GESTION DES PROCESSUS ET MÉCANISMES DE THREADS

1. Gestion des Processus

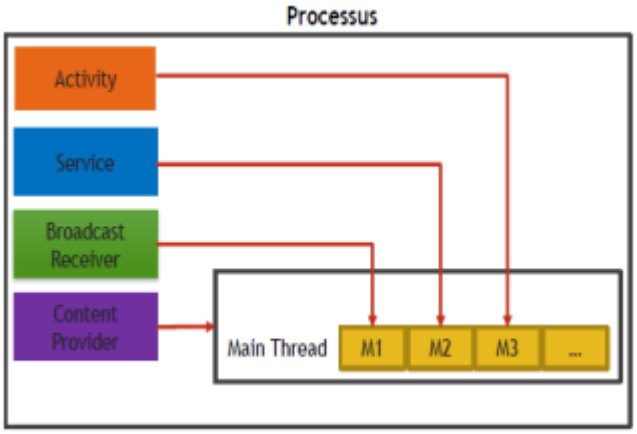
Le système d'exploitation (SO) gère les processus, qui sont des entités représentant des applications ou des tâches en cours d'exécution. Chaque processus s'exécute dans un environnement isolé avec sa propre mémoire.

iOS

- **Gestion des processus** : iOS utilise un modèle basé sur **Darwin** (un noyau Unix), où chaque application fonctionne dans un **sandbox** (bac à sable). Cela signifie que chaque application est isolée et ne peut pas interférer directement avec une autre.
- **Processus et threads** : Chaque application dispose d'un seul processus principal. À l'intérieur de ce processus, plusieurs threads peuvent être créés pour gérer des tâches en arrière-plan.

Android

- **Gestion des processus** : Android utilise le **noyau Linux** pour la gestion des processus. Chaque application fonctionne dans un processus distinct, mais plusieurs applications peuvent partager le même processus si elles sont signées avec le même certificat.
- **Processus et threads** : Android fonctionne également dans un environnement multitâche avec la possibilité de créer plusieurs threads pour des tâches parallèles (par exemple, en utilisant des **AsyncTask**, **Handler**, ou **Thread**).

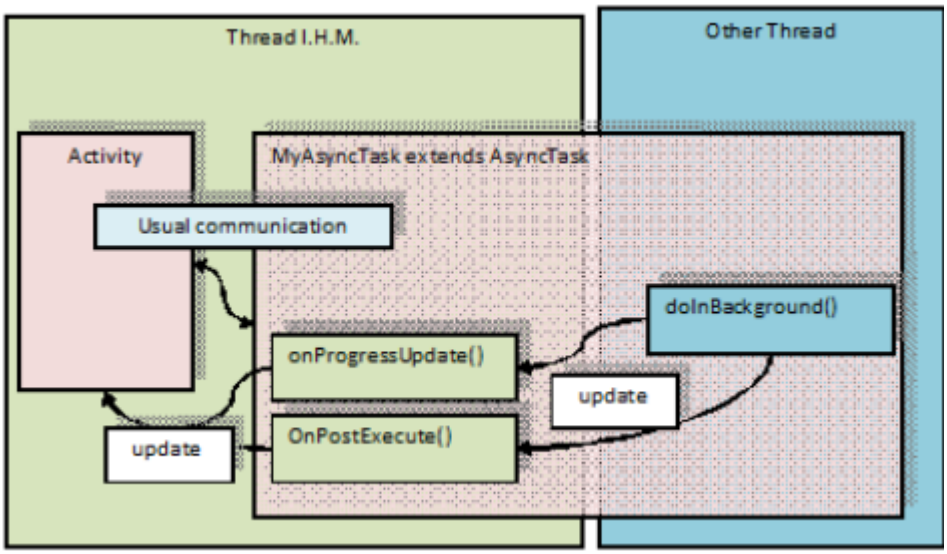


Windows

- **Gestion des processus** : Windows gère les processus via le **Windows NT Kernel**. Chaque application s'exécute dans un processus isolé, mais les processus peuvent interagir via des mécanismes comme les **pipes** et les **sockets**.
- **Processus et threads** : Windows permet la création de multiples threads dans un même processus. Le système d'exploitation gère la planification de ces threads grâce à son gestionnaire de threads, ce qui permet des traitements parallèles.

2. Mécanismes de Threads

Les threads sont les plus petites unités d'exécution dans un processus. Un processus peut avoir plusieurs threads exécutés simultanément.



iOS

- **Threads** : iOS utilise **Grand Central Dispatch (GCD)** et **NSOperationQueue** pour la gestion des threads. Ces mécanismes permettent de gérer facilement les threads en arrière-plan et de garantir une exécution fluide de l'interface utilisateur sur le **main thread**.
- **GCD** : GCD permet de gérer des queues de tâches (série ou parallèle), ce qui est plus performant que d'utiliser des threads manuels.

Android

- **Threads** : Android utilise des **threads** pour exécuter des tâches en arrière-plan. Le système d'exploitation est conçu pour éviter que les opérations longues (comme les appels réseau ou la lecture de fichiers) ne bloquent l'interface utilisateur. Des classes comme `AsyncTask` ou `ThreadPoolExecutor` sont utilisées pour gérer l'exécution des tâches en parallèle.
- **Gestion des threads** : La bibliothèque **Handler** est utilisée pour gérer les communications entre le thread principal (UI) et les autres threads.

Windows

- **Threads** : Windows prend en charge la création de threads via des API comme **CreateThread** et **ThreadPool**. Ces threads peuvent s'exécuter simultanément dans un même processus. Windows gère automatiquement les priorités et les ressources allouées aux threads via son gestionnaire de threads.
- **Multi-threading** : Windows fournit un **scheduling préemptif** pour les threads, ce qui permet de garantir que les threads peuvent être exécutés de manière équitable en fonction de leurs priorités.

SCHEDULING (PLANIFICATION)

La planification des threads permet à un système d'exploitation de déterminer quel thread ou processus sera exécuté à un moment donné. Cette planification peut être préemptive (le système choisit le thread suivant) ou coopérative (le thread actuel choisit de céder le contrôle).

1. **iOS (Grand Central Dispatch - GCD)**

- **Modèle de Scheduling** : iOS utilise un système de **scheduling basé sur les queues de GCD**. Cela permet de gérer les tâches concurrentes efficacement, en plaçant les tâches dans des queues de haute ou basse priorité. Le **dispatch queue** est un mécanisme central dans iOS, avec des tâches exécutées de manière concurrente ou en série, selon le type de queue (parallèle ou séquentielle).
- **Thread Pool** : iOS gère également un **thread pool** interne qui alloue des threads à différentes queues de tâches, optimisant l'utilisation du processeur.

2. **Android**

- **Modèle de Scheduling** : Android utilise un mécanisme de **scheduling préemptif** avec un système basé sur le **noyau Linux**. Le système Linux attribue un temps d'exécution (quantum) à chaque processus et thread, puis sélectionne celui qui doit s'exécuter en fonction de la priorité.
- **Scheduling sur le Main Thread** : Android privilégie l'exécution des tâches liées à l'interface utilisateur sur le **main thread**, et les autres tâches (par exemple, les appels réseau ou la gestion de la base de données) doivent être exécutées sur des threads secondaires ou en arrière-plan pour ne pas bloquer l'interface.

Les Nices en fonction du type de processus

THREAD_PRIORITY_LOWEST	19
THREAD_PRIORITY_BACKGROUND	10
THREAD_PRIORITY_DEFAULT	0
THREAD_PRIORITY_FOREGROUND	-2
THREAD_PRIORITY_DISPLAY	-4
THREAD_PRIORITY_URGENT_DISPLAY	-8
THREAD_PRIORITY_VIDEO	-10
THREAD_PRIORITY_AUDIO	-16
THREAD_PRIORITY_URGENT_AUDIO	-19

3. **Windows**

- **Modèle de Scheduling** : Windows utilise un **scheduling préemptif** pour gérer les threads. Le **Windows scheduler** répartit les ressources entre les threads en fonction de leur priorité et de la durée pendant laquelle ils se sont exécutés. Il utilise un algorithme de planification basé sur les **priorités dynamiques**, où les threads ayant une priorité plus élevée seront exécutés en premier.
- **Thread Pools** : Windows utilise également des **pools de threads** qui permettent de gérer efficacement les tâches asynchrones sans avoir à créer de nouveaux threads manuellement.

CONCLUSION

La gestion des processus et des threads ainsi que le scheduling varient d'un système d'exploitation à l'autre, mais chaque système met en œuvre des mécanismes permettant d'exécuter plusieurs tâches simultanément tout en préservant la fluidité et la réactivité des applications.

- **iOS** : Utilise **Grand Central Dispatch (GCD)** et **NSOperationQueue** pour gérer les threads, avec une forte abstraction des threads manuels et une gestion efficace des tâches en arrière-plan.
- **Android** : Utilise un modèle de **scheduling préemptif** basé sur le noyau Linux et fournit des outils comme les **AsyncTask** et les **Handlers** pour gérer l'exécution des tâches sans bloquer l'interface utilisateur.
- **Windows** : Utilise un système de **planification préemptive** des threads, avec des mécanismes comme les **thread pools** pour gérer la concurrence et optimiser l'utilisation des ressources.

Chaque plateforme vise à maximiser la performance des applications tout en garantissant une expérience utilisateur fluide et réactive.