**Anneaux de protection**

Le processeur crée les anneaux de protections, pas le kernel.

* 0 : mode superviseur (kernel)
* 1
* 2
* 3 : user

En mode virtualisation, le 0 est l’hyperviseur et le 1 est le kernel.

**Périphériques**

* Mode caractère : flux d’octets
* Mode bloc : ex sda
* Network

1 numéro majeur (pilote)

1 numéro mineur (le périph)

Chaque périph possède un UUID.

**Kernel**

* Vmlinuz : image kernel
* System.map : symboles du noyau
* Initrd : chargement des drivers

Dépendance entre ces fichiers et régénérés à chaque compil.

**Modules**

**Disks**

* LVM : logical volume manager
* Partition classique

Partition = partie du DD destiné à accueillir un système de fichier

Table des partitions :

Partitionnement – création swap/FS (+ blanking = formatage)

**Démarrage**

1. BIOS effectue le POST (Power On Self Test : test des composants)

UEFI remplaçant du BIOS mais exige GPT

1. BIOS charge le MBR : 512 premiers octets (table de partitions principale + bootloader = chargeur d’amorçage)

Table des partitions :

* MBR (master boot record)
  + 4 Partitions primaires maxi
  + 2.2 To maxi pour une partition
  + Impossible de démarrer un HDD MBR avec un système UEFI
* GPT (GUID partition table)

1. GRUB
2. Kernel se charge, init pilotes, démarre process kswapd et monte /
3. Kernel démarre Init : PID 1 ne possède pas de père et père de tous les processus

Init = sysVinit, upstart ou systemd

**Authentification**

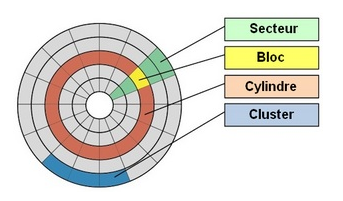
PAM : Pluggable authentication module

Entre l’utilisateur et le service demandé.

Conf des services utilisant PAM dans /etc/pam.d/

**Hardware**

* HDD



**Services**

Systemd permet de gérer des units : similaire aux services et jobs mais domaine d’application plus vaste

Ils peuvent être activés via plusieurs paramètres (réseau, hardware connecté etc...)

/lib/systemd/system copy des fichiers units : permet de starter ou stoper les units

Ne pas editer ce fichier, mais overider ce fichier dans un autre :

/etc/systemd/system

Types d’Units :

* .service : start et stop et dans quel contexte
* .socket : infos sur la socket d’écoute
* .device
* .mount
* .automount
* …

**Directories**

**bin/** Programmes utilisateurs essentiels au démarrage

**boot/** Bootloader, initrd, kernel…

**dev/** Periph access

**etc/** Confs.

**Apt/ -sources.list** Liste des dépôts

**passwd/ :** User infos

Username, passwd, UID, GID, UID infos, Home directory, Shell (or cmd)

**shells/ :** Shells valides.

**Shadow/ :** User passwd infos.

* Username
* Hash : $1$ MD5, $2a$ Blowfish, $5$ SHA 256, $6$ SHA 512

$X$ SALT $ HASH

* LastChanged : UNIX format (1970)
* Min : #days between Passwd changes.
* Max : #days valid
* Warn : #days before receiving warning.
* Inactive : #days after passwd expires : disabling.
* Expire : Absolute date which acount expires.

**Skel/ :** Fichiers copiés dans le répertoire du user lors de sa création

**Group/ :** Name :mdp :GID :member1, member2…

**Gshadow/ :**

**pam.d/ :** Conf des services utilisant PAM

**fstab/ :** Liste des periph montés automatiquement au démarrage

**init.d/ :** Fichiers de scripts de services (start stop etc pour chaque services)

**-login.defs :** Conf for **new** users acount

**-Os-release**

**-services :** Ports -> Portname

**-hostname**

**-resolv.conf :** Serveurs DNS

**Home/**

**Initrd/** Construction image noyau en RAM

**lib/** Librairies partagées essentielles au démarrage

.a = lib that contains many .o / .so = shared lib / lib linked to /bin /sbin

**Modules/** Modules du noyau

**Security/** Librairies PAM

**mnt/** Point de montage temp

**opts/** Applications tierces (non issues de la distrib)

**proc/** Pseudo fichier, occupe 0 octets en mémoire.

**XX/ :** PID processus en exec

**acpi/wakeup/** Liste des devices permettant de sortir de veille prolongée

**bus/**

**cmdline/ :** Params passés au noyau lors de son chargement

**cpuinfo/**

**crypto/ :** tous les algos installés et utilisables

**devices/ :** Tous les periph blocs et caractères du système.

**Filesystems/ :** Tous les VFS pris en compte.

**Interrupts/ :** Liste des IRQ par proc (niquel pour checker si NAPI enabled)

**Iomem/ :** @RAM utilisés

**Ioports/ :**

**Irq/**

**Kcore/ :** Représente la RAM du système

**Loadavg/ :** Charge du système

**Meminfo/ :** Usage de la RAM

**Modules/ :** Modules chargés par le noyau

**Mounts/ :** Lien symbolique, table de montage

**Net/ :** Lien symbolique, accès aux tables ARP, routage, sockets…

**Partitions/ :**

**Swaps/ :** Infos sur le swap du système

**sys/** Fichiers de conf du kernel

**kernel/ -pid-max :** Valeur du PID-max

**-uptime :** Temps écoulé en secondes depuis démarrage et durée d’inactivité

**-loadavg :** Charge moyenne depuis 1, 5, 15 min, #process, PID dernier process utilisé

**-version** : version du noyau (pae = physical adress extension = jusqu’à 64 Go de RAM).

**-vmallocinfo :** Infos sur la mémoire virtuelle

**sys/** Pseudo fichier, occupe 0 octets en RAM

**root/**

**sbin/** Programmes systèmes essentiels au démarrage

**tmp/**

**usr/ bin/**

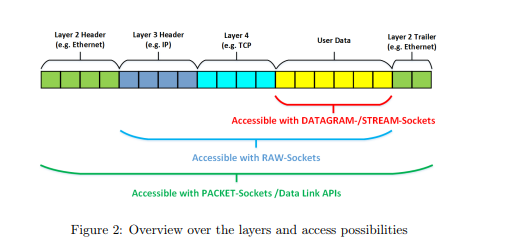
**sbin/**

**lib/** Linked to /usr/bin /usr/sbin

**var/**

**Network**

Enp0s1 : ethernet bus 0 slot 1



Kernel doit être compilé pour accepter l’utilisation des sockets RAW.

UNIX\_SOCKET : mode utilisateur

RAW\_SOCKET : couche 3

PACKET\_SOCKET : couche 2

Librairies : Libpcap, libnet.

IRQ ok quand peu de traffic, polling sinon avec buffer conséquent. (NAPI active polling au dela d’un certain seuil)

NIC envoie un signal à la réception d’un paquet.

* 1 ksoftIRQd s’occupe de tourner en boucle sur chaque CPU : permet de soulager l’interrupt handler qui va bloquer les autres IRQ.KsoftIRQ s execute à un haut niveau de PRIO mais pas autant que le handler d’IRQ, et il tourne avec IRQ activés.
* 1 structure softnet\_data crée sur chaque CPU (qui contient entre autre un poll\_list).

Linux utilise NAPI :permet d’éviter les tempêtes IRQ ( linux utilise les IRQ et au dela d’un certain seuil utilise NAPI : sondage du device ethernet toutes les X secondes : permet en gros de traiter plusieurs paquets d’un coup plutôt que un par un : technique du polling : mais attention si le buffer est trop plein : pertes de paquets !)

**Interrupt**

* Interruptions (asynchrone généré par HW)
* Exceptions (synchrone généré par proc) : les appels systems en font partis

1 interrupt handler pour chaque IRQ et donc pour chaque device

Interrupt handling : Top half et Bottom half :

* Les top désactive IRQ et refilent le taf aux bottom (execution ultra rapide) : il s’agit des interrupt handlers.
* Les bottom eux procede au traitement de l’IRQ avec haut niveau de prio mais IRQ activés

SoftIRQ, tasklets, workqueues dans cette catégorie

Register an interrupt handler :

/\* request\_irq: allocate a given interrupt line \*/

int request\_irq(unsigned int irq,

irq\_handler\_t handler,

unsigned long flags,

const char \*name,

void \*dev)

Enlever l’inscription d’un interrupt handler :

void free\_irq(unsigned int irq, void \*dev)

Handler :

static irqreturn\_t intr\_handler(int irq, void \*dev)

Execution d’un IRQ :

unsigned int do\_IRQ(struct pt\_regs regs)

Il va acquitter le HW et désactiver les IRQ sur cette ligne puis appeler le bon handler si aucun n’est activé sur cette ligne puis appel handle\_IRQ\_event()

1. Démarrage des services (Debian 1-5 : sysVinit, Debian 6-7 : upstart, Debian >7 : systemd)

**Processus**

Un job = regroupement de plusieurs processus = abstraction au niveau du bash (process group ID)

**IPC**

* Sémaphores
* Shared memory
* Messages Queues

**Memory**

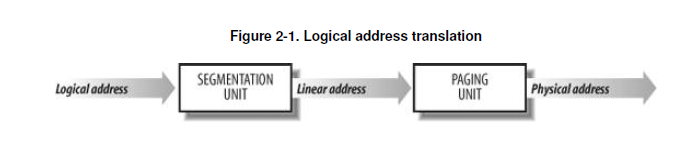
MMU fait la translation entre adresses physiques et adresses logiques (mémoire virtuelle)

Page (mémoire virt) et frames (mémoire physique) de 4 ou 8 kb en RAM

Une partie de la RAM fait office de cache

Une partie de la RAM stocke l’image du noyau et ses structures

Le reste



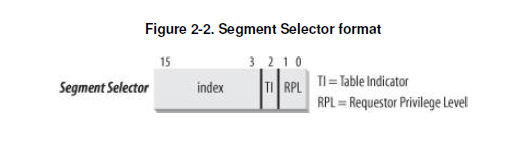
Segments : taille différentes et pages : taille fixes

RAM découpée en frames de 4k

Memory arbiter garantit les accès concurrents à la RAM (plusieurs CPU, DMA…)

Segmentation :

16 bits pour le segment identifier (ou segment selector) et 32 pour l’offset dans le segment



CS : codesegment register

SS : stack segment register

DS : data segment register

Pagination :

Gère les droits.

Table des pages gère la correspondance

Physical memory

* Nodes : bande de mémoire référencé par pg\_data\_t (struct pglist\_data) dans linux/mmzone.h. Tous les nodes sont référencé dans une pgdat\_list . La liste est chainée le next est situé dans pg\_data\_t
* Un node est divisé en zones référencé par zone\_t (struct zone\_struct) dans linux/mmzone.h
  + ZONE\_DMA : @basses : first 16 Mb
  + ZONE\_NORMAL : 16 -> 896 Mb
  + ZONE\_HIGHMEM -> >896 Mb
* Page frame référencé par struct page et toutes les pages dans un tableau mem\_map

**VFS**

* Inode : infos sur un fichier
  + Droits
  + Proprio
  + Device + numéro d’inode
  + …
* File
  + Droits d’ouverture
  + Offset
  + …
* Descripteur de fichier

Exécutable :

* SUID : permet d’endosser les droits du owner
* SGID : permet d’endosser les droits du groupe
* Sticky : permet de conserver en RAM après extinction mais obsolète