1. **Syllabus**

Semestre : 6 Parcours SI

Unité d'enseignement fondamentale : UEF1

Matière : Système d'exploitation 2

Crédits : 5 Coefficient : 3

Objectifs de l'enseignement :

Une étude approfondie du système Unix est recommandée pendant les séances de TD et de TP. La programmation des threads et des mécanismes de l'exclusion mutuelle se fera en C sous Unix.
Les modèles producteur/consommateur, lecteur/rédacteurs et des philosophes avec plusieurs variantes seront étudiés de façon théorique (développement d'algorithmes en pseudo-langage) en TD puis implémentés en C sous Unix durant les séances de TP.

Connaissances préalables recommandées : système d'exploitation 1.

Contenu de la matière :

Chapitre 1 :

- Rappels sur la notion de SE.

- Notions de programme, processus, thread et ressource partagée.

Chapitre 2 : Synchronisation de processus.

- Problème de l'accès concurrent à des ressources et sections critiques (Problème de l'exclusion mutuelle)

- Outils de synchronisation :

  - Evénements, Verrous
  - Sémaphores
  - Moniteurs
  - Régions critiques.
  - Expressions de chemins

Chapitre 3 : La communication interprocessus

  - Partage de variables (modèles : producteur/ consommateur, lecteurs/ rédacteurs)
  - Boites aux lettres
  - Echange de messages (modèle du client/ serveur)

Chapitre 4 : L'inter blocage

  - Modèles
  - Prévention
  - Evitement
  - Détection/ Guérison

Mode d'évaluation : Examen (60%), contrôle continu (40%)

Chapitre 1 :

- Rappels sur la notion de SE.

- Notions de programme, processus, thread et ressource partagée.

1. QU'EST-CE QU'UN SYSTEME D'EXPLOITATION ?

Un système d'exploitation est un programme qui contrôle l'exécution de tous les autres programmes (applications). Un système d'exploitation joue le rôle d'intermédiaire entre l'utilisateur (ou les utilisateurs) et l'ordinateur. Les principales objectives de ce programme sont :

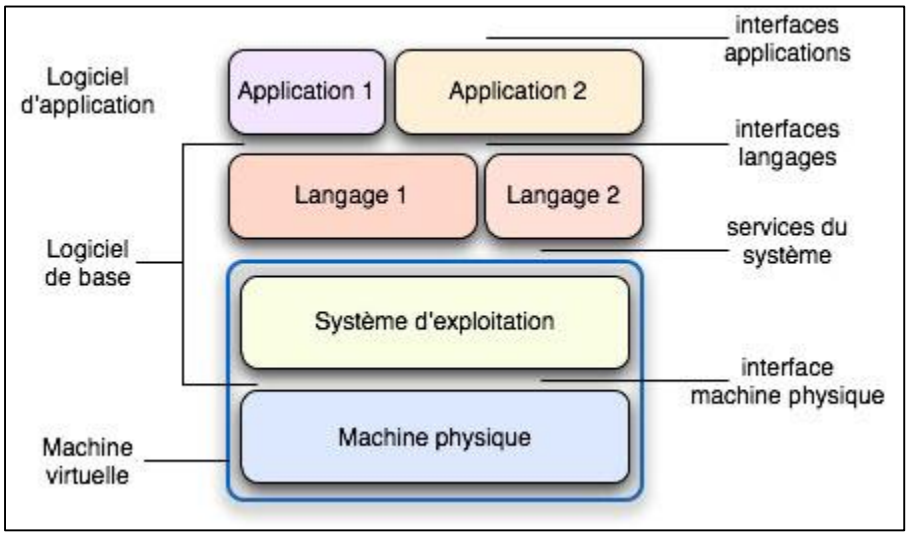


Figure 1 Qu'est-ce qu'un système d'exploitation

- ✓ **Commodité**, Le système doit répondre de façon prévisible à des conditions d'erreurs même celles causées par une panne matérielle.
- ✓ **Efficacité**, Le système doit se protéger et protéger ses utilisateurs contre des attaques délibérées ou accidentelles menées par des programmes d'utilisateurs.
- ✓ **Extensibilité,** il existe différents niveaux de compatibilité : compatibilité source : l'application doit être recompilée. compatibilité binaire : exécution directe (très difficile entre des processeurs différents).
- ✓ Similaire A un Gouvernement ☺

2. HISTORIQUE :

Les systèmes d'exploitation, comme le matériel informatique, ont subi une série de changements révolutionnaires appelés générations. Dans le cas du matériel informatique, les générations ont été marquées par des avancées majeures dans les composants, des tubes à vide (première génération) aux transistors (deuxième génération), en passant par les circuits intégrés (troisième génération) et les circuits intégrés à grande et très grande échelle (quatrième génération). Les générations successives de matériel informatique se sont toutes accompagnées d'une réduction spectaculaire des coûts, de la taille, des émissions de chaleur et de la consommation d'énergie, ainsi que d'une augmentation spectaculaire de la vitesse et de la capacité de stockage.

1) Les premiers ordinateurs étaient conçus pour exécuter une série de tâches simples, comme une calculatrice. Les caractéristiques de base des systèmes d'exploitation ont été développées dans les années 1950, comme les fonctions de moniteur résident qui pouvaient exécuter automatiquement différents programmes les uns après les autres pour accélérer le traitement.

2) Dans les années 40, les premiers systèmes électroniques numériques n'avaient pas de système d'exploitation. Les systèmes électroniques de cette époque étaient programmés sur des rangées de commutateurs mécaniques ou par des fils de liaison sur des cartes de connexion.
   Il s'agissait de systèmes à usage spécifique qui, par exemple, généraient des tableaux balistiques pour l'armée ou contrôlaient l'impression des chèques de paie à partir de données figurant sur des cartes perforées.
   Après l'invention des ordinateurs programmables à usage général, des langages machine (consistant en des chaînes de chiffres binaires 0 et 1 sur des bandes de papier perforé) ont été introduits pour accélérer le processus de programmation (Stern, 1981).

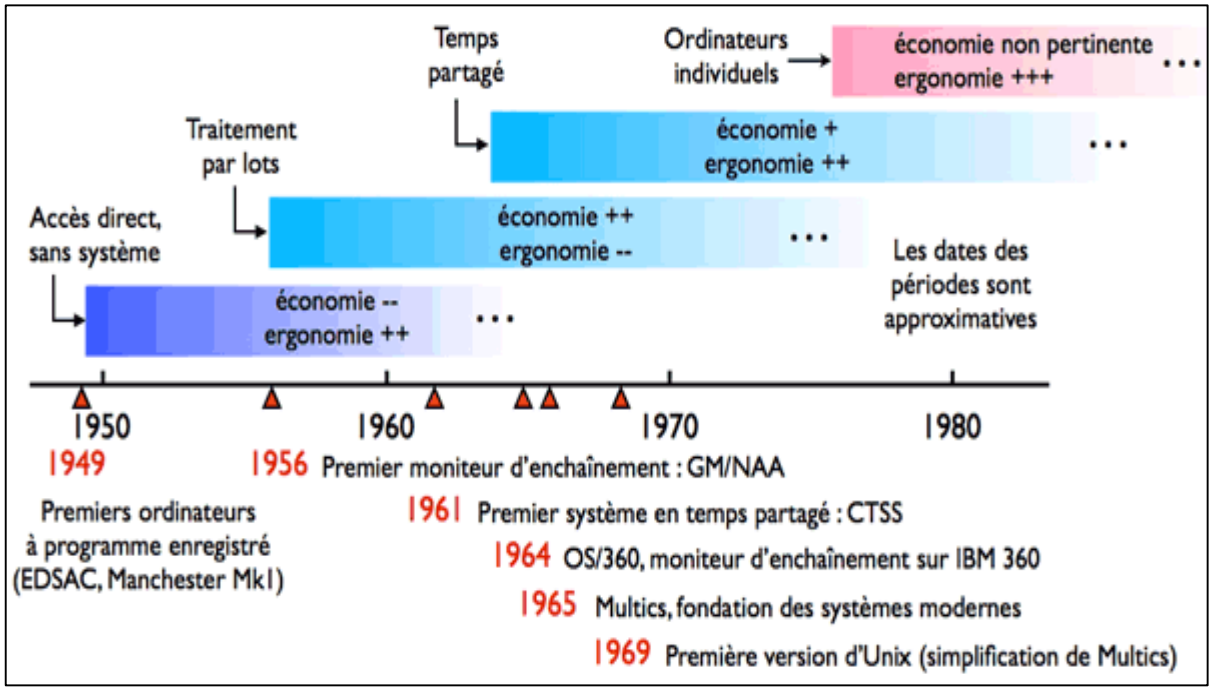


Figure 2 Chronologie simplifiée des premiers systèmes

3) SE/360 a été utilisé sur la plupart des ordinateurs centraux IBM à partir de 1966, y compris les ordinateurs utilisés par le programme Apollo.
4) Au début des années 1950, un ordinateur ne pouvait exécuter qu'un seul programme à la fois. Chaque utilisateur avait l'usage exclusif de l'ordinateur pour une période limitée et arrivait à une heure précise avec son programme et ses données sur des cartes de papier perforé ou des bandes perforées.
   Le programme était chargé dans la machine, et la machine était mise au travail jusqu'à ce que le programme soit terminé ou qu'il tombe en panne. Les programmes pouvaient généralement être débogués via un panneau frontal à l'aide d'interrupteurs à bascule et de voyants. On dit qu'Alan Turing était passé maître dans ce domaine sur la première machine Manchester Mark 1, et qu'il déduisait déjà la conception primitive d'un système d'exploitation des principes de la machine universelle de Turing.
5) Plus tard, les machines ont été livrées avec des bibliothèques de programmes, qui étaient liées au programme de l'utilisateur pour l'aider dans des opérations telles que l'entrée et la sortie et la compilation (génération du code machine à partir du code symbolique lisible par l'homme).
6) À l'université de Cambridge, en Angleterre, la file d'attente des tâches était à un moment donné une corde à linge à laquelle étaient suspendues des bandes avec des pinces à linge de différentes couleurs pour indiquer la priorité des tâches.

7) Une amélioration a été apportée par le superviseur de l'Atlas. Introduit avec le Manchester Atlas en 1962, il est considéré par beaucoup comme le premier système d'exploitation moderne reconnaissable.

3. DEFINITION SYSTEME D'EXPLOITATION

Un système d'exploitation est un programme (logiciel) qui joue un rôle d'interface entre le matériel et l'utilisateur.

Un système d'exploitation est un **logiciel** qui gère le matériel informatique et les ressources logicielles, et qui fournit des services communs aux programmes informatiques.

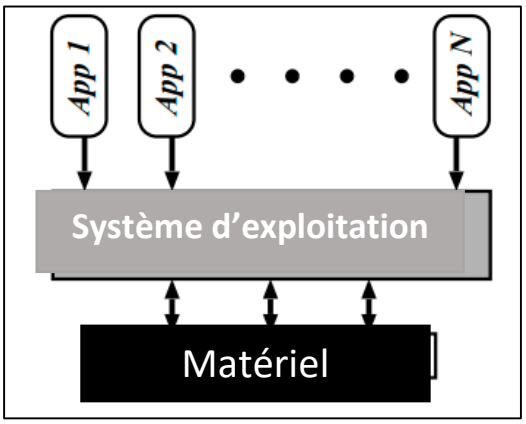


Figure 3 Position du système d'exploitation

Le rôle principal d'un système d'exploitation, c'est **le partage des ressources**. Il se fait entre les programmes appelés plus justement les processus. Ce rôle de policier du système d'exploitation permet d'éviter les conflits d'utilisation de la mémoire, des périphériques d'entrées/sorties, des interfaces réseau... etc.

- ✓ **Exemple 1**

On peut facilement imaginer ce qui arriverait si trois programmes essayaient d'imprimer en même temps sans qu'un certain ordre ne soit respecté. Ce travail d'alternance assuré par le système d'exploitation permet de mettre de l'ordre dans un chaos potentiel.

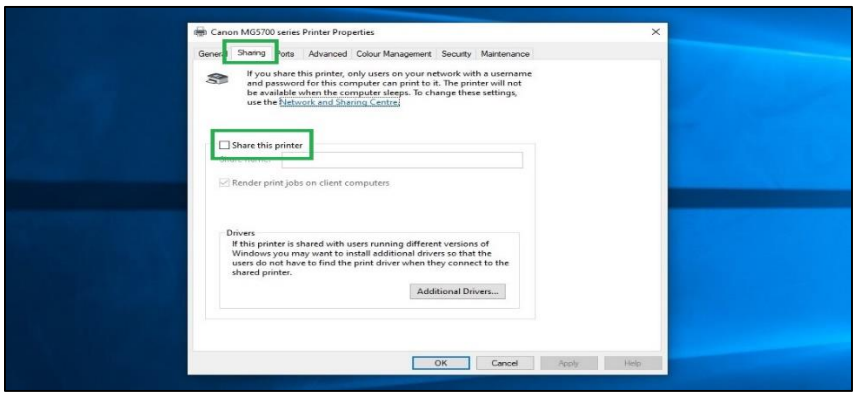


Figure 4 Exemple partage Imprimante

- ✓ **Exemple 2**

De plus, lorsque l'ordinateur est utilisé par plusieurs usagers (presque tout le temps), le partage de la mémoire et surtout sa protection demeure une priorité absolue. En tout temps, un bon système d'exploitation connaît l'utilisateur d'une ressource, ses droits d'accès, son niveau de priorité.

4. LES COMPOSANTS D'UN SYSTEME D'EXPLOITATION

Le système d'exploitation est composé d'un ensemble d'applications permettant de gérer les interactions avec le matériel. Parmi cet ensemble de logiciels on distingue généralement les éléments suivants :

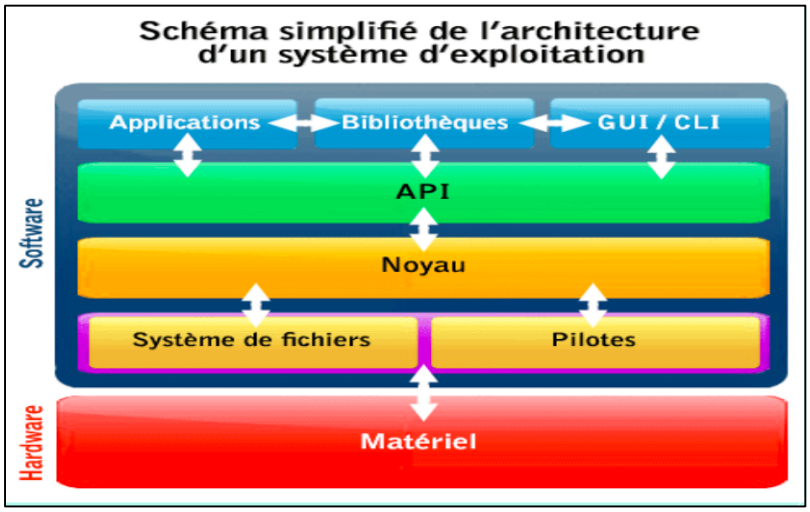


Figure 5 Les composants d'un système d'exploitation

- ✓ **Le noyau** (kernel) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
- ✓ **L'interpréteur de commande** (**shell** - coquille par opposition au noyau) permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes, afin de permettre à l'utilisateur de piloter les périphériques en ignorant tous des caractéristiques du matériel qu'il utilise, de la gestion des adresses physiques, etc.
  - **Exemple** de commandes :
    - *ls* : lister les répertoires et les fichiers du répertoire courant
    - *mv* x y : changer le nom du fichier/répertoire « x » en « y »
- ✓ **Le système de fichiers** (file system) permet d'enregistrer les fichiers dans une arborescence.

5. LES FONCTIONNALITES DU SYSTEME D'EXPLOITATION

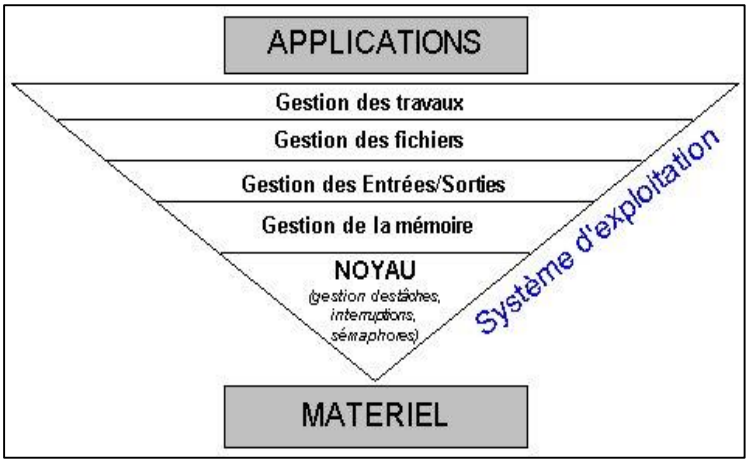


Figure 6 Les fonctionnalités du système d'exploitation

- ✓ ***Gestion des processeurs***. Gérer l'allocation des processeurs entre les différents programmes grâce à un algorithme d'ordonnancement.
- ✓ ***Gestion des processus***. Gérer l'exécution des processus en leur affectant les ressources nécessaires à leur bon fonctionnement.
- ✓ ***Gestion des mémoires***. Gérer l'espace mémoire alloué à chaque processus.
- ✓ ***Gestion des périphériques***. Contrôler l'accès des processus aux ressources matérielles par l'intermédiaire des pilotes.
- ✓ ***Gestion des fichiers*.** Gérer la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

6. **GÉNÉRATIONS DES SYSTÈMES D'EXPLOITATION**

Dans cette section, nous présentons les types de systèmes d'exploitation. Les systèmes d'exploitation informatiques modernes peuvent être classés en six groupes, qui se distinguent par la nature de l'interaction qui a lieu entre l'utilisateur de l'ordinateur et son programme pendant son traitement. Ces groupes sont appelés systèmes d'exploitation par lots, à temps partagé et en temps réel.

1. **Mono utilisateur/Mono tâche**, c'est le cas le plus simple : Un seul utilisateur à la fois et une seule tâche à la fois. Les systèmes d'exploitation des premiers micro-ordinateurs ne dépassaient pas ce niveau de complexité.
2. **Multitâches,** il assure le partager le temps du processeur entre plusieurs programmes. Le passage de l'exécution d'un programme à un autre peut être initié :
    i. Par les programmes eux-mêmes (coopératif)
    ii. Par le S.E (préemptif)
3. **Multiutilisateurs (temps partagé),** Plusieurs utilisateurs peuvent utiliser simultanément une même machine pour des applications similaires ou différentes. Et chaque utilisateur à l'impression d'être le seul à utiliser l'ordinateur.
4. **Multiprocesseurs,** un processeur central (maître) peut coordonner une série de tâches sur plusieurs autres processeurs (esclaves). Organisation à l'extérieur d'une série de tâches sur plusieurs processeurs (systèmes répartis).
5. **Temps réels,** servent pour le pilotage et le contrôle des déroulements externes (exemple centrale électrique), doivent garantir des temps de réactions données pour des signaux extérieurs urgents.

6. **Distribués,** doivent permettre l'exécution d'un seul programme sur plusieurs machines. distribuer les tâches et les remettre ensemble. Pour gros calculs, exemple inversion de grandes matrices.

7. NOTIONS DE BASE

1. PROCESSUS

i. **Définition** : Un processus (en anglais, process), en informatique, est un programme en cours d'exécution par un ordinateur. Auquel on associe :

- θ **Un Contexte du processeur (CPU) : l'ensemble des registres (CO, SW, SP, Rx, …)**
- θ **Un Contexte de la Mémoire Centrale : segments de code, segments de données, …**

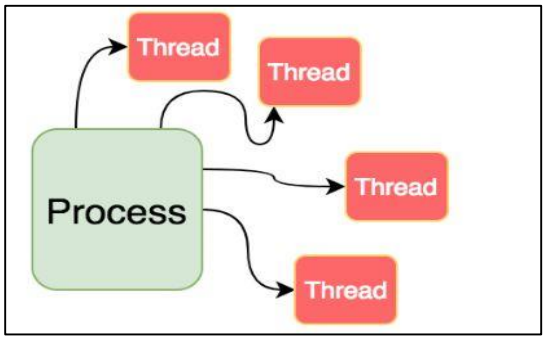


Figure 7 Un processus dans le système d'exploitation

ii. LES ETATS D'UN PROCESSUS

Un processus est un programme actif. On peut également dire qu'il s'agit d'un programme en cours d'exécution. Il est plus que le code de programme car il comprend le compteur de programme, la pile de processus, les registres, le code de programme, etc. Par rapport à cela, le code du programme n'est qu'une section de texte. Un processus passe par différents états au cours de son exécution. Ces états peuvent être différents selon les systèmes d'exploitation.

Dans les systèmes, un programme ne quitte pas l'unité centrale avant de terminer son exécution. Pendant cette période, il dispose de toutes les ressources de la machine. Par contre, ce n'est pas le cas dans les systèmes multiprogrammés et temps partagé, un processus peut se trouver dans l'un des états suivants :

1. **Elu** : (en cours d'exécution) : si le processus est en cours d'exécution
2. **Bloqué** : attente qu'un événement se produit ou bien ressource pour pouvoir continuer
3. **Prêt** : si le processus dispose de toutes les ressources nécessaires à son exécution à l'exception du processeur

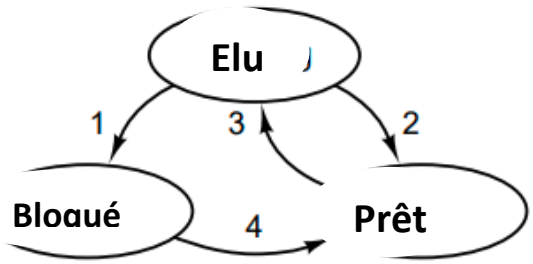


Figure 8 Les états d'un processus

iii. REPRESENTATION INTERNE DES PROCESSUS

Au moment du chargement d'un programme exécutable, le SE lui crée une structure représentant le processus associé. Cette structure devrait contenir toutes les informations nécessaires permettant l'évolution dynamique du processus au sein du SE.

Pour manipuler tous les processus, le SE détient d'une table de processus dont chaque entrée contient un pointeur vers un PCB d'un processus.

Process Control Block (PCB)

- Chaque processus:
  - a un bloc de contrôle qui le décrit.
  - a un IDentifieur unique.
  - peut avoir des enfants ou un père.
  - a un état (voir la prochaine acétate).
  - a ses registres, sa mémoire et sa pile.
  - a une priorité.
- Les processus peuvent partager de la mémoire, des processus, des fichiers, des I/Os et autres.

Process ID (PID)
Pointer to parent process
Pointers to child processes
Process state
Program Counter
Registers
Memory pointers
Priority information
Accounting information
Pointers to shared ressources

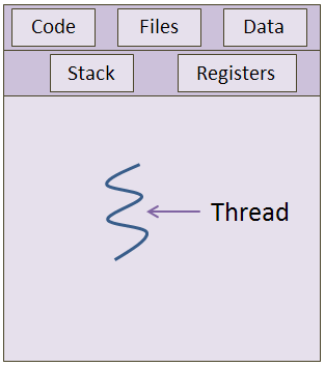
PCB Typique

- ✓ **Pointeur** - Il s'agit d'un pointeur de pile qui doit être sauvegardé lorsque le processus passe d'un état à un autre pour conserver la position actuelle du processus.
- ✓ **État du processus** - Il stocke l'état respectif du processus.
- ✓ **Numéro de processus** - Chaque processus se voit attribuer un identifiant unique connu sous le nom d'ID de processus ou **PID** qui stocke l'identifiant du processus.
- ✓ **Compteur de programme** - Il stocke le compteur qui contient l'adresse de l'instruction suivante qui doit être exécutée pour le processus.
- ✓ **Registre** - Il s'agit des registres du processeur qui comprennent : l'accumulateur, la base, les registres et les registres d'usage général.
- ✓ **Limites de mémoire** - Ce champ contient les informations sur le système de gestion de la mémoire utilisé par le système d'exploitation. Cela peut inclure les tables de pages, les tables de segments, etc.
- ✓ **Liste des fichiers ouverts** - Cette information comprend la liste des fichiers ouverts pour un processus.

2. TACHE (THREAD) :

Dans un système d'exploitation, un processus est une tâche ou un programme qui peut être exécuté par l'ordinateur. Pensez à l'application MS Word, qui est un processus exécuté sur un ordinateur. Mais une application peut faire plus d'une chose à la fois, ce qui signifie qu'un processus donné dans un système d'exploitation peut avoir un ou plusieurs threads. Les *threads* représentent le traitement réel du code.

Un processus possède ses propres registres système et sa propre pile mémoire, ce qui l'aide à exécuter des threads. Les threads sont parfois appelés processus légers. Le graphique ci-dessous montre un processus avec un seul thread à l'intérieur :



i. DIFFERENCES ENTRE PROCESSUS ET THREAD

Les processus et les threads fonctionnent ensemble, mais ils présentent de nombreuses différences entre eux. Généralement, les processus sont assez lourds (comme MS Word), tandis que les threads sont plus légers (comme l'option de sauvegarde en arrière-plan). Le tableau ci-dessous met en évidence certaines des différences entre les deux.

Processus	**Threads**

Lors de la commutation d'un processus, les ressources du système d'exploitation sont requises	Aucune ressource du système d'exploitation n'est nécessaire pour le changement de fil.
Si un processus est bloqué, les autres processus en attente dans la file d'attente sont également bloqués.	Si un thread est bloqué, un autre thread dans le même processus peut toujours s'exécuter.
Chaque processus utilise le même code et possède sa propre mémoire.	Tous les threads peuvent partager des fichiers et des processus enfants.
Une application comportant plusieurs processus utilisera davantage de ressources système.	Les processus utilisant plusieurs threads utilisent moins de ressources système.
Chaque processus fonctionne sur sa propre	Les threads peuvent accéder aux données des autres threads.

Lorsqu'un processus présente un code un petit peu long ou compliqué, il peut être décortiqué en un ensemble d'unités de traitements élémentaires ayant une cohérence logique dont la fonction est bien déterminée. Cette unité est appelée une tâche.

- ✓ Un thread est une subdivision d`un processus
- ✓ Les différents threads d'un processus partagent l'espace adressable et les ressources d'un processus
- ✓ Les threads sont des processus légers exécutés «à l'intérieur» d'un processus
- ✓ L'exécution des threads est concurrente
- ✓ Il existe toujours au moins un thread : le thread principal
- ✓ La durée de vie d'un thread ne peut pas dépasser celle du processus qui l'a crée

ii. MULTI-THREADING

Une application multithread est un logiciel qui dès sa conception a été partagé en différentes sous-applications appelées threads.

A l'inverse de ce qui passe dans le cadre multitâche où chaque processus dispose d'une partie distincte de la mémoire pour s'exécuter indépendamment des autres. Les threads issus de même application partagent un même espace mémoire.

- ✓ Un thread pour interagir avec l'utilisateur ;
- ✓ Un thread pour reformater en arrière-plan ;
- ✓ Un thread pour sauvegarder périodiquement le document ;
- ✓ Un thread la vérification automatique de l'orthographe et de la grammaire.

3. RESOURCE PARTAGER

Une ressource désigne toute entité dont a besoin un processus pour s'exécuter.

- ✓ Ressource matérielle (processeur, périphérique)
- ✓ Ressource logicielle (fichier, variable).

Une ressource est caractérisée :

- ✓ par un état : libre / occupée

- ✓ par son nombre de points d'accès (nombre de processus pouvant l'utiliser en même temps)

Une ressource critique à un seul point d'accès.