Pilotes de périphériques pour le noyau Linux — Introduction

MS SE SE758

Guillaume Duc
guillaume.duc@telecom-paris.fr
2021–2022

Plan

Objectifs pédagogiques

- Être capable d'écrire un pilote de périphérique pour le noyau Linux et d'intégrer ce pilote dans les sources du noyau
  - Spécificités du développement pour le noyau (espace noyau vs. espace utilisateur, outils disponibles, débogage. . .)
  - Abstractions utilisées dans le noyau pour les pilotes (modèle de périphérique, frameworks pour la communication avec les applications. . .)
  - Mécanismes d'interaction avec le matériel (utilisation des bus de communication, interruptions. . .) et savoir les utiliser
  - Interactions avec le noyau (gestion mémoire, ordonnancement, attente. . .)
  - Mécanismes de communication avec les applications (appels systèmes, transfert de données. . .)

Travaux pratiques 2021–2022

- L'objectif des séances de travaux pratiques est de développer un pilote complet pour un périphérique (un accéléromètre ADXL345 connecté sur un bus I^2C)
- Utilisation de `qemu` pour simuler un système embarqué à base de processeur ARM Cortex-A
  - Pas besoin de maquettes de TP (travail à la maison, passage en distanciel...)
  - Plus simple à déboguer

Travaux pratiques 2021–2022

- Question : pourquoi faire les TP sur une machine différente (carte de TP ou machine simulée par QEMU) plutôt que de cibler directement votre machine (PC de la salle de TP ou PC personnel) ?

Modalités d'évaluation

- Note finale basée sur
  - 50 % rendus de TP à la fin de la semaine (le code de votre pilote dans votre dépôt git)
  - 50 % examen écrit (23/03/2022 10h15-11h45)
    - Questions type QCM et/ou questions ouvertes
    - Seul document autorisé : une page A4 avec ce que vous voulez inscrit dessus
    - Tout autre document, support de cours, ordinateur, téléphone, calculatrice... interdit

Dernières remarques

- N'hésitez pas à poser des questions pendant le cours et les TP !
- N'hésitez pas à signaler toute erreur que vous trouverez dans le cours, les textes des TP, etc.

Plan

Le noyau Linux

- *Stricto sensu*, Linux désigne juste le noyau
- C'est un système d'exploitation généralise, destiné à s'exécuter sur un très grand nombre d'architectures et de systèmes (embarqué, ordinateur personnel, serveur, super-calculateur)

Le noyau Linux

Bref historique

- En 1991, alors étudiant à Helsinki (Finlande), *Linus Torvalds* (âgé de 21 ans), commence à écrire un petit noyau destiné à fonctionner sur un processeur 80386
- Le 25 août 1991, il annonce son projet sur le newsgroup `comp.os.minix`

```
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 protabl
(uses 386 task switching etc), and it probably never will support anything other than
AT-harddisks, as that's all I have :-(.
```

Le noyau Linux

Bref historique

- Septembre 1991 : version 0.01 publiée sur un serveur FTP, environ 10.000 lignes de code
- Décembre 1992 : version 0.99 distribuée sous licence GNU GPL
- Grâce à la démocratisation d'Internet dans la début des années 90, Linux fédère une communauté grandissante de développeurs
- Mars 1994 : Version 1.0.0 (175.000 lignes de code)
- Janvier 1999 : Version 2.2.0 (1.800.000 lignes de code)
- Janvier 2001 : Version 2.4.0 (3.400.000 lignes de code)
- Décembre 2003 : Version 2.6.0 (6.000.000 lignes de code)

Le noyau Linux

Bref historique

- Juillet 2011 : Version 3.0 (abandon de la numérotation 2.x avec x pair pour la branche stable et impair pour la branche de développement)
- Avril 2015 : Version 4.0
- Mars 2019 : Version 5.0
- Mars 2021 : Version 5.11
  - ~29 millions de ligne de code
  - Environ 14.000 contributeurs

Le noyau Linux

Matériel nécessaire minimal

- Processeur 32 bits ou plus (voir *Embeddable Linux Kernel Subset* [1] pour un support minimal des processeurs 16 bits)
  - Architectures supportées (v5.9) : DEC Alpha, ARC, ARM, ARM 64, C6x, C-SKY, Renesas H8, Qualcomm Hexagon, Intel IA-64, Freescale 68k, Xilinx Microblaze, MIPS, Andes NDS32, Altera Nios II, OpenRISC, HP PA-RISC, PowerPC, RISC-V, IBM System/390, SuperH, SPARC, x86, x86-64, Tensilica Xtensa
- *Memory Management Unit (MMU)* (voir µClinux pour les systèmes sans MMU)
- Plusieurs Mo de RAM (le minimum dépend de la configuration du noyau, aux alentours de 4–8 Mo, mais il faut ensuite compter les exécutables)

Les composants logiciels minimums d'un système Linux

- Le noyau Linux a besoin d'un certain nombre de composants logiciels supplémentaires (il ne peut pas faire grand chose seul)
- Un mécanisme pour lui permettre d'identifier le matériel présent (processeur, mémoire, périphériques...)
  - Arbre des périphériques (*device tree*) ou structure de données passée par le *bootloader* (ATAG...)
  - Description statique de la plate-forme compilée avec le noyau
  - Mécanisme automatique pour identifier les périphériques (tables ACPI, énumération des périphériques USB...)

Les composants logiciels minimums d'un système Linux (suite)

- Un chargeur d'amorçage (*bootloader*)
  - Initialisation de base de la plate-forme (arbre d'horloge, caches, contrôleur de DRAM...)
  - Chargement du noyau depuis un support de stockage de masse (disque dur, mémoire flash...)
  - Mise en place en mémoire des structures complémentaires pour le démarrage du noyau (arbre des périphériques, disque mémoire initial (*initrd*...))

Les composants logiciels minimums d'un système Linux (suite)

- Un système de fichiers racine
  - Soit un système minimal en RAM mis en place à partir d'une image fournie au noyau lors du démarrage (`initrd` ou `initramfs`)
  - Soit un système de fichiers plus complet monté depuis un support de stockage de masse ou depuis le réseau
  - Remarque : dans beaucoup de systèmes les deux méthodes ci-dessus sont utilisées l'une après l'autre (sera détaillé dans la suite)

Les composants logiciels minimums d'un système Linux (suite)

- Un processus `init` (lancé depuis `/sbin/init`)
  - Premier processus exécuté par le noyau
  - Ancêtre de tout les autres processus
  - Une fois ce processus lancé, la phase de démarrage du noyau est terminée
  - C'est ce processus qui va être en charge de la suite du démarrage du système

Les autres composants usuels

- Fichiers de configuration et exécutables pour le démarrage (`SysV init` ou `systemd`)
- Bibliothèques partagées (au minimum `libc`)
- Shell et utilitaires classiques (`sh`, `mount`, `test`...)
  - La plupart des bibliothèques et des outils essentiels proviennent du projet GNU (*GNU's Not Unix*), d'où le nom GNU/Linux
  - C'est moins vrai dans le domaine de l'embarqué (Newlib, BusyBox...)
- Démons (`udev`, `sshd`...)
- Une console (clavier/écran, série)

Plan

Références I

[1] Embeddable linux kernel subset.
https://github.com/jbruchon/elks, 2020.