Initiation à l'informatique

Licence de Science et Technologie - Parcours SV

V. Le Brun
Laboratoire d'Astrophysique de Marseille
vincent.lebrun@oamp.fr
04 91 05 69 78

Cours : 6 * 1 heure
TP : 8 * 2 heures

Modalités d'examens :

* 1 Contrôle sur machine en binôme
  - 2 heures — 8

* Examen Janvier
  - 2 heures — 12

20

PLAN

I. Notions de base sur les ordinateurs

II. Systèmes d'exploitation

III. Notions de programmation

I. Notions de base sur les ordinateurs

1. Qu'est ce qu'on peut faire avec?
2. Sur quoi?
3. Comment?
4. Structure des ordinateurs actuels.

I.1. Qu'est ce qu'on peut faire avec un ordinateur ?

Faire réaliser

des traitements effectifs

sur des données discrètes.

I.1. Qu'est ce qu'on

ordin

C'est l'ordinateur qui travaille ...

Faire réaliser

des traitements effectifs

sur des données discrètes.

I.1. Qu'est ce qu'on pe
ordinat

peuvent être décrits par des algorithmes

Faire réaliser

des **traitements effectifs**

sur des données discrètes.

I.1. Qu'est ce qu'on peut faire avec un ordinateur ?

Faire réaliser

des traitements ef

sur des données discrètes.

Finies, que l'on peut énumérer et différencier.

Algorithme

(vient du nom du mathématicien arabe Al Khowarizmi)

Méthode **systématique**
décomposée
en **étapes élémentaires**
et permettant de résoudre
à coup sûr
un **type donné** de problème

Ce qu'on ne peut pas faire avec un algorithme

- Résoudre un problème qu'on ne sait pas **comment** résoudre (Théorème de Fermat, $a^n + b^n = c^n$)

- Réaliser des opérations qui ne peuvent s'écrire de façon systématique (Ecrire une symphonie originale)

Ce qu'on peut faire avec un algorithme

- Déterminer si un nombre est premier :
  - ★ Prendre tous les entiers i entre 2 et n
    - ✴ Effectuer la division de n par i
  - ★ Si au moins un reste est nul, le nombre n'est pas premier

Attention

Le fait que cet algorithme soit correct ne veut pas dire qu'il soit utilisable sur un ordinateur...

Un algorithme c'est donc :

1. Des opérations élémentaires
   a. Division
   b. Calcul de reste
2. Des tests (vérifications)
3. Une structure

Données discrètes

Découpées en morceaux, discontinues

Autres dénominations : numériques, digitales,

par opposition à

Analogiques, continues,

Exemples

1. Texte écrit (combinaisons d'un nombre fini de lettres)

2. Code génétique (combinaison de 4 acides aminées)

3. Données informatiques (combinaison de 0 et 1)

Contre-Exemples

1. Texte lu (un son est un phénomène physique continu (variations de pression))
2. Photographie classique (action chimique sur les atomes d'argent)
3. $\mathfrak{R}$ (non seulement il y a une infinité de réels, mais certains d'entre eux comme π, demandent une infinité de nombres)

Les données informatiques

1. L'unité de base est le bit (**Bi**nary Digi**t**)
2. Peut être stocké dans n'importe quel dispositif physique à 2 états
3. Représenté par 0 et 1

Exemples de dispositifs physiques

1. Lampe allumée ou éteinte (les premiers ordinateurs en utilisait)
2. Aimant orienté N ou S (Disques durs, bandes magnétiques)
3. Un creux ou une bosse sur une surface (CDs, DVDs)
4. Un fil électrique dans lequel circule ou non un courant (circuits imprimés)

Unités utilisées en informatique

1. bit : 2 états possibles

   Rien à faire avec

2. octet = 8 bits : 2^8 = 256 états

   Permet de stocker un alphabet

3. kilo-octet = 2^{10} = 1024 ~ 1000 octets

   Permet de stocker une demi page de texte

Unités utilisées en informatique

1. Mega-octet = 2^{10} = 1024 ko ~ 10^6 octets

   Un livre, une image couleur numérisée

   Un CD = 700 Mo
2. Giga Octet = 2^{10} = 1024 Mo ~ 10^9 octets ($1{,}07\ 10^9$ exactement)

   Petit film

   1 DVD = 4,7 Go, disques durs ~ 250 Go
3. Tera Octet = 2^{10} = 1024 Go ~ 10^{12} octets ($1{,}1\ 10^{12}$ exactement)

Comment stocker des caractères en numérique ?

1. Alphabet = 26 caractères, x 2 pour les majuscules, + ponctuation, parenthèses, symboles spéciaux (&@§! $), lettres accentuées (ù ñ ö Å Ø)

Un peu plus de 100 caractères

Un octet suffit

Le Code ASCII

American Standard Code for Information Interchange

D'abord sur 7 bit (ASCII standard)

Code étendu sur 1 octet : accents,...

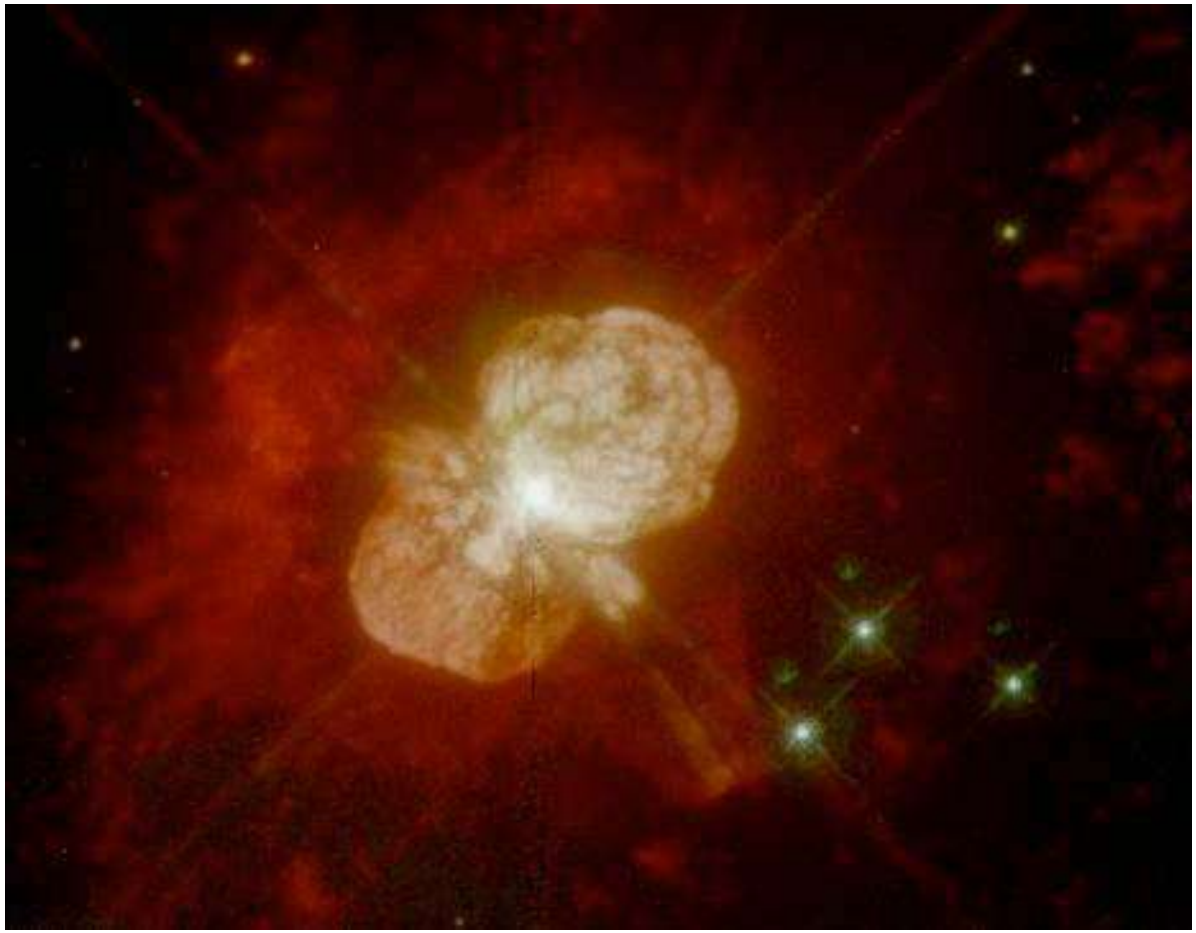
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of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

Source: www.LookupTables.com

Pour lire un fichier ASCII

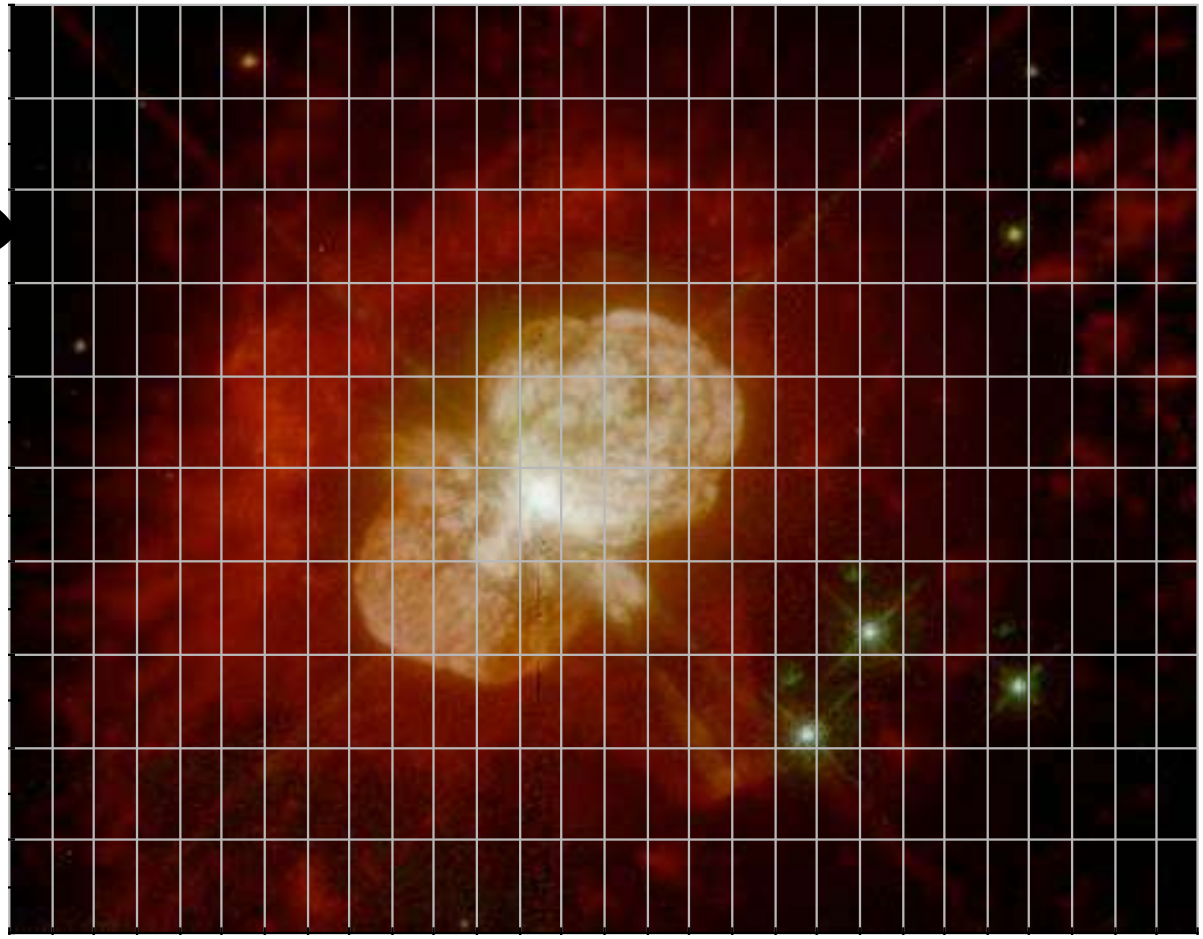
1. Lire les données octet par octet
2. lire dans la table le caractère correspondant à chaque octet
3. Afficher le caractère

Les Images



Les Images

Pixel (Picture Element)

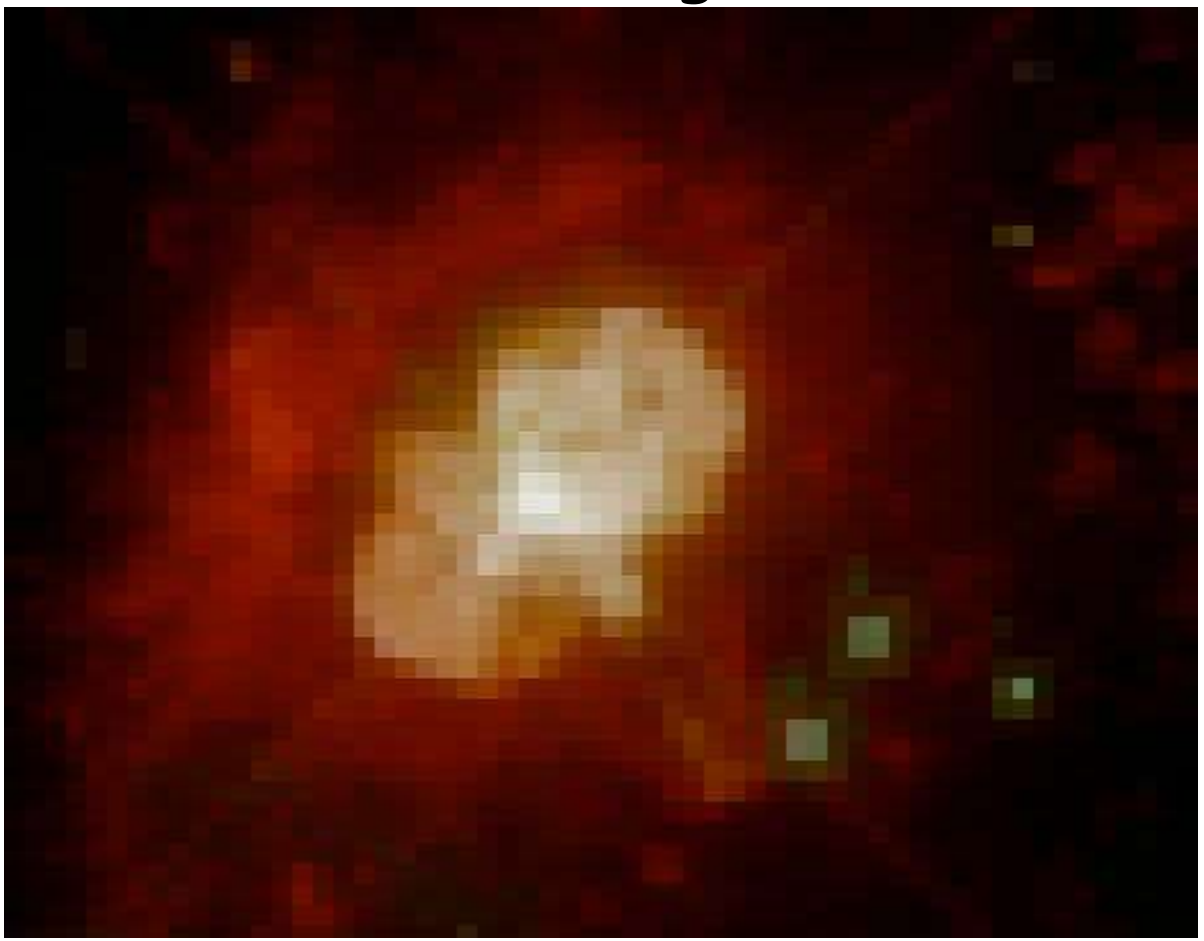


Les Images

Pour chaque Pixel, on calcule l'intensité lumineuse, représenté par un nombre entier entre 0 et 255 (un octet) pour les images N/B, ou 3 entiers entre 0 et 255 pour la couleur (RGB).

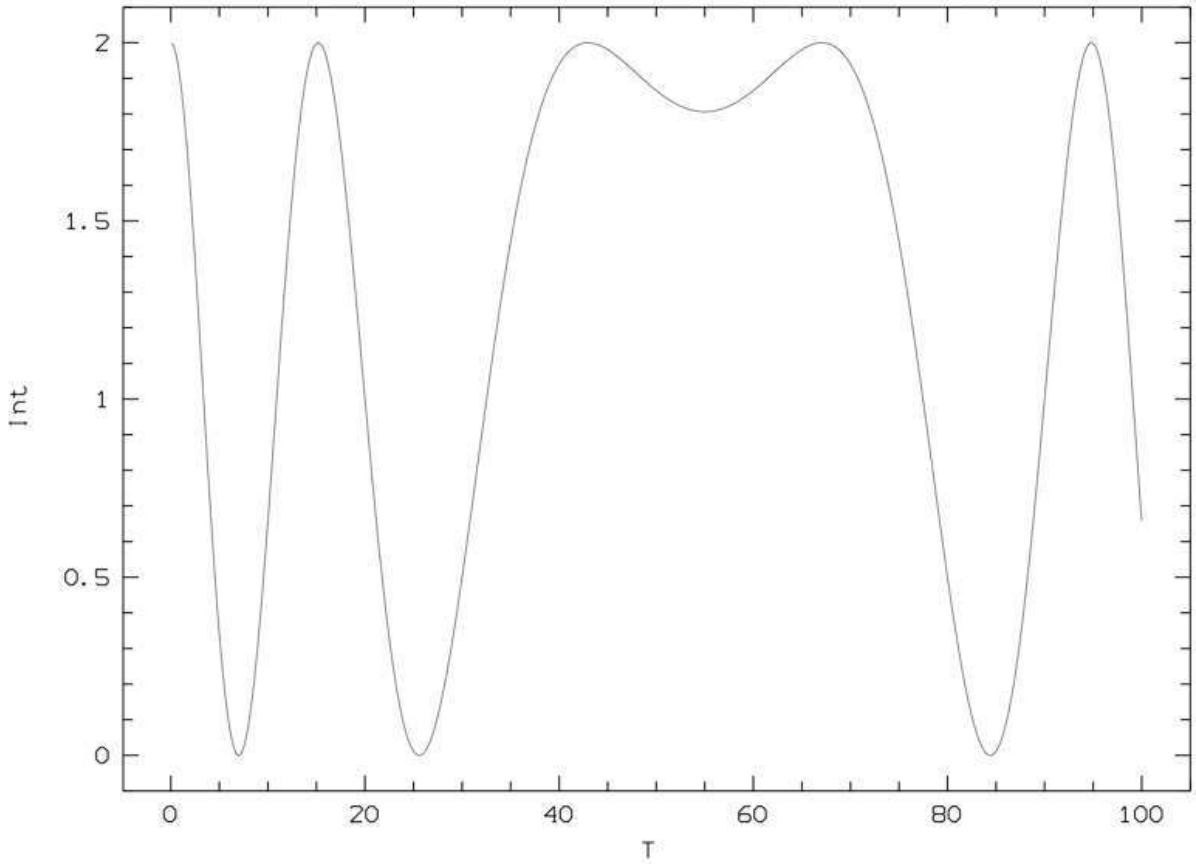
Au passage, $256^3 \sim 16$ millions

Les Images



Le son

A une fréquence donnée :



1. On découpe le signal en intervalle de temps (1/40000 s) et de fréquence
2. On calcule l'intensité sonore dans chaque intervalle
3. On code le nombre selon une norme

Pourquoi 1/40000 s ??

1. l'oreille humaine est sensible jusqu'à environ 20000 Hz, c'est à dire une vibration durant 1/20000 s

2. Pour bien échantillonner un phénomène, il faut au moins le représenter par 2 valeurs

3. C'est minimal et ça ne suffit pas toujours...

Les Nombres

Les entiers : codage en binaire (base 2)

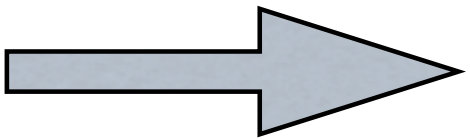
Les entiers relatifs : plus compliqué, mais peuvent être utilisés directement

Les réels : codage complexe, plus nécessité de programmer les opérations

Structure des ordinateurs

Travaux théoriques de Turing (1936) :

Existe-t-il un dispositif capable de prouver si une proposition logique quelconque est vrai ou fausse ?

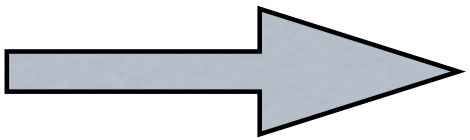


Machine de Turing, dédiée à un problème donné

Structure des ordinateurs

Machine Universelle de Turing :

A partir de la description d'une certaine machine M, et de données D, est capable de simuler l'action de M sur D



Un ordinateur est une Machine Universelle de Turing

Structure des ordinateurs

Une machine de Turing est un modèle théorique, donc sans limitation (de temps ou de capacité), au contraire d'un dispositif physique ...

Structure des ordinateurs

Travaux (Pratiques) de Von Neuman :

description de l'architecture de base d'une Machine Universelle

Cette architecture est toujours utilisée en 2005...

Structure des ordinateurs

Envoie un signal électrique qui marque le début d'un cycle d'instruction

De sa vitesse dépend la capacité de traitement de l'ordinateur (MAIS PAS SEULEMENT)

Cadence actuelle : ~ 4 Ghz

Structure des ordinateurs

Evolution des cadences :

1980 : 5 - 8 Mhz
1990 : 50 MHz
1995 : 200 MHz
2000 : 1 GHz
2005 : 4 GHz

Structure des ordinateurs

Random Acces Memory

Tableau à deux colonnes de nombres binaires :
1 - Adresses : fixes, codées sur N bits, ce qui donne 2^N adresses possibles: N conditionne la taille maximale de la RAM
N=4 ⇒ 16 cases

RAM

0000	00101000
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Structure des ordinateurs

Random Acces Memory

Tableau à deux colonnes de nombres binaires :
2 - Mots mémoires, codés sur un nombre fixe de bits (8 ici, 32 ou 64 sur les ordinateurs)
Ces mots peuvent aussi bien décrire des données que des instructions (principe de la MU)

RAM

0000	00101000
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Structure des ordinateurs

Random Acces Memory

512 Mo = $2^9 * 2^{10} * 2^{10}$ Octets

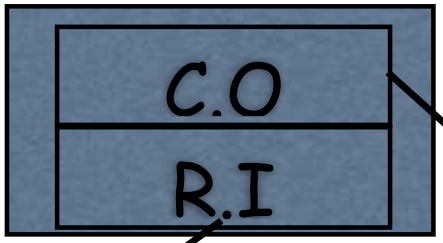
permet donc de stocker :
2^{27} mots de 4 octets ~ 134 10^6 mots
adresses codées sur 27 bits

RAM

0000	00101000
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Structure des ordinateurs

Unité de Commande



Compteur Ordinal :
Registre (petite unité de mémoire vive) contenant l'adresse dans la RAM de l'instruction en cours d'exécution (taille d'une adresse RAM)

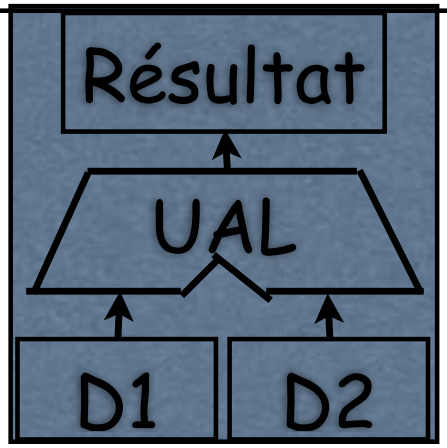
Registre d'Instruction :
Contient l'instruction en cours d'exécution (taille d'un mot mémoire).

Structure des ordinateurs

UAL : Unité arithmétique et logique : ensemble de circuits imprimés réalisant des opérations élémentaires (+, -, ...)

Résultat : Registre contenant le résultat de l'instruction

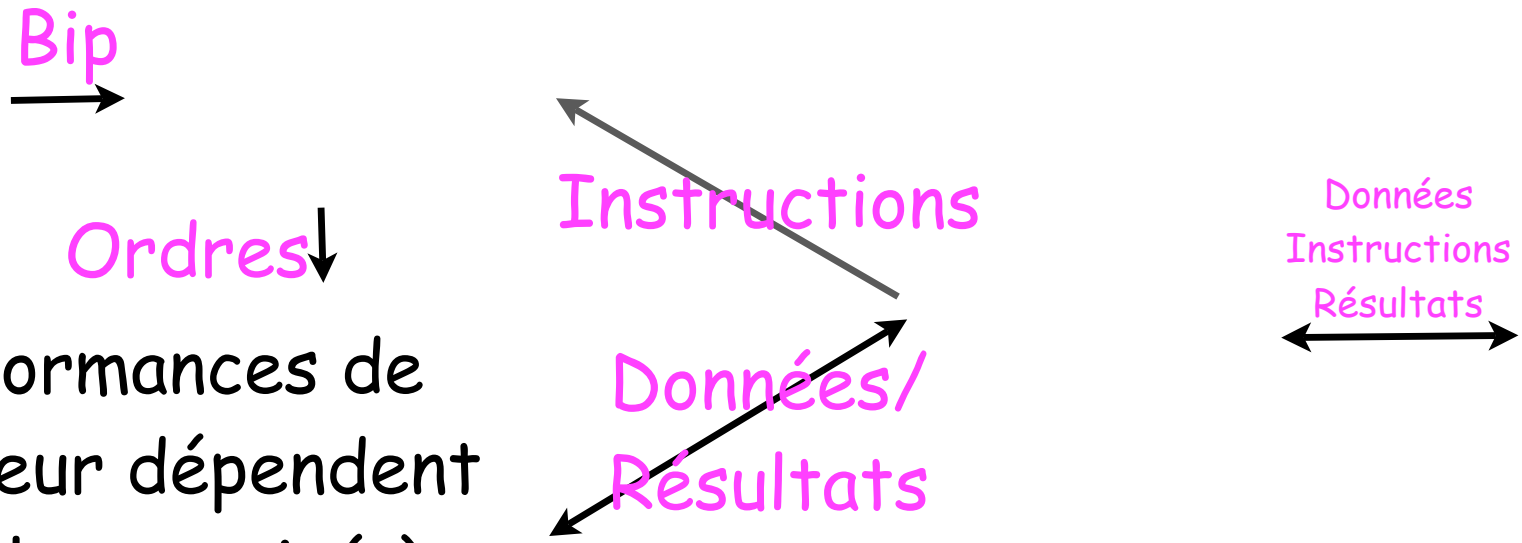
Unité de Traitement



D1, D2 : Registres contenant les valeurs utilisées par l'instruction en cours

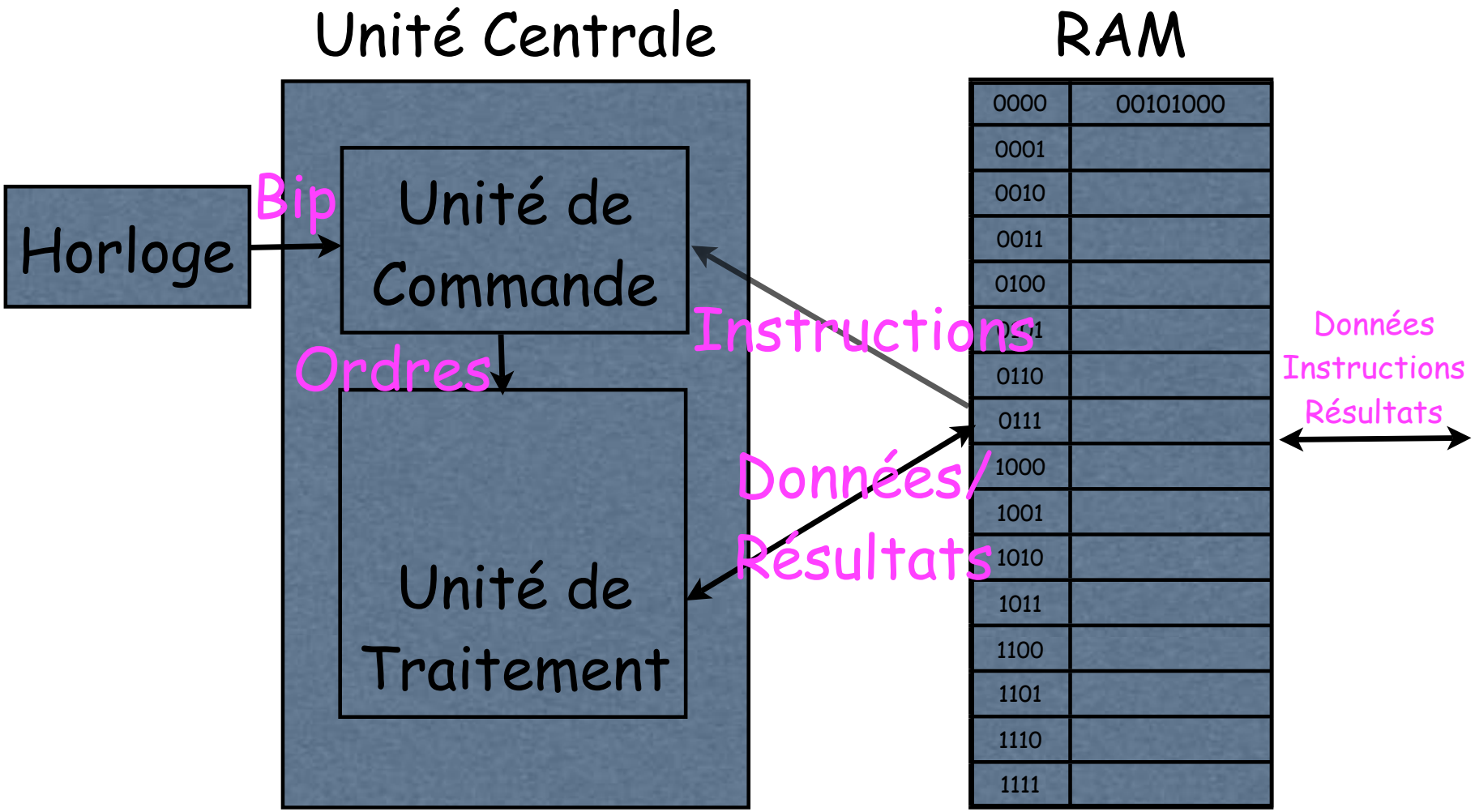
Structure des ordinateurs

Bus : Ensemble de fils pouvant transporter plusieurs bits en parallèle



Les performances de l'ordinateur dépendent aussi de la capacité à réaliser des circuits avec les bus les plus courts

Structure des ordinateurs



Structure des ordinateurs : Déroulement d'un cycle

1 -L' adresse de l'instruction à exécuter est chargée dans le C.O

2 -Le code de l'instruction est chargée dans le R.I

3 -Il est ensuite passé dans l'UAL, qui identifie les adresses des éventuelles données nécessaires, et les charge depuis la RAM

Structure des ordinateurs : Déroulement d'un cycle

4 - L'UAL exécute l'instruction

5 - Le résultat est chargé à l'adresse adéquate de la RAM

6 - Retour à l'étape 1

Structure des ordinateurs :

Les instructions passées au processeur forment le langage machine, ou assembleur.
Chaque opération est codée par un nombre binaire (0001 pour addition, 0010 pour comparaison, ...).
Le langage machine est propre à une type de processeur, en général entre 50 et 300 commandes

Comment entrer les données et récupérer les résultats ?

1. Programmation directement en langage machine, création d'une interface avec les bus externes de la RAM...

2. Utiliser une machine dotée d'un système d'exploitation

Système d'exploitation

Fait le lien entre utilisateur et processeur

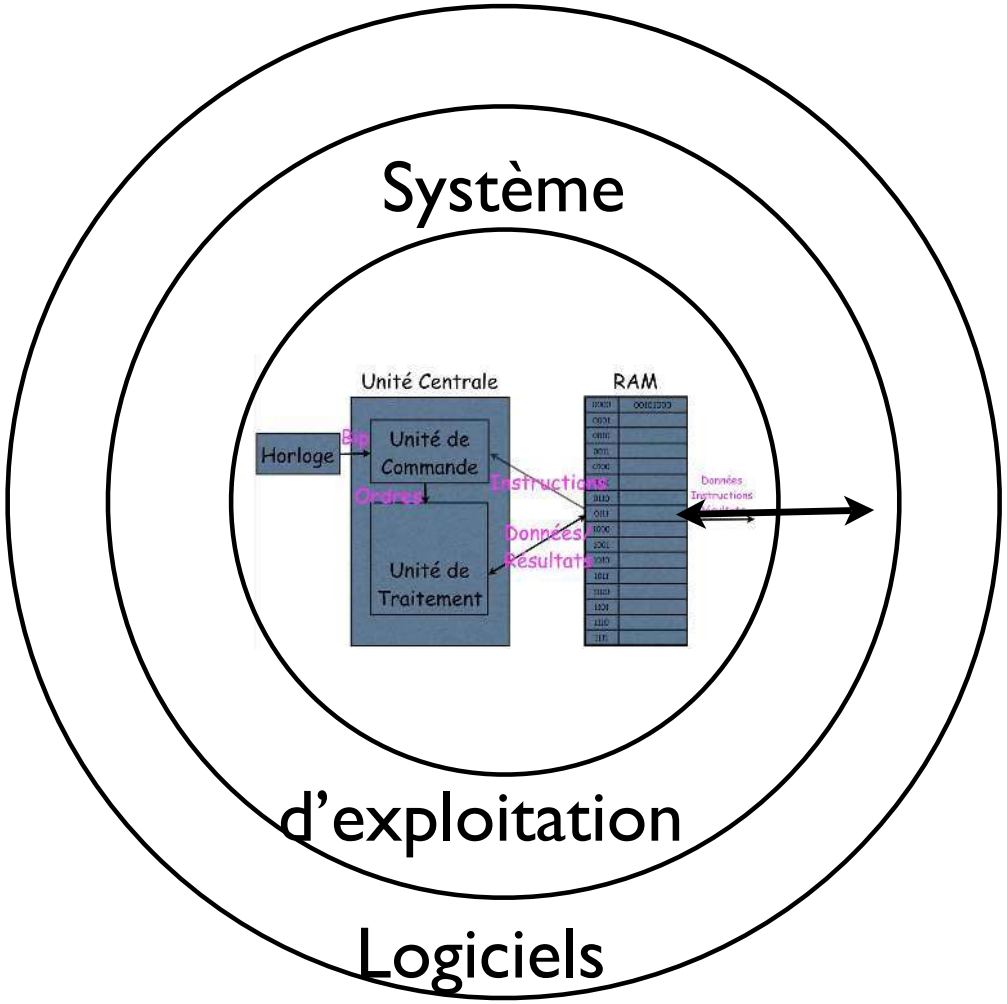
Il se caractérise par des commandes utilisateur qui lui sont propres

un OS peut être développé pour n'importe quelle machine, si il possède les interfaces adéquates avec le CPU

(Unix existe pour PCs, MACs, ...)

"L'utilisateur" désigne aussi bien l'utilisateur humain qu'un logiciel, qui envoie des commandes à l'OS : un logiciel "UNIX" fonctionnera sur toute machine ou UNIX est installé.

Structure 'en couche'



Il manque toujours des outils
d'interface avec l'utilisateur humain :

Rôle des périphériques

Les périphériques

Leur rôle est de permettre d'entrer et de lire les données et instructions vers ou depuis le CPU, via l'OS

Ils sont gérés par l'OS

Types de Périphériques

Entrée : cartes perforées, clavier, souris,scanner, lecteur DVD

Sortie : Ecran, imprimante, graveur

Entrée/Sortie : disque dur, modem, ...

Les grandes familles d'OS

DOS/Windows : machines de type PC, applications bureautique. Code propriétaire, interfaces imposées avec l'OS

Très large diffusion.

Les grandes familles d'OS

MacOS : uniquement sur machines Apple. Diffusion plus restreinte. Base Unix, facilité accrue de développement

Unix : adaptable sur toutes machines, accès aux programmes sources, grande facilité de développement

Les systèmes UNIX

Crée dans les années 70 aux laboratoires BELL pour permettre l'utilisation **simultanée** des ordinateurs (pour éviter les exécutions séquentielles)

Première utilisation à Berkeley

Les systèmes UNIX

Développement en parallèle du langage C, qui est à la fois de syntaxe facile, et proche du matériel pour permettre une gestion directe du CPU et des périphériques.

UNIX

La première machine sous UNIX avait 24kO de RAM, dont 16 pour le système (24 kO = 300 lignes de programme), disque dur de 512 kO, fichiers de 64 kO maximum (16000 nombres)

UNIX

Premier système multi-utilisateur et multi-tâches :

Gestion spécifique des fichiers et des instructions

Unix : gestion multi-utilisateurs

Chaque utilisateur est identifié lorsqu'il se connecte au système par un identifiant (login) et un mot de passe (password)

L'OS positionne alors l'utilisateur sur la partie de l'arborescence de fichiers qui lui appartient.

UNIX - Les processus

Process en anglais : c'est un programme en cours d'exécution.

L'OS affecte un quantum de temps à un process, puis à un autre, etc, etc...

Permet l'exécution simultanée et efficace de plusieurs programmes

Unix : gestion multi-utilisateurs

Chaque processus possède un numéro unique.

Les processus ont un propriétaire, un niveau de priorité

Un utilisateur ne peut pas interférer avec les processus des autres

Notion de fichier

Un fichier est un ensemble de données binaires, qui, en plus de son contenu, possède aussi une adresse dans l'arborescence de fichiers.

Les répertoires/dossiers/directories sont des fichiers spéciaux qui contiennent d'autres fichiers

Les fichiers

On distingue les 'fichiers ASCII', qui contiennent du texte codé selon la norme ASCII, et les fichiers binaires/ exécutables, qui contiennent soit des données codées selon une autre norme, soit des fichiers pouvant être transmis directement au CPU pour exécution

Unix : les fichiers

Chaque fichier possède un nom :

c'est une chaîne alphanumérique quelconque (2048 caractères pour l'adresse totale), majuscules et minuscules sont différentes.

Unix : les fichiers

Chaque fichier possède un nom :

toto.txt

TOTO.txt

toto.txt.gz

Toto_1-R@é*.f.totototo.jpeg

Unix : les fichiers

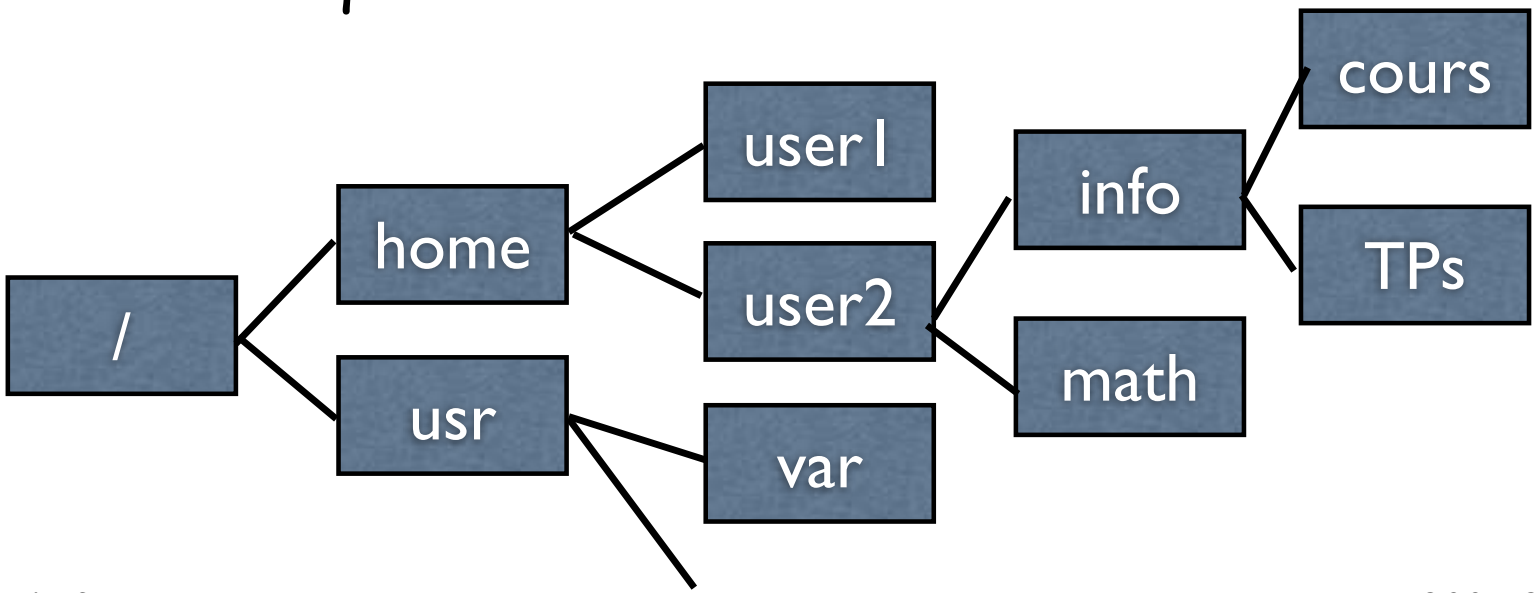
Chaque fichier possède :

un propriétaire (un des utilisateurs identifiés dans le système)

une adresse dans le système de fichiers

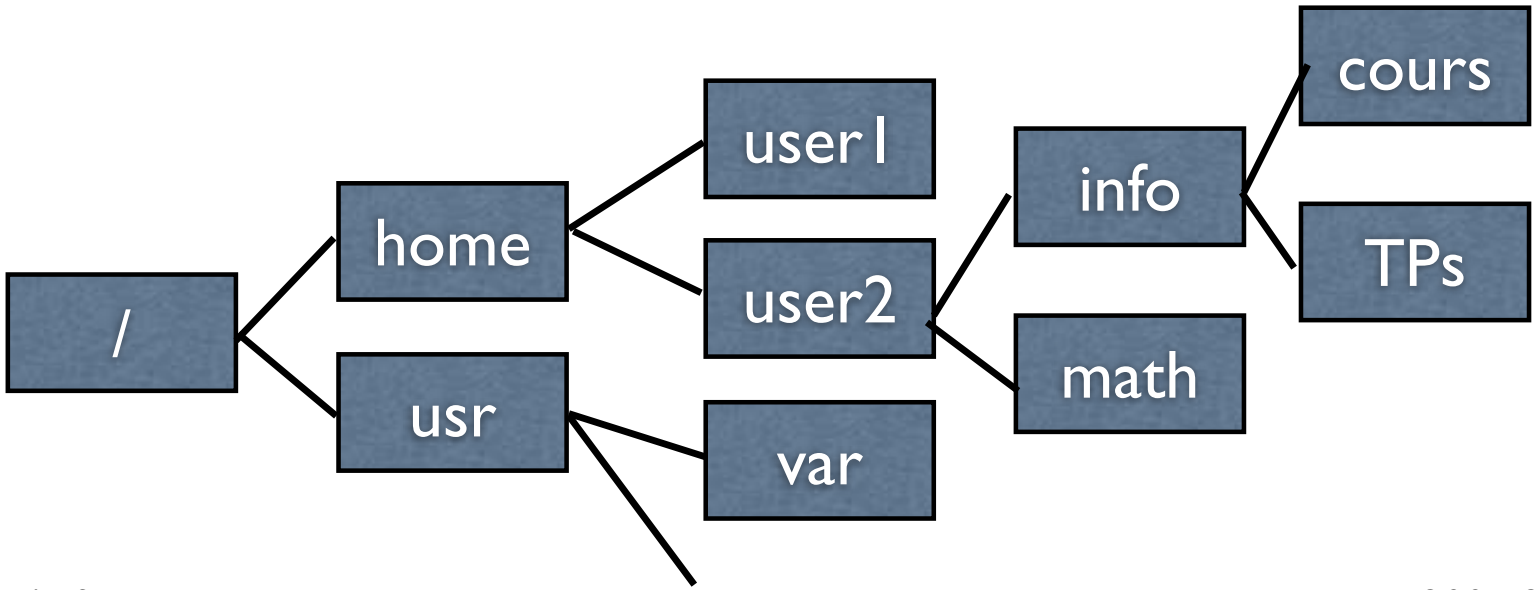
Arborescence

Les répertoires sont imbriquées les uns dans les autres à partir du répertoire 'racine' qui les contient tous



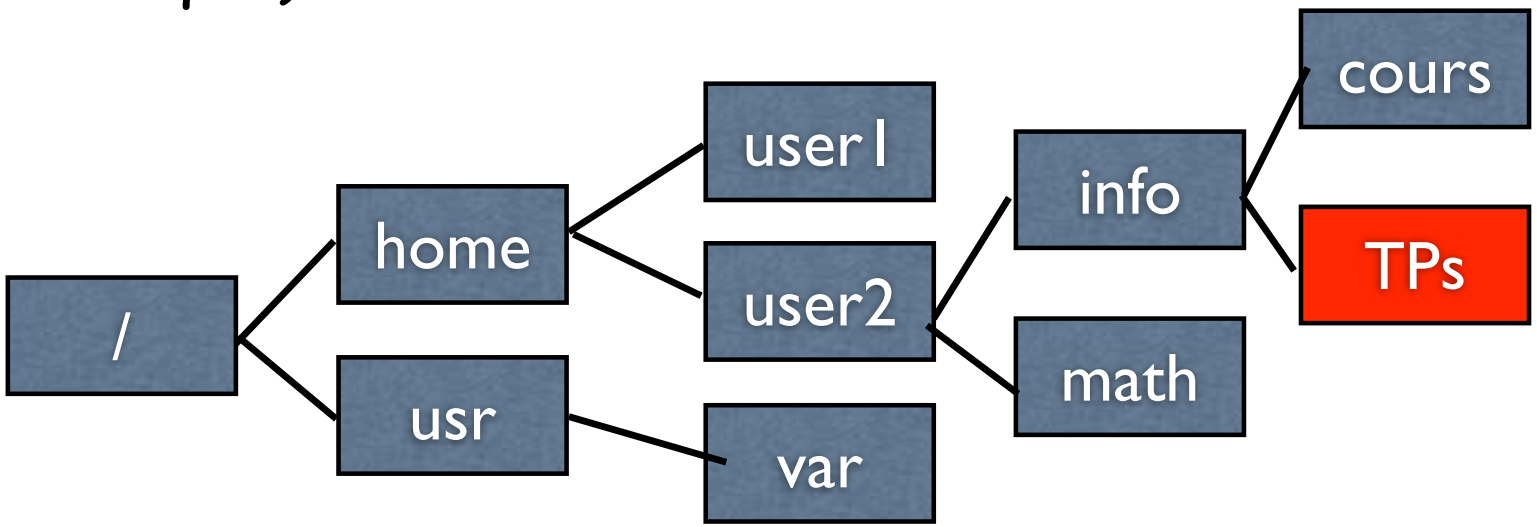
Arborescence

le symbole / (slash) sert à séparer les différents niveaux de l'arborescence



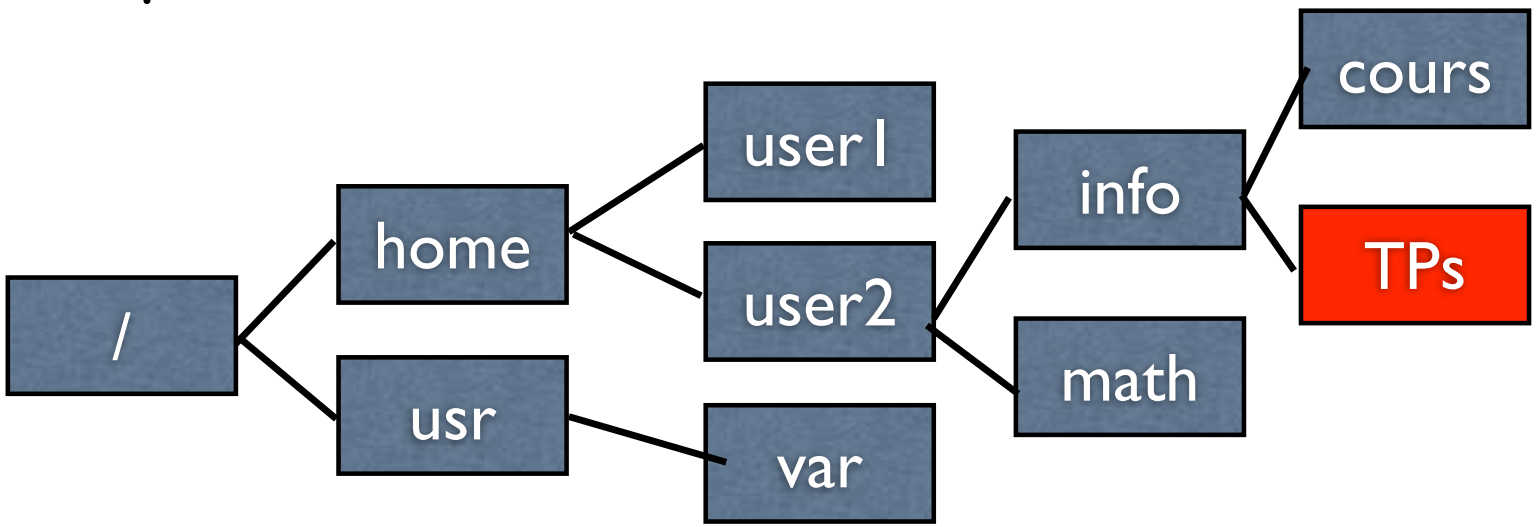
Arborescence

/home/user2/info/TPs adresse absolue de ce répertoire (commence par /, unique)



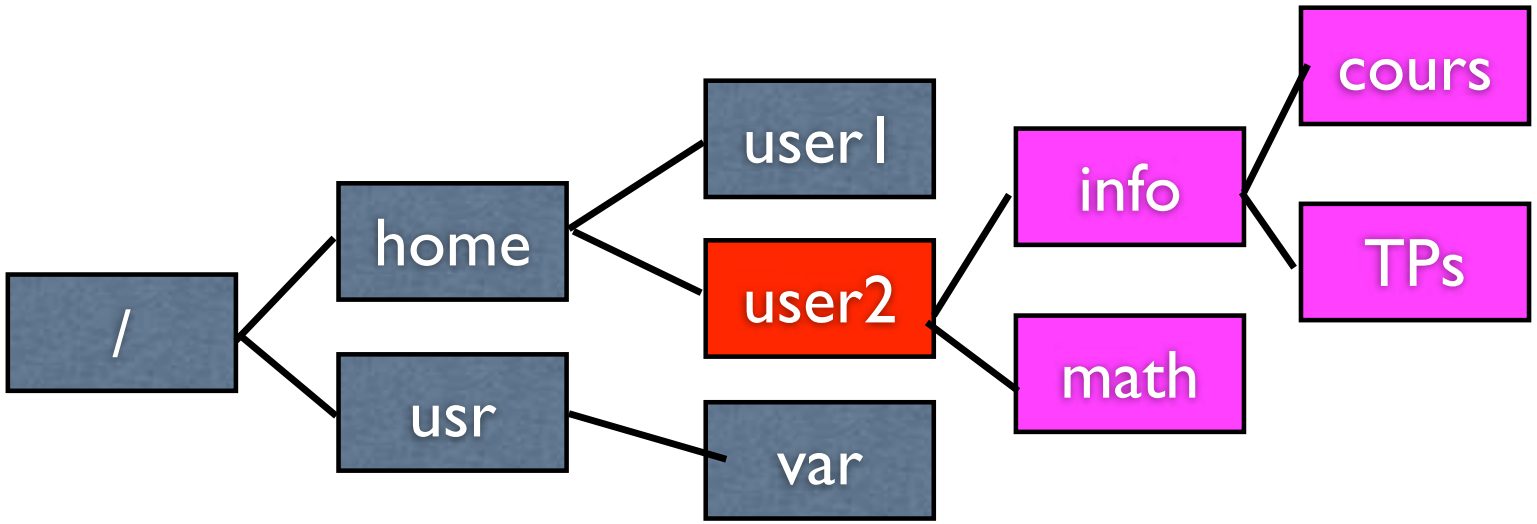
Arborescence

le répertoire depuis lequel on travaille s'appelle le répertoire de travail, ou répertoire courant



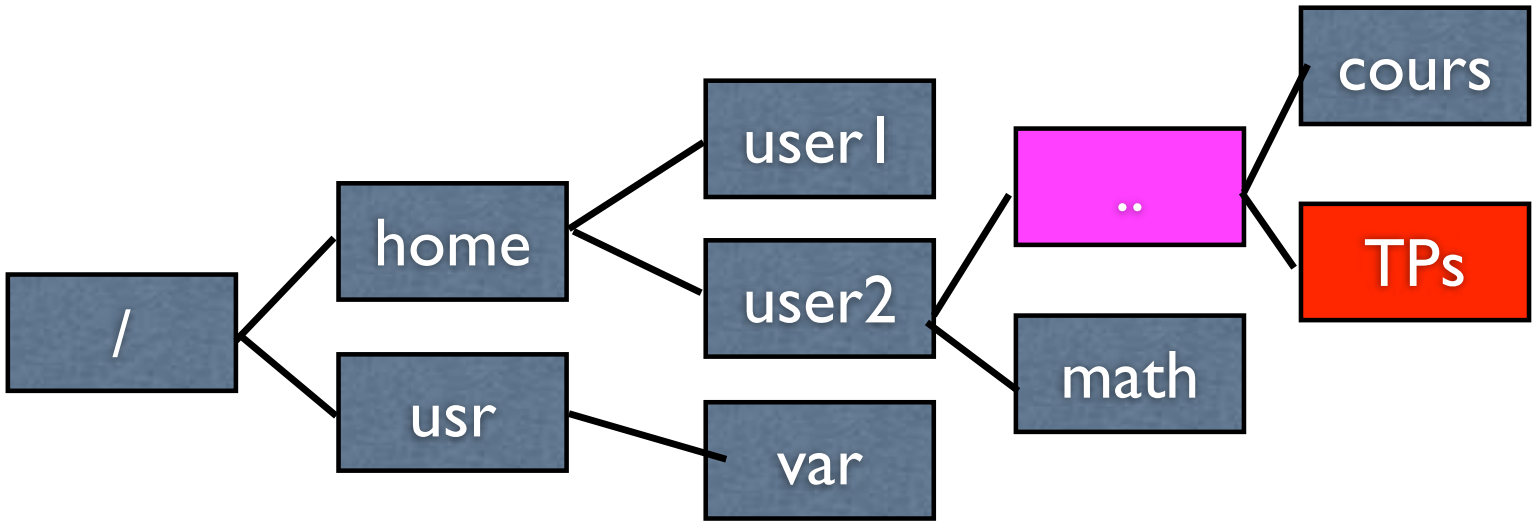
Arborescence

Répertoire Principal de l'utilisateur user2. Tous les fichiers inclus (répertoires fils ou sous-répertoires) dedans lui appartiennent par défaut.



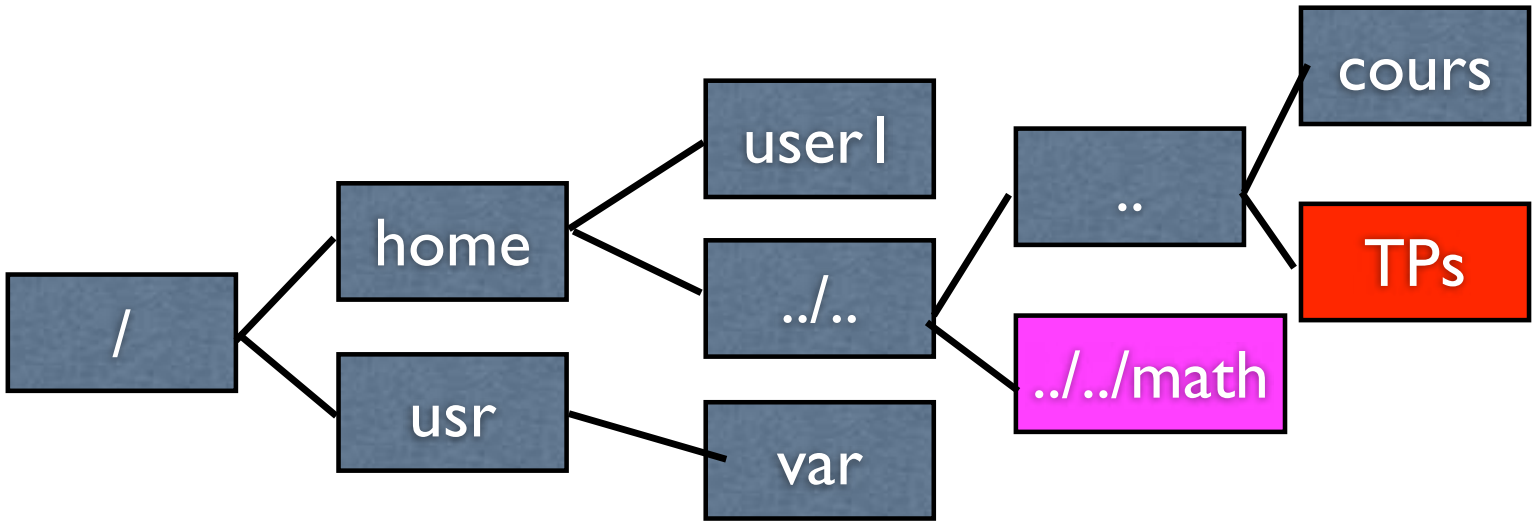
Arborescence

Par rapport à un répertoire donné ..
désigne le répertoire immédiatement supérieur (répertoire parent) dans l'arborescence



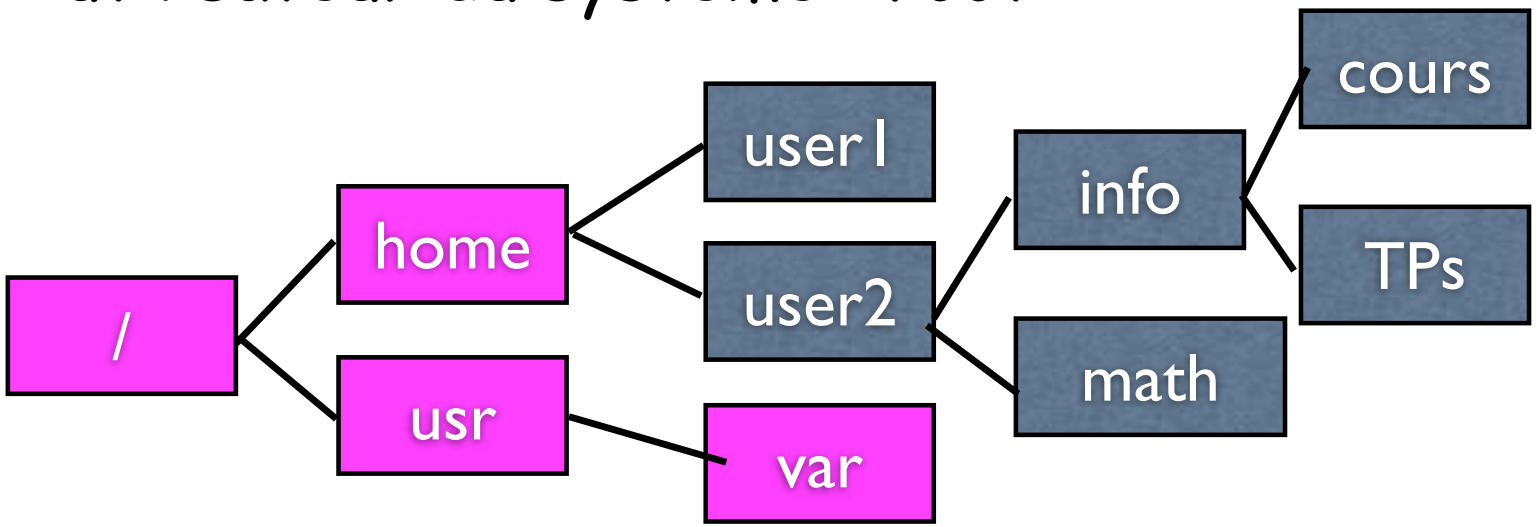
Arborescence

Par rapport à un répertoire donné, le chemin relatif désigne le trajet pour s'y rendre : ../../math



Arborescence

Les répertoires hors répertoires principaux appartiennent au super-utilisateur du système : root



Les droits des fichiers UNIX

Par rapport à un fichier donné, il existe trois catégories d'utilisateurs :

1. u(ser) : son propriétaire
2. g(roup) : le groupe de son propriétaire (collaborateurs proches)
3. o(thers) : les autres utilisateurs

Les droits sur les fichiers UNIX

Pour chaque catégorie d'utilisateurs, il existe 3 types de droits :

1. r(ead) : droit en lecture
2. w(rite) : droit en écriture
3. (e)x(ecute) : droit en exécution

Les droits sur les fichiers UNIX

Le droit en lecture donne la possibilité de lire le contenu d'un fichier, et donc de le copier

Pour un répertoire, cela correspond à pouvoir obtenir la liste des fichiers qu'il contient

Les droits sur les fichiers UNIX

Le droit en écriture donne le droit de modifier le contenu du fichier (et donc de le détruire).

Pour un répertoire, cela permet d'ajouter un fichier (pour enlever un fichier, il faut aussi le droit d'écriture sur le fichier)

Les droits sur les fichiers UNIX

Le droit en exécution autorise, pour un programme (codé en binaire ou non), la réalisation de ce programme.

Pour un répertoire, cela autorise à traverser ce répertoire pour se positionner dans un de ses sous-répertoires.

Commandes de base

Les noms des commandes sont construits à partir des noms anglais de l'action correspondante, en ne gardant que les consonnes significatives

Commandes de base

Une commande est suivi d'options (qui modifient son mode de fonctionnement) et des arguments (en général des fichiers) auxquels elle s'applique.

Commandes de base

list : ls

remove : rm

change directory : cd

print working directory : pwd

Les différents modes d'exécution

mode synchrone : on tape le nom de la commande, elle s'exécute, on récupère la main

mode asynchrone : on tape le nom de la commande suivi du symbole & : On récupère la main immédiatement, mais le déroulement de la commande se poursuit

Les différents modes d'exécution

le mode asynchrone est utile pour lancer des commandes longues sans bloquer la fenêtre, ou pour lancer d'autres fenêtres (éditeur, logiciel, ...) tout en gardant la main...

Les différents modes d'exécution

mode batch : at date commande : permet de lancer une commande à une date précise.

mode cron (d'après le nom de son inventeur) : permet de répéter une commande à intervalles réguliers

Les alias

Un alias est un moyen de créer une nouvelle commande ou de modifier le fonctionnement d'une commande

Par défaut la commande rm ne demande pas de confirmation avant de détruire un fichier.

Les alias

alias rm 'rm -i'

change le mode de fonctionnement de rm, qui demandera systématiquement confirmation par la suite (option -i)

UNIX - Les 'Wild Cards',

Ce sont des caractères spéciaux qui peuvent remplacer un ou plusieurs caractères, par exemple pour pouvoir gérer plusieurs fichiers ayant des noms proches.

Même si ce n'est pas interdit, il est déconseillé de les utiliser dans les noms de fichiers

UNIX - Les 'Wild Cards',

* remplace n'importe quelle chaîne de caractères, y compris la chaîne vide.

? remplace un caractère et un seul

[a-j] remplace tous les caractères compris entre a et j inclus

UNIX - Les 'Wild Cards',

Il est possible de combiner plusieurs de ces symboles pour fabriquer des recherches complexes

ls toto?.* affichera les fichiers toto1.txt, toto2.for, ...

mais pas toto.txt ni toto22.c

UNIX - Les variables d'environnement

Elles servent à configurer le système selon les besoins de l'utilisateur.

Elles ont un nom, écrit en majuscule (convention) : HOME, PRINTER, PATH, PWD, ... et une valeur.

Une vingtaine au total.

UNIX - Les variables d'environnement

On peut lire cette valeur avec la commande echo

echo chaîne de caractères

va répéter cette chaîne. Utile dans un programme pour afficher un résultat ou donner une instruction.

UNIX - Les variables d'environnement

On peut lire cette valeur avec la commande echo

echo $NOM_DE_VARIABLE

va afficher la valeur de la variable.

UNIX - Les variables d'environnement

HOME contient l'adresse du répertoire principal.

PRINTER contient le nom de l'imprimante à utiliser par défaut.

PATH contient...

UNIX - La variable PATH

Toute commande UNIX correspond à un programme qui doit être exécuté.
Question : comment le système peut-il savoir ou se trouve le programme en question ?

Question subsidiaire : un utilisateur crée une commande : Comment la faire exécuter comme une autre commande ?

UNIX - La variable PATH

La variable PATH contient une liste de répertoires de l'arborescence, dans lesquels le système cherche le programme correspondant à une commande.

/bin/ls est le programme qui est exécuté lorsque l'on tape ls

UNIX - La variable PATH

Le répertoire /bin/ est dans la liste de PATH.

Pour faire exécuter un programme qui n'est pas dans un de ces répertoires il faut soit donner son adresse complète, soit modifier le contenu de PATH

UNIX - La variable PATH

Pour savoir où se trouve le programme qui est exécuté :

which ls

/bin/ls

UNIX - Pipelines

Il est possible sous UNIX de faire passer le résultat d'une commande à une autre commande sans stocker le résultat intermédiaire dans un fichier.

Exemple : dans la liste de mes fichiers, je ne veux que ceux du mois de Septembre

UNIX - Les Pipeline

La commande `ls -l` donne les fichiers avec leur date de modification

La commande `grep` (get regular expression) permet de sélectionner une chaîne de caractère dans un fichier

UNIX - La commande grep

Si on sait pas : commande man (manual),
donne l'aide pour la commande choisie :

man grep

Syntaxe : grep -e expression fichier

UNIX - Les Pipeline

le symbole ` | ` permet de connecter les deux commandes :

ls -l | grep -e Sep

UNIX - Les redirections

Un système UNIX est configuré pour avoir une 'sortie standard'

Par défaut, le résultat d'une commande est toujours dirigée sur cette sortie

UNIX - Les redirections

Sur tous les systèmes actuels cette sortie est l'écran.

Il est cependant possible de rediriger les résultats vers un fichier ou un autre périphérique (imprimante, ...)

UNIX - Les redirections

Le symbole de redirection est le `>`

ls > liste.txt

enregistrera le résultat dans le fichier liste .txt

UNIX - Les redirections

Il est également possible de lire les paramètres d'un programme dans un fichier, avec le symbole `< `

Notions de programmation

Un programme est un ensemble d'instructions stockées dans un fichier et écrites dans un langage donné, langage qui traduit ses instructions en langage machine pour le processeur

Notions de programmation

Il existe 3 familles de langages programmation :

compilés, semi-interprétés et interprétés

Notions de programmation
Les langages compilés

Un programme (source) écrit en langage compilé est tout d'abord transformé en un fichier exécutable (donc écrit en langage machine), qui est celui qui sera exécuté. Cette transformation est effectuée par un programme spécial appelé compilateur

Notions de programmation

Inconvénient : à chaque modification du programme, il faut le compiler à nouveau pour obtenir l'exécutable correspondant

Avantage : on est sûr que la structure du programme est correcte avant l'exécution, qui est très rapide

Notions de programmation

Principaux langages compilés :

C, Fortran,

(calcul scientifique)

Notions de programmations

Un langage interprété utilise directement le source, lors de l'exécution chaque commande est transformée en langage machine avant d'être effectuée ⇒exécution plus lente

Notions de programmations

De plus, une erreur de programmation n'est détectée que lors de son exécution, donc après une certaine durée d'exécution.

Exemples : shell (langage de programmation UNIX), basic, HTML

Notions de programmations

Langage semi-interprété : même principe qu'un langage interprété, mais la syntaxe est vérifié avant exécution afin de garantir que le programme va a son terme (ce qui ne garantit pas qu'il est correct...)

Exemple : Perl, Java, ...

Fonctions communes à tous les langages

Variable : une variable est un petit espace mémoire réservé par le programme, dont le contenu est destiné à être utilisé lors des opérations programmées, et qui peut être modifié. La seule grandeur obligatoire pour une variable est son nom

Notions de programmation - Les variables

Selon le langage de programmation, les variables doivent être déclarées au début du programme : meilleur gestion de la mémoire (langage C). Certains langage ont des modes qui obligent cette déclaration (Fortran, Perl). Pour d'autre la déclaration est impossible (shell)

Notions de programmation - Les variables

Selon le langage de programmation, les variables ont un type spécifique : elles ne peuvent contenir qu'un seul genre de valeur : entier, réel, chaîne de caractère. La taille mémoire affectée est différente selon le type

Notions de programmation - Les variables

En langage C, Fortran, la déclaration de type est obligatoire. En perl, c'est le langage qui décide, selon le type d'opérations qui est réalisé, si la variable doit être traitée comme un nombre ou une chaîne de caractère.

Notions de programmation - Les variables

La déclaration de type est contraignante mais empêche les opérations impossibles (nombre + caractère), ou qui peuvent entraîner des erreurs d'arrondi (résultat réel stocké dans une variable entière)

La non-déclaration de type peut rendre la lecture du programme moins facile

Notions de programmation - L'affectation

L'affectation est l'opération qui permet de modifier le contenu d'un variable = lui affecter une valeur. Cette valeur peut être écrite directement ou être le résultat d'une opération. Le signe d'affectation est en général le signe = MAIS IL NE S'AGIT PAS D'UNE EGALITE

Notions de programmation - L'affectation

Affectations :

a = 3

a = b + c

toto = titi

toto = "titi"

b + 4 = c + 2

Notions de programmation - L'affectation

Affectations :

a = 3

a = b + c

toto = titi

toto = "titi"

~~b + 4 = c + 2~~

Notions de programmation - Les structures

En plus de commandes classiques, tous les langages de programmation possèdent des structures :

- Boucles itératives
- Boucles conditionnelles
- Structures conditionnelles

Programmation - Boucles itératives

Ces boucles permettent de répéter un nombre fini et défini à l'avance de fois une certaine série d'opérations. Le nombre de répétitions est donné par la variation d'une variable entière entre deux valeurs, ou bien par une liste de valeurs définies à l'avance qu'une variable prendra successivement.

Programmation - Boucles itératives

Exemples :

pour i compris entre 1 et 100, calculer i**2

pour i compris entre 1 et 10, faire a = a + b

pour file dans la liste nom1, nom2, nom3, copier file dans le répertoire $HOME/programmes

Programmation - Boucles conditionnelles

Ces boucles permettent de répéter une certaine série d'opérations sous condition.

while : la boucle tourne tant qu'une condition est réalisée

until : la boucle tourne jusqu'à ce qu'une certaine condition soit réalisée

Programmation - Boucles conditionnelles

Tant que a < b ,faire

```
a = a + 1
c = 2*c + a
```

Les opérations peuvent ne pas être réalisées

jusqu'à ce que a = b faire

```
a = a + 10
c = b + a
```

Les opérations sont toujours réalisées au moins une fois

Programmation - Boucles conditionnelles

```
a=10
b=1
Tant que a > b ,faire
    a = a + 1
    c = 2*c + a
```

Mal programmées, ces boucles peuvent ne pas s'arrêter ...
Il faut également faire en sorte que les instructions à l'intérieur de la structure modifie la condition

Notions de programmation - Les structures conditionnelles

Ces structures permettent de réaliser des instructions si une condition est réalisée, et éventuellement d'autres instructions sinon.

Les structures peuvent être imbriquées sur plusieurs niveaux

Notions de programmation - Les structures conditionnelles

```
Si condition1 alors
        bloc_instructions_1
sinon si condition2 alors
        bloc_instructions_2
sinon si.....
....
sinon
        dernier_bloc_instructions
fin
```