

## Master-1 informatique, module NOY

### Introduction à Nachos (Not Another Completely Heuristic Operating System)



istic Informatique  
Électronique

1

## Généralités

---

- ❑ Historique
  - Conçu à l'University of California at Berkeley (Tom Anderson)
  - Modification au cours des années
- ❑ Objectifs
  - Faciliter la compréhension du fonctionnement interne d'un système d'exploitation
  - Moyen pour atteindre ces objectifs
    - Noyau **complet**
      - tous les concepts fondamentaux existants dans les systèmes d'exploitation (processus, threads, pagination, fichiers, ...)
    - Noyau **simple**
      - Rien que les concepts fondamentaux
      - Architecture matérielle simplifiée (émulée par logiciel). Tous le système (noyau + user) dans un processus Unix

istic Informatique  
Électronique

Introduction à Nachos

2

2

## Concepts de base

---

- ❑ Processus
  - Threads : unités d'exécution élémentaires
  - Espaces d'adressage : espace mémoire partagé par les threads d'un processus
- ❑ Outils de synchronisation entre threads
  - Sémaphores
  - Verrous : sémaphores sans compteur dédiés à l'exclusion mutuelle
- ❑ Fichiers
  - Accès séquentiel et aléatoire
  - Accès en concurrence, pas de droits d'accès
- ❑ Répertoires
  - Arborescence "à la Unix", sans notion de répertoire de travail

## Les travaux pratiques

---

- ❑ Ordonnancement et synchronisation
  - Mécanisme d'appels système
  - Ordonnancement
    - Multi-thread
      - création d'un nouveau thread
      - changement de contexte entre threads
      - destruction
  - Synchronisation
    - Mise en place de sémaphores, verrous (locks), variables de condition
- ❑ Entrées-sorties caractère
  - Par attente active
  - Sous interruption

## Les travaux pratiques

---

- ❑ Pagination à la demande
  - Résolution des défauts de page
  - Remplacement de page (algorithme de l'horloge)
  - Résolution des problèmes de synchronisation pour les processus multi-thread
- ❑ Fichiers mappés

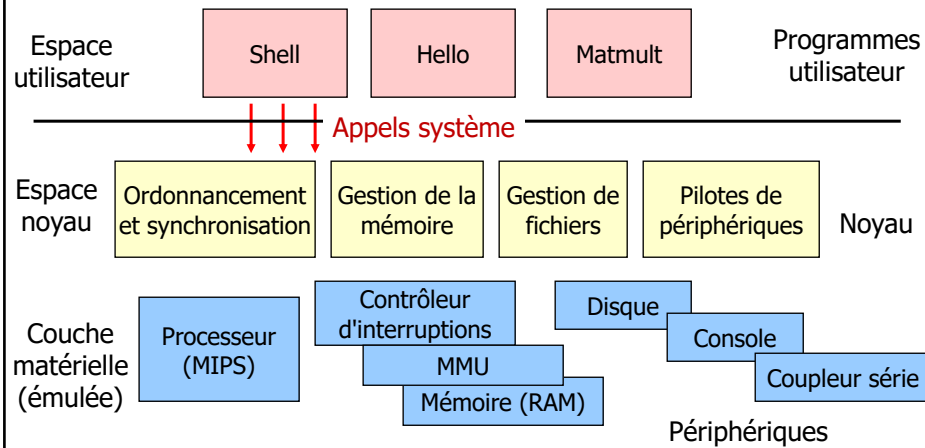
## Les travaux pratiques

### Démarche

---

- ❑ Source "à trous"
  - Vous avez le squelette du système à votre disposition, il n'y a "plus qu'à" le compléter
- ❑ Organisation des sources par répertoires
  - Au tout début, vous n'avez que peu de choses à regarder (répertoires kernel+utility)
  - Nombre de fichiers source à étudier augmente au fil des TPs
- ❑ Vous ne pouvez pas éviter :
  - de lire le **polycopié** et/ou **documentation en ligne**
  - de **regarder les fichiers source** !

## Vue générale (macroscopique) de Nachos



## Arborescence des sources

- ❑ **Machine émulée**
  - Répertoire machine
  - Ne doit pas être modifié !
- ❑ **Noyau** (dans l'ordre d'utilité dans les TPs)
  - Répertoire kernel : ordonnancement, synchronisation, démarrage
  - Répertoire utility : classes utilitaires (listes, bitmap, statistiques)
  - Répertoire drivers : pilotes de périphériques (disque, console, coupleur série)
  - vm : système de gestion de mémoire virtuelle
  - filesys : système de gestion de fichiers
- ❑ **Programmes utilisateur**
  - userlib : mini librairie C pour Nachos (appels systèmes + fonctions utilitaires)
  - test : vos programmes utilisateur (+ quelques tests de base)

## Machine

---

- ❑ Processeur
  - MIPS R2000/3000 (processeur RISC)
  - Registres généraux entiers
    - Registre \$0 : câblé à zéro
    - Registre \$2 : valeur de retour des fonctions
    - Registre \$31 : adresse de retour de fonction
    - Registres \$4, \$5, \$6, \$7 : paramètres des fonctions
  - Registres généraux flottants
  - Registres spécialisés (PC, codes condition, \$sp, \$gp)
- ❑ Contrôleur d'interruptions
  - Interruptions masquables / démasquables
- ❑ Mémoire vive (RAM)

## Machine

---

- ❑ Timer
- ❑ Matériel pour la traduction d'adresse (MMU)
- ❑ Les périphériques
  - Disque
  - Coupleur série
  - Console
  - Méthodes asynchrones d'entrées/sorties
  - Configurables pour que les périphériques demandent une interruption en fin d'entrée-sortie
- ❑ Une classe C++ par élément de la machine

## Machine : fichiers et classes

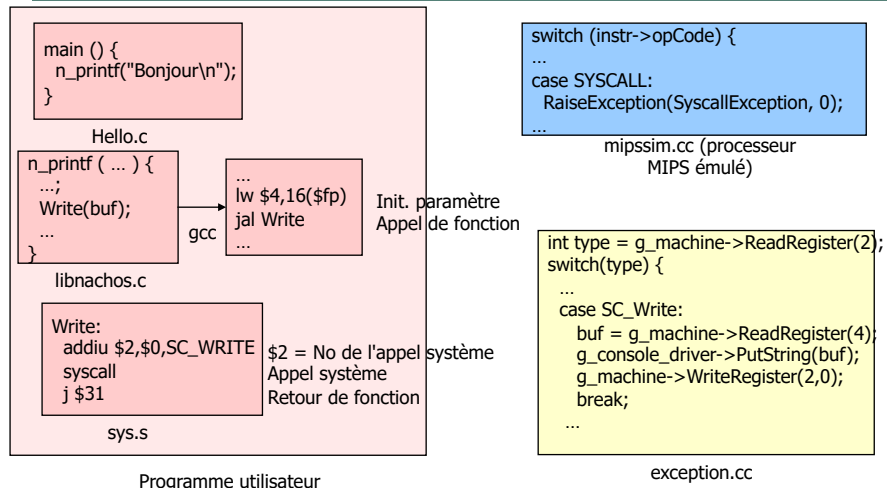
Fichier	Classe(s)	Description
ACIA.cc h	ACIA	Coupleur série
console.cc h	Console	Console
disk.cc h	Disk	Disque
interrupt.cc h	PendingInterrupt Interrupt	Demande d'interruption Contrôleur d'interruption
machine.cc h	Instruction	Décodage d'instructions
mipssim.cc h	Machine	Exécution d'instructions
mmu.cc h	MMU	Traduction d'adresses
translationtable.cc h	TranslationTable	
timer.cc h	Timer	Timer
sysdep.cc h	Aucune	Fonctions C pour assurer l'indépendance avec le système hôte

## Les programmes utilisateur

- ❑ Utilisation d'un compilateur croisé standard (gcc) -> exécutables ELF
- ❑ Librairie d'exécution Nachos (userlib)
  - Liée avec les programmes utilisateur
  - sys.s : code des appels système (déjà écrit)
  - libnachos.c : portage des principales fonctions de la libc
    - fonctions d'entrées-sorties (n\_printf), gestion des chaînes (n\_strcpy, n\_strcmp, ...), conversion (n\_atoi, n\_atof, ...)
- ❑ Fichiers (C)

Fichier	Description
libnachos.cc h	Portage de quelques fonctions de la libc sur Nachos
sys.s	Code des appels système
ldscript.lds	Plan mémoire des exécutables exploité par l'éditeur de liens

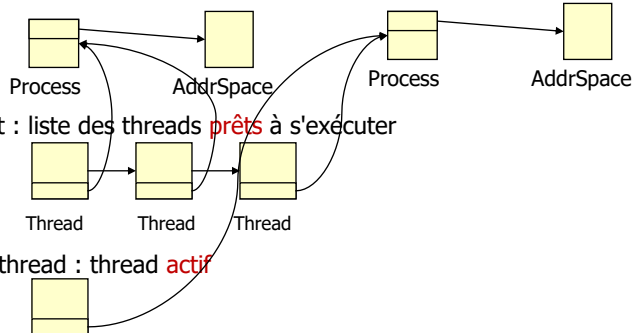
## Noyau Mécanisme d'appel système



13

## Noyau : synchronisation et ordonnancement Structures de données

- Process, Thread, AddrSpace



- g\_alive : liste de **tous** les threads créés (prêts, actifs, bloqués)
- g\_obj\_ids : liste des identificateurs de **tous les objets** (thread, sémaphores, fichiers, ...) : test de validité des types d'argument lors des appels système

14

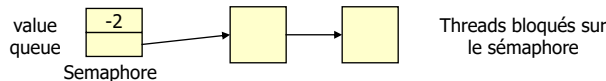
## Noyau : synchronisation et ordonnancement

### Fonctionnement de l'ordonnanceur

- ❑ Ordonnancement FIFO
  - On sélectionne toujours le thread en tête de `g_ready_list`
  - Pas de notion de priorité
  - Pas à la base de partage de temps entre Threads (mais tout est là pour le faire)

## Noyau : synchronisation et ordonnancement

### Outils de synchronisation

- ❑ Sémaphores
  - Structure de données (classe Semaphore)

value  
queue

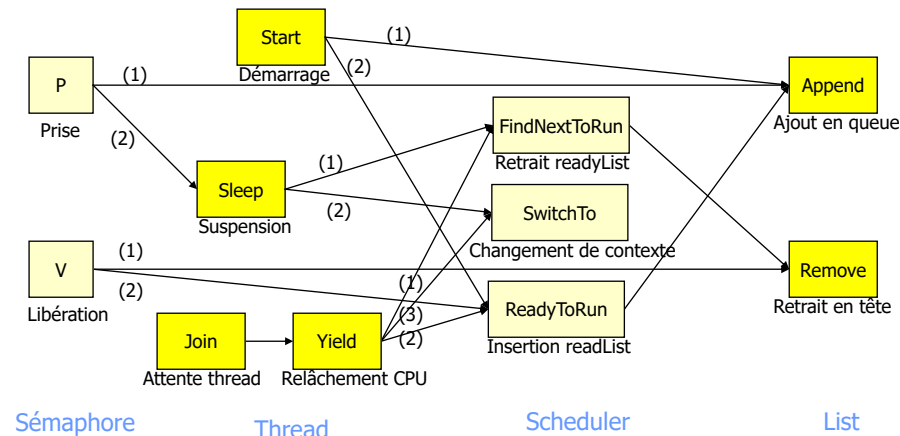
Semaphore

Threads bloqués sur le sémaphore
  - Fonctionnement : sémaphore à compteur classique (voir cours)
  - Exemple : P sur un sémaphore de valeur 0
    - Décrémenter le compteur (champ value)
    - Insérer le thread appelant (`g_current_thread`) dans la file d'attente (champ queue)
    - Endormir le thread appelant (`g_current_thread->Sleep()`)
- ❑ Verrous (Locks) : sémaphores d'exclusion mutuelle
  - Pas de compteur (un booléen suffit)
  - Vérification que celui qui libère le verrou est celui qui l'a acquis



## Noyau : synchronisation et ordonnancement

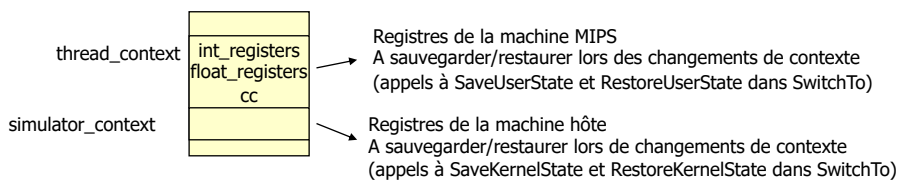
### Graphe d'appel



## Noyau : synchronisation et ordonnancement

### Contexte et changement de contexte

#### Contexte d'un Thread

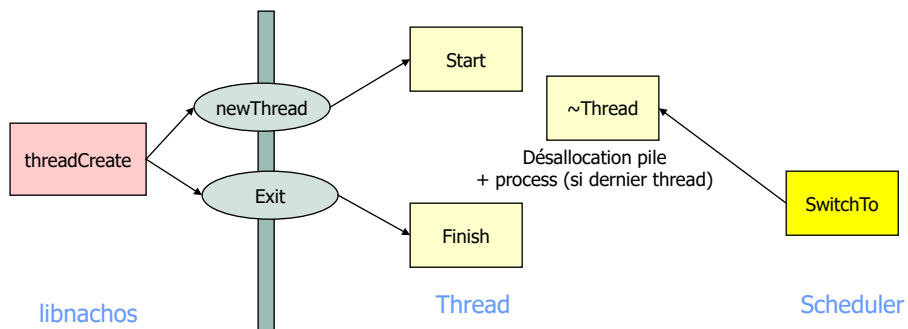


#### Initialisation du contexte à la création d'un thread (thread->Start)

- Allocation pile utilisateur (addrSpace->StackAllocate)
- Initialisation contexte utilisateur (initUserContext)
- Allocation pile noyau (AllocBoundedArray(KERNELSTACKSIZE))
- Initialisation contexte noyau (initKernelContext)
- Insertion du thread dans la liste Alive et dans la file des prêts (scheduler->ReadyToRun)

## Noyau : synchronisation et ordonnancement

### Terminaison et destruction d'un thread



- Finish ne peut pas détruire l'objet Thread et libérer les ressources associées (pile) car le thread s'exécute toujours -> positionnement variable globale threadToBeDestroyed et destruction lors du changement de contexte

## Ordonnancement et synchronisation : classes et fichiers

Fichier	Classe	Description
addrspace.cc h	AddrSpace	Contexte mémoire d'un processus
exception.cc	ExceptionHandler (fn C)	Point d'entrée dans le noyau après un appel système
main.cc	main (fn C)	Point d'entrée dans l'exécutable Nachos
msgerror.cc h	SyscallError	Affichage des messages d'erreur
process.cc h	Process	Processus
scheduler.cc h	Scheduler	Ordonnanceur
synch.cc h	Semaphore, Lock, Condition	Code des outils de synchronisation
syscall.h	aucune	Numéros et interface des appels systèmes
system.cc h	aucune	Démarrage et arrêt du système
thread.cc h	Thread	Thread

## Variables globales

- Relatives à la machine MIPS

Nom	Description
g_machine	Processeur cible (MIPS)
g_machine->interrupt	Etat de la machine vis à vis des interruptions
g_machine->timer	Timer de la machine
g_machine->mmu	Unité de gestion mémoire
g_machine->acia	Coupleur série

- Interrupt, timer, mmu, acia: membres publics de l'objet machine

## Variables globales

- Relatives au noyau Nachos

Nom	Description
g_cfg	Paramètres de configuration du noyau
g_scheduler	Ordonnanceur
g_current_thread	Thread actif
g_thread_to_be_destroyed	Thread dont la destruction est planifiée
g_alive	Liste des threads créés
g_user_obj	Liste de tous les objets (thread, sémas, ...)
g_stats	Objet de mémorisation des statistiques
g_swap_manager	Gestionnaire de swap
g_page_fault_manager	Gestionnaire de défaut de page
g_physical_mem_manager	Gestionnaire de mémoire physique
g_console_driver	Driver console
g_disk_driver	Driver série
g_acia_driver	Driver disque
g_syscallerror	Gestionnaire de messages d'erreur
g_file_system	Système de gestion de fichiers
g_openfile_table	Table des fichiers ouverts

## Logistique

---

- ❑ Ce que j'attends de vous
  - Récupération des sources
  - Examen des sources pour remplir le questionnaire d'exploration
- ❑ Ce que je n'attends pas :
  - Plagiat : recopie des TPs (ou programmes de test) des autres ☹
- ❑ Remarques
  - Questionnaire entre dans la note de TP
  - Le questionnaire n'est qu'un prétexte pour entrer dans le code
- ❑ Où trouver les choses (special Covid)
  - Sources de Nachos : Teams, répertoire *nachos*
  - Questionnaire d'exploration : [nachos/doc/roadmapfrench.pdf](#), avant les sujets de TP
  - Doxygen nachos : <http://www.irisa.fr/alf/downloads/puaut/docnachos/>

## Logistique

---

- ❑ Installation
  - Copier binaire compilateur croisé (Teams, répertoire *nachos*)
    - Attention, binaires 32 bits
  - Modifier *Makefile.config* (une des cibles avec `-DETUDIANTS_TP`) avec le chemin d'installation du compilateur croisé
    - Make compile le noyau et les applis
  - Lancement: *nachos* (sort rapidement, code de synchro non implémenté)
- ❑ Rendu de TP
  - Au fil de l'eau
  - Démo à votre encadrant de TP en séance + « interrogatoire »
  - Sources correspondants à rendre pour « note de style »

## Notions de C++ utiles pour les TPs

---

### ❑ Déclaration de classe

```
class Scheduler {  
public:                                // Champs accessibles en dehors de la classe  
    Scheduler();                      // Constructeur  
    ~Scheduler();                    // Destructeur  
    void ReadyToRun(Thread* thread); // Un exemple de méthode publique  
    int nbthreads;                   // Un exemple de champ public  
private:                             // Champs privés  
    Listint *readyList;              // Un exemple de champ privé  
};
```

### ❑ Création et destruction d'objet

```
Scheduler *mon_scheduler = new Scheduler(); // Allocation et appel du constructeur  
delete mon_scheduler;                       // Appel au destructeur et désallocation
```

### ❑ Appel de méthode et accès aux champs

```
mon_scheduler->ReadyToRun(t);  
mon_scheduler->nbthreads ++;
```