Ecole Nationale
Supérieure
de l'Electronique
ENSEA et de ses Applications

Linux Embarqué

Séance 1 : Introduction, Fichiers

Laurent Fiack
Bureau D212 – laurent.fiack@ensea.fr

Les TD/TP

- 8h de cours
- Cours/TD en salle BYOD, ramenez vos PC (un pour 2 ?)
- 3 séances de TP (c'est peu, on arrivera peut-être pas au bout)
  - Démarrage cible embarquée
  - Mise en place de l'environnement de travail
  - Structure de driver
  - Écriture d'un driver

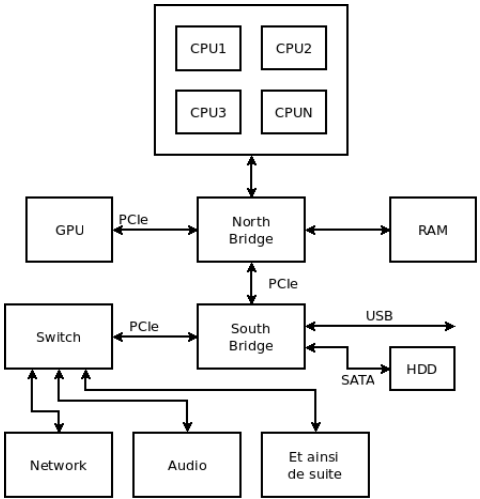
Menu du jour

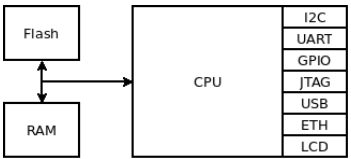
Linux embarqué

Pourquoi Linux dans l'embarqué ?

- Code source libre
  - Compilable sur plusieurs plateformes (en particulier ARM)
  - Rien à payer
  - Écriture de drivers relativement bien documentée
  - Facile (hum) à porter
- Beaucoup de soft disponibles (openCV...)
- Gestion de temps-réel (mou)
- Léger (quelques MB pour le noyau et quelques libs)
- Toutes les CPU "applicatifs" fournissent une distribution Linux

Architectures : PC vs Système embarqué



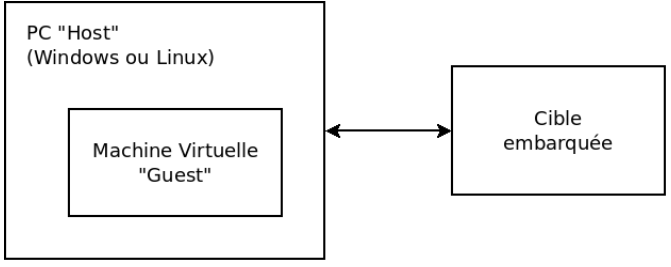


Différence avec Linux sur bureau

- Moins de mémoire, moins de puissance...
- Généralement pas d'interface graphique
  - Liaison série
  - SSH
- Bootloader différent
  - UBoot vs Grub
  - Possibilité de charger par le réseau
  - Chargement dans la RAM
- On ne compile pas sur la cible
  - Cross-compilation
  - `gcc-arm-linux-gnueabihf`

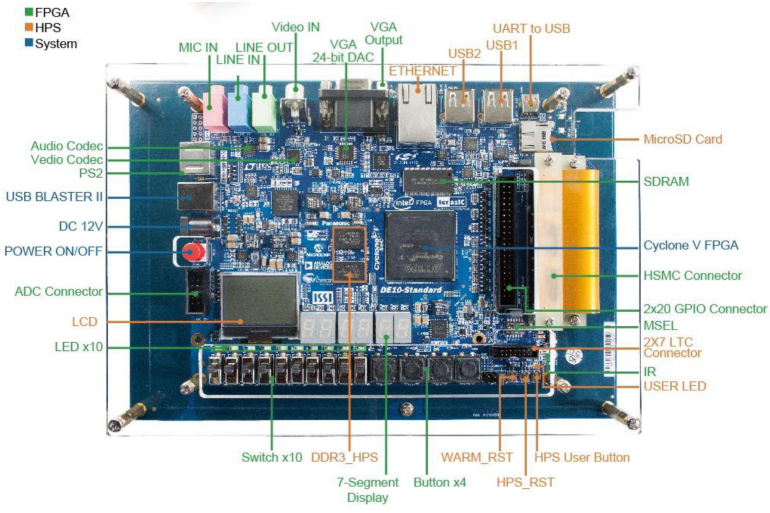
Cross-compilation

- Souvent dans une machine virtuelle
  - Permet de maîtriser les versions des compilateurs et des librairies
  - On utilise une VM même si on est déjà sous Linux

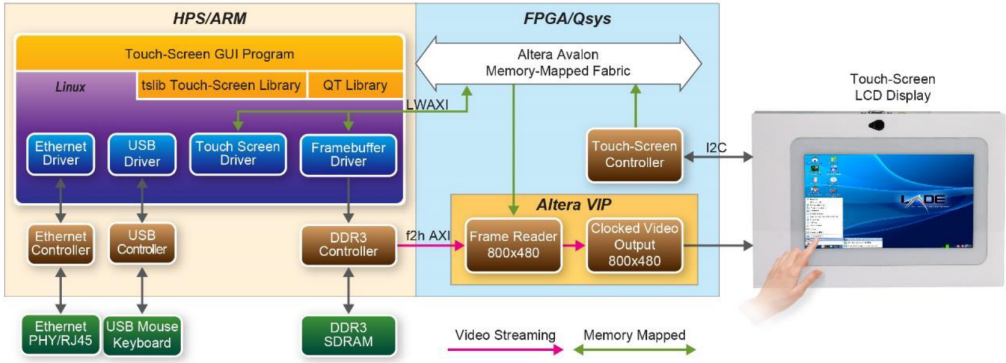


DE-10 Veek

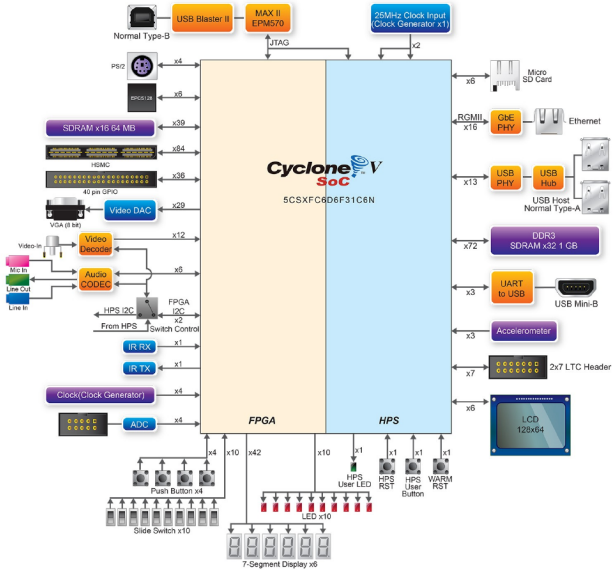
DE-10 Veek


FPGA
HPS
System
Video IN
VGA Output
UART to USB
MIC IN
LINE OUT
LINE IN
VGA 24-bit DAC
USB2
USB1
ETHERNET
Audio Codec
Vedio Codec
PS2
USB BLASTER II
DC 12V
POWER ON/OFF
ADC Connector
LCD
LED x10
MicroSD Card
SDRAM
Cyclone V FPGA
HSMC Connector
2x20 GPIO Connector
MSEL
2X7 LTC Connector
IR
USER LED
Switch x10
DDR3_HPS
7-Segment Display
Button x4
WARM_RST
HPS_RST
HPS User Button

DE-10 Veek


HPS/ARM
FPGA/Qsys
Touch-Screen GUI Program
Linux
tslib Touch-Screen Library
QT Library
Altera Avalon
Memory-Mapped Fabric
LWAXI
Ethernet Driver
USB Driver
Touch Screen Driver
Framebuffer Driver
Touch-Screen Controller
I2C
Touch-Screen
LCD Display
Altera VIP
Ethernet Controller
USB Controller
DDR3 Controller
f2h AXI
Frame Reader 800x480
Clocked Video Output 800x480
Ethernet PHY/RJ45
USB Mouse Keyboard
DDR3 SDRAM
Video Streaming
Memory Mapped

DE-10 Veek


Normal Type-B
USB Blaster II
MAX II EPM570
JTAG
25MHz Clock Input (Clock Generator x1)
x2
PS/2
x4
RS232 2X
x6
Micro SD Card
x6
SDRAM x16 64 MB
x39
RGMII x16
GbE PHY
Ethernet
HSMC
x84
40 pin GPIO
x36
Cyclone V SoC
5CSXFC6D6F31C6N
x13
USB PHY
USB Hub
USB Host Normal Type-A
VGA (8 bit)
Video DAC
x29
Video-in
Video Decoder
x12
DDR3 SDRAM x32 1 GB
x72
Mic In
Line Out
Line In
Audio CODEC
x6
x3
UART to USB
USB Mini-B
HPS I2C
FPGA I2C
From HPS
x2 Switch Control
x3
Accelerometer
IR RX
x1
IR TX
x1
x7
2x7 LTC Header
Clock(Clock Generator)
x4
x6
LCD 128x64
ADC
x4
FPGA
HPS
x4
x10
x42
x10
x1
x1
x1
x1
HPS User LED
HPS RST
HPS User Button
WARM RST
Push Button x4
LED x10
Slide Switch x10
7-Segment Display x6

Démarrage de la cible

- Le noyau est déjà compilé
- Il existe une distribution déjà utilisable
- Il faut flasher la carte SD
  - Une image est disponible
  - Il faut la copier sur la carte SD
  - On utilise `dd` (et pas `cp`)
  - L'image est plus petite que la carte SD
  - On peut étendre le système de fichiers
- Se connecter en série
- Configurer le réseau
  - Configurer le fichier `/etc/network/interfaces.d`
  - `ifconfig` ou `ip addr`
  - `ping`
  - `ssh`

Rôle d'un système d'exploitation

Rôle d'un système d'exploitation

- Gérer l'exécution des programmes
  - Lancement, terminaison
  - Intégrité des programmes, sécurité des données
  - Servir d'interface entre les différents programmes en cours d'exécution
- Gérer la mémoire
  - Allocation, droits d'accès, etc...
- Gérer l'accès au matériel
  - Servir d'interface entre les programmes et le matériel
  - Répondre aux demandes du matériel (interruptions par exemple)
  - Fournir une interface cohérente : mécanisme de Driver

Gestion du matériel

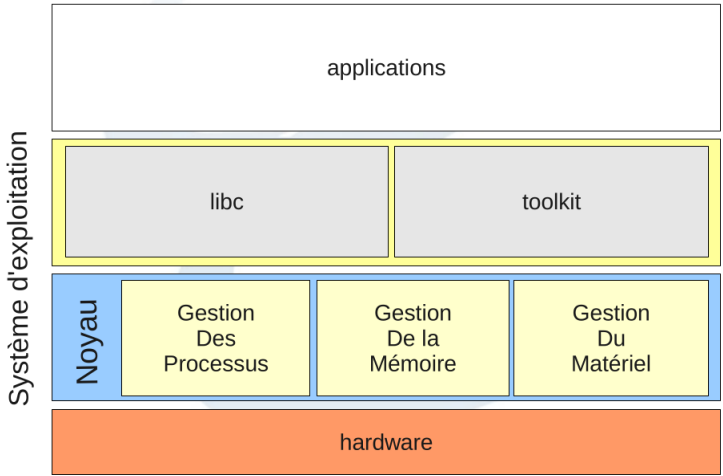
- Accès à une ressource matérielle
  - Périphériques variés : Disque, clavier, écran, imprimante, réseau...
  - Accès aux différents périphériques à travers une unique interface.
  - En proposant une interface, le système d'exploitation simplifie le développement des programmes.
- Gestion des interruptions
  - Ajouter un serveur d'interruption sans compromettre les autres programmes.
- Sécurisation des ressources critiques
  - Accès concurrent à une ressource matérielle
  - Exclusion mutuelle

Gestion des tâches

- Lancement, par l'utilisateur, de nouveaux programmes
- Contexte d'exécution
  - Registres matériels (pointeur de pile, PC, etc...)
  - Bibliothèques dynamiques
- Commutation de processus
  - Plusieurs programmes peuvent être en cours d'exécution en même temps.
  - Sauvegarde/changement de contexte
  - Partage du CPU pour les différents programmes (ordonnancement)
  - Sécuriser les données propres à chaque programme (eg. interdiction à un programme de modifier les données d'un autre programme)
- Communication entre programmes
  - Certains programmes ont besoin de communiquer entre eux (échange de données, attente de fin, etc).
  - Interface unifiée de communication (IPC – InterProcess Communication).

Types de systèmes d'exploitation

Anatomie d'un système d'exploitation



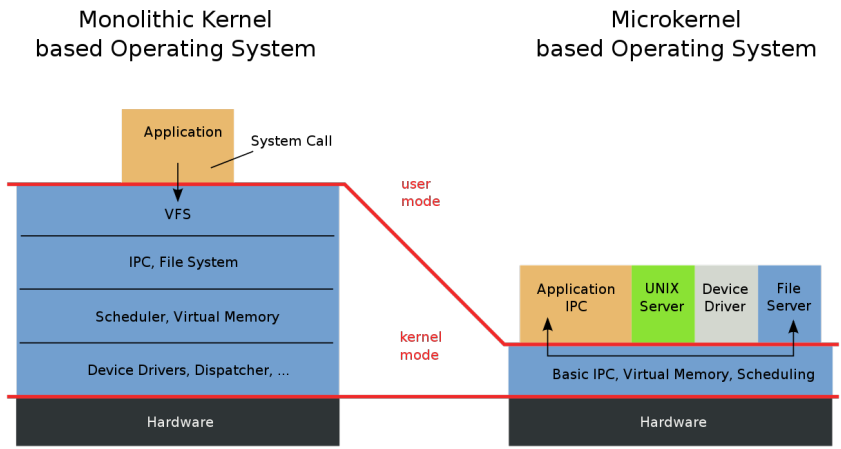
Types de systèmes d'exploitation

- En fonction des usages :
  - Grand public : réactifs, plein de programmes variés, peu de contraintes matérielles et temporelles
  - Temps réel : comportement prédictible, répondant à des contraintes temporelles fixes
  - Embarqués : peu de programmes, connus à l'avance, mais forte contraintes matérielles (peu de mémoire, etc)
- Débit vs Latence
  - Trouver le bon compromis
  - Temps-réel dur vs temps-réel mou

Types de noyaux

- Monolithique :
  - Tout est inclu dans le noyau :
  - Gestion des processus, de la mémoire, pilotes de périphériques, protocoles de communication, IPC, etc...
  - ex : Linux, Solaris
- Micronoyau :
  - Le strict minimum est inclu dans le noyau :
  - Gestion des processus, de la mémoire et certaines IPC
  - En dehors du noyau : Drivers, réseau, GUI, systèmes de fichiers, etc...
  - Le reste est délégué à des applications spécifiques appelées services
  - ex : Mach, L4, la plupart des RTOS

Noyaux monolithiques vs micronoyaux



Noyaux modulaires

- Noyaux monolithique, mais :
  - Possibilité d'ajouter et de retirer des fonctionnalités au noyau en cours d'exécution
  - Plus petit
  - Modules et services dynamiques
  - Nombre limité de pilotes chargé en mémoire
  - Chargés à la demande si besoin
- Exemple : Linux

Caractéristiques des systèmes d'exploitation

- Multi-tâches
  - Plusieurs programmes peuvent s'exécuter en même temps
- Multi-utilisateurs
  - Plusieurs utilisateurs peuvent utiliser l'ordinateur en même temps
- Multi-plateforme
  - Le système peut être installé sur des machines très variées

Multi-tâches

- Plusieurs programmes peuvent s'exécuter en même temps
  - Gérer la création et la suppression de tâches
  - Veiller à ce que chaque tâche ait accès aux ressources demandées en un temps raisonnable
  - Veiller à l'intégrité des données (variables) de chaque tâches (gestion de droits)

Multi-utilisateur

- Plusieurs utilisateurs sur la machine
  - Gérer l'accès aux ressources critiques
  - Veiller à l'intégrité des données (fichiers) de chaque utilisateur
  - Permettre un système de communication entre les utilisateurs

Multi-plateforme

- Le système peut s'installer sur différents types d'environnements
  - Offrir une interface indépendante de l'architecture matérielle aux programmes
  - Supporter plusieurs jeux d'instruction
  - Gérer plusieurs architectures matérielles différentes

Linux et ses spécificités

Historique : les débuts

- 69 : Thomson et Ritchie
  - Mise au point de la première version UNIX
  - PDP7/9 ; noyau 16Ko ; Process 8Ko, Fichier 64Ko
- 72 : Kernighan et Ritchie : langage C
- 73 : Réécriture du noyau UNIX en C
  - Gestion des processus
  - Gestion des fichiers
  - Banalisation des E/S
- 76 : Microprocesseurs 8/16 bits
- 77 : Thomson à l'université de Berkeley
  - 500 installations
- 82 : Commercialisation par ATT
  - Unix-based et Unix-like (100 000 install.)
- 84 : Efforts de normalisation
  - DEC (Ultrix)
  - Gould (UTX)
  - HP (HP-UX)

Historique : Naissance du noyau Linux

- 83 : Projet GNU (Richard Stallman)
  - Collection de logiciels libres
  - OS complet, mais pas de noyau spécifique
- 85 : MINIX par A. Tanenbaum
- 88 : Standard sur les stations de travail
- 89 : Premier BSD libre
- Sept 91 : Linux 0.01 (inspiré par Minix)
- Oct 91 : Linux 0.03 (bash et gcc)
- Dec 91 : Linux 0.10 (premières contributions externes)
- Jan 92 : Linux 0.12 (mémoire virtuelle, licence GPL)
- Mars 92 : linux 0.95 (init/login, X)
- 93 : NetBSD et FreeBSD

Historique : Évolution

- Mars 94 : Linux 1.0 (stable en production, fournit des services comparables à UNIX)
- Mars 95 : Linux 1.2 (nombreuses architectures, modules chargeables)
- Juillet 96 : Linux 2.0 (multiprocesseur, mascotte Tux)
- Janvier 99 : Linux 2.2
- 99 : MacOSX (noyau hybride)
- 2001 : Linux 2.4
- 2003 : Linux 2.6 (noyau préemptible)
- 2007 : Linux 2.6.23 (ordonnanceur CFS)
- 2008 : Linux 2.6.26 (Kgdb)
- 2011 : Linux 3.0.0 (13M lignes de code)
- 2015 : Linux 4.0.0
- 2019 : Linux 5.0.0

Caractéristiques de Linux

- Noyau monolithique : comme la plupart des Unix
- Modules : des fonctionnalités peuvent être activées ou désactivées
- Threads du noyau : certaines fonctionnalités sont indépendantes
- Support multithread natif
- Noyau préemptif : un processus peut être interrompu même en mode noyau
- Système multiprocesseur
- Nombreux systèmes de fichiers
- Nombreuses architectures supportées
- Petit et compact
- Performant
- Libre

Distributions

- Linux = Noyau, OS = GNU/Linux
- Système d'exploitation complet basé sur Linux
  - Le noyaux Linux
  - Une libc et un compilateur C (souvent gcc)
  - Un gestionnaire de boot pour charger le noyau (grub, lilo, uboot...)
  - Un système d'initialisation pour lancer les processus offrant les services standards (réseau, impression, interface graphique, ...)
  - Un shell (bash, dash, ksh, ...)
  - Un gestionnaire de packages (apt, rpm, portage, ...)
  - Un dépôt de logiciels préparés en packages (et vérifiés pour bien fonctionner avec le reste du système)

Familles de distributions

- Deb : Debian, ubuntu, et dérivée
  - Gestionnaire de paquets *.deb
  - Debian : serveur, Ubuntu : desktop
- RPM : RedHat, SuSE, Mandriva, et dérivée
  - Gestionnaire de paquets *.rpm
  - RedHat : serveur, Mandriva : desktop
- Slackware
  - plus vieille distribution (paquets *.tgz)
- Distributions sources : gentoo, sourcemage, ...
  - Recompiler tous les logiciels (paquets sources, donc)
  - Gentoo : gestionnaire de paquets portage très puissant
- Et plein d'autres
  - Au hasard : Arch linux, Manjaro...

Concepts fondamentaux

Modes d'utilisation :

- Mode utilisateur : routine du programme en cours d'exécution
  - Espace mémoire réservé
  - Contexte matériel particulier (pointeurs de pile, de programme,etc)
  - Droits restreints
- Mode noyau : routines du noyau
  - Espace mémoire total
  - Contexte matériel particulier
  - Droit d'accès totaux
  - On y accède par une interruption (soit matérielle, soit logicielle)

Concepts fondamentaux

- Processus : instance d'un programme en cours d'exécution et son contexte
- Fichiers : tout est fichier, les périphériques, les documents, etc
- Droits : les fichiers ont différents niveaux de droits (user, group, other)
- Réentrant : plusieurs processus peuvent simultanément s'exécuter en mode noyau

Contexte user/kernel

- L'utilisation de la machine est divisée en deux mode d'exécution :
  - Le mode noyau (kernel mode)
  - Le mode utilisateur (user mode)
- Le mode utilisateur est le mode d'utilisation des utilisateurs :
  - Les utilisateurs
  - Les services (serveur de fichier, web, imprimante, ...)
  - root (et oui)

Droits et modes

- La mémoire est réservée à un mode d'utilisation
  - Les applications en mode user ne peuvent pas accéder :
    - À l'espace mémoire du mode kernel
    - À l'espace mémoire des autres applications
    - À certains périphérique
- Pour gérer les droits, les processeurs dispose de plusieurs mode d'exécution
  - Un mode protégé correspondant au mode user, dans lequel tout n'est pas accessible
  - Un mode privilégié correspondant au mode kernel dans lequel on peut tout modifier

Accès au mode noyau

- Les zones de mémoire contenant le code du noyau sont en mode privilégié
  - On ne peut pas y accéder en mode user
  - On y accède par une interruption
  - Le système passe en mode privilégié
  - Le noyau traite l'interruption
  - Retour en mode user et poursuite du programme

Passage en mode noyau

Le passage en mode noyau se fait :

- Lors d'une interruption matérielle
  - Le système passe en mode privilégié et exécute alors le code du noyau prévu pour cette interruption
- Lors d'une anomalie
  - Le système lève alors une exception, passe en mode privilégié et exécute le code prévu pour cette exception
- Lors d'une demande de service au noyau via un appel système
  - Il s'agit d'une interruption logicielle voulue par l'application appelante
  - Le système passe en mode privilégié et exécute le code du noyau correspondant à l'appel système

Appels systèmes

- Ce sont des fonctions dont
  - L'appel se fait dans un programme en user mode
  - L'exécution se fait en mode kernel
  - Le retour se fait dans le programme appelant en user mode
- L'appel se fait par une interruption logicielle unique
  - Cette interruption permet de passer en mode noyau
  - Les appels systèmes sont numérotés
  - Une table permet de faire la correspondance entre le numéro de l'appel et la fonction du noyau à appeler

Les fichiers

Fichier

Définition commune

- Ensemble de données réunies sous un même nom
  - Séquence d'octets
- Enregistré sur un support de stockage (mémoire de masse)

- Les fichiers ont un format
  - Convention pour le formatage des données et métadonnées
  - Ex : exécutables, compressés, textes, documents, images, audio, vidéos...
  - Extensions facultatives sous Linux (mais souvent utilisées)
  - Executables et fichers textes souvent sans extension
- Organisés au sein d'un système de fichiers (FS)

Systèmes de fichiers

- File System (FS)
- Organisation des fichiers au sein d'un volume physique ou logique
- Gestion des répertoires, hierarchie arborescente
- Contient des métadonnées nécessaire pour (entre autres) retrouver les fichiers
  - Droits d'accès en lecture, écriture et exécution selon l'utilisateur, le groupe, ou les autres
  - Dates de dernier accès, de modification des métadonnées, de modification des données
  - Propriétaire et groupe propriétaire du fichier
  - Taille du fichier
  - Nombre de blocs utilisés par le fichier
  - Type de fichier : fichier simple, lien symbolique, répertoire, périphérique, etc.
- On dit qu'on *monte* un système de fichiers
- Le dossier où l'on monte le FS est appelé *point de montage*
- Ex : NTFS, FAT, FAT32, ext2, ext3, ext4, zfs, btrfs, etc.

Plusieurs types de fichiers

- Sous Linux, on dit que *tout est fichier*

- Fichiers ordinaires
  - Suite d'octets quelconque
- Répertoires
  - Conteneurs d'autres fichiers
- Liens
  - Liens physiques (h)
  - Liens symboliques (l)
- Fichiers spéciaux : périphériques /dev
  - En mode caractère (c)
  - En mode block (b)
- Pipes et pipes nommés, sockets...

Commandes utiles

- `ls` : affiche le contenu d'un dossier.
  - Certaines options sont très utiles.
  - `ls -la`
- `touch` : modifier la date de dernière utilisation du fichier.
  - Crée le fichier s'il n'existe pas (c'est plus utile que ça en a l'air !).
  - Exemple : `touch test.txt`
- `echo` : affiche une chaine de caractères.
  - Exemple : `echo "Bonjour à toutes et à tous"`
  - Peut être redirigé vers un fichier.
  - Exemple : `echo "Bonjour à toutes et à tous" > test.txt`
- `cat` : afficher le contenu du fichier
  - Exemple : `cat test.txt`
- `file` : affiche le type de fichier
  - Exemple : `file /usr/bin/ls`

Droits d'accès des fichiers

- Les trois principaux droits sur des fichiers sont :
  - La lecture
  - L'écriture
  - L'exécution
- Les différents droits sont associés à trois types d'utilisateurs :
  - Le propriétaire du fichier
  - Les utilisateurs appartenant au groupe auquel appartient le fichier
  - Tous les autres utilisateur

Groupe

- Un utilisateur appartient à un ou plusieurs groupes
- Servent à rassembler des utilisateurs afin de leur attribuer des droits communs
- Ex : sur un système doté d'une carte son : groupe audio autorisé à en faire usage

Modifier les droits

- `chown` permet de changer le propriétaire du fichier
  - `chown <utilisateur> fichier`
  - `chown -R <utilisateur> dossier`
  - `chown <utilisateur>:<groupe> fichier`
- `chgrp` permet de changer le groupe associé au fichier
  - `chgrp <groupe> fichier`
- `chmod` permet de changer les droits d'un fichier
  - `chmod 755 fichier`
  - `-rwxr-xr-x`
- **Attention :** *Execution* d'un dossier = droit d'entrer dans le dossier

Les fichiers
Structure des répertoires

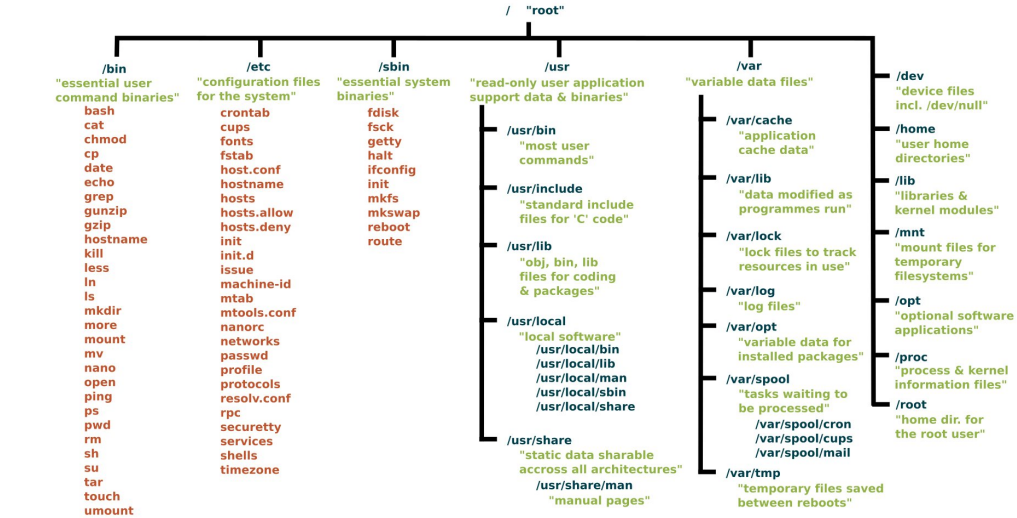
Structure des répertoires

- Contrairement à Windows, Linux voit ses disques comme une unique arborescence.
- Une partition contient la racine du système de fichier, qu'on note /
- D'autres partitions peuvent être *montées* dans des répertoires
  - Une clé USB pourra par exemple être montée dans `/media/CLE_USB`
- Une fois les différents systèmes de fichiers montés, leur utilisation est transparente.

Structure des répertoires

- La structure standard des répertoires est décrite par le FHS (Filesystem Hierarchy Standard)
- Plus ou moins respecté par la plupart des distributions Linux.
- `man hier`

Structure des répertoires



Les fichiers
Appels système

Appels système basiques

- Création : `creat()`
- Ouverture : `open()`
- Fermeture : `close()`
- Lecture : `read()`
- Écriture : `write()`
- Recherche : `lseek()`
- Suppression : `unlink()`
- Infos : `stat()`, `fstat()`, `lstat()`

RTFM

- RTFM = Read The F*cking Manual = Lisez le fichu manuel
  - À lire quand même :
    `https://blog.otso.fr/2018-04-25-il-est-temps-dabandonner-le-rtfm`
- Organisé en plusieurs sections
  1. Commandes utilisateur : executable depuis un shell
  2. Appels système : appels systèmes
  3. Fonctions de bibliothèque : libC, math...
  4. Fichiers spéciaux : fichiers dans /dev
  5. Formats de fichier
  6. Jeux
  7. Divers
  8. Administration système : /sbin, /usr/sbin
- Ambiguité entre les sections 1 et 2 ou 3

```
man 2 read
man 3 printf
```

Les fichiers

La bibliothèque standard : la LibC

open vs fopen

- open est un appel système, fopen fait partie de la libC
- fopen encapsule open
- fopen est à préferer dans la plupart des cas
  - `fopen` fournit des buffer d'entrée/sortie et est plus rapide que `open`
  - `fopen` est plus portable
  - `FILE *` permet d'utiliser fscanf et d'autres fonctions de stdio
  - `open` n'existe pas sur toutes les plateformes/tous les OS

Exemple avec fopen

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

const char *str = "Arbitrary string to be written to a file.\n";

int main(void) {
    const char* filename = "innn.txt";

    FILE* output_file = fopen(filename, "w+");
    if (!output_file) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }
    fwrite(str, 1, strlen(str), output_file);
    printf("Done Writing!\n");

    fclose(output_file);
    exit(EXIT_SUCCESS);
}
```

Exemple avec open

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

const char *str = "Arbitrary string to be written to a file.\n";

int main(void) {
    const char* filename = "innn.txt";

    int fd = open(filename, O_RDWR | O_CREAT);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    write(fd, str, strlen(str));
    printf("Done Writing!\n");

    close(fd);
    exit(EXIT_SUCCESS);
}
```

Bibliothèque, dynamique vs statique

- Bibliothèque (ou Librairie) = collection de fonction déjà compilées
- Utilisable dans un programme
- Librairie peut être dynamique ou statique
- Dans une librairie statique :
  - Les fonctions sont ajoutées à l'exécutable
  - Elles sont chargées dans la RAM au moment de l'exécution du programme
  - Il peut y avoir plusieurs instances
  - Fichiers .a (linux) et .lib (windows)
- Dans une librairie dynamique (ou partagée) :
  - L'exécutable ne contient que des liens vers les fonctions
  - Elles sont chargées dans la RAM, indépendemment
  - Il n'y a qu'une seule instance en RAM
  - Fichiers .so (shared object) et dll (dynamique link library)
- La libC est dynamique
- La commande `ldd` liste les bibliothèques partagées

Les systèmes de fichiers

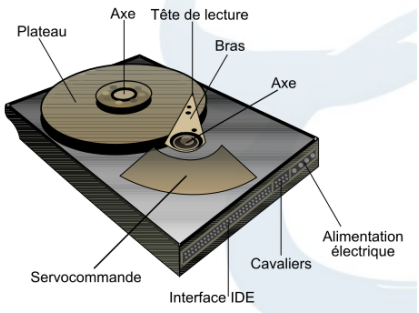
Système de fichiers (*Filesystem*, FS)

Définition

Un système de fichier est une structure de données permettant de stocker et d'organiser des fichiers et leurs données sur des supports de mémoire de masse

- La grande force de linux est de pouvoir cohabiter avec beaucoup d'autres OS, notamment en étant capable d'utiliser de nombreux systèmes de fichiers
  - ext2/3/4, MINIX, NTFS, FAT16/32, JFS, Reiser3/4, UDF, HFS, XFS, brtfs, ...
- Sur les système UNIX, les fs sont une arborescence de répertoires contenant des fichiers

Disque dur



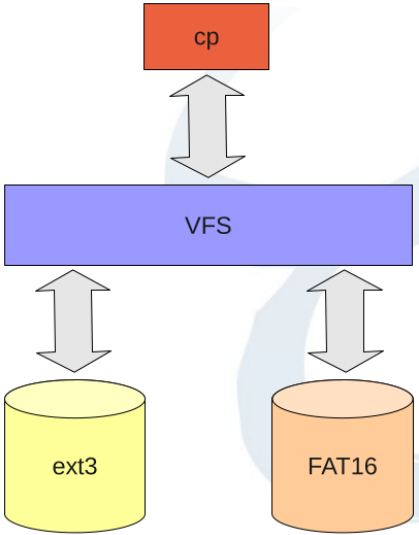
- Un disque dur est un ensemble de plateaux rigides magnétiques en rotation
- À l'origine, le support magnétique est dans un état indéterminé
- Il faut le formatter
  - Le formatage de bas niveau de blocs de 512 ou 1024 octets rend la surface du disque conforme à ce qu'attend le contrôleur hardware
  - Le formatage haut niveau crée certaines unités fonctionnelles signifiantes pour les système d'exploitation

Spécificté des FS

- Un système de fichiers correspond à l'organisation des blocs disponibles sur le disque
- Les opérations élémentaires d'accès aux fichiers (`open()`, `read()`, `write()`, etc) sont donc dépendantes du FS
- Pour que les programmes n'aient pas à se soucier du type de système de fichier, Linux met en place une couche d'abstraction appelée le VFS (*Virtual Filesystem Switch*)
- Le VFS se charge d'utiliser la bonne procédure d'accès en fonction du système de fichier utilisé

Les systèmes de fichiers
VFS

VFS




```
inf = open("/mnt/usbdisk/exam.pdf",
        O_RDONLY, 0);
outf = open("/home/toto/exam.pdf",
        O_WRONLY|O_CREAT|O_TRUNK, 0600);

do{
  i = read(inf, buf, 4096);
  write(outf, buf, i);
} while(i);

close(outf);
close(inf);
```

Systèmes de fichiers supportés par le VFS

- FS sur disque
  - Dédiés à Linux (ext2/3/4, brtfs, Tux3, ...)
  - Unix (sysv, UFS, MINIX, ...)
  - Propriétaires (VFAT, NTFS, HFS, AFFS, ...)
  - Journalisés (JFS, XFS, ...)
- FS en réseau
  - Permettent d'accéder facilement à un FS situé sur un autre ordinateur (NFS, CIFS, NCP, AFS, ...)
- FS spéciaux
  - FS virtuels permettant de simuler l'accès à certaines ressources comme des accès à des fichiers (/proc, /dev, ...)

Modèle objet du VFS

Développé en C pur pour des question de performances

- L'objet superblock
  - Stocke des informations sur le système de fichier (block de contrôle du fs sur le disque)
- L'objet inode
  - Contient des informations spécifique à un fichier (block de contrôle de fichier sur le disque), identifié par un numéro d'inode unique
- L'objet dentry
  - Information entre une entrée de répertoire (nom de fichier) et le fichier correspondant (enregistré sur le disque de manière spécifique au fs)
- L'objet file
  - Contient des informations sur l'état d'un fichier ouvert et son utilisation par un processus (n'existe qu'en RAM)

Les systèmes de fichiers

Les inodes

Les inodes

Définition

Un inode (*index node*) est une structure de données contenant des informations à propos d'un fichier ou répertoire stocké dans un systèmes de fichiers

- À chaque fichier correspond un numéro d'inode (*i-number*) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.
- Chaque fichier a un seul inode, même s'il peut avoir plusieurs noms (chacun de ceux-ci fait référence au même inode).
  - Chaque nom est appelé lien.
- Les inodes peuvent contenir aussi des informations concernant le fichier :
  - Son créateur (ou propriétaire),
  - Son type d'accès (lecture, écriture et exécution),
  - etc.

Importance des inodes

- Les inodes contiennent notamment les métadonnées des fichiers,
  - En particulier celles concernant les droits d'accès.
- Les inodes sont créés lors de la création du système de fichiers.
- La quantité d'inodes est généralement déterminée lors du formatage et dépend de la taille de la partition
- Indique le nombre maximum de fichiers que le système de fichiers peut contenir.

Inode et périphérique

- Le numéro d'inode est un entier unique pour le système de fichier dans lequel il est stocké.
- Le numéro d'inode d'un fichier peut être affiché avec la commande

  `ls -i <fichier>`
- Le terme inode se réfère usuellement aux inodes dans les périphériques bloc qui gèrent des fichiers réguliers, des répertoires et éventuellement des liens symboliques.
- Ce concept est particulièrement important pour réussir à réparer un système de fichiers endommagé (eg. avec fsck).

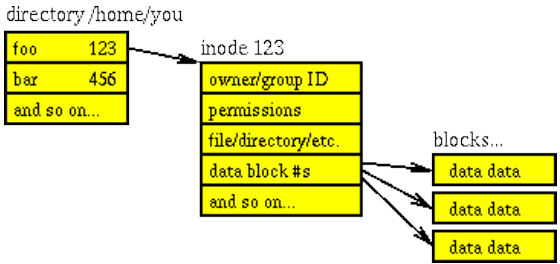
Spécification POSIX

- Le standard POSIX s'est basé sur les systèmes de fichiers traditionnels d'Unix.
- Cette norme impose donc que les fichiers réguliers aient les attributs suivants :
  - La taille du fichier en octets ;
  - Identifiant du périphérique contenant le fichier ;
  - L'identifiant du propriétaire du fichier (UID) ;
  - L'identifiant du groupe auquel appartient le fichier (GID) ;
  - Le numéro d'inode qui identifie le fichier dans le système de fichiers ;
  - Le mode du fichier qui détermine quel utilisateur peut lire, écrire et exécuter ce fichier ;
  - horodatage (timestamp) pour :
    - La date de dernière modification de l'inode ctime (affichée par la commande stat ou par ls -lc, modification des droits du fichier),
    - La date de dernière modification du fichier mtime (affichée par le classique ls -l),
    - La date de dernier accès atime (affichée par la commande stat ou par ls -lu) ;
  - Un compteur indiquant le nombre de liens physiques sur cet inode (Nlinks).

Remarque : les inodes ne contiennent pas le nom des fichiers.

Lien avec les noms de fichiers

- Les inodes ne contiennent pas le nom des fichiers, seulement les autres métadonnées
- Les repertoires sont constitués d'associations de structures contenant un nom de fichier et un numéro d'inode



- Le driver du système de fichier doit chercher un nom de fichier et le convertir en numéro d'inode.

Liens physique et liens symbolique

Lien physique

- Un lien physique est un nom de fichier additionnel pour un fichier existant
- Il pointe sur le même inode
- Il ne peut pas pointer sur un répertoire, ni sur un fichier dans un autre système de fichier.

```
ln fichier1 hlink1
ls -i hlink1
```

Liens physique et liens symbolique

Lien symbolique

- Un lien symbolique est un **fichier** qui pointe sur un autre fichier
- Il possède son propre inode
- Il peut pointer sur un répertoire ou sur un autre disque, voir sur un NFS.

```
ln -s fichier1 slink1
ls -i slink1
```

Les systèmes de fichiers
Quelques systèmes de fichiers

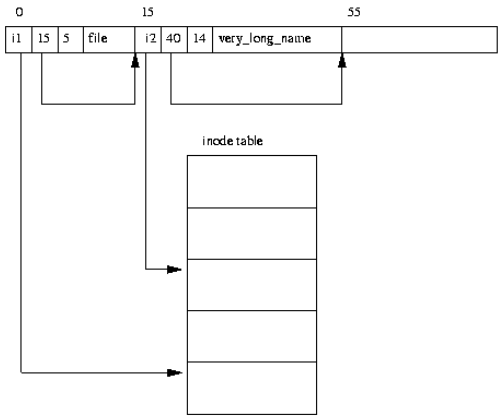
Système ext2

- Inspiré du système de fichier MINIX
- Définition d'une taille de blocs optimale (1024 à 4096 octets)
- Regroupe les blocs en groupes
  - Les blocs et les inodes d'un même groupe sont proches sur le support physique, ce qui réduit le temps d'accès
- Allocation par anticipation
  - ext2 réserve plusieurs blocs à création d'un fichier, afin d'anticiper l'augmentation de la taille du fichier

Organisation des répertoires

- Les répertoires sont des fichiers spéciaux
- Utilisés pour créer et conserver les chemins d'accès aux fichiers dans le FS.
- Un dossier est une liste d'entrées contenant les informations suivantes :
  - **Inode** : L'inode de l'entrée. Il s'agit d'un index pour le tableau des inodes.
  - **Name length** : La longueur de cette entrée en octets
  - **Name** : Le nom de cette entrée.
- Les deux premières entrées pour chaque dossier sont toujours `.` et `..` qui sont respectivement le dossier courant et le dossier parent.

Exemple



Système ext3

- Système de journalisation
- Possibilité d'agrandissement des partitions
- Indexation des répertoires contenant beaucoup de fichiers par H-Tree
  - Accès plus rapide aux fichiers d'un répertoire
- Compatibilité avec ext2
  - Possibilité de monter une partition ext3 comme une partition ext2, mais en perdant les options nouvelles

Journalisation

- Principe :
  - On écrit d'abord dans le journal
  - Puis sur les blocs de données
  - Si tout ce passe bien
    - On invalide les entrées du journal, on a le nouveau fichier
  - Si crash pendant l'écriture dans le journal
    - On ignore le contenu du journal, et garde l'ancien fichier
  - Si crash pendant l'écriture sur les blocs de donnees
    - On recopie le journal dans les blocs de données, on obtient le nouveau fichier
- Plus robuste, mais plus lent

Types de journalisation

- Journalisé
  - On écrit les données et les méta-données dans le journal
  - Le plus sûr, mais le plus lent
- Ordonné
  - On n'écrit que les méta-données dans le journal (nombre de blocs utilisés, etc)
  - Moins sûr (possible perte de données, mais pas de corruption du fs), comportement par défaut d'ext3
- Différé
  - On n'écrit que les modification des méta-données dans le journal
  - Le plus véloce, mais le moins robuste

Système ext4

- 1 Eo (1 million de To) de taille max
- Fichiers jusqu'à 16To
- Gestion par extents
  - Ensembles de blocs continus
  - Facilite la gestion des gros fichiers (jusqu'à 128Mo par extend)
- Journal checksumming
  - Vérification de la consistence du journal
- Allocation multiblocs
  - Lors de l'augmentation de la taille d'un fichier, plusieurs blocs peuvent être alloués d'un coup

Système ext4

- Nombre de fichiers dans un répertoire sans limite
- Allocation retardée
  - N'allouer les blocs qu'au dernier moment de l'écriture sur le disque
- fsck rapide (ne check pas les inodes inutilisées)
- Utilisation des barrières pour garantir l'intégrité
  - Les disques ont la facheuse tendance à réorganiser les ordres d'écritures $\rightarrow$ forcer l'exécution de certains write