

Systèmes d'exploitation



iOS



SINA/INF/SAR

Joëlle Luter

2020

OBJECTIFS

- Connaître le rôle d'un système d'exploitation, ses fonctionnalités, et ses interactions avec les différents éléments d'un système informatique.
- Comprendre les mécanismes mis en œuvre par les systèmes d'exploitation pour remplir leur rôle.



Objectifs détaillés

- Savoir quel est le rôle d'un système d'exploitation, connaître ses fonctionnalités, son positionnement et ses interactions avec les différents éléments d'un système informatique.
- Savoir choisir un système d'exploitation adapté aux contraintes spécifiques d'un système d'information donné.
- Comprendre les mécanismes mis en œuvre par les systèmes d'exploitation pour remplir leur rôle.
- Comprendre l'influence du système d'exploitation dans le comportement des applications, être capable d'utiliser cette connaissance dans un projet de développement informatique.
- Disposer des notions théoriques suffisantes pour aborder les mécanismes de programmation multitâche concurrente.

PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Présentation)

3

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

Architecture des systèmes

Systèmes d'exploitation

Le système d'information *Représentation des données*



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Le système d'information)

2

Plan détaillé

Le système d'information

Définition

Fonctions de base

Représentation de l'information

Le matériel

Les programmes

Définitions

Programme source

Exécution d'un programme

Chaîne de compilation

Interprétation

Le système d'exploitation

Notions fondamentales

Gestion de la mémoire

Gestion du processeur

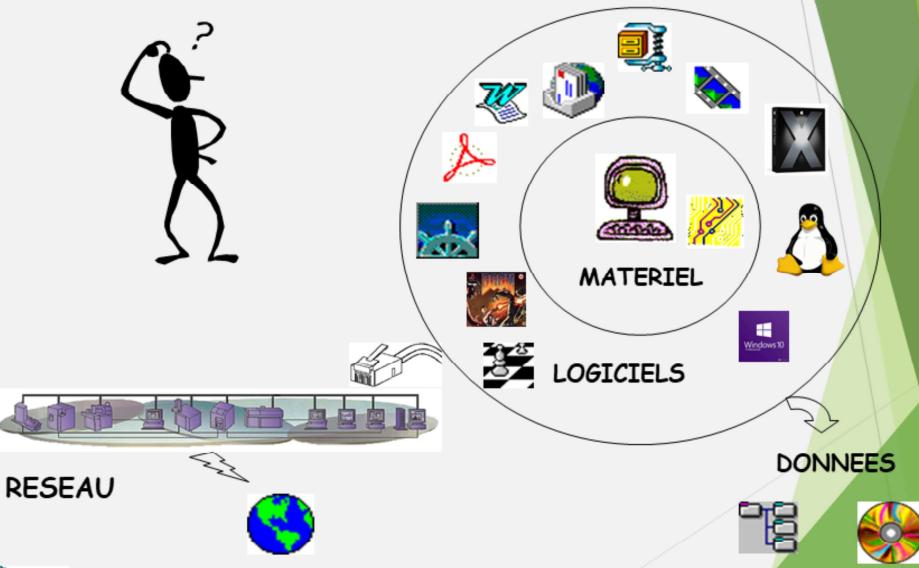
Gestion des entrées-sorties

Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'INFORMATION



Un système d'exploitation s'intègre dans l'ensemble d'un système d'information et est en interaction avec tous ses composants.

On rappellera dans cette première partie les éléments d'un système d'information.

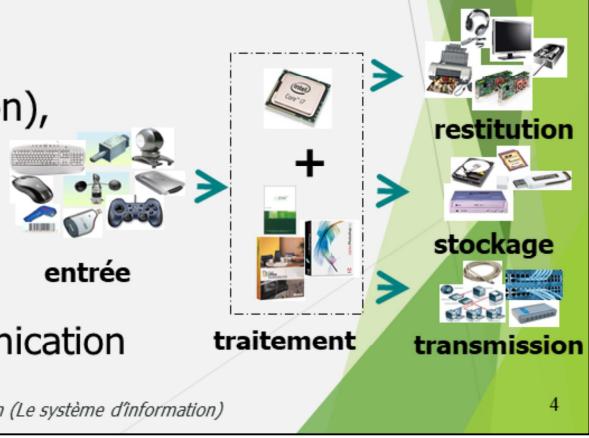
LE SYSTÈME D'INFORMATION

DÉFINITION

Ensemble des **moyens** destinés à répondre à un **besoin** spécifique de traitement et d'exploitation de l'**information**

FONCTIONS DE BASE

- entrée (saisie, acquisition),
- traitement,
- sortie
 - restitution,
 - stockage,
 - transmission/communication



Systèmes d'exploitation (Le système d'information)

4

Besoin : fonctionnalités définies dans le cahier des charges (domaine de l'ingénierie système)

Information : le cœur du problème. L'objectif de tout système informatique est le traitement de l'information.

Moyens : matériel + logiciels (auxquels il faudrait rajouter les « acteurs » intervenant tout au long du cycle de vie du système, mais ce n'est pas le propos de ce cours)

Les fonctions de base du système d'information sont centrées sur le flux de l'information (flux des données)

Dispositifs d'entrée : clavier, souris, webcam, manette, capteurs, réseau, ...

Traitement : matériel (unité centrale : processeur + mémoire) + logiciel

Dispositifs de sortie :

- Restitution : écran, casque audio, imprimante, actionneur, ...
- Stockage : mémoires de masse (disques, mémoires flash, ...)
- Transmission : réseau, ports, bus

LA PRÉSENTATION DE L'INFORMATION



Systèmes d'exploitation (Le système d'information)



5

LA REPRÉSENTATION DE L'INFORMATION

L'information doit être représentée sous une forme exploitable par le matériel.

➤ Point de vue utilisateur

nombres,
texte,
images,
son, ...

1205 . 258

Bonjour !



➤ Point de vue machine

digits binaires = bits (0 ou 1)



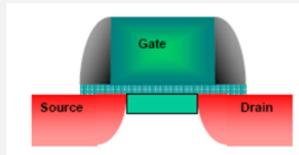
Systèmes d'exploitation (Le système d'information)

6

Remarque : la représentation binaire (0, 1) est déjà une abstraction vis-à-vis de la réalité machine (niveaux électriques).

LA PRÉSENTATION DE L'INFORMATION

Composant de base de la microélectronique



Interrupteur (MOS)

Fermé, le courant passe : VRAI, 1

Ouvert, le courant ne passe pas : FAUX, 0

-> **représentation binaire de l'information**



Systèmes d'exploitation (*Le système d'information*)

7

Ordre de grandeur : les processeurs Intel Ivy Bridge comptent 1,4 milliard de transistors pour 160 mm²

LA REPRÉSENTATION DE L'INFORMATION

CODAGE INTERNE : L'OCTET

Unité de base : 1 octet = 8 binary digits (bits)

valeur (0 ou 1)								
poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

De zéro à deux cent cinquante cinq



Systèmes d'exploitation (Le système d'information)

8

Attention à l'intervalle de valeurs correspondant à la représentation interne choisie !

Codage de l'information (pour rappel)

Base 10 { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

trois : **3** dix : **10** dix-sept : **17**

Base 2 { 0, 1 }

trois : **11** dix : **1010** dix-sept : **10001**

Base 16 { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }

trois : **3** dix : **A** dix-sept : **11**

LA REPRÉSENTATION DE L'INFORMATION

CODAGE INTERNE : CONSÉQUENCES

- Représentation numérique des données
- Nombres entiers naturels seulement
- Comment représenter les entiers relatifs, les réels, le texte, le son, l'image ?



La représentation interne binaire ne permet de stocker que des valeurs numériques entières positives ou nulles.

Tous les autres types d'information (caractères, nombres entiers signés, nombres réels) devront donc être codés à l'aide de valeurs entières.

LA REPRÉSENTATION DE L'INFORMATION

LES NOMBRES

➤ Nombres entiers

n octets + convention de codage du signe

➤ Nombres réels

norme IEEE 754, signe + exposant + mantisse

(*float = 32 bits : 1 bit signe, 8 bits exposant, 23 bits mantisse*)



Systèmes d'exploitation (Le système d'information)

10

Représentation des nombres entiers signés

La représentation de $-x$ est obtenue par le complément à 1 ou à 2 de x :

Complément à 1

- on complémente (inverse) tous les bits

Complément à 2

- on code en binaire la valeur absolue de x ,
- on complémente (inverse) tous les bits,
- on ajoute 1 au résultat obtenu.

Par exemple, pour représenter la valeur décimale - 22 sur 1 octet :

- on code 22 en binaire : $x = \text{00010110}$
- on complémente : 11101001
- on ajoute 1 : $-x = \text{11101010}$

Représentation des réels (norme IEEE 754)

Le signe est codé sur 1 bit, l'exposant et la mantisse sur 32 ou 64 bits selon que le nombre est représenté en simple ou double précision.

$$x = (-1)^{\text{SM}} * 1.\text{M} * 2^{(\text{Eb}-127)}$$

SM : signe de la mantisse sur 1 bit

M : mantisse sur 23 ou 52 bits

Eb : exposant biaisé sur 8 ou 11 bits

On parle de représentation en virgule flottante.

LA REPRÉSENTATION DE L'INFORMATION

LES NOMBRES : plages de valeurs



- Les nombres sont codés sur un nombre **limité** d'octets
- Dans un programme, chaque variable a un type qui détermine sa taille en octets et son intervalle de valeurs
- Exemples :
 - Variable numérique non signée codée sur 1 octet : [0;255]
 - Variable numérique signée codée sur 1 octet : [-128;+127]
 - Que se passe-t-il si on tente d'affecter la valeur 256 à une variable numérique non signée sur 1 octet ?



Que se passe-t-il si on affecte la valeur 256 à une variable numérique non signée sur 1 octet ?

256 donne 100000000 en binaire (nécessite donc 9 digits pour être représenté)
Lors du chargement dans la variable, le digit de gauche est tronqué.

La variable contient donc la valeur binaire 00000000, soit zéro.

Toute représentation a donc une limite : le nombre d'octets sur lequel une donnée est codée, qui implique une valeur maximale atteignable.

Au-delà de cette valeur, la représentation devient incohérente.

LA REPRÉSENTATION DE L'INFORMATION

LES NOMBRES : plages de valeurs

Conséquences d'un débordement



Boeing 787-9 Dreamliner (2015)



Unités de contrôle des générateurs moteurs

Compteur en centisecondes codé sur 4 octets (un entier signé en architecture 32 bits)

Valeur maximale :
 2^{31} centisecondes = 248, 55 jours

Quand ce délai est atteint,
arrêt immédiat des moteurs

Systèmes d'exploitation (Le système d'information)

12

Bug rapporté par Boeing en 2015 à la FAA : débordement d'entier dans le code des unités de contrôle des générateurs moteurs. Quand le bug apparaît, les générateurs passent en mode sécurité-défaut, causant une interruption totale du système électrique et l'arrêt immédiat des moteurs.

Solution immédiate adoptée : directive adressée par la FAA à toutes les compagnies aériennes demandant de redémarrer les générateurs à intervalle régulier jusqu'à la disponibilité d'un correctif.

Boeing a depuis corrigé le bug !

LA REPRÉSENTATION DE L'INFORMATION

LES NOMBRES : plages de valeurs

Conséquences d'un débordement Système de référence inertielle



Ariane 5 (1996)



Systèmes d'exploitation (Le système d'information)

Calcul de vitesse horizontale codé sur 2 octets alors que la valeur réelle est supérieure.

Transmission d'une valeur erronée au calculateur qui effectue une correction de trajectoire non pertinente conduisant à l'arrachement des boosters et à l'autodestruction de la fusée.

13

Un dépassement d'entier a conduit le système de référence inertielle à transmettre une valeur de vitesse horizontale fausse au calculateur principal. Celui-ci a alors procédé à une correction importante de trajectoire par rapport à une déviation qui, en fait, ne s'était pas produite.

0,05 secondes plus tard, le système de secours a subi le même problème.

Le lanceur s'est alors incliné de plus de 20 degrés, ce qui a entraîné la séparation des boosters de l'étage principal et le déclenchement de l'autodestruction de la fusée.

Le CNES avait réutilisé pour la fusée Ariane 5 le logiciel de navigation qui avait été conçu pour Ariane 4.

La vitesse horizontale d'Ariane 5 étant beaucoup plus importante que celle d'Ariane 4, la valeur maximale pouvant être contenue dans la variable prévue a été dépassée, fournissant ainsi une donnée fausse.

La charge utile de la fusée était composée de 4 satellites d'une valeur globale de 370 millions de dollars.

LA REPRÉSENTATION DE L'INFORMATION

LE TEXTE (LES CARACTÈRES)



position touche
clavier (code de
balayage)

valeur
numérique

dessin du
caractère

Tables de correspondance caractère <-> valeur numérique
ASCII, ASCII étendu, UNICODE



Systèmes d'exploitation (Le système d'information)

14

Le processeur ne connaît que la valeur numérique associée au caractère.

ASCII (0 à 127) American Standard Code for Information Interchange

Standard de 7 bits pour représenter un caractère

- 0 (00h) à 31 (1Fh) : caractères de contrôle : sauts, bips, ...
- 48 (30h) à 57 (39h) : chiffres dans l'ordre (valeur = 4 bits de poids faible)
- 65 (41h) à 90 (5Ah) : majuscules
- 97 (61h) à 122 (7Ah) : minuscules (on modifie le 5ème bit pour passer des majuscules aux minuscules : +32)
- caractères spéciaux (ponctuation, parenthèses, ...)

ASCII étendu (0 à 255)

8 bits pour représenter un caractère

Inclut la table ASCII (0 à 127) + caractères accentués + ...

Pour la compatibilité, il est nécessaire d'expliciter le codage utilisé pour les caractères complémentaires : par exemple, iso-latin-1.

Standard Unicode

Normalise les caractères de la plupart des langues (langues européennes, arabe, asiatiques, ...), ainsi que de nombreux symboles en y associant un nom et une valeur numérique (point de code) : plus de 100000 caractères définis.

Formats de représentation interne des points de code : UTF-8, UTF-16, UTF-32

Le standard ASCII est inclus dans le standard Unicode.

LA PRÉSENTATION DE L'INFORMATION

LE TEXTE (LES CARACTÈRES) : LA TABLE ASCII

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr		Dec	Hx	Oct	Html	Chr		Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)		32	20 040	6#32;	Space	'		64	40 100	6#64;	B	96	60 140	6#96;	'			
1	1 001	SOH	(start of heading)		33	21 041	6#33;	!			65	41 101	6#65;	A	97	61 141	6#97;	a			
2	2 002	STX	(start of text)		34	22 042	6#34;	"			66	42 102	6#66;	B	98	62 142	6#98;	b			
3	3 003	ETX	(end of text)		35	23 043	6#35;	#			67	43 103	6#67;	C	99	63 143	6#99;	c			
4	4 004	EOT	(end of transmission)		36	24 044	6#36;	\$			68	44 104	6#68;	D	100	64 144	6#100;	d			
5	5 005	ENQ	(enquiry)		37	25 045	6#37;	%			69	45 105	6#69;	E	101	65 145	6#101;	e			
6	6 006	ACK	(acknowledge)		38	26 046	6#38;	&			70	46 106	6#70;	F	102	66 146	6#102;	f			
7	7 007	BEL	(bell)		39	27 047	6#39;	'			71	47 107	6#71;	G	103	67 147	6#103;	g			
8	8 010	BS	(backspace)		40	28 050	6#40;	(72	48 110	6#72;	H	104	68 150	6#104;	h			
9	9 011	TAB	(horizontal tab)		41	29 051	6#41;)			73	49 111	6#73;	I	105	69 151	6#105;	i			
10	A 012	LF	(NL line feed, new line)		42	2A 052	6#42;	*			74	4A 112	6#74;	J	106	6A 152	6#106;	j			
11	B 013	VT	(vertical tab)		43	2B 053	6#43;	+			75	4B 113	6#75;	K	107	6B 153	6#107;	k			
12	C 014	FF	(NP form feed, new page)		44	2C 054	6#44;	,			76	4C 114	6#76;	L	108	6C 154	6#108;	l			
13	D 015	CR	(carriage return)		45	2D 055	6#45;	-			77	4D 115	6#77;	M	109	6D 155	6#109;	m			
14	E 016	SO	(shift out)		46	2E 056	6#46;	.			78	4E 116	6#78;	N	110	6E 156	6#110;	n			
15	F 017	SI	(shift in)		47	2F 057	6#47;	/			79	4F 117	6#79;	O	111	6F 157	6#111;	o			
16	10 020	DLE	(data link escape)		48	30 060	6#48;	0			80	50 120	6#80;	P	112	70 160	6#112;	p			
17	11 021	DC1	(device control 1)		49	31 061	6#49;	1			81	51 121	6#81;	Q	113	71 161	6#113;	q			
18	12 022	DC2	(device control 2)		50	32 062	6#50;	2			82	52 122	6#82;	R	114	72 162	6#114;	r			
19	13 023	DC3	(device control 3)		51	33 063	6#51;	3			83	53 123	6#83;	S	115	73 163	6#115;	s			
20	14 024	DC4	(device control 4)		52	34 064	6#52;	4			84	54 124	6#84;	T	116	74 164	6#116;	t			
21	15 025	NAK	(negative acknowledge)		53	35 065	6#53;	5			85	55 125	6#85;	U	117	75 165	6#117;	u			
22	16 026	SYN	(synchronous idle)		54	36 066	6#54;	6			86	56 126	6#86;	V	118	76 166	6#118;	v			
23	17 027	ETB	(end of trans. block)		55	37 067	6#55;	7			87	57 127	6#87;	W	119	77 167	6#119;	w			
24	18 030	CAN	(cancel)		56	38 070	6#56;	8			88	58 130	6#88;	X	120	78 170	6#120;	x			
25	19 031	EM	(end of medium)		57	39 071	6#57;	9			89	59 131	6#89;	Y	121	79 171	6#121;	y			
26	1A 032	SUB	(substitute)		58	3A 072	6#58;	:			90	5A 132	6#90;	Z	122	7A 172	6#122;	z			
27	1B 033	ESC	(escape)		59	3B 073	6#59;	;			91	5B 133	6#91;	[123	7B 173	6#123;	{			
28	1C 034	FS	(file separator)		60	3C 074	6#60;	<			92	5C 134	6#92;	\	124	7C 174	6#124;				
29	1D 035	GS	(group separator)		61	3D 075	6#61;	=			93	5D 135	6#93;]	125	7D 175	6#125;	}			
30	1E 036	RS	(record separator)		62	3E 076	6#62;	>			94	5E 136	6#94;	^	126	7E 176	6#126;	-			
31	1F 037	US	(unit separator)		63	3F 077	6#63;	?			95	5F 137	6#95;	_	127	7F 177	6#127;	DEL			

Source: www.LookupTables.com

Systèmes d'exploitation (Le système d'information)

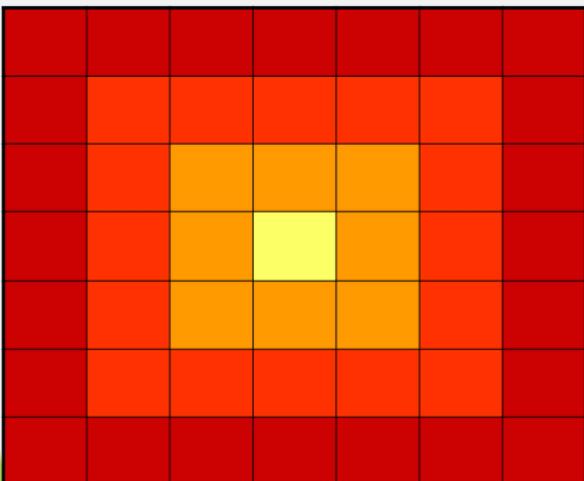
15

Le code ASCII (American Standard Code for Information Interchange) date de 1960.

On notera que les 32 premiers caractères (0-31) sont des caractères non-imprimables.

LA REPRÉSENTATION DE L'INFORMATION

LES IMAGES



Systèmes d'exploitation (*Le système d'information*)

Image bitmap

définition : 49 pixels

pixel

- position X
- position Y
- apparence

16

Les images peuvent être représentées de manière matricielle ou vectorielle.

Formats matriciels

Les images sont stockées sous forme d'ensemble de pixels (points définis par une valeur numérique correspondant à leur apparence).

Format **bitmap** : pas de compression. Toutes les images de même résolution ont la même taille. Qualité élevée, taille importante.

Format **jpeg** (Joint Photographic Expert Group) : format compressé. Perte de qualité de l'image en fonction du taux de compression. Codage sur 24 bits -> 16 millions de couleurs. Ne supporte pas la transparence.

Format **gif** (Graphics Interchange Format) : 256 couleurs max (codage 8 bits) -> fichiers de petite taille.

Format **png** (Portable Network Graphic) : format compressé sans perte de données. Codage sur 24 bits -> 16 millions de couleurs. Taille > format jpeg.

Formats vectoriels

Les images ne sont plus stockées sous forme de pixels, mais sous forme d'objets graphiques définis par leur géométrie ou leur fonction.

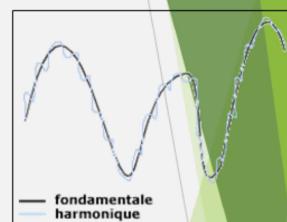
Vidéo = images numériques + son + cadencement temporel. Standards MPEG-x (Moving Picture Expert Group)

LA REPRÉSENTATION DE L'INFORMATION

LE SON

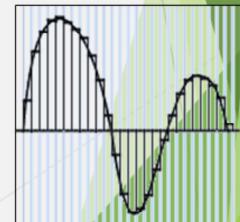
Son physique

Variation continue de la pression de l'air (ou du courant dans un électroaimant)



Son numérique

Échantillonnage des variations de pression à intervalle de temps donné -> codage d'une valeur par échantillon



Systèmes d'exploitation (Le système d'information)

17

Toute information analogique est continue en temps et en amplitude. Elle doit être numérisée avant de pouvoir être traitée.

Le son en est un exemple, mais le même principe s'applique à tout traitement de grandeurs physiques continues.

La numérisation est faite par un dispositif matériel spécifique appelé Convertisseur Analogique-Numérique (**CAN**)

Elle utilise la méthode de Modulation d'Impulsions Codée (MIC ou PCM - Pulse Coded Modulation - en anglais).

Cette méthode consiste à prélever à intervalle de temps régulier (**échantillonnage**) la valeur de l'amplitude de l'onde puis à la coder sur un certain nombre de bits (**quantification et codage**).

Pour le son, la fréquence d'échantillonnage du standard CD est de 44,1 kHz avec une quantification sur 16 bits.

Le standard DVD audio a une fréquence d'échantillonnage variable de 96 kHz à 192 kHz avec une quantification sur 16 ou 24 bits

Pour la diffusion, on utilise un convertisseur numérique/analogique (**CNA**) qui va générer une tension (ou plus rarement un courant) proportionnelle aux valeurs numériques issues du traitement (méthodes PWM – Pulse Width Modulation - et/ou réseau de résistances).

Systèmes d'exploitation

Le système d'information

Le matériel



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Le système d'information)

2

Plan détaillé

Le système d'information

Définition

Fonctions de base

Représentation de l'information

Le matériel

Les programmes

Définitions

Programme source

Exécution d'un programme

Chaîne de compilation

Interprétation

Le système d'exploitation

Notions fondamentales

Gestion de la mémoire

Gestion du processeur

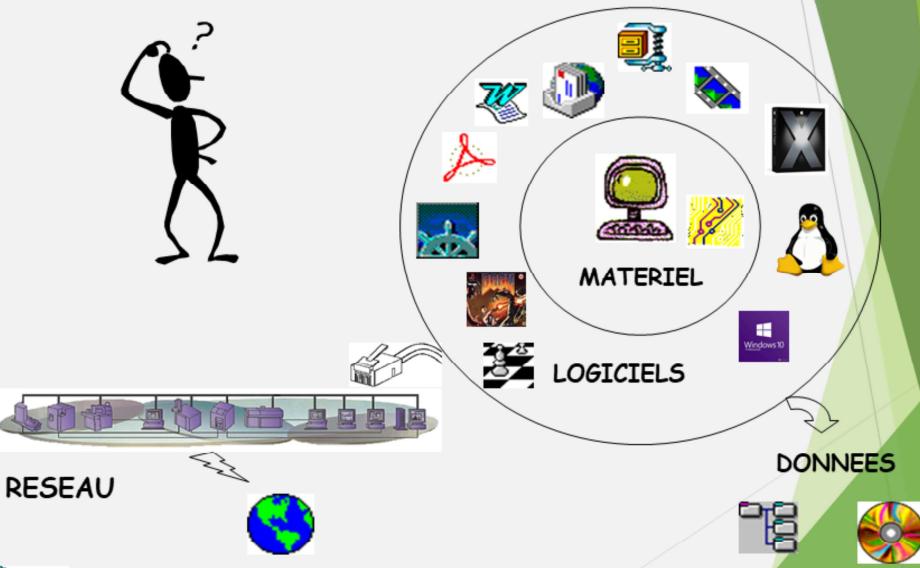
Gestion des entrées-sorties

Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'INFORMATION



Un système d'exploitation s'intègre dans l'ensemble d'un système d'information et est en interaction avec tous ses composants.

On rappellera dans cette première partie les éléments d'un système d'information.

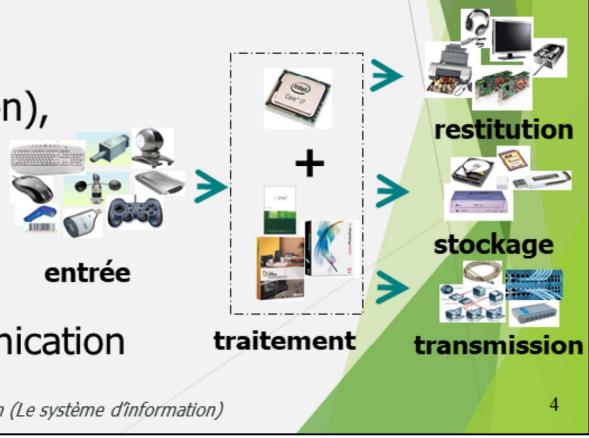
LE SYSTÈME D'INFORMATION

DÉFINITION

Ensemble des **moyens** destinés à répondre à un **besoin** spécifique de traitement et d'exploitation de l'**information**

FONCTIONS DE BASE

- entrée (saisie, acquisition),
- traitement,
- sortie
 - restitution,
 - stockage,
 - transmission/communication



Systèmes d'exploitation (Le système d'information)

4

Besoin : fonctionnalités définies dans le cahier des charges (domaine de l'ingénierie système)

Information : le cœur du problème. L'objectif de tout système informatique est le traitement de l'information.

Moyens : matériel + logiciels (auxquels il faudrait rajouter les « acteurs » intervenant tout au long du cycle de vie du système, mais ce n'est pas le propos de ce cours)

Les fonctions de base du système d'information sont centrées sur le flux de l'information (flux des données)

Dispositifs d'entrée : clavier, souris, webcam, manette, capteurs, réseau, ...

Traitement : matériel (unité centrale : processeur + mémoire) + logiciel

Dispositifs de sortie :

- Restitution : écran, casque audio, imprimante, actionneur, ...
- Stockage : mémoires de masse (disques, mémoires flash, ...)
- Transmission : réseau, ports, bus

LE MATÉRIEL



Le matériel participe à toutes les fonctions du système d'information (entrée, sortie, traitement).

LE MATÉRIEL

Citer des composants matériels d'un système informatique ...



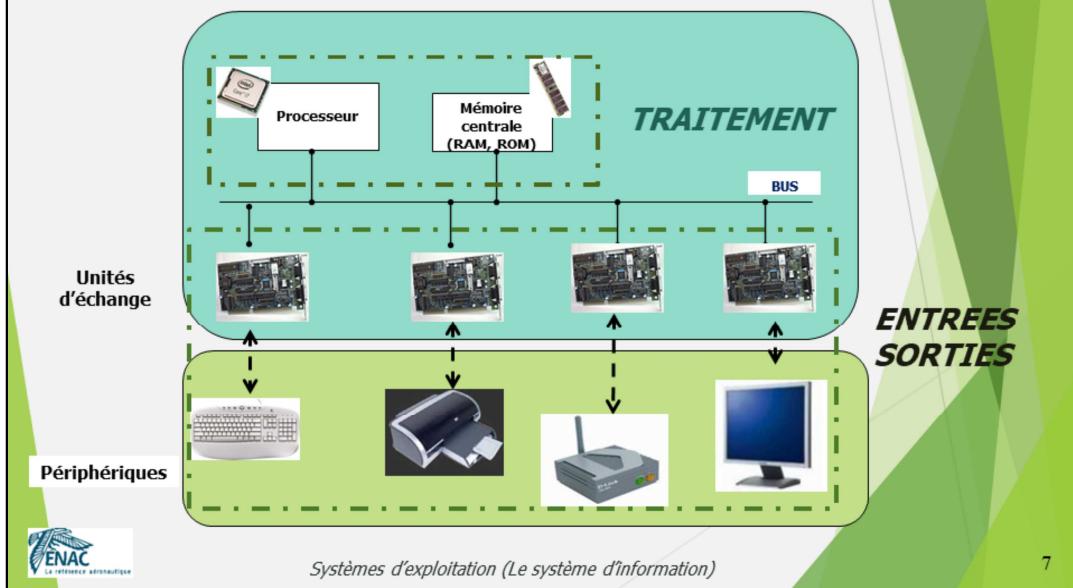
Systèmes d'exploitation (*Le système d'information*)

6

Activité interactive

LE MATÉRIEL

SCHÉMA FONCTIONNEL



Un système à processeur contient au minimum les éléments suivants :

- un ou plusieurs processeurs
- de la mémoire (mémoire centrale)
- des bus qui assurent la circulation de l'information entre les divers composants
- des circuits d'entrée-sortie (périphériques) qui assurent l'entrée, la sortie et le stockage de l'information.

Les périphériques, les bus et la mémoire nécessitent des composants électroniques de contrôle. On parle de contrôleurs. Ceux-ci sont dans certains cas appelés interfaces voire « cartes ». On regroupe les contrôleurs et interfaces de périphériques sous le terme générique d'unité d'échange.

LE MATÉRIEL

A QUOI SERT ... LE PROCESSEUR ?



Systèmes d'exploitation (Le système d'information)

8

LE MATÉRIEL

A QUOI SERT ... LA RAM ?



Systèmes d'exploitation (*Le système d'information*)

9

LE MATÉRIEL

A QUOI SERT ... LA ROM ?



Systèmes d'exploitation (Le système d'information)

10

LE MATÉRIEL

A QUOI SERT ... UNE UNITÉ D'ÉCHANGE?

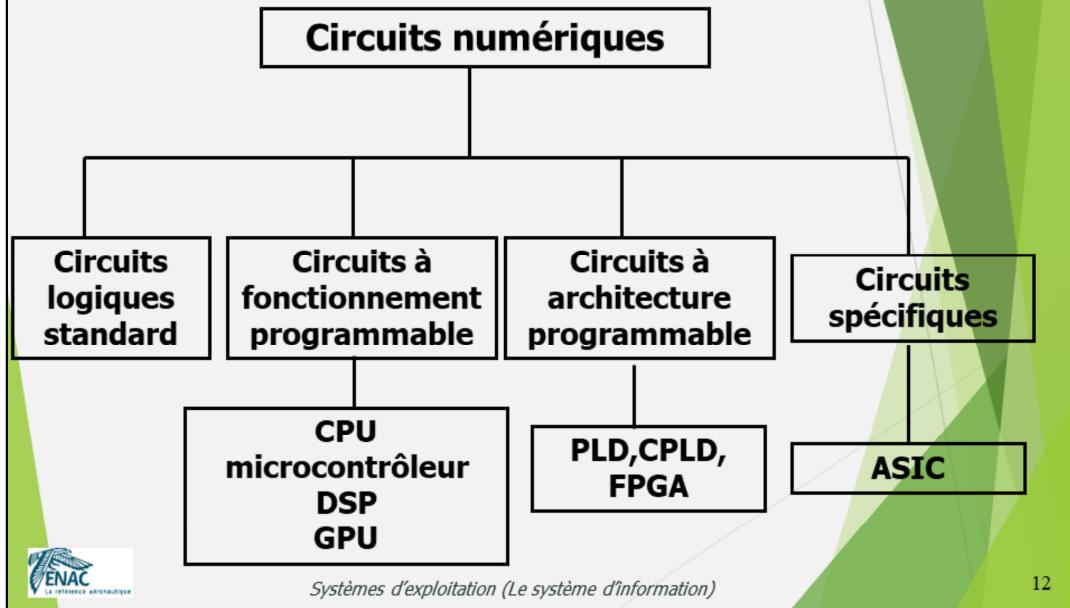


Systèmes d'exploitation (*Le système d'information*)

11

LE MATÉRIEL

CIRCUITS NUMÉRIQUES DE TRAITEMENT



Plusieurs types de circuits numériques peuvent assurer ou participer au traitement de l'information :

Circuits à fonctionnement programmable : l'architecture matérielle est figée, le traitement est assuré par le logiciel.

CPU : Central Processing Unit (processeur généraliste d'architecture Von Neuman, qui peut exécuter tout type de traitements, mais n'est optimisé pour aucun type particulier)

DSP : Digital Signal Processor (processeur d'architecture Harvard, optimisé pour les calculs de traitement du signal)

GPU : Graphics Processing Unit (processeur optimisé pour le traitement graphique, mais qui peut aussi être utilisé pour d'autres applications de calcul parallèle)

Microcontrôleur (MCU : MicroController Unit) : circuit à processeur qui intègre sur la même puce : un CPU, de la mémoire, des circuits d'entrée-sortie, des timers, des convertisseurs, etc Il est à noter que certains microcontrôleurs peuvent supporter un système d'exploitation, par exemple le STM32 F4.

Circuits à architecture programmable : l'architecture matérielle des circuits est programmable

PLD : Programmable Logic Device

CPLD : Complex Programmable Logic Device

FPGA : Field-Programmable Gate Array

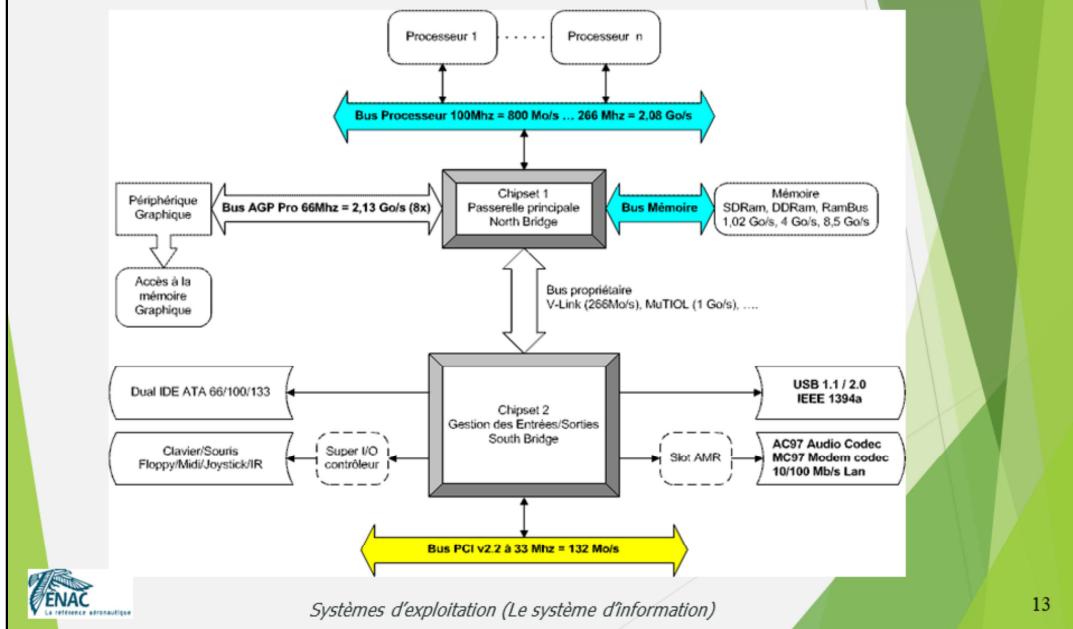
Circuits intégrés spécifiques : l'architecture matérielle est figée et assure seule le traitement

ASIC : Application-Specific Integrated Circuit

Dans le cadre de ce cours, nous nous intéresserons aux circuits à processeur (CPU) pouvant supporter un système d'exploitation.

LE MATÉRIEL

CARTE MÈRE INTEL PENTIUM



Un **chipset** est un circuit électronique regroupant plusieurs composants assurant chacun une fonctionnalité.

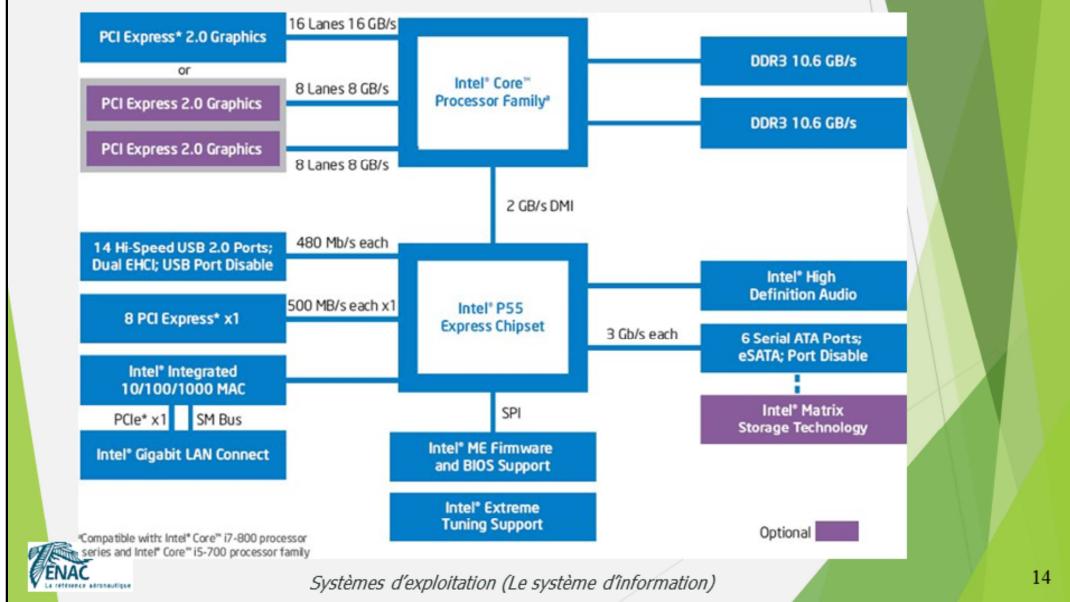
Dans cette carte mère assez ancienne, il y a 2 chipsets appelés respectivement North Bridge et South Bridge.

Le North Bridge, plus proche du processeur assure les fonctionnalités nécessitant un traitement rapide : gestion de la mémoire, du bus système, de l'accès à la carte graphique.

Le South Bridge assure la gestion des bus d'accès aux périphériques autres que la carte graphique et intègre plusieurs interfaces (série, USB, audio, réseau, ...).

LE MATÉRIEL

CARTE MÈRE INTEL I5



Systèmes d'exploitation (Le système d'information)

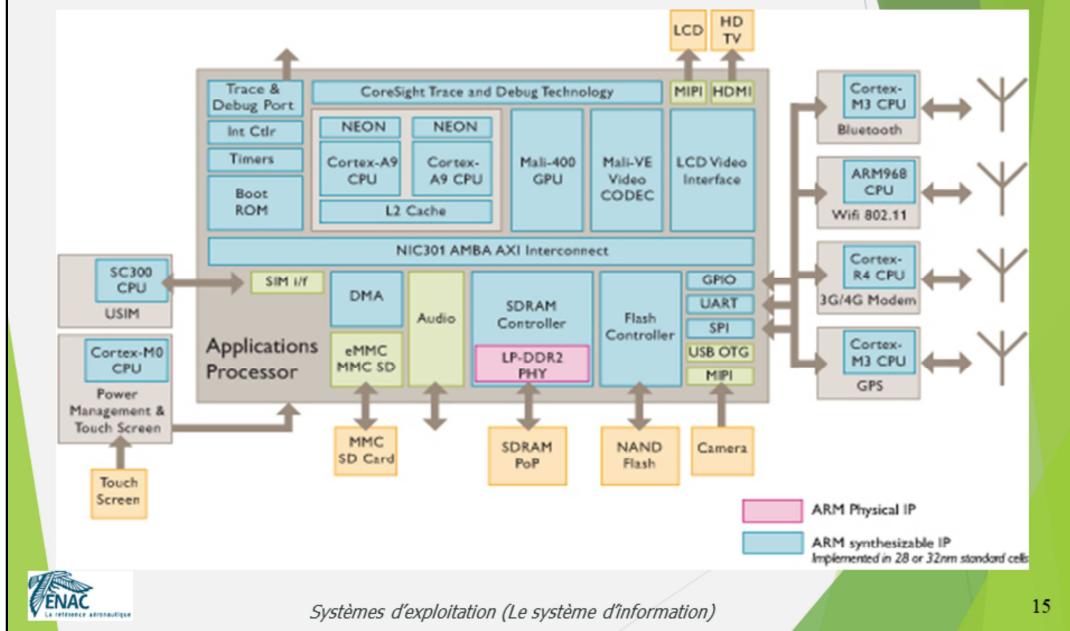
14

Dans cette carte mère plus récente, les fonctionnalités précédemment comprises dans le North Bridge sont directement intégrées dans la puce du processeur (gestion du bus graphique PCI-Express, contrôleur mémoire).

Il n'y a plus qu'un seul chipset qui assure le rôle précédemment attribué au South Bridge.

LE MATÉRIEL

S.o.C. ARM POUR SMARTPHONE



Sur certains systèmes embarqués (ici un smartphone), tous les composants sont groupés sur une même puce. On parle de SoC = System on Chip.

On y retrouve les composants des cartes mères de type PC. Tous les contrôleur sont intégrés (graphique, ...).

On notera en plus la présence de composants physiques complémentaires : timer, codec Video, port de debug, GPIO (General Purpose Input/Output, ensemble de connecteurs d'entrée sortie non spécifiques), ...

Systèmes d'exploitation

Le système d'information

Les programmes



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Le système d'information)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions**
- Programme source**
- Exécution d'un programme**
- Chaîne de compilation**
- Interprétation**

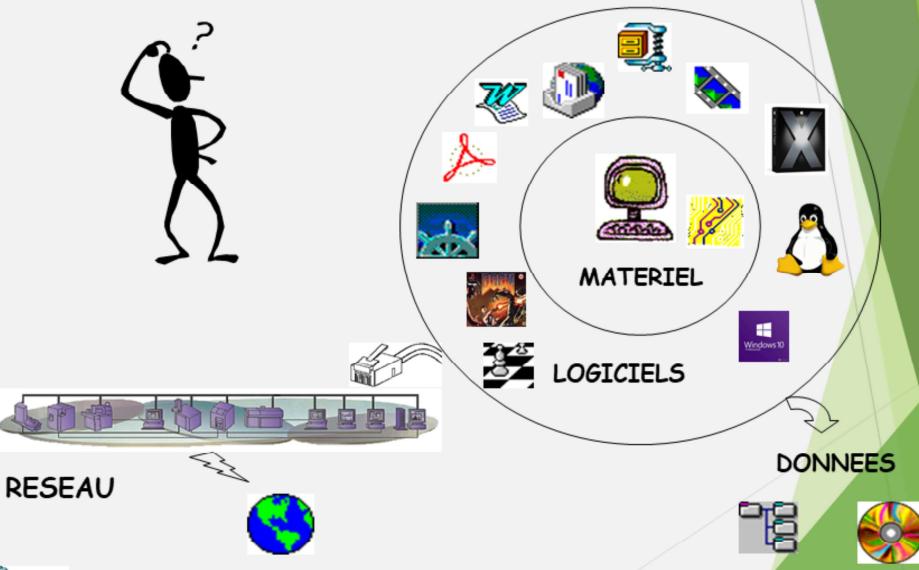
Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'INFORMATION



Un système d'exploitation s'intègre dans l'ensemble d'un système d'information et est en interaction avec tous ses composants.

On rappellera dans cette première partie les éléments d'un système d'information.

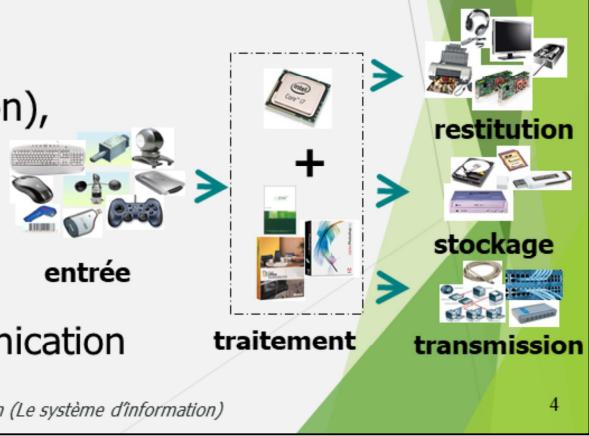
LE SYSTÈME D'INFORMATION

DÉFINITION

Ensemble des **moyens** destinés à répondre à un **besoin** spécifique de traitement et d'exploitation de l'**information**

FONCTIONS DE BASE

- entrée (saisie, acquisition),
- traitement,
- sortie
 - restitution,
 - stockage,
 - transmission/communication



Systèmes d'exploitation (Le système d'information)

4

Besoin : fonctionnalités définies dans le cahier des charges (domaine de l'ingénierie système)

Information : le cœur du problème. L'objectif de tout système informatique est le traitement de l'information.

Moyens : matériel + logiciels (auxquels il faudrait rajouter les « acteurs » intervenant tout au long du cycle de vie du système, mais ce n'est pas le propos de ce cours)

Les fonctions de base du système d'information sont centrées sur le flux de l'information (flux des données)

Dispositifs d'entrée : clavier, souris, webcam, manette, capteurs, réseau, ...

Traitement : matériel (unité centrale : processeur + mémoire) + logiciel

Dispositifs de sortie :

- Restitution : écran, casque audio, imprimante, actionneur, ...
- Stockage : mémoires de masse (disques, mémoires flash, ...)
- Transmission : réseau, ports, bus

LES PROGRAMMES

Bonjour !



0010011000
111001*



* Qu'est-ce qu'il dit?



Systèmes d'exploitation (Le système d'information)

5

Les programmes fournissent la logique du traitement de l'information dans le cas de circuits à fonctionnement programmable (systèmes à processeur).

Voici quelques rappels sur la notion de programme ou comment dialoguer avec le matériel ...

LES PROGRAMMES

DÉFINITIONS

Programme : suite d'instructions destinées à faire exécuter un traitement à un ordinateur.

Programme source : code (texte) d'un programme, écrit par le programmeur dans un langage de programmation.

Programme exécutable : code binaire, résultat de la traduction des instructions des programmes source en langage machine. Spécifique au système et au processeur. Seuls les programmes source écrits dans des langages compilés sont traduits en programmes exécutables.



Un programme a une logique de déroulement, spécifiée par son **algorithme**. Cet algorithme sera transcrit dans un langage de programmation et constituera le **programme source**.

Le processeur est le composant qui va exécuter le programme. Il dispose de ses propres caractéristiques et en particulier de son propre langage (appelé jeu d'instructions). Il ne saura donc exécuter qu'un programme écrit dans son jeu d'instruction. Il s'agit du **programme exécutable**.

LES PROGRAMMES

LE PROGRAMME SOURCE

Description des données + instructions

EXEMPLE

Faire saisir au clavier 2 nombres entiers

Les additionner

Afficher le résultat de l'opération

➤ Données

3 nombres entiers (« variables » a, b et c)

➤ Instructions

Saisie de a et b au clavier (2 opérations d'entrée)

Addition de a et b (opération arithmétique)

Stockage dans c (affectation)

Affichage de c à l'écran (opération de sortie)



Systèmes d'exploitation (Le système d'information)

7

Description des données

Les données manipulées par le programme sont stockées en mémoire centrale (RAM). On parle de variable pour faire référence à l'emplacement mémoire qui leur est réservé.

Une variable est **obligatoirement** caractérisée par :

- son type (type de la donnée : nombre entier ou réel, caractère, adresse, type composé, ...),
- sa taille (sauf dans le cas particulier de l'allocation dynamique, la taille d'une variable est connue dès l'écriture du programme source),
- sa durée de vie (temps d'exécution d'un bloc d'instructions, d'une fonction, du programme, ...),
- sa visibilité (quelles sont les portions de code qui peuvent l'utiliser ?),
- son emplacement de stockage en RAM (selon les caractéristiques précitées, une variable sera stockée à un emplacement différent).

Ces caractéristiques sont fixées dans le programme source, de manière explicite ou implicite selon le langage de programmation utilisé.

Instructions

Les principaux types d'instructions présentes dans un programme source sont

- des affectations (stockage d'une valeur en mémoire),
- des calculs (opérations arithmétiques),
- des opérations d'entrée/sortie (lecture ou écriture d'information sur un périphérique),
- des structures de contrôle (test, boucles).

LES PROGRAMMES

LE PROGRAMME SOURCE

Les langages : une classification complexe

- Dépendance / l'architecture matérielle (oui, non)
- Type de programmation (impérative, fonctionnelle, ...)
- Structuration du code (procédural, objet, ...)
- Mode d'exécution (compilé, interprété)

Python ? C ? C++ ? Java ?



Systèmes d'exploitation (Le système d'information)

8

Les langages de programmation sont tous destinés à transcrire la logique d'un algorithme. Cependant, il existe de nombreux langages.

Selon quels critères choisir un langage de programmation ?

Le choix d'un langage dépend de plusieurs facteurs, plus ou moins objectifs.

Parmi ceux-ci, on pourra citer :

- le type d'algorithme à transcrire,
- les performances (rapidité et empreinte mémoire) requises,
- la portabilité,
- la taille du projet,
- la nécessité de réutilisation de composants logiciels,
- mais aussi la culture de l'entreprise, les préférences des programmeurs, ...

LES PROGRAMMES

LE PROGRAMME SOURCE

Dépendant de l'architecture : l'assembleur

- = jeu d'instructions du processeur
- faible facteur d'expansion : 1 ligne de code par instruction donnée au processeur
- code non portable d'un processeur à un autre
- programmation de « bas niveau », non structurée
- mais ... accès à toutes les fonctionnalités offertes par le processeur



Systèmes d'exploitation (*Le système d'information*)

9

Jeu d'instructions du processeur = langage machine

Le jeu d'instructions est le « vocabulaire » propre à un processeur, c'est à dire la liste des opérations élémentaires qu'il est capable d'effectuer. Ces opérations sont chargées en ROM (microprogramme) ou sont câblées (circuits logiques).

Leur nombre est très variable (de moins de 50 à plus de 300 instructions selon le type de processeur).

On distingue les processeurs **RISC** (Reduced Instruction Set Computer), câblés, avec peu d'instructions très rapides (environ 1 cycle par instruction) des **CISC** (Complex Instruction Set Computer), à microprogramme, avec un jeu d'instructions très complexe.

processeurs ARM : RISC

processeurs Intel : CISC

LES PROGRAMMES

LE PROGRAMME SOURCE

Notre exemple en (pseudo) assembleur ...

```
etiq1: IN AL, @registreEtatClavier  
        CMP AL, valeurClavierPrêt  
        JNZ etiq1  
        IN AL, @registreDonnéeClavier  
        MOV BL, AL  
etiq2: IN AL, @registreEtatClavier  
        CMP AL, valeurReady  
        JNZ etiq2  
        IN AL, @registreDonnéeClavier  
        ADD AL, BL  
        MOV AX, @variableResultat  
etiq3: IN AL, @registreEtatEcran  
        CMP AL, valeurEcranPrêt  
        JNZ etiq3  
        OUT AX, @registreDonnéeEcran
```

Systèmes d'exploitation (Le système d'information)

10

Remarque

Ces quelques lignes donnent une idée du type de code à écrire pour transcrire notre programme en assembleur Intel. Il ne s'agit pas du programme complet.

Les mots-clés IN, OUT, CMP, JNZ, ADD, MOV appartiennent au jeu d'instruction des processeurs Intel.

IN : lecture dans un registre de contrôleur de périphérique

OUT : écriture dans un registre de contrôleur de périphérique

CMP : comparaison de 2 valeurs

JNZ : branchement conditionnel selon le résultat d'une comparaison

ADD : addition

MOV : affectation

LES PROGRAMMES

LE PROGRAMME SOURCE

Langages indépendants de l'architecture

- instructions évoluées (fort facteur d'expansion)
- normalisés => portabilité
- impératifs : Cobol, Fortran, Pascal, ADA, C, Java, C++, C#, PHP, Python, Go, ...
- fonctionnels Lisp, OCaml, Scheme, Erlang, Scala ...
- procéduraux : Pascal, C, ADA 83, ...
- objet : C++, Java, C#, SmallTalk, ADA 95, PHP, Python, R ...



Systèmes d'exploitation (Le système d'information)

11

Les langages dits de « haut niveau » sont indépendants du jeu d'instructions du processeur et constituent une avancée vers la notion de portabilité.

La portabilité totale d'un programme source dépend aussi de l'utilisation stricte des instructions et fonctions standard du langage.

Langages impératifs : logique proche de celle du fonctionnement de la machine (Von Neumann). Basés sur le stockage d'information en mémoire (affectation).

Langages fonctionnels : Basés sur le lambda calcul.

Langages objets : Langages impératifs offrant un regroupement des données et du code qui les manipule dans une même entité (objet), pour une meilleure structuration, réutilisation et maintenabilité.

LES PROGRAMMES

LE PROGRAMME SOURCE

Notre exemple en C ...

```
#include <stdio.h>

int main(void) {
    int a, b, c;
    scanf("%d", &a);
    scanf("%d", &b);
    c = a + b;
    printf("%d\n", c);
    return 0;
}
```

- Directives du préprocesseur
- Instructions du langage
- Appels de fonctions externes



Systèmes d'exploitation (Le système d'information)

12

- Directives du préprocesseur

Le préprocesseur est un outil utilisé spécifiquement dans le traitement d'un programme source écrit en langage C.

Ses directives permettent d'effectuer des manipulations de traitement de texte sur le code source : inclusion de texte, remplacement de chaînes de caractères (« macros »), etc...

Le traitement du code source par le préprocesseur est préalable à la compilation.

- Instructions du langage

Lexique/syntaxe du langage. Elles sont directement traduites en langage machine par le compilateur.

- Appels de fonctions externes

Invocation de code non présent dans le fichier source. Ce code doit être compilé séparément et sera intégré au programme exécutable au moment de l'édition des liens. Le code machine (ou code objet) des fonctions externes peut se trouver dans un fichier objet ou dans une bibliothèque (regroupement de plusieurs fichiers objets en un seul fichier).

Dans l'exemple ci-dessus `printf()` et `scanf()` sont des fonctions qui font partie de la bibliothèque standard du langage C.

LES PROGRAMMES

LE PROGRAMME SOURCE

Notre exemple en Java ...

```
public class Calcul {  
  
    public static void main(String [] arg) {  
        int a, b, c;  
        Scanner in = new Scanner(System.in);  
        a=in.nextInt();  
        b=in.nextInt();  
        c = a + b;  
        System.out.println(c);  
    }  
}
```



Systèmes d'exploitation (Le système d'information)

13

Il y a peu de différences entre ce code source Java et le code source précédent en C, mises à part :

- la présence d'instructions permettant de structurer le code en « classes », et de donner des droits spécifiques d'utilisation au code et aux données (ici, mot-clé « public »),
- la syntaxe d'appel des méthodes (fonctions internes à une classe).

LES PROGRAMMES

LE PROGRAMME SOURCE

Les langages

EN RÉSUMÉ

Assembleur : au moins 15 lignes, non portable
C, Java : 4 à 5 lignes, portable

MAIS ...

Certaines fonctionnalités doivent être codées en assembleur, notamment les configurations processeur et mémoire.



LES PROGRAMMES

L'EXECUTION DU PROGRAMME



Je veux exécuter mon programme ! Que dois je faire ?

Langages compilés

- Traduction du programme source en programme exécutable (langage machine) préalable à l'exécution
- Le système d'exploitation gère l'exécution du programme
- C, ADA, C++, Go, Fortran, ...

Langages interprétés

- Analyse du code source pendant l'exécution
- Un interpréteur gère l'exécution des instructions
- Seul l'interpréteur est connu du système et du matériel
- Python, JAVA, PHP, Perl ...



Systèmes d'exploitation (Le système d'information)

15

Certains langages peuvent être interprétés ou compilés (Lisp, ...), certains uniquement compilés (C, C++) ou uniquement interprétés (PHP), certains sont semi-compilés (Java), c'est-à-dire qu'ils subissent une précompilation qui génère un bytecode destiné à être interprété, pour d'autres (Python), un bytecode est créé lors de la 1^{ère} interprétation, ce qui permet d'améliorer ensuite les performances.

LES PROGRAMMES

LE PROGRAMME EXÉCUTABLE

Spécifique au processeur

- **Taille du mot** : taille des registres de données et d'adresses (32 bits, 64 bits, ...).
- **Endianness** : ordre de stockage des octets en mémoire (little endian, big endian).
- **Alignement mémoire** : l'adresse physique du début du premier octet d'un mot doit être multiple d'une constante.
- **Jeu d'instructions** : ensemble des opérations élémentaires qu'un processeur peut réaliser.



Systèmes d'exploitation (Le système d'information)

16

Taille du mot : Correspond en C à la taille du type int

Endianness

Exemple :

Soit la valeur hexadécimale sur 4 octets : AA BB CC DD

- ordre de stockage en mémoire little endian : DD CC BB AA

- ordre de stockage en mémoire big endian : AA BB CC DD

Alignement mémoire

Certains processeurs nécessitent que les objets soient stockés dans des zones mémoire dont l'adresse physique du premier octet est un multiple d'une constante, en principe puissance de 2. Cette constante est appelée le facteur d'alignement. Si l'adresse d'une information ne respecte pas cet alignement, l'exécution par le processeur d'une instruction qui y accède sera plus lente ou, sur certaines architectures, échouera.

Exemple : Les processeurs 32 bits sont optimisés pour récupérer des données 32 bits, c'est à dire 4 octets. Pour être efficace, une variable sur 4 octets doit être alignée à une adresse divisible par 4.

LES PROGRAMMES

LE PROGRAMME EXÉCUTABLE

Spécifique au système d'exploitation

- **Lancement** : le système d'exploitation lit le fichier contenant le programme exécutable et le charge en mémoire en fonction des informations présentes dans le code exécutable. Il fournit au processeur les informations nécessaires (contexte) au démarrage de son exécution.
- **Appels système** : le programme utilise les services du système, en particulier pour l'accès aux ressources matérielles. Il exécute alors du code spécifique. Ce code est fourni par les API système (appelées directement ou par les fonctions standard des langages).



Le programme exécutable dépend du système d'exploitation :

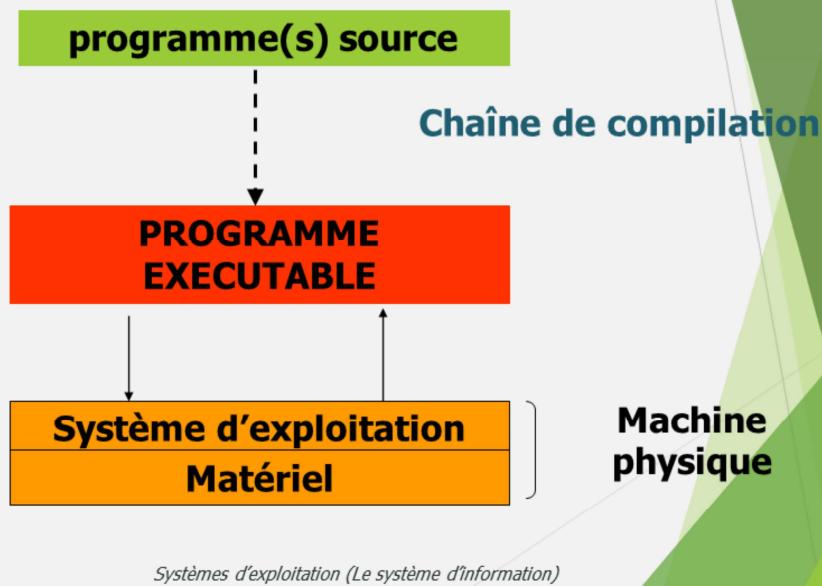
- pour son lancement :
 - chargement en mémoire,
 - démarrage de son exécution
- pour la gestion du cycle de vie de son exécution
- pour l'accès aux périphériques
- pour divers autres services (fichiers, communication interprocessus, etc ...)

Exécution de code spécifique (appels système)

Chaque système d'exploitation fournit un ensemble de fonctions (API). Ces fonctions peuvent être appelées directement dans les programmes utilisateurs. Elles sont aussi invoquées par les fonctions standard des langages. Elles ne contiennent généralement pas le traitement complet d'accès au matériel, mais provoquent un passage en mode noyau et un branchement vers l'exécution de code contenu dans le noyau du système. On parle de « fonction enveloppe » ou « wrapper » d'appel système.

LES PROGRAMMES

LA CHAÎNE DE COMPILEMENT



Le programme source est transformé en programme exécutable par un ensemble d'outils logiciels (chaîne de compilation).

Comme le programme exécutable va être pris en charge par une « machine physique », il doit être spécifique à un système d'exploitation et à une architecture matérielle.

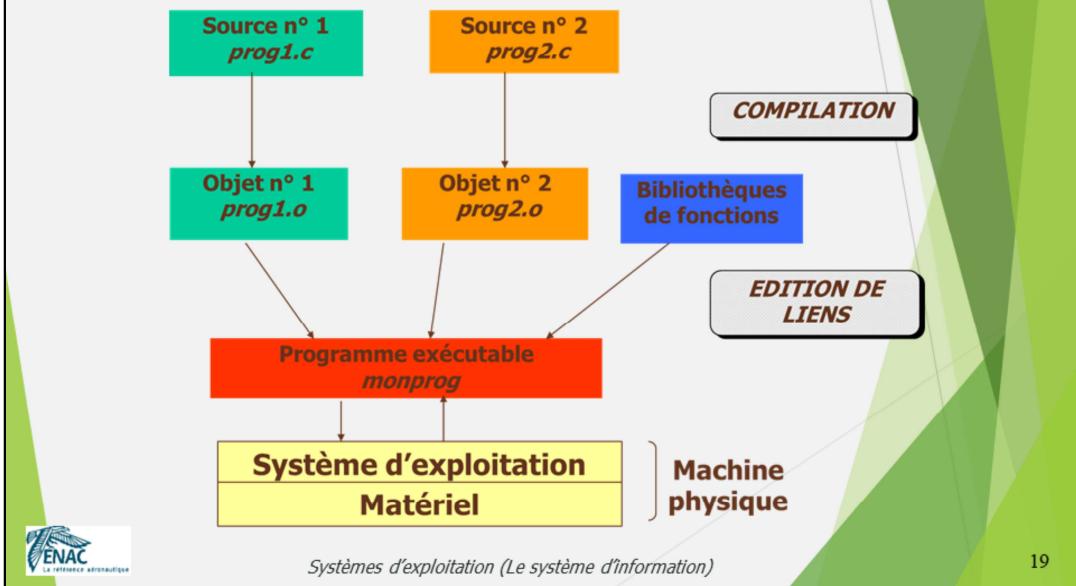
La chaîne de compilation est donc aussi spécifique à la machine physique. Elle est composée d'un **compilateur** et d'un **éditeur de liens**.

Dans le cas d'une « compilation croisée », c'est-à-dire la compilation sur une machine physique d'un exécutable destiné à un autre type de machine physique, la chaîne de compilation est spécifique aux 2 machines : la machine de développement et la machine cible.

LES PROGRAMMES

LA CHAINE DE COMPILEATION

Exemple



Dans cet exemple, le code écrit en langage C est contenu dans 2 fichiers source (*prog1.c* et *prog2.c*).

Chaque fichier source doit être compilé séparément et donnera un fichier objet. L'édition des liens regroupera le code des 2 fichiers objets et y ajoutera le code objet des fonctions « externes » appelées (édition de liens statique). Ces fonctions externes sont contenues dans une bibliothèque.

Remarque

Dans le cas d'une édition de liens avec des **bibliothèques dynamiques** (par exemple, fichier *.so* sous Linux, fichier *.dll* sous Windows), le code des fonctions ne sera pas inclus dans l'exécutable. Il y aura uniquement une référence permettant de localiser la fonction dans une bibliothèque dynamique chargée au cours de l'exécution.

Si plusieurs processus utilisent la même bibliothèque dynamique simultanément, celle-ci ne sera chargée qu'une seule fois en mémoire.

LES PROGRAMMES

LA CHAINE DE COMPILEATION

Le compilateur

- Logiciel spécifique (langage + système + matériel)
- Chaque fichier source est compilé individuellement
- Contrôle et traduit le programme source en langage machine (création d'un fichier objet binaire par fichier source)
- Les références extérieures (code des fonctions et définition de variables globales) issues d'un autre fichier source ne sont pas résolues à la compilation



En règle générale, la compilation s'effectue en 6 phases :

L'analyse lexicale : Identification des séquences de caractères, élimination des éléments inutiles (commentaires,...), détection des erreurs sur les identificateurs et les caractères, stockage des identificateurs et de leurs caractéristiques dans la table des symboles.

L'analyse syntaxique : Contrôle des éléments par rapport à la syntaxe, génération de l'arbre syntaxique, détection des erreurs : parenthèses, construction des blocs, ...

L'analyse sémantique : Analyse de la signification des phrases du langage, vérification de la concordance des types, détection des erreurs sur la déclaration des identificateurs, la compatibilité des types, ...

La génération du code intermédiaire : 1ère étape de la génération du code objet : génération d'un code pour une *machine abstraite* : pas de référence explicite aux registres de la machine cible.

L'optimisation du code : A pour objectif de rendre le code généré plus rapide à l'exécution et moins encombrant en mémoire. Tient compte de la machine cible (processeur)

La génération du code objet : Traduction de chaque instruction du code intermédiaire en une séquence d'instructions propres à la machine cible, attribution des positions en mémoire pour les données et les instructions du programme et gestion de l'allocation des registres CPU. Le code généré est *relogable* (adressage relatif à l'adresse 0).

Le programme objet généré par la compilation contient donc le code machine des instructions du programme, la liste des références (fonctions appelées et variables globales) extérieures, la liste des adresses référençables depuis l'extérieur du module.

LES PROGRAMMES

LA CHAINE DE COMPILEMENT

L'éditeur de liens

- Traite les références extérieures non résolues par la compilation en localisant le code des fonctions et les variables globales dans les différents fichiers objets et bibliothèques
- Regroupe toutes les ressources dans un seul fichier : crée le programme exécutable



Systèmes d'exploitation (*Le système d'information*)

21

L'édition des liens combine plusieurs fichiers objets issus de la compilation en un seul programme exécutable.

Il localise les symboles des références extérieures (fonctions, variables globales) dans les différentes bibliothèques et fichiers objets, et les stocke dans un même fichier au format exécutable requis par le système.

Après la compilation, chaque code objet commençait à l'adresse mémoire 0. L'éditeur de liens ordonne les fonctions et le programme principal et produit un module final relatif à l'adresse mémoire 0 (code relogable).

LES PROGRAMMES

LA CHAINE DE COMPILEMENT

Notre exemple en C ...

```
#include <stdio.h>  
  
int main(void) {  
    int a, b, c;  
    scanf("%d", &a);  
    scanf("%d", &b);  
    c = a + b;  
    printf("%d\n", c);  
    return 0;  
}
```

prog.c

compilation

prog.o

*édition
de liens*

prog

Bibliothèque C

code objet de printf

code objet de scanf



Systèmes d'exploitation (Le système d'information)

22

prog.c : fichier contenant le programme source

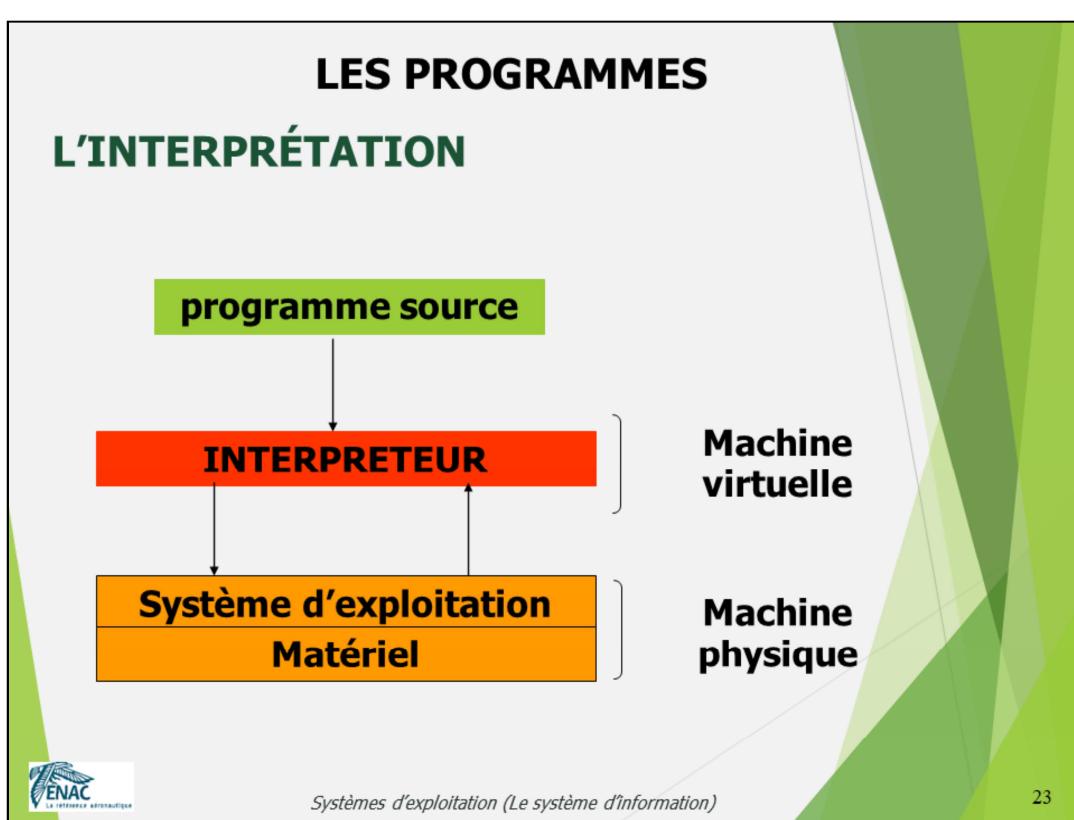
prog.o : fichier contenant le programme objet

prog : fichier contenant le programme exécutable

Bibliothèque C : fichier contenant le code objet des fonctions standard du langage C

LES PROGRAMMES

L'INTERPRÉTATION



Un interpréteur est un programme exécutable qui, pour chaque ligne du programme source, effectue les contrôles nécessaires puis exécute lui-même les instructions.

Par exécution, on comprendra que le code de chaque instruction du langage est déjà présent sous forme exécutable dans l'interpréteur et que l'interprétation d'une instruction consiste à appeler ce code.

Remarque

« Il existe un autre type d'interpréteur, les compilateurs **JIT** (Just In Time). Leur principe est de compiler à la volée le code source en binaire natif. Le gros intérêt est que le binaire est exécuté directement par l'ordinateur, il n'a pas besoin de machine virtuelle. Pourquoi tous les interpréteurs ne sont pas des compilateurs JIT ? Parce que c'est loin d'être simple à mettre en œuvre. Les compilateurs natifs sont habituellement assez lents; cela ne pose pas le moindre problème, car quand on compile du code C ou C++, le but est d'obtenir un programme dont l'exécution sera la plus rapide possible; ils intègrent d'ailleurs plein d'optimisations qui ralentissent la compilation mais qui accélèrent l'exécution.

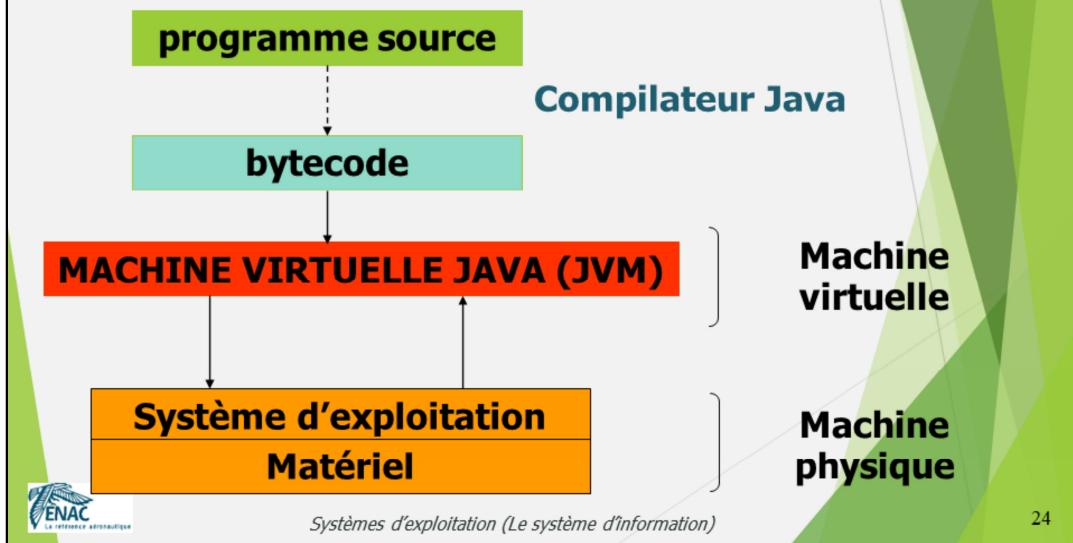
Un compilateur JIT doit donc être capable de compiler nativement dans un délai très court, afin que l'exécution d'un programme interprété ne semble pas systématiquement ralenti. »

(extrait d'un article www.geek-directeur-technique.com)

LES PROGRAMMES

JAVA : UN CAS PARTICULIER

Langage interprété ou presque ...



Java est un langage **interprété**, donc indépendant de la machine physique. C'est aussi un langage précompilé : le compilateur Java génère un pseudo-code appelé **bytecode** indépendant du système d'exploitation et du matériel. Cette phase de précompilation permet un gain de temps lors de l'interprétation en assurant les contrôles lexicaux et syntaxiques, en éliminant les commentaires et en compactant le code fourni à l'interpréteur . Ce bytecode est interprété par la machine virtuelle JAVA (JVM), logiciel porté sur la plupart des systèmes d'exploitation. Le système JAVA est portable : compilateur écrit en JAVA, environnement d'exécution écrit en C ANSI, conforme POSIX.

Systèmes d'exploitation

Le système d'exploitation

Notions fondamentales



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Le système d'exploitation)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

- #### **Notions fondamentales**
- Gestion de la mémoire
 - Gestion du processeur
 - Gestion des entrées-sorties
 - Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'EXPLOITATION



NOTIONS FONDAMENTALES

RÔLES DU SYSTÈME D'EXPLOITATION

Assurer l'interface entre l'utilisateur et le matériel

- interface utilisateur, commandes, outils
- lancement de l'exécution des programmes

Assurer l'interface entre le programme et le matériel

- gérer son déroulement en optimisant l'utilisation des ressources matérielles :
 - gestion du processeur
 - gestion de la mémoire centrale
 - gestion des entrées/sorties
 - gestion des fichiers



Systèmes d'exploitation (Le système d'exploitation)

4

Le système d'exploitation est un logiciel d'interface à plusieurs niveaux :

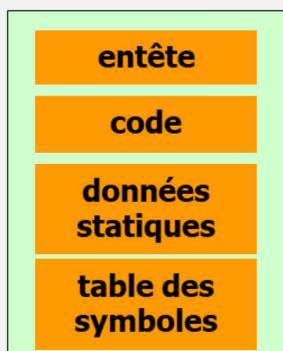
- interface entre l'utilisateur et le système informatique (IHM),
- interface entre les programmes et le matériel.

NOTIONS FONDAMENTALES

LANCLEMENT D'UN PROGRAMME

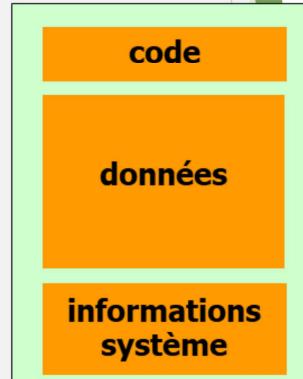


un fichier exécutable



Chargement en mémoire centrale

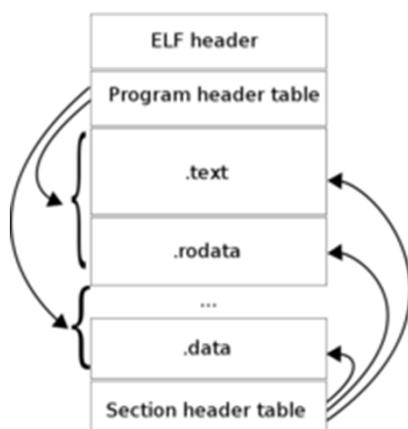
un processus



Le **fichier exécutable** est stocké sur une mémoire permanente (disque, ROM, etc ...).

L'entête (« header ») du programme exécutable contient des informations permettant au système de le charger en mémoire.

Exemple : structure d'un fichier exécutable au format ELF (Unix)



Le **processus** est créé en RAM par le système d'exploitation.

NOTIONS FONDAMENTALES

NOTION DE PROCESSUS

Image mémoire d'un programme en cours d'exécution

- Créé et géré par le système d'exploitation
- Durée de vie = durée de l'exécution du programme



Systèmes d'exploitation (Le système d'exploitation)

6

On parle de **processus** pour désigner un programme en cours d'exécution.

Un processus est caractérisé par son espace d'adressage (espace mémoire de ses instructions et de ses données), associé à un environnement d'exécution processeur (registres) et à des informations de gestion système.

On complètera donc la définition d'un processus de la manière suivante:

Un processus est l'image mémoire d'un programme en cours d'exécution.

Remarque : les instructions (code) d'un même programme peuvent être exécutées simultanément dans plusieurs processus (cas d'un programme « lancé » plusieurs fois en parallèle). Si le système et le programme lui-même sont prévus pour assurer correctement cette exécution parallèle, on parlera alors de programme ou de code **réentrant**.

NOTIONS FONDAMENTALES

NOTION DE PROCESSUS : Illustration



Gestionnaire des tâches Windows 10

Gestionnaire des tâches						
Fichier Options Affichage						
Processus Performance Historique des applications Démarrage Utilisateurs Détails Services						
Nom	PID	Statut	Nom d'util...	Processeur	Mémoire (plage de travail privée active)	
aesm_service.exe	14728	En cours d'exécution	Système	00	24 Ko	
ApplicationFrameHo...	7824	En cours d'exécution	luterjo	00	9 536 Ko	
armsvc.exe	5076	En cours d'exécution	Système	00	28 Ko	
AuthManSvr.exe	13256	En cours d'exécution	luterjo	00	1 972 Ko	
backgroundTaskHos...	1564	Interrompu	luterjo	00	0 Ko	
browser_broker.exe	16296	En cours d'exécution	luterjo	00	1 184 Ko	
Calculator.exe	372	Interrompu	luterjo	00	0 Ko	
chrome.exe	2520	En cours d'exécution	luterjo	00	68 964 Ko	

Nom du fichier exécutable Identifiant du processus (PID) Place occupée en mémoire

 Systèmes d'exploitation (Le système d'exploitation) 7

NOTIONS FONDAMENTALES

CARACTÉRISTIQUES

- Matériel : architecture machine cible (x86, ARM, MIPS, ...)
- Utilisateurs
- Tâches
- Mode de fonctionnement des applications

**Le type de système dépendra de ces caractéristiques
-> critères de choix**



NOTIONS FONDAMENTALES

MATÉRIEL CIBLE

- Ordinateur individuel : Windows, MacOs, Linux
- Serveur : Unix/Linux, Windows
- Téléphone : Androïd, iOS, Windows ...
- Microcontrôleur : Linux embarqué, ChibiOS, FreeRTOS ...
- Routeur réseau : Linux, Cisco IOS, Junos, ...



Systèmes d'exploitation (Le système d'exploitation)

9

Pour chaque type de matériel, il existe des systèmes d'exploitation, standard ou spécifiques.

La liste ci-dessus n'est pas exhaustive mais permet de voir que la plus grande part du marché des systèmes d'exploitation est détenue par les systèmes de type Unix (Linux, Androïd, MacOs) et Windows.

NOTIONS FONDAMENTALES

MULTI-UTILISATEUR



Chaque utilisateur possède un compte et un profil

- Authentification : nom + mot de passe
- Espace de stockage
- Protection des ressources (droits d'accès)
- Personnalisation de l'environnement



Systèmes d'exploitation (Le système d'exploitation)

10

NOTIONS FONDAMENTALES

MULTITÂCHE

Plusieurs tâches (= suites d'instructions) peuvent s'exécuter "simultanément", se synchroniser et échanger des informations.

Intérêt

- Confort d'utilisation
- Gain de temps (parallélisation des traitements)
- Réactivité
- Architecture d'applications modulaires et évolutives



NOTIONS FONDAMENTALES

MULTITÂCHE

Définitions

- Un **thread** est un flot d'instructions (= tâche). Il s'exécute toujours dans le cadre d'un processus. Il peut être exécuté en parallèle avec d'autres threads.
- Un système multitâche **temps partagé** gère le partage du temps de travail du processeur entre les différentes tâches en cours
- Un système multitâche **préemptif** peut décider de suspendre ou d'interrompre l'exécution d'une tâche



Littéralement, un thread est un « fil » d'instructions. Plusieurs threads appartenant au même processus peuvent s'exécuter parallèlement.

Tous les threads d'un même processus partagent le même espace d'adressage de données (data et bss). Chaque thread possède sa pile et son contexte d'exécution.

Intérêt

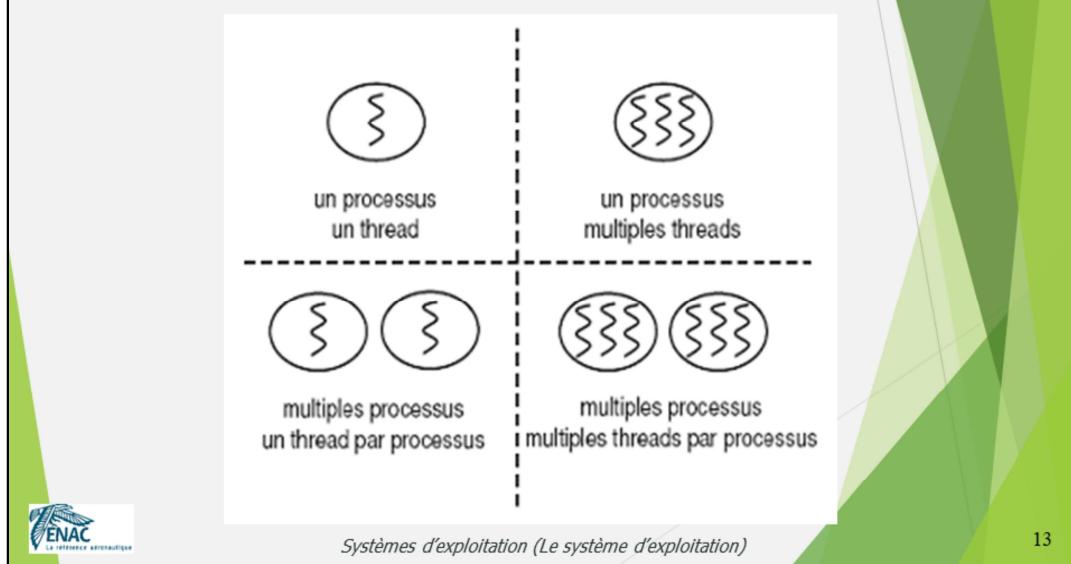
- L'intérêt principal d'une application multitâche est sa réactivité.
- Systèmes multi-processeurs : un programme peut être facilement découpé en fonctions pouvant être exécutées en parallèle (un thread par processeur).
- Systèmes mono-processeurs : le lancement d'un thread étant moins coûteux en temps et en ressources que le lancement d'un processus, le découpage d'un programme en threads permet d'améliorer les performances.

Les moyens de communication entre threads sont plus faciles à mettre en œuvre. Cependant, l'accès aux données communes reste à synchroniser.

NOTIONS FONDAMENTALES

MULTITÂCHE

Modèles d'exécution



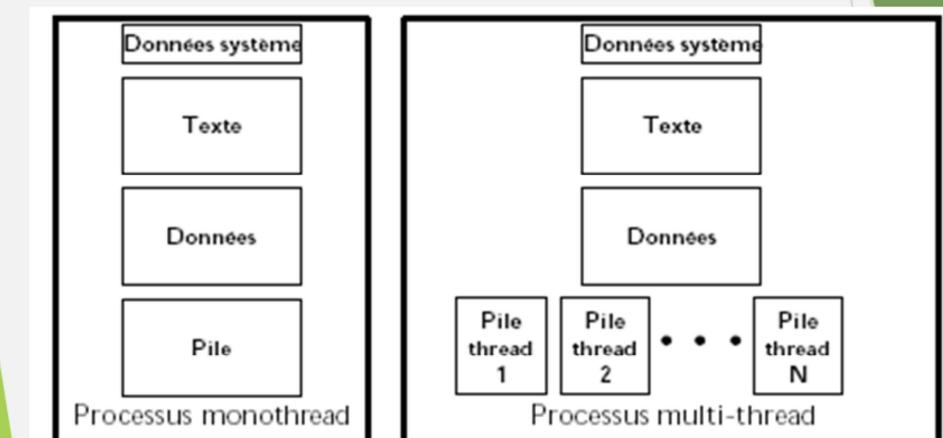
Il existe deux types de répartition des tâches sur une machine multiprocesseur :

- asymétrique (ASMP) : un processeur est réservé pour le système, les applications se partagent les autres (relation maître/esclave).
- symétrique (SMP) : le système d'exploitation et les applications sont répartis sur les différents processeurs.

NOTIONS FONDAMENTALES

MULTITÂCHE

Modèles mémoire



La zone de **Texte** contient le code du programme.

La zone de **Données** correspond à la zone de données accessible (partagée) par toutes les fonctions du processus (variables globales, variables allouées dynamiquement).

La zone de **Pile** contient les variables locales de la fonction en cours d'exécution. On notera que chaque thread possède sa propre pile.

La plupart des systèmes d'exploitation distinguent 2 piles pour des raisons de protection des données :

- La pile utilisateur : variables locales aux fonctions s'exécutant en mode utilisateur
- La pile système : variables locales aux appels système

NOTIONS FONDAMENTALES

MULTITÂCHE

Processus vs. thread

Eléments communs aux threads d'un processus	Eléments spécifiques à chaque thread
Espace d'adressage	Compteur d'instructions
Variables globales	Pile (variables locales)
Fichiers ouverts	Etat (cycle de vie)
Signaux et timers	Priorité
Droits d'accès	



Systèmes d'exploitation (Le système d'exploitation)

15

Niveau processus

Espace d'adressage : tout l'espace d'adressage du processus est commun aux threads. Cela signifie que le code de tous les threads peut accéder aux zones de données (variables globales et locales) du processus.

Fichiers ouverts : la table des descripteurs de fichiers est commune à tous les threads d'un processus. Cela signifie que si plusieurs threads utilisent le même descripteur, la position courante dans le fichier est partagée. Si on ne veut pas ce fonctionnement, chaque thread devra ouvrir le fichier pour obtenir son propre descripteur.

Droits d'accès : un processus est lancé par un utilisateur. Lors de l'accès à une ressource (fichier, ...), les droits d'accès à cette ressource sont vérifiés par rapport à cet utilisateur et à ses groupes. Ceci est valable pour le code de tous les threads du processus.

Niveau thread

Compteur d'instructions = prochaine instruction à exécuter. L'exécution de chaque thread est indépendante de celle des autres. Lors d'une commutation de contexte, le compteur d'instructions est sauvegardé pour reprendre le thread à la prochaine instruction.

Pile : chaque thread dispose de ses propres variables locales.

Etat : le thread est une unité d'ordonnancement. Chaque thread dispose de son propre cycle de vie et a donc un état (prêt, actif, en attente)

Priorité : selon le mécanisme d'ordonnancement utilisé, il est possible d'attribuer une priorité à chaque thread.

NOTIONS FONDAMENTALES

MODE DE FONCTIONNEMENT

Le choix d'un système répond aux questions suivantes

- Quel type d'utilisation fait-on du système ?
- Quelles applications veut-on faire tourner ?
- Quelles sont les contraintes de fonctionnement de ces applications ?

Systèmes interactifs, batch, temps réel ...



NOTIONS FONDAMENTALES

MODE DE FONCTIONNEMENT

Système interactif

L'utilisateur interagit avec le système et les applications via des périphériques (écran, clavier, souris, ...)

- Le temps de réponse moyen* doit être le plus faible possible

(*) le **temps de réponse** d'une tâche est la durée séparant l'instant de soumission de la tâche de sa fin d'exécution



Les systèmes interactifs poursuivent deux buts principaux :

- ils doivent **maximiser le taux d'occupation du processeur**, c'est-à-dire le rapport entre le temps où le processeur est actif et le temps total écoulé. En théorie, ce taux peut varier entre 0% et 100 % ; dans la pratique, on peut observer un taux d'occupation variant entre 40 % et 95 %.
- ils doivent **minimiser le temps de réponse des tâches**, c'est-à-dire la durée séparant l'instant de soumission de la tâche de sa fin d'exécution. Au mieux (« best case »), le temps de réponse peut être exactement égal au temps d'exécution de la tâche, lorsque celle-ci a immédiatement été élue et s'est exécutée sans être interrompue.

NOTIONS FONDAMENTALES

MODE DE FONCTIONNEMENT

Système off-line (batch)

Pas d'interaction homme-machine pendant l'exécution.
Les données sont lues et écrites sur des supports de masse.

- Le débit* doit être le plus important possible

(*) *le débit du processeur est le nombre moyen de tâches traitées par unité de temps*



Le but de ce type de système est de **maximiser le débit du processeur**, c'est-à-dire le nombre moyen de tâches traitées par unité de temps.

NOTIONS FONDAMENTALES

MODE DE FONCTIONNEMENT



Système temps réel

Systèmes autonomes, embarqués *

- Les tâches sont activées par un événement extérieur
- Le temps de calcul est critique : le temps de réponse à un événement extérieur doit être garanti (prédictibilité du temps de réponse) :
 $\text{temps réponse} < \text{constante de temps du procédé}$
- La réaction à une situation doit être déterministe

(*) *Un système embarqué est un système électronique, piloté par un logiciel et complètement intégré au système qu'il contrôle, prévu pour fonctionner sur des petites machines ou des appareils électroniques autonomes*



La contrainte sur les systèmes temps réels est donc le déterminisme des temps de réponse alors que les autres systèmes se doivent de faire au mieux (politique du « best effort »). Ils doivent garantir le respect des contraintes temporelles des tâches.

Exemples : traitement de données radar dans un véhicule mobile, contrôle de processus industriel robotisé, ABS, ...

On classifie généralement les systèmes temps réel en « temps réel dur » (hard) ou « temps réel mou » (soft).

Conséquences d'un non respect des contraintes temporelles associées :

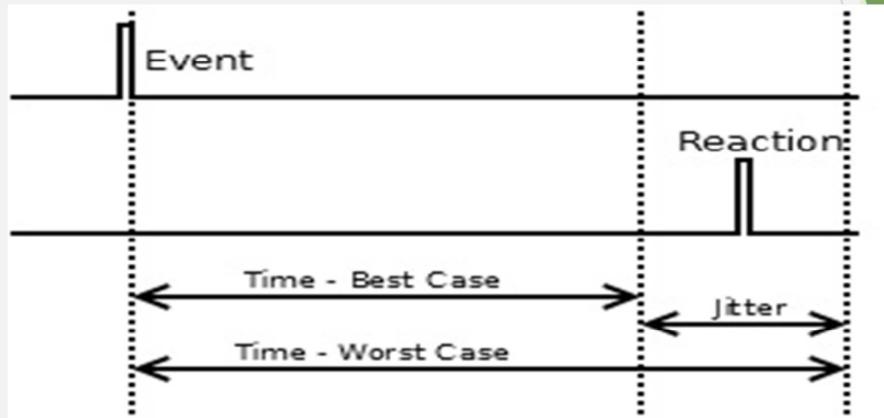
Temps réel « mou » ou « souple » : ne pas respecter la contrainte dégrade la qualité de service. Exemple : streaming vidéo, routeur wifi, ...

Temps réel « dur » ou « strict » : ne pas respecter la contrainte cause un échec total du système. Exemples : véhicule autonome, système avion, missile, pacemaker, ...

NOTIONS FONDAMENTALES

MODE DE FONCTIONNEMENT

Système temps réel



Systèmes d'exploitation (Le système d'exploitation)

20

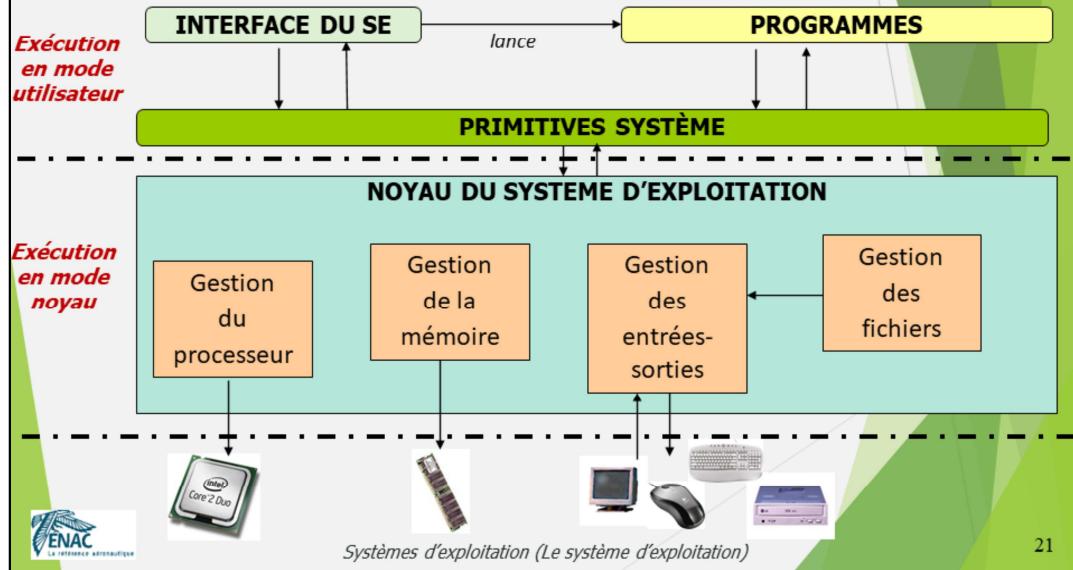
Un système temps réel garantit que la réaction de l'application à un événement aura lieu dans un intervalle de temps connu (jitter) et donc que le deadline ne sera pas dépassé. Ce jitter doit être le plus faible possible.

Pour cela, il a lui-même un comportement prédictible et fournit aux concepteurs d'applications les outils leur permettant de maîtriser les temps d'exécution des programmes.

...

NOTIONS FONDAMENTALES

INTERACTIONS



21

Le système d'exploitation est en interaction avec l'utilisateur par son interface, avec les programmes utilisateurs auxquels il rend des services grâce aux primitives système et avec le matériel dont il assure la gestion.

Une primitive système (ou appel système) permet à un programme utilisateur de demander un service au système d'exploitation, donc de déclencher l'exécution de code système.

Le code d'une tâche peut accéder à son espace d'adressage personnel à tout moment de son exécution. Lors qu'elle effectue un appel système, elle pourra de plus accéder à la mémoire du système d'exploitation : code des primitives système, tables du système (fichiers, processus, ...), pile système et disposer du jeu d'instruction complet du processeur.

Nous voyons ici apparaître deux **modes d'exécution** qui permettent d'assurer la protection de la mémoire du système d'exploitation. Les instructions des processus applicatifs (exécutées en **mode utilisateur**) ne peuvent manipuler que les zones de données utilisateur. Les instructions du système d'exploitation (code des appels système) doivent pouvoir accéder à toute l'étendue de la mémoire (elles sont exécutées en **mode superviseur**, ou **mode noyau**). En mode utilisateur, une tentative d'exécution d'une instruction privilégiée ou un accès à une zone mémoire réservée provoque une erreur du processeur (exception).

La présence de plusieurs modes d'exécution est un mécanisme proposé par les processeurs (par exemple, les processeurs Intel offrent 4 modes d'exécution différents). La plupart des systèmes d'exploitation n'en utilisent que deux.

Systèmes d'exploitation

Le système d'exploitation

Gestion de la mémoire



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- **Principes et mise en œuvre**

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Gestion de la mémoire)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

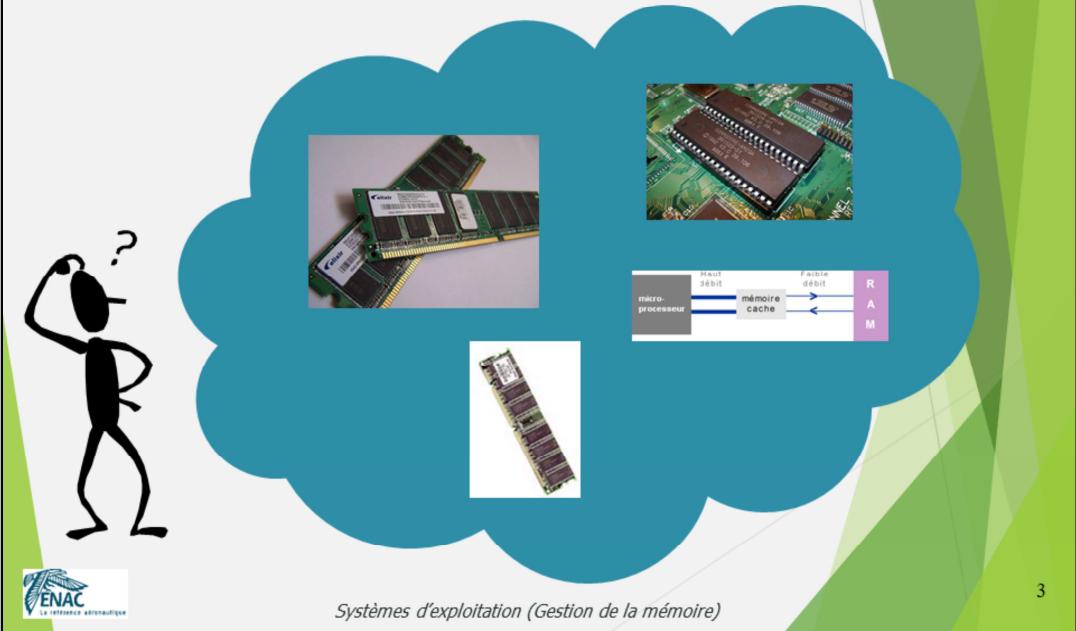
- Notions fondamentales
- Gestion de la mémoire**
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'EXPLOITATION

GESTION DE LA MÉMOIRE



Cette partie présente :

- la gestion de la RAM,
- les mécanismes de mémoire cache,
- l'utilisation de la ROM au démarrage du système.

GESTION DE LA MÉMOIRE

CAHIER DES CHARGES



Lancer un programme = charger un processus en mémoire



Système multitâches

- Plusieurs processus cohabitent (système et applications)
 - > **gestion** de l'espace
 - > **optimisation** et **extension**
- le code des programmes lit et écrit dans la mémoire
 - > **protection** des informations



Systèmes d'exploitation (Gestion de la mémoire)

4

Tout programme doit être chargé en mémoire (RAM) avant de démarrer son exécution : création du processus par le système d'exploitation.

GESTION DE LA MÉMOIRE

ZONES MÉMOIRE D'UN PROCESSUS



Espace d'adressage utilisateur

code + données (pile, tas, données statiques)

Bloc de contrôle

Informations permettant au système de gérer l'exécution du processus

Pile système



Systèmes d'exploitation (Gestion de la mémoire)

5

La structure interne d'un processus peut varier d'un système à l'autre, mais, dans la mesure où il est composé de code, de données et des informations de gestion nécessaires au système d'exploitation, le principe général reste le même.

Sous Linux, un processus se compose :

- d'un espace d'adressage utilisateur linéaire, comportant le code (instructions exécutables) et les zones de données "visibles" par le programmeur,
- d'un bloc de contrôle,
- d'une pile (zone de données) système utilisée lors de l'exécution des procédures du noyau (appels système).

Le bloc de contrôle contient les informations générales sur le processus permettant au système de gérer son exécution, en particulier : état, lieu en mémoire, priorité, temps CPU, signaux, numéro, propriétaire, adresse physique, taille, ...

Une zone de mémoire particulière est associée à ce bloc de contrôle. Elle contient des données nécessaires au système pour gérer l'exécution du processus: adresse du bloc de contrôle, contexte d'exécution, descripteurs de fichiers ouverts par le processus, paramètres des appels systèmes en cours, répertoire de travail, attributs utilisateur, nombre de segments de mémoire partagée ...

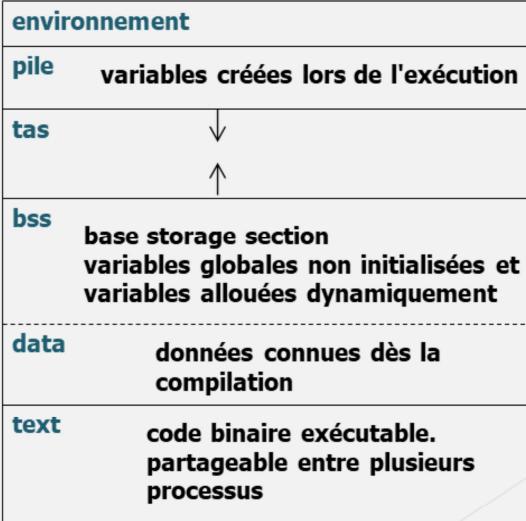
GESTION DE LA MÉMOIRE

ESPACE D'ADRESSAGE D'UN PROCESSUS

taille max. = 3 Go
(noyaux 32 bits)



0



Systèmes d'exploitation (Gestion de la mémoire)

segment de pile

segment de données

segment de code



6

Sous Linux, la taille maximale de l'espace d'adressage d'un processus dépend de l'architecture du processeur.

En architecture 32 bits, elle est limitée à 4 Go (dont 3 Go utilisateur et 1 Go pour les données noyau).

En architecture 64 bits, elle est limitée à 1 To.

GESTION DE LA MÉMOIRE



L'ADRESSAGE

Chaque instruction ou donnée est accessible par son adresse

Adresse mémoire physique



Mémoire centrale divisée en "cellules"

1 cellule = 1 mot mémoire

adresse d'un mot = emplacement physique

Accès à une cellule = adressage

Pour toute opération de lecture ou d'écriture

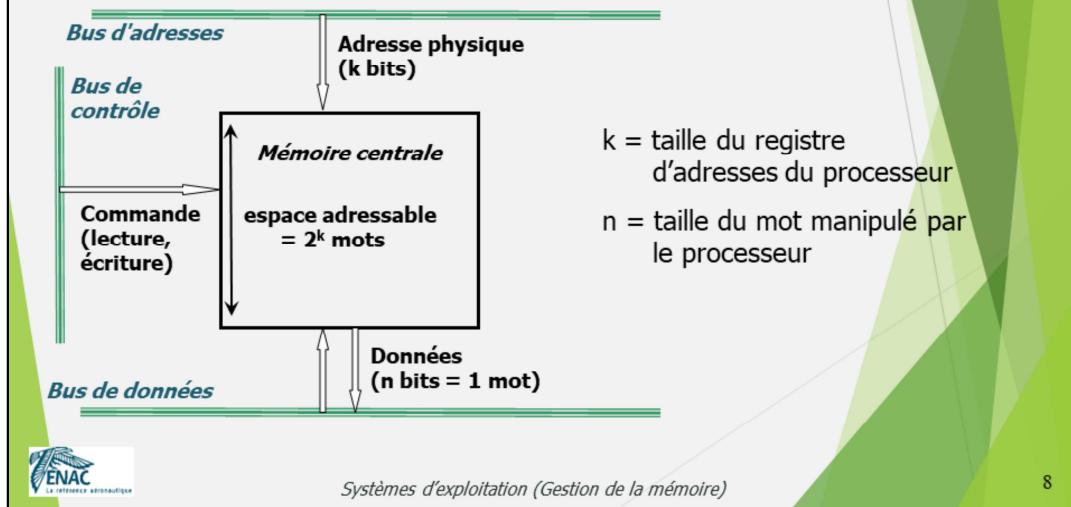
Différents modes d'adressage : direct, indirect, immédiat, implicite, indexé, basé, absolu ou relatif.

GESTION DE LA MÉMOIRE



L'ADRESSAGE

Communication entre le processeur et la RAM



Rappel :

La circulation des informations dans un système informatique se fait via les bus.

Le **bus système** (ou FSB, Front Side Bus) assure la communication entre le processeur et la mémoire. Il véhicule trois types d'informations : les adresses, les données et les commandes.

Il regroupe donc :

- le **bus d'adresses** : transmission par le processeur de l'adresse mémoire de lecture ou d'écriture. Sa largeur doit être cohérente avec celle du registre d'adresse.
- le **bus de données** : transfert entre processeur et mémoire centrale des mots lus ou écrits.
- le **bus de contrôle** : transmission par le processeur de l'opération à effectuer (commande de lecture, d'écriture, de synchronisation).

GESTION DE LA MÉMOIRE

L'ADRESSAGE



Le code exécutable ne contient - a priori - pas d'adresse physique (code relogable)

➤ Code relogé au chargement (translation d'adresses)

Adresses calculées au chargement en mémoire selon les indications du code exécutable. Difficile d'assurer la protection.

➤ Utilisation de registres de base et de limite

Adresses calculées au fur et à mesure en ajoutant le contenu du registre de base à l'adresse générée par l'éditeur de liens.

Utilisation d'un registre de limite pour assurer la protection.



Que deviennent les adresses présentes dans le programme exécutable lors du chargement en mémoire ?

2 techniques sont envisageables : translation d'adresse ou décalage par rapport à une adresse de base.

La translation d'adresses implique le recalcul des adresses contenue dans l'exécutable au moment du chargement (adresse physique initiale de chargement + adresse contenue dans le code). Les conséquences de cette technique sont un temps de chargement important et l'impossibilité de contrôler en cours d'exécution la validité d'une adresse.

L'utilisation de registres de base et de limite permet de calculer les adresses au fur et à mesure de leur utilisation (adresse de base + adresse contenue dans le code). La présence du registre de limite permet de contrôler si l'adresse appartient à une plage autorisée.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION (historique)



Partitions de taille fixe

- Taille des partitions figée au démarrage du système
- 1 processus par partition



Systèmes d'exploitation (Gestion de la mémoire)

10



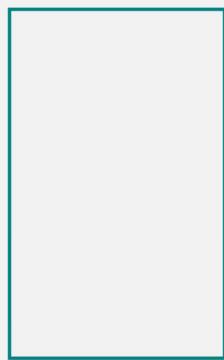
GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION (historique)



Partitions de taille variable

- Taille des partitions adaptée aux processus
- 1 processus par partition



Systèmes d'exploitation (Gestion de la mémoire)



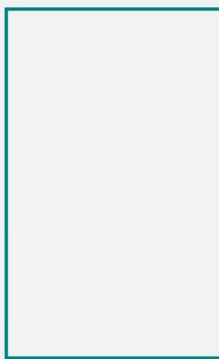
GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION (historique)



Partitions de taille variable

- Taille des partitions adaptée aux processus
- 1 processus par partition



Systèmes d'exploitation (Gestion de la mémoire)



GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION (historique)



Conclusion

- Compromis difficile entre l'optimisation de l'occupation mémoire et le temps d'exécution des programmes ...
- Quelle solution adopter ?



Systèmes d'exploitation (Gestion de la mémoire)

GESTION DE LA MÉMOIRE



TECHNIQUES DE GESTION

La segmentation

- Optimise l'occupation
- Découpage du programme en segments de taille variable
- Segments non obligatoirement contigus en mémoire
- Qui décide du découpage ?



Systèmes d'exploitation (Gestion de la mémoire)

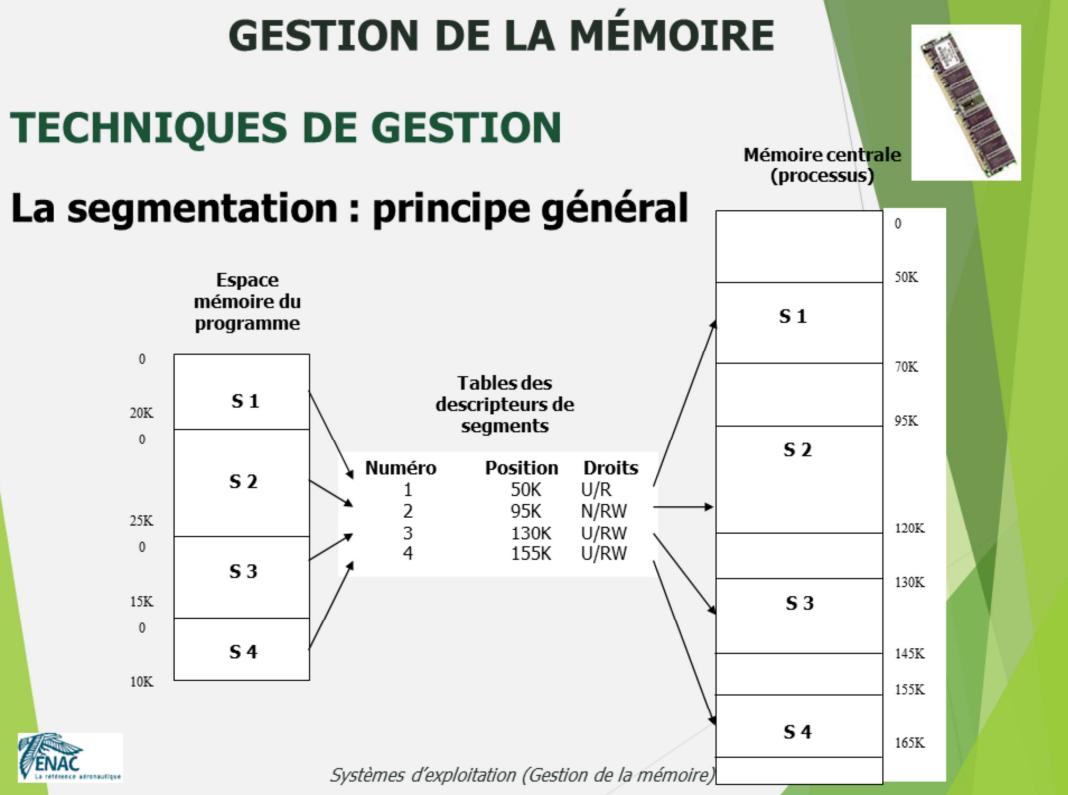
Le nombre de segments d'un programme dépend de l'architecture matérielle.

Par exemple, sur un processeur Intel, un processus peut occuper jusqu'à 6 segments : un segment pour le code, les autres pour les données statiques et la pile.

La mise en œuvre de la segmentation est faite par le système d'exploitation qui exploite tout ou partie des possibilités offertes par le processeur.

Le choix d'un modèle de segmentation dépend des contraintes de performance et de sécurité requises :

- une utilisation complète des possibilités offertes assure une sécurité maximale en augmentant la protection des zones de données,
- en revanche, plus un processus est segmenté, plus la commutation de contexte est lourde, ce qui peut causer un ralentissement important de l'exécution.



On présentera ici 3 modèles pouvant être implémentés par un système d'exploitation fonctionnant sur une architecture Intel.

Basic Flat Model

Il est physiquement impossible de supprimer la segmentation de l'espace d'adressage du processeur. Au moins deux descripteurs de segments, un pour les données et un pour le code, sont implicitement utilisés.

Cependant, pour pouvoir exécuter rapidement des systèmes sans segmentation, les segments peuvent être autorisés à être mémorisés dans tout l'espace d'adressage physique.

Protected Flat Model

Ce modèle, repose sur le même principe que le Basic Flat Model. Il offre aussi une très grande liberté quant aux implantations des segments en mémoire physique, mais les limite toutefois aux seules adresses réellement accessibles.

Multi-Segment Model

Ce modèle utilise toutes les possibilités de la segmentation. Chaque processus possède sa propre table de descripteurs de segments. De plus, les segments peuvent être privés à chaque processus ou être partagés. La protection de l'espace mémoire est alors maximale.

Les informations sur les segments sont contenus dans la GDT (Global Description Table) qui contient un descripteur par segment.

Chaque descripteur contient les droits d'accès au segment (mode d'utilisation et droit en écriture), son adresse de début (base address), son adresse de fin (limit address), son type ainsi que des informations sur son état.

GESTION DE LA MÉMOIRE



TECHNIQUES DE GESTION

La segmentation : le programme exécutable

- Découpé en segments
 - Adresses référencées dans le programme sous la forme
 - (**segment, offset**)
- segment : numéro du segment dans lequel se trouve l'information à accéder
 - offset : emplacement dans le segment de l'information à accéder (adresse relative au début du segment)

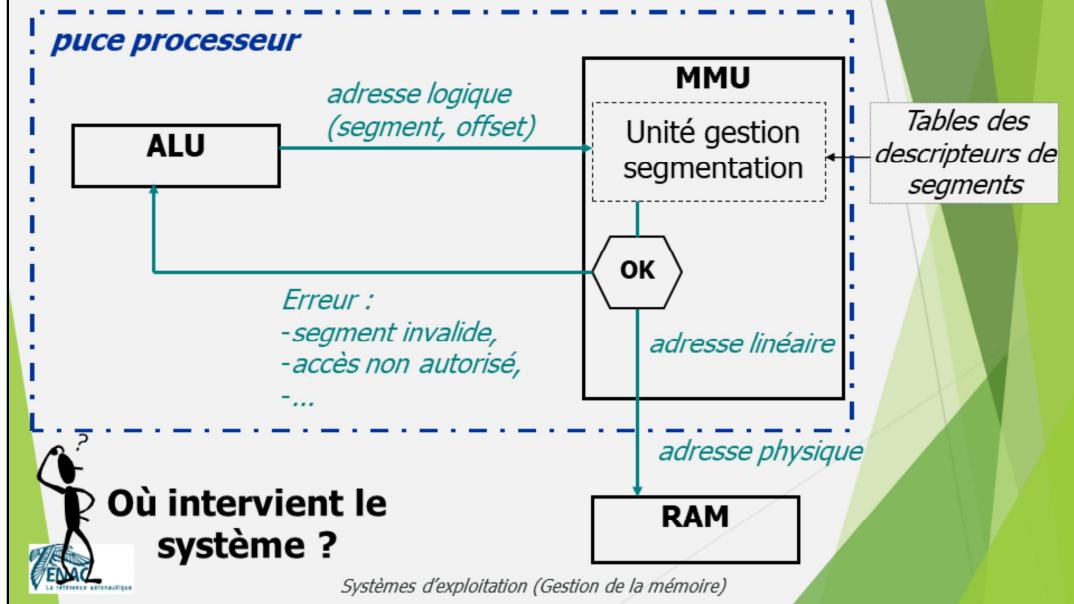


Systèmes d'exploitation (Gestion de la mémoire)

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION

La segmentation : recherche d'une adresse



L'adresse logique est celle référencée dans le code du processus. Elle est de la forme (segment , offset) avec :

- segment : numéro du segment dans lequel se trouve l'information à accéder
- offset : emplacement dans le segment de l'information à accéder (adresse relative au début du segment)

L'adresse linéaire est l'emplacement de l'information à accéder dans l'espace d'adressage linéaire du processeur. Si la gestion de la mémoire virtuelle n'est pas activée, l'adresse linéaire est considérée comme une adresse physique en RAM et est transmise par la MMU sur le bus d'adresse sans traduction préalable.

L'adresse physique est l'adresse transmise à la RAM via le bus d'adresse.

Contrôle d'accès aux segments

Chaque segment peut se voir attribuer des droits d'accès par le système (par exemple, lecture simple pour le segment de code et lecture/écriture pour les segments de données et de pile).

Pour un système mettant en œuvre les droits d'accès aux segments, un code s'exécutant en mode utilisateur n'aura accès qu'aux segments de données et de pile de son processus, un code s'exécutant en mode noyau aura accès à toute l'étendue de la mémoire.

Gestion des erreurs

Les erreurs générées par la MMU sont des exceptions (traps). En cas d'erreur, le processeur se déroute vers la fonction de traitement de l'exception.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION



La segmentation

- En résumé, la segmentation répond aux contraintes
 - de **gestion** de l'espace
 - d'**optimisation**
 - de **protection** des informations
- Mais pas à l'**extension** de la place mémoire ...



Systèmes d'exploitation (Gestion de la mémoire)

Les segments sont adressés en RAM. Le découpage en segments offre une meilleure granularité de l'occupation de la RAM, mais si celle-ci est pleine, il devient impossible de créer de nouveaux processus.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION



La mémoire virtuelle

- Mémoire réelle = RAM
 - taille limitée (chère !)
 - quelles solutions quand la RAM est pleine ?
- Mémoire virtuelle
 - **extension** de la RAM
 - mémoire virtuelle = RAM + mémoire auxiliaire
 - adresses internes au processus indépendantes du support physique



Systèmes d'exploitation (Gestion de la mémoire)

Les techniques de mémoire virtuelle permettent de stocker tout ou partie d'un processus sur une mémoire auxiliaire.

L'espace d'adressage dit virtuel comprend la RAM et la zone dédiée en mémoire auxiliaire (appelée fréquemment espace d'échange ou zone de swap).

La taille de l'espace d'échange généralement conseillée est la suivante :

Votre ordinateur dispose de 6 Go de RAM ou plus Allouez un espace d'échange égal à la taille de votre RAM ;

Votre ordinateur dispose de 1 Go de RAM à 4 Go Allouez un espace d'échange de 1× à 1,5× la taille de votre RAM ;

Votre ordinateur dispose de moins de 1 Go de RAM Allouez un espace d'échange de 1,5× à 2× la taille de votre RAM.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION



La mémoire virtuelle : pagination

- Technique de mise en œuvre de la mémoire virtuelle
- Processus et mémoire virtuelle découpés en « pages » de même taille (entre 4 Ko et 4 Mo)
- Tables des pages : assurent la correspondance entre adresses physiques et virtuelles
- Peut s'utiliser conjointement avec la segmentation



Systèmes d'exploitation (Gestion de la mémoire)

Dans le cadre de pagination associée à la segmentation, chaque segment est divisé en pages de taille prédéfinie (4Ko, 2Mo ou 4Mo), qui peuvent être indifféremment stockées en RAM ou sur une mémoire de masse.

La taille des pages est fixée par le système d'exploitation. Son choix doit offrir le meilleur compromis entre un temps de traitement rapide (grandes pages, donc moins d'échanges entre mémoire centrale et auxiliaire) et une optimisation de l'espace mémoire (petites pages, donc moins de perte de place).

Les informations concernant les pages sont stockées dans des structures de données maintenues par le système : répertoires des tables de pages (un par processus) et tables de pages. On y trouvera en particulier leur emplacement physique, leurs droits d'accès et un indicateur de modification.

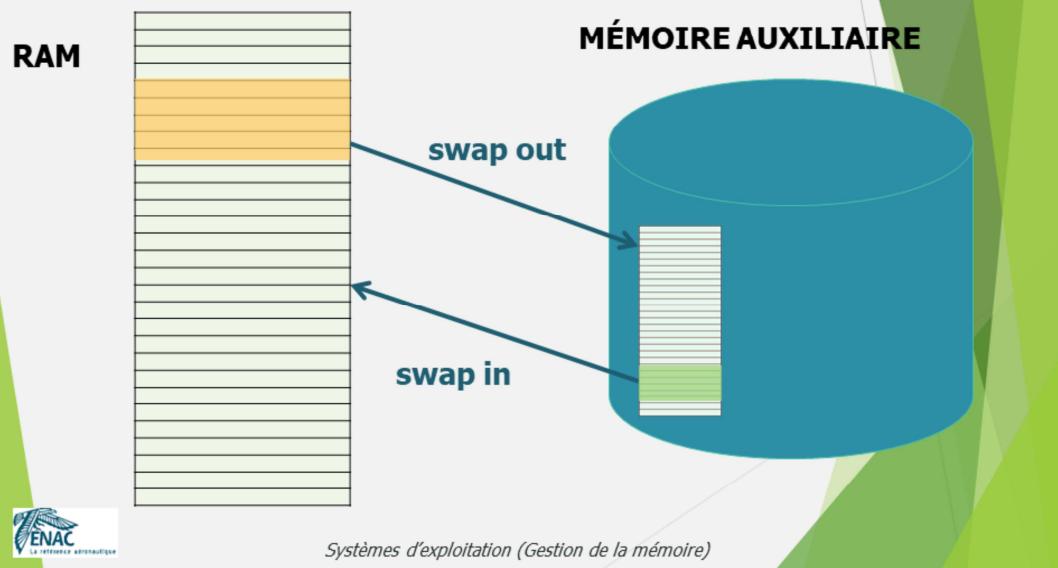
En multitâche, chaque processus possèdera son propre espace d'adressage virtuel, c'est à dire son propre répertoire des tables de pages.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION



La mémoire virtuelle : pagination



Quand l'occupation de la RAM est proche de sa capacité, le système libère de la place en déplaçant des pages en mémoire auxiliaire (opération de swap out).

Quand le contenu de pages situées en mémoire auxiliaire est nécessaire pour l'exécution du code d'un processus, celles-ci sont rechargées en RAM (opération de swap in ou trashing).

La pagination implique donc un va-et-vient d'informations entre RAM et mémoire auxiliaire.

Avantages :

- chargement de plus de processus (ou de processus plus grands) que ne permet la capacité physique de la RAM.

Inconvénients :

- baisse des performances dues au temps d'accès à la mémoire auxiliaire
- non prédictibilité du temps d'exécution dans le cas d'un système temps réel.

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION

La mémoire virtuelle sous Windows 10



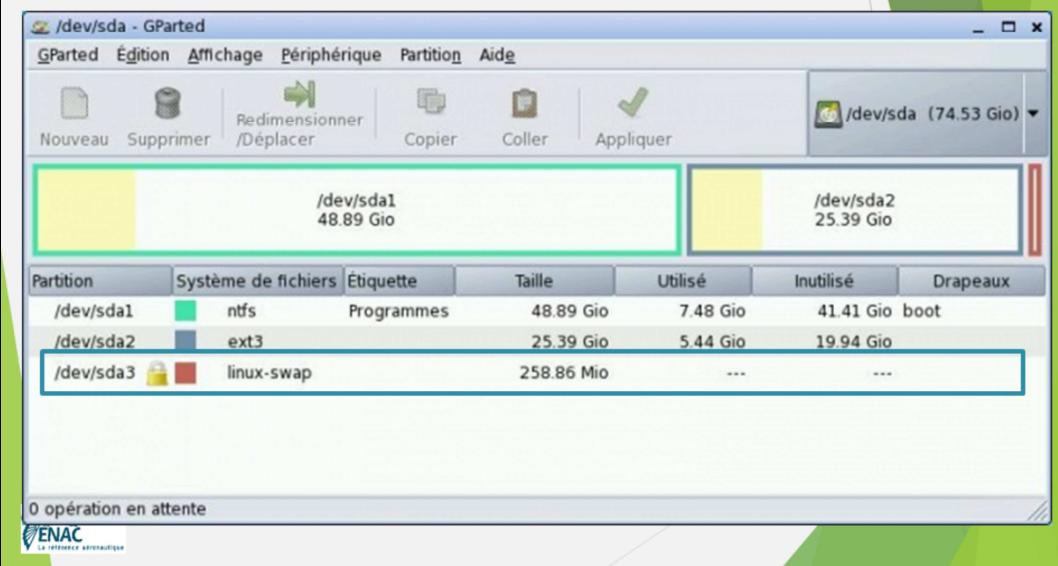
La capture d'écran montre l'interface de Windows 10 avec l'explorateur de fichiers ouvert, montrant les lecteurs C:, D:, E: et F:. Ensuite, une fenêtre des "Options de performances" est ouverte, montrant les paramètres pour les applications et la mémoire virtuelle. La section "Mémoire virtuelle" indique que le fichier d'échange est nommé pagefile.sys et indique sa taille actuelle à 1280 Mo. Un bouton "Modifier..." est visible à droite.

L'espace d'échange sous Windows est visible à la racine de la partition système (généralement disque C:), sous la forme d'un fichier nommé pagefile.sys.
Sa taille maximale est paramétrable (options de performance).

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION

La mémoire virtuelle sous Linux



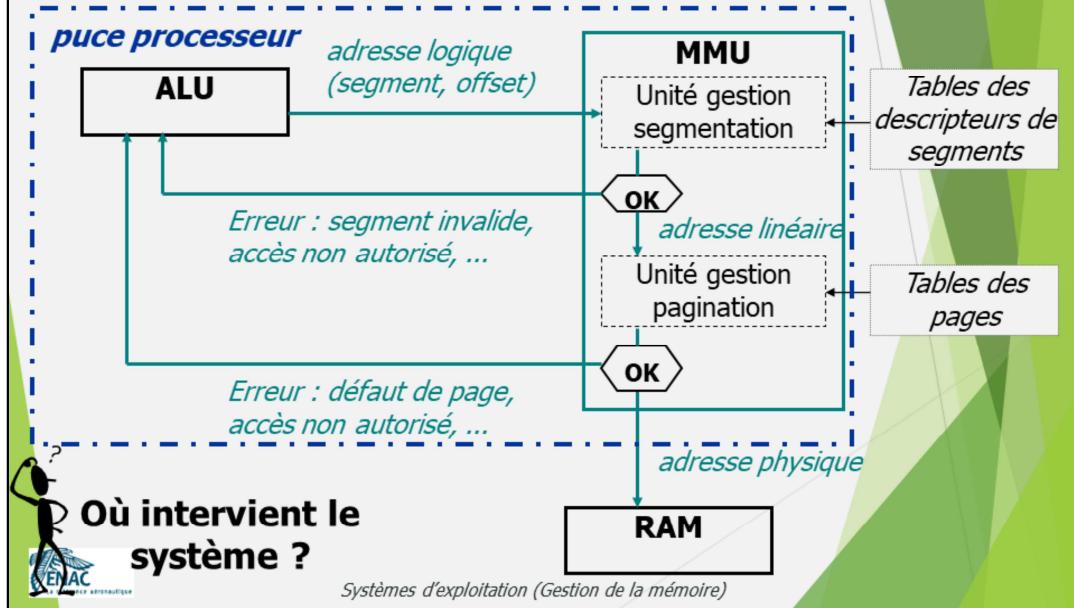
L'espace d'échange sous Linux est stocké historiquement dans une partition séparée : partition de type « swap ».

Il est néanmoins possible de gérer l'espace d'échange sur la partition système (/swapfile)

GESTION DE LA MÉMOIRE

TECHNIQUES DE GESTION

La pagination : recherche d'une adresse



Les adresses linéaires fournies à l'unité de gestion de la pagination comportent 3 informations : l'entrée dans le répertoire des pages, l'entrée dans la table des pages et l'offset dans la page.

Si la page recherchée n'est pas en RAM, la MMU lève une exception de défaut de page. Le système d'exploitation (via la routine de traitement associée à cette exception) charge la page requise en mémoire (opération de *trashing*). On parle alors de pagination à la demande.

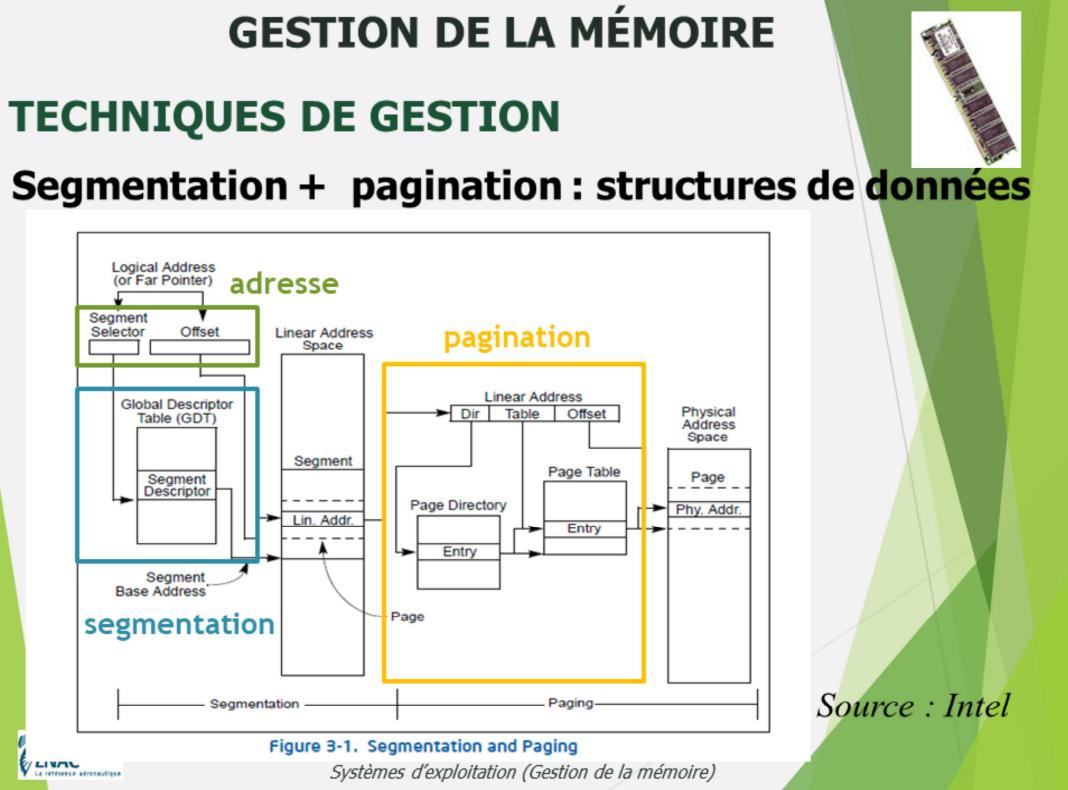
Une fois la page chargée, la demande de lecture est réitérée.

Gestion de la mémoire centrale sous Linux et Windows

Les choix de gestion de la mémoire centrale adoptés par ces deux systèmes d'exploitation sont très proches : segmentation minimale (Basic Flat Model) et gestion de la mémoire virtuelle par pagination à la demande.

Ce choix s'explique par le fait que les mécanismes de pagination et de segmentation sont redondants : découpage des processus en segments ou en pages et protection des accès au segment ou à la page.

De plus, dans le cas de Linux, il existe une contrainte de portabilité d'une architecture matérielle à une autre. Dans certaines architectures, la gestion de la segmentation est beaucoup plus limitée que dans d'autres. Limiter l'utilisation de la segmentation permet donc une meilleure portabilité du système.



Les structures de données permettant au processeur de gérer la segmentation et la pagination sont stockées en RAM.

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Hiérarchie des supports de stockage

temps
d'accès



registres
CPU

RAM

disque magnétique

bande magnétique

coût



Systèmes d'exploitation (Gestion de la mémoire)



Ordres de grandeur :

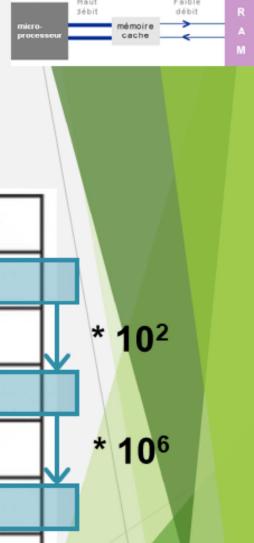
Une addition ou multiplication (travail sur les registres CPU) prend entre 1 et 5 cycles d'horloge, un accès à la RAM prend entre 400 et 1000 cycles d'horloge selon l'architecture du processeur

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Temps d'accès aux supports de stockage

Device	Typical access times
CPU registers	0.25 nsec
Cache memory (SRAM)	1-10 nsec
Conventional memory (DRAM)	10-50 nsec
Flash memory	120 µsec
Magnetic disk drive	10-50 msec
Optical disk drive	100-500 msec
Magnetic tape	0.5 and up sec



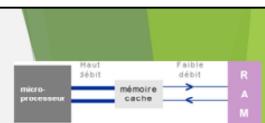
Source : <http://www-labs.iro.umontreal.ca/~monnier/1215/notes-cpumem2.pdf>

Systèmes d'exploitation (Gestion de la mémoire)



GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE



Définition

Zone de mémoire intermédiaire entre deux composants de vitesses d'accès très différentes.

Stockage temporaire d'informations pour éviter l'accès systématique au composant le plus lent.

Principe de localité

90% du temps d'exécution d'un programme passé dans 10% du code

- localité temporelle
- localité spatiale



Systèmes d'exploitation (Gestion de la mémoire)

Localité spatiale : accès à des informations proches physiquement sur le support mémoire

Localité temporelle : accès consécutifs (ou proches dans le temps) à une même information

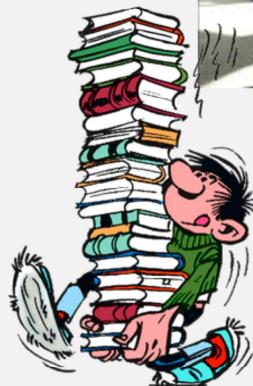
2 mécanismes de cache :

- stockage par avance lors d'un accès à un support (prefetching), issu du principe de localité spatiale,
- stockage d'informations (code ou données) déjà accédées pour usage ultérieur, issu du principe de localité temporelle.

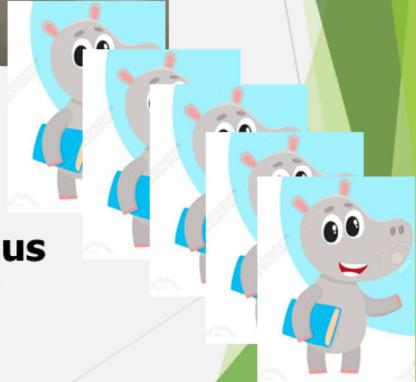
GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Principe



Qui est le plus efficace ?



Systèmes d'exploitation (Gestion de la mémoire)

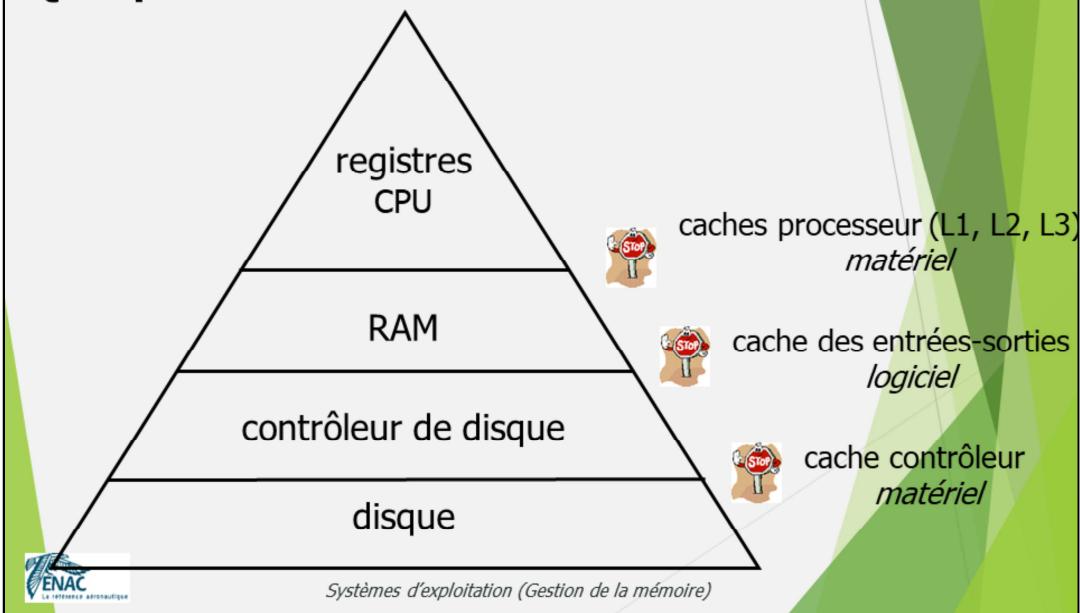
La mémoire cache s'apparente ici à l'endroit (bureau, étagère), proche de l'endroit où l'étudiant travaille, où la pile de livres récupérée en une seule visite à la bibliothèque serait stockée.

De même, un livre utilisé une fois ne serait pas rapporté avant le prochain usage.

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Quelques caches ...



Cache matériel : ajout de mémoire physique intermédiaire.

Exemple de caches processeur :

Le processeur Intel Core i5 Ivy Bridge 750 possède un cache L1 de 64 Ko / cœur, un cache L2 de 256 Ko / cœur et un cache L3 de 8 Mo partagé entre les cœurs.

Cache logiciel : il n'y a pas d'ajout de mémoire physique, mais la mise en œuvre d'un traitement permettant de stocker par avance plus d'information dans la zone mémoire la plus rapide.

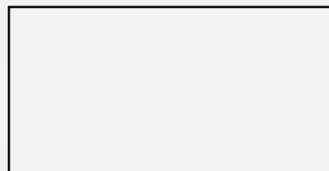
Par exemple, le système d'exploitation optimise la gestion des disques en récupérant à chaque accès plus d'informations que nécessaire (localité spatiale, stockage par avance) et en la stockant en RAM. Cela lui permet ainsi de réduire le nombre d'accès au disque.

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Cache processeur : lecture de données

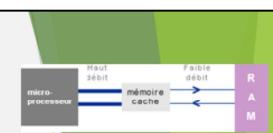
registres processeur



cache



RAM



Systèmes d'exploitation (Gestion de la mémoire)

Lecture de données

Si les données sont dans le cache

 lecture dans le cache

sinon

 lecture physique des données

 stockage dans le cache

 lecture dans le cache

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Cache processeur : écriture de données

registres processeur

donnée1 modifiée

donnée1

cache

donnée1

donnée2

RAM

Systèmes d'exploitation (Gestion de la mémoire)

Écriture de données (cache write-back)

Si les données sont dans le cache

modification dans le cache

marquage 'à écrire'

sinon

lecture physique des données

stockage dans le cache

lecture des données dans le cache

modification dans le cache

marquage 'à écrire'

Que se passe-t-il si plusieurs éléments accèdent à la mémoire en écriture ?

C'est en particulier le cas des périphériques en mode DMA (**A**ccès **D**irect à la **M**émoire sans passer par le processeur) et des systèmes multiprocesseurs avec mémoire commune

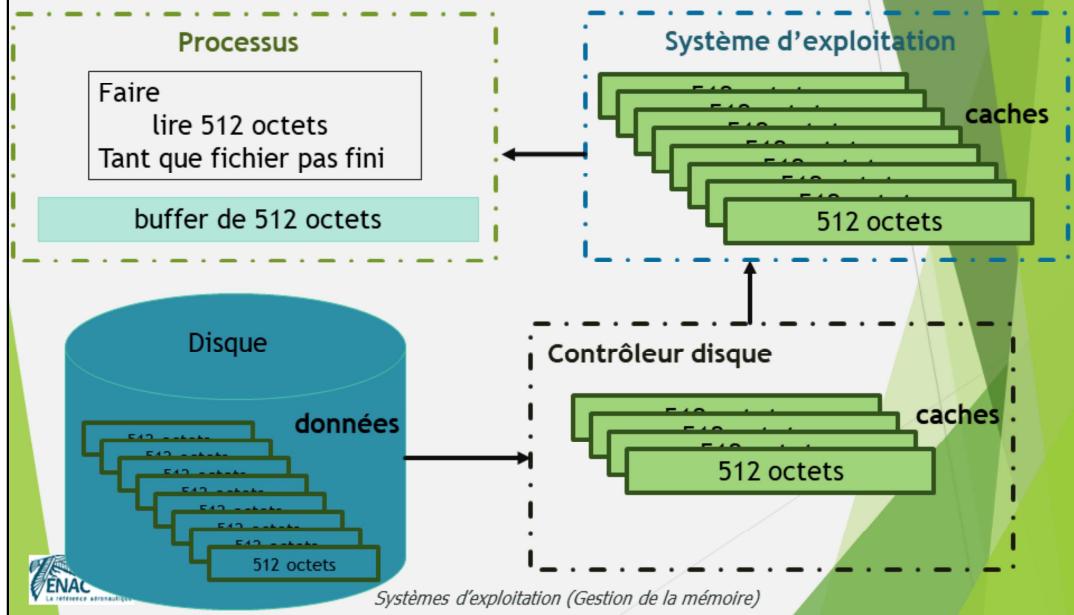
-> Nécessite des algorithmes supplémentaires pour gérer la cohérence de la mémoire.

Il existe aussi un autre type de cache en écriture (cache write-through) où la donnée écrite dans le cache est simultanément écrite en mémoire centrale.

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE

Cache des entrées-sorties



Le cache des entrées-sorties permet d'optimiser les accès aux fichiers.

En lecture :

Le système d'exploitation demande au contrôleur plus de données que le programme (par exemple 2048 octets alors que le programme en a demandé 512). Il les stocke ensuite dans ses caches en RAM. Dans l'exemple donné, cela évitera 3 accès au contrôleur de disque.

En écriture

Le système d'exploitation ne stocke pas d'information sur disque à chaque demande. Il stocke les données dans ses caches.

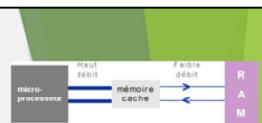
L'écriture effective sur disque (synchronisation) se fera :

- à la fermeture du fichier par le programme
- quand le système aura besoin de place dans ses caches
- à intervalle régulier par une tâche système.

La plupart des contrôleurs de disques gèrent aussi des cachets qui leur permettent de récupérer des données par avance sur le disque.

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE



Remplacement des données dans les caches

Quelques algorithmes de remplacement :

Not Most Recently Used

Least Recently Used, Least Frequently Used

First In First Out

Random

Avantages et inconvénients des caches



Réduction des accès au support le plus lent



Transferts différés, déterminisme ?



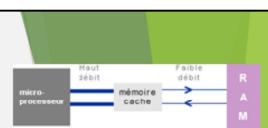
Systèmes d'exploitation (Gestion de la mémoire)

Quelques algorithmes de remplacement des données dans les caches

- Random : simple, peu efficace,
- FIFO : + efficace que le Random. Limite : une donnée utilisée fréquemment peut être quand même supprimée du cache si elle est la plus ancienne.
- LRU : la plus efficace, mais complexe à mettre en œuvre (gestion de la date)
- LFU : variante de LRU, plus simple à mettre en œuvre (gestion d'un compteur d'utilisation au lieu de la date)
- NMRU : élimination aléatoire d'une ligne autre que la plus récemment utilisée

GESTION DE LA MÉMOIRE

LA MÉMOIRE CACHE



Caches et programmation

L'efficacité des caches dépend aussi de la conception des programmes.

- Algorithme
- Choix des structures de données
- Code
- Instructions assembleur



Systèmes d'exploitation (Gestion de la mémoire)

L'objectif est de limiter le nombre de « cache miss » (situations où la donnée recherchée n'est pas dans le cache).

- **Algorithme** : préférer les algorithmes « cache oblivious ». Basés sur le principe « diviser pour mieux régner », l'objectif est de découper les traitements de manière à ne travailler que sur des données stockées en cache.

- **Structures de données** : on se rappellera par exemple qu'un tableau est une structure qui respecte la localité spatiale alors que les données d'une liste chaînée ou d'un arbre ne sont pas stockées de manière contigüe.

- **Code** : le code doit limiter l'empreinte mémoire, c'est-à-dire d'une part diminuer la quantité de mémoire utilisée afin qu'un maximum d'informations puisse rester dans le cache et d'autre part assurer des accès les plus consécutifs possibles à la mémoire.

Exemple :

En C, les lignes d'une matrice sont stockées les unes à la suite des autres en mémoire. En Fortran, ce sont les colonnes qui sont consécutives. On choisira donc un parcours ligne/ligne en C et colonne/colonne en Fortran pour améliorer les performances.

- **Instructions assembleur** : instruction de prefetching

GESTION DE LA MÉMOIRE

CHARGEMENT DU SYSTÈME EN MÉMOIRE



- Au démarrage : la RAM est "vide"



Comment charger le système? Qui le charge ?

- Chargeur primaire (*en ROM, dépendant du matériel*)
 - détecte/détermine le périphérique de démarrage
 - lance à partir de celui-ci le chargeur du système (chargeur secondaire, bootloader)
- Chargeur secondaire (*sur périphérique de démarrage*)
 - lance directement le système ou multi-boot
 - grub (Linux), winload (Windows 7, 8), U-boot, ...

Systèmes d'exploitation (Gestion de la mémoire)



Exemple : séquence de démarrage des processeurs Intel

- Exécution de l'instruction située à l'adresse FFFFFFF0h, généralement un « jump » qui pointe sur la première instruction du programme situé sur la ROM de la carte-mère (fourni avec elle par le fabricant).
- Power Good : test du courant fourni par l'alimentation.
- POST (Power-On Self-Test) : vérifications de configuration matérielle (code du BIOS, mémoire, intégrité de la carte-mère, affichage de « press F2 to enter setup », initialisation de la RAM et des périphériques).
- Recherche de l'OS (chargeur primaire) :
 - lit les périphériques (disque, CD/CVD, réseau, ...) dans l'ordre configuré dans le setup,
 - pour chacun, regarde s'il est amorçable (bootable), c'est-à-dire s'il contient un OS (indiqué sur un disque par la présence du Magic Number 0xAA55 dans le premier secteur)
 - dès qu'il trouve un périphérique amorçable, il lance le chargeur secondaire (bootloader) présent.
- Démarrage de l'OS (chargeur secondaire) :
 - Si n'y a qu'un seul OS, démarrage du système
 - Sinon, le chargeur d'amorçage (GRUB, LILO, rEFIt, Boot Camp, etc ...) propose à l'utilisateur de choisir l'OS qu'il souhaite lancer parmi ceux configurés.

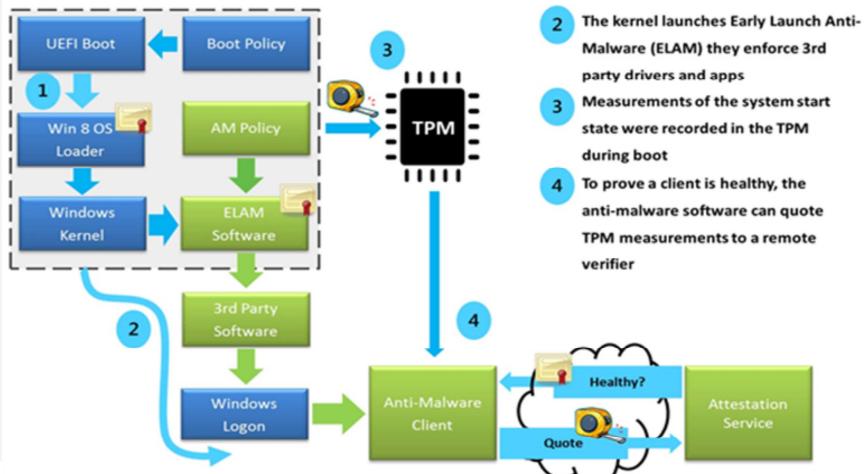
GESTION DE LA MÉMOIRE

CHARGEMENT DU SYSTÈME EN MÉMOIRE

Séquence de démarrage Windows



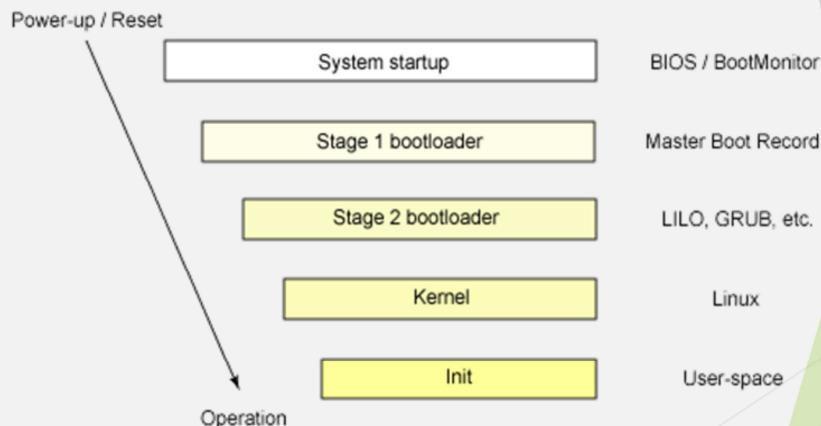
Windows 8 Platform Integrity Architecture



GESTION DE LA MÉMOIRE

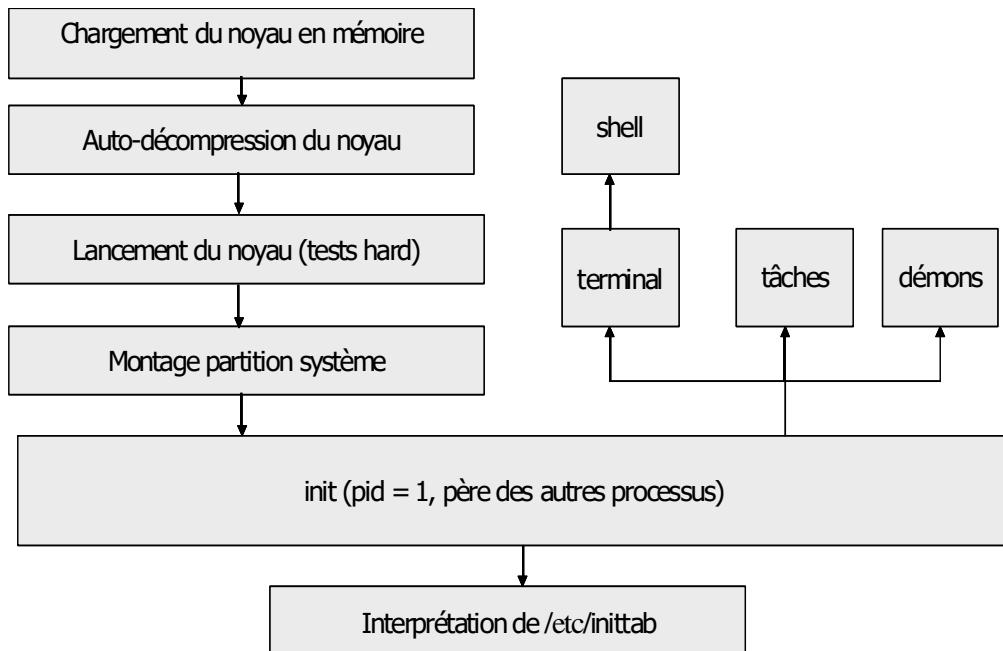
CHARGEMENT DU SYSTÈME EN MÉMOIRE

Séquence de démarrage Linux



Systèmes d'exploitation (Gestion de la mémoire)

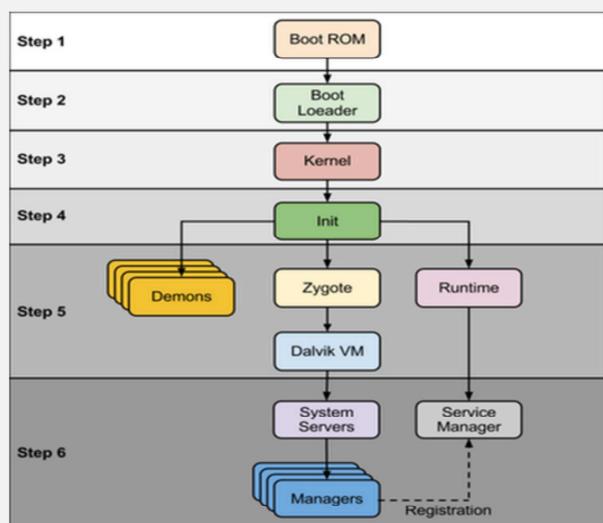
Le schéma ci-dessous détaille la phase de démarrage à partir du chargement du noyau en mémoire.



GESTION DE LA MÉMOIRE

CHARGEMENT DU SYSTÈME EN MÉMOIRE

Séquence de démarrage Android



Systèmes d'exploitation (Gestion de la mémoire)

Systèmes d'exploitation

Le système d'exploitation

Gestion du processeur



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Gestion du processeur)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur**
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

Architecture des systèmes

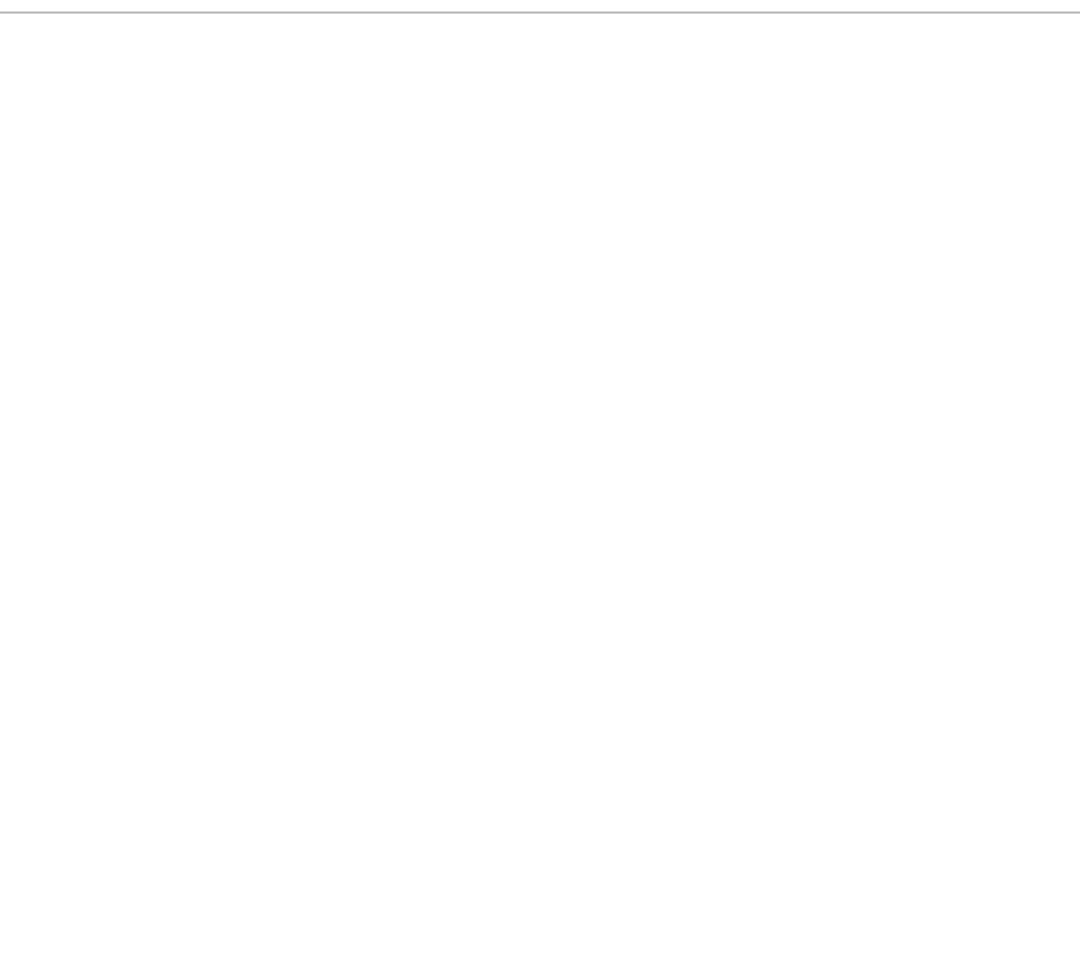
LE SYSTÈME D'EXPLOITATION

GESTION DU PROCESSEUR



Systèmes d'exploitation (Gestion du processeur)

3



GESTION DU PROCESSEUR

CAHIER DES CHARGES



Le processeur exécute les instructions des programmes

- Indiquer au processeur quel code exécuter

Système multitâches : le temps de travail du processeur est partagé entre les tâches

- Gérer et optimiser le partage du temps processeur
- Gérer le basculement d'une tâche à l'autre

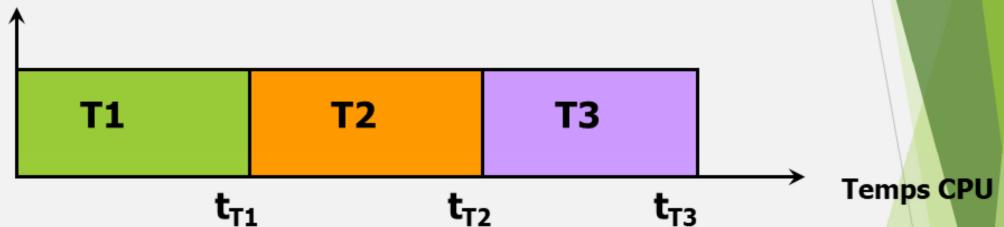


GESTION DU PROCESSEUR

PARTAGE DU TEMPS PROCESSEUR



Système monotâche



$$\text{Temps total d'exécution} = t_{T1} + t_{T2} + t_{T3}$$



Systèmes d'exploitation (Gestion du processeur)

5

Avec un système monotâche, un seul programme utilisateur constitué d'un seul thread peut être chargé en mémoire et exécuté à la fois. Il faudra attendre la fin de son exécution pour lancer un autre.

Le système d'exploitation n'effectue aucun traitement concernant la gestion du temps de travail du processeur.

Son seul rôle en monotâche est de fournir au processeur l'adresse de début du programme.

Exemple de système monotâche : DOS (première version : 1981, dernière version : 2000)

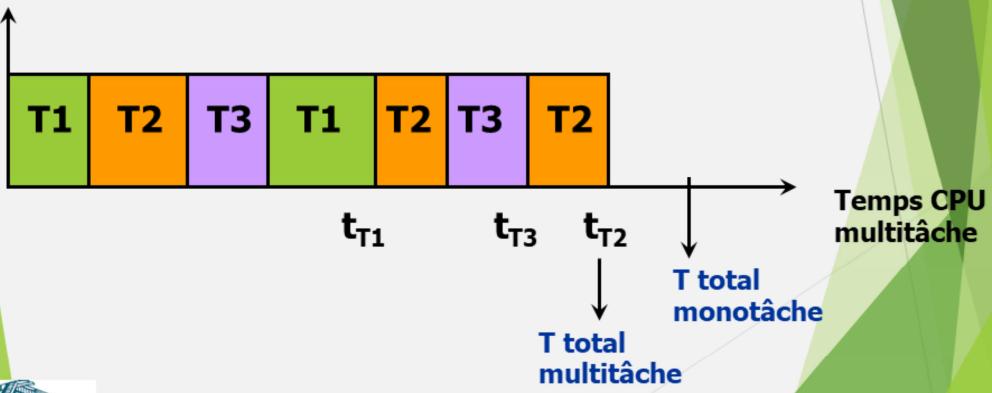
GESTION DU PROCESSEUR



PARTAGE DU TEMPS PROCESSEUR

Système multitâche

Objectif : **Temps total d'exécution < $t_{T1} + t_{T2} + t_{T3}$**



On considère ici le temps de travail d'un processeur. En effet, même sur une machine multiprocesseurs (ou multi-cœurs), le temps de travail d'un processeur (ou cœur) sera partagé entre plusieurs tâches si le nombre de tâches est supérieur au nombre de processeurs.

Avec un système multitâches, plusieurs programmes peuvent être simultanément en cours d'exécution.

Pour la gestion du partage du temps de travail du processeur, on distingue le multitâches coopératif du multitâche préemptif.

En **multitâche coopératif**, les tâches décident elles-mêmes de leur occupation du processeur. Un programme qui ne libère pas explicitement le processeur le gardera jusqu'à la fin de son exécution. Ce mécanisme peut amener à un blocage complet du système si un processus ne rend jamais la main suite à un bug.

Les systèmes Microsoft avant Windows 98 et Apple avant Mac OS X fonctionnaient sous le mode coopératif.

En **multitâche préemptif**, le système est maître du temps de travail du processeur. Il peut reprendre la main à tout moment. Il accorde des « tranches » de temps processeur (time slices) aux tâches selon différents critères. Une tâche ne peut plus bloquer le système et la gestion des ressources devient beaucoup plus performante.

Les systèmes de type Unix fonctionnent sur le mode préemptif depuis leur création (1969).

GESTION DU PROCESSEUR



PARTAGE DU TEMPS PROCESSEUR

Système multitâche

Objectif : **Temps total d'exécution < $t_{T1} + t_{T2} + t_{T3}$**



Comment?

En utilisant les temps d'inactivité du processeur :

- entrées/sorties
- attentes

Mais ...

Attention au temps de commutation de contexte



L'exécution d'une tâche est donc une suite de burst CPU et d'attentes.

On appelle commutation de contexte le changement de tâche maître du processeur.

En multitâche préemptif, une tâche peut « perdre » le processeur :

- Quand elle demande l'accès à une ressource non disponible immédiatement (entrée-sortie),
- Quand elle se met explicitement en attente (synchronisation entre tâches, timer),
- Au bout d'un temps prédéfini d'utilisation continue du processeur (« quantum de temps », de l'ordre de 10 à 50 ms) dans certains modes de gestion du temps processeur.
- A l'arrivée d'une tâche de priorité supérieure dans certains modes de gestion du temps processeur.

GESTION DU PROCESSEUR



CONTEXTE D'UNE TÂCHE

= environnement d'exécution + mot d'état du processeur

Environnement d'exécution

- adresse de la prochaine instruction à exécuter
- valeurs des variables globales
- piles utilisateur et système
- informations particulières (zone U sous UNIX), ...



Lors de la commutation de contexte, les informations concernant l'état d'exécution de la tâche en cours sont sauvegardées et celles de la nouvelle tâche sont chargées dans les registres du processeur et dans les tables gérées par le système d'exploitation.

GESTION DU PROCESSEUR



CONTEXTE D'UNE TÂCHE

= environnement d'exécution + mot d'état du processeur

Mot d'état du processeur

- état d'exécution (actif ou en attente)
- mode de fonctionnement
- masque d'interruptions
- contexte accessible en mémoire : adresse des tables de segments
- compteur ordinal, ...



Au cours de la commutation de contexte, le contexte d'exécution est sauvegardé et celui de la nouvelle tâche est chargé à la place.

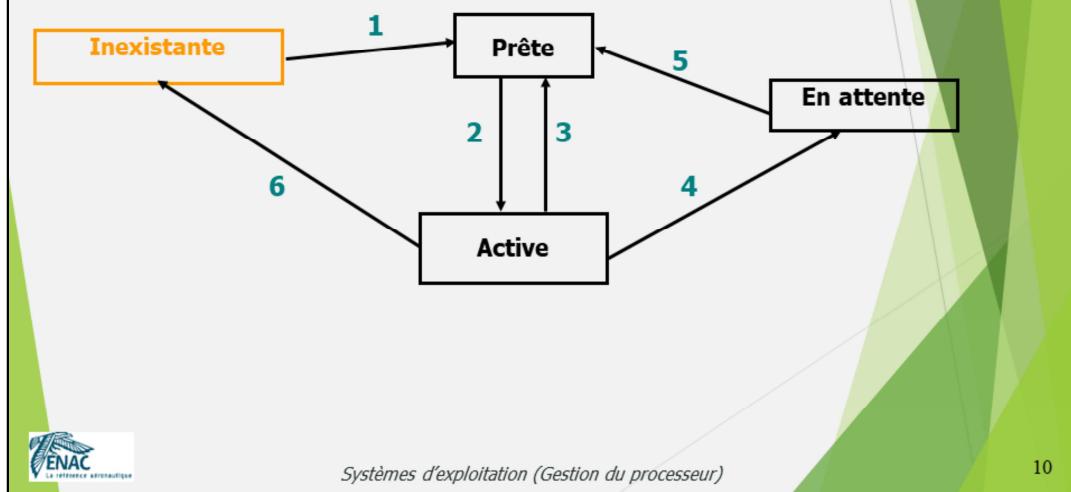
Cette opération possède un coût en temps, dépendant de la quantité d'information manipulée (90% environ du temps de la commutation de contexte est passé dans la gestion de la mémoire). Si la commutation se fait entre threads, cette quantité d'information est moindre (le code et les données communs aux threads d'un même processus ne sont pas commutés).

Ce temps peut devenir critique pour les performances si les commutations de contexte sont trop fréquentes ou dans le cas des systèmes temps réel. Dans un système temps réel, le délai de latence de l'allocateur (temps de réaction et de commutation de contexte) doit aussi être garanti.

Les systèmes offrent généralement une « garantie au pire » qui assure un délai maximal d'exécution de la commutation (par exemple, de l'ordre de 20 µs sous Linux RT).

GESTION DU PROCESSEUR

CYCLE DE VIE D'UNE TÂCHE



États de la tâche

Prête (ou éligible) La tâche attend que le processeur lui soit affecté

En attente La tâche est en attente d'un événement (fin d'une entrée-sortie, signal de réveil, échéance d'un timer, ...)

Active (ou élue) Le processeur exécute les instructions de la tâche, en mode utilisateur ou en mode noyau (ou mode superviseur)

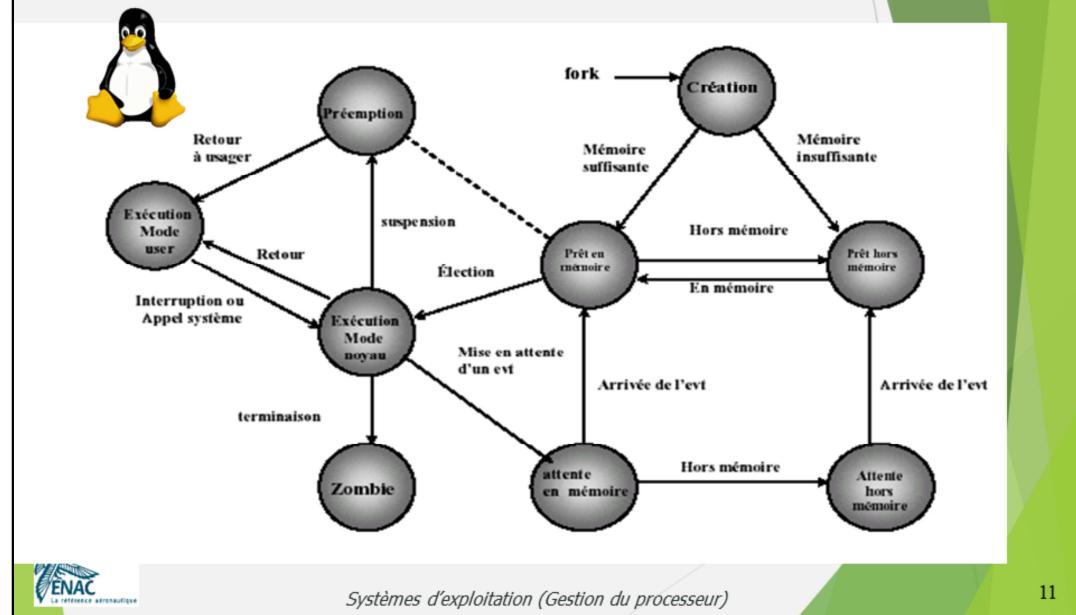
Inexistante La tâche n'est pas encore créée ou son exécution est terminée. Elle n'est pas connue du système.

Transitions

- 1 - Chargement en mémoire d'une tâche par appel système d'un processus existant.
- 2 – L'ordonnanceur choisit la tâche.
- 3 - L'ordonnanceur choisit une autre tâche.
- 4 - La tâche se met en attente d'un événement.
- 5 - L'événement attendu se produit.
- 6 - Terminaison de l'exécution. Sortie des tables du système.

GESTION DU PROCESSEUR

CYCLE DE VIE D'UNE TÂCHE SOUS LINUX



Sur ce schéma du cycle de vie d'une tâche sous Linux, on voit apparaître la notion de mémoire virtuelle (pour les états prêt et attente hors mémoire), ainsi que les modes d'exécution de la tâche (noyau ou utilisateur).

Les états « hors mémoire » correspondent à des états où la tâche est en attente d'une ressource (processeur ou autre) et que tout ou partie du processus n'est pas en RAM mais dans l'espace d'échange (swap).

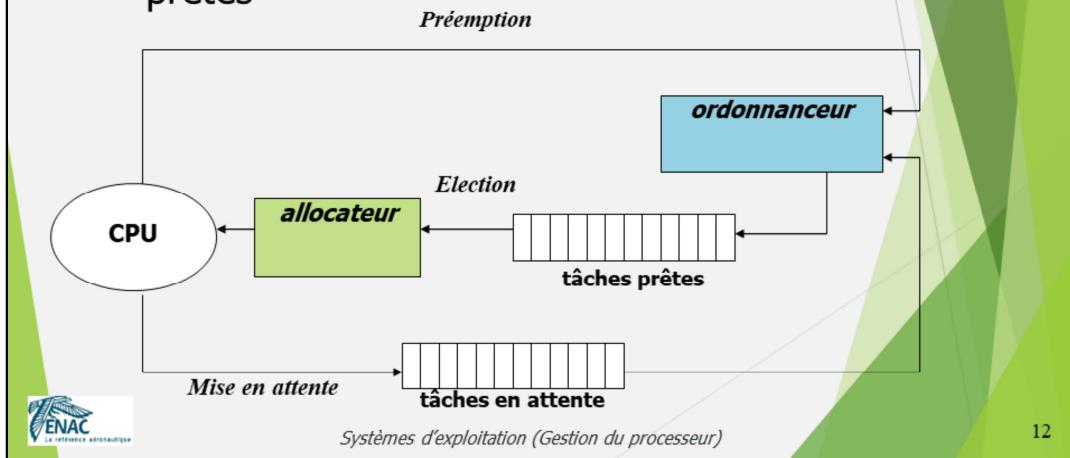
L'état « zombi » correspond à l'intervalle de temps pendant lequel une tâche ayant terminé son exécution existe encore dans les tables du système.

GESTION DU PROCESSEUR



GESTION DU CYCLE DE VIE

- Allocateur : gère la commutation de contexte
- Ordonnanceur : gère la file d'attente des tâches prêtes



L'**allocateur** (ou **dispatcher**) alloue le processeur dès qu'il est disponible à la tâche située en tête de la file d'attente des tâches prêtes. Il assure donc la commutation de contexte.

La planification du travail (ou **ordonnancement**) est assurée par l'**ordonnanceur** (ou **scheduler**). Son rôle est de créer les nouvelles tâches, gérer l'attribution du processeur aux tâches en cours (partage du temps de travail du processeur), exclure mutuellement les tâches demandant simultanément l'accès à une même ressource non partageable (mise en attente d'une tâche tant que la ressource n'est pas disponible).

Il détermine donc l'ordre de passage des tâches et leur durée d'occupation du processeur.

Sous Linux, l'ordonnanceur est réveillé à chaque interruption d'horloge (recalcul des priorités, gestion du quantum de temps, ...).

GESTION DU PROCESSEUR



GESTION DU CYCLE DE VIE

L'ordonnancement : quelle politique ?

Choix dépendant des contraintes de fonctionnement des applications :

- optimiser le temps de réponse,
- optimiser le temps de restitution,
- respecter les échéances,
- ...



Les mécanismes d'ordonnancement sont conçus pour optimiser l'exécution des tâches selon certains critères. Parmi ceux-ci, on peut citer :

- l'utilisation maximale de la CPU,
- la capacité de traitement maximale,
- l'utilisation maximale des périphériques,
- la diminution du temps moyen de restitution (temps total écoulé entre le lancement d'une tâche et sa terminaison),
- la diminution du temps moyen d'attente de la ressource processeur,
- la diminution du temps moyen de réponse des tâches interactives,
- l'équité entre les tâches,
- le débit des tâches à traiter,
- ...

En fonction du type de système, les critères, donc les algorithmes implémentant ceux-ci, vont différer.

Un mécanisme d'ordonnancement peut être :

- **non-préemptif** (sans réquisition) : lorsqu'une tâche occupe la ressource processeur, elle la garde jusqu'à ce qu'elle la libère volontairement, qu'elle passe dans l'état final ou qu'elle demande une autre ressource.
- **préemptif** (avec réquisition) : l'ordonnanceur est maître de la gestion de la ressource processeur. Il peut reprendre la CPU à une tâche sans que celle-ci l'ait volontairement libérée.

GESTION DU PROCESSEUR



GESTION DU CYCLE DE VIE

Quelques politiques d'ordonnancement

- FIFO
- Tourniquet (Round-Robin) = FIFO préemptif + quantum de temps
- Tourniquet avec priorités statiques
- Tourniquet avec priorités dynamiques
- Plus court temps d'exécution, plus court temps d'exécution restant (SGBD)



FIFO : l'ordonnanceur gère une file d'attente circulaire de tâches prêtes où la première arrivée est la première servie. Cet algorithme peut être préemptif ou non, selon qu'une tâche qui attend un événement libère le processeur pendant cette attente ou qu'elle le garde.

Avantages : L'utilisation de la CPU est maximale. Algorithme équitable.

Inconvénients : L'attente peut être longue pour des tâches de courte durée.

Round-Robin : c'est un ordonnancement FIFO préemptif, limité à une durée d'exécution prédéfinie. Au fur et à mesure de leur arrivée dans la file, les tâches se voient attribuer un quantum de temps. Quand une tâche arrive au bout de son quantum, elle est interrompue et reprend place à la fin de la file d'attente. L'ordonnanceur prend la main à intervalles réguliers. Il est réveillé par l'interruption d'horloge du processeur.

Cet algorithme est équitable : en présence de n tâches, le délai d'attente maximum dans la file est de $(n-1) * \text{quantum}$. L'utilisation de la CPU et le temps moyen de réponse dépendent du choix du quantum : selon la valeur choisie, l'ordonnanceur favorise soit les calculs, soit les entrées/sorties.

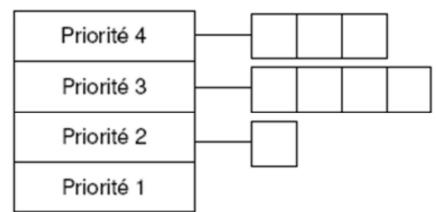
Dans l'algorithme Round Robin, toutes les tâches ont une importance égale. Si cette configuration n'est pas représentative du système d'information, il faut introduire la notion de priorité entre tâches -> **Tourniquet à priorités statiques, Tourniquet à priorités dynamiques**.

GESTION DU PROCESSEUR



GESTION DU CYCLE DE VIE

L'ordonnancement tourniquet à priorités dynamiques



- Préemptif
- Files d'attentes circulaires (tourniquet, Round-Robin)
- Quantum de temps (fixe ou variable)
- Priorité recalculée en fonction de l'activité de la tâche
- « Aging »



Systèmes d'exploitation (Gestion du processeur)

15

Dans un ordonnancement à priorités statiques, si le système est très chargé, les tâches de faible priorité peuvent ne jamais obtenir la CPU (situation de **famine**).

Pour pallier à ce défaut, la priorité peut être un paramètre dynamique qui permettra de changer la tâche de file d'attente. On introduit ici la notion de **files réactives**.

L'ajustement dynamique de la priorité d'une tâche est calculé par l'ordonnanceur.

Lorsqu'une tâche est créée, on lui attribue un niveau de priorité de base.

Elle entre dans la file des tâches prêtes à son niveau de priorité et attend son tour (Round Robin).

Selon l'usage qu'elle a fait de la CPU, sa priorité peut être modifiée (à la baisse ou à la hausse).

Par exemple, si une tâche est rapidement suspendue suite à une entrée/sortie (**burst I/O**) et qu'elle n'a utilisé qu'une petite fraction f de son quantum de temps, sa priorité peut être augmentée (proportionnellement à $1/f$).

Elle réobtiendra rapidement l'accès au processeur car elle sera placée dans une file de priorité supérieure.

Un algorithme de vieillissement (**aging**) pourra être appliqué aux tâches qui attendent dans les files à basse priorité afin de les inclure dans des files de niveaux plus favorables.

Cet algorithme est équitable. L'utilisation de la CPU et le temps moyen de réponse sont bons. La gestion de la priorité dynamique peut prendre en compte plusieurs critères (occupation CPU, durée des entrées/sorties, type d'entrées/sorties effectuées, ...).

GESTION DU PROCESSEUR



GESTION DU CYCLE DE VIE

Quelques politiques d'ordonnancement temps réel

- Rate Monotonic,
- Deadline Monotonic,
- Earliest Deadline First,
- Least Laxity First,
- ...



Systèmes d'exploitation (Gestion du processeur)

16

Pour répondre aux contraintes d'ordonnancement temps réel, on distingue deux types de tâches :

Les tâches périodiques : elles correspondent aux mesures effectuées sur le procédé (par exemple, lecture de mesures effectuées par des capteurs). Elles doivent se réveiller régulièrement (toutes les P unités de temps).

Les tâches apériodiques : elles correspondent aux événements (par exemple, arrivée d'une alarme). Elles sont donc réveillées de manière aléatoire.

Quelques algorithmes d'ordonnancement temps réel :

Rate-Monotonic scheduling (RM)

Algorithme dynamique et préemptif basé sur des priorités fixes. Avec cet algorithme, la priorité d'une tâche est fonction de sa période, de telle sorte que la tâche de plus petite période est la tâche la plus prioritaire.

Deadline Monotonic scheduling (DM ou Inverse Deadline)

Cet algorithme constitue une généralisation de l'algorithme Rate Monotonic à des tâches quelconques. Il est basé sur le même principe que RM mais la priorité d'une tâche est fonction inverse de son délai critique. La tâche la plus prioritaire est donc la tâche de plus petit délai critique.

Earliest Deadline First scheduling (EDF)

Algorithme dynamique et préemptif basé sur des priorités dynamiques. A l'instant t , la tâche la plus proche de son échéance est la plus prioritaire.

Least Laxity First (LLF)

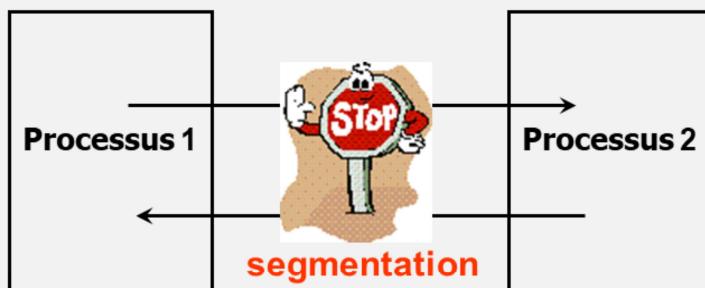
Algorithme dynamique et préemptif basé sur des priorités dynamiques.

GESTION DU PROCESSEUR

PROBLÉMATIQUE DES APPLICATIONS MULTITÂCHES



Communication entre tâches



Systèmes d'exploitation (Gestion du processeur)

17

Les processus ne possèdent a priori aucune zone de mémoire commune et leur code ne peut pas accéder à la mémoire d'un autre.

Les systèmes d'exploitation multitâche offrent des mécanismes permettant la communication interprocessus. Parmi ceux-ci, on trouve :

- zone de mémoire partagée,
- boîte à lettres (file de messages asynchrones),
- pipeline,
- mécanisme d'échange de messages synchrones,
- etc ...

Ces mécanismes sont accessibles via des API spécifiques aux systèmes.

GESTION DU PROCESSEUR

PROBLÉMATIQUE DES APPLICATIONS MULTITÂCHES



Synchronisation des tâches



Race condition : conflit survenant quand 2 tâches accèdent en concurrence à une même ressource



Il n'est pas a priori possible de savoir dans quel ordre le système d'exploitation va ordonner les différentes tâches d'une application. Quand celles-ci accèdent à une même zone mémoire, cela peut conduire à des conflits et induire un comportement non déterministe.

Dans ce cas, l'application devra contraindre l'ordonnancement des tâches grâce à des moyens de synchronisation offerts par le système. On citera en particulier :

- Les mutex
- Les sémaphores
- Les mécanismes d'attente/réveil

Ces mécanismes sont accessibles via des API spécifiques aux systèmes.

Systèmes d'exploitation

Le système d'exploitation

Gestion des entrées-sorties



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Gestion des entrées-sorties)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties**
- Gestion des fichiers

La virtualisation

Architecture des systèmes

LE SYSTÈME D'EXPLOITATION

GESTION DES ENTRÉES-SORTIES

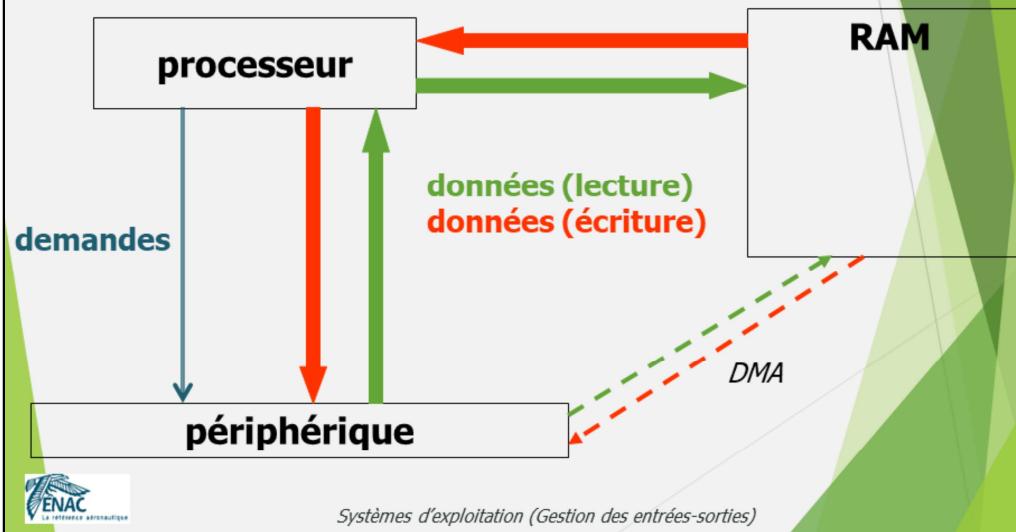


GESTION DES ENTRÉES-SORTIES

DEFINITION



= transfert de données entre la mémoire centrale et un périphérique



Le transfert des données s'effectue par lecture/écriture dans les registres du contrôleur de périphérique. Il existe deux techniques permettant au processeur d'accéder à ces registres :

- Certains processeurs utilisent les mêmes instructions pour accéder à la mémoire et aux périphériques. Dans ce cas, un espace mémoire particulier est réservé pour les registres d'entrée/sortie. Selon la valeur de l'adresse, l'écriture se fait en RAM ou sur un périphérique. On parle d'*entrée/sortie mappée en mémoire*.
- Certains processeurs possèdent un jeu d'instruction spécial pour la lecture et l'écriture sur les périphériques. Cette solution est celle adoptée dans l'architecture Intel.

Accès direct à la mémoire (DMA) : transfert d'information d'un périphérique vers la RAM sans passer par les registres du processeur.

En cas de transfert d'une grande quantité de données entre deux périphériques, il peut être plus efficace de les laisser communiquer directement entre eux plutôt que de solliciter le processeur. Le processeur indique au contrôleur de DMA quels sont les périphériques qui doivent communiquer, le nombre et éventuellement l'adresse des données à transférer, puis il initie le transfert. Le processeur n'est pas sollicité durant le transfert, et le contrôleur DMA lui signale sa terminaison par une interruption.

Le contrôleur DMA est relié au processeur (2 lignes de contrôle) et se rend maître du bus pendant le transfert.

Pour aller plus loin : processeurs d'entrée/sortie

GESTION DES ENTRÉES-SORTIES

STRUCTURE INTERNE D'UN CIRCUIT PÉRIPHÉRIQUE



Registre de données : reçoit les données à transférer vers ou en provenance du périphérique

Registre d'état : contient le mot d'état, dont chaque bit est représentatif de l'état du contrôleur et du périphérique associé

Registre de commande : contient le mot de commande

Systèmes d'exploitation (Gestion des entrées-sorties)

5

Un circuit périphérique (ou contrôleur, interface) communique avec l'extérieur via ses registres :

- les commandes et données venant du processeur sont écrites dans les registres du circuit périphérique,
- le circuit périphérique fournit des informations (état, données) en les stockant dans ses registres où le processeur pourra venir les lire.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Comment s'établit le dialogue ?

Le processeur émet une demande

- lecture
- écriture
- opération de contrôle

en écrivant dans les registres du contrôleur

- la commande
- les données éventuelles



Systèmes d'exploitation (Gestion des entrées-sorties)

6

La demande d'entrée-sortie est synchrone avec l'exécution du programme.

Elle est toujours à l'initiative du processeur.

Par exemple, si le programme demande une écriture sur disque, le processeur écrit la demande dans le registre de commande du contrôleur de disque et les informations à écrire dans ses registres de données.

Dans le cas d'une entrée-sortie DMA, la demande est transmise au contrôleur DMA et non au périphérique. Le contrôleur DMA se chargera de la suite de l'entrée-sortie.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Comment le périphérique répond-il au processeur ?

2 techniques possibles :

- test de mot d'état (*scrutation, polling*)
- interruption



Systèmes d'exploitation (Gestion des entrées-sorties)

7

Le contrôleur de périphérique ne sait pas accéder à la mémoire centrale ni envoyer des données au processeur.

Le processeur devra être informé de la terminaison de l'entrée-sortie (par test de mot d'état ou par interruption) puis aller ensuite récupérer les éventuelles données dans les registres du circuit périphérique.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Attente sur test de mot d'état

- A l'initiative du processeur
- Scrutation (polling) : le programme boucle tant que le périphérique n'est pas prêt
- Attente active : monopolise le processeur pendant l'entrée-sortie



Systèmes d'exploitation (Gestion des entrées-sorties)

8

Mode programmé (test de mot d'état, scrutation, polling)

Le processeur scrute le registre d'état du contrôleur pour savoir s'il peut écrire ou s'il y a une donnée à lire, puis il accède au registre de données.

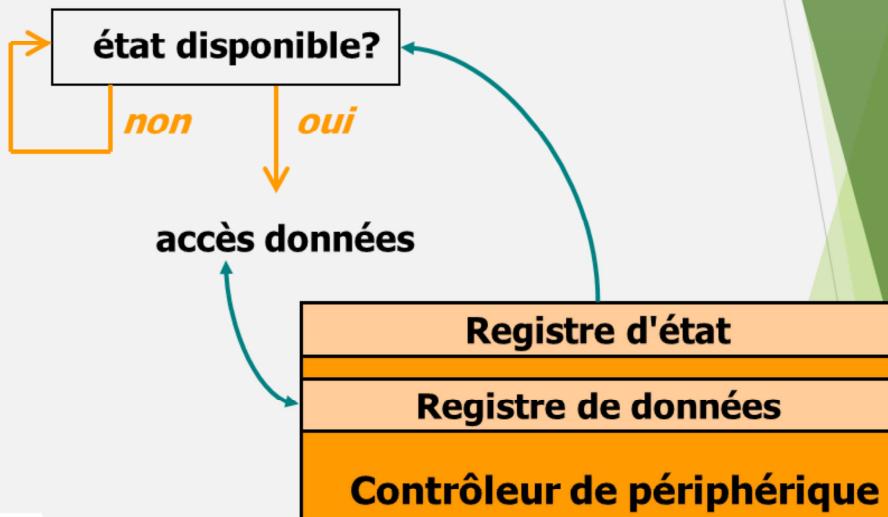
Ce mode est synchrone et induit une charge importante pour le processeur (*attente active*)

Remarque : le polling se justifie dans les cas où le seul travail du processeur est d'attendre un événement extérieur ou dans l'exécution d'une séquence de code non interruptible. On le gardera pour du code système ou pour une machine dédiée.

GESTION DES ENTRÉES-SORTIES DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Attente sur test de mot d'état



Le programme attend la terminaison de l'entrée-sortie en effectuant une boucle de test sur le contenu du registre d'état selon l'algorithme :

```
Faire {  
    lecture du registre d'état  
} tant que (valeur lue != valeur attendue)
```

Le processeur reste donc occupé par le programme pendant le temps de l'entrée-sortie.

Ce mode de dialogue n'est pas adapté au multitâche où une tâche doit pouvoir céder le processeur pendant le déroulement de l'entrée-sortie.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Interruption

- A l'initiative du périphérique
- Prise en compte asynchrone : le processeur prend en compte l'interruption indépendamment du fil d'exécution du programme
- Le processeur reste disponible pendant l'entrée/sortie



Systèmes d'exploitation (Gestion des entrées-sorties)

10

Interruption

Lorsque le contrôleur a une information à transmettre, il envoie un signal d'interruption qui sera pris en compte de manière asynchrone par le processeur. L'attente de transfert n'est plus bloquante. Ce mode limite la charge du processeur, mais le bus système reste très sollicité pour l'échange des données.

Ce mode de dialogue est adapté au multitâche dans la mesure où la tâche libère le processeur pendant le déroulement de l'entrée-sortie.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Interruption

Signal envoyé de manière asynchrone au processeur par un dispositif physique (contrôleur d'interruption, autre processeur, ...) pour l'avertir d'un événement externe.

Les déroutements survenant sur un événement interne sont gérés de la même manière.

- Événements externes : venant des périphériques
- Événements internes (exceptions) : division par zéro, erreur d'adressage, défaut de page, ...



Ici, le terme « asynchrone » signifie que le signal n'est pas envoyé à un moment prévu dans le fil d'exécution d'un programme.

3 types de mécanismes sont gérés par les processeurs Intel :

- interruptions externes (IRQ périphériques) relayées par le contrôleur d'IT -> asynchrone
- « interruptions » internes ou exceptions (proc., mémoire) -> synchrone
- « interruptions » logicielles ou trappes (appels aux fonctions BIOS, appels système) -> synchrone

de manière identique en ce qui concerne la recherche du code à exécuter.

L'adresse de la fonction de traitement de ces interruptions est placée dans une table accédée directement par le processeur.

Remarque

Le terme "interruption logicielle" est un abus de langage, dans la mesure où il ne s'agit pas d'un événement qui se produit de manière asynchrone ou imprévue, mais d'une instruction du programme (INT), exécutée par le processeur de manière synchrone.

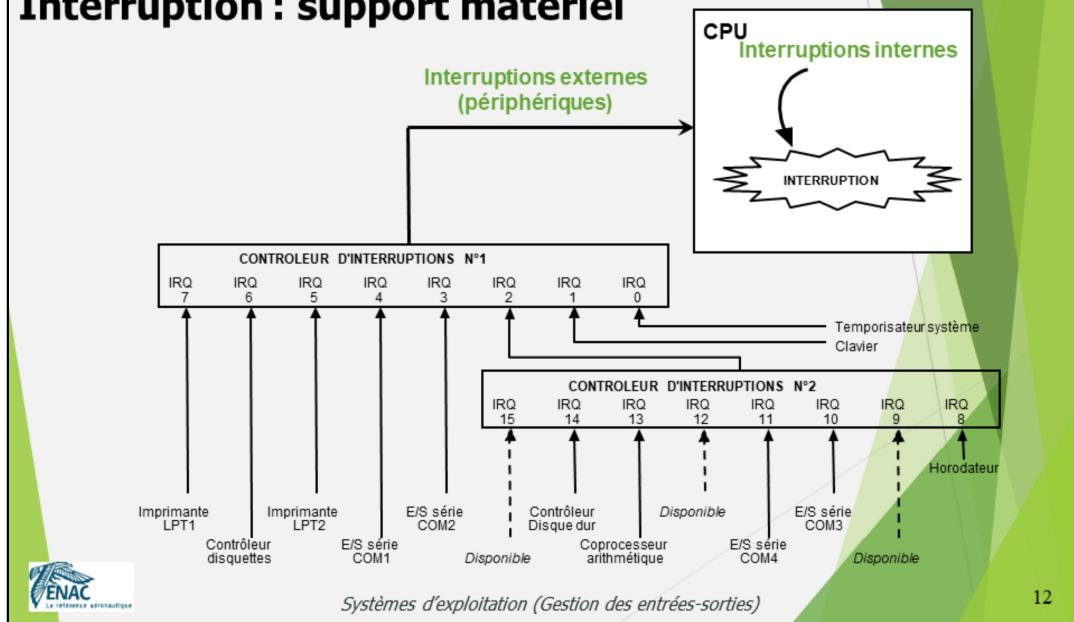
Les interruptions internes s'exécutent aussi de manière synchrone.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Interruption : support matériel



Le schéma ci-dessus représente l'architecture « historique » des cartes mères Intel.

Les périphériques sont connectés à des contrôleurs d'interruption (dispositifs matériels présents sur la carte-mère).

Le contrôleur d'interruption est chargé de relayer le signal au processeur et de lui fournir le numéro du périphérique qui l'a émis (IRQ).

On notera que :

- Les interruptions sont hiérarchisées en niveaux (priorités de traitement des interruptions) ce qui évite les conflits et permet de protéger les processus contre des interruptions peu prioritaires.
- Le nombre de niveaux d'interruptions varie selon l'architecture matérielle et le système

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE

Interruption : support matériel



The screenshot shows the Windows 7 Device Manager interface. It displays three main entries with their resource parameters:

- Horloge système**: Plage d'E/S 0040 - 0043, Plage d'E/S 0050 - 0053, IRQ 0x00000000 (00)
- Keyboard Device Filter**: Plage d'E/S 0060 - 0060, Plage d'E/S 0064 - 0064, IRQ 0x00000001 (01)
- Port de communication (COM1)**: Plage d'E/S 03F8 - 03FF, IRQ 0x00000004 (04)

Gestionnaire de périphériques Windows 7

Systèmes d'exploitation (Gestion des entrées-sorties)

ENAC
Le référentiel aéronautique

13

Les plages d'entrée/sortie correspondent aux plages d'adresses mémoire réservées pour les registres du contrôleur.

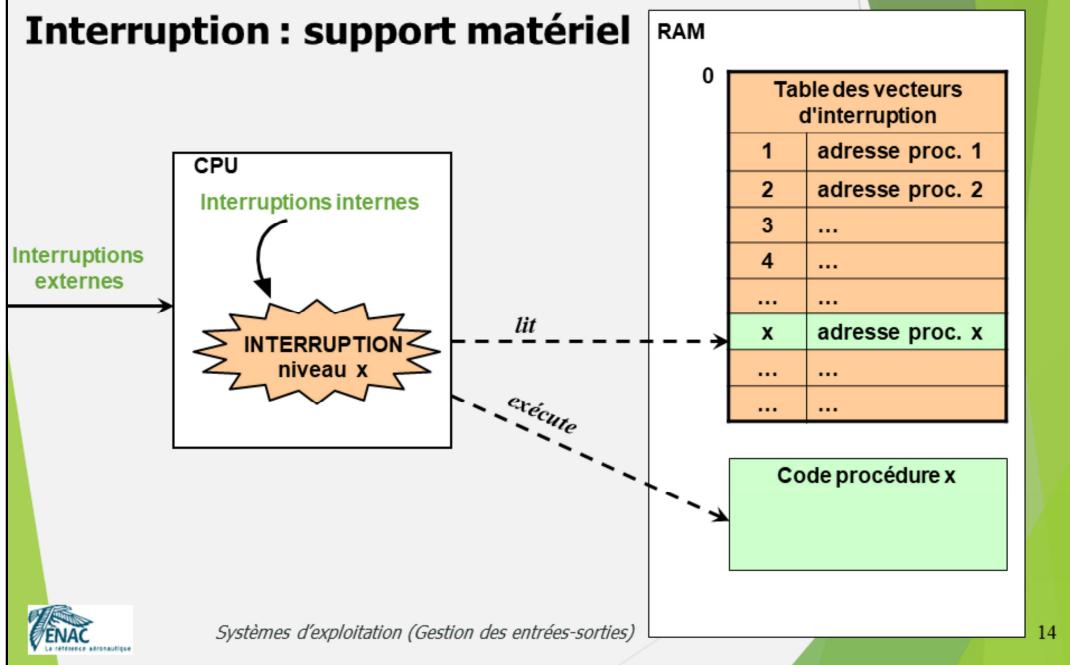
L'IRQ (pour Interrupt ReQuest) correspond au niveau d'interruption, c'est-à-dire au numéro de l'entrée du contrôleur d'interruption sur laquelle est connecté le périphérique.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE



Interruption : support matériel



Lorsque le processeur reçoit un signal en provenance du contrôleur d'interruption (interruption externe), il détermine le niveau d'interruption concerné (numéro d'IRQ).

Ensuite, le mode de traitement des interruptions et des exceptions est le même : lecture de la table des vecteurs d'interruption (IDT : Interrupt Descriptor Table) et exécution du traitement associé.

L'entrée dans la table des vecteurs d'interruption est déterminée à partir du numéro d'IRQ.

GESTION DES ENTRÉES-SORTIES

DIALOGUE PROCESSEUR-PÉRIPHÉRIQUE

Interruption : Vecteurs d'interruption du Pentium



Numéro de vecteur	Usage
0	Division par zéro
1	Exception de débogage
2	Interruption nulle
3	Point d'arrêt
4	Dépassement de capacité de registre interne
5	Dépassement de limite
6	Code d'opération non valide
7	Pérophérique indisponible
8	Erreur sur nombre double précision
9	réservé : gestion coprocesseur
10	segment d'état de tâche invalide
11	segment absent
12	Erreur dans la pile
13	Erreur de protection
14	Défaut de page
15	réservé
16	Erreur dans nombre en virgule flottante
17	Contrôle alignement
18	Contrôle hardware
19 - 31	réservé
32 - 255	Interruptions masquables

Systèmes d'exploitation (Gestion des entrées-sorties)

15

On remarque ici que le début de la table est consacré aux exceptions. Les interruptions matérielles (interruptions dites « masquables ») viennent ensuite.

GESTION DES ENTRÉES-SORTIES

ACCÈS AU PÉRIPHÉRIQUE



Qui accède au périphérique ? Le programme utilisateur ?

Oui, mais

- Programmation lourde et spécifique à un périphérique
- Les "vieux" systèmes le permettent (DOS, ...)
- La plupart des systèmes actuels ne permettent pas de coder des accès directs au matériel (limitation des priviléges)

Les systèmes offrent un service de gestion des entrées-sorties. Les programmes émettent des requêtes à ce service.



Systèmes d'exploitation (Gestion des entrées-sorties)

16

Dans les systèmes d'exploitation utilisant les droits d'accès de la segmentation, les adresses des contrôleurs de périphériques ne font pas partie des segments autorisés en lecture/écriture en mode utilisateur. Les programmes utilisateur ne peuvent donc pas y accéder. Seul le système qui s'exécute en mode noyau le peut.

Exemple de code d'accès à un périphérique (port série) par test de mot d'état en C sous MS-DOS :

```
int main(void) {
    char car;
    while(1) {

        // Lecture d'un caractère sur le port COM2
        while(!(inportb(0x2FD)&0x1)) {}
        car = inportb(0x2F8);

        // Ecriture d'un caractère sur le port COM1
        while(!(inportb(0x3FD)&0x20)) {}
        outportb(0x3F8, car);

    }
    return 0;
}
```

GESTION DES ENTRÉES-SORTIES



Que doit offrir le système ?

- Indépendance programme/périphérique
- Performance
- Partage des périphériques
- Prise en compte des nouveaux périphériques



Systèmes d'exploitation (Gestion des entrées-sorties)

17

Indépendance programme périphérique (abstraction matérielle)

Elle permet la transparence du changement de matériel et la portabilité des programmes d'une configuration matérielle à une autre.

Elle facilite la programmation en utilisant les mêmes fonctions quelque soit le périphérique et en y accédant par un nom (nommage des périphériques).

Elle doit offrir une vision bloquante des entrées/sorties.

Performance

Le processeur doit être libéré pendant l'attente des données.

Le système doit minimiser le nombre d'accès aux périphériques.

Partage des périphériques

Plusieurs tâches pouvant demander à accéder au même périphérique simultanément, il convient de gérer l'accès concurrent et de mettre les demandes en file d'attente.

Prise en compte des nouveaux périphériques

L'installation d'un nouveau périphérique doit être possible sans modification du système.

GESTION DES ENTRÉES-SORTIES



Quelles contraintes doit gérer le système ?

➤ Diversité des périphériques

écran, clavier, souris, imprimante, disque, lecteur/graveur de CD-ROM, lecteur/graveur de DVD, port série, webcam, scanner, capteurs, modem, USB, carte son, ...

➤ Différents modes de transfert des données

- mode bloc (transfert de n octets à la fois)
- mode caractère (transfert d'1 octet à la fois)



GESTION DES ENTRÉES-SORTIES



Quelles solutions adoptées ?



➤ Appels système

Interface de programmation (API) fournie par le système.

Fonctions indépendantes du périphérique, qui est traité comme un fichier.

C standard

```
nbEcrits = fwrite ( buffer, sizeof(char), nbAEcrire, f );
```



Windows

```
WriteFile ( f, buffer, nbAEcrire, &nbEcrits, NULL );
```



Unix

```
nbEcrits = write ( f, buffer, nbAEcrire );
```



Systèmes d'exploitation (Gestion des entrées-sorties)

19

Les systèmes offrent des interfaces de programmation utilisables pour tous les types de périphériques.

Ces fonctions (dites « fonctions enveloppes » ou « wrappers ») encapsulent le basculement en mode noyau et invoquent l'appel système proprement dit par un mécanisme de type interruption logicielle (exécution de code système).

Les mêmes fonctions seront utilisées pour les accès aux périphériques et pour les accès aux fichiers.

Elles assurent l'indépendance du programme vis-à-vis des périphériques.

L'accès à un périphérique par appel système est subordonné à la présence d'un pilote associé à ce périphérique.

GESTION DES ENTRÉES-SORTIES



Quelles solutions adoptées ?



➤ Pilotes de périphériques (drivers)

Un driver par type de contrôleur de périphérique :

- Transmet la requête d'entrée/sortie par écriture dans les registres du contrôleur de périphérique
- Initialise et supervise le périphérique (contrôles et traitement des erreurs)
- Termine le transfert



Les pilotes de périphériques sont des logiciels spécifiques à un système et à un périphérique.

Ils assurent l'indépendance du programme et du système vis-à-vis des périphériques en effectuant les accès au contrôleur et permettent la prise en compte rapide d'un nouveau périphérique : seul le driver est à installer.

La présence de drivers permet l'accès aux périphériques par appels système.

Un pilote contient un certain nombre de fonctions qui constituent des points d'entrée associés aux appels système.

Sous Linux, Les principaux points d'entrée sont :

- **open** : ouverture du périphérique. Détection et initialisation du périphérique.
- **read** : lecture de données sur le périphérique et transmission dans l'espace d'adressage du processus appelant.
- **write** : écriture de données sur le périphérique.
- **release** : fermeture de l'accès au périphérique (correspond à l'appel *close*).
- **ioctl** : configuration du périphérique.

Un appel système déclenche donc l'invocation de la fonction du pilote correspondante.

On notera que les pilotes offrent une représentation des périphériques sous forme de fichiers.

GESTION DES ENTRÉES-SORTIES



Quelles solutions adoptées ?



➤ Amélioration des performances

- Cache des entrées-sorties en mode bloc
- Bufferisation des entrées-sorties en mode caractère

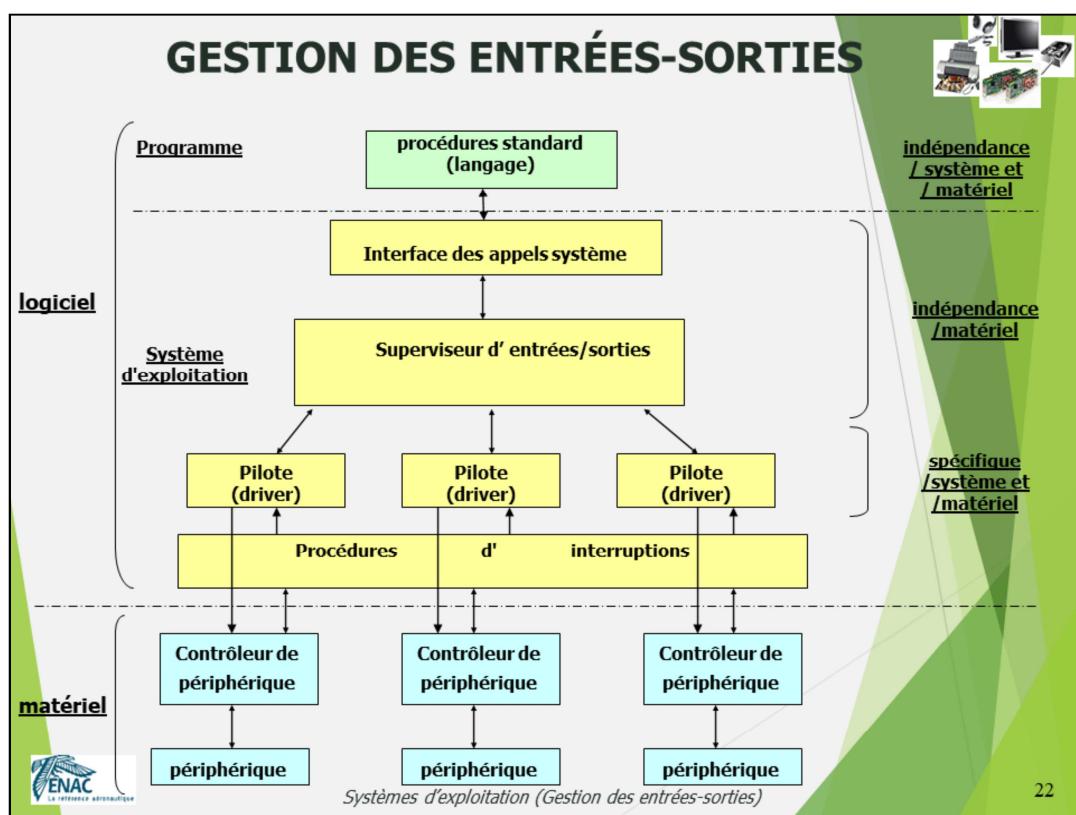


Systèmes d'exploitation (Gestion des entrées-sorties)

21

Amélioration de la performance en réduisant le nombre d'accès aux périphériques.

GESTION DES ENTRÉES-SORTIES



22

Le superviseur d'entrée-sortie assure :

- la gestion des chemins d'accès et des files d'attente de chaque ressource,
- la gestion du cache (mode bloc) et de la bufferisation (mode caractère),
- la construction et la transmission des requêtes aux pilotes de périphérique,
- la signalisation des erreurs.

Les procédures d'interruptions sont invoquées sur réception d'une requête d'interruption et s'appuient généralement sur les drivers pour l'accès au périphérique.

Systèmes d'exploitation

Le système d'exploitation

Gestion des fichiers



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (Gestion des fichiers)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers**

La virtualisation

Architecture des systèmes

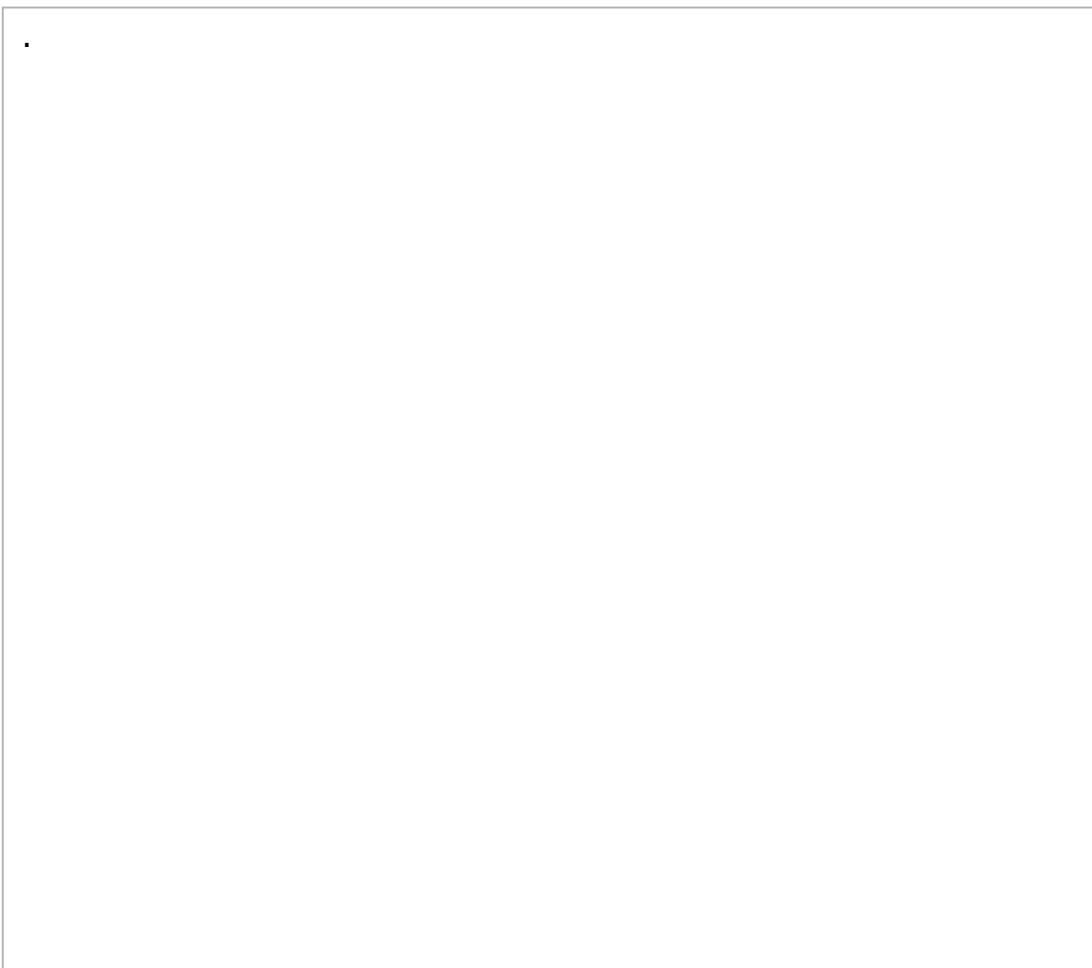
LE SYSTÈME D'EXPLOITATION

GESTION DES FICHIERS



Systèmes d'exploitation (Gestion des fichiers)

3



GESTION DES FICHIERS

DEFINITIONS



Un fichier est une collection de données.

Point de vue de l'utilisateur

- ensemble logique,
- nom,
- chemin d'accès

Point de vue physique

- support (disque dur, CD-ROM, mémoire flash, ...),
- accès aux données pris en charge par le système

Point de vue du programmeur

- structure de l'information,
- mode d'accès



GESTION DES FICHIERS

CAHIER DES CHARGES



- Proposer aux utilisateur une vue logique et non physique des fichiers,
- Offrir une organisation hiérarchique (arborescence de répertoires),
- Donner des outils de manipulation (déplacement, suppression, renommage, fusion, ...),
- Gérer les autorisations d'accès et le partage des fichiers.



Systèmes d'exploitation (Gestion des fichiers)

5

GESTION DES FICHIERS

VUE PHYSIQUE vs. VUE LOGIQUE



Fichier physique

Ensemble de blocs de taille fixe, ayant un emplacement précis sur le support, non obligatoirement contigus.
L'accès au fichier se résume à l'accès à ces blocs.

Fichier logique

Entité logique possédant un nom, stockée sur une mémoire auxiliaire (disque dur, DVD-ROM, clé USB, ...).

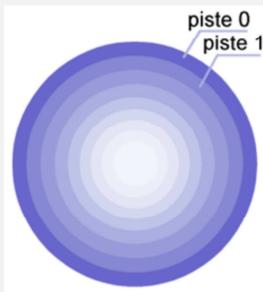


GESTION DES FICHIERS

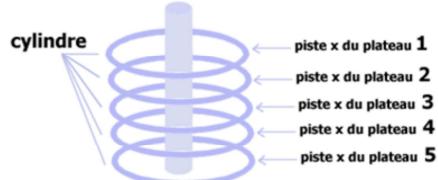
SYSTÈME DE GESTION DES FICHIERS



Disque physique



Formatage physique
(bas niveau)



Systèmes d'exploitation (Gestion des fichiers)

7

L'unité d'accès minimale à un disque dur est un bloc de 512 octets. Un disque peut donc être vu comme un ensemble de blocs numérotés (coordonnées du bloc).

Remarque : ce découpage est issu de la structure des disques magnétiques. Pour garantir l'indépendance programmes/matériel, les contrôleurs de mémoires permanentes (disques SSD, clés USB, ...) présentent la même organisation à l'utilisateur.

GESTION DES FICHIERS

SYSTÈME DE GESTION DES FICHIERS



Disque logique = partition

1 partition = 0 ou 1 système de fichier

Outils de partitionnement : fdisk, GParted, ...

Formatage (haut niveau)

Écriture par un système des informations permettant de gérer l'accès aux données de la partition => création d'un **système de fichiers**



Ne pas confondre avec le formatage physique



Un disque physique peut être « découpé » en plusieurs disques logiques ou partitions. Chaque partition sera vue par l'utilisateur comme un disque indépendant. Ce « découpage » est appelé **partitionnement**.

Un **système de fichiers** (File System) est créé par l'opération de **formatage** d'une partition.

Le formatage consiste en l'écriture sur la partition des structures de données nécessaires au système pour mémoriser l'emplacement des fichiers et pour y accéder ultérieurement.

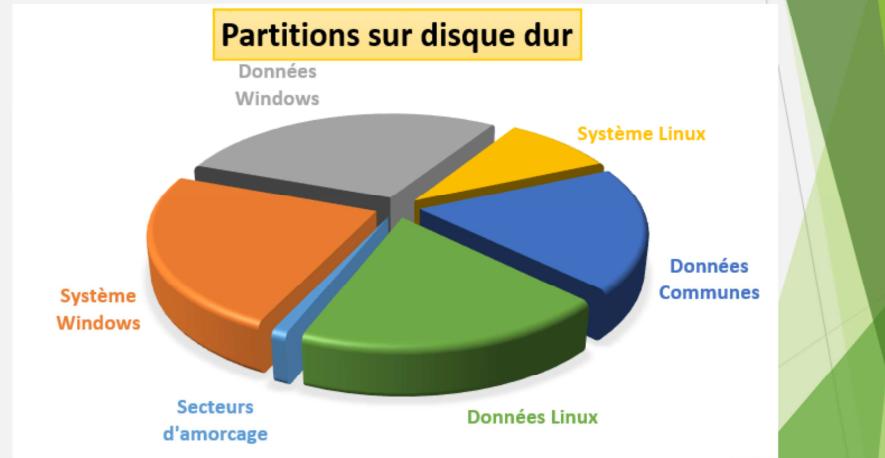
Si un disque dur est partitionné, chaque partition devra être formatée séparément.

GESTION DES FICHIERS

SYSTÈME DE GESTION DES FICHIERS



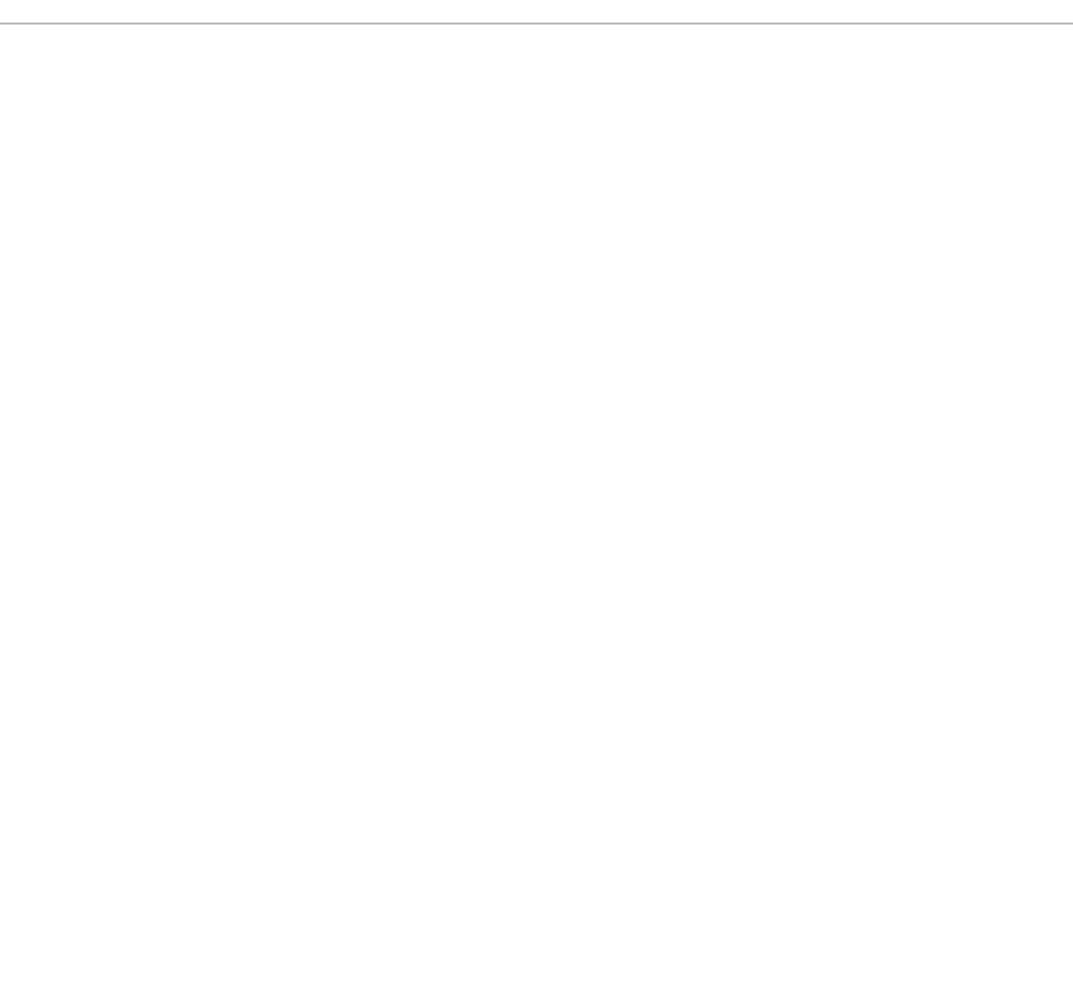
Exemple de partitionnement d'un disque dur



Source : <https://apcpedagogie.com/>

Systèmes d'exploitation (Gestion des fichiers)

9

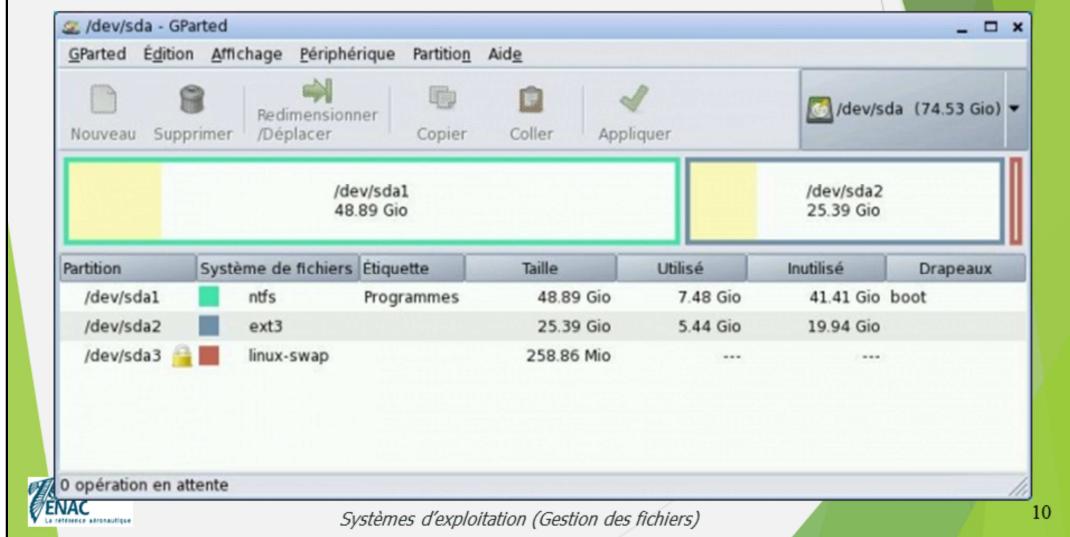


GESTION DES FICHIERS

SYSTÈME DE GESTION DES FICHIERS



Exemple de disque dur après partitionnement et formatage



L'outil GParted nous montre ici un disque dur contenant 3 partitions :

/dev/sda1 : partition Windows (système de fichiers NTFS)

/dev/sda2 : partition Linux (système de fichiers ext3)

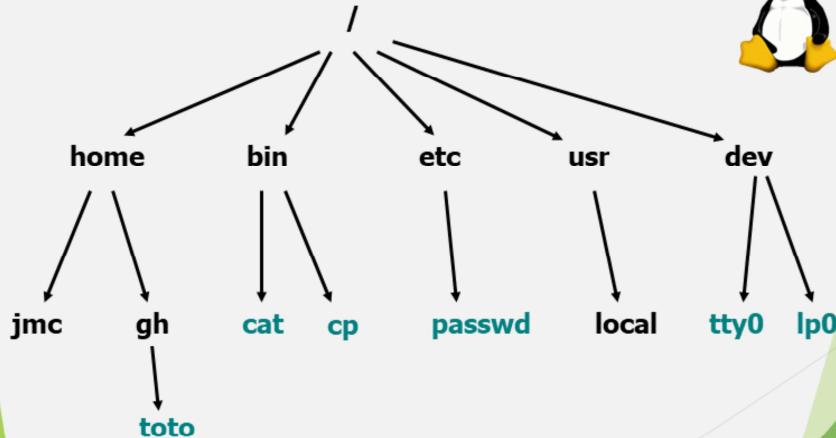
/dev/sda3 : partition Linux (espace d'échange linux-swap pour gestion de la mémoire virtuelle)

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX



Exemple d'arborescence de fichiers



Systèmes d'exploitation (Gestion des fichiers)

11

Les systèmes de fichiers pour les systèmes Unix présentent tous la même vision à l'utilisateur : un répertoire racine unique où les différents volumes logiques ou physiques sont vus comme des sous-dossiers (mécanisme de montage de système de fichiers sur un système racine).

Dans le monde Unix, on pourra citer les systèmes de fichiers suivants :
ufs, nfs (système de fichiers en réseau), reiserfs, ext2, ext3, ext4, ...

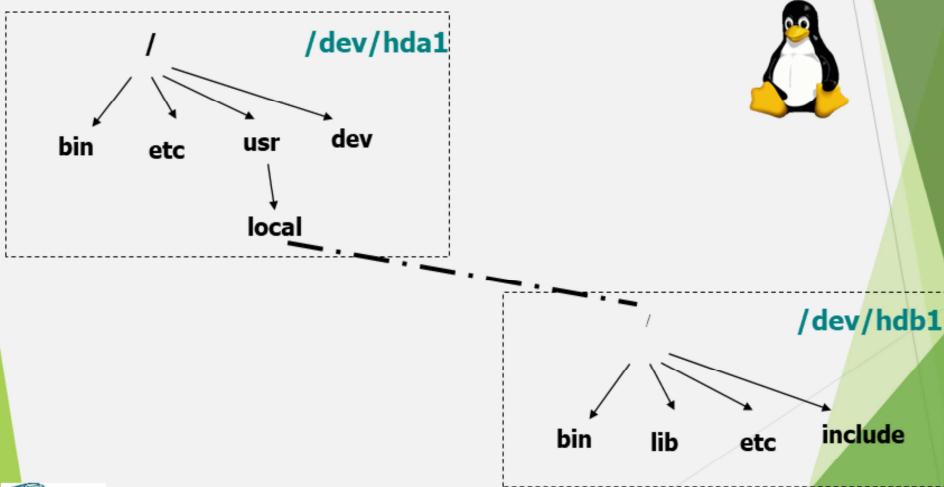
On notera que Linux sait aussi accéder en lecture et écriture aux systèmes de fichiers Windows (FAT et NTFS).

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX



Exemple de montage de système de fichiers



Systèmes d'exploitation (Gestion des fichiers)

12

Dans cet exemple une partition située sur un deuxième disque dur (`/dev/hdb1`) est rattachée au dossier `/usr/local` de la partition système située sur un premier disque (`/dev/hda1`).

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX



Accès aux blocs physiques



➤ un fichier -> un **i-node** (table d'informations)

➤ **Répertoire (= catalogue, dossier)**

1 entrée dans le répertoire par fichier :

- nom,
- n° d'i-node



A la création d'un fichier, une table d'informations appelée i-node est écrite dans les structures de données du système de fichiers. Cet i-node contient toutes les informations concernant ce fichier, excepté ses données.

Un i-node possède un numéro unique dans le système de fichiers.

On parle de nœud pour désigner tout élément présent dans le système de fichier (fichier « ordinaire », dossier, fichier périphérique, ...).

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX



Accès aux blocs physiques

Contenu d'un i-node



- N° du périphérique logique contenant le dossier père du nœud
- Type du nœud (dossier, fichier spécial, fichier ordinaire, ...)
- Droits d'accès
- Taille en octets
- Nombre de liens
- Utilisateur propriétaire
- Groupe propriétaire
- Dates (création, dernier accès, dernière modification)
- Adresses des blocs de données physiques



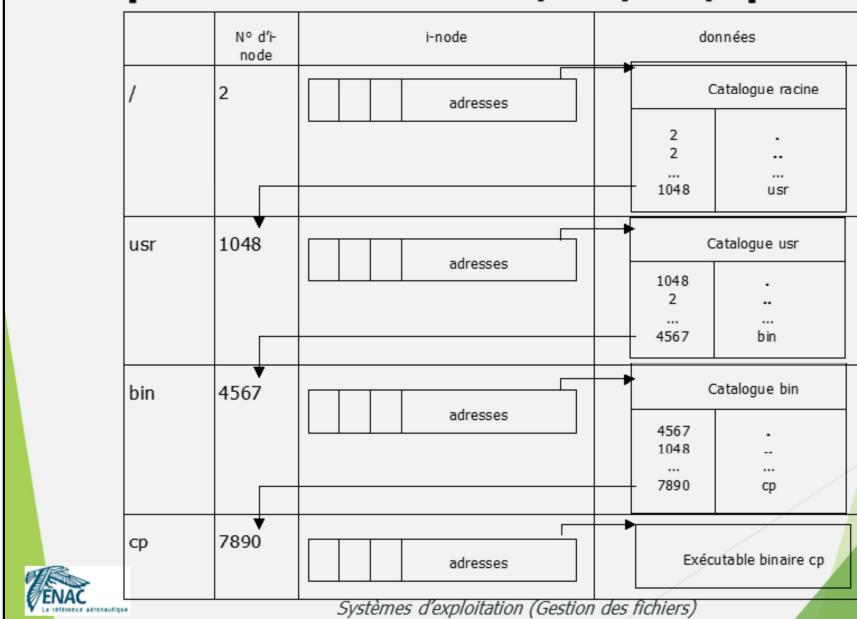
Systèmes d'exploitation (Gestion des fichiers)

14

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX

Exemple : accès au fichier /usr/bin/cp



15

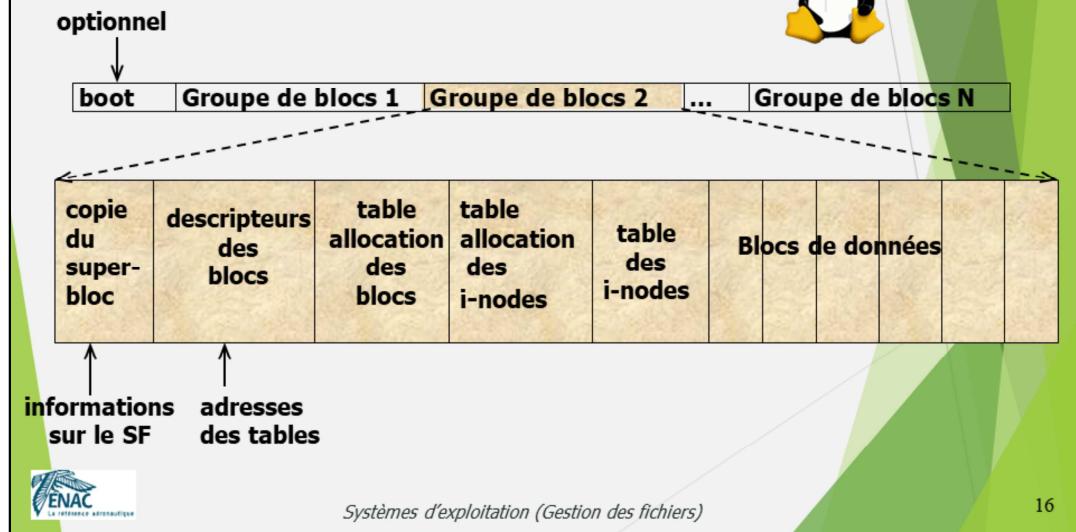
La recherche part du dossier racine dont le numéro d'inode est toujours égal à 2.

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS UNIX



Structure du système de fichier ext2



GESTION DES FICHIERS

GESTION DES FICHIERS SOUS WINDOWS



Exemple d'arborescence de fichiers



Systèmes d'exploitation (Gestion des fichiers)

17

Contrairement aux systèmes de fichiers Unix, les systèmes de fichier Windows présentent à l'utilisateur la vision d'une arborescence par volume logique (partition).

Historique des systèmes de fichier Microsoft :

Système	Systèmes de fichiers supportés
DOS	FAT
Windows 95	FAT
Windows 95 OSR2	FAT, FAT32
Windows 98	FAT, FAT32
Windows NT4	FAT, NTFS (version 4)
Windows 2000	FAT, FAT32, NTFS (versions 4 et 5)
Windows XP, Vista, 7, 8, 10	FAT, FAT32, NTFS

On notera que Windows ne sait accéder qu'aux systèmes de fichier FAT et NTFS.

GESTION DES FICHIERS

GESTION DES FICHIERS SOUS WINDOWS



Accès aux blocs physiques en système NTFS



= Master File Table + données

Système constitué d'une table (MFT) et des blocs de données

Une entrée (« enregistrement ») dans la MFT par fichier

- fichier court (<1ko) : le fichier est dans l'entrée de MFT
- fichier long : l'enregistrement contient les liens vers les blocs physiques présents en zone de données



GESTION DES FICHIERS

GESTION DES FICHIERS SOUS WINDOWS



Windows 10

Accès aux blocs physiques en système NTFS

- Fichier < 1ko

Infos Standard	Nom	Descripteur Sécurité	Données
----------------	-----	----------------------	---------

- Fichier > 1ko

Infos Standard	Nom	Descripteur Sécurité	"Chaînage"
----------------	-----	----------------------	------------



(source J.M. Arjona)

Systèmes d'exploitation (Gestion des fichiers)

19



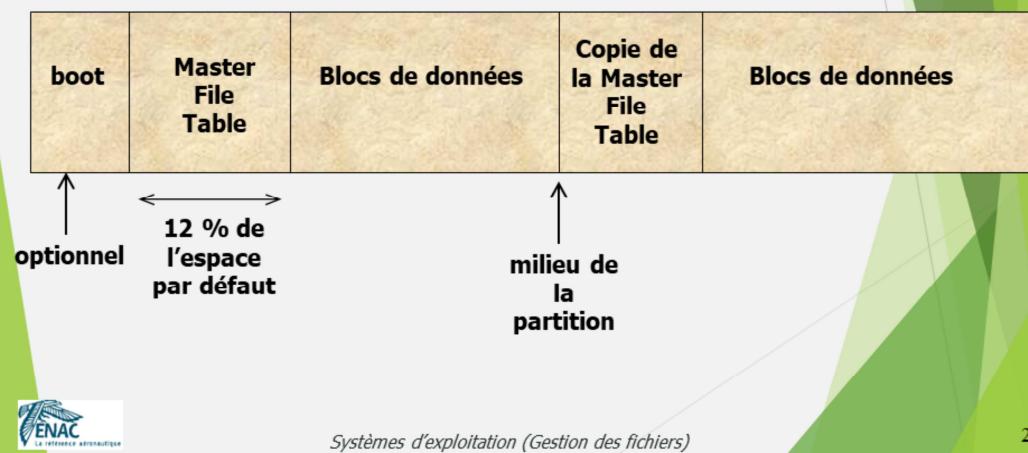
GESTION DES FICHIERS

GESTION DES FICHIERS SOUS WINDOWS



Windows 10

Structure du système NTFS



GESTION DES FICHIERS

DROITS D'ACCÈS AUX FICHIERS



- Système multi-utilisateur => droits d'accès aux ressources
- Implémentation très différente selon le système
- Deux exemples :
 - Systèmes de fichiers UNIX
 - Système de fichiers NTFS (Windows NT, 2000, XP, Vista, 7, 8, 10)



Systèmes d'exploitation (Gestion des fichiers)

21

Les autorisations dans un système multiutilisateurs sont principalement mises en œuvre à travers le système de fichiers.

GESTION DES FICHIERS

DROITS D'ACCÈS AUX FICHIERS



Droits d'accès sous Unix

- Concernent tous les types de ressources (fichiers, répertoires, périphériques, pipes, sockets, ...)
- 3 types de droits : lecture, écriture, exécution
- Ces droits sont attribués pour 3 catégories d'utilisateurs
 - le propriétaire du fichier
 - les utilisateurs du groupe propriétaire
 - les autres utilisateurs

-> *Remarque: root a tous les droits*



Systèmes d'exploitation (Gestion des fichiers)

22

Droit en lecture (r) : visualisation ou lecture du contenu

Droit en écriture (w) : modification du contenu

Droit en exécution (x) : exécution (programme), entrée (dossier)

Remarque : il existe sous Unix 3 autres types de droits dits « droits spéciaux » : bit setuid, bit setgid et sticky bit. Ces droits remplacent le droit d'exécution pour respectivement le propriétaire, le groupe et les autres utilisateurs.

GESTION DES FICHIERS

DROITS D'ACCÈS AUX FICHIERS



Windows 10

Droits d'accès sous Windows/NTFS

- Concernent les fichiers et répertoires
- Les droits peuvent être définis pour des utilisateurs particuliers et des groupes d'utilisateurs
- Les autorisations NTFS sont constituées de permissions et d'interdictions. Les interdictions sont prioritaires sur les autorisations (i.e. tout ce qui n'est pas strictement interdit peut être autorisé)
- Les autorisations sont regroupées en groupes logiques d'autorisation



Systèmes d'exploitation (Gestion des fichiers)

23

Autorisations spéciales	Contrôle total	Modification	Lecture et exécution	Afficher le contenu du dossier	lecture	écriture	
Parcourir le dossier/Exécuter le fichier	x	x	x	x			
Liste du dossier/Lecture de données	x	x	x	x	x		
Attributs de lecture	x	x	x	x	x		
Lire les attributs étendus	x	x	x	x	x		
Création de fichiers/écriture de données	x	x				x	
Création de dossiers/Ajout de données	x	x				x	
Attributs d'écriture	x	x				x	
écriture d'attributs étendus	x	x				x	
Suppression de sous-dossiers et de fichiers	x						
Supprimer	x	x					
Autorisations de lecture	x	x	x	x	x	x	
Modifier les autorisations	x						
Appropriation	x						
Synchroniser	x	x	x	x	x	x	

Autorisations spéciales	Contrôle total	Modification	Lecture et exécution	Lecture	écriture	
Parcourir le dossier/Exécuter le fichier	x	x	x	-	-	
Liste du dossier/Lecture de données	x	x	x	x	-	
Attributs de lecture	x	x	x	x	-	
Lire les attributs étendus	x	x	x	x	-	
Création de fichiers/écriture de données	x	x	-	-	x	
Création de dossiers/Ajout de données	x	x	-	-	x	
Attributs d'écriture	x	x	-	-	x	
écriture d'attributs étendus	x	x	-	-	-	
Suppression de sous-dossiers et de fichiers	x	-	-	-	-	
Supprimer	x	x	-	-	-	
Autorisations de lecture	x	x	x	x	-	
Modifier les autorisations	x	-	-	-	-	
Appropriation	x	-	-	-	-	
Synchroniser	x	x	x	x	x	

Dossiers

Fichiers

Systèmes d'exploitation

Pour aller plus loin ...

La virtualisation



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

➤ Le système d'information

- Fonctions de base
- Représentation des données
- Le matériel
- Les programmes

➤ Le système d'exploitation

- Notions fondamentales
- Principes et mise en œuvre

➤ Pour aller plus loin ...

- Virtualisation
- Architecture



Systèmes d'exploitation (La virtualisation)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

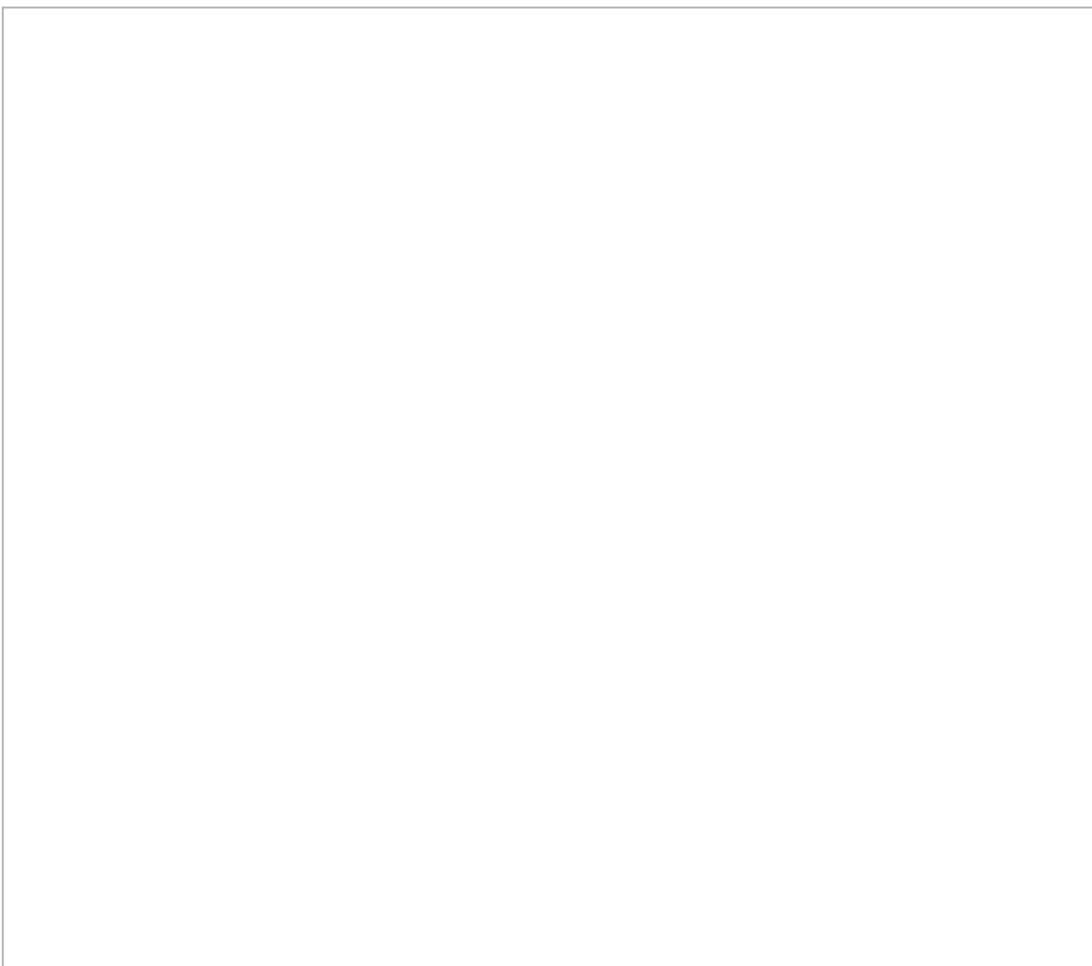
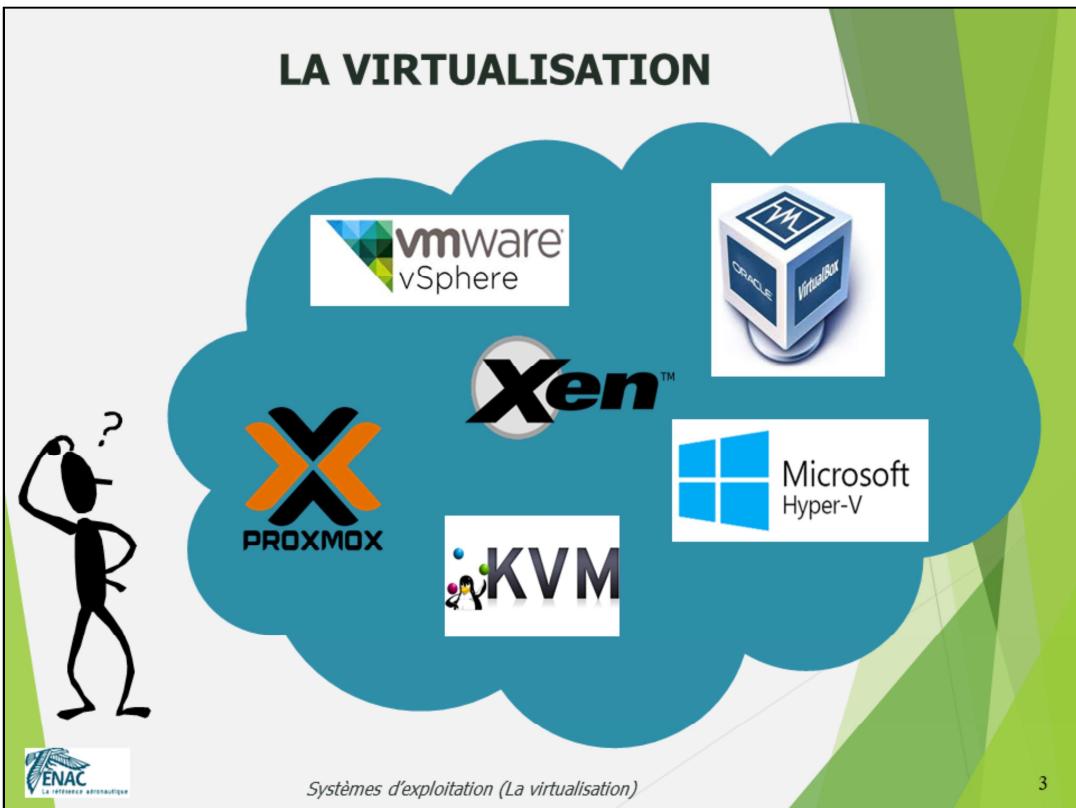
Le système d'exploitation

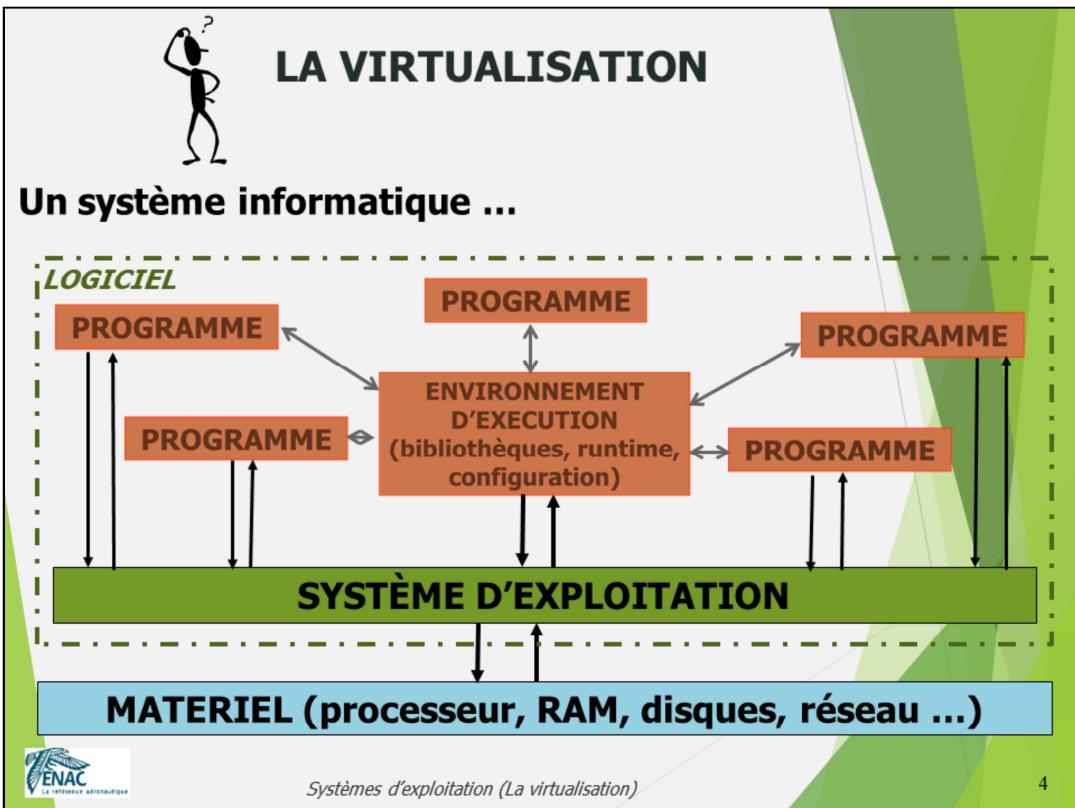
- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

- Architecture des systèmes

LA VIRTUALISATION





Un système informatique est composé de matériel (processeur, mémoire, périphériques) et de logiciel (système d'exploitation, programmes).

Ces différents composants sont fortement liés dans un système informatique traditionnel :

- Un programme s'exécute sur un système d'exploitation donné et nécessite des ressources particulières (bibliothèques, interpréteur pour certains langages, configuration, ...).
- Certains systèmes d'exploitation sont dédiés à un matériel spécifique (par exemple Windows ou MacOS).

LA VIRTUALISATION

PROBLÉMATIQUE



Que se passe-t-il :

- Si les programmes ont besoin de versions d'environnement d'exécution différentes ?
- Si les programmes ont besoin de systèmes d'exploitation différents ?
- Si le matériel est sous-exploité ?

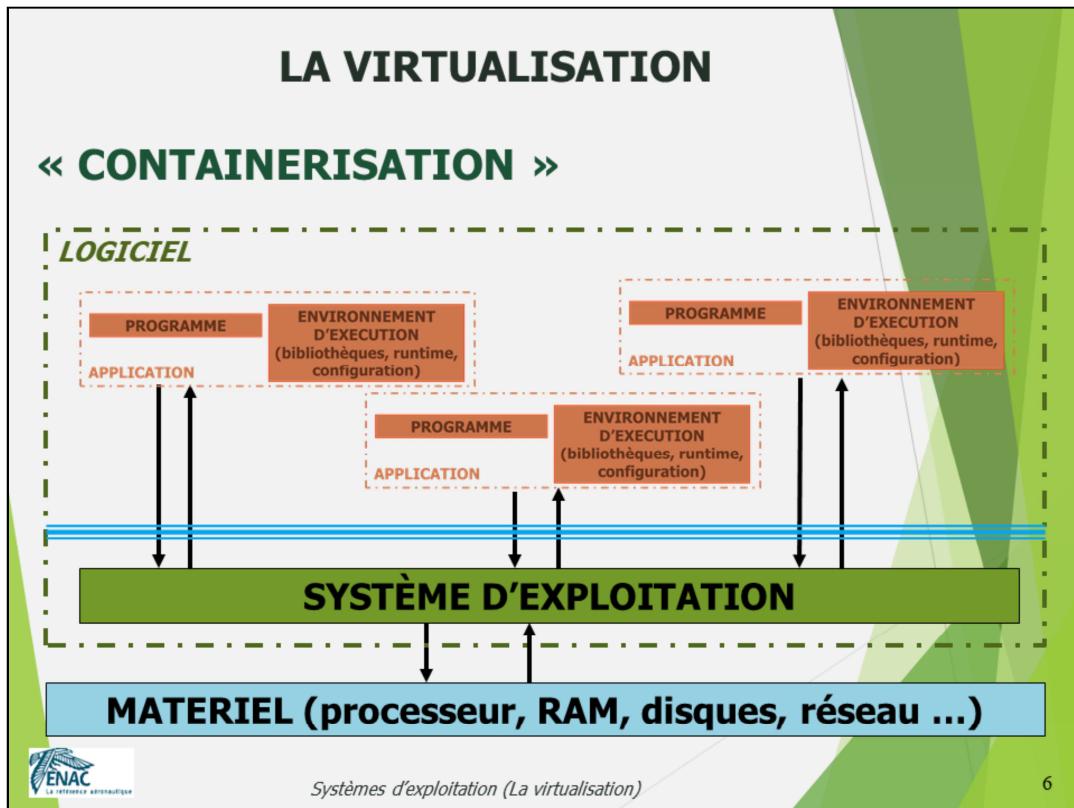


Systèmes d'exploitation (La virtualisation)

5

LA VIRTUALISATION

« CONTAINERISATION »



La containerisation permet de répondre au problème de la cohabitation de programmes nécessitant des environnements d'exécutions différents.

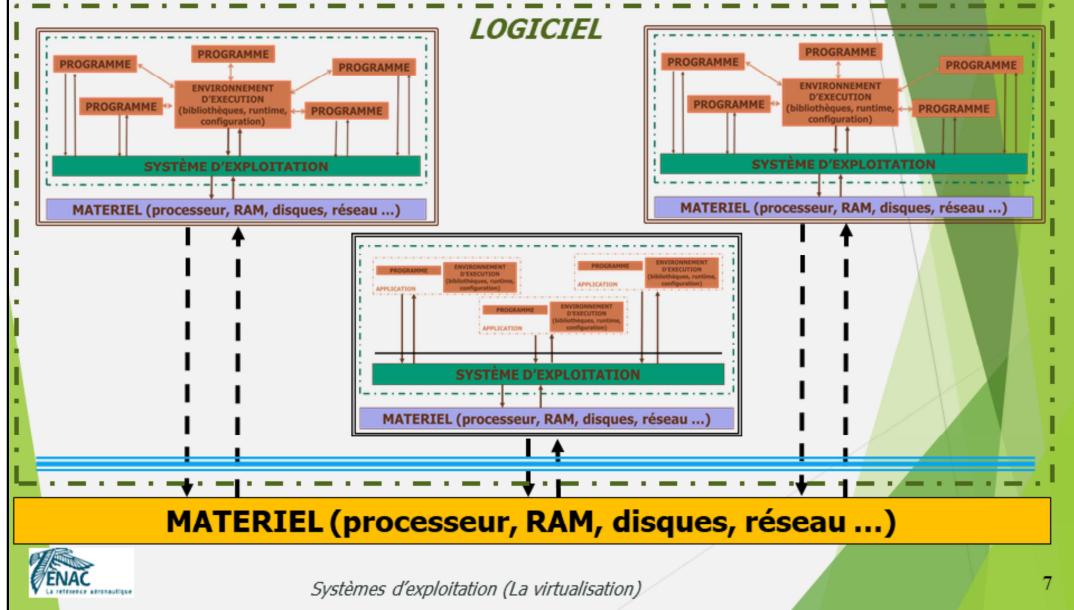
Chaque « container » contient tous les éléments nécessaires à l'exécution d'une application.

Il ne s'agit pas vraiment de virtualisation.

Exemple de logiciel de containerisation : Docker

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES



La virtualisation de systèmes permet de faire cohabiter plusieurs systèmes d'exploitation sur une même machine physique.

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Faire tourner plusieurs systèmes d'exploitation en parallèle sur une même machine physique

- Hyperviseur : logiciel gérant la virtualisation d'un ou plusieurs OS
- 2 types de virtualisation
 - Type 1 (bare-metal)
 - Type 2 (hosted)



Systèmes d'exploitation (La virtualisation)

8

Le principe de la virtualisation est d'exécuter un système d'exploitation dans une machine virtuelle, chaque machine physique étant capable de faire tourner une ou plusieurs machines virtuelles.

Les concepts de la virtualisation logicielle ont été théorisés en 1974 par Popek et Goldberg¹

Ils définissent les termes de « machine virtuelle » pour désigner l'environnement virtualisé et celui de « moniteur de machine virtuelle » qu'on appellera hyperviseur.

La virtualisation est basée sur les trois critères suivants :

- Equivalence : tout programme s'exécutant dans l'environnement virtualisé a le même comportement que sur la machine d'origine (à l'exception éventuelle de la disponibilité des ressources et de la dépendance aux temps de réponse).
- Efficacité : la majeure partie des instructions du processeur virtuel sont exécutées par le processeur physique afin de réduire le surcoût de temps d'exécution lié à l'interprétation de ces instructions.
- Contrôle des ressources : le moniteur de machine virtuelle est maître de l'attribution des ressources aux programmes s'exécutant dans l'environnement virtualisé.

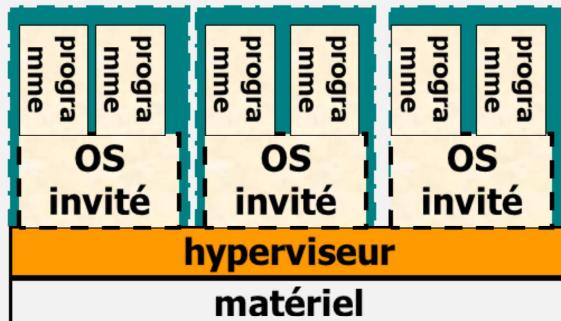
1. Popek, G.J., and Goldberg, R.P. Formal Requirements for Virtualizable Third Generation Architectures, Communications of the ACM, July 1974, Volume 17, Number 7

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Hyperviseur de type 1 (hyperviseur bare-metal)

Logiciel s'exécutant directement sur la plate-forme matérielle



Exemples : Proxmox, Citrix XenServer, VMware vSphere Hypervisor (ESXi), Microsoft Hyper-V, Linux KVM



Un hyperviseur de type 1 est installé directement sur la couche matérielle. Il s'agit d'un noyau allégé et optimisé pour la virtualisation de machines.

Au démarrage de la machine physique, l'hyperviseur prend directement le contrôle du matériel, et alloue l'intégralité des ressources aux machines hébergées.

L'intérêt de ce type d'hyperviseur est qu'il permet d'allouer la quasi-totalité des ressources disponibles aux machines virtuelles.

Un seul hyperviseur de type 1 peut être installé sur une machine physique. Pour virtualiser beaucoup de machines, ou des machines demandant un nombre de ressources conséquent, il faudra disposer d'une machine physique disposant d'une puissance équivalente à l'intégralité des machines virtualisées, ou prévoir plusieurs machines physiques avec d'autres hyperviseurs.

Virtualiser les serveurs d'une entreprise permet d'utiliser de façon optimale les ressources matérielles, tout en mutualisant la consommation électrique et la maintenance. Cela permet également d'optimiser l'installation, le déploiement et la migration des machines, et de faciliter la mise en production des serveurs.

L'hyperviseur devra être :

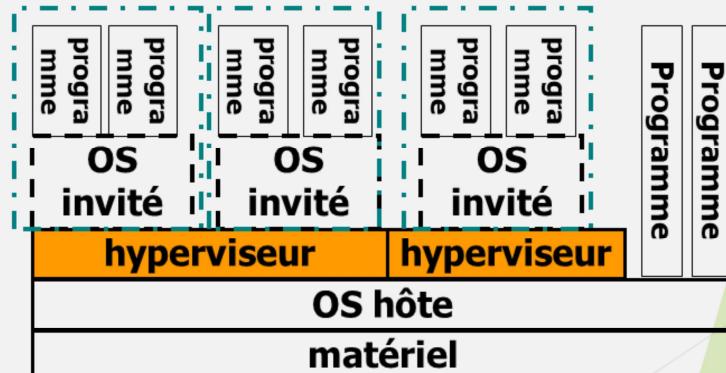
- compatible avec le matériel sur lequel il s'exécute,
- compatible avec les OS qui sont installés dans les machines virtuelles.

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Hyperviseur de type 2 (hosted)

Logiciel s'exécutant sur un système d'exploitation



Exemples : Oracle VM Virtual Box, VMware Workstation, ...



Un hyperviseur de type 2 est installé au dessus d'un système d'exploitation. Il se lance la plupart du temps comme une simple application.

L'intérêt de ce type d'hyperviseur est la possibilité d'en exécuter plusieurs en même temps car ceux-ci ne s'installent pas directement sur la couche matérielle.

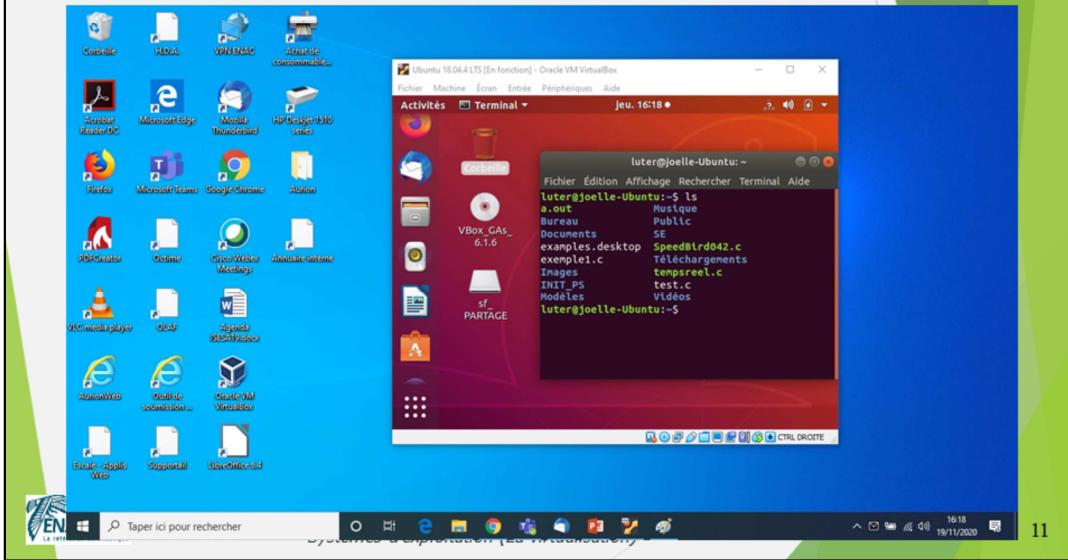
En revanche, ce type d'hyperviseur ne peut pas fournir autant de ressources matérielles que les hyperviseurs de type 1 puisqu'il est installé sur un système d'exploitation, lui-même consommateur de ressources.

On installera la plupart du temps un hyperviseur de type 2 sur un poste client (type PC) en alternative au multiboot ou pour effectuer des tests de configuration.

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Exemple : Linux Ubuntu dans une VM Virtual Box sous Windows 10 (virtualisation hosted)



Dans la copie d'écran ci-dessus, on voit tourner un système Linux Ubuntu dans une fenêtre sous Windows 10.

Le logiciel de virtualisation utilisé est Oracle Virtual Box grâce auquel on a créé une machine virtuelle dans laquelle a été installée une image de la distribution Linux Ubuntu 18.04 LTS.

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Problématique

- Exécution du noyau d'un système dans l'espace utilisateur
 - protection mémoire ?
 - instructions réservées ?
- Déroutement des exceptions
 - performances ?
 - instructions ne lançant pas d'exceptions ?
- Gestion mémoire et accès aux périphériques ?



Systèmes d'exploitation (La virtualisation)

12

En mode utilisateur, l'accès à la mémoire est limité (contrôles effectués par la MMU) et une partie du jeu d'instructions du processeur n'est pas accessible.

Une opération interdite provoque dans la plupart des cas le lancement d'une exception. Le principe de base de la virtualisation est de dérouter ces exceptions pour remplacer le traitement d'erreur associé par un traitement spécifique.

Le déroutement des exceptions pose 2 problèmes :

- le code des exceptions étant remplacé par un code d'émulation, le temps d'exécution est plus long
- certaines instructions processeur peuvent être exécutées en mode utilisateur ou en mode noyau, elles ne déclenchent donc pas d'exception pouvant être utilisée pour la virtualisation.

Le partage de la mémoire physique et l'accès aux périphériques devront être aussi mis en œuvre.

LA VIRTUALISATION

VIRTUALISATION DE SYSTÈMES

Comment ça marche ?

- Support matériel de la virtualisation
 - instructions
 - MMU
 - pas pour toutes les architectures
- Virtualisation complète
 - OS invité standard
- Paravirtualisation
 - OS invité adapté (pilotes spécifiques)



Systèmes d'exploitation (La virtualisation)

13

Virtualisation, paravirtualisation ou émulation ?

Plusieurs types de virtualisation existent, chacun ayant ses propres avantages et inconvénients. Ainsi, il existe des solutions de **virtualisation complète**, l'hyperviseur se chargeant de créer un environnement virtuel complet en **simulant du « faux » matériel**. Le système d'exploitation invité n'aura alors accès qu'à ces ressources simulées, et non aux ressources matérielles réelles. Ce type de virtualisation est toutefois limité aux systèmes d'exploitation prévus pour la même **architecture matérielle** (x86, x64, ARM, ...) que le processeur physique de la machine hôte.

Pour dépasser cette limite, il faut faire appel à l'**émulation** : l'hyperviseur crée alors un environnement virtuel complet, en allant jusqu'à **simuler un microprocesseur** qui peut alors avoir une architecture matérielle différente de celle du CPU hôte. Le principal inconvénient de ce type de solution est alors le niveau de performances, souvent médiocre.

La **paravirtualisation** est un autre type de solution de virtualisation. Ici, le système d'exploitation invité est conscient de s'exécuter dans un environnement virtualisé, ce qui nécessite bien entendu certaines modifications logicielles (par exemple l'installation de pilotes ou d'une surcouche logicielle). En contrepartie, il devient capable d'interagir avec l'hyperviseur et de lui demander, le cas échéant, de transmettre directement les appels systèmes au matériel du serveur hôte. Les performances « virtuelles » sont alors théoriquement proches de celles qu'il serait possible d'atteindre avec le matériel réel.

»

(Yannick Guerrini, <http://www.tomshardware.fr/articles/virtualisation-serveur,2-779-2.html>)

Systèmes d'exploitation

Pour aller plus loin ...

Architecture des systèmes d'exploitation



SINA/INF/SAR

Joëlle Luter

2020



PLAN DU COURS

- **Le système d'information**
 - Fonctions de base
 - Représentation des données
 - Le matériel
 - Les programmes
- **Le système d'exploitation**
 - Notions fondamentales
 - Principes et mise en œuvre
- **Pour aller plus loin ...**
 - Virtualisation
 - Architecture



Systèmes d'exploitation (Architecture)

2

Plan détaillé

Le système d'information

- Définition
- Fonctions de base
- Représentation de l'information
- Le matériel

Les programmes

- Définitions
- Programme source
- Exécution d'un programme
- Chaîne de compilation
- Interprétation

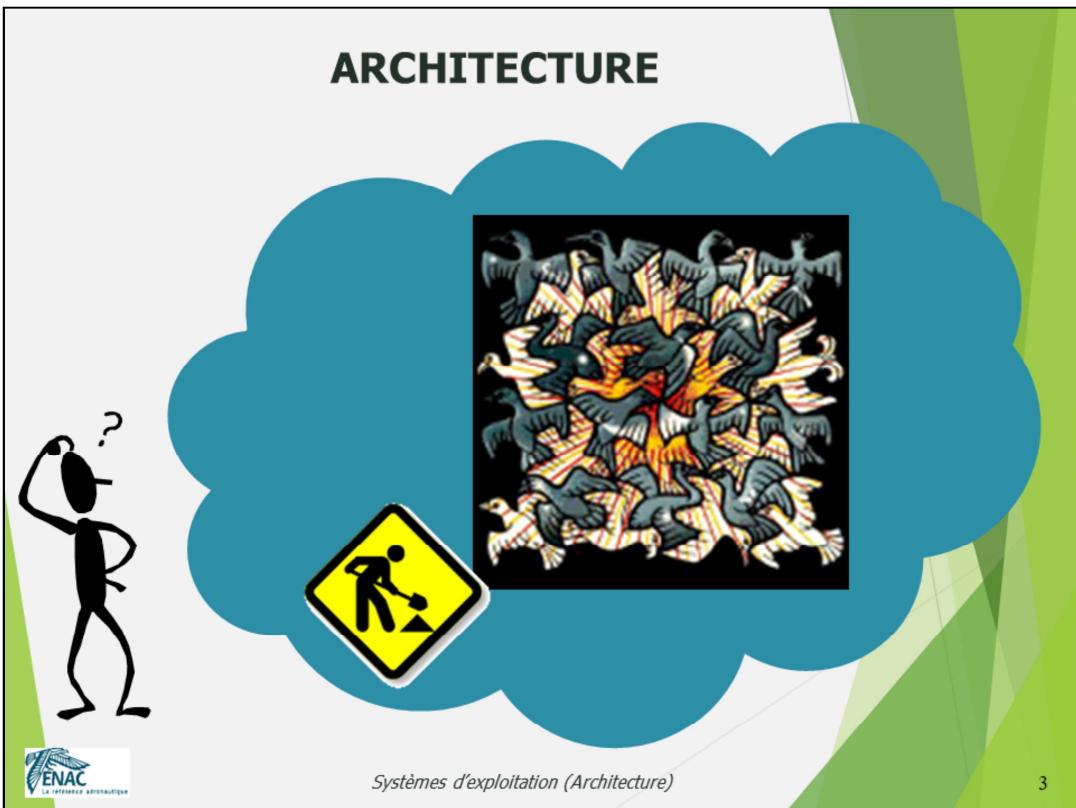
Le système d'exploitation

- Notions fondamentales
- Gestion de la mémoire
- Gestion du processeur
- Gestion des entrées-sorties
- Gestion des fichiers

La virtualisation

Architecture des systèmes

ARCHITECTURE



L'architecture d'un logiciel décrit son découpage en différents composants et la manière dont ceux-ci interagissent.

ARCHITECTURE

DÉFINITION

- Organisation logicielle du noyau du système d'exploitation
- Différents types d'architecture :
 - Monolithique
 - Multi-couches
 - Micro-noyau

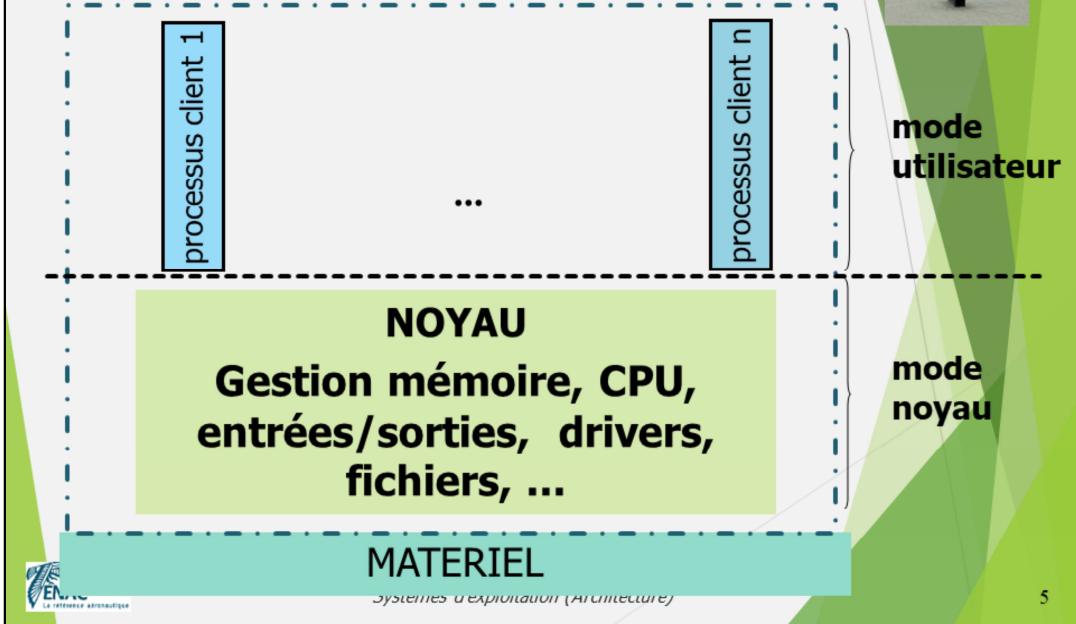


Systèmes d'exploitation (Architecture)

4

ARCHITECTURE

ARCHITECTURE MONOLITHIQUE



Architecture monolithique

Toutes les fonctions du noyau sont regroupées dans le même espace d'adressage (i.e. le même processus)

Un exemple : Les versions historiques des systèmes Unix

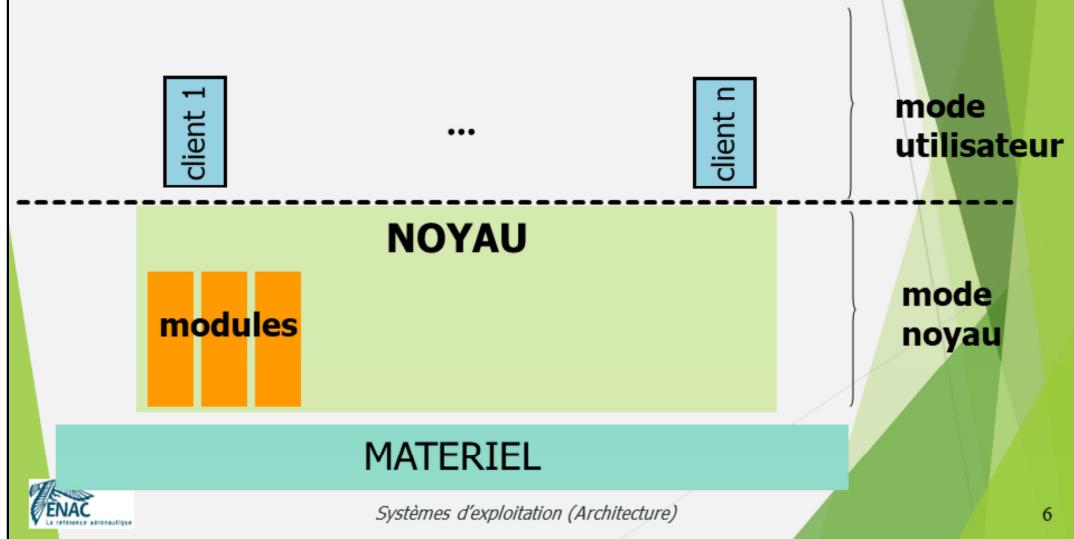
Avantages : Très performant (rapidité d'exécution)

Inconvénients : Difficile à maintenir et à faire évoluer

ARCHITECTURE

ARCHITECTURE MONOLITHIQUE MODULAIRE

Un compromis : les modules dynamiques



Les versions actuelles des systèmes Unix (Linux, ...) permettent de charger des modules (par exemple des pilotes de périphériques) pendant que le système est en cours d'exécution.

Ces modules peuvent aussi être lancés automatiquement au démarrage.

ARCHITECTURE

ARCHITECTURE MULTICOUCHES

Un exemple théorique



Processus utilisateur

- 4 Gestion des fichiers
- 3 Communication inter processus
- 2 Gestion des entrées/sorties
- 1 Gestion de la mémoire
- 0 Gestion des processus

MATERIEL



Systèmes d'exploitation (Architecture)

7

Dans une architecture multicouches, chaque couche s'appuie sur les services de la couche inférieure (on peut par exemple citer le modèle OSI dans le domaine du réseau).

Chaque couche possède son propre espace d'adressage. Il faut donc prévoir un système de communication entre les couches.

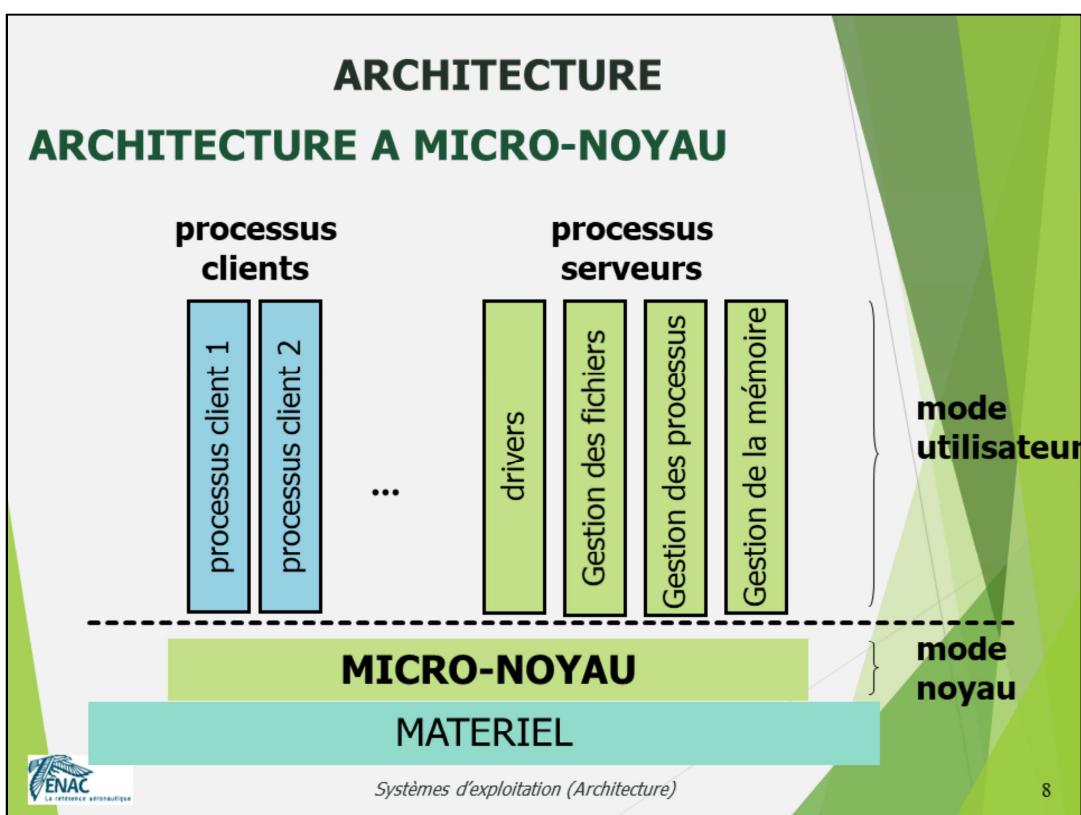
Ce modèle reste théorique dans le domaine des systèmes d'exploitation où un découpage pertinent est difficile à appliquer. Cependant, certains systèmes présentent une architecture partiellement multicouches.

Avantages : sécurité, portabilité, facilité de maintenance dues à la modularité

Inconvénients : moins performant qu'un système monolithique à cause de la communication entre les couches

ARCHITECTURE

ARCHITECTURE A MICRO-NOYAU



Les fonctionnalités de base du système qui requièrent un accès à la mémoire et au processeur ou une grande rapidité d'exécution s'exécutent en mode noyau : serveur de messages (IPC), gestion des interruptions, liaison avec la MMU, allocation du processeur, couche d'accès au matériel.

Le micro-noyau est donc une base de système qui fonctionne comme un serveur. Les autres fonctionnalités du système ont le rôle de client et communiquent via le noyau.

Avantages : fiabilité, modularité, possibilité de faire tourner des fonctionnalités appartenant à plusieurs systèmes en parallèle

Inconvénients : intégration, performance

Plusieurs systèmes d'exploitation intègrent un micronoyau :

certains UNIX

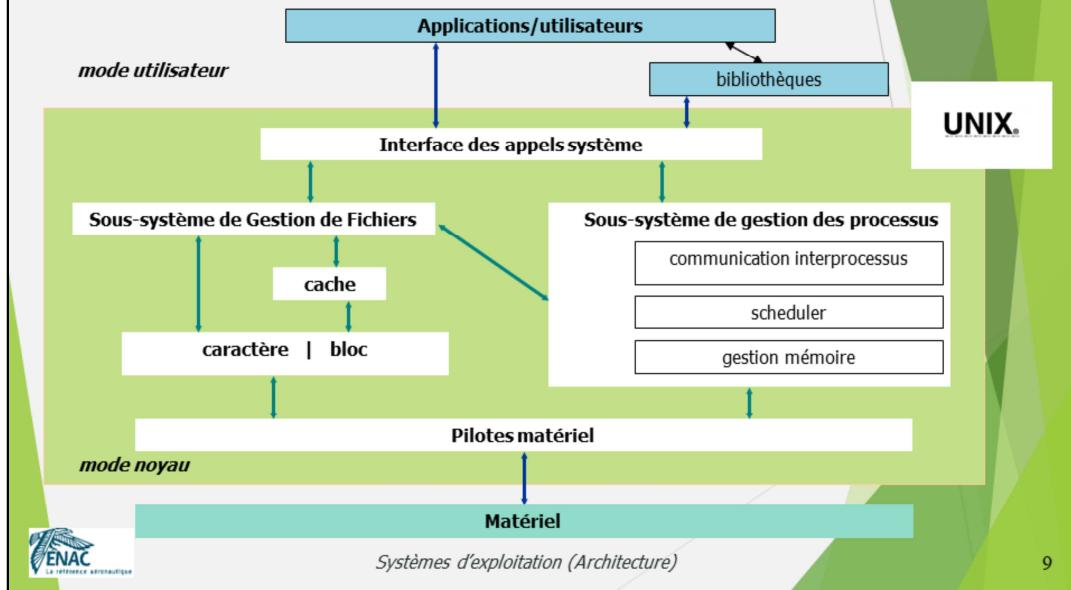
Darwin (OS X, IOS 7) = micro-noyau MACH 3 + services système BSD

QNX

Windows 2000 / XP / 7 / 8 / Phone 8 / 10

ARCHITECTURE

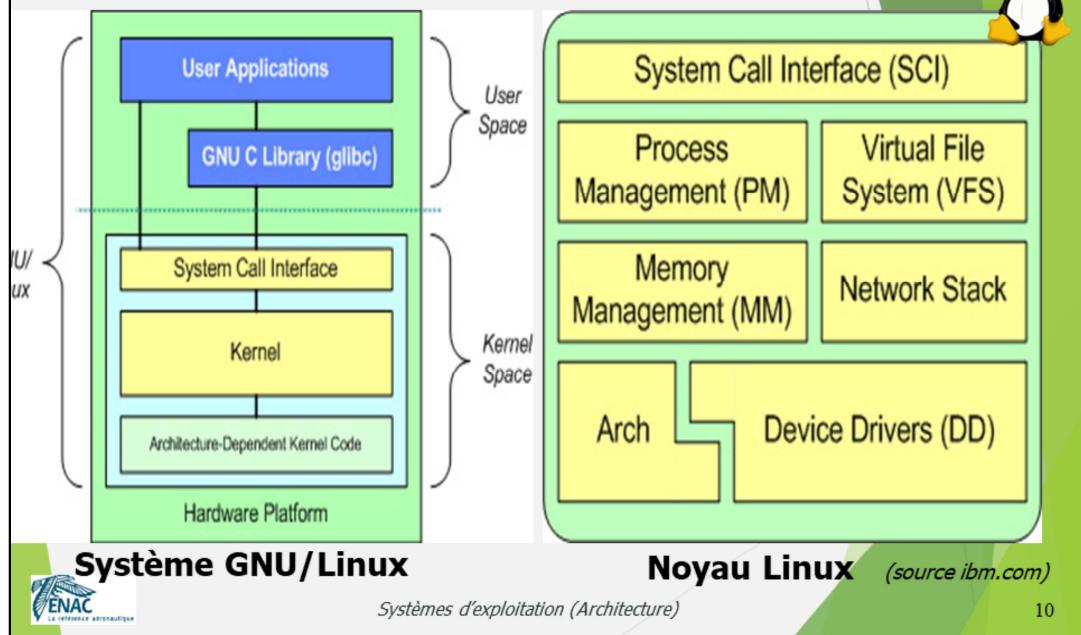
UNIX : architecture monolithique



Les versions historiques d'Unix sont monolithiques. On voit ici que les drivers sont intégrés au noyau, ce qui impose de recréer l'exécutable du système lors de l'ajout d'un nouveau pilote.

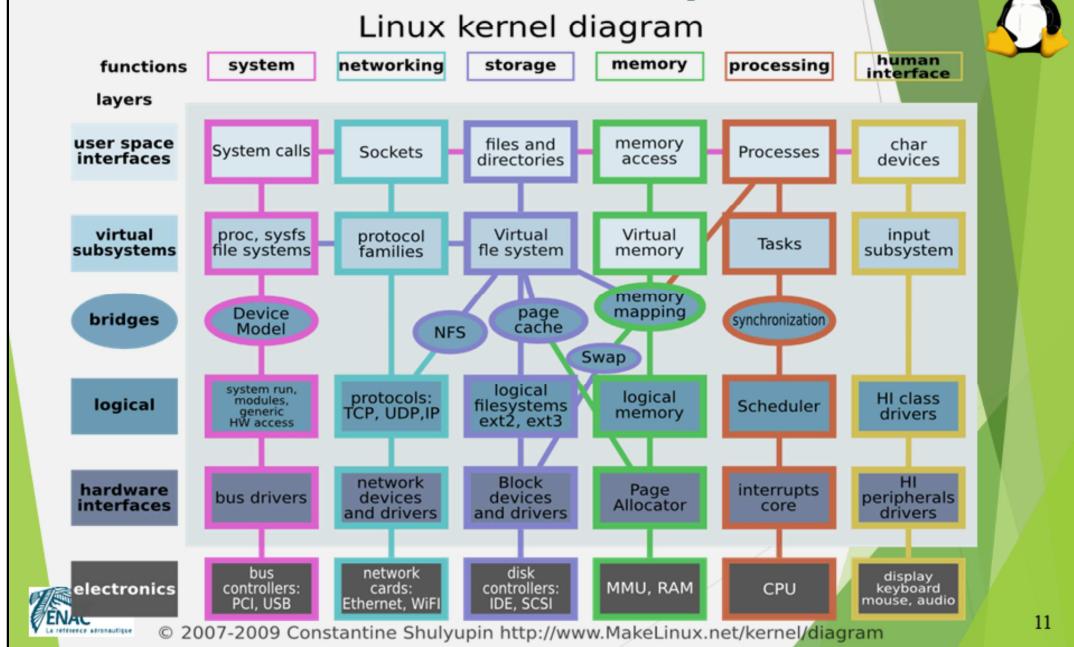
ARCHITECTURE

LINUX : architecture monolithique modulaire



ARCHITECTURE

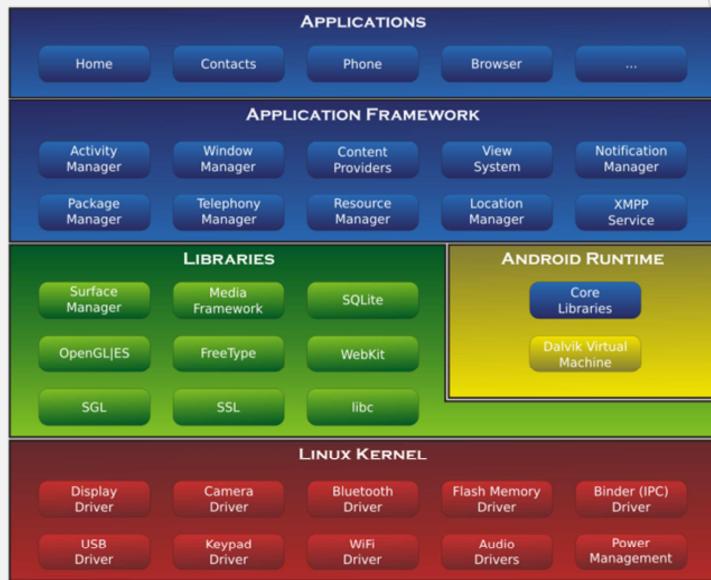
LINUX : architecture monolithique modulaire



11

Interaction des blocs de fonctionnalités à l'intérieur du noyau Linux.

ARCHITECTURE ANDROÏD : sur un noyau Linux



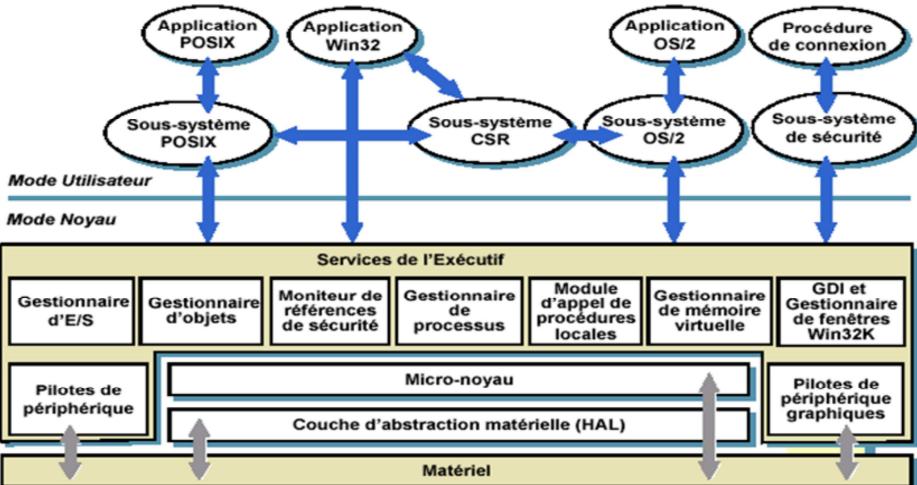
Systèmes d'exploitation (Architecture)

12

Android est basé sur une version de Linux destinée aux systèmes mobiles.

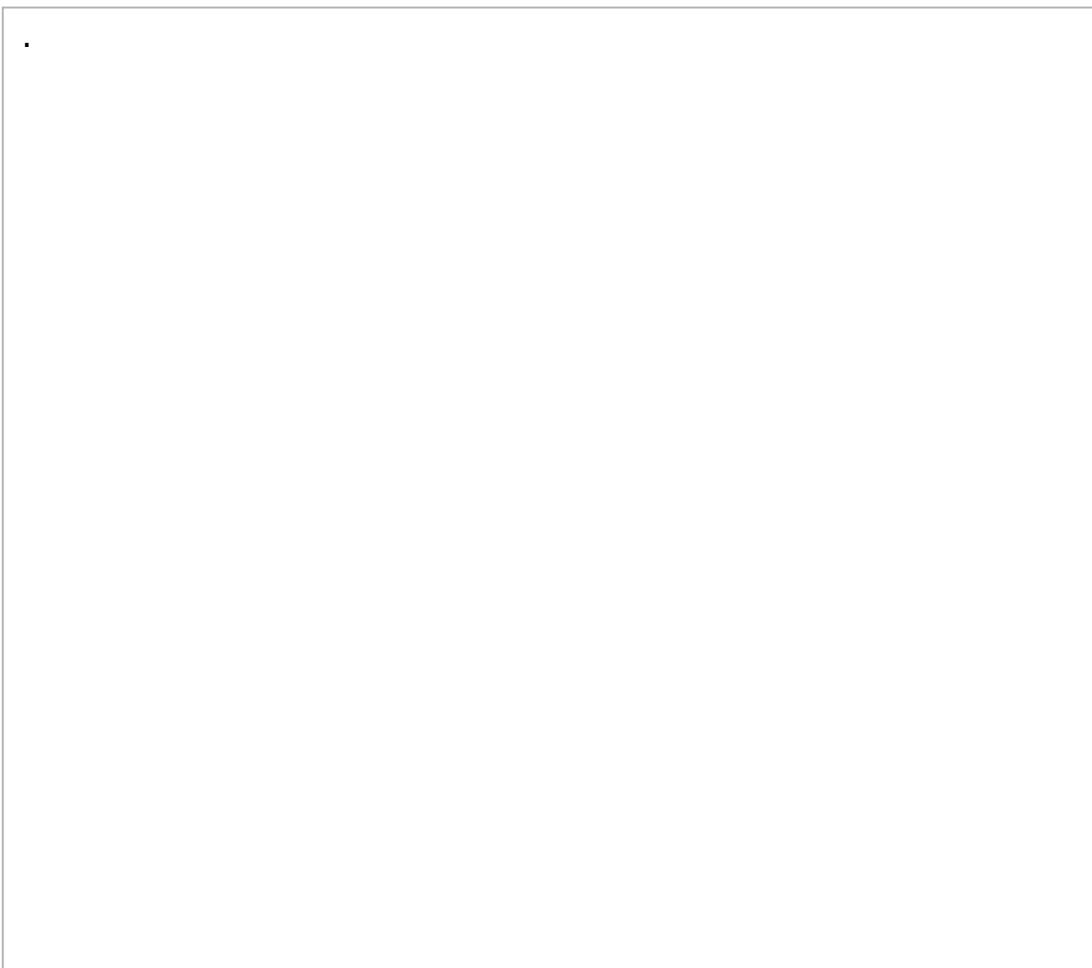
ARCHITECTURE

WINDOWS 2000/.../10 : architecture mixte



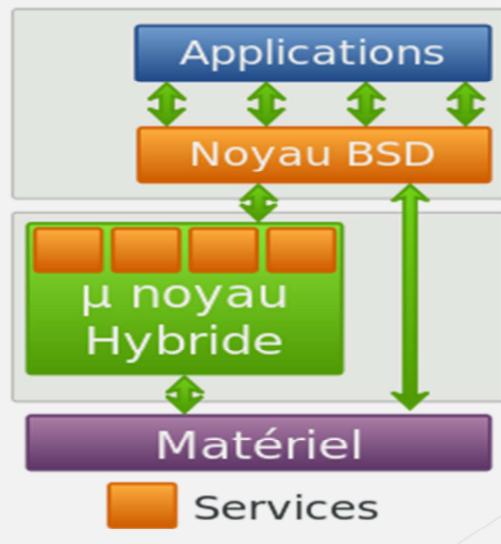
Systèmes d'exploitation (Architecture)

13



ARCHITECTURE

XNU (DARWIN) : architecture mixte



14

