

## Séance 6

# Système d'exploitation Windows



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

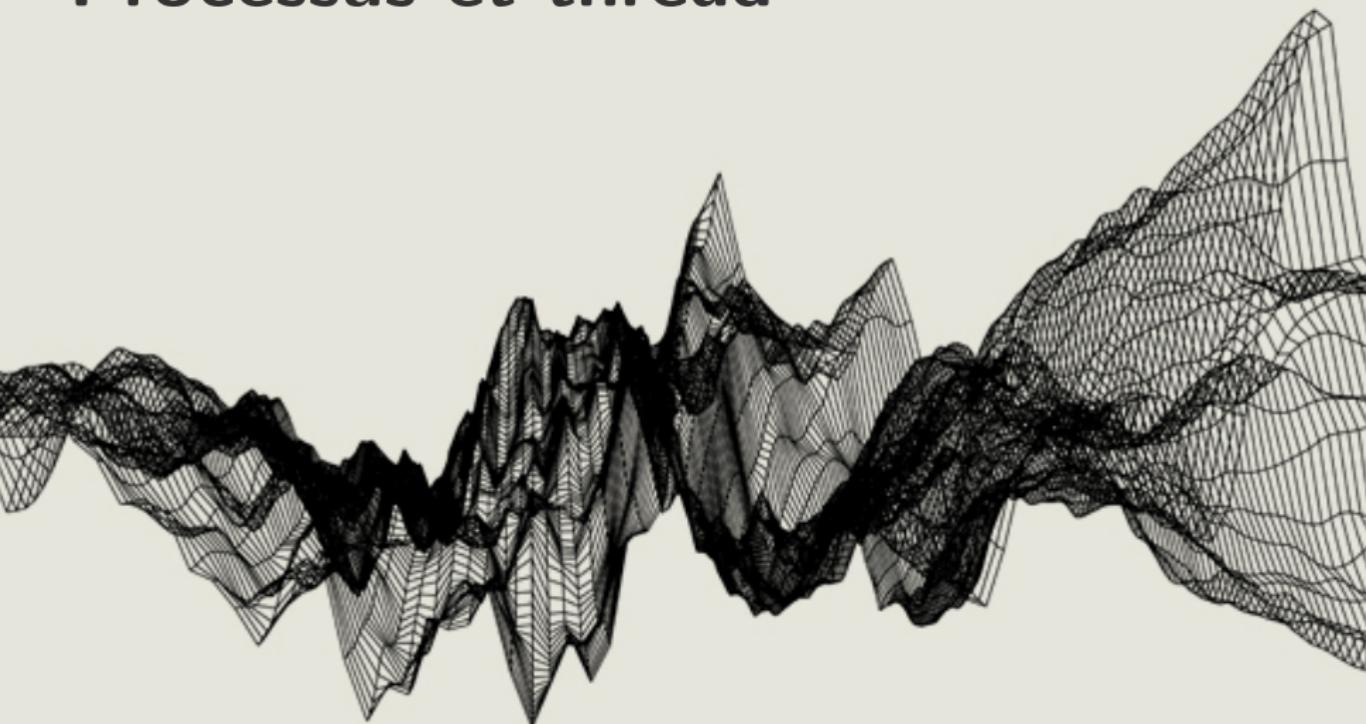
# Rappels

- Blabla
  - Blabla
  - Blabla
- Blabla
  - Blabla
  - Blabla
  - Blabla

# Objectifs

- Blabla
  - Blabla
  - Blabla
- Blabla
  - Blabla
  - Blabla
  - Blabla

# Processus et thread



# Exécution d'application

- Une **application** est composée de plusieurs processus
  - Le processus fournit les ressources nécessaires pour l'exécution
  - Un seul thread (primaire) au début et création possible future
- Le **thread** est l'entité ordonnancée pour exécution
  - Job object rassemble des processus pour gestion commune
  - Thread pool permet de l'exécution asynchrone efficace 
  - Un thread peut être lui-même composé de fibers



- **Fibers** au sein d'un thread, ordonnancés par l'application
  - Portage d'applications qui géraient leurs propres threads
  - Sélection d'un fiber à exécuter pour chaque thread
  - Fiber ordonnancés de manière non préemptive
- Mécanisme **User-mode Scheduling** (UMS) pour application
  - Permet à l'application d'ordonnancer ses threads
  - Utile pour travail court avec peu d'appels système

# Processus Windows

- Processus implémentés comme **des objets** sur Windows
  - Peut être créé comme nouveau ou comme copie d'un existant
  - Un processus exécutable contient un ou plusieurs threads
  - Processus et threads ont des mécanismes de synchronisation
- Relation intime entre un **processus et ses ressources**
  - Un token d'accès sécurité est détenu par chaque processus
  - Détient des handles pour les objets détenus par le processus

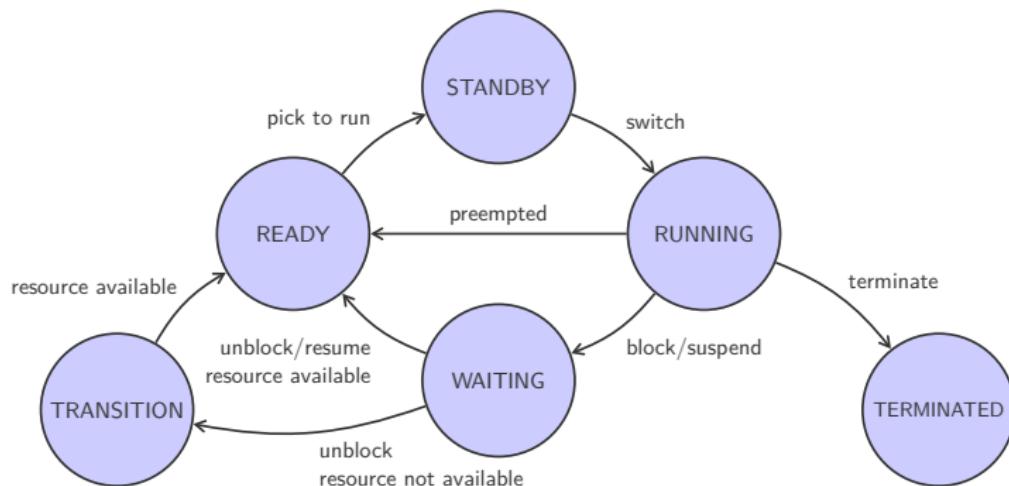
# Objet processus et thread

- Deux types d'objets manipulés par Windows
  - **Processus** entité correspondant à un job utilisateur  
*Possède des ressources comme mémoire et fichiers ouverts*
  - **Thread** unité de travail qui peut être dispatchée  
*Exécution séquentielle et qui peut être interrompue*
- Un processus Windows doit contenir **au moins un thread**
  - Exécution parallèle des threads sur système multiprocesseur
  - Exécution concurrente des processus

# États des threads

- Six états différents pour un thread Windows

*Ready, Standby, Running, Waiting, Transition, Terminated*





# Concurrence

# Mécanisme de concurrence

- Mécanismes de **synchronisation** entre les threads

*Font partie de l'architecture orientée objet de Windows*

- **Fonction d'attente** bloque un thread sur sa propre exécution

- Attente sur un critère jusqu'à ce qu'il soit satisfait
  - WaitForSingleObject attend un signal d'un objet (ou timer)

# Executive Dispatcher Object (1)

- Existence de plusieurs **objets dispatcher**
  - Sont chacun dans l'état signalé ou non signalé
  - Un thread peut être bloqué en attendant un objet non signalé
- **Windows Executive** gère les attentes des threads
  - Thread fait une requête d'attente au Windows Executive
  - Libération threads en attente lors passage vers signalé

# Executive Dispatcher Object (2)

- **Event object** signale l'occurrence d'un évènement
- **Mutex object** force accès exclusif mutuel à une ressource  
*Un seul thread à la fois peut accéder à la ressource*
- **Semaphore object** similaire à l'objet mutex  
*Prend la forme d'un compteur au lieu d'une valeur binaire*
- **Waitable timer object** intervalle régulier ou instant précis

# Section critique

- Section critique pour threads d'un même processus

*Plus rapide que de passer par dispatcher objects*

- Marqueurs dans le code du processus et de ses threads

*Sous la forme de fonctions à appeler (enter, tryenter, leave)*

- Efficace pour acquisition section critique sur courte période

# Gestion de la mémoire



# Virtual Memory Manager Windows

- Gestion mémoire par **Virtual Memory Manager** de Windows  
*Contrôle l'allocation de la mémoire et la pagination*
- Fonctionne sur **plusieurs plateformes** et tailles de page
  - Intel et AMD64 avec pages de 4 Kio
  - Intel Itanium avec pages de 8 Kio

# Map des adresses virtuelles

- Espace d'adresses sur 32 bits pour chaque processus
  - Total de 4 Gio de mémoire virtuelle par processus
  - Moitié de l'espace virtuel réservée pour l'OS
- Mémoire virtuelle découpée en quatre blocs
  - 0x00000000-0x0000FFFF gérer affectation pointeur NULL
  - 0x00010000-0x7FFEFFFF espace user en pages chargées en RAM
  - 0x7FFF0000-0x7FFFFFFF gérer affectation bad-pointer
  - 0x80000000-0xFFFFFFFF système 2 Gio

# Pagination



- Utilisation **espace user** de 2 Gio (32 bits) ou 8 Tio (64 bits)
  - Découpe en pages de taille fixe pouvant être mises en RAM
  - Mais l'OS gère les adresses en régions contigües
- **Trois états** possibles pour chacune des régions
  - **Disponible** adresses actuellement pas utilisées
  - **Réservé** réservé à l'avance pour un processus, pas utilisable
  - **Commité** initialisées pour utilisation, en disque ou RAM

# Working set

- Allocation mémoire de type **variable avec portée locale**

*Initialisation des working sets à la création du processus*

- Ajustement des **working sets** selon trois grands principes

- Augmentation nombre pages résidentes si mémoire disponible
- Suppression pages LRU du working set si mémoire faible
- Monitoring gros processus qui consomme pleins de pages

# Swapping

- Swapfile.sys rejoint pagefile.sys pour **mémoire temporaire**  
*Sous-système qui donne mémoire sur le disque*
- Gestion d'éléments récemment **retiré de la mémoire**
  - Éléments swappés pourraient revenir plus vite que ceux pagés
  - Gestion des requêtes des Windows Store apps sur Metro UI
  - Mémoire de 256 Mio (versus deux fois la RAM pour pagefiles)

# Ordonnancement



# Ordonnancement

- Windows designé pour être **le plus responsive** possible
  - Besoins d'un seul utilisateur avec application interactive
  - Répondre aux clients en tant que serveur
- Ordonnanceur préemptif avec système de **niveaux de priorité**
  - Algorithme de type Round-Robin dans chaque niveau
  - Variation dynamique des priorités pour certains niveaux
  - Ordonnancement au niveau des threads

# Priorité des processus et threads

- Priorités organisées en **deux bandes** temps réel et variable
  - Chaque bande consiste en 16 niveaux de priorité
  - Un thread est placé dans une bande et n'en change pas
- **Gestion différente** des priorités selon la classe
  - Priorité fixe qui ne change jamais pour la classe temps réel
  - Possibilité d'un boost de priorité pour la classe variable
  - Files Roud-Robin pour temps réel et FIFO pour variable

# Classe variable

- Priorité initiale d'un thread déterminée par deux quantités
  - Base du processus entre 1 et 15 (0 réservé pour thread idle)
  - Base du thread avec priorité relative à son processus (-2..2)
- Priorité des threads variable entre celle de base et 15
  - Priorité boostée lors d'une interruption pour évènement E/S
  - Diminuée si thread boosté a consommé son quantum
  - Thread processus priorité basse, thread E/S priorité haute

# Ordonnancement multiprocesseur

- Support de configuration **multiprocesseur et multicœurs**
  - Exécution des threads sur n'importe quel processeur
  - Éviter les processeurs idle ou coincé avec thread basse priorité
- Deux **politiques d'affinité** pour le dispatching des threads
  - **Soft** en affectant toujours sur le même processeur
  - **Hard** en restreignant à certaines processeurs

# Entrées/sorties



# Gestionnaire E/S

- Interaction avec **quatre types** de composants kernel
  - Gestionnaire de cache pour tous les fichiers du système  
*Un thread lazy écrit périodiquement sur le disque en batch*
  - Driver système de fichiers  
*Routage requêtes E/S vers driver software du système de fichiers*
  - Driver réseau  
*Offre capacité réseau et permet l'utilisation de fichiers distants*
  - Driver périphérique hardware  
*Accès registre hardware via Hardware Abstraction Layer*

# Opération E/S

- Supporte des opérations **synchrones et asynchrones**

*Opérations asynchrones préférées pour des raisons d'efficacité*
- Plusieurs techniques pour savoir que **opération d'E/S est finie**
  - Via un file object qu'une demande est satisfaite
  - Via un event objet pour accès simultané sur même ressource
  - Via appel de procédure asynchrone (APC) placé dans une file
  - Via attente sur un port d'E/S terminée
  - Via polling du thread sur le périphérique

# Gestion de volume

- Windows supporte **deux niveaux de RAID**
  - **Hardware** combine disques physiques en un logique
  - **Software** espace disque non contigu en une partition logique
- Deux **fonctionnalités** spéciales liées aux volumes
  - Copie en profondeur pour faciliter le backup
  - Chiffrement du contenu d'un volume

# Système de fichiers



# NTFS

- Windows utilise le **New Technology File System (NTFS)**
- Système de fichiers **flexible et puissant**
  - Récupération crash système et disque par reconstruction  
*Modèle basé sur des transactions et stockage redondant*
  - Sécurité renforcée avec le modèle des objets de Windows
  - Support de très grands disques et fichiers
  - Plusieurs flux de données possibles pour un seul fichier
  - Journalisation des modifications, compression et chiffrement
  - Hard link pour POSIX et liens symboliques

# Volume NTFS

- Trois concepts liés au stockage sur le disque
  - Plus petite unité de stockage sur disque est le secteur (512 o)
  - Un ou plusieurs secteurs contigus forment un cluster ( $2^x$  o)
  - Plusieurs clusters forment un volume pour système de fichiers
- NTFS travaille au niveau des **secteurs**

*Taille maximale d'un fichier est  $2^{32}$  clusters =  $2^{48}$  o*

# Blabla

- Blabla
  - Blabla
  - Blabla
- Blabla
  - Blabla
  - Blabla
  - Blabla

# Crédits

- <https://www.flickr.com/photos/sorarize/5307400678>
- <https://www.flickr.com/photos/carlberger/8738784610>
- <https://www.flickr.com/photos/artetetra/11884033136>
- <https://www.flickr.com/photos/bernhardhanakam/36049853750>
- <https://www.flickr.com/photos/driph/3904965733>