier 1999. De nombreuses fonctionnalités internes avaient été ajoutées, surtout celles concernant les normes Posix.1b (temps-réel) et Posix.1c (threads). Le passage à la bibliothèque GlibC, en remplacement des précédentes bibliothèques C spécifiques Linux, représenta aussi une étape importante. Comme toujours, de nouveaux pilotes de périphériques firent leur apparition.

On put remarquer également à partir des noyaux 2.2 un élargissement sensible du public concerné par Linux : le développement des environnements graphiques faciles d'accès (Kde, Gnome…), la multiplication des distributions "grand public" simplifiant l'installation du système, et la couverture plus large des périphériques ont rendu moins ardus les premiers contacts avec Linux. Il devint de plus en plus fréquent de rencontrer dans les entreprises des PC fonctionnant sous Linux.

La démocratisation de Linux, passa aussi par un meilleur support des périphériques "multimédia" : carte son, carte vidéo, etc. ainsi qu'une simplification de l'accès aux fonctionnalités graphiques élémentaires à travers le support Frame Buffer offert par le noyau.

Le développement du noyau 2.2 fut ralentit durant quelques temps par une période de flottement, déclenchée par le déménagement de Linus Torvalds, venu s'installer en Californie pour travailler dans une compagnie jusque là assez obscure : Transmeta. Le mystère entretenu pendant un certain temps autour de l'activité de cette société (la création de microprocesseurs génériques) engendra un flot de spéculations diverses sur les intentions de Linus (qui, du fait de son déménagement, était absent d'Internet) et sur l'avenir incertain de Linux.

En septembre 1998, nous pouvons également relever un point important pour la communauté Linux francophone : le premier numéro de notre confrère "Linux Magazine France". D'abord trimestriel puis bimestriel, il deviendra vite mensuel avec le succès que l'on connaît malgré un paysage encombré par de nombreuses publications - souvent éphémères - essayant de profiter de la vague ascendante Linux.

Cycle de développement

Le schéma de développement des noyaux Linux est globalement le suivant :

- Quelques semaines - voire quelques mois - après la publication d'un noyau stable, lorsque les corrections indispensables ont été apportées, et que les rapports de bogue semblent diminuer, Linus Torvalds crée une nouvelle branche avec un noyau de développement sur lequel chacun est libre d'apporter des contributions variées.
- Au bout de quelques mois, lorsque suffisamment de nouveautés ont été proposées et acceptées, Linus décide un gel des fonctionnalités (*features freeze*). À partir de ce moment, on n'ajoute plus de fonctionnalités dans le noyau, mais les développeurs se consacrent à l'implémentation de celles choisies.
- Quand le développement se ralentit, et que les tests semblent se stabiliser, Linus déclare un gel du

code (*code freeze*), étape durant laquelle le code ne doit plus être modifié en profondeur, mais seulement corrigé en fonction des tests des utilisateurs. Assez rapidement, des versions pré- du noyau sont publiées, indiquant que la stabilisation est proche, et que les utilisateurs sont encouragés à tester les nouveautés. Enfin, lorsque les tests apparaissent concluants, la version zéro du noyau stable est publiée.

- On notera qu'au moment de la mise en projet d'un nouveau noyau de développement, Linus Torvalds transmet à une autre personne la charge d'assurer la maintenance du noyau stable : David Weinehall pour les 2.0, Alan Cox pour les 2.2, et Marcelo Tosatti pour les 2.4.

Linus assure la cohérence des sources officielles du noyau de développement, bien qu'il existe des branches parallèles destinées à tester des algorithmes particuliers. Ces noyaux, proposés par des personnalités importantes du développement Linux sont numérotés avec un préfixe identifiant leur auteur : -ac (Alan Cox), -mjc (Michael Cohen), -aa (Andrea Arcangeli), etc. Ils sont en général encore moins stables que les versions de développement officielles, Linus Torvalds ayant tendance à être (beaucoup) plus regardant pour accepter des modifications que les auteurs de ces noyaux expérimentaux. On se gardera donc bien de les utiliser pour des systèmes en production.

Reconnaissance

Le noyau 2.4, apparu en janvier 2001, recelait de nombreuses améliorations en profondeur. Par exemple la gestion de la mémoire virtuelle fut largement modifiée pour éviter des copies inutiles de zones mémoire (buffer recopié dans la mémoire cache du noyau avant écriture sur disque). De même un effort important avait été mené pour améliorer les performances de Linux sur des systèmes multiprocesseurs. En revanche un nombre assez important de problèmes fut mis en évidence lors du déploiement sur une grande échelle de ce noyau. La complexité de certains bogues (liés à la mémoire virtuelle, au périphérique de swap, au système de fichiers virtuel, etc.) est telle qu'ils ne se manifestent qu'après un nombre très élevé de tests et sur des plates-formes les plus diverses. Malheureusement, la plupart des utilisateurs de Linux mettent peu d'entrain à essayer les noyaux -pre qui devraient justement servir à éliminer les derniers bogues. Il fallut par exemple attendre les noyaux 2.4.11 pour que les rapports de bogues liés à la mémoire virtuelle et au périphérique de swap disparaissent.

La règle disant qu'aucune nouvelle fonctionnalité ne doit être apportée dans un noyau de version stable (paire) a été violée à plusieurs reprises par Linus Torvalds dans la série des 2.4. Le cas le plus visible fut probablement l'ajout dans le noyau 2.4.15 du système de fichiers ext3. Jusqu'alors, il n'était disponible que sous forme de patch, maintenu par Andrew Morton, mais Linus le jugea suffisamment stable et utile pour l'intégrer dans le noyau 2.4.15-pre2. Il faut reconnaître que ce système de fichiers dispose d'un mécanisme de journalisation, ce qui permet des redémarrages rapides après un arrêt intempestif (indispensable pour des serveurs demandant un taux de fonctionnement le plus élevé possible), et cohabite parfaitement avec son prédécesseur, le système de fichiers ext2. Signalons aussi que plusieurs distributions avaient déjà intégré le patch ext3 dans les premiers noyaux 2.4 qu'elles proposaient. Remarquons également que le grand concurrent d'ext3, le système de fichiers reiserfs avait été introduit dans le noyau 2.4.1 alors qu'il était absent du 2.4.0.

Le noyau 2.4 marqua l'orientation de Linux vers les technologies offertes par de nouveaux périphériques : bus USB, interface IEEE 1394, tuners, webcam et cartes d'acquisition vidéo, interfaces ADSL… De même on vit apparaître le support pour certains dispositifs liés aux systèmes industriels ou embarqués (systèmes de fichiers journalisés, cartes flash, support d'Eproms, disques RAID, etc.)

Conclusion.

Au moment où je rédige ces lignes, le dernier noyau stable de la ligne 2.4 est le vingt-et-unième du nom. De nombreuses discussions entre développeurs concernent déjà les projets d'expérimentation qu'ils comptent mener dans le futur noyau 2.7. Le noyau de développement en cours était numéroté 2.5.75, mais Linus Torvalds vient de publier depuis deux jours un noyau 2.6.0-test1 pour inciter les utilisateurs à commencer les essais avec cette version "presque" finie. Espérons que les tests seront menés par le plus grand nombre possible de personnes, afin de raccourcir le temps de stabilisation de cette nouvelle mouture de Linux.

Quoiqu'il en soit, ce noyau incorporera des améliorations sensibles au niveau des performances : nouvel ordonnanceur, amélioration de la mémoire virtuelle, points de préemption pour les processus temps-réel souple Posix.1b, support des threads Posix.1c amélioré avec l'aide de la nouvelle bibliothèque NPTL, etc. Le nombre de périphériques reconnus augmentera évidemment, en intégrant dans le noyau officiel des modules disponibles de manière indépendante pour le noyau 2.4.

Il est indéniable que Linux a atteint une maturité certaine. Il est capable d'équiper une gamme de systèmes extraordinairement variée, depuis le micro-système tenant sur une barrette SIMM 72 broches (voir le projet µC-Linux), jusqu'aux "fermes" composées de plusieurs dizaines ou centaines de stations de calcul à hautes performances destinées au traitement de données scientifiques ou à la production d'images de synthèse. L'évolution des fonctionnalités offertes est devenue plus lente à mesure que le système est plus complet. Le travail des développeurs centraux est à présent d'optimiser des procédures très complexes du noyau, tandis qu'un nombre croissant de programmeurs "satellites" réalisent des pilotes de périphérique les plus variés.

Comme tous les passionnés de Linux, nous resterons donc en attente des évolutions à venir, qu'elles concernent le développeur à travers de nouvelles interfaces de programmation, l'administrateur par des possibilités de configuration pointue du système ou le simple utilisateur par une gestion toujours améliorée d'un nombre croissant de périphériques.

Bibliographie

- "The creation of the UNIX operating system" - http://www.bell-labs.com/history/unix/.
- "Operating Systems: Design and Implementation" - Andrew S. Tanenbaum & Albert S. Woodhull - Second Edition - Prentice Hall 1997.
- "Il était une fois Linux" - Linus Torvalds & David Diamond - Osman Eyrolles Multimédia 2001.
- Les noyaux Linux : ftp://ftp.kernel.org/pub/linux/kernel/.
- Résumés de la liste de discussion linux-kernel : http://kt.zork.net/kernel-traffic/latest.html.

Christophe Blaess
http://www.blaess.fr/christophe/