Éléments d'architecture des ordinateurs

Jean-Pierre Becirspahic
Lycée Louis-Le-Grand

Naissance de l'informatique

2e congrès international des mathématiciens, Paris 1900

10e problème de Hilbert :

De la possibilité de résoudre une équation diophantienne.

Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers rationnels.

Naissance de l'informatique

2e congrès international des mathématiciens, Paris 1900

10e problème de Hilbert :

De la possibilité de résoudre une équation diophantienne.

Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers rationnels.

En termes modernes :

Existe-t-il un algorithme déterminant si une équation diophantienne a des solutions rationnelles ?

Naissance de l'informatique

2e congrès international des mathématiciens, Paris 1900

10e problème de Hilbert :

De la possibilité de résoudre une équation diophantienne.

Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers rationnels.

En termes modernes :

Existe-t-il un algorithme déterminant si une équation diophantienne a des solutions rationnelles ?

Pour répondre à cette question, il faut donner une définition précise de ce qu'est un algorithme : la thèse de Church (1936) s'y attelle.

Naissance de l'informatique

2e congrès international des mathématiciens, Paris 1900

10e problème de Hilbert :

De la possibilité de résoudre une équation diophantienne.

Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers rationnels.

En termes modernes :

Existe-t-il un algorithme déterminant si une équation diophantienne a des solutions rationnelles ?

Pour répondre à cette question, il faut donner une définition précise de ce qu'est un algorithme : la thèse de Church (1936) s'y attelle.

Remarque. Grace aux travaux de Church, Matiiassevitch répond par la négative au 10e problème de Hilbert en 1970.

Naissance de l'informatique

Deux paternités distinctes

Une approche théorique. Vers 1930 quelques logiciens cherchent à répondre à la question suivante : *qu'est-ce qu'un calcul ?* et fondent la théorie de la calculabilité.

Naissance de l'informatique

Deux paternités distinctes

Une approche théorique. Vers 1930 quelques logiciens cherchent à répondre à la question suivante : *qu'est-ce qu'un calcul ?* et fondent la théorie de la calculabilité.

- Church découvre le «lambda-calcul», qui définit précisément par une axiomatique ce qu'est une fonction calculable.

Naissance de l'informatique

Deux paternités distinctes

Une approche théorique. Vers 1930 quelques logiciens cherchent à répondre à la question suivante : *qu'est-ce qu'un calcul ?* et fondent la théorie de la calculabilité.

- Church découvre le «lambda-calcul», qui définit précisément par une axiomatique ce qu'est une fonction calculable.
- Turing lie la notion de calcul à la notion de machine : est calculable toute fonction qui peut être exécutée par une machine (abstraite).

Naissance de l'informatique

Deux paternités distinctes

Une approche théorique. Vers 1930 quelques logiciens cherchent à répondre à la question suivante : *qu'est-ce qu'un calcul ?* et fondent la théorie de la calculabilité.

- Church découvre le «lambda-calcul», qui définit précisément par une axiomatique ce qu'est une fonction calculable.
- Turing lie la notion de calcul à la notion de machine : est calculable toute fonction qui peut être exécutée par une machine (abstraite).

Une approche concrète. Peut après 1945 des ingénieurs commencent à construire des super-calculateurs. Parmi eux, Von Neuman développe une architecture qui, dans ses principes fondamentaux, continue encore aujourd'hui à régir la plus-part des ordinateurs actuels.

Naissance de l'informatique

Deux paternités distinctes

Une approche théorique. Vers 1930 quelques logiciens cherchent à répondre à la question suivante : *qu'est-ce qu'un calcul ?* et fondent la théorie de la calculabilité.

- Church découvre le «lambda-calcul», qui définit précisément par une axiomatique ce qu'est une fonction calculable.
- Turing lie la notion de calcul à la notion de machine : est calculable toute fonction qui peut être exécutée par une machine (abstraite).

Une approche concrète. Peut après 1945 des ingénieurs commencent à construire des super-calculateurs. Parmi eux, Von Neuman développe une architecture qui, dans ses principes fondamentaux, continue encore aujourd'hui à régir la plus-part des ordinateurs actuels.

Aujourd'hui, on pourrait dire que la science algorithmique est la lointaine descendante des travaux de Church et Turing ; la programmation sur ordinateur sa réalisation concrète.

Principaux composants d'un ordinateur

Un ordinateur est une machine de traitement de l'information : il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme.

Principaux composants d'un ordinateur

Un ordinateur est une machine de traitement de l'information : il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme.

- L'acquisition de l'information se fait par l'intermédiaire de périphériques d'entrées (clavier, souris, micro, webcam, scanner, écran tactile, etc.)

Principaux composants d'un ordinateur

Un ordinateur est une machine de traitement de l'information : il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme.

- L'acquisition de l'information se fait par l'intermédiaire de périphériques d'entrées (clavier, souris, micro, webcam, scanner, écran tactile, etc.)
- Le stockage de l'information utilise la mémoire d'un ordinateur : mémoire de masse (disque dur, clé USB, etc.) destinée au stockage persistant et mémoire vive (RAM) pour traiter les données.

Principaux composants d'un ordinateur

Un ordinateur est une machine de traitement de l'information : il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme.

- L'acquisition de l'information se fait par l'intermédiaire de périphériques d'entrées (clavier, souris, micro, webcam, scanner, écran tactile, etc.)
- Le stockage de l'information utilise la mémoire d'un ordinateur : mémoire de masse (disque dur, clé USB, etc.) destinée au stockage persistant et mémoire vive (RAM) pour traiter les données.
- La transformation de l'information est le rôle du processeur (CPU) : unité de commande (lecture et décodage des instructions) et unité de traitement (exécution des instructions).

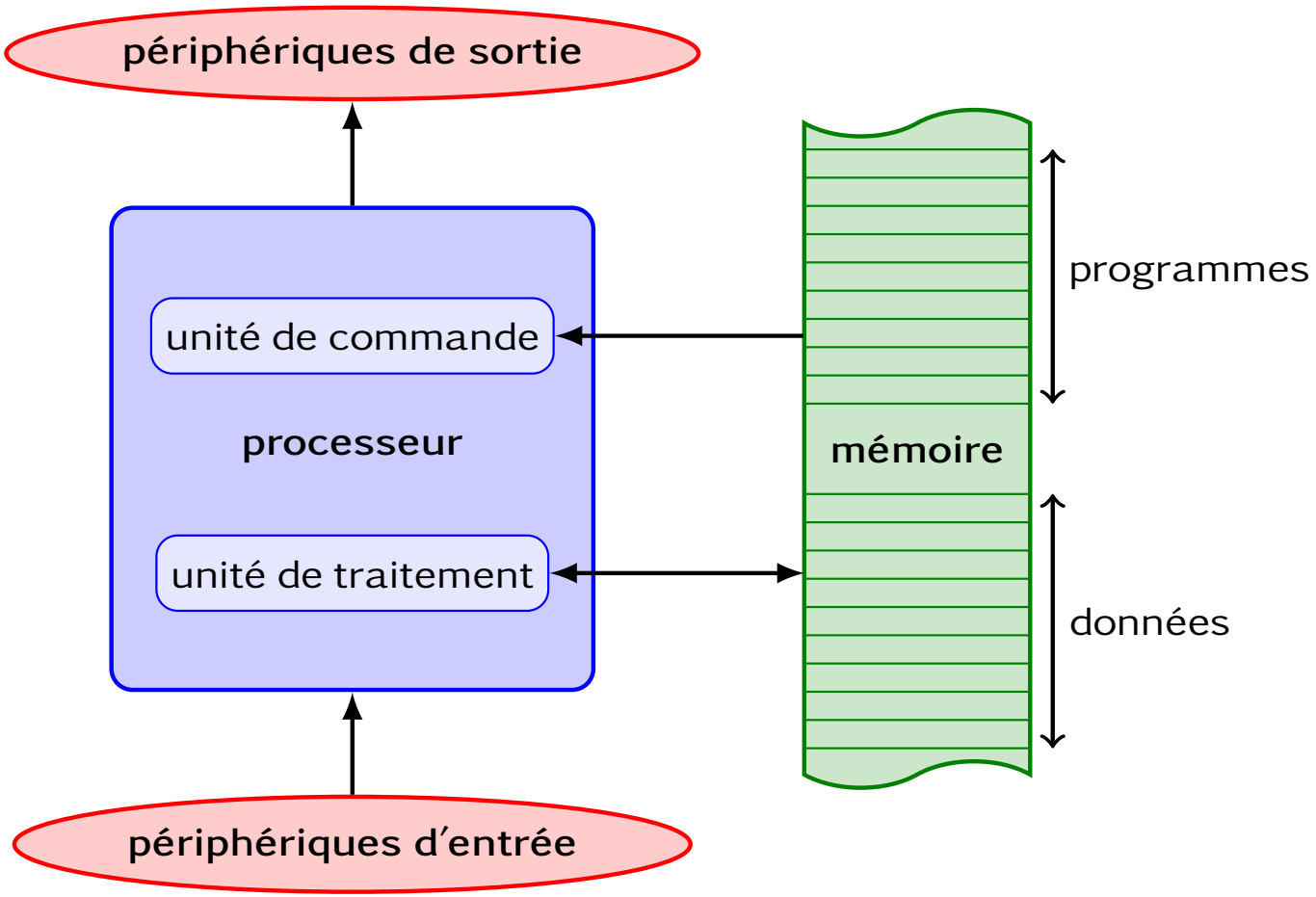
Principaux composants d'un ordinateur

Un ordinateur est une machine de traitement de l'information : il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme.

- L'acquisition de l'information se fait par l'intermédiaire de périphériques d'entrées (clavier, souris, micro, webcam, scanner, écran tactile, etc.)
- Le stockage de l'information utilise la mémoire d'un ordinateur : mémoire de masse (disque dur, clé USB, etc.) destinée au stockage persistant et mémoire vive (RAM) pour traiter les données.
- La transformation de l'information est le rôle du processeur (CPU) : unité de commande (lecture et décodage des instructions) et unité de traitement (exécution des instructions).
- La restitution de l'information utilise les périphériques de sorties (écran, imprimante, enceintes acoustiques, etc.)

Principaux composants d'un ordinateur

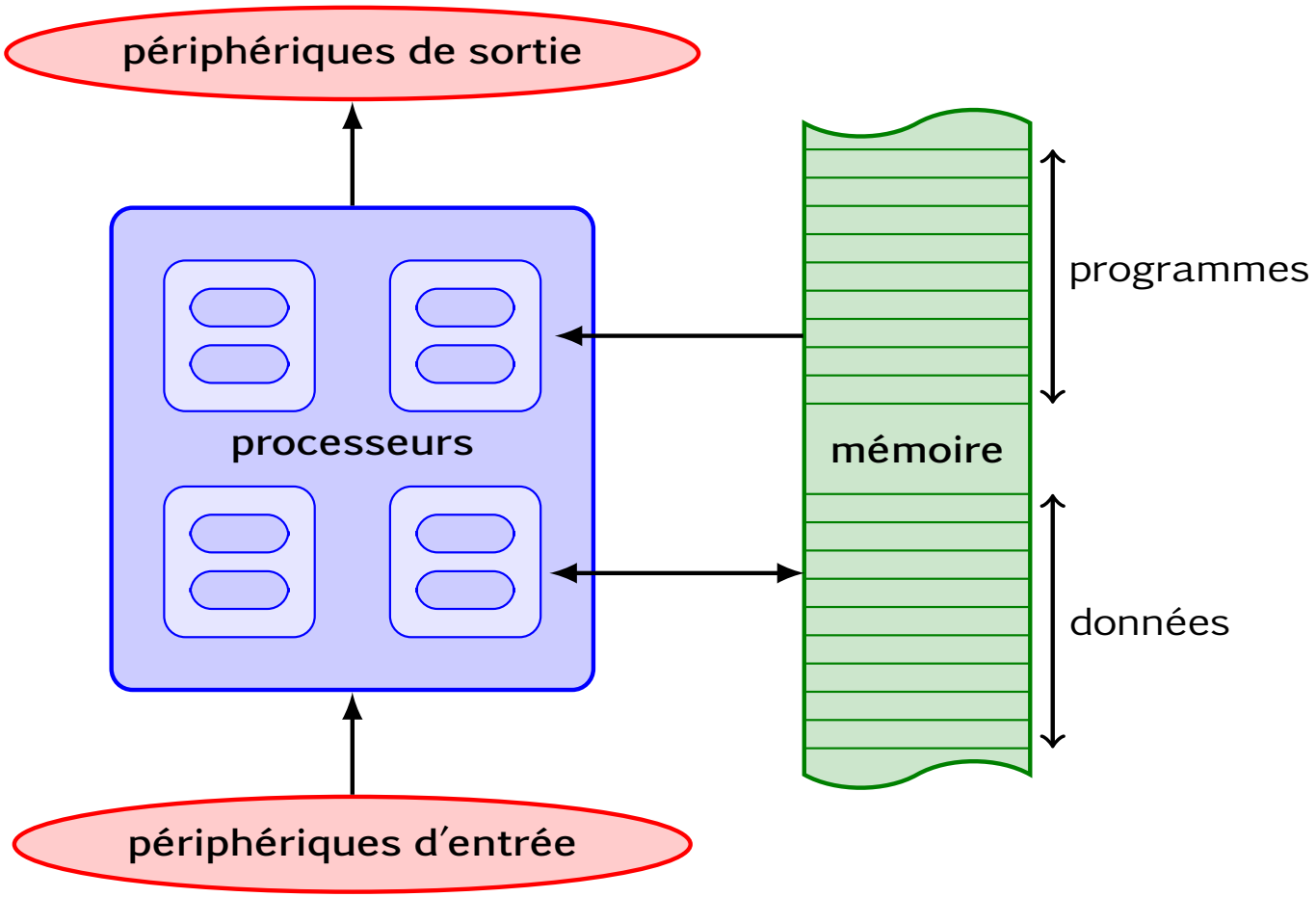
Ces différents éléments s'articulent suivant le schéma :



Le sens des flèches (bus) indique le sens de transit de l'information.

Principaux composants d'un ordinateur

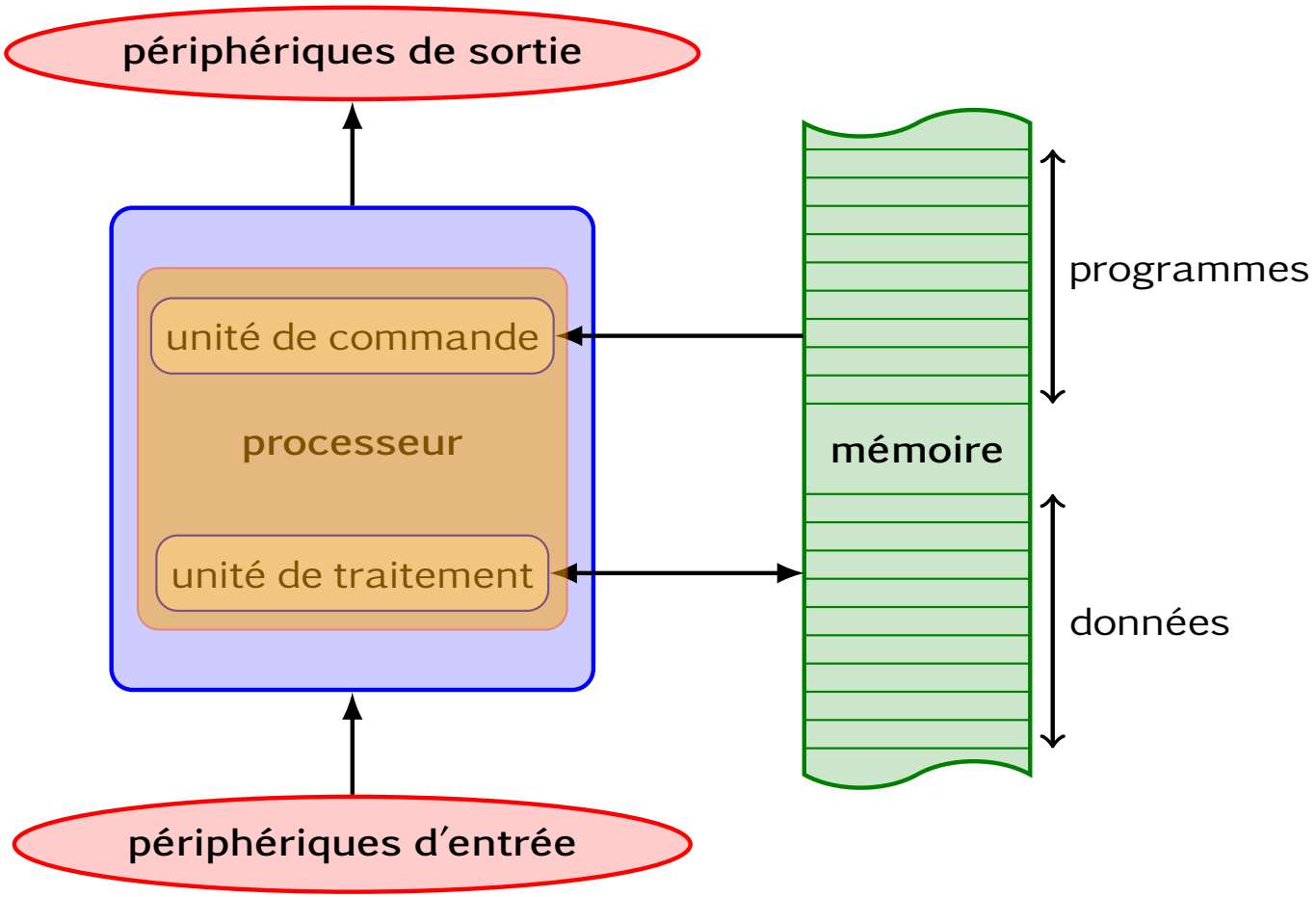
Ces différents éléments s'articulent suivant le schéma :



Principale différence aujourd'hui : les ordinateurs actuels intègrent des processeurs multiples, ainsi que les périphériques d'entrée/sortie (carte vidéo, carte son ...)

Principaux composants d'un ordinateur

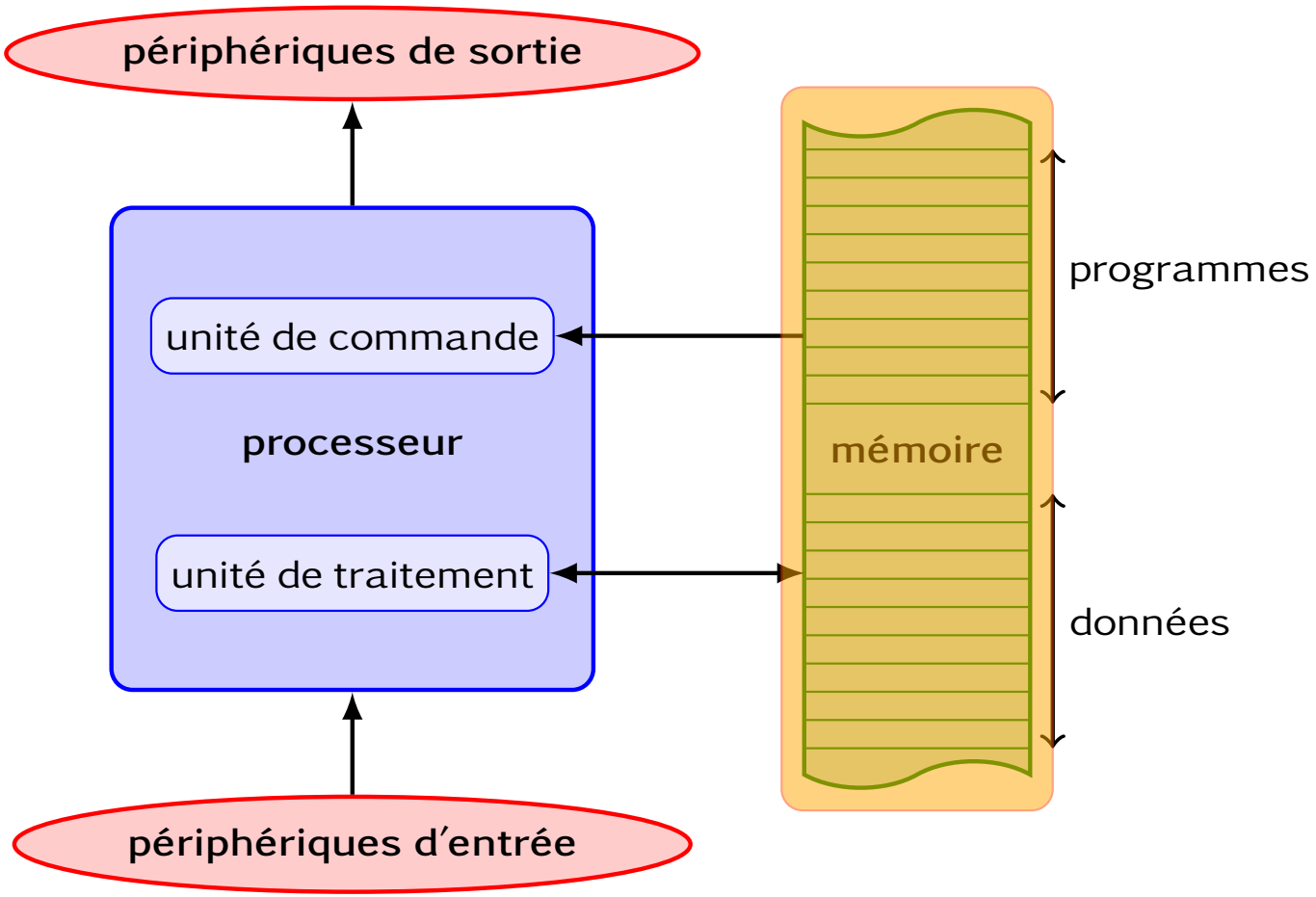
Ces différents éléments s'articulent suivant le schéma :



Premier apport de Von Neuman : séparation de l'unité de commande et de l'unité de traitement.

Principaux composants d'un ordinateur

Ces différents éléments s'articulent suivant le schéma :



Second apport de Von Neumann : instructions et données sont rangées dans un même espace de stockage.

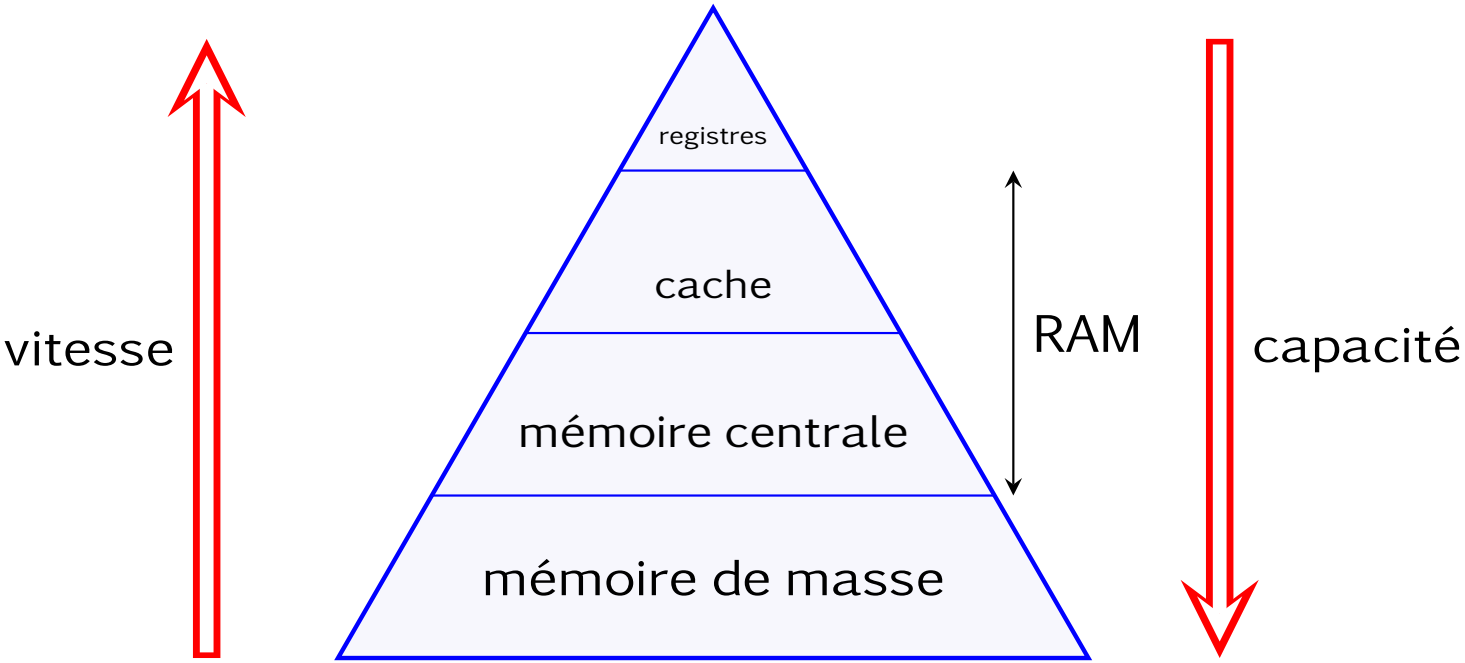
Les différents types de mémoire

Il faudra toujours bien distinguer :

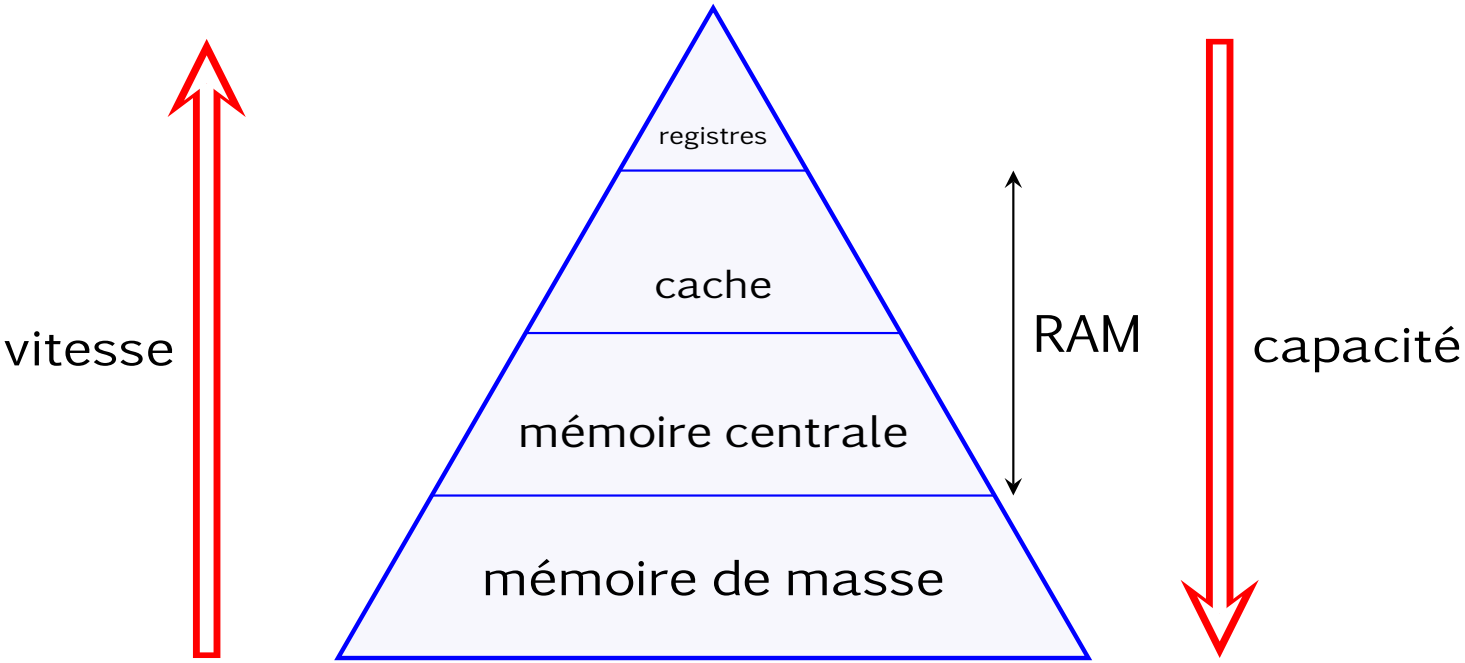
- le type de l'information traitée (un nombre, un texte, une image, un son, un programme ...)
- et sa représentation en machine, qui se réduit toujours à une succession de bits (0 ou 1).

La même succession de bits peut représenter des objets de nature différente.

Les différents types de mémoire

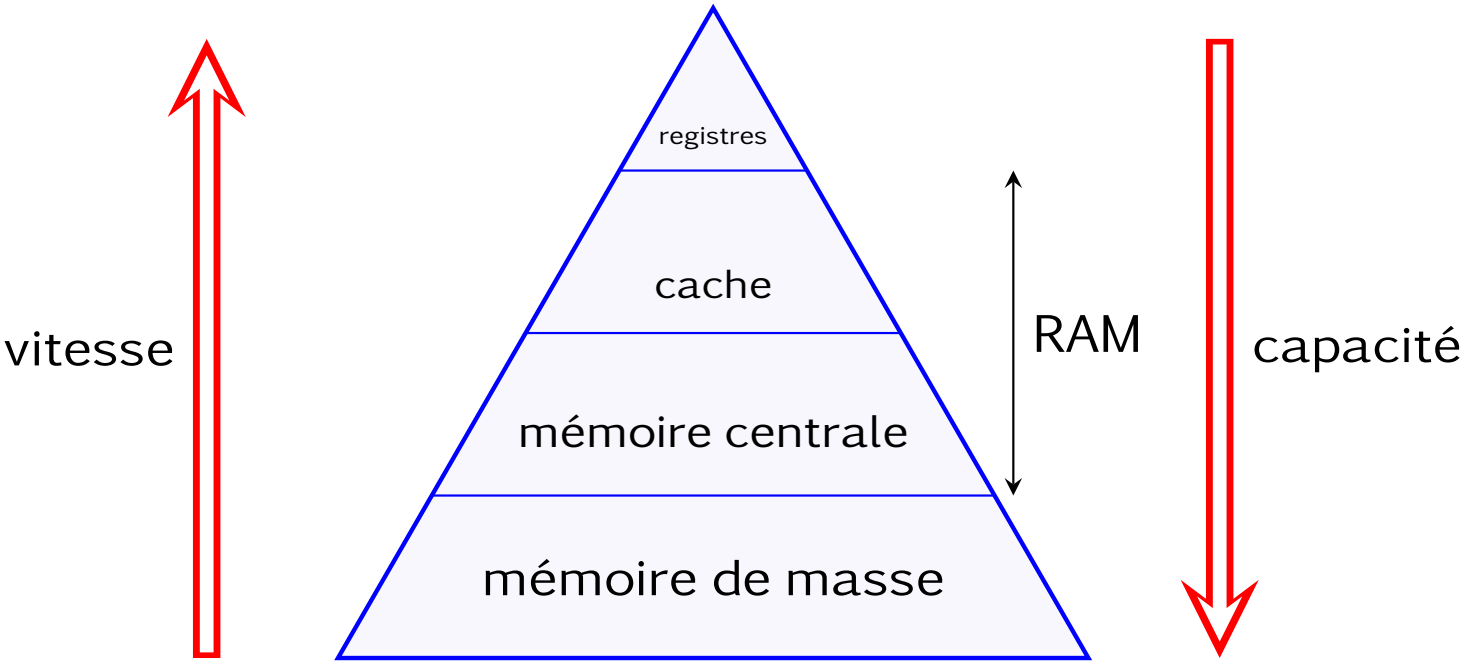


Les différents types de mémoire



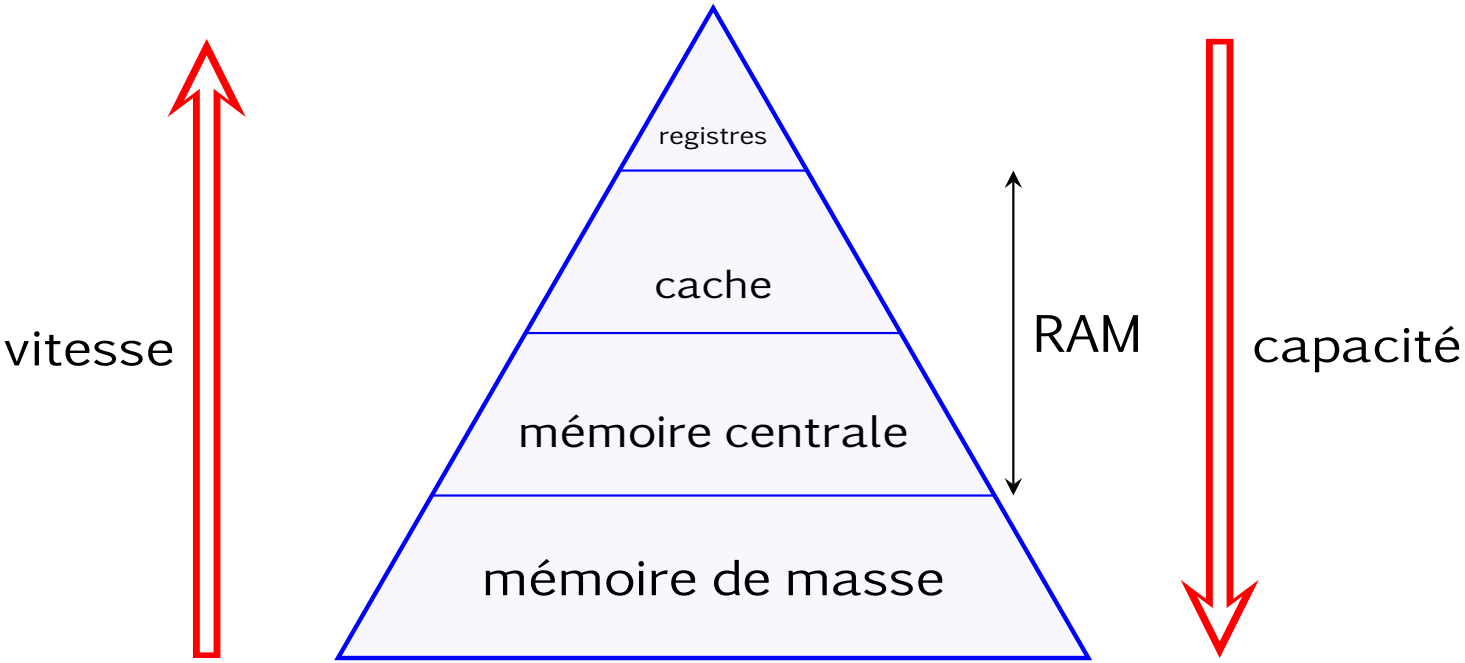
- Les registres sont des emplacements mémoires internes au processeur dans lesquels ces derniers exécutent les instructions. Leur nombre varie d'une dizaine à une centaine, et leur capacité dépasse rarement quelques dizaines d'octets.

Les différents types de mémoire



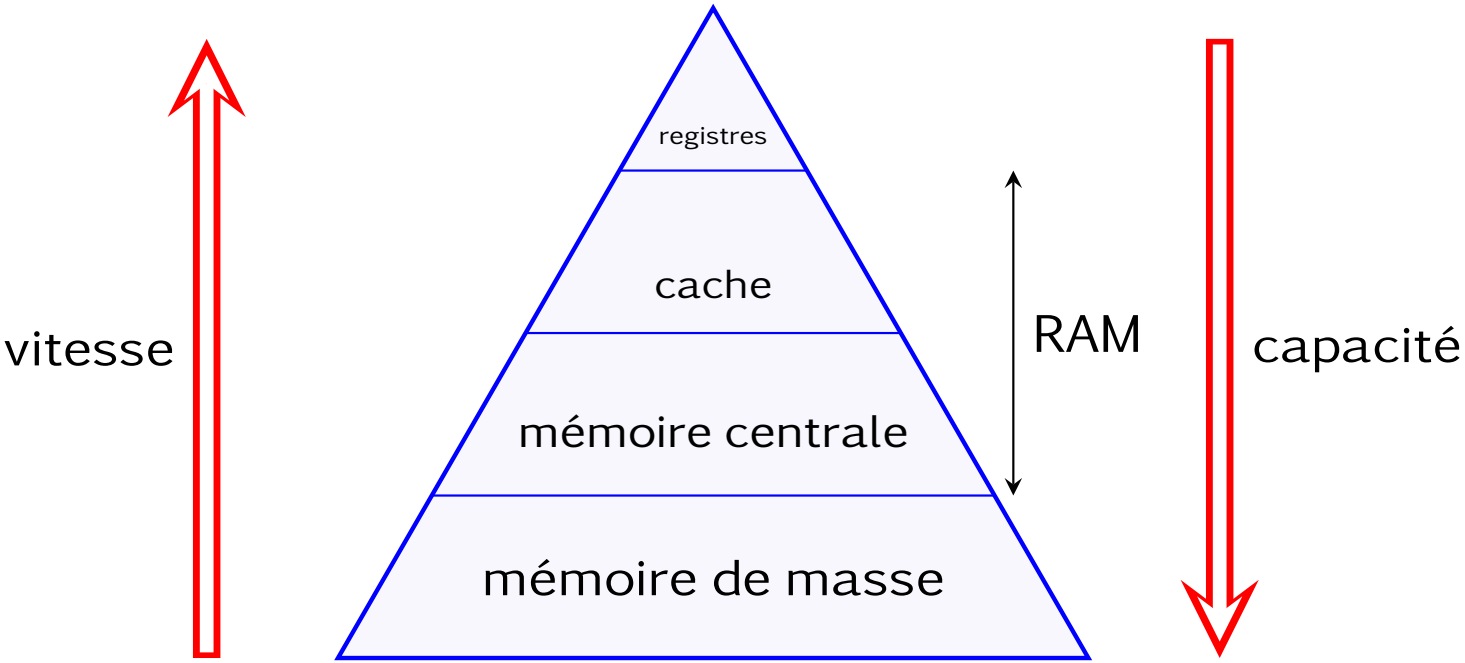
- La mémoire cache est une mémoire rapide permettant de réduire les délais d'attente des informations stockées dans la mémoire centrale. Il existe plusieurs niveaux de caches qui se distinguent par leur vitesse d'accès. La capacité d'un cache se compte en kilo-octets pour les deux premiers niveaux et en mega-octets pour le troisième.

Les différents types de mémoire



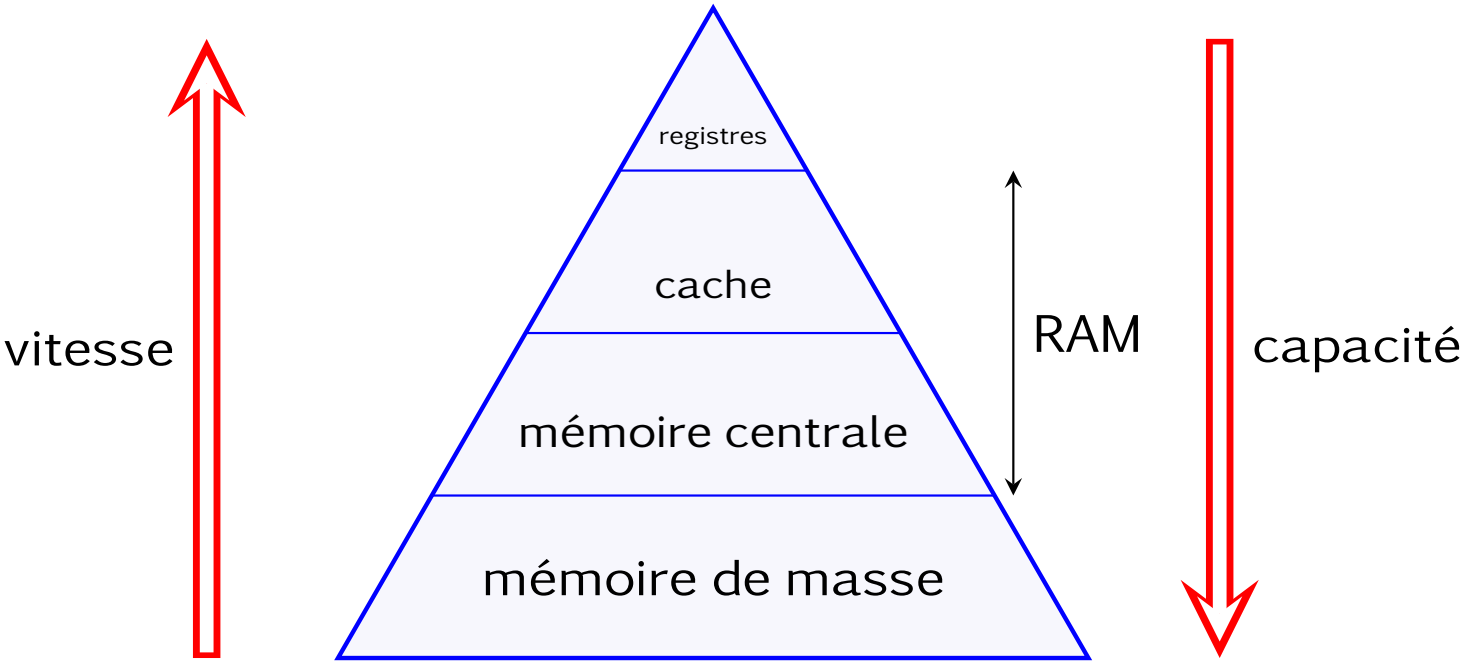
- La mémoire centrale contient le code et les données des programmes exécutés par le processeur ; elle se compte actuellement en giga-octets.

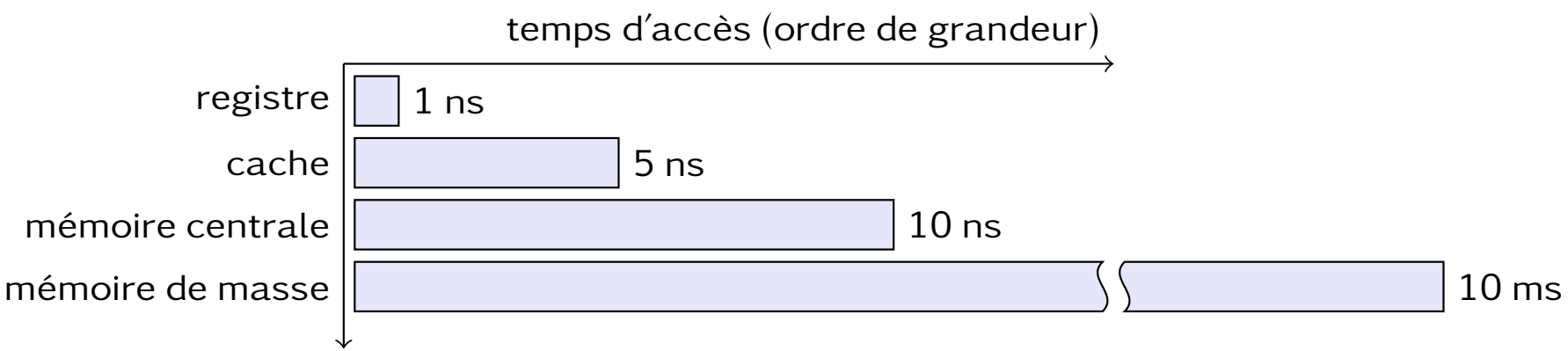
Les différents types de mémoire



- La mémoire de masse désigne tous les moyens de stockage pérenne dont on dispose : disque dur, clé USB, DVD, etc. Lors du lancement d'une application, les données essentielles sont copiées dans la mémoire centrale, et les programmes sont conçus pour minimiser les accès à la mémoire de masse, très lente. La mémoire de masse tend de plus en plus à être mesurée en tera-octets.

Les différents types de mémoire





La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).
- $1\,000$ ko $= 10^6$ octets forment un megaoctet (1 Mo) ;
  $1\,024$ kio $= 2^{20}$ octets forment un mebioctet (1 Mio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).
- $1\,000$ ko $= 10^6$ octets forment un megaoctet (1 Mo) ;
  $1\,024$ kio $= 2^{20}$ octets forment un mebioctet (1 Mio).
- $1\,000$ Mo $= 10^9$ octets forment un gigaoctet (1 Go) ;
  $1\,024$ Mio $= 2^{30}$ octets forment un gibioctet (1 Gio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).
- $1\,000$ ko $= 10^6$ octets forment un megaoctet (1 Mo) ;
  $1\,024$ kio $= 2^{20}$ octets forment un mebioctet (1 Mio).
- $1\,000$ Mo $= 10^9$ octets forment un gigaoctet (1 Go) ;
  $1\,024$ Mio $= 2^{30}$ octets forment un gibioctet (1 Gio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

Un emplacement mémoire (d'un byte) est repéré par son *adresse*, manipulée dans les registres internes des processeurs.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).
- $1\,000$ ko $= 10^6$ octets forment un megaoctet (1 Mo) ;
  $1\,024$ kio $= 2^{20}$ octets forment un mebioctet (1 Mio).
- $1\,000$ Mo $= 10^9$ octets forment un gigaoctet (1 Go) ;
  $1\,024$ Mio $= 2^{30}$ octets forment un gibioctet (1 Gio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

Un emplacement mémoire (d'un byte) est repéré par son *adresse*, manipulée dans les registres internes des processeurs.

Un processeur 32 bits utilise des registres de 32 bits donc peut gérer au plus 2^{32} adresses différentes, soit 4 Gio de mémoire vive.

La mémoire principale

Une information est représentée et manipulée par l'ordinateur sous forme binaire : c'est une suite de 0 et de 1 (une suite de bits).

- 8 bits forment un octet ;
- $1\,000 = 10^3$ octets forment un kilooctet (1 ko) ;
  $1\,024 = 2^{10}$ octets forment un kibioctet (1 kio).
- $1\,000$ ko $= 10^6$ octets forment un megaoctet (1 Mo) ;
  $1\,024$ kio $= 2^{20}$ octets forment un mebioctet (1 Mio).
- $1\,000$ Mo $= 10^9$ octets forment un gigaoctet (1 Go) ;
  $1\,024$ Mio $= 2^{30}$ octets forment un gibioctet (1 Gio).

Un byte est la plus petite unité de mémoire à laquelle on peut accéder ; il correspond (dans la majorité des cas) à 8 bits, c'est à dire un octet.

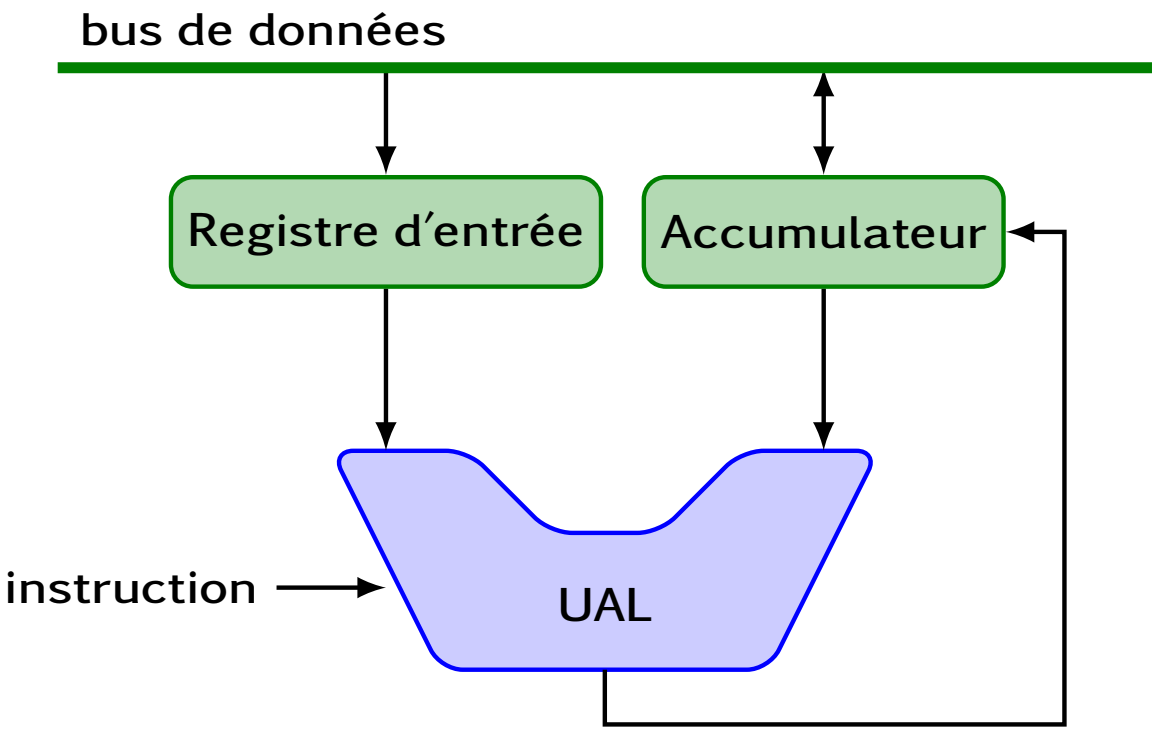
Un emplacement mémoire (d'un byte) est repéré par son *adresse*, manipulée dans les registres internes des processeurs.

Un processeur 64 bits utilise des registres de 64 bits : les instructions, les entiers et les adresses sont codés sur 8 octets. C'est la norme dans les ordinateurs actuels (processeurs intel i5, i7...).

Le processeur

Un processeur (ou CPU, pour *Central Process Unit*) est composé :

- d'une unité de commande, qui décode les instructions d'un programme et les transcrit en une succession d'instructions élémentaires envoyées à l'unité de traitement ;
- d'une unité de traitement, composée d'une unité arithmétique et logique (UAL), d'un registre d'entrée et d'un accumulateur.



Le processeur

L'unité de traitement manipule :

- des entiers (codés sur 32 ou 64 bits) ;
- des adresses mémoire ;
- des instructions, qui constituent le *langage machine*.

Le processeur

L'unité de traitement manipule :

- des entiers (codés sur 32 ou 64 bits) ;
- des adresses mémoire ;
- des instructions, qui constituent le *langage machine*.

Un programme est une suite d'instructions, autrement dit une suite de mots mémoire écrits en binaire. Pour en faciliter la lecture et l'écriture, on représente ces mots par des mnémoniques (LOAD, ADD, JMP...) ; on parle alors de *langage assembleur*.

Le processeur

L'unité de traitement manipule :

- des entiers (codés sur 32 ou 64 bits) ;
- des adresses mémoire ;
- des instructions, qui constituent le *langage machine*.

Un exemple de programme : ranger à l'adresse c la somme des nombres se trouvant aux adresses a et b.

- en langage assembleur :

```
LOAD A     ; copie du contenu de A dans l'accumulateur
ADD B      ; ajoute du contenu de B dans l'accumulateur
STO C      ; stockage du contenu de l'accumulateur à l'adresse C
```

- en PYTHON :

```
c = a + b
```

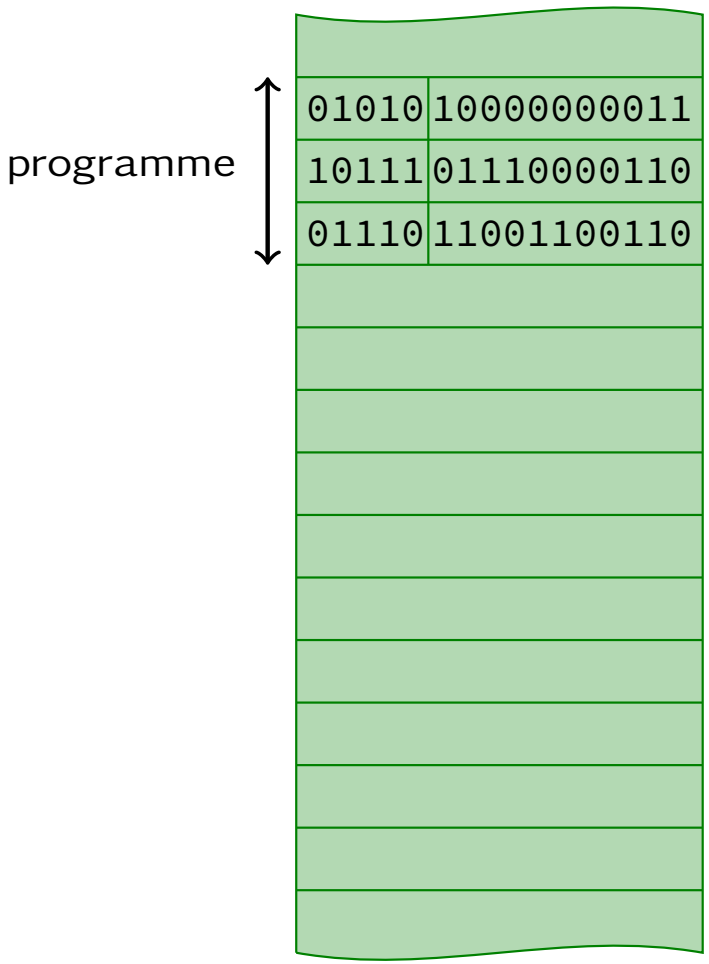
Le processeur

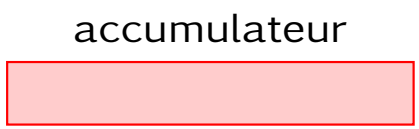
Mots-mémoire de 16 bits : chaque instruction comporte un code d'opération sur 5 bits et une adresse sur 11 bits.

assembleur	machine	rôle
LOAD X	01010	charge dans l'accumulateur le contenu de X
STO X	01110	range le contenu de l'accumulateur à l'adresse X
ADD X	10111	charge dans le registre le contenu de l'adresse X et l'ajoute à l'accumulateur

Le processeur

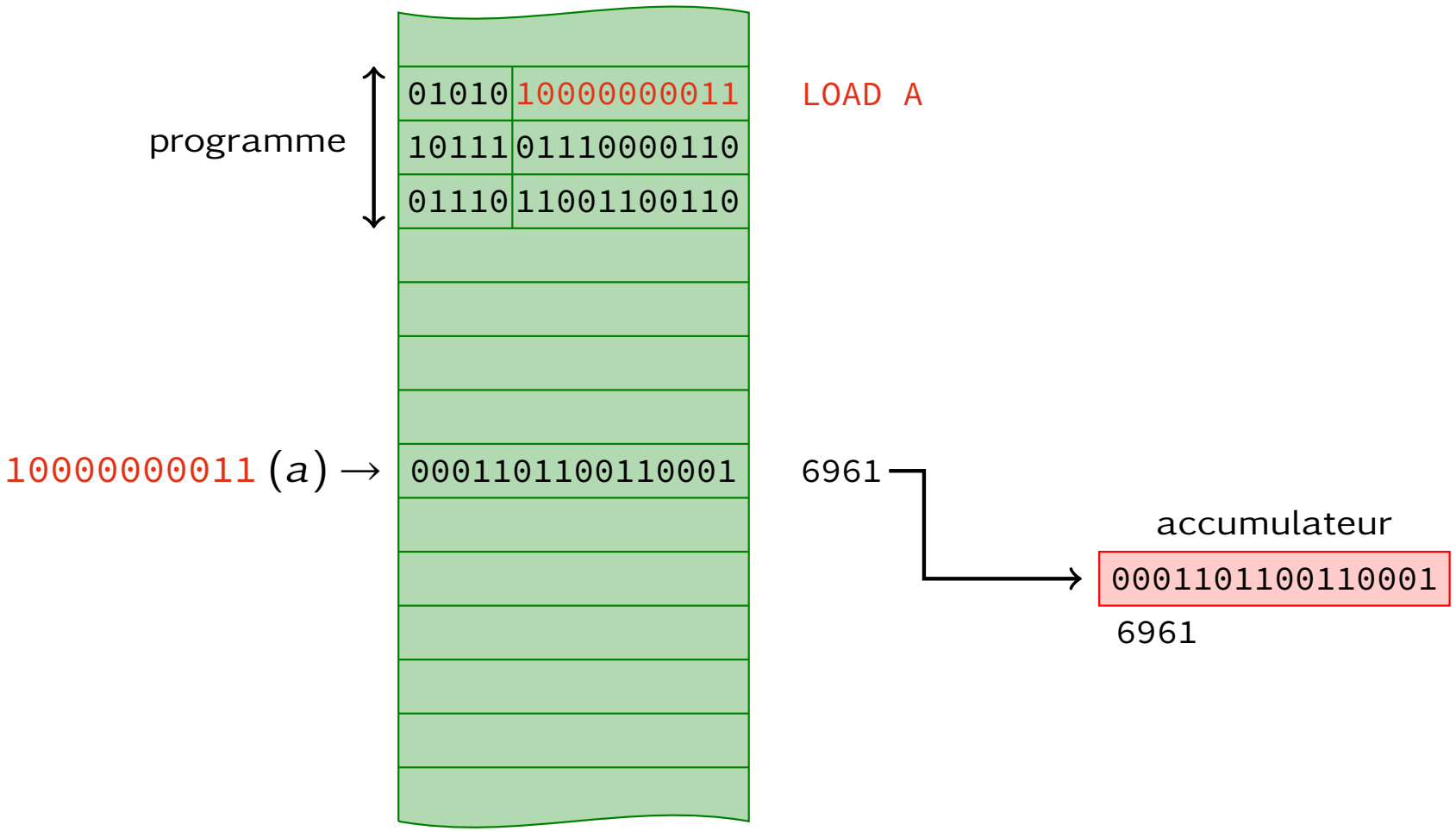
Mots-mémoire de 16 bits : chaque instruction comporte un code d'opération sur 5 bits et une adresse sur 11 bits.





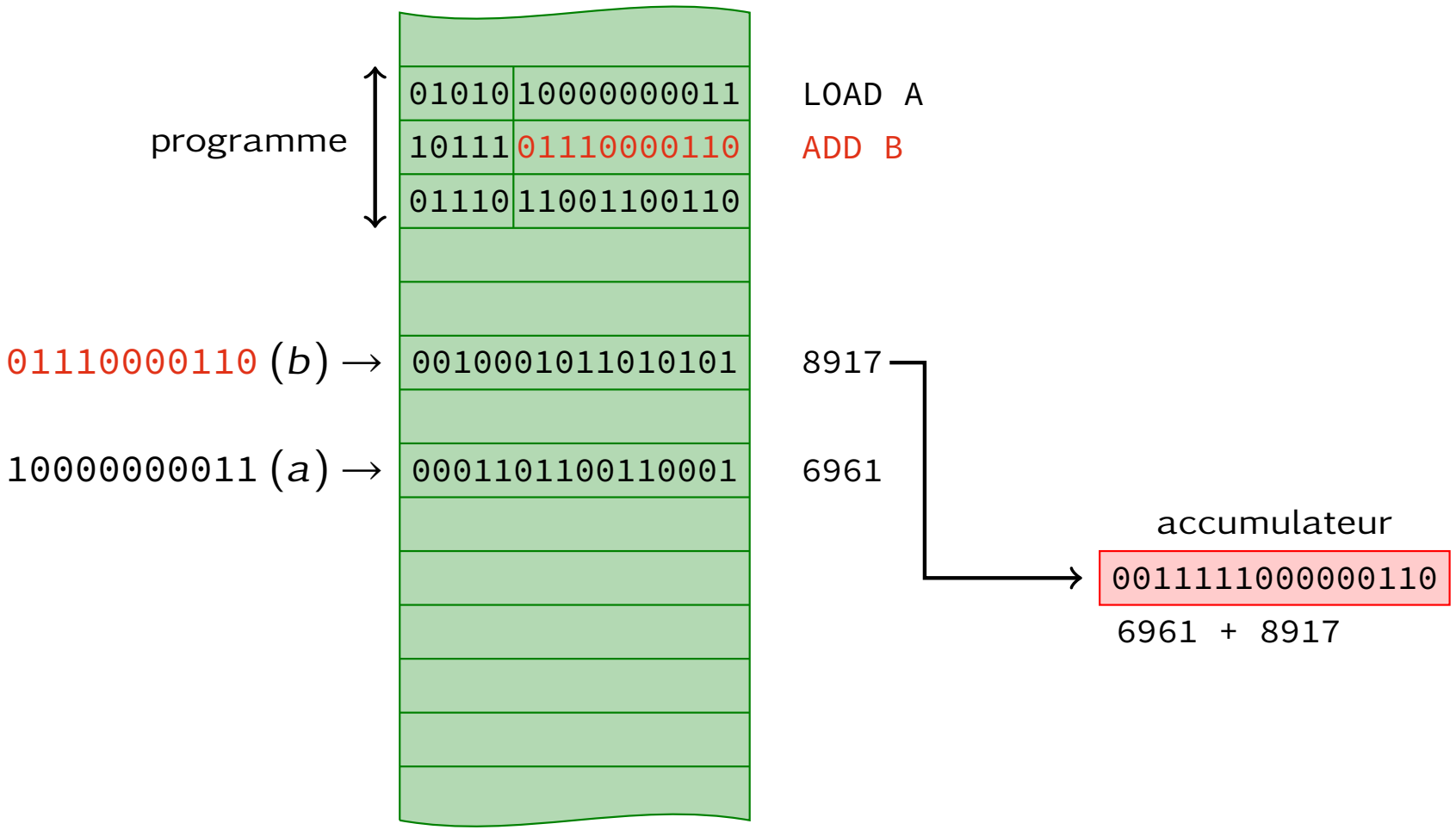
Le processeur

Mots-mémoire de 16 bits : chaque instruction comporte un code d'opération sur 5 bits et une adresse sur 11 bits.



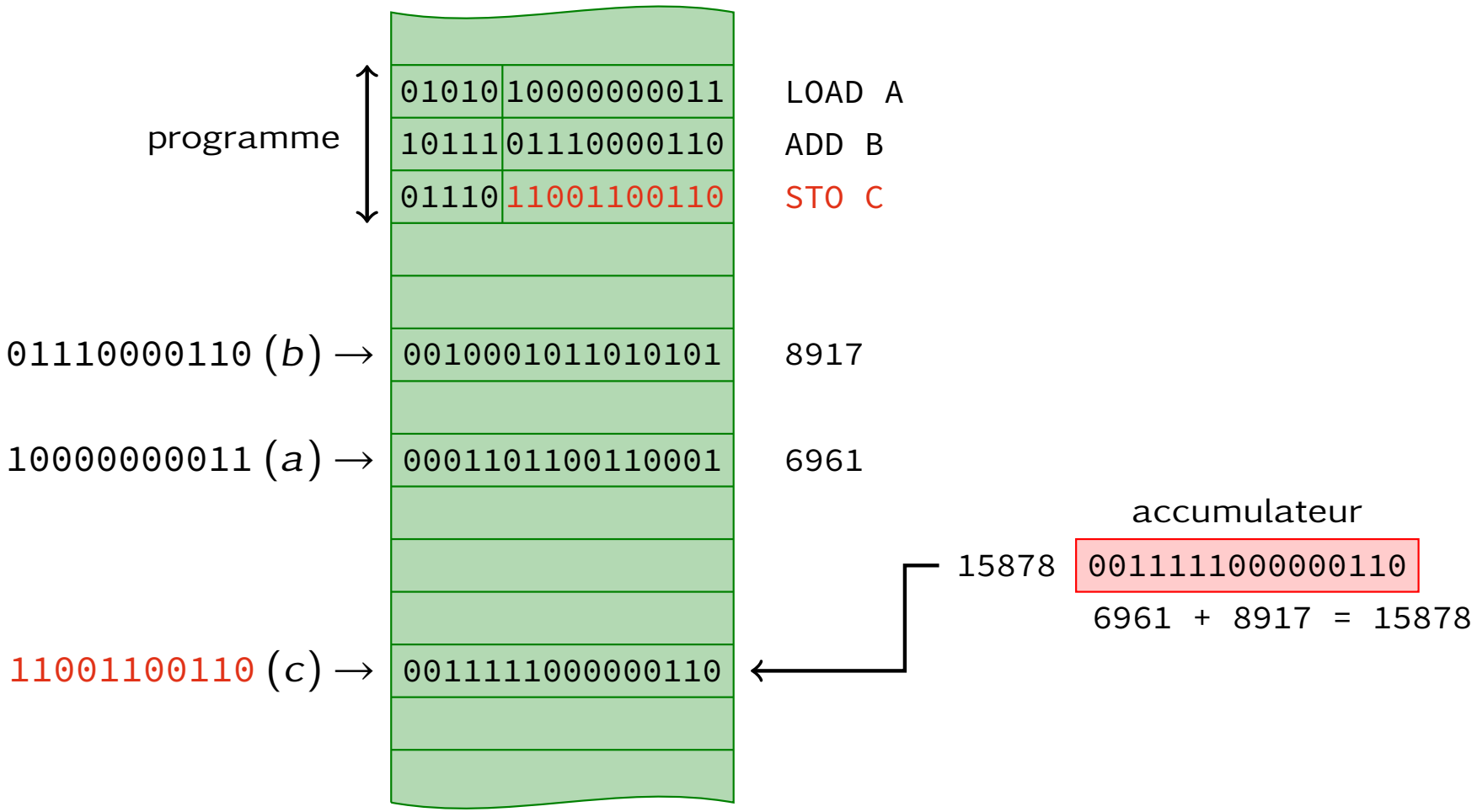
Le processeur

Mots-mémoire de 16 bits : chaque instruction comporte un code d'opération sur 5 bits et une adresse sur 11 bits.



Le processeur

Mots-mémoire de 16 bits : chaque instruction comporte un code d'opération sur 5 bits et une adresse sur 11 bits.



Langages de programmation

Le défaut majeur de la programmation en assembleur : un programme en assembleur est écrit en termes de ce que sait faire la machine.
En particulier, il est dépendant de type de processeur présent dans l'ordinateur : si on change de processeur, il faut réécrire le programme.

Langages de programmation

Le défaut majeur de la programmation en assembleur : un programme en assembleur est écrit en termes de ce que sait faire la machine.
En particulier, il est dépendant de type de processeur présent dans l'ordinateur : si on change de processeur, il faut réécrire le programme.

L'invention des langages de programmation va permettre de changer de paradigme en rédigeant un programme en termes de ce que veut faire l'utilisateur.

Langages de programmation

Le défaut majeur de la programmation en assembleur : un programme en assembleur est écrit en termes de ce que sait faire la machine.
En particulier, il est dépendant de type de processeur présent dans l'ordinateur : si on change de processeur, il faut réécrire le programme.

L'invention des langages de programmation va permettre de changer de paradigme en rédigeant un programme en termes de ce que veut faire l'utilisateur.

- FORTRAN (1954) est le premier langage de haut niveau ayant connu une large diffusion ; il était (et est toujours) destiné au calcul scientifique.

Langages de programmation

Le défaut majeur de la programmation en assembleur : un programme en assembleur est écrit en termes de ce que sait faire la machine.
En particulier, il est dépendant de type de processeur présent dans l'ordinateur : si on change de processeur, il faut réécrire le programme.

L'invention des langages de programmation va permettre de changer de paradigme en rédigeant un programme en termes de ce que veut faire l'utilisateur.

- Fortran (1954) est le premier langage de haut niveau ayant connu une large diffusion ; il était (et est toujours) destiné au calcul scientifique.

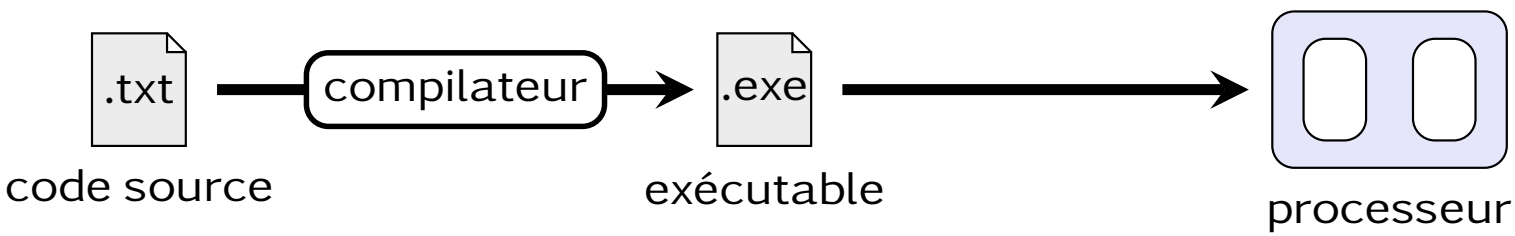
Pour exécuter un programme écrit en Fortran, on utilise un compilateur, un programme chargé de traduire le code source en langage machine.

Langages de programmation

Compilateurs et interprètes

Pour exécuter un code écrit dans un langage de programmation, il faut posséder un traducteur entre ce dernier et le langage machine. Schématiquement, il en existe de deux types :

- un compilateur traduit le code de manière définitive pour créer un *exécutable* ;

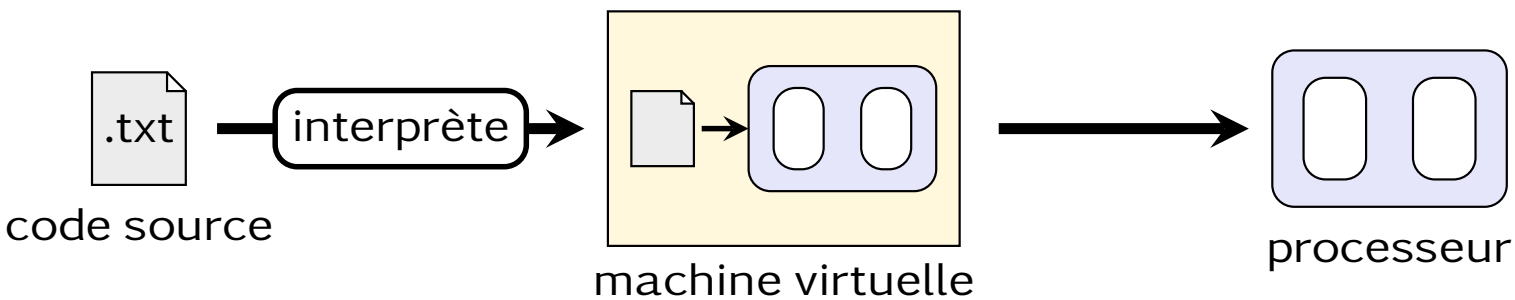


Langages de programmation

Compilateurs et interprètes

Pour exécuter un code écrit dans un langage de programmation, il faut posséder un traducteur entre ce dernier et le langage machine. Schématiquement, il en existe de deux types :

- un compilateur traduit le code de manière définitive pour créer un *exécutable* ;
- un interprète simule une machine virtuelle dont le langage machine serait le langage de programmation lui-même. Le code est lu, analysé et exécuté instruction par instruction par l'interprète.

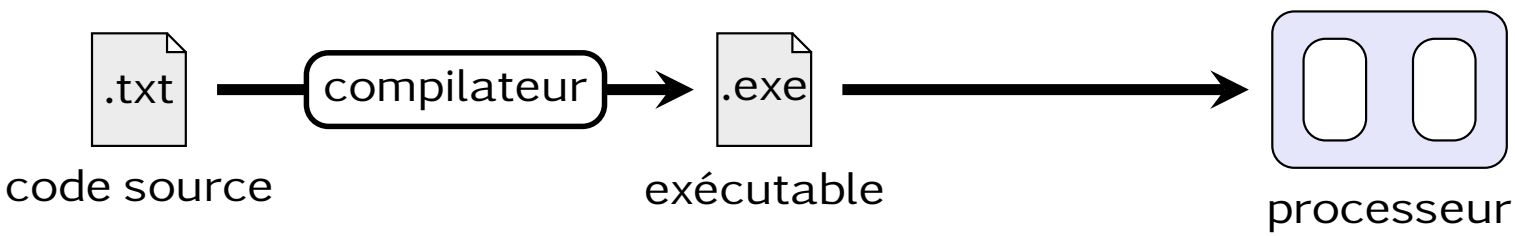


Langages de programmation

Compilateurs et interprètes

Pour exécuter un code écrit dans un langage de programmation, il faut posséder un traducteur entre ce dernier et le langage machine. Schématiquement, il en existe de deux types :

- un compilateur traduit le code de manière définitive pour créer un *exécutable* ;
- un interprète simule une machine virtuelle dont le langage machine serait le langage de programmation lui-même. Le code est lu, analysé et exécuté instruction par instruction par l'interprète.



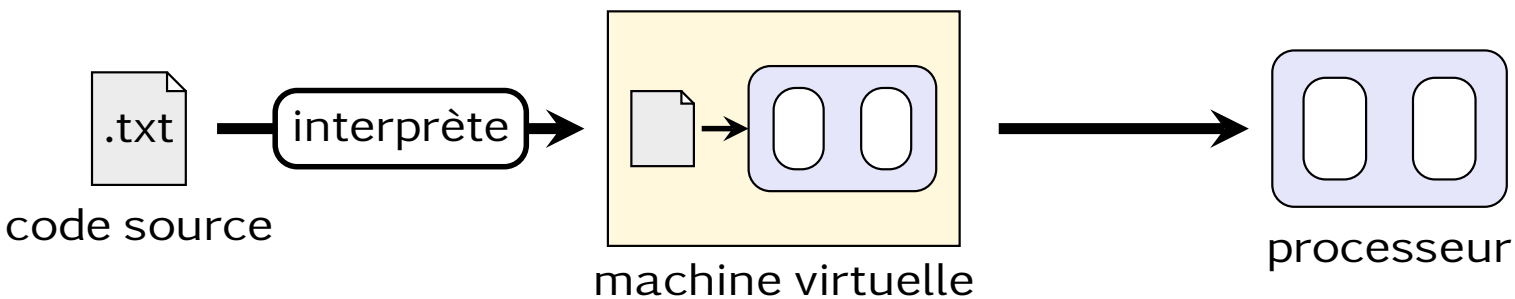
L'exécutable créé par le compilateur dépend de la machine sur lequel il a été créé, contrairement à un programme interprété est indépendant de la machine (puisque c'est l'interprète qui dépend de la machine).

Langages de programmation

Compilateurs et interprètes

Pour exécuter un code écrit dans un langage de programmation, il faut posséder un traducteur entre ce dernier et le langage machine. Schématiquement, il en existe de deux types :

- un compilateur traduit le code de manière définitive pour créer un *exécutable* ;
- un interprète simule une machine virtuelle dont le langage machine serait le langage de programmation lui-même. Le code est lu, analysé et exécuté instruction par instruction par l'interprète.



Mais un programme compilé est nettement plus rapide qu'un programme interprété : toute la phase d'analyse et de vérification du code source a déjà été faite lors de la phase de compilation.

Langages de programmation

Algol, Cobol, Lisp

À partir de 1955 un grand nombre de langages furent proposés, souvent pour répondre à des faiblesses du Fortran.

- L'Algol est le fruit d'un travail universitaire visant à définir un langage «universel» de programmation. Peu utilisé, il a néanmoins posé les bases de l'algorithmique et a influencé de nombreux langages plus récents (Python parmi tant d'autres).

Langages de programmation

Algol, Cobol, Lisp

À partir de 1955 un grand nombre de langages furent proposés, souvent pour répondre à des faiblesses du Fortran.

- L'Algol est le fruit d'un travail universitaire visant à définir un langage «universel» de programmation. Peu utilisé, il a néanmoins posé les bases de l'algorithmique et a influencé de nombreux langages plus récents (Python parmi tant d'autres).
- Le langage Cobol a été créé pour répondre à un besoin en informatique de gestion, domaine dans lequel Fortran était peu adapté. Bien que présentant de nombreux défauts, il est néanmoins resté pendant longtemps un langage de référence en matière d'informatique de gestion.

Langages de programmation

Algol, Cobol, Lisp

À partir de 1955 un grand nombre de langages furent proposés, souvent pour répondre à des faiblesses du Fortran.

- L'Algol est le fruit d'un travail universitaire visant à définir un langage «universel» de programmation. Peu utilisé, il a néanmoins posé les bases de l'algorithmique et a influencé de nombreux langages plus récents (Python parmi tant d'autres).
- Le langage Cobol a été créé pour répondre à un besoin en informatique de gestion, domaine dans lequel Fortran était peu adapté. Bien que présentant de nombreux défauts, il est néanmoins resté pendant longtemps un langage de référence en matière d'informatique de gestion.
- Lisp est le premier langage fonctionnel inspiré du lambda-calcul de Church. C'est l'ancêtre de tous les langages fonctionnels et en particulier de Caml. C'est aussi le premier langage interprété.

Langages de programmation

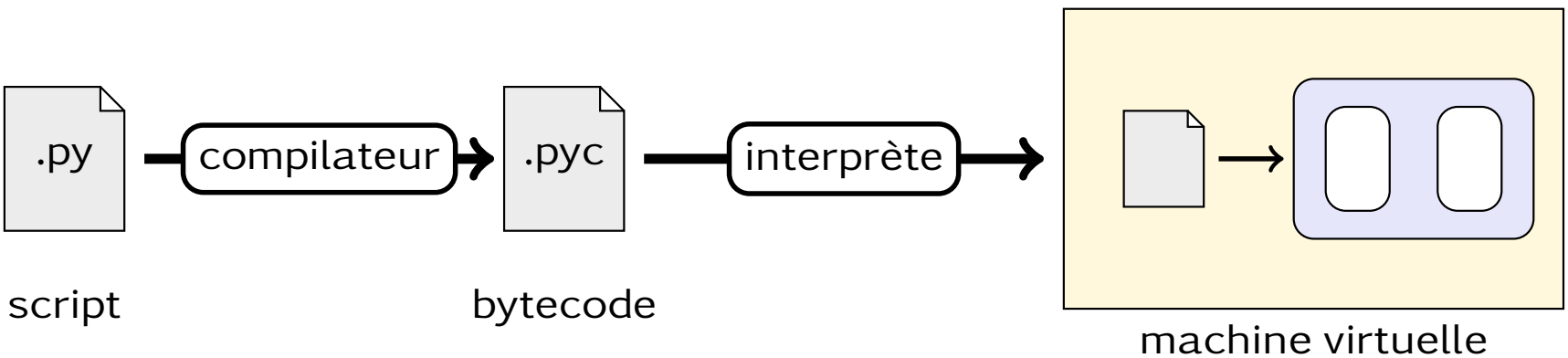
Python

Python est un langage de programmation de style majoritairement impératif, utilisé dans de nombreux domaines : développement web et internet, calcul numérique et scientifique, éducation.

Langages de programmation

Python

Python est un langage de programmation de style majoritairement impératif, utilisé dans de nombreux domaines : développement web et internet, calcul numérique et scientifique, éducation.
Il s'agit d'un langage interprété, à une nuance près : un script PYTHON est tout d'abord compilé en un langage intermédiaire (le *bytecode*) puis interprété par une machine virtuelle.



le bytecode reste indépendant de la machine mais est plus proche du langage machine ; c'est pourquoi les performances des interprètes de bytecode sont meilleures que celles des interprètes de langages de plus haut niveau.

Système d'exploitation

Le premier programme exécuté lors de la mise en marche est le système d'exploitation (Linux, Mac OS X, Windows pour les ordinateurs, Android, iOS, Windows phone pour les smartphones).

Système d'exploitation

Le premier programme exécuté lors de la mise en marche est le système d'exploitation (Linux, Mac OS X, Windows pour les ordinateurs, Android, iOS, Windows phone pour les smartphones).

Ses rôles principaux :

- servir d'interface entre les programmes et les périphériques d'entrées/sorties ;

Système d'exploitation

Le premier programme exécuté lors de la mise en marche est le système d'exploitation (Linux, Mac OS X, Windows pour les ordinateurs, Android, iOS, Windows phone pour les smartphones).

Ses rôles principaux :

- servir d'interface entre les programmes et les périphériques d'entrées/sorties ;
- gérer la mémoire de masse (disque dur) ;

Système d'exploitation

Le premier programme exécuté lors de la mise en marche est le système d'exploitation (Linux, Mac OS X, Windows pour les ordinateurs, Android, iOS, Windows phone pour les smartphones).

Ses rôles principaux :

- servir d'interface entre les programmes et les périphériques d'entrées/sorties ;
- gérer la mémoire de masse (disque dur) ;
- fournir à chaque utilisateur une machine virtuelle par l'intermédiaire de laquelle il pourra interagir avec la machine.

Système d'exploitation

Le premier programme exécuté lors de la mise en marche est le système d'exploitation (Linux, Mac OS X, Windows pour les ordinateurs, Android, iOS, Windows phone pour les smartphones).

Ses rôles principaux :

- servir d'interface entre les programmes et les périphériques d'entrées/sorties ;
- gérer la mémoire de masse (disque dur) ;
- fournir à chaque utilisateur une machine virtuelle par l'intermédiaire de laquelle il pourra interagir avec la machine.

L'utilisateur commande au système d'exploitation par l'intermédiaire d'une interface :

- textuelle (le *terminal* des systèmes Unix) ;
- ou graphique (représentation imagée des répertoires et fichiers, manipulation à l'aide d'un pointeur).

Système d'exploitation

Unix

1970 : naissance d'Unix, entièrement écrit en C (langage proche de la machine physique donc très bien adapté à l'écriture de systèmes d'exploitation).

Système d'exploitation

Unix

1970 : naissance d'Unix, entièrement écrit en C (langage proche de la machine physique donc très bien adapté à l'écriture de systèmes d'exploitation).
Unix est organisé autour d'un noyau de base qui sert de support à un interprète de langage de commande, le shell, que l'on utilise par l'intermédiaire d'une interface textuelle (le *terminal*) ou graphique.

Système d'exploitation

Unix

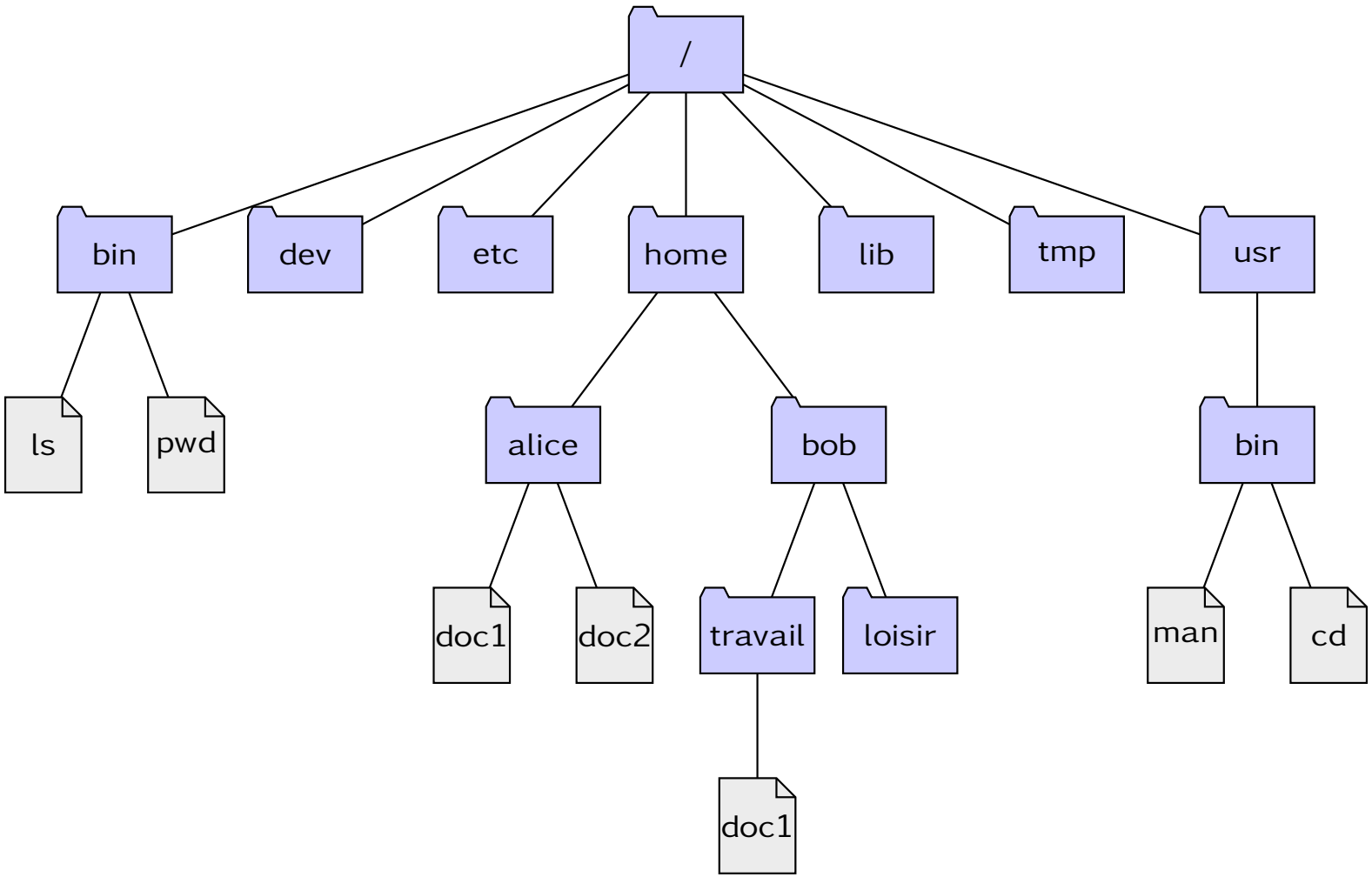
1970 : naissance d'Unix, entièrement écrit en C (langage proche de la machine physique donc très bien adapté à l'écriture de systèmes d'exploitation).
Unix est organisé autour d'un noyau de base qui sert de support à un interprète de langage de commande, le shell, que l'on utilise par l'intermédiaire d'une interface textuelle (le *terminal*) ou graphique.

Les systèmes d'exploitation sont multi-tâches et multi-utilisateurs : ils sont capables de partager des ressources selon une hiérarchie de droits d'accès.
Chaque utilisateur se voit attribuer un compte comportant :

- un identifiant associé à un mot de passe ;
- un répertoire d'accueil personnel destiné à héberger tous les sous-répertoires et fichiers qui lui appartiennent.

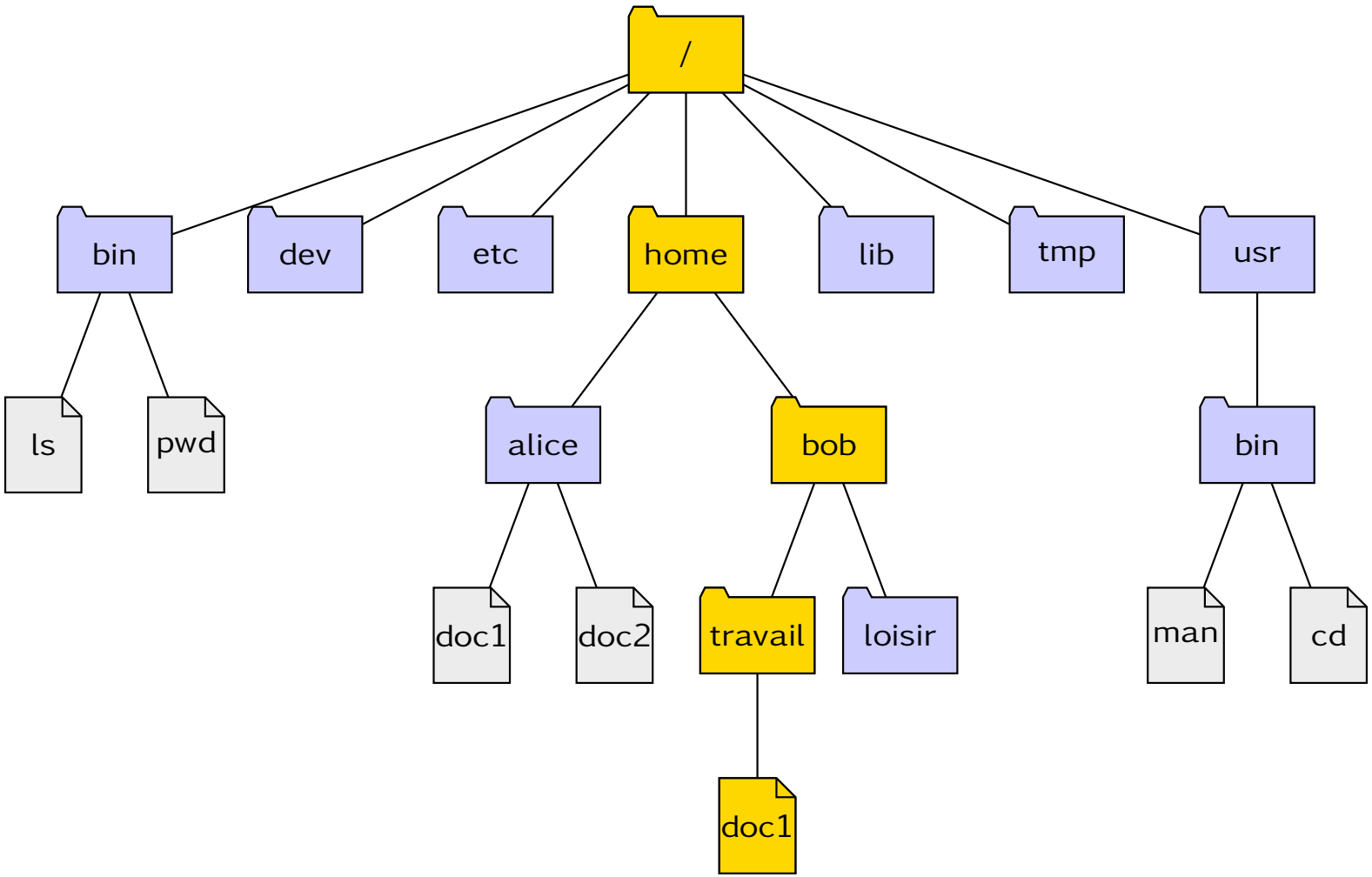
Hiérarchie des fichiers

La structure est arborescente : les bifurcations sont les *répertoires*, les *fichiers* se trouvent aux extrémités des branches.



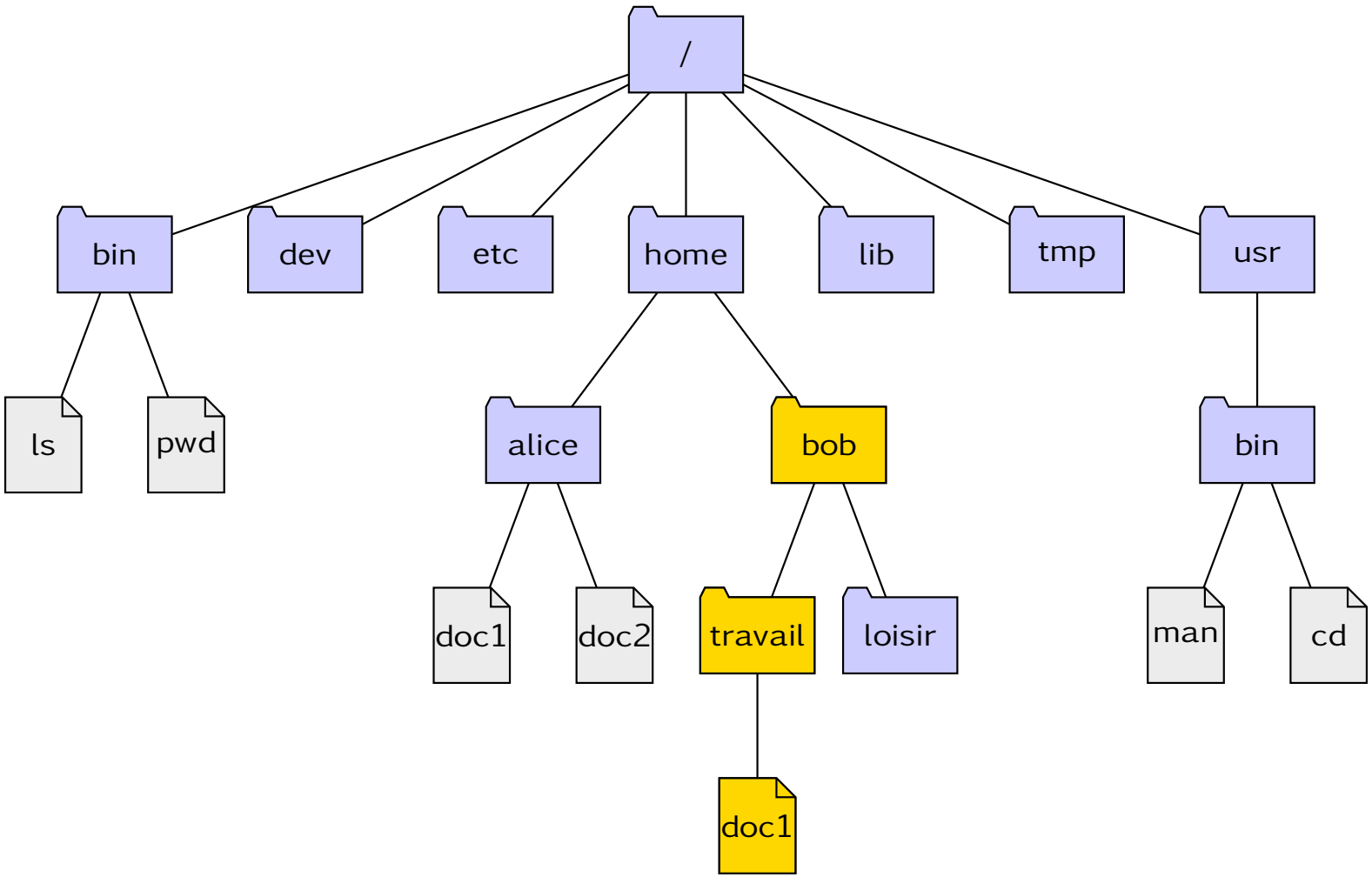
Hiérarchie des fichiers

Chaque élément est décrit par :
un chemin absolu `/home/bob/travail/doc1` ou relatif `~/travail/doc1`



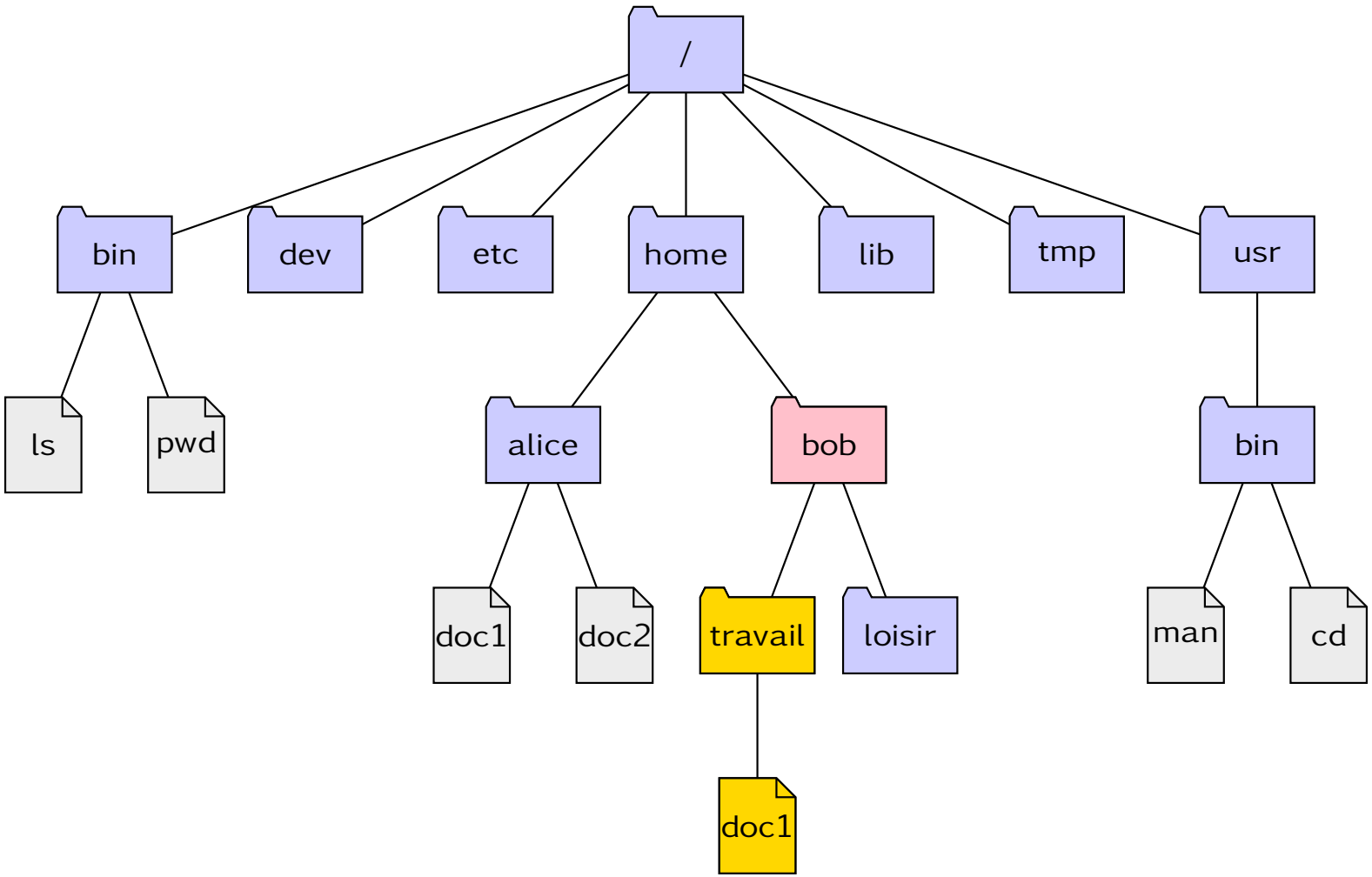
Hiérarchie des fichiers

Chaque élément est décrit par :
un chemin absolu `/home/bob/travail/doc1` ou relatif `~/travail/doc1`



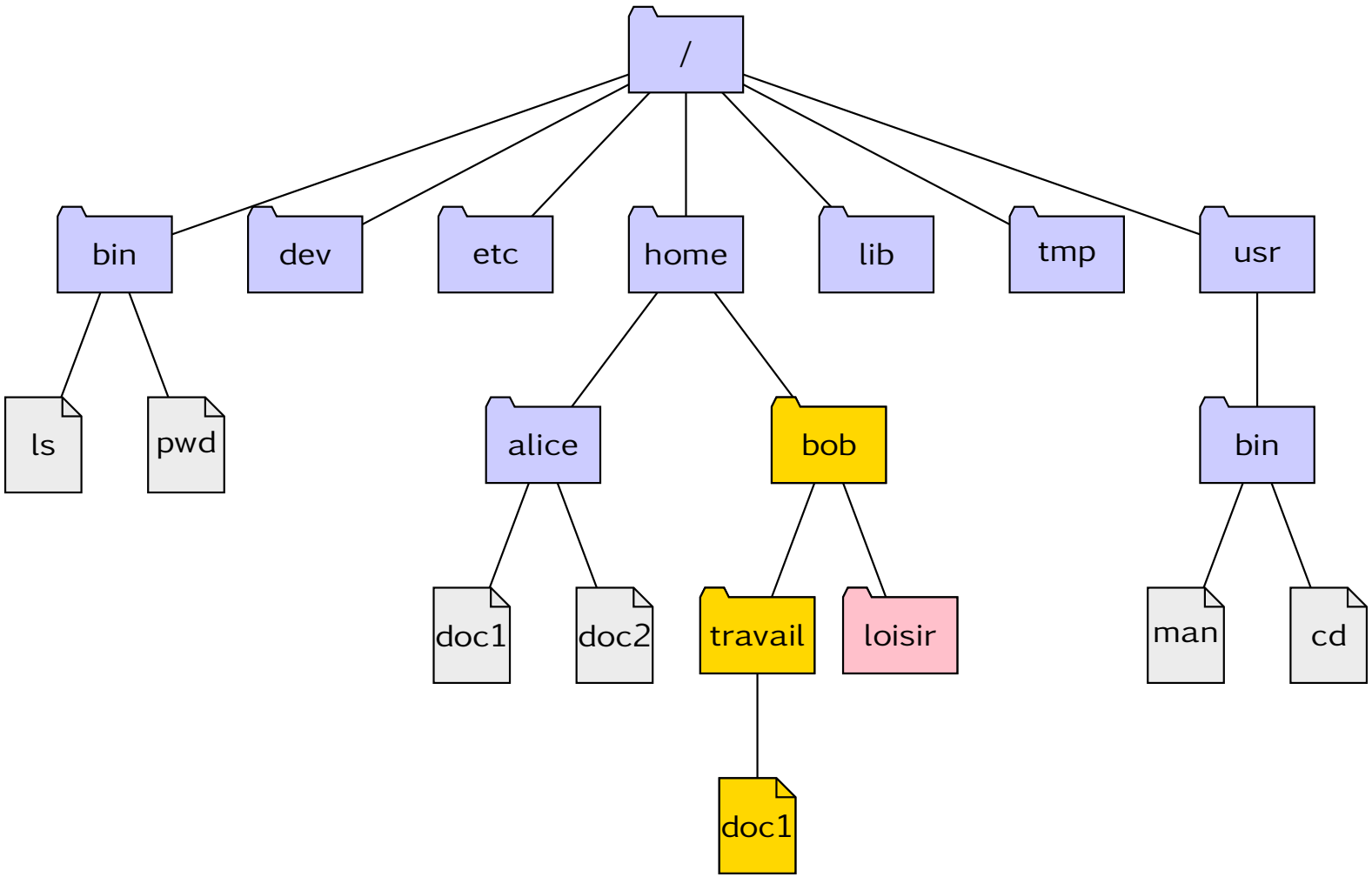
Hiérarchie des fichiers

Un élément peut aussi être décrit relativement au *répertoire courant* : `./travail/doc1` ou `travail/doc1` (si /home/bob est le répertoire courant).



Hiérarchie des fichiers

Si `/home/bob/loisir` devient le nouveau répertoire courant, le même élément est cette fois décrit par : `../travail/doc1`.



Quelques instructions UNIX

- `pwd` affiche le répertoire courant ;
- `ls` affiche le contenu du répertoire courant ;
- `cd` change le répertoire courant ;
- `mv` déplace (et renomme) un fichier ou un dossier ;
- `cp` copie (et renomme) un fichier ;
- `rm` supprime un fichier ;
- `mkdir` crée un nouveau répertoire (vide) ;
- `rmdir` supprime un répertoire vide.

Quelques instructions UNIX

- `pwd` affiche le répertoire courant ;
- `ls` affiche le contenu du répertoire courant ;
- `cd` change le répertoire courant ;
- `mv` déplace (et renomme) un fichier ou un dossier ;
- `cp` copie (et renomme) un fichier ;
- `rm` supprime un fichier ;
- `mkdir` crée un nouveau répertoire (vide) ;
- `rmdir` supprime un répertoire vide.

Exemple.
Alice crée un dossier `public` dans son répertoire courant et y déplace le fichier `doc2` en le renommant `doc3` :

```
mkdir ~/public
mv ~/doc2 ~/public/doc3
```

ou

```
mkdir ~/public
cd ~/public
mv ../doc2 /doc3
```

Droits d'accès

Chaque fichier possède un certain nombre d'attributs précisant sa nature et les droits d'accès qui sont attribués au propriétaire et aux autres utilisateurs :

- droit en lecture (r) : pour lire le fichier ;
- droit en écriture (w) : pour modifier le fichier ;
- droit d'exécution (x) : pour exécuter le programme.

Droits d'accès

Chaque fichier possède un certain nombre d'attributs précisant sa nature et les droits d'accès qui sont attribués au propriétaire et aux autres utilisateurs :

- droit en lecture (`r`) : pour lire le fichier ;
- droit en écriture (`w`) : pour modifier le fichier ;
- droit d'exécution (`x`) : pour exécuter le programme.

Les répertoires possèdent des droits analogues :

- `r` : pour afficher son contenu (fichiers et sous-répertoires) ;
- `w` : pour modifier son contenu ;
- `x` : pour y accéder.

Droits d'accès

Chaque fichier possède un certain nombre d'attributs précisant sa nature et les droits d'accès qui sont attribués au propriétaire et aux autres utilisateurs :

- droit en lecture (`r`) : pour lire le fichier ;
- droit en écriture (`w`) : pour modifier le fichier ;
- droit d'exécution (`x`) : pour exécuter le programme.

Les répertoires possèdent des droits analogues :

- `r` : pour afficher son contenu (fichiers et sous-répertoires) ;
- `w` : pour modifier son contenu ;
- `x` : pour y accéder.

Seuls l'utilisateur *root* (l'administrateur du système) et le propriétaire d'un fichier/répertoire peuvent modifier les droits d'accès.

Droits d'accès

Il y a trois classes d'utilisateurs : `user` (le propriétaire), `group` (le groupe auquel il appartient), `other` (tous les autres).
Exemple. Un fichier dont le droit d'accès est `rwxr-xr--` est accessible :

- en lecture/écriture/exécution (`rwx`) pour le propriétaire ;
- en lecture/exécution pour son groupe (`r-x`) ;
- en lecture seule pour les autres (`r--`).

Droits d'accès

Il y a trois classes d'utilisateurs : `user` (le propriétaire), `group` (le groupe auquel il appartient), `other` (tous les autres).

Exemple. Un fichier dont le droit d'accès est `rwxr-xr--` est accessible :

- en lecture/écriture/exécution (`rwx`) pour le propriétaire ;
- en lecture/exécution pour son groupe (`r-x`) ;
- en lecture seule pour les autres (`r--`).

$\texttt{rwx} = (111)_2 = 7, \qquad \texttt{r-x} = (101)_2 = 5, \qquad \texttt{r--} = (100)_2 = 4.$

Pour attribuer ces droits au fichier `doc1`, Alice écrit :

```
chmod 754 ~/doc1
```

Droits d'accès

Il y a trois classes d'utilisateurs : `user` (le propriétaire), `group` (le groupe auquel il appartient), `other` (tous les autres).
Exemple. Un fichier dont le droit d'accès est `rwxr-xr--` est accessible :

- en lecture/écriture/exécution (`rwx`) pour le propriétaire ;
- en lecture/exécution pour son groupe (`r-x`) ;
- en lecture seule pour les autres (`r--`).

$\texttt{rwx} = (111)_2 = 7, \qquad \texttt{r-x} = (101)_2 = 5, \qquad \texttt{r--} = (100)_2 = 4.$
Pour attribuer ces droits au fichier `doc1`, Alice écrit :

```
chmod 754 ~/doc1
```

Pour autoriser Bob à déposer des fichiers dans son répertoire `public` sans voir le contenu :

```
chmod 733 ~/public
```

$3 = (011)_2 = \texttt{-wx}.$