

A. Notions de base sur les systèmes d'exploitation

Mise en oeuvre de la protection/isolation : notion d'espace d'adressage, de modes d'exécution user/superviseur, introduction des appels système.

Sommaire

- Rappels d'architecture des machines
- Introduction aux systèmes d'exploitation multiprogrammés
- Fonctions d'un système d'exploitation
- Notions de base

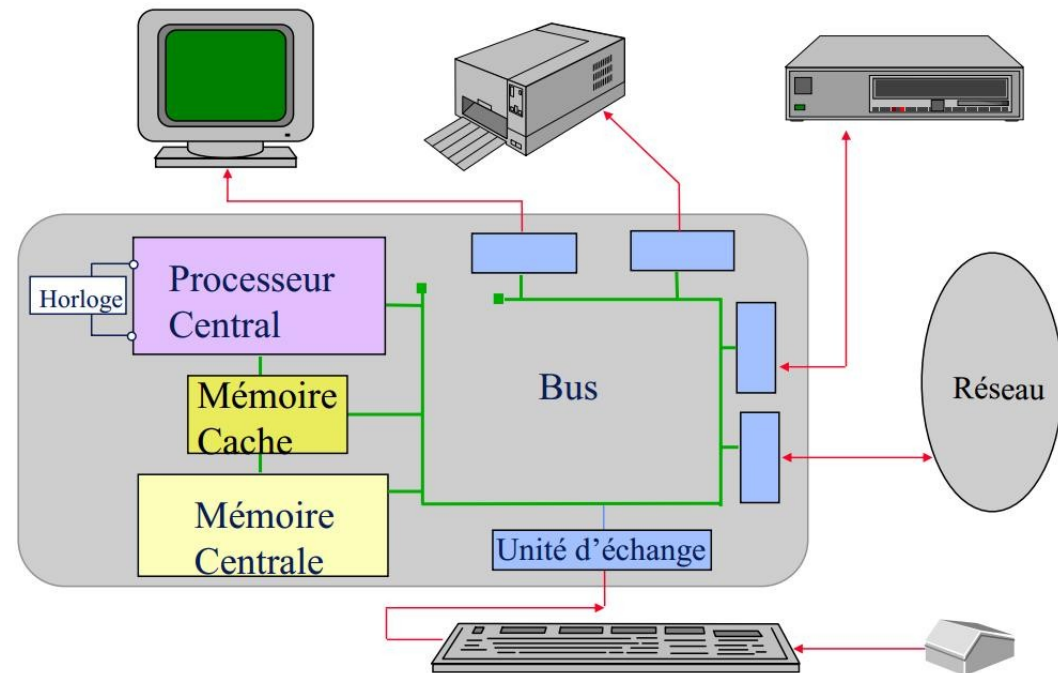
Introduction

- Le **système d'exploitation** est un **ensemble** de **programmes** qui **réalise l'interface** entre le **matériel** de **l'ordinateur** et les **utilisateurs**, d'une part afin de **construire au-dessus du matériel** une **machine virtuelle** plus **facile** d'emploi et plus **conviviale**.
- Prendre en charge la **gestion** des **ressources** de la machine et le **partage** de celles-ci.
- Dans ce **chapitre** :
 - nous allons **définir** plus précisément les **rôles** d'un **système d'exploitation** dans un **environnement multiprogrammé** et les différentes **fonctions** qui composent ce système d'exploitation.
 - Nous présentons également les **différents types** de **systèmes d'exploitation**
 - Enfin, les **notions** de **base** sur lesquelles **repose** le **fonctionnement** du **système d'exploitation** sont décrites :
 - Appels système
 - Modes d'exécutions
 - Notion d'interruptions

Rappels d'architecture des machines

Structure générale de la machine physique

- Le **processeur** est chargé d'exécuter les instructions des programmes placés en mémoire centrale. Le processeur est cadencé par une horloge.
- La **mémoire centrale** contient les instructions et données des programmes à exécuter. Avec le disque, la mémoire cache, elle constitue un **système de hiérarchie de mémoire** qui permet de rapprocher la vitesse de la mémoire centrale de celle du processeur.
- Les **unités d'échanges** (UE) réalisent l'interface entre les périphériques et le processeur.
- Tous les composants de la machine communiquent par l'intermédiaire du **bus**.

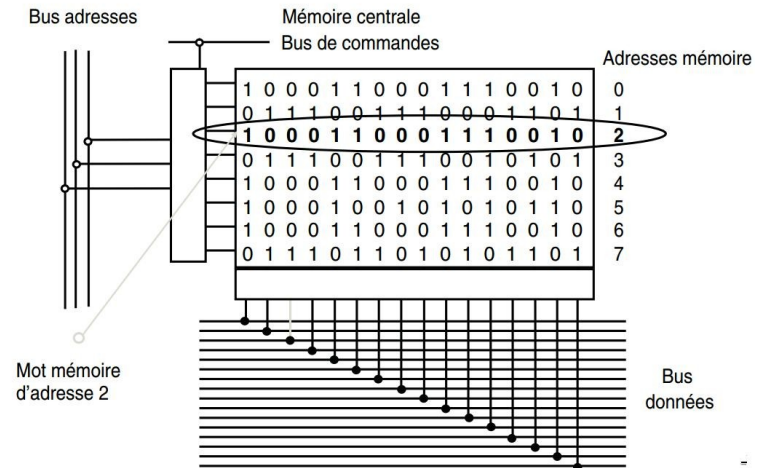


Structure matérielle générale

Rappels d'architecture des machines

La mémoire centrale

- La mémoire centrale assure la **fonction** de **stockage** de **l'information** qui peut être **manipulée** par le **microprocesseur** (processeur central)
- Toute **information** stockée en **mémoire** centrale est **représentée** sous la forme d'une suite de **digits binaires**. La **figure** présente l'**organisation** générale d'une **mémoire** centrale.
- La **mémoire** est **découpée** en cellules mémoires : les **mots mémoires**.
- On peut ainsi trouver des mots de **1 bit**, **4 bits** (quartet) ou encore **8 bits** (octet ou byte), **16 bits** voire **32** ou **64 bits**.
- Chaque **mot** est **repéré** dans la mémoire par une **adresse**, un numéro qui identifie le mot mémoire.
- Ainsi un **mot** est un **contenant accessible** par son **adresse**.

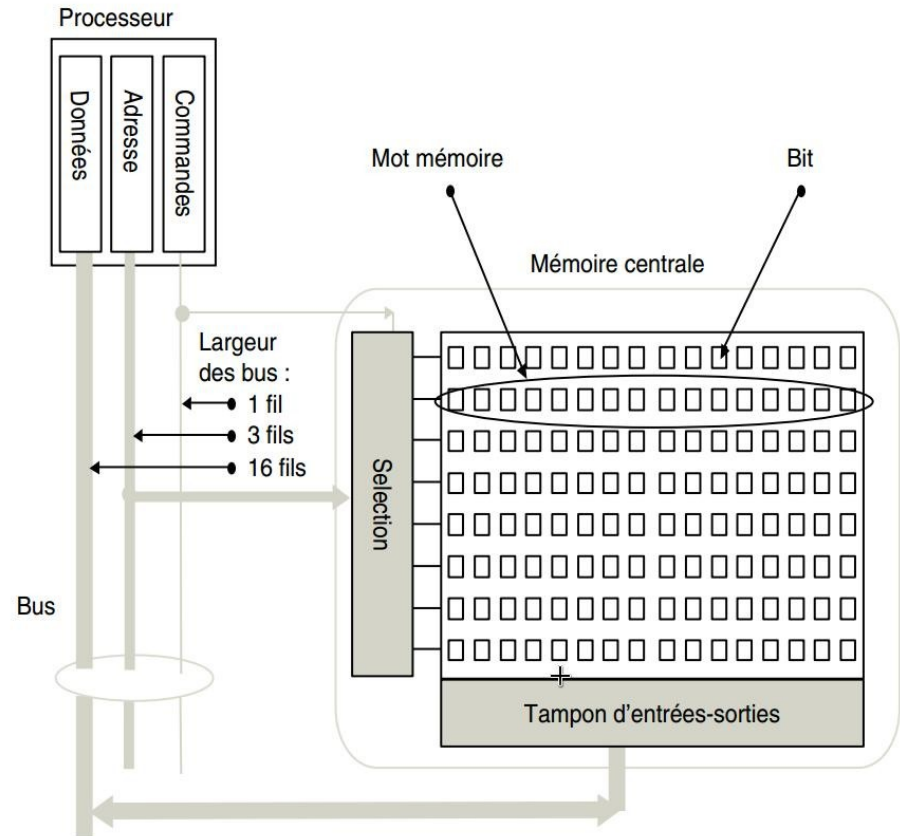


La mémoire centrale (note 1).

Rappels d'architecture des machines

Le bus de communication (note 2)

- Le bus de communication peut se **représenter** comme une **nappe de fils** **transportant des signaux** et **permettant l'échange des informations** entre les **différents modules du processeur**.
- Le **nombre de fils** du bus **détermine** sa **largeur** et définit ainsi le nombre d'informations différentes que peut véhiculer le bus.
- Le **bus** est construit comme un **ensemble de trois bus** :
 - le **bus d'adresses** transporte des combinaisons de signaux qui sont interprétées comme des nombres entiers représentant **l'adresse** d'un **mot mémoire**.
 - le **bus de données** permet **l'échange des informations** (les contenus) entre les **différents modules**.
 - le **bus de commandes** : c'est par ce bus que le **microprocesseur indique** la **nature** des **opérations** qu'il veut **effectuer**.

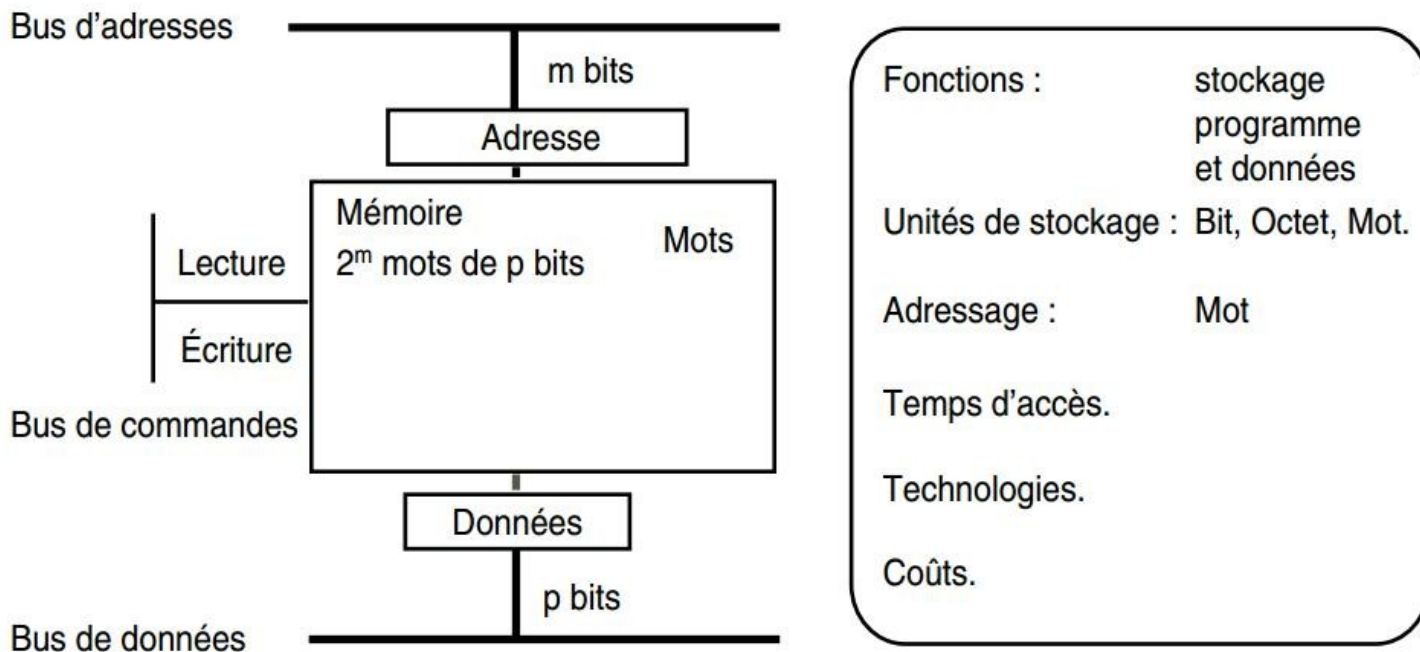


Bus de communication.

Rappels d'architecture des machines

Le bus de communication

- La **figure résume** les différents **points abordés** concernant la **mémoire** et les **bus** permettant la **communication** avec la mémoire.

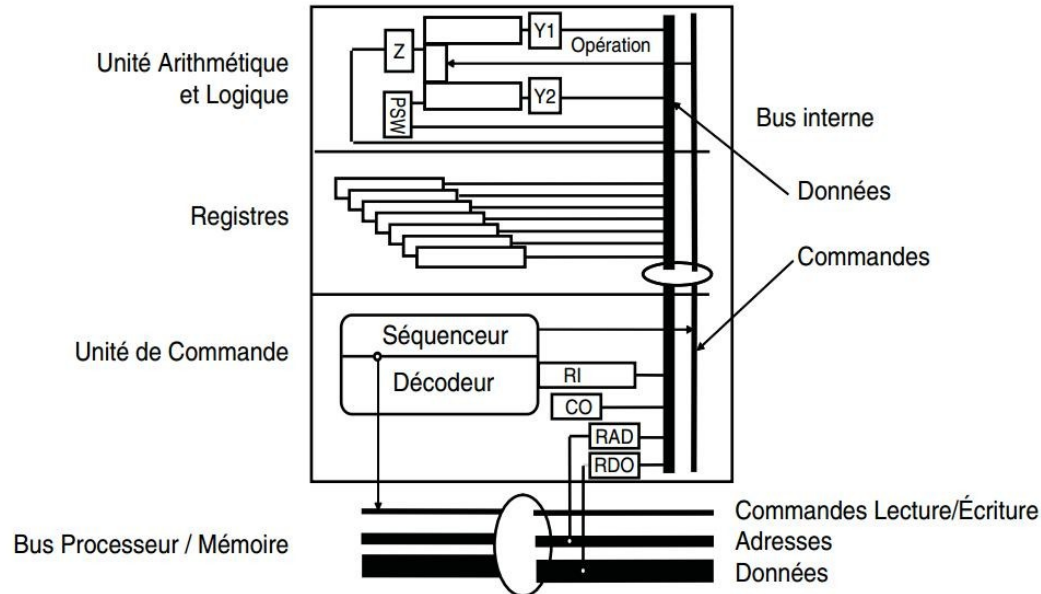


Bus de communication et mémoire centrale.

Rappels d'architecture des machines

Le processeur central ou microprocesseur

- Le **microprocesseur** (unité centrale) a pour objet **d'exécuter les instructions machines placées en mémoire centrale**. La figure présente son architecture générale.
- Il est **constitué de quatre parties** : l'unité arithmétique et logique (**UAL**), les **registres**, l'**unité commande** et le **bus de communication interne** permettant l'échange des données et des commandes entre les différentes parties du microprocesseur.



**Le
microprocesseur.**

Rappels d'architecture des machines

Le processeur central ou microprocesseur

- **Les registres** : ce sont des **zones de mémorisation** de l'information **internes** au **microprocesseur**. Ils sont de **faible capacité** et de **temps d'accès très faible**.
- **L'unité arithmétique et logique (UAL)** : ce module est chargé de l'**exécution** de tous les **calculs** que peut réaliser le **microprocesseur**. Dans ce module se trouvent également des **registres** dont l'objet est de **contenir** les **données sur lesquelles** vont **porter** les **opérations** à effectuer. Le registre particulier, le **PSW** (Program Status Word), qui joue un **rôle fondamental** de **contrôle** de l'**exécution** d'un **programme** et qui à tout instant **donne** des informations importantes sur l'**état** de notre **microprocesseur**. Par **exemple**, si une opération provoque un **dépassement de capacité**.
- **L'unité de commande** : Elle **exécute** les **instructions** machines et pour cela **utilise** les **registres** et l'**UAL** du microprocesseur.
- **Le compteur ordinal CO** : C'est un **registre d'adresses**. À chaque instant il **contient** l'**adresse** de la **prochaine instruction à exécuter**.
- **Le registre d'instruction RI** : C'est un **registre de données**. Il **contient** l'**instruction à exécuter**.
- **Le décodeur** : Il s'agit d'un **ensemble de circuits** dont la **fonction** est d'**identifier** l'**instruction à exécuter** qui se **trouve** dans le registre **RI**, puis d'**indiquer** au **séquenceur** la **nature** de cette **instruction** afin que ce dernier puisse **déterminer** la **séquence des actions à réaliser**.

Rappels d'architecture des machines

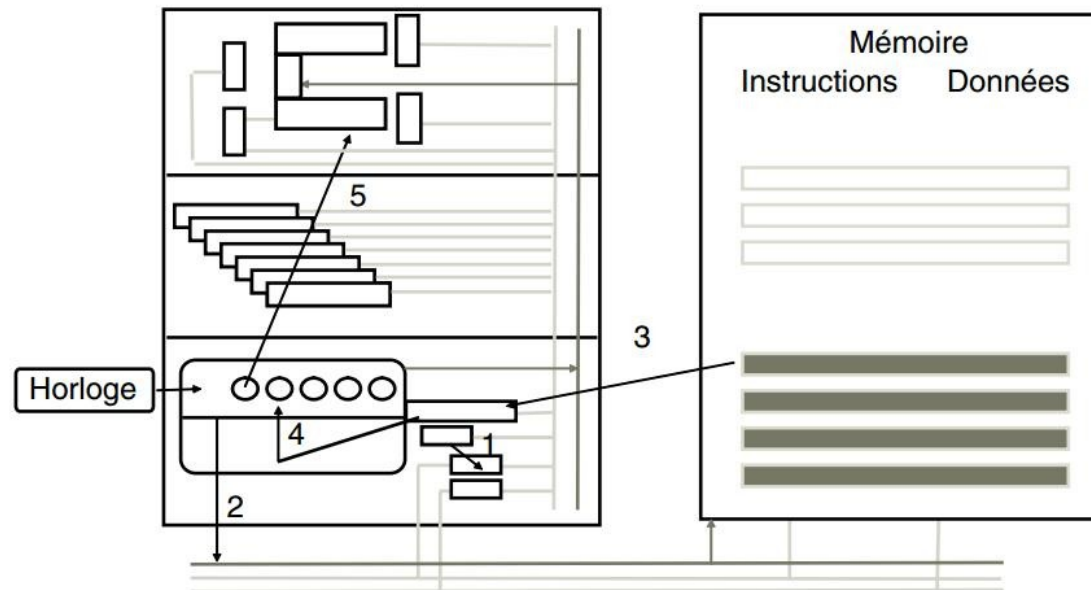
Le processeur central ou microprocesseur

- **Le séquenceur** : Il s'agit d'un **ensemble** de **circuits** permettant l'**exécution** effective de l'**instruction** placée dans le registre **RI**. Le séquenceur **exécute**, **rythmé** par l'**horloge** du **microprocesseur**, une **séquence** de **microcommandes** (micro-instructions) **réalisant** le **travail associé** à cette **instruction** machine. Pour son fonctionnement le séquenceur **utilise** les **registres** et l'**UAL**. Ainsi l'**exécution** effective d'une **instruction** machine se **traduit** par l'**exécution** d'une **séquence** de **micro-instructions** exécutables par les circuits de base du microprocesseur.
- **Le registre RAD** : C'est un **registre d'adresses**. Il est **connecté** au **bus d'adresses** et **permet** la **sélection** d'un **mot mémoire** via le circuit de sélection.
- **Le registre RDO** : C'est un **registre de données**. Il **permet l'échange d'informations** (contenu d'un mot mémoire) **entre** la **mémoire centrale** et le **processeur** (registre). (note 1)

Rappels d'architecture des machines

FONCTIONNEMENT : RELATION MICROPROCESSEUR / MÉMOIRE CENTRALE

- L'objet de cette partie est de **présenter** succinctement comment **s'exécute** un **programme** machine sur le matériel que nous venons de définir.
- Pour être exécutable une **instruction** doit **nécessairement** être présente en **mémoire** centrale.
- La **mémoire** centrale **contient** donc des **instructions** et des **données**.
- Dans la mémoire centrale les instructions et les données sont **séparées** et **occupent** des **espaces** mémoires **différents**.
- À la **fin** du **chargement** du **programme** machine et des **données** en **mémoire** le compteur ordinal **CO** reçoit l'**adresse** de la **première instruction** du programme à exécuter.
- L'exécution peut alors commencer. Le principe général d'exécution est illustré dans la figure. (note 1)



Exécution d'une instruction.

Rappels d'architecture des machines

Les unités d'échanges

- Les unités d'échanges **permettent** la **communication entre** les modules du **processeur** et les **périphériques**. Cette fonction de communication est complexe.
- Une unité d'échange a une **double nature** :
 - elle est en **communication**, via le **bus interne** du **processeur**, avec la **mémoire centrale** et le **microprocesseur**. Des **instructions machines spécifiques** **permettent** les **échanges** entre **mémoire centrale** et **unité d'échange**. On trouve des instructions **d'écriture** permettant au **microprocesseur** de **placer** des **informations** dans l'**unité d'échange** et des **instructions de lecture** permettant au **microprocesseur d'acquérir** des **informations** à partir des **unités d'échanges**. Dans nos ordinateurs les **unités d'échanges** ne **communiquent pas directement** avec le **bus interne** du processeur **mais** au **travers** de **bus d'extension** (USB, FireWire, Fiberchannel, **PCI...**).
 - elle est en **communication** avec les **périphériques** et à ce titre doit être capable de les **piloter**.

Rappels d'architecture des machines

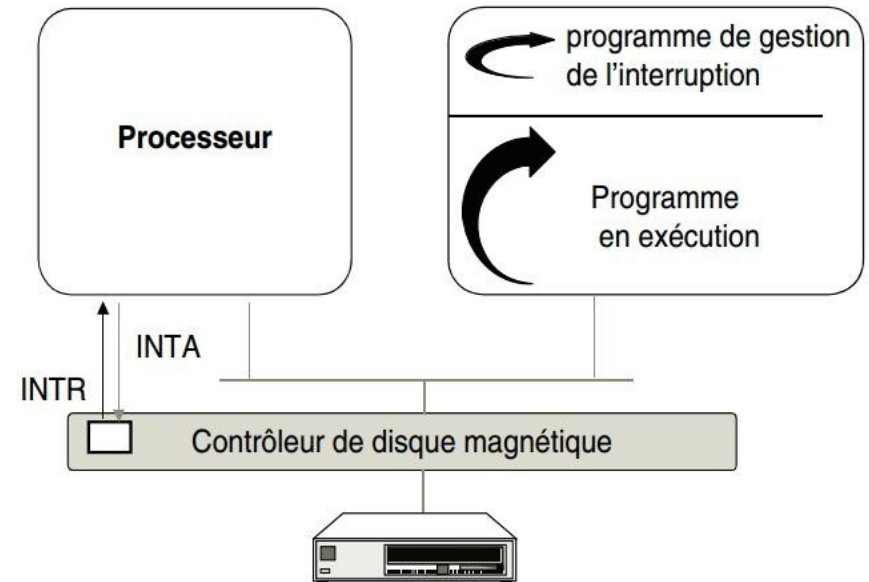
Notion d'interruptions

- Une **interruption** est un mécanisme permettant de **stopper l'exécution** du **programme en cours** afin d'aller **exécuter** une **tâche** jugée plus **prioritaire**.
- Elle est **caractérisée** par un **numéro** et un **traitement associé** (la routine ou traitant d'interruption)
- On distingue principalement **deux types d'événements** :
 - d'un **périphérique**. On parle alors d'**interruptions externes** qui **permettent** à un **périphérique** de se **manifester** auprès du **processeur** ;
 - d'un **programme** en cours d'exécution. Il s'agit d'**interruptions logicielles internes** souvent nommées **appels systèmes**. Il s'agit de **permettre** à un **programme** en cours d'**exécution** de se **dérouter vers** un **programme** du **système d'exploitation** qui doit **gérer** une **tâche particulière** (instruction I/O) ;
 - du **processeur** lui-même pour **traiter** des **événements exceptionnels** de type **division par zéro**, **dépassement de capacité** lors d'une opération arithmétique, on parle de **trappes**.
- La prise en compte et le traitement d'une interruption s'appuient sur un mécanisme relativement complexe.

Rappels d'architecture des machines

Notion d'interruptions

- Afin d'**illustrer** ce mécanisme nous prenons l'**exemple** du **traitement** d'une **interruption** externe, donc **produite** par un **périphérique**. **(note 1)**
- Pour **prendre en compte** une **interruption** et la **traiter** il faut en **déterminer** son **origine** car il y a plusieurs sources possibles d'interruptions, puis **exécuter** le **programme adapté**.
- Ce **mécanisme s'appuie** pour **partie** sur le **matériel** (un périphérique positionne un signal indiquant qu'il veut alerter le processeur) et pour **partie** sur du **logiciel** de traitement de l'interruption.
- La **figure représente** la **prise en compte** d'un **événement externe** provoquant une interruption.



Prise en compte d'une interruption externe.

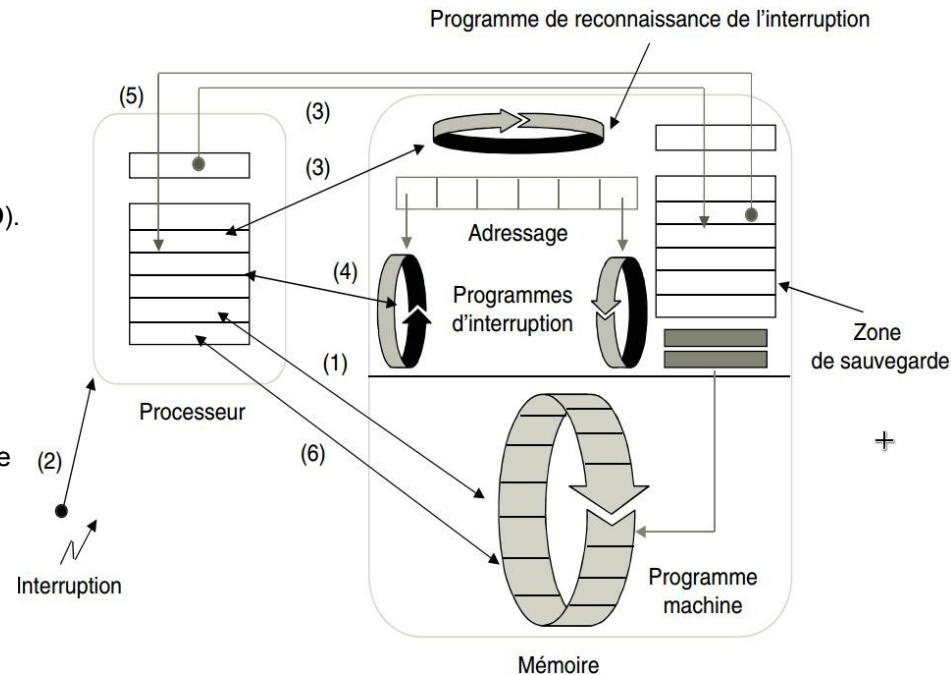
Rappels d'architecture des machines

Notion d'interruptions

- La **figure** résume le mécanisme. On **trouve** :
 - les **programmes de gestion des interruptions** (reconnaissance et traitement)
 - et une **zone mémoire**, appelée **vecteur d'interruptions**, contenant les **adresses mémoires** des **programmes de traitement d'une interruption**.
 - Lorsque le **périphérique** signale une interruption il **dépose** sur le **bus** le **numéro de l'interruption**, ce **numéro identifie** un point d'entrée dans le **vecteur d'interruptions** donc l'**adresse du programme de traitement**.

La **prise en compte** d'une **interruption** se fait selon la **séquence** :

1. le **programme utilisateur dispose** du **processeur** (registres).
2. une **interruption** est **postée** par un périphérique ;
3. il y a **sauvegarde** du **contexte** matériel d'exécution (registres, dont le **CO**).
Le **programme de reconnaissance s'exécute** et lit le **numéro de l'interruption**. Le numéro de l'interruption **permet l'identification** de l'**adresse du programme de traitement** ;
4. le compteur ordinal **CO** est **chargé** avec l'**adresse du programme de traitement** et celui-ci **s'exécute** ;
5. le **contexte d'exécution** du **programme utilisateur** est **rechargé** dans le **processeur** à la **fin** de l'**exécution** du **programme d'interruption** ;
6. le **programme utilisateur reprend** son **exécution**.



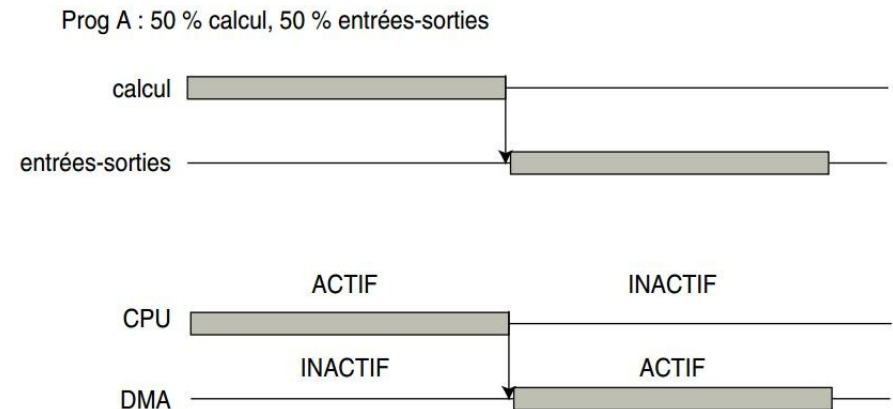
Prise en compte d'une interruption.

Introduction aux systèmes d'exploitation multiprogrammés

Rôle et définition d'un système d'exploitation multiprogrammé

- Analysons le temps d'utilisation du processeur pour une machine monoprogrammée et une machine multiprogrammée :

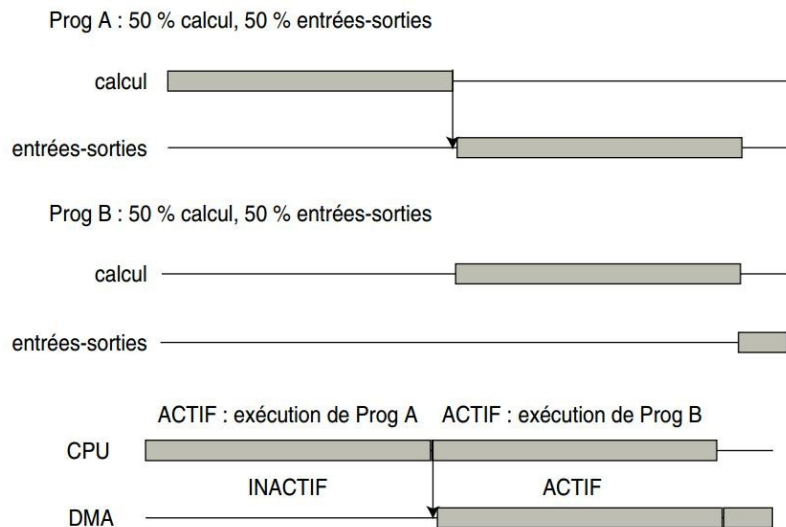
Dans le cadre d'une **machine monoprogrammée**, c'est-à-dire une machine pour laquelle un **seul programme utilisateur** est présent en **mémoire** centrale, cela veut dire que le **processeur reste inactif à chaque** fois que ce programme utilisateur réalise une **opération d'entrées-sorties**. Une telle inaction, illustrée sur le chronogramme de la **figure**, n'est **pas souhaitable**.



Chronogramme d'activité pour le processeur et le DMA, pour une machine monoprogrammée.

Pour **remédier** à cette inaction, la **machine** doit devenir **multiprogrammée**, c'est-à-dire qu'au moins un **autre programme utilisateur** doit être **placé en mémoire** centrale, qui sera **exécuté** pendant l'**opération d'entrées-sorties** du premier.

L'**exemple** de la **figure** est un exemple **théorique** et idéalisé. Dans la **réalité**, pour parvenir à **occuper** à près de **100 % le processeur**, il faut **placer en mémoire** centrale un très **grand nombre de programmes**.



Chronogramme d'activité pour le processeur et le DMA, pour une machine multiprogrammée.

Introduction aux systèmes d'exploitation multiprogrammés

Rôle et définition d'un système d'exploitation multiprogrammé

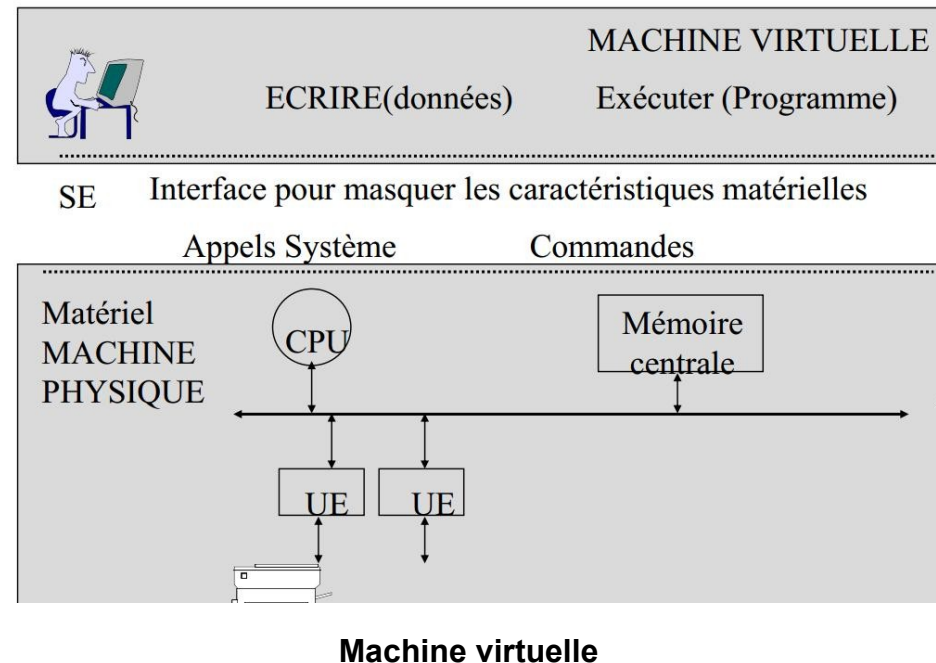
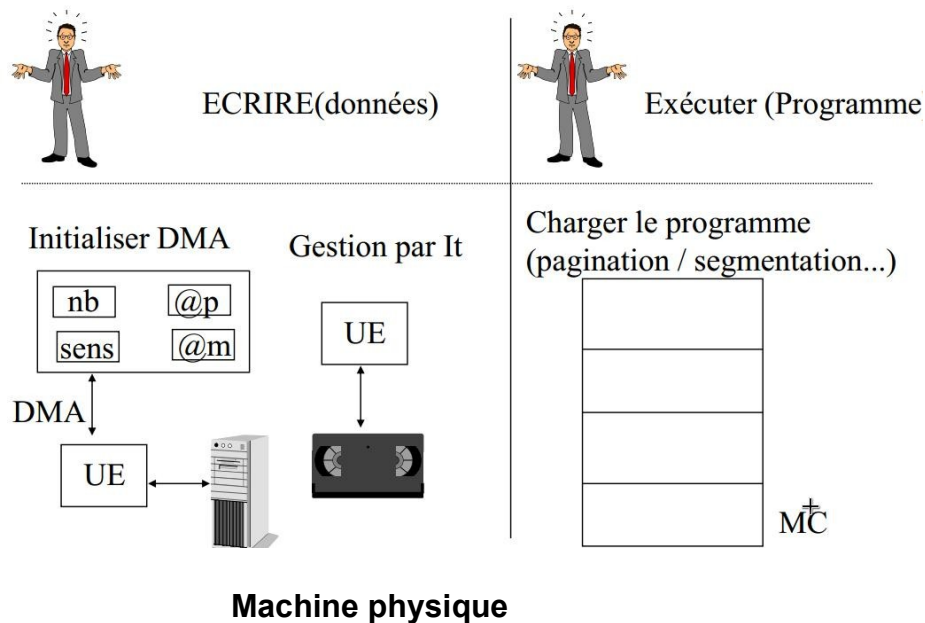
Un premier rôle : assurer le partage de la machine physique

- **Gérer le partage** de la **machine physique** et des **ressources matérielles** entre les **différents programmes**.
- **Assurer l'équité d'accès** aux **ressources** matérielles.
- **Assurer** également que les **accès** des programmes à ces **ressources s'effectuent correctement**, c'est-à-dire que les **opérations réalisées** par les **programmes** sont **licites** pour la cohérence des ressources : on parle alors de **protection des ressources**.
- Le **partage** des **ressources** va **concerner** principalement le **processeur**, la **mémoire** centrale et les **périphériques** d'entrées-sorties. Plus précisément, les **questions** suivantes vont devoir être résolues :
 - dans le cadre du **partage** du **processeur** : parmi tous les **programmes** chargés en **mémoire** centrale, **lequel** doit **s'exécuter** ?
 - dans le cadre du **partage** de la **mémoire** centrale : comment **allouer** la mémoire centrale aux différents **programmes** ? Comment **disposer** d'une **quantité suffisante** de **mémoire** pour y placer tous les programmes nécessaires à un **bon taux d'utilisation** du **processeur** ? Comment assurer la **protection** entre ces différents **programmes utilisateurs** ? Par protection, on entend ici **veiller** à ce qu'un **programme** donné **n'accède pas** à une **plage mémoire** allouée à un **autre programme** ;
 - dans le cadre du **partage** des **périphériques** : dans quel **ordre traiter** les **requêtes d'entrées-sorties** pour optimiser les transferts ?

Introduction aux systèmes d'exploitation multiprogrammés

Rôle et définition d'un système d'exploitation multiprogrammé

Un second rôle : rendre conviviale la machine physique (note 1)

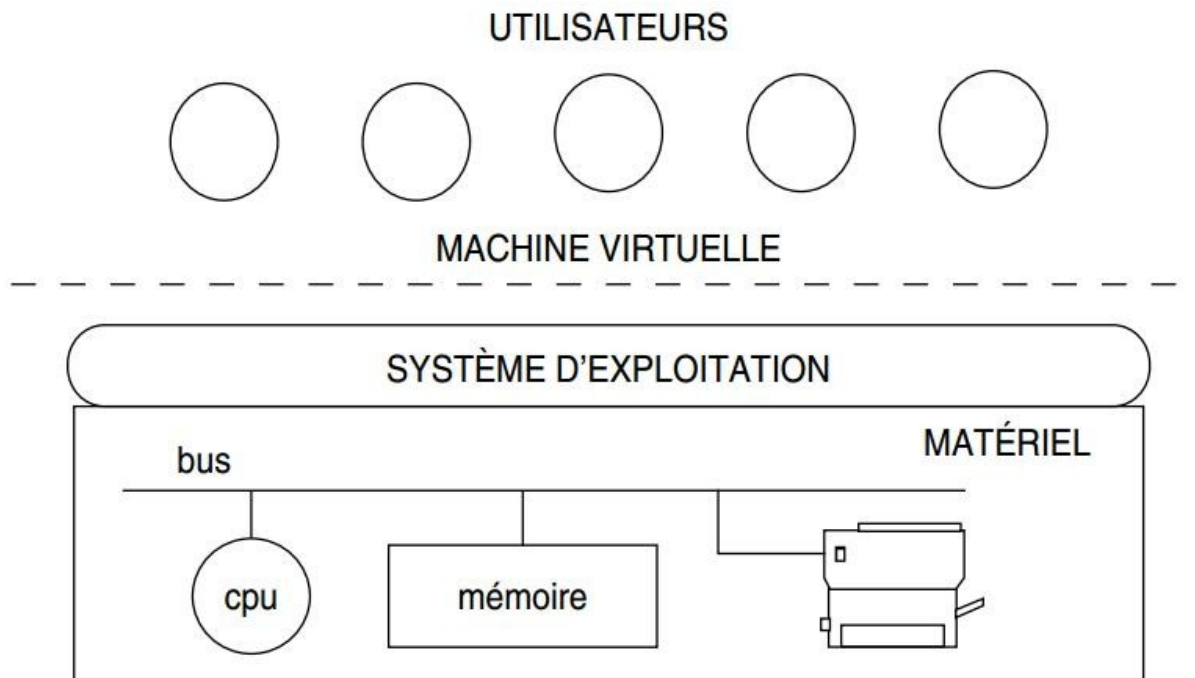


Introduction aux systèmes d'exploitation multiprogrammés

Rôle et définition d'un système d'exploitation multiprogrammé

Définition du système d'exploitation multiprogrammé

- Le **système d'exploitation** est donc un **ensemble de programmes** qui **réalise l'interface** entre le **matériel** de l'**ordinateur** et les **utilisateurs**. Il a **deux objectifs** principaux (figure) :
 - **construction** au-dessus du matériel d'une **machine virtuelle plus facile d'emploi** et plus **conviviale**;
 - **prise en charge** de la **gestion** de plus en plus complexe des **ressources** et **partage** de celles-ci.



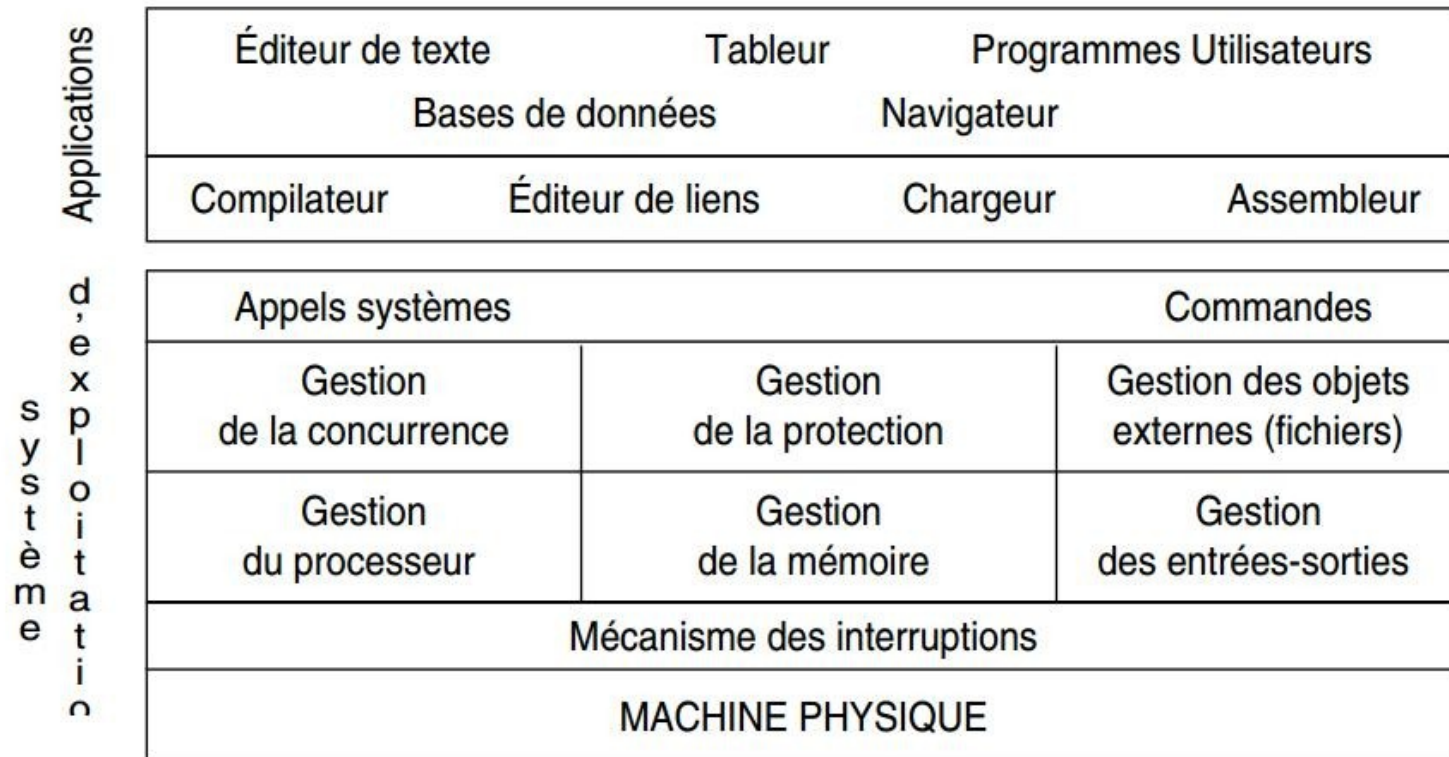
**Place du système
d'exploitation.**

Introduction aux systèmes d'exploitation multiprogrammés

Structure d'un système d'exploitation multiprogrammé

Composants d'un système d'exploitation

- Le système d'exploitation réalise donc une couche logicielle placée entre la machine matérielle et les applications.
- Le système d'exploitation peut être découpé en plusieurs grandes fonctionnalités présentées sur la figure.



Fonctionnalités du système d'exploitation.

Introduction aux systèmes d'exploitation multiprogrammés

Structure d'un système d'exploitation multiprogrammé

Composants d'un système d'exploitation (suite)

- **La fonctionnalité de gestion du processeur** : le système doit **gérer l'allocation** du **processeur** aux **différents programmes** pouvant s'exécuter. Cette allocation **se fait** par le **biais** d'un **algorithme d'ordonnancement** qui planifie l'exécution des programmes. Selon le type de système d'exploitation, l'algorithme d'ordonnancement répond à des objectifs différents.
- **La fonctionnalité de gestion de la mémoire** : Le système doit **gérer l'allocation** de la **mémoire** centrale entre les **différents programmes** pouvant s'exécuter, c'est-à-dire qu'il doit **trouver** une **place libre suffisante** en **mémoire** centrale pour que le **chargeur** puisse y **placer** un **programme** à exécuter. Comme la **mémoire physique** est **souvent** trop **petite** pour **contenir** la **totalité** des **programmes**, la gestion de la mémoire se fait selon le principe de la **mémoire virtuelle** : à un instant donné, **seules** sont **chargées** en mémoire centrale, les **parties** de **code** et **données utiles** à l'exécution.
- **La fonctionnalité de gestion des entrées-sorties** : Le système doit **gérer l'accès** aux **périphériques**, c'est-à-dire faire la **liaison** entre les **appels** de **haut niveau** des **programmes utilisateurs** (exemple **getchar()**) et les **opérations** de **bas niveau** de l'**unité d'échange** responsable du périphérique (unité d'échange **clavier**) : c'est le **pilote** d'entrées-sorties (driver) qui **assure** cette **correspondance**.
- **La fonctionnalité de gestion des objets externes** : La **gestion** de l'**allocation** des **mémoires** de **masse** ainsi que l'**accès** aux **données** stockées s'appuient sur la **notion** de fichiers et de système de gestion de fichiers (**SGF**).
- **La fonctionnalité de gestion de la concurrence** : Le système offre des **outils** de **communication** et de **synchronisation** entre **programmes**.
- **La fonctionnalité de gestion de la protection** : Le système doit **fournir** des **mécanismes** garantissant que ses **ressources** (processeur, mémoire, fichiers) ne peuvent être **utilisées** que par les **programmes** auxquels les **droits nécessaires** ont été **accordés**. Il faut notamment **protéger** le **système** et la machine des **programmes utilisateurs** (mode d'exécution utilisateur et superviseur).

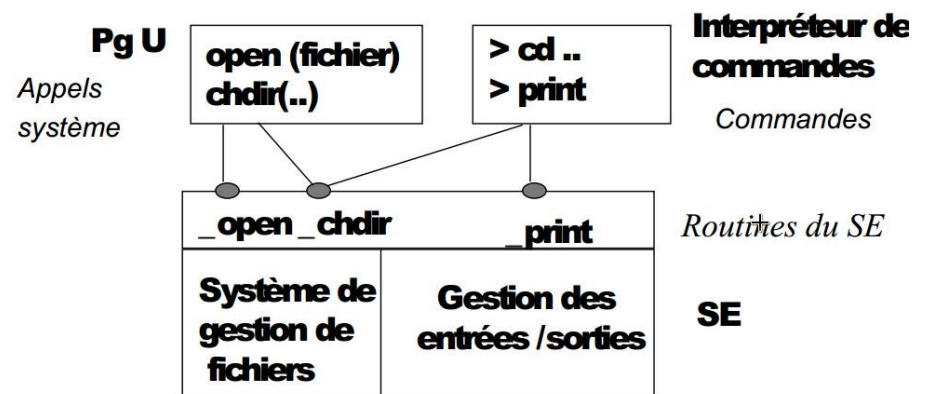
Introduction aux systèmes d'exploitation multiprogrammés

Structure d'un système d'exploitation multiprogrammé

Composants d'un système d'exploitation (suite)

- Les **fonctionnalités** du système d'exploitation **utilisent** les **mécanismes** offerts par le **matériel** de la machine physique pour **réaliser** leurs **opérations**. Notamment, le **système d'exploitation s'interface** avec la couche **matérielle**, par le **biais** du **mécanisme** des **interruptions**, qui lui permet de prendre **connaissance** des **événements** survenant sur la machine matérielle.
- Par ailleurs, le **système d'exploitation s'interface** avec les **applications** du **niveau utilisateur** par le **biais** des **fonctions prédéfinies** que chacune de ses **fonctionnalités** offre. Ces **fonctions** que l'on **qualifie** de **routines systèmes** constituent les **points d'entrées** des **fonctionnalités** du **système d'exploitation** et sont **appelables** depuis les **applications** de niveau **utilisateur**. Ces **appels** peuvent se faire à **deux niveaux** :
 - dans le **code** d'un **programme utilisateur** à l'aide d'un **appel système**, qui n'est autre qu'une forme d'appel de procédure amenant à l'exécution d'une routine système ;
 - depuis le **prompt** de l'**interpréteur de commandes** à l'aide d'une commande. L'interpréteur de commandes est un outil de niveau utilisateur qui accepte les commandes de l'utilisateur, les analyse et lance l'exécution de la routine système associée.

- La **norme POSIX** (*Portable Operating System Interface*) définit l'ensemble des services et primitives que doit offrir un système d'exploitation dit ouvert pour permettre l'écriture d'applications portables entre systèmes différents.



Les deux niveaux d'appel des routines systèmes

Introduction aux systèmes d'exploitation multiprogrammés

Principaux types de systèmes d'exploitations multiprogrammés

- Les **systèmes d'exploitation multiprogrammés** peuvent être **classés** selon **différents types** qui dépendent des **buts** et des **services** offerts par les systèmes. **Trois grandes classes** de systèmes peuvent être **définies** : les **systèmes à traitements par lots**, les **systèmes multi-utilisateurs interactifs** et les **systèmes temps réels**.
- **Systèmes multiutilisateurs interactifs et en temps partagé** :
 - **l'utilisateur** est "derrière son clavier et son écran" ; il **soumet** des **exécutions** et **attend** les **résultats** : il **faut donc réduire** au **maximum** le **temps d'attente** et **faire croire** à **l'utilisateur** qu'il est **seul à utiliser** la **machine**.
 - systèmes **adaptés** à la **mise au point de programmes** (exemple : UNIX, Linux...)
- **Systèmes à traitements par lots** :
 - les **programmes** sont **exécutés** en **différé**, les **uns** à la **suite** des **autres**.
 - systèmes **dédiés** aux **travaux** de **production** (exemple : MVS...)
 - on peut noter que **beaucoup** de **systèmes** offrent simultanément un **service** de **temps partagé** et un service de **traitement par lots** (VMS)
- **Systèmes temps réel (réactifs)** :
 - les **programmes** en **exécution** sont **soumis** à des **contraintes** de **temps**, c'est-à-dire que leurs **exécutions** doivent être **impérativement achevées** à une date butoir appelée **échéance**. Comme ces systèmes sont **souvent interfacés** à un **environnement** dynamique (**procédé**) délivrant des **événements synchrones** ou **asynchrones** auxquels ils doivent **réagir**, on parle aussi de **systèmes réactifs**.
 - systèmes adaptés à la commande de procédé ([voir le lien suivant pour des exemple de système RTOS](#))

Introduction aux systèmes d'exploitation multiprogrammés

Notions de base

Introduction

- Le **système d'exploitation s'interface** avec les **applications** du niveau **utilisateur** par le **biais** de fonctions prédéfinies qualifiées de **routines systèmes** et qui constituent les points d'entrées des fonctionnalités du système.
- Ces **appels** peuvent être de **deux natures** et se faire soit par le biais d'un **appel système**, soit par le biais d'une **commande** du langage de commandes.
- **L'exécution des routines systèmes s'effectue** sous un **mode privilégié**, appelé **mode superviseur** ou mode maître.

Modes d'exécutions et commutations de contexte

Modes d'exécutions

- Un **programme utilisateur s'exécute** par **défaut** selon un mode qualifié de mode esclave ou **mode utilisateur** :
 - les actions du programme sont restreintes, permet de protéger la machine.
- Le **système d'exploitation**, quant à lui, **s'exécute** dans un mode privilégié encore appelé **mode superviseur**, pour lequel **aucune restriction** de droits n'existe.

Introduction aux systèmes d'exploitation multiprogrammés

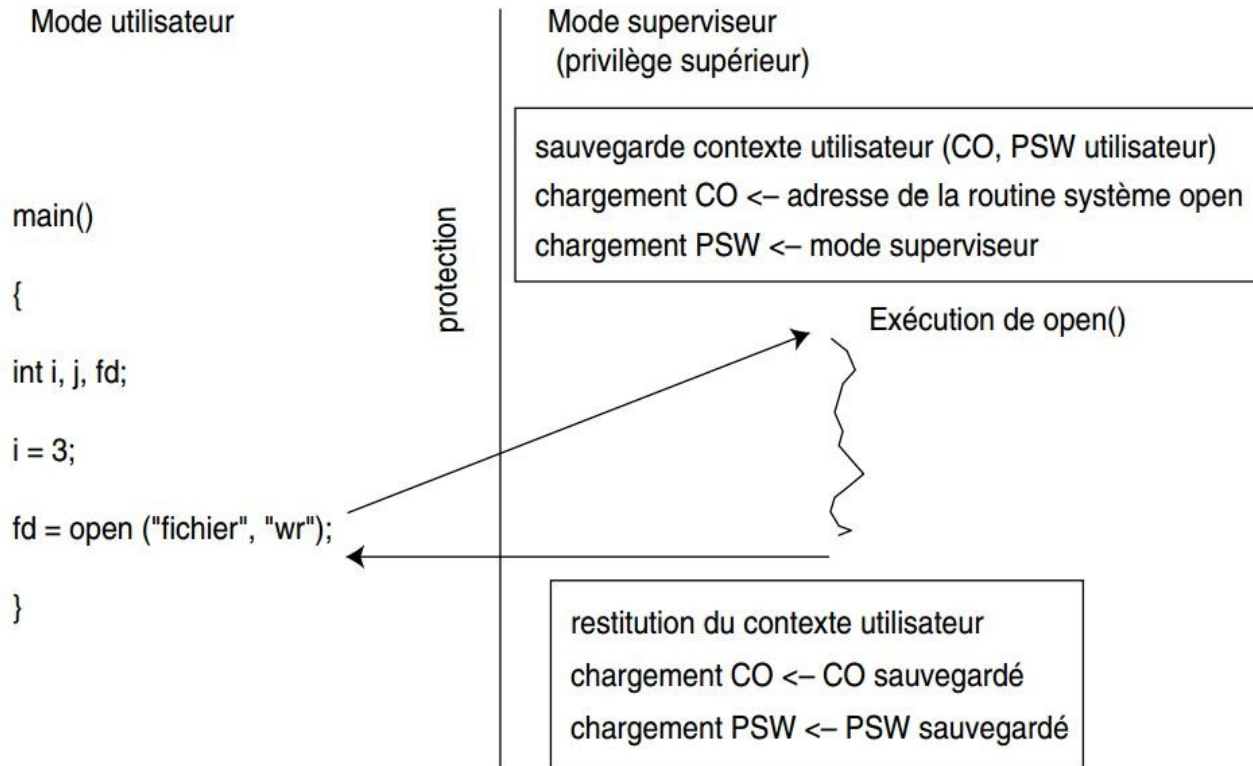
Notions de base

Modes d'exécutions et commutations de contexte (suite)

Commutations de contexte

Sur la figure :

1. Un programme utilisateur demande l'exécution d'une routine du SE, par le biais d'un appel système (open()) dans l'exemple).
2. Le programme quitte le mode utilisateur pour le mode superviseur.
3. Sauvegarde du contexte, les registres du processeurs (CO, PSW).
4. Chargement dans les registres du processeur du nouveau contexte.
5. A la fin de l'exécution de la routine, on restaure le contexte utilisateur.
6. Reprise du programme utilisateur.



**Commutations de
contexte.**

Introduction aux systèmes d'exploitation multiprogrammés

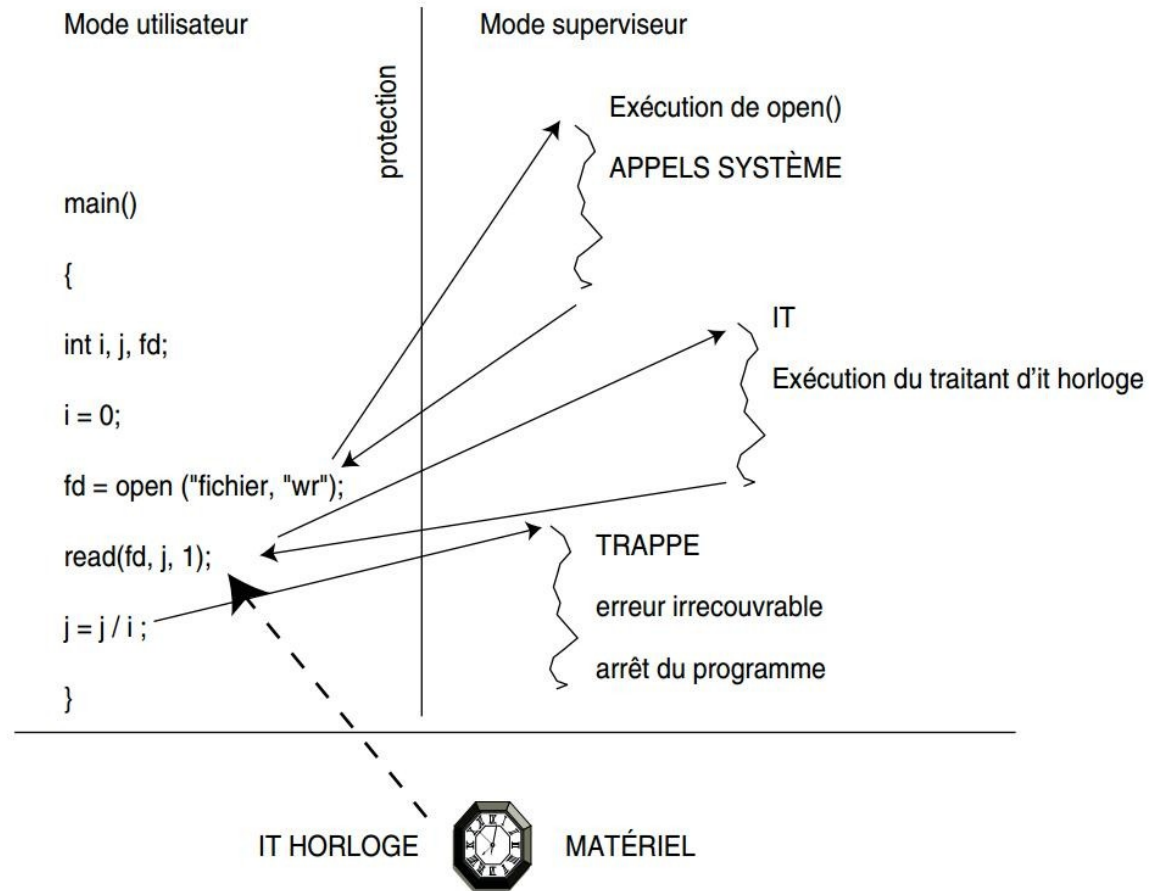
Notions de base

Modes d'exécutions et commutations de contexte (suite)

Commutations de contexte (suite)

Trois causes majeures provoquent le passage du **mode utilisateur** au **mode superviseur** (figure) :

1. le programme utilisateur **appelle** une **fonction du système**. C'est une **demande explicite** de passage en mode superviseur ; Le programme quitte le mode utilisateur pour le mode superviseur.
2. **l'exécution** par le programme utilisateur d'une **opération illicite** (division par 0, instruction machine interdite, violation mémoire...) : c'est la **trappe** ou **l'exception**.
3. la **prise en compte** d'une **interruption** par le **matériel** et le système d'exploitation. Le **programme utilisateur** est alors **stoppé** et **l'exécution de la routine d'interruption** associée à l'interruption survenue est **exécutée** en **mode superviseur**.



Trois causes de commutations de contexte.

- **Trappes** et **appels systèmes** sont parfois **qualifiés d'interruptions logicielles** pour les opposer aux **interruptions matérielles** levées par les **périphériques** du processeur.

Introduction aux systèmes d'exploitation multiprogrammés

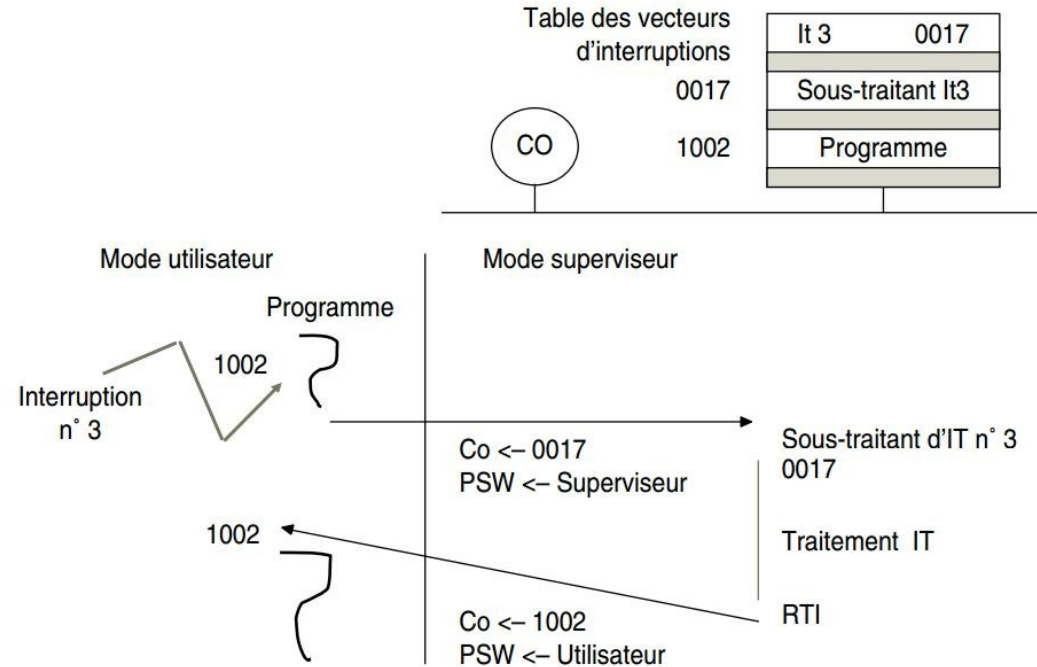
Notions de base

Gestion des interruptions matérielles et logicielles

Prise en compte d'une interruption matérielle

Sur la figure :

1. **Exécution** par le processeur du **programme utilisateur** courant (adresse du programme dans l'exemple, **1002**).
2. **Survenue** d'une **interruption matérielle**, dans l'exemple interruption n°3.
3. Le **programme utilisateur** est **stoppé**, il **quitte le mode utilisateur** pour le **mode superviseur**.
4. **Sauvegarde du contexte utilisateur** :
 1. sauvegarde matérielle : registres **CO** et **PSW**
 2. sauvegarde des **registres généraux** par le SE.
5. Le **CO** est **chargé** avec l'**adresse du traitant d'interruption**, l'information est recherchée dans la **table des vecteurs d'interruptions** (dans l'exemple, **0017**).
6. Le **traitant d'interruption** est **exécuté**.
7. Le **contexte utilisateur** est **restauré**.
8. Le **programme utilisateur** reprend son **exécution en mode utilisateur**.



Prise en compte d'une interruption.

Conclusion

- A FAIRE

TP

TP : A FAIRE