

R3.05 - Programmation Système

Introduction

Cyril Grelier

Université de Lorraine - IUT de Metz



La **Programmation Système** consiste à écrire des programmes qui font partie ou interagissent avec le **système d'exploitation (OS - *Operating System*)**.

L'**OS** est un ensemble de logiciels qui contrôlent l'utilisation des **ressources matérielles** et fournissent une **interface standard** aux logiciels applicatifs.

Rôles principaux :

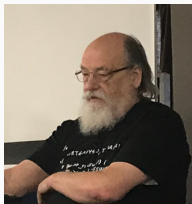
- Gestion du **processeur** et des **processus**.
- Gestion de la **mémoire**.
- Accès aux **périphériques**.
- Gestion du **système de fichiers**.
- Gestion du **réseau**.

- Au démarrage
 - L'OS est le premier logiciel chargé
 - Lance les processus système (sessions, environnement graphique, services)
- On retrouve des OS partout :
 - PC, serveurs, smartphones, systèmes embarqués, supercalculateurs, wearables, ...
- Exemples d'OS :
 - Unix, GNU/Linux, Windows, macOS, Android, ...
- Typologie
 - *Mono- vs Multi-utilisateur*
 - *Mono- vs Multi-tâche (temps partagé)*

Ce cours se concentre sur les systèmes **Unix/Linux** : multi-utilisateur, multi-tâche préemptif

Préemptif : le système peut interrompre une tâche en cours pour en exécuter une autre

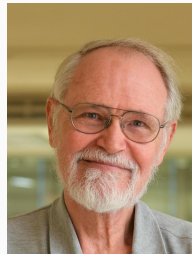
- Développé à partir de 1969 aux **Bell Labs** (AT&T)
- Conçu pour être simple, portable, multi-tâche et multi-utilisateur
- A inspiré de nombreux systèmes modernes (BSD, Linux, macOS, ...)



Ken Thompson



Dennis Ritchie



Brian Kernighan

GNU (*GNU's Not Unix*)

- Lancé en 1983 par [Richard Stallman](#) (FSF)
- Outils : **GCC**, **glibc**, binutils, coreutils, **bash**, ...



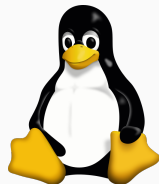
Logo GNU



Richard Stallman

Linux

- Noyau démarré en 1991 par [Linus Torvalds](#)
- Inspiré de Minix/Unix, libre (licence **GPLv2**)



Tux, mascotte Linux



Linus Torvalds

GNU = outils/logiciels – *Linux* = noyau (ordonnancement, mémoire, pilotes, ...)

GNU/Linux = système complet.

Distributions Linux

- **Bureau** : Ubuntu, Linux Mint, Fedora, Pop!_OS, Manjaro, Elementary OS...
- **Serveur** : Debian, AlmaLinux, CentOS Stream, Rocky Linux...
- **Embarqué** : Raspberry Pi OS, OpenWRT, Yocto...
- **Sécurité** : Kali Linux, BlackArch, Parrot OS...
- **Spécialisées** : Arch Linux, Alpine Linux, NixOS, Tails...

Toutes reposent sur le noyau Linux et des interfaces conformes à POSIX.



Voir [liste des distributions](#)

Interface POSIX et bibliothèque standard du C (libc)

POSIX

Portable Operating System Interface

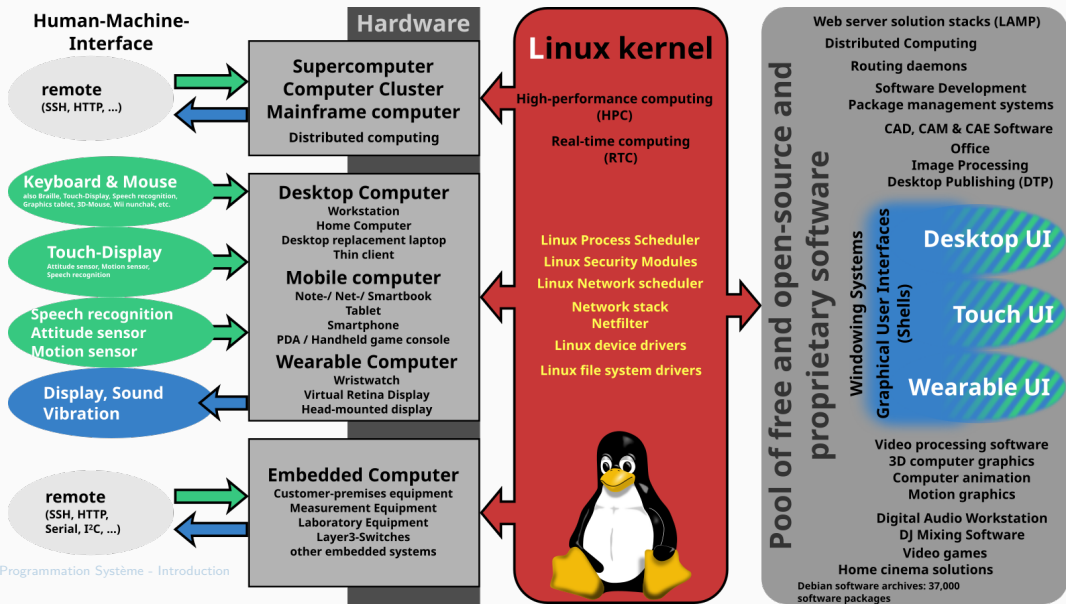
- Norme définie par l'IEEE
- Assure la portabilité des programmes entre systèmes Unix
- Spécifie une interface standard (API) pour les appels système :
 - Gestion des processus : fork, exec, ...
 - Gestion des fichiers : open, close, ...
 - Entrées/sorties : read, write, ...
 - Signaux, threads, etc.
- Décrit des interfaces, pas leur implémentation

La libc

Bibliothèque standard du C (C Standard Library)

- Implémente les interfaces POSIX et les fonctions normalisées du langage C
- Sert d'intermédiaire entre les programmes et le noyau
- Regroupe :
 - Fonctions POSIX :
fork, open, write, ...
 - Fonctions de la norme C :
printf, malloc, ...
- Sous Linux : implémentée par la glibc

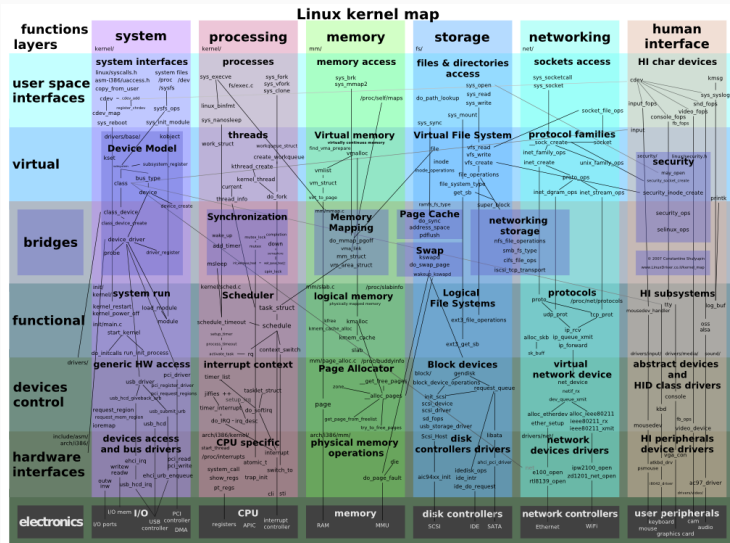
Le noyau Linux : Cœur du système



Carte du noyau (Kernel map)

Sous-systèmes principaux :

- System
- Processing
- Memory
- Storage
- Networking
- Human Interface



Exemple : `strace cat fichier.txt`

1. L'utilisateur saisit `cat fichier.txt`.
2. Le **shell** fork puis `exec /bin/cat`.
3. `cat` utilise la **libc** → syscalls : `openat`, `read`, `write`, `close`.
4. Le noyau vérifie droits, localise le fichier, alloue des structures et renvoie un **descripteur**.
5. Les données sont lues puis écrites vers la sortie standard.

`strace` permet d'afficher tous les appels système effectués par un programme.