Le noyau Linux 0.01

Le but de ce document est d'étudier la toute première version du noyau Linux : le noyau 0.01. On donnera cependant quelques indications sur l'évolution du noyau, mais ce n'est pas le but essentiel.

I- Le noyau Linux à étudier

1) Noyaux et distribution

Le système d'exploitation Linux est un gros programme et, comme tel, difficile à appréhender. Mais, en fait, il faut distinguer plusieurs niveaux. Ce que l'on entend par Linux, le plus souvent, concerne une distribution, telle que Red Hat, Suse, Mandrake, Debian, Ubuntu... Une distribution comprend le système d'exploitation proprement dit, plus exactement le noyau, les utilitaires traditionnellement associés à Unix (un éditeur de texte, un compilateur C...), l'interface graphique X Window System1 et beaucoup d'autres applications utilisateur.

Notre but, dans ce TP, est uniquement d'étudier le noyau Linux.

2) Noyau minimal

Même pour le seul noyau, les sources ont une taille non négligeable : 58 Mo pour la version 2.2.18, par exemple. Ceci s'explique, en particulier, par le grand nombre de périphériques pris en compte. Il est évidemment inutile de s'occuper de tous les périphériques et de tous les types de tels périphériques du point de vue pédagogique. Nous étudierons donc un noyau minimal, ne mettant pas nécessairement toutes les activités en application et ne contenant que quelques périphériques à titre d'exemple. Le noyau Linux 0.01 est intéressant du point de vue pédagogique. Il ne concerne que le micro-processeur Intel 80386, il ne prend en compte qu'un nombre très limité de périphériques, qu'un seul système de fichiers et qu'un seul type d'exécutables, mais ces défauts pour l'utilisateur deviennent un avantage lorsqu'on veut étudier les sources en entier.

3) Obtention des sources

L'ensemble des sources des noyaux de Linux, depuis le tout premier jusqu'au dernier, se trouve sur le site : http://ftp.cdut.edu.cn/pub/linux/kernel/history/. Nous étudions, dans une première étape, les sources du tout premier noyau, nettement moins imposant et contenant évidemment l'essentiel. Les sources du noyau 0.01 se trouvent également à l'adresse suivante : http://www.kernel.org/pub/linux/kernel/Historic/.

4) Programmation sous Linux

L'obtention du noyau Linux et la compréhension des fichiers source nous entraînent à faire un détour par la programmation sous Linux. A priori, nous ne devrions rien à avoir à dire sur la programmation. Que ce soit Linux ou un autre système d'exploitation, nous devrions avoir des

sources portables. En fait, ce n'est pas le cas pour des raisons historiques dues à certains choix initiaux de Linus Torvalds, jamais remis en cause ensuite.

Les sources reposent sur l'utilisation de la commande **make** (un outil pour gérer les codes source répartis sur de nombreux fichiers), sur des fichiers en langage C non standard (il s'agit du langage C de GCC avec quelques utilisations de particularités de ce compilateur), sur des fichiers en langage d'assemblage, pour Intel 80386 pour ce qui nous concerne et, enfin, sur des scripts **bash** modifiés. La syntaxe du langage d'assemblage ne suit pas celle de l'assembleur MASM de Microsoft (qui fut longtemps la référence) mais celle qui suit un style dit **AT&T**.

5) Versions du noyau Linux

Linux distingue les noyaux stables des noyaux en développement avec un système de numérotation simple. Chaque version est caractérisée par trois nombres entiers séparés par des points. Les deux premiers identifient la version, le troisième la parution (release en anglais). Un second numéro pair identifie un noyau stable ; impair, il dénote un noyau en développement. Les nouvelles parutions d'une version stable visent essentiellement à corriger des erreurs signalées par les utilisateurs ; les algorithmes principaux et les structures de données du noyau ne sont pas modifiés. Les versions en développement, en revanche, peuvent différer les unes des autres de façon importante. Les développeurs du noyau sont libres d'expérimenter différentes solutions qui peuvent éventuellement conduire à des changements rigoureuses du noyau. La notation de la version 0.01 ne suit pas le principe de la numérotation décrit ci-dessus avec trois nombres entiers séparés par des points. Linus Torvalds voulant seulement indiquer que nous sommes très loin d'une version stable, qui porterait le numéro 1.0, il a choisi le numéro 0.01 pour laisser de la place pour encore d'autres versions intermédiaires.

II- Les sources du noyau 0.01

1) Vue d'ensemble sur l'arborescence

Les sources du noyau 0.01 occupent 230 Ko, comportent 11 dossiers et 88 fichiers. Le premier niveau de l'arborescence du code source est simple :

```
/boot
/fs
/include
/init
/kernel
/lib
/mm
/tools
```

Elle s'inspire de l'arborescence du code source de **Minix**. Nous avons vu, dans le cours, que les systèmes d'exploitation se composent de quatre parties principales : le gestionnaire des processus, le gestionnaire de la mémoire, le gestionnaire des fichiers et le gestionnaire des périphériques d'entrée-sortie. Les procédures des bibliothèques standard C utilisées par le noyau (**open()**, **read()**,...) se trouvent dans le répertoire lib (pour LIBrary). Les répertoires mm

(pour Memory Management) et fs (pour File System) comportent le code du gestionnaire de mémoire et du gestionnaire de fichiers, respectivement.

Le répertoire include contient les fichiers d'en-têtes nécessaires au système Linux. Il sert à la constitution du noyau, mais également à la programmation Linux une fois le noyau constitué.

Les trois derniers répertoires contiennent les outils de mise en place : le répertoire boot permet de démarrer le système ; le répertoire init d'initialiser le système (il ne contient que la fonction principale **main()**) ; le répertoire tools permet de construire le noyau.

2) L'arborescence détaillée

Le répertoire boot

Pour le noyau 0.01, ce répertoire ne contient que deux fichiers en langage d'assemblage : boot.s et head.s. Voici les fonctions de ces deux fichiers :

- boot.s contient le code du secteur d'amorçage de la disquette à partir de laquelle on démarre Linux, de l'initialisation des périphériques de l'ordinateur, de la configuration de l'environnement pour pouvoir passer en mode protégé des micro-processeurs Intel. Il donne ensuite la main au code **startup_32().**

- head.s permet de configurer l'environnement d'exécution pour le premier processus Linux (processus 0) puis de passer à la fonction **start_kernel()**, qui est la fonction principale du code C.

Le répertoire init

Le répertoire init contient un seul fichier : le fichier main.c qui, comme son nom l'indique, contient la fonction principale du code C. Cette fonction initialise les périphériques (en mode protégé) puis fait appel au processus 1.

Le répertoire include

Le répertoire include est évidemment le répertoire par défaut des fichiers d'en-têtes C qui ne font pas partie de la bibliothèque C standard. Il s'agit des fichiers d'en-têtes qui sont propres à Linux (propres à Unix pour la plupart) ou, faisant partie de la bibliothèque C standard, qui doivent être implémentés suivant le système. Ces fichiers se trouvent soit dans le répertoire lui-même, soit dans l'un des trois sous-répertoires :

- asm contient des fichiers d'en-têtes dont le code est écrit en langage d'assemblage ;
- linux contient des fichiers d'en-têtes propres à Linux (n'existant pas sur les autres distributions Unix) ;
- sys est un sous-répertoire classique d'Unix, contenant les fichiers d'en-têtes concernant le système.

Le répertoire lui-même contient d'abord des fichiers d'en-têtes faisant partie de la bibliothèque standard C mais qu'il faut implémenter suivant le système :

- ctype.h (pour Character TYPEs) permet le traitement des caractères en distinguant des classes de caractères (chi_re, alphabétique, espace...) ;

- errno.h (pour ERRor NumerO) permet d'associer un numéro à des constantes symboliques représentant les erreurs rencontrées ;
- signal.h définit les valeurs de code d'un ensemble de signaux ;
- stdarg.h (pour STandarD ARGument) définit des macros permettant d'accéder aux arguments d'une fonction, telle la fonction **printf()**, acceptant une liste variable d'arguments;
- stddef.h (pour STandarD DEFinitions) contient un certain nombre de définitions standard (sic) ;
- string.h contient des fonctions permettant de manipuler les chaînes de caractères ;
- time.h concerne les calculs sur l'heure et la date.

Il contient ensuite des fichiers d'en-têtes propres à Unix :

- a.out.h contient le format propre au type d'exécutable a.out, qui était le plus utilisé avant l'arrivée du format ELF ;
- const.h contient diverses valeurs de constantes ;
- fcntl.h contient les fonctions permettant de manipuler les descripteurs de fichiers ;
- termios.h contient les constantes et les fonctions concernant les terminaux ;
- unistd.h (pour UNIx STandarD) contient les constantes et les fonctions standard d'Unix ;
- utime.h (pour User TIME) permet de changer la date et l'heure d'un nœud d'information.

Le sous-répertoire asm contient quatre fichiers :

- io.h (pour Input/Output) contient la définition des macros, en langage d'assemblage, permettant d'accéder aux ports d'entrée-sortie ;
- memory.h contient la définition de la macro **memcpy()** ;
- segment.h contient la définition des fonctions en ligne d'écriture et de lecture d'un octet, d'un mot ou d'un mot double ;
- system.h contient la définition de fonctions nécessaires à l'initialisation.

Le sous-répertoire linux contient neuf fichiers :

- config.h contient les données nécessaires au démarrage du système (concernant la capacité mémoire et le disque dur) ;
- fs.h (pour File System) contient les définitions des tableaux de structures pour les fichiers ;
- hdreg.h (pour Hard Disk REGisters) contient des définitions pour le contrôleur de disque dur de l'IBM PC-AT;
- head.h contient des constantes nécessaires pour le fichier head.s ;
- kernel.h contient la déclaration de fonctions nécessaires pour le mode noyau (comme la fonction **printk()**) ;
- mm.h (pour Memory Management) contient la déclaration de fonctions de manipulation de la mémoire ;
- sched.h (pour SCHEDuler) contient la défition des structures et la déclaration des fonctions nécessaires à la manipulation des processus ;
- sys.h (pour SYStem call) contient la déclaration des appels système ;
- tty.h contient la définition de structures et la déclaration de fonctions concernant le terminal (tty pour TeleTYpe), nécessaires pour le fichier tty_io.c .

Le sous-répertoire sys contient cinq fichiers :

- stat.h contient la déclaration des fonctions renvoyant les informations sur les fichiers ;
- times.h contient la déclaration de la fonction renvoyant le nombre de tops d'horloge écoulés depuis le démarrage du système ;

- types.h contient la définition d'un certain nombre de types ;
- utsname.h contient la déclaration de la fonction donnant le nom et des informations sur le noyau ;
- wait.h contient la déclaration des fonctions permettant de suspendre l'exécution du processus en cours jusqu'à ce un processus fils se termine ou qu'un signal soit envoyé.

Le répertoire kernel

Il contient dix-sept fichiers, outre le fichier Makefile :

- asm.s contient les routines de service de la plupart des 32 premières interruptions, c'est-à-dire de celles qui sont réservées par Intel ;
- console.c contient les paramètres, les variables et les fonctions nécessaires à l'affichage sur le moniteur (nécessite les structures définissant un terminal) ;
- exit.c contient les fonctions nécessaires pour quitter un processus autrement que par return ;
- fork.c contient les fonctions nécessaires pour créer un processus fils ;
- hd.c contient le pilote du disque dur ;
- keyboard.s contient la routine de service associée à IRQ1, c'est-à-dire à l'interruption matérielle provenant du clavier ;
- mktime.c contient la fonction permettant de transformer la date exprimée en secondes depuis 1970 en année, mois, jour, heure, minute et seconde ;
- panic.c contient une fonction utilisée par le noyau pour indiquer un problème grave ;
- printk.c contient une fonction analogue à la fonction printf() du langage C mais qui peut être utilisée par le noyau ;
- rs_io.c contient la routine de service associée aux interruptions matérielles des ports série (rs rappelant la norme RS232) ;
- sched.c contient le séquenceur (scheduler en anglais) qui permet de changer de processus pour rendre le système d'exploitation multi-tâches ;
- serial.c contient l'implémentation de deux fonctions servant aux ports série ;
- sys.c contient la définition de beaucoup de fonctions de code sys_XX() d'appels système ;
- system_call.s contient du code en langage d'assemblage permettant d'implémenter les appels système ;
- traps.c contient le code en langage C des routines de service associées aux 32 premières interruptions, c'est-à-dire celles réservées par Intel ;
- tty_io.c contient les fonctions nécessaires au fonctionnement du terminal ;
- vsprintf.c contient le code permettant de définir à la fois les fonctions printk() et printf() ;

Le répertoire lib

Le répertoire lib contient onze fichiers, outre le fichier Makefile :

- _exit.c contient la définition de la fonction associée à l'appel système de terminaison d'un processus **_exit()** ;
- close.c contient la définition de la fonction associée à l'appel système de fermeture d'un fichier **close()** ;
- ctype.c contient la définition du tableau de définition des types de chacun des 256 caractères (majuscule, chiffre...) ;
- dup.c contient la définition de la fonction associée à l'appel système **dup()** ;
- errno.c contient la déclaration de la variable errno ;
- execv.c contient la définition de la fonction associée à l'appel système **execv()** ;

- open.c contient la définition de la fonction associée à l'appel système d'ouverture d'un fichier **open()** ;
- setsid.c contient la définition de la fonction associée à l'appel système **setsid()** ;
- string.c contient des directives de compilation ;
- wait.c contient la définition de la fonction associée à l'appel système **wait()** ;
- write.c contient la définition de la fonction associée à l'appel système d'écriture sur un fichier **write()**.

Le répertoire fs

Le répertoire fs contient dix-huit fichiers, outre le fichier Makefile :

- bitmap.c contient le code permettant de gérer les tables de bits d'utilisation des nœuds d'information et des blocs ;
- block_dev.c contient le code permettant de gérer les périphériques bloc ;
- buffer.c contient le code permettant de gérer l'antémémoire de blocs ;
- char_dev.c contient le code permettant de gérer les périphériques caractère ;
- exec.c contient le code permettant d'exécuter un nouveau programme ;
- fcntl.c contient le code permettant de manipuler les descripteurs de fichiers ;
- file_dev.c contient les fonctions d'écriture et de lecture dans un fichier ordinaire ;
- file_table.c contient la déclaration de la table des fichiers ;
- inode.c contient la déclaration de la table des nœuds d'information en mémoire ainsi que les fonctions permettant de la gérer ;
- ioctl.c contient la déclaration de la table ioctl[ ] et quelques fonctions associées ;
- namei.c (pour NAME I-node) contient les fonctions permettant de nommer les fichiers ;
- open.c contient les fonctions permettant d'ouvrir et de changer les droits d'accès d'un fichier ;
- pipe.c permet de mettre en place les tubes de communication ;
- read_write.c contient les fonctions permettant de se positionner, de lire et d'écrire sur un fichier ;
- stat.c contient les fonctions permettant d'obtenir des informations sur un fichier ;
- super.c contient les définitions et les fonctions concernant les super-blocs ;
- truncate.c contient les fonctions permettant d'effacer un fichier ;
- tty_ioctl.c contient les fonctions permettant de paramétrer un terminal.

Le répertoire mm

Le répertoire mm contient deux fichiers, outre le fichier Makefile :

- memory.c contient les fonctions concernant la gestion des pages ;
- page.s contient la routine de service de l'interruption matérielle concernant le défaut de page.

Le répertoire tools

Le répertoire tools contient un seul fichier : build.c. Il s'agit d'un programme C indépendant qui permet de construire l'image du noyau.