Informatique Embarquée
M2 / 2017

OS / µ-Noyaux / OS TR

Références

- Voir "Programming Under Chorus", Jean-Marie Rifflet
- Merci aussi à Ivan Boule
- L4 / Nicta:

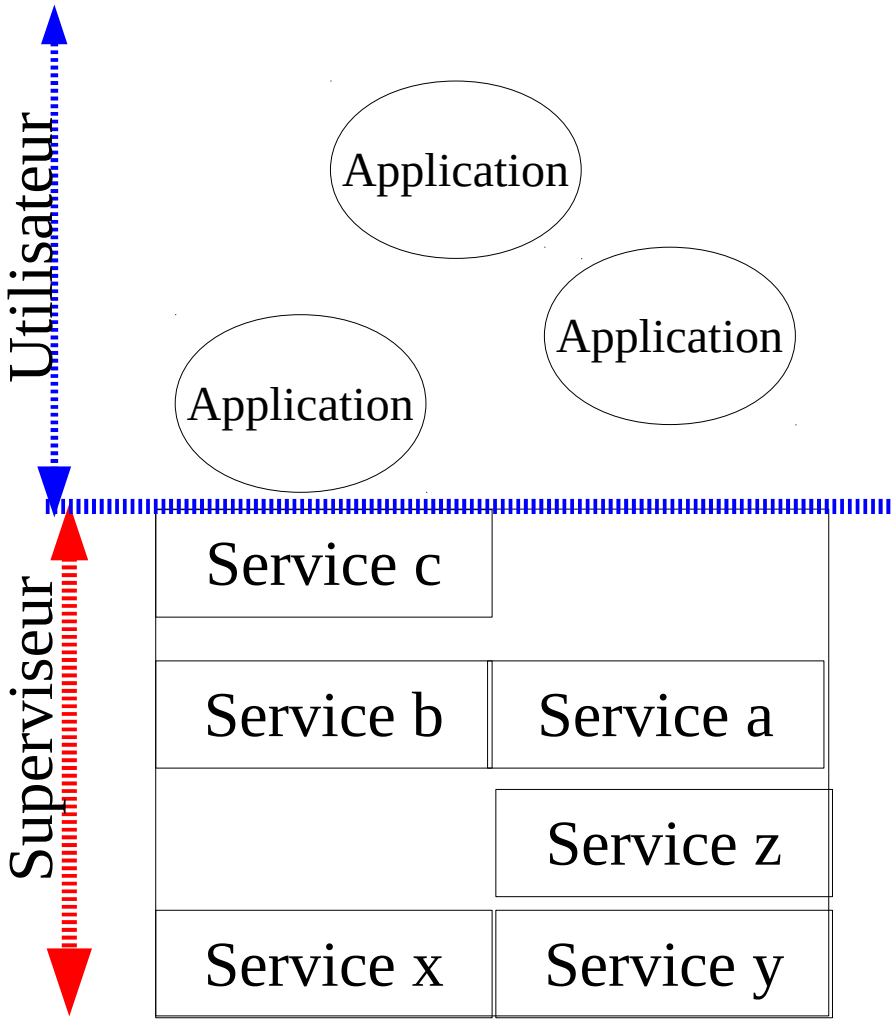
  http://www.cse.unsw.edu.au/~cs9242/10/lectures/01-intro.pdf

Fonctions d'un OS

- Question N°1:
  - Lister les principales fonctions d'un système d'exploitation de type Unix (Linux, BSD, Solaris), Windows ou MacOS
  - Différencier ce qui relève du noyau lui-même de ce qui est fourni par les utilitaires
- Question N°2:
  - Serait-il possible d'architecturer de manière différentes ces mêmes services?

Structure d'un OS monolithique

- L'ensemble des services offerts par le noyau sont intégrés dans une même unité d'exécution en mode privilégié
- Seules les applications sont isolées les unes des autres



Micro-noyaux: Motivations

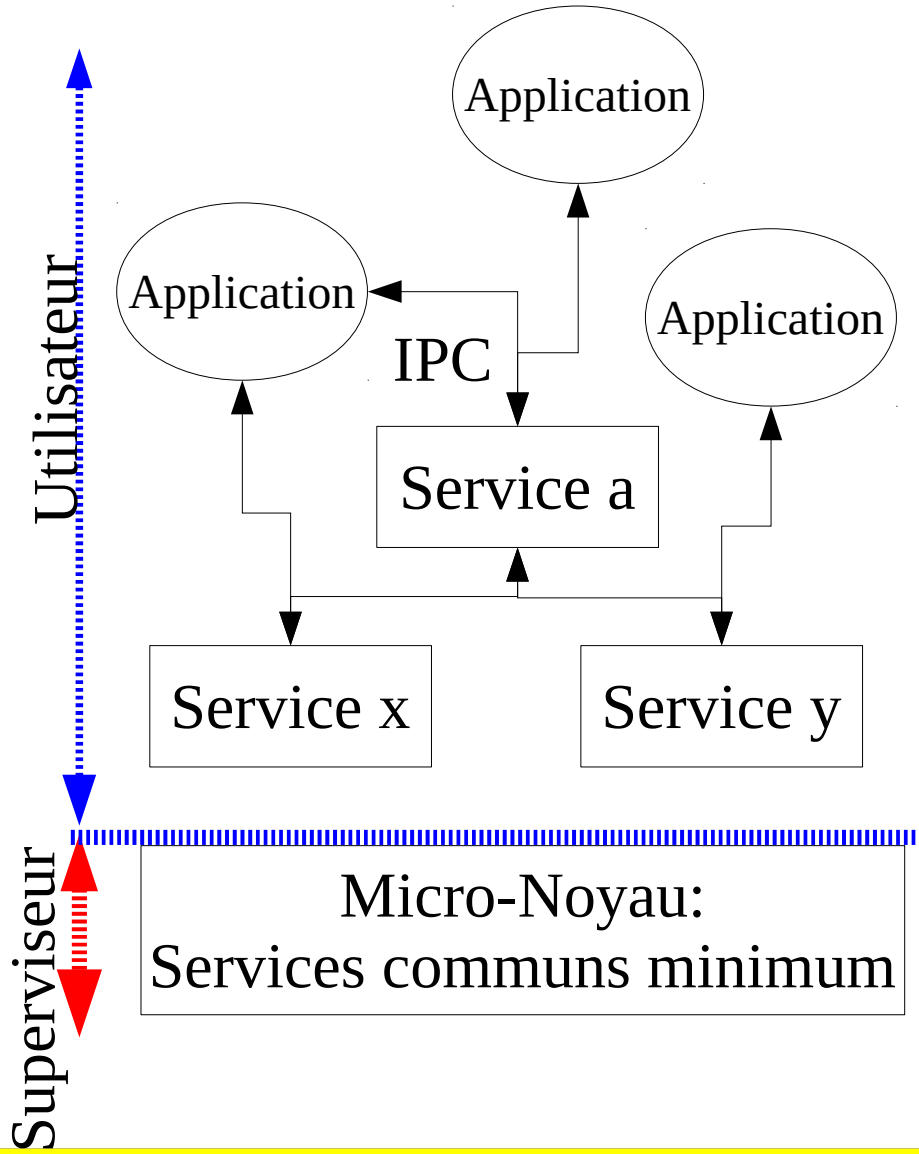
- Les noyaux traditionnels dits "monolithiques" exécutent l'ensemble de leur code en mode superviseur:
  - Tendance à peu / mal définir les interfaces entre les composants,
  - La défaillance d'un composant entraîne la défaillance du système complet,
  - Tout le code doit être considéré / validé comme sûr,
- Pas de nécessité absolue:
  - Gain en performance, "Facilité" d'évolution

Micro-noyaux: Motivations

- Définir une architecture différente:
  - N'exécuter en mode superviseur que ce qui est strictement nécessaire
    - Question: qu'est-il réellement "nécessaire" d'exécuter en mode superviseur?
  - Exécuter les services usuels d'un OS dans des "serveurs" isolés les uns des autres en mode applicatif.
  - D'où la nécessité de définir un mécanisme de communication pour invoquer ces serveurs: IPC.
  - Variante: avec un IPC "réseau" on peut alors construire un système distribué!

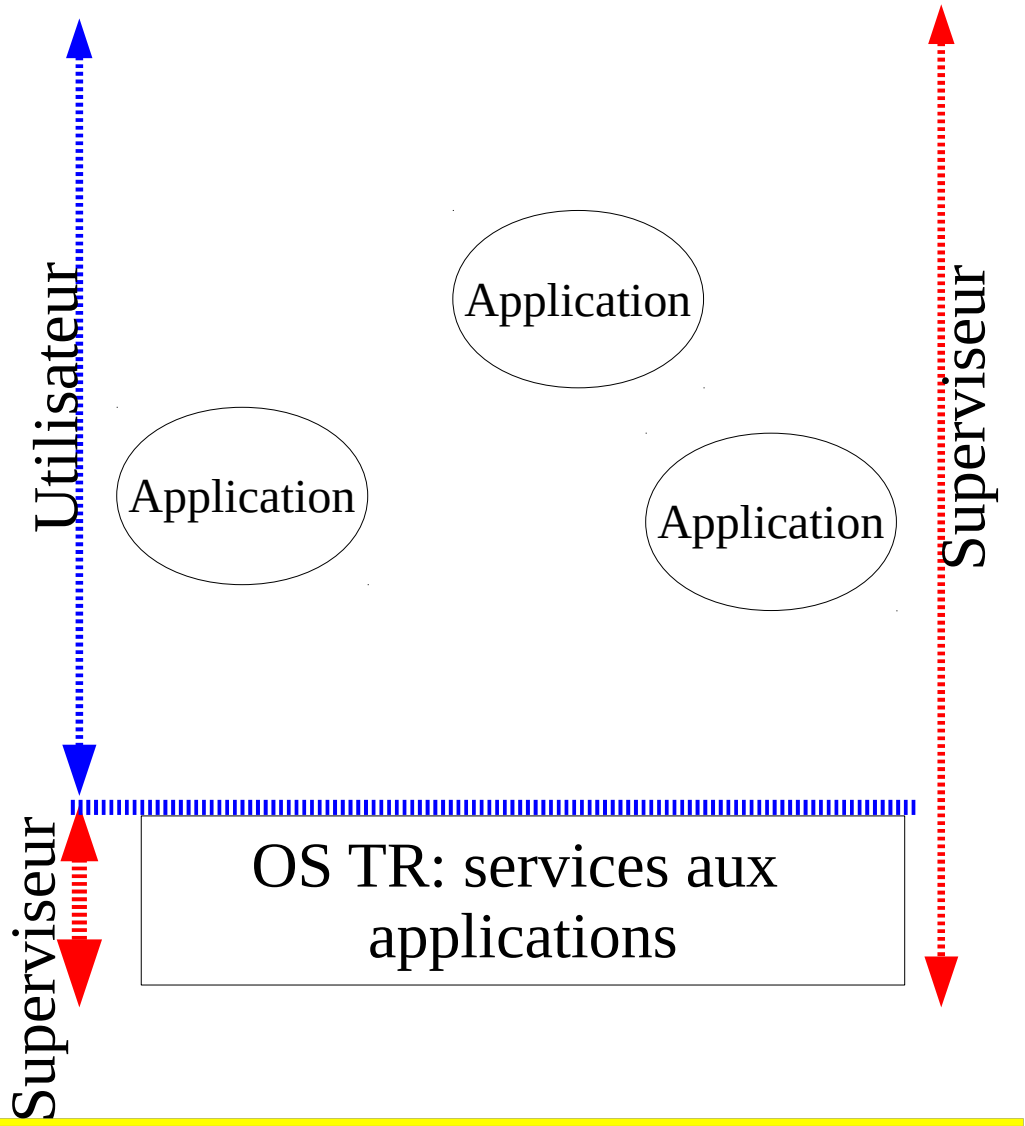
Système à base de Micro-Noyau

- La plus grande partie des services est maintenant rendue par des serveurs (modulaires) s'exécutant en mode utilisateur.
- Cas particulier: tous les services sont rendus par un seul serveur.



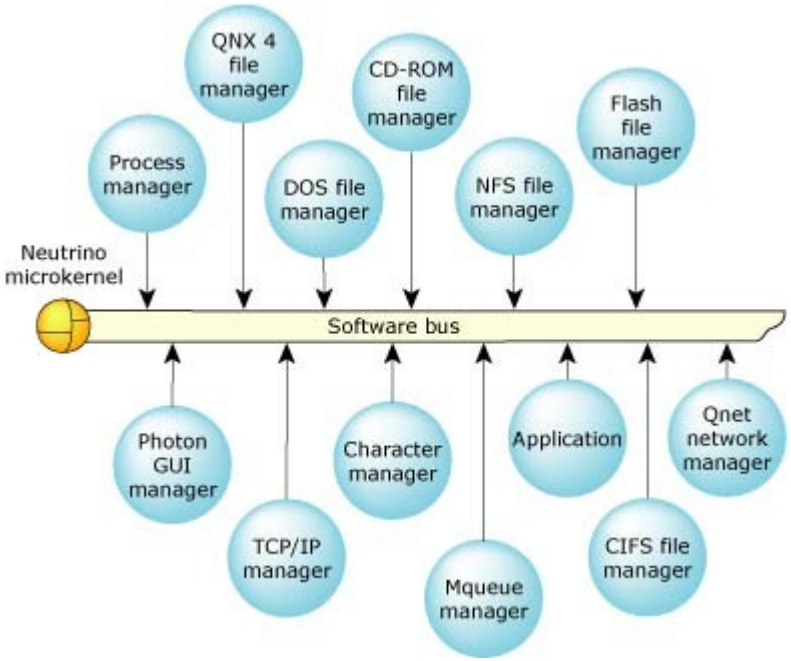
OS Embarqués, Temps-réel

- Le noyau fournit un ensemble de services plus restreint que les OS généralistes (Linux)
- Services aux applications, et non pas services aux serveurs (µ-noyaux)



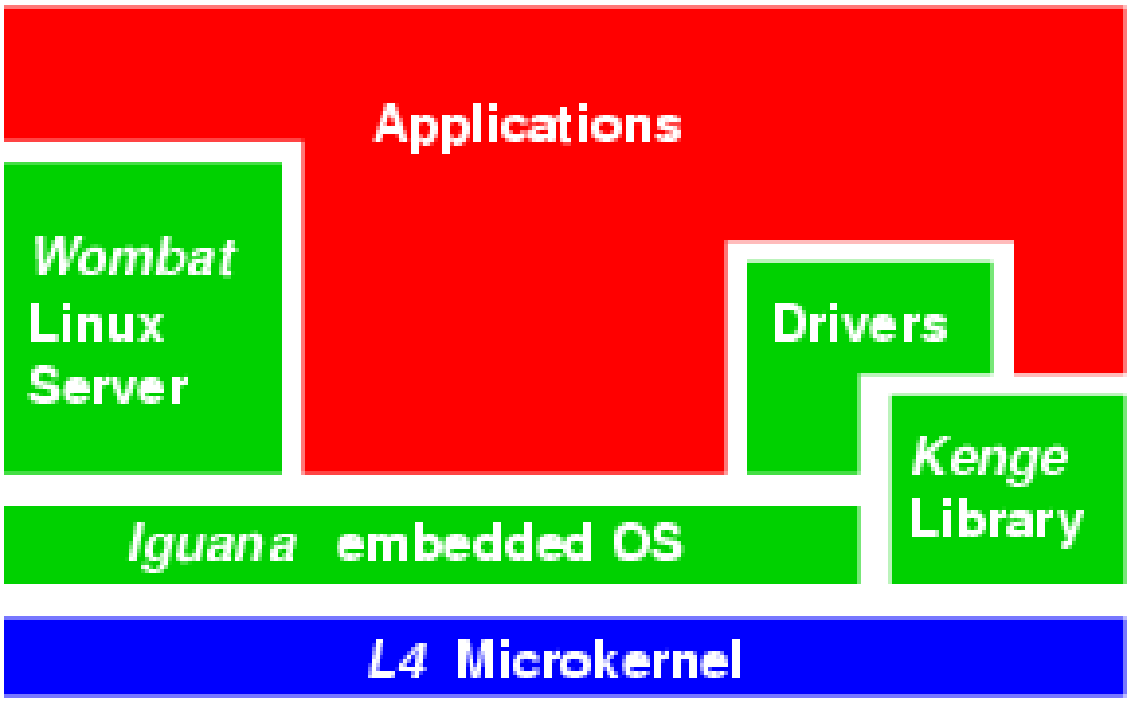
µ-Noyaux: QNX

- QNX:
  - Real-time, Posix
  - Commercial, Open source
- Collection de serveurs
- Très utilisé en "télématique" (automobile)
- Arch: x86, MiPS, PPC, ARM, SH-4...



µ-Noyaux: L4

- ProjetS Open Source
- OKLabs (commercial)
- Nombreuses variantes
- Pas d'IPC distribué
- peu/pas de support temps-réel (suivant les versions)
- Supporte des adaptations du noyau Linux en mode utilisateur



- Versions OKLabs
  - Utilisés Téléphonie
  - + Microviseur

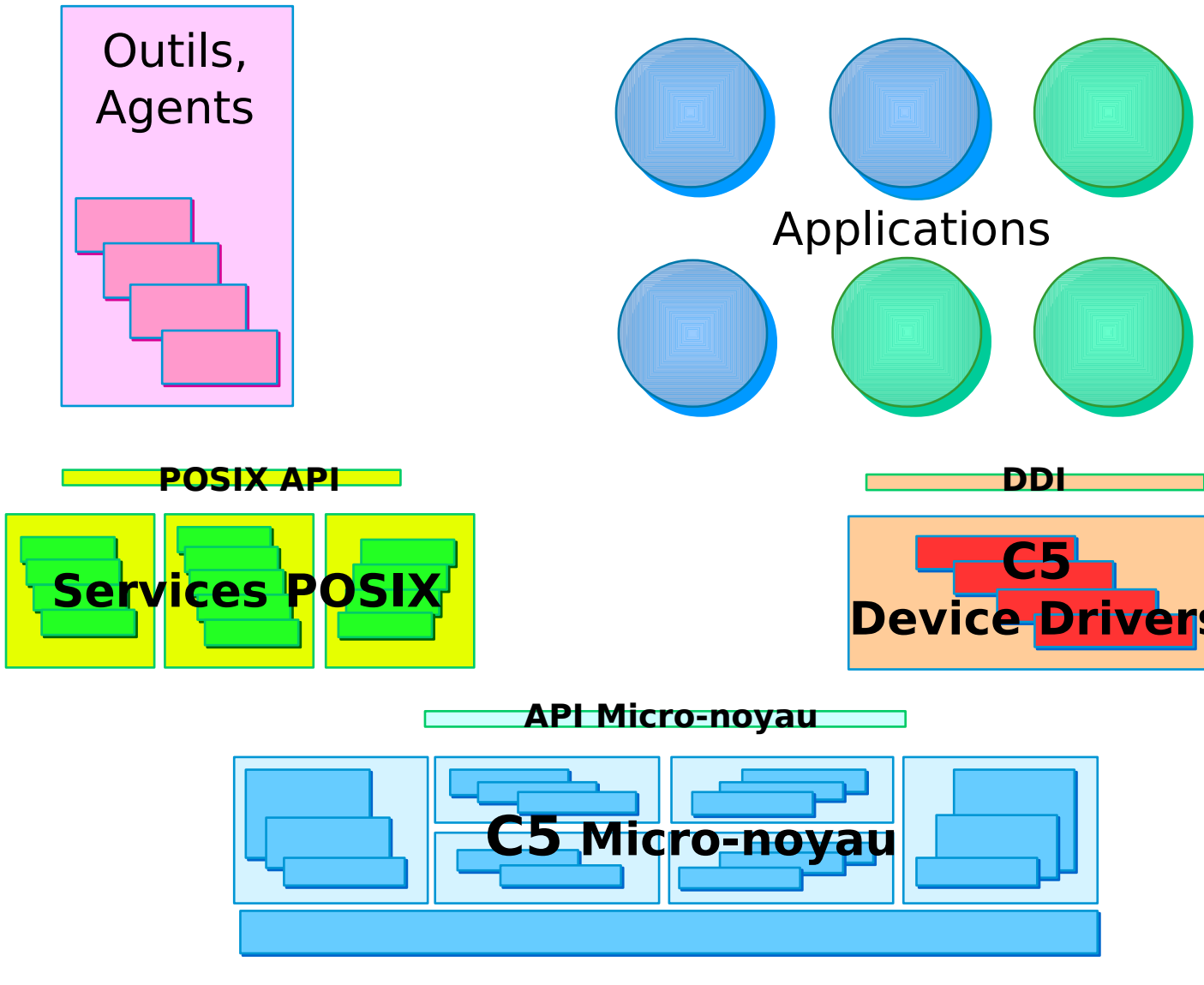
µ-Noyaux: autres

- Mach: (dans MAC OS X)
  - Pas vraiment utilisé sous forme de µ-noyau
  - Ni destiné à l'embarqué ni au Temps-réel.
  - Édition de liens avec un portage de "FreeBSD"
  - IPC typé assez "lourd" restreint au site local
- IBM K42, Microsoft Singularity
- Chorus / C5
  - Commercial / Open Source. Historiquement utilisé dans: embarqué / temps-réel / télecom

OS Temps-réel (non Linux)

- Il y en a pléthore: commerciaux, open-source, faits maison...
- Commerciaux
  - WindRiver VxWorks,
  - Nucleus, pSos,
  - VRTX, ThreadX,
  - WindowsCE,
  - Symbian, RMX,
  - OSE, OS-9, ITron,
  - Ardence RTX....
- Open Source
  - eCOS
  - FreeRTOS,
  - RTEMS,

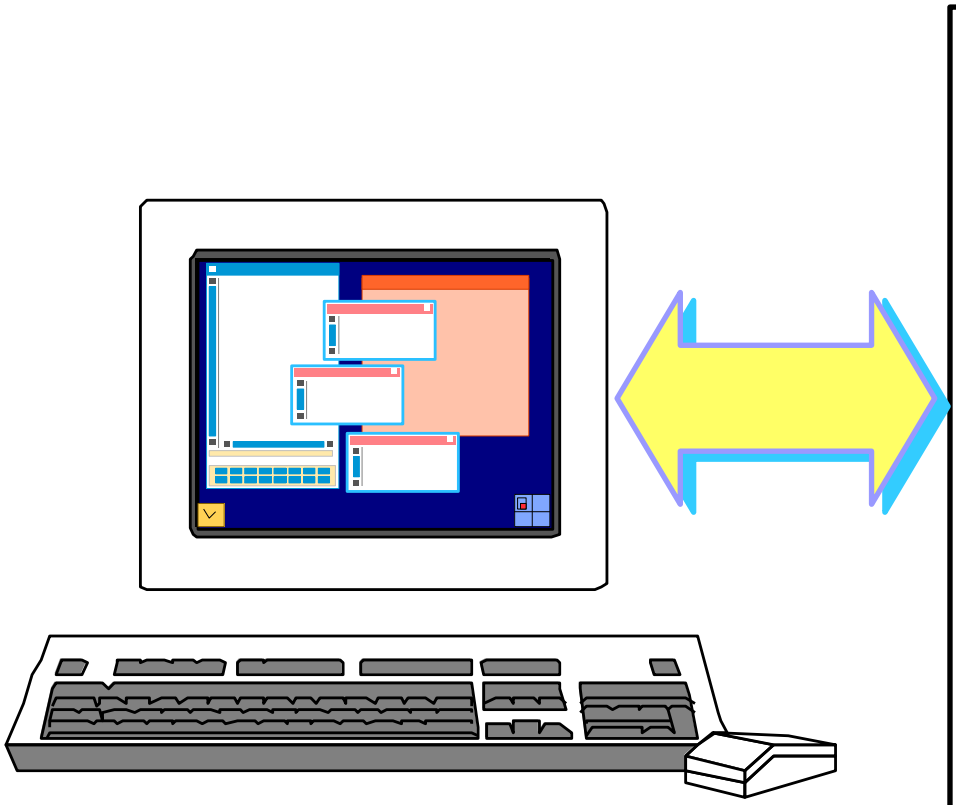
Chorus / C5 - Architecture



ChorusOS / C5 Système Temps-Réel

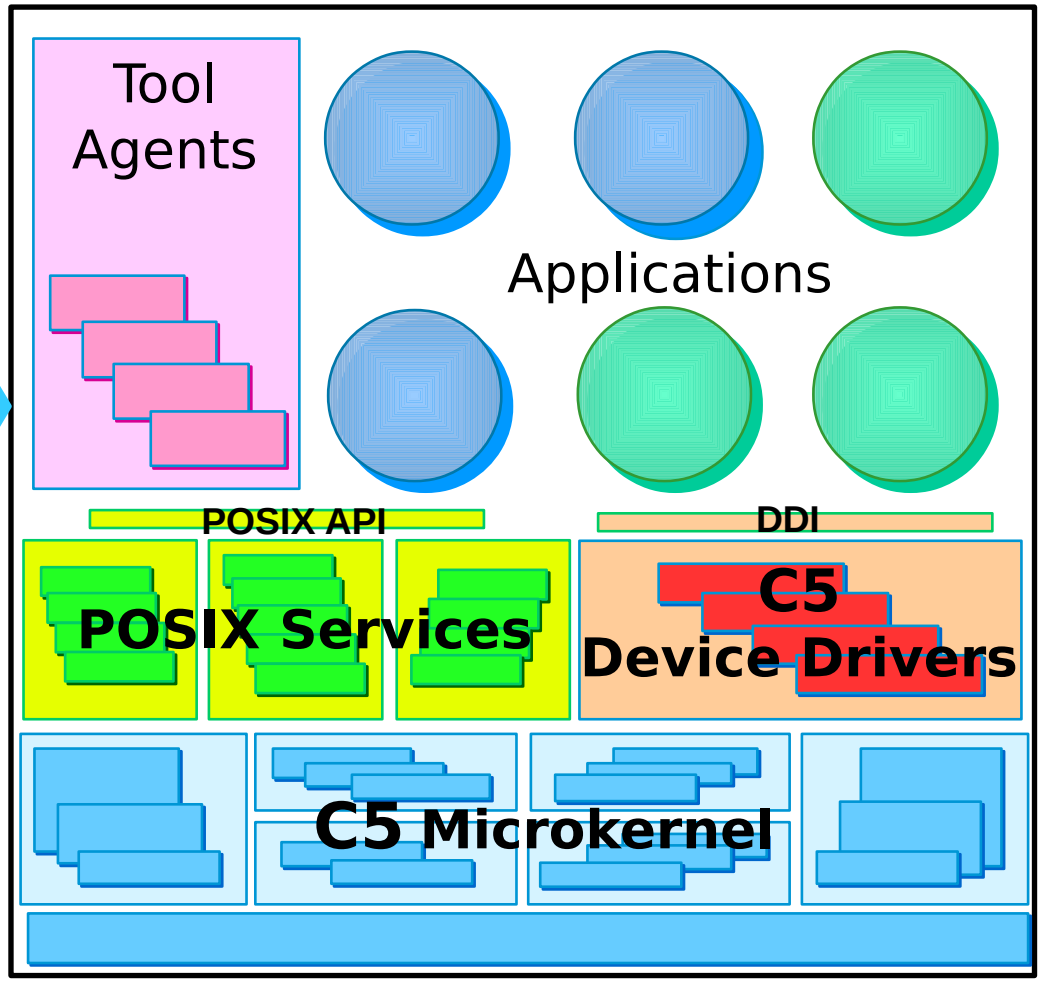
- C5 micro-noyau temps réel (5ème génération Chorus)
  - Verrouillage grain fin (*fine-grain locking*)
  - Protocoles IPC (*exactly-once RPC*)
  - Gestion(s) mémoire flexible
  - Device Drivers Framework
- Sous-système POSIX (dérivé de FreeBSD)
- Commandes d'administration embarquées autonomes
- Environnement Hôte/Cible (*Host/Target*)

Développement croisé



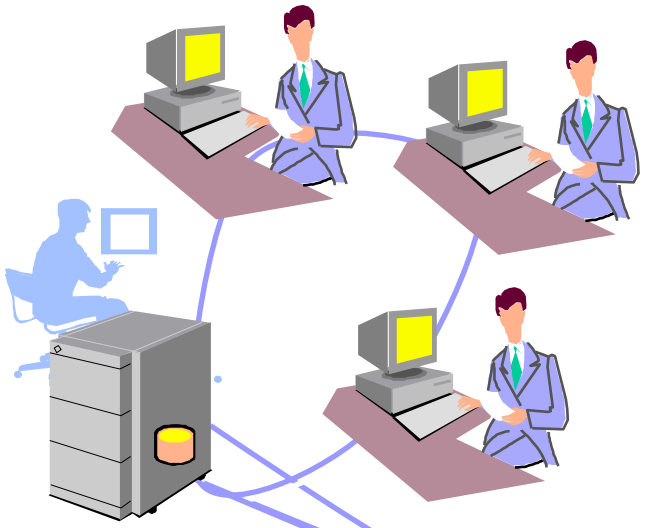
Système Hôte

(Linux / Solaris)



Plateforme cible

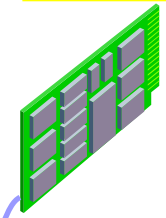
Environnement de Développement

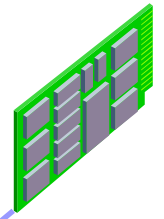


Linux / Solaris

Cibles Embarquées

- Téléchargement applicatio
- Debugger embarqué

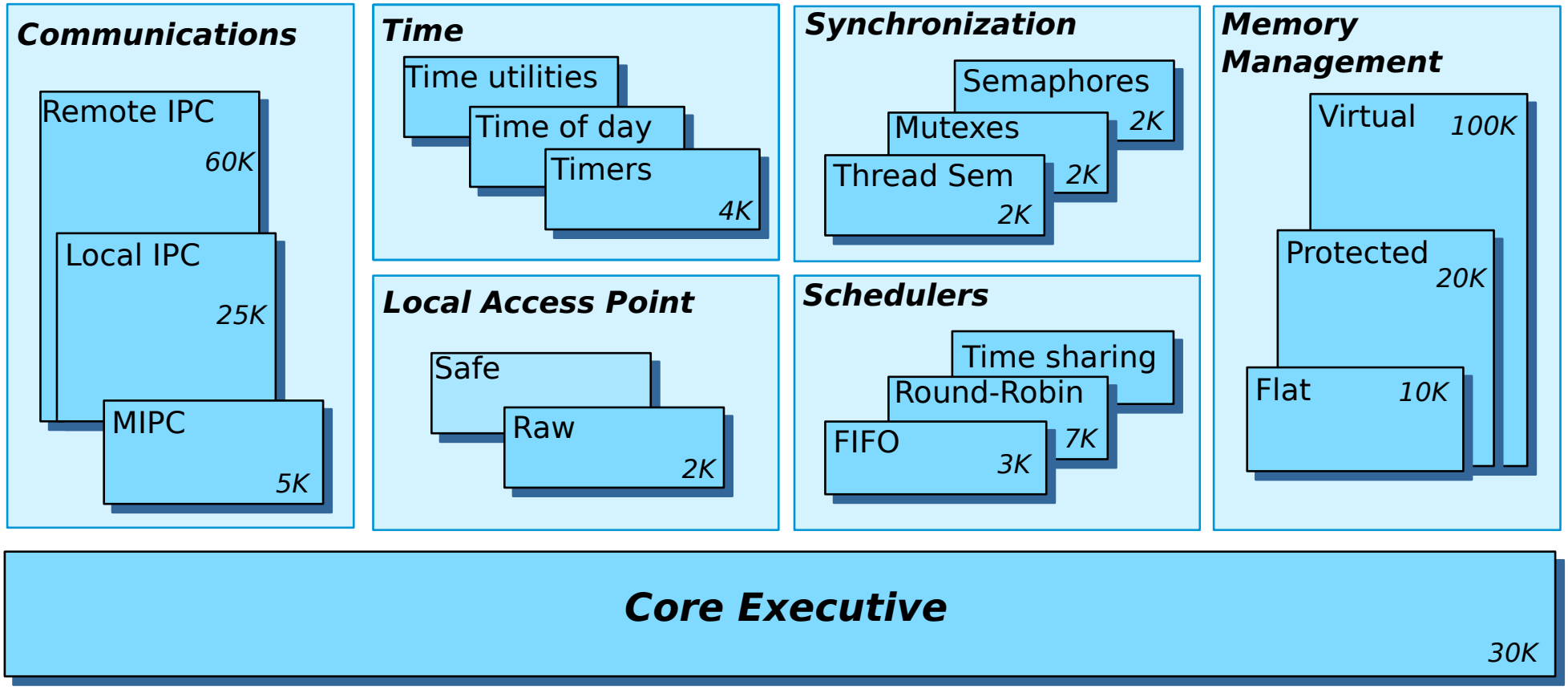




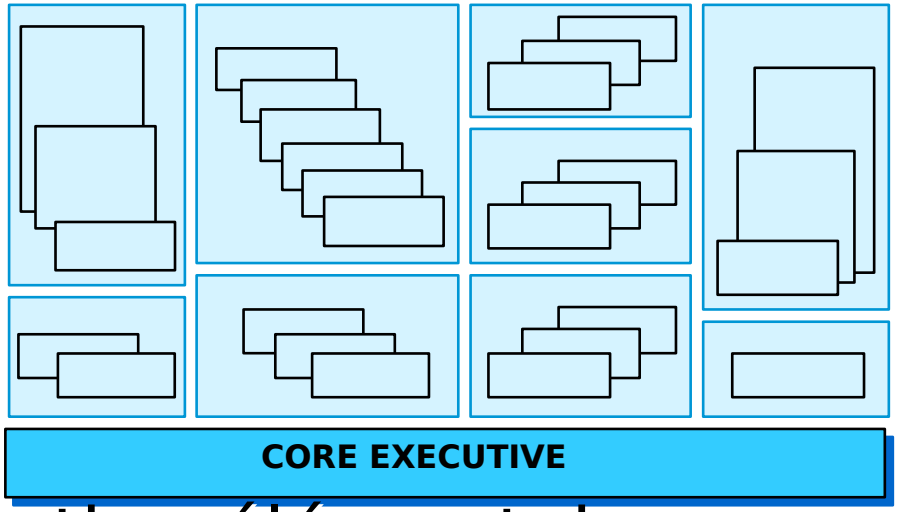
- C5 configuration
- C and C++ Development Toolchain
- C and C++ Symbolic Debugger
- Utilitaires administration
- Jeu de bibliothèques

- Liens:
  - Ligne série
  - Ethernet
  - JTAG

Services du µNoyau C5



Core Executive



- Services de base
  - « acteurs »
  - threads
  - Service de synchronisation élémentaire
- Support des modules µNoyau
  - initialisation
  - per thread/actor data
- Démarrage des acteurs de boot

OKL4

- Les informations qui suivent sont extraites de

  http://www.cse.unsw.edu.au/~cs9242/10/lectures/01-intro.pdf

  - Courtesy Gernot Heiser, NICTA
- Les pages suivantes n'ont pour but que de présenter un rapide survol de OKL4, et n'ont en aucun cas la prétention d'être un introduction complète à ce système.

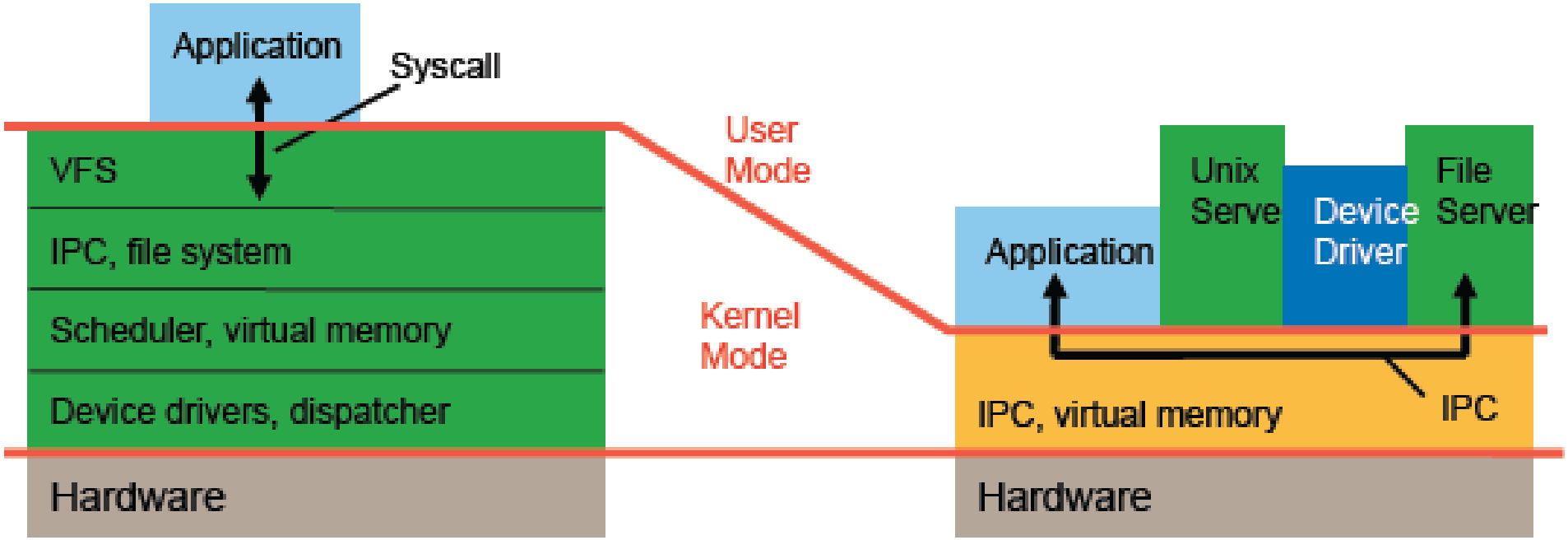
Copyright Notice

UNSW Open Kernel Labs
Be open. Be safe.

These slides are distributed under the Creative Commons Attribution 3.0 License

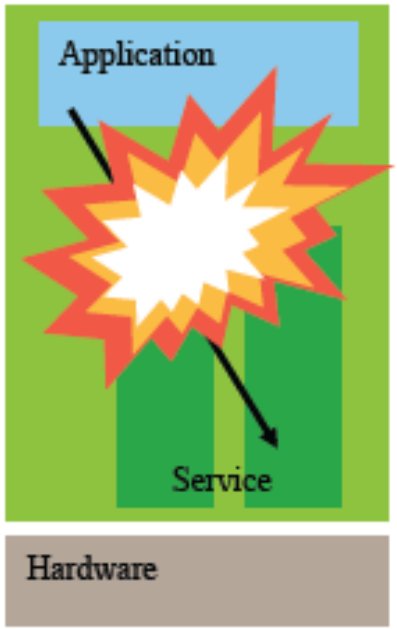
- You are free:
  - **to share** — to copy, distribute and transmit the work
  - **to remix** — to adapt the work
- Under the following conditions:
  - **Attribution.** You must attribute the work (but not in any way that suggests that the author endorses you or your use of the work) as follows:
    - "Courtesy of Gernot Heiser, [Institution]", where [Institution] is one of
    - "UNSW" or "NICTA"
- The complete license text can be found at http://creativecommons.org/licenses/by/3.0/legalcode

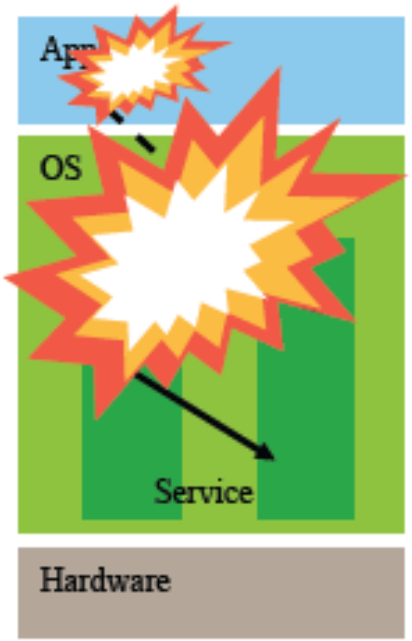
OKL4: Approche µ-Noyau

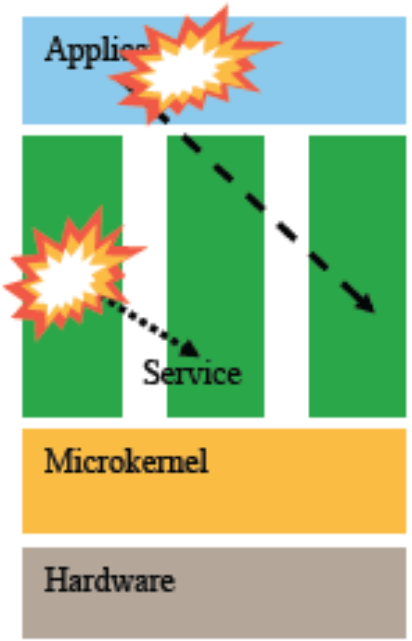


Courtesy Gernot Heiser, NICTA

TCB: Trusted Computing Base







System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	all code	1,000,000's LOC	10,000's LOC

OKL4: Principes de base

- Abstractions:
  - Espaces d'adressage (MMU virtuelle)
  - Threads (CPU virtuels)
  - Capacités -Capability- (nommage et protection)
  - Temps
- Mécanismes
  - IPC : communication par messages
  - Mappage mémoire vers espaces d'adressages
- Concepts autres:
  - Exceptions

OKL4: Espace d'adressage

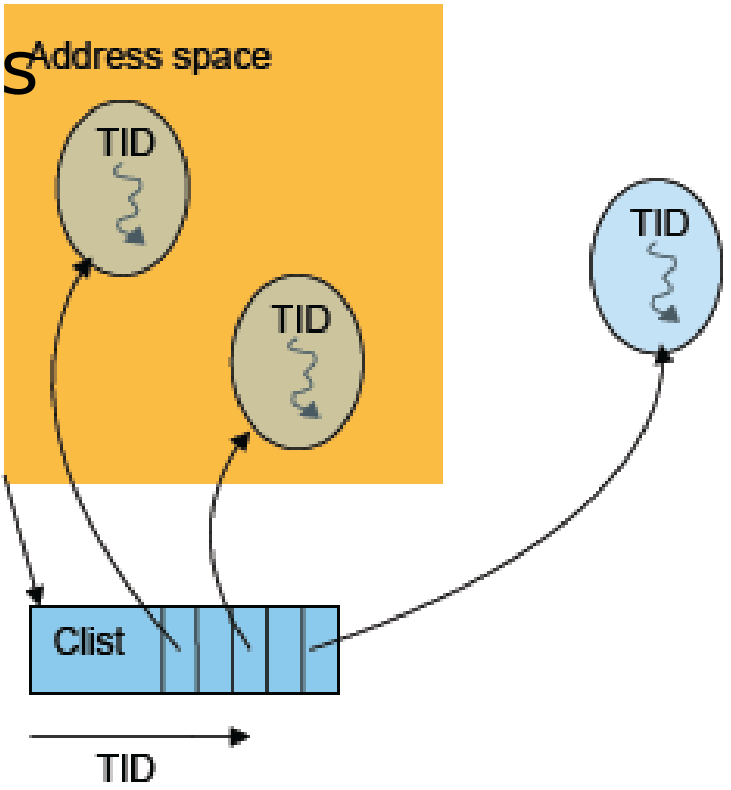
- Espace d'adressage: unité de protection
  - Initialement vide
  - Rempli par « ajout de page frame »
- Ajout/ retrait zones mémoire
  - via MapControl() (appel au µNoyau)
  - Initialement seulement via tâche racine privilégiée
    - (Sorte de init)
  - Versions > 2.2:
    - Plus de tâche racine, plus d'appels systèmes privilégiés
    - Utilisation des « Capacités » (voir plus loin)

OKL4: Threads

- Thread: Unité d 'exécution
  - Ordonnancée par le µ-Noyau
- Source/destination des IPC
  - Souvent on utilise des « ports » (indirection)
- La capacité -nom local- utilisée pour nommer la thread
  - Détermine les droits d'émission / réception
  - Appelée Thread Id (raison historique)
- Associée à un espace d'adressage
- Paramètres d'ordonnancement

OKL4: Capacités

- Capacité:
  - Désigne les threads
  - Index local dans la liste des capacités de l'espace d'adressage.
- Détermine les droits
  - Envoi de message à destination de la thread
  - Autres
- Nom local d'une ressource globale

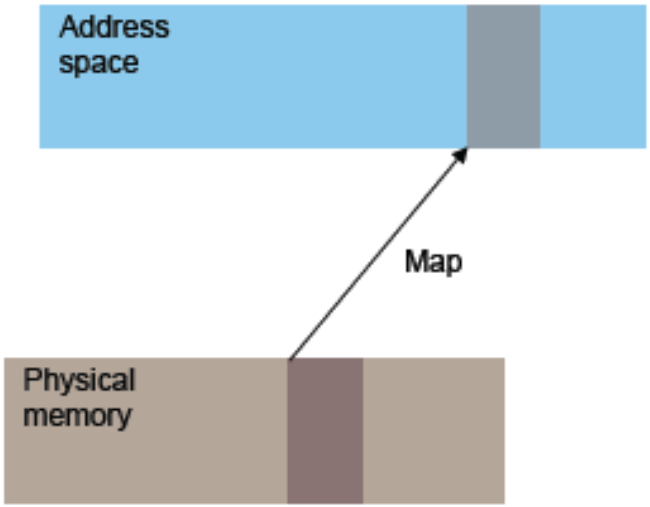


OKL4: IPC

- Envoi de message synchrone
- Copie directe des données de l'émetteur au récepteur
- Messages courts transférés via registres
- Messages longs transférés par le µ-Noyau
- Appel réception bloquant ou par « poll »
- Il existe une variante asynchrone

OKL4: Mappage mémoire

- Association entre une page physique et une adresse virtuelle
  - MapControl()
  - Capacité Mémoire
- Faisable / Fait depuis espace utilisateur
  - Délégation (d'une partie) de la gestion mémoire hors du µ-Noyau



OKL4: Gestion des Exceptions

- Interruptions:
  - Envoi de messages depuis des « pseudo-threads » matérielles
  - Vers des threads pré-déclarées (mode utilisateur)
- Fautes de page
  - Envoi d'un message de la thread fautive vers une thread du serveur responsable de la gestion de l'espace d'adressage (appelé pager)
  - Message réponse « intercepté » par µ-Noyau
- Autres exceptions: mécanisme similaire

OKL4 2.1: API

- 9 Appels systèmes privilégiés
  - Contrôle des ressources
  - Peuvent seulement être invoqués par tâche racine
- 7 appels non privilégieés
  - Services fournis aux applications
  - Utilisables par toute application
- 3 protocoles de communication
  - Communication µ-Noyau – espace utilisateur
  - IPC support des exceptions

OKL4 2.1: API

- Appels privilégiés:
  - ThreadControl
  - SpaceControl
  - MapControl
  - CapControl
  - MutexControl
  - InterruptControl
  - SecurityControl
  - CacheControl
  - PlatformControl
- Appels non-privilégiés:
  - ExchangeRegisters
  - Ipc
  - Schedule
  - ThreadSwitch
  - Mutex
  - MemoryCopy
  - SpaceSwitch
- Protocoles
  - Page Fault
  - Interruption
  - Exception

OKL4 2.1: Threads

- Habituellement une thread représente
  - Abstraction d'exécution définie par
    - Registres (généraux et registres d'état) et une Pile
- Une thread OKL4 dispose aussi de:
  - Registres Virtuels *(vrais registres ou mémoire selon ABI)*
    - Définis par le µ-Noyau, visibles depuis application
    - Thread Control Registers + Message Control Registers
  - Priorité (ordonnancement), quantum de temps
  - Espace d'adressage

OKL4 2.1: Threads

- Etat: Thread Control Block
  - KTCB: état géré et accessible seulement par le µ-Noyau
  - UTCB: état visible par application
    - Ne nuit pas à la sécurité
    - Modifié seulement via les librairies appropriées

OKL4 2.1: Gestion des threads

```
L4_ThreadControl (
  thread, /* new TID */
  addr_spc, /* A.S. to create thread in */
  scheduler, /* scheduler of new thread */
  pager, /* pager of new thread */
  exc_hdlr, /* exception handler */
  resources, /* thread resources */
  utcb); /* utcb address
```

- Création: tid + AS
- Destruction: Tid + AS = Nil

OKL4 2.1: Activation Thread

```
L4_ExchangeRegisters(
    L4_ThreadId_t target,  L4_Word_t control,
    L4_Word_t sp, L4_Word_t ip, L4_Word_t flags,
    ......);
L4_Start_SpIp( L4_ThreadId_t thread,
  L4_Word_t sp,  L4_Word_t ip); /* Bilbiothèque */
```

- Permet un contrôle fin sur les registres.
  - Seul moyen de définir le contexte initial: SP / IP
- Par ailleurs, le µ-Noyau ne gère pas les extensions automatiques de pile utilisateur

OKL4 2.1 : IPC

- Échange de messages toujours synchrone (rendez-vous)
  - Transfert effectué seulement quand l'émetteur et le récepteur sont prêts!
  - Le premier est bloqué jusqu'à temps que l'autre soit prêt.
- Bénéfices:
  - Synchronisation implicite
  - Pas de bufferisation des données dans le µ-Noyau
  - Données copiées au plus une fois

OKL4 2.1 : IPC

- Unique appel système IPC:
  - Inclut un envoi et une réception de message, chacun étant atomique et optionnel
- Opération d'émission spécifie une thread destinataire
- Opération de réception peut :
  - Spécifier un émetteur particulier (“closed receive”)
  - Attendre des messages quelconques (“open wait”)
- Émission / Réception peuvent être:
  - Bloquante (jusqu'à ce que l'autre soit prêt)
  - « polling » - erreur si l'autre n'est pas prêt

OKL4 2.1 : IPC messages

- Données contenues dans les
  - Message Registers (entre 8 et 64)
- Registres Virtuels
  - Utilisent soit de vrais registres matériels, soit UTCB
- IPC: effectue une simple copie des données
  - Registres matériels (pas de copie!)
  - Permet une implémentation légère et rapide pour ces cas optimisés.
- Messages plus grands => MemoryCopy

OKL4 2.1: Ordonnancement

- Priorités fixes de 0 à 255 + Round-Robin à priorités égales
- Ordonnanceur invoqué:
  - Sur expiration de time slice, Yield
  - Blocage / déblocage (IPC call par exemple)
- « Héritage d'ordonnancement »: IPC, mutex
- Appels:
  - Schedule (manipulation des paramètres)
  - ThreadSwitch (don du temps restant à une thread X)

OKL4 2.1 : Création d'un « processus »

- Création d'un nouvel espace d'adressage (AS) :
  - Appel : SpaceControl() +...
- Mappage mémoire dans AS :
  - Appel : MapControl()
    - Code, données, pile
    - Attention, le µ-Noyau n'a pas de SGF!
- Créer une thread
  - Appel :ThreadControl()
- Démarrer la 1ère thread
- Appel : ExchangeRegisters()

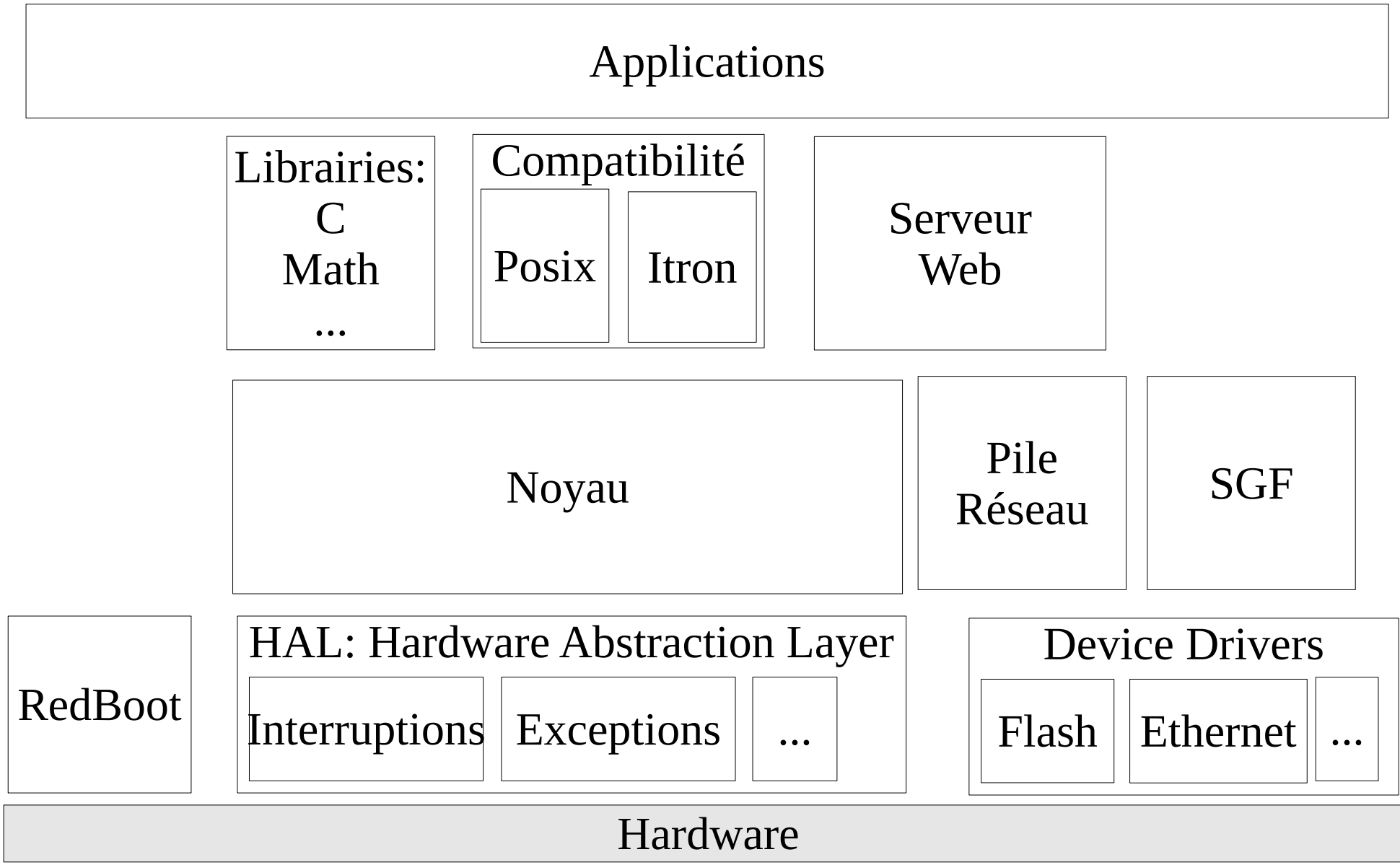
Ecos: références

- http://ecos.sourceware.org
- EMBEDDED SOFTWARE DEVELOPMENT WITH ECOS™ Anthony J. Massa *(PRENTICE HALL)*

eCos

- Système embarqué Temps-réel Open Source
- Processeurs supportés:
  - ARM
  - Intel x86
  - MIPS
  - PowerPC
  - Fujitsu FR-V, Hitachi H8/300, Matsushita AM3x
  - NEC V8xx, Samsung CalmRISC16/32, SuperH
  - SPARC, SPARClite

eCos


Applications
Librairies:
C
Math
...
Compatibilité
Posix
Itron
Serveur
Web
Noyau
Pile
Réseau
SGF
RedBoot
HAL: Hardware Abstraction Layer
Interruptions
Exceptions
...
Device Drivers
Flash
Ethernet
...
Hardware

Services eCos

- Applications multi-thread
  - Pas de notion de tâche ou processus
  - Espace d'adressage commun avec le noyau eCos
    - Pas de protection!
- Principaux Services
  - Gestion de threads
  - Ordonnancement
  - Synchronisation
- Système de fichiers, réseau : paquetages additionnels.
- Écrit en C++, API en C

eCos : principe

- Application doit fournir mémoire nécessaire pour les descripteurs systèmes et la pile
  - eCos n'alloue pas de mémoire dynamiquement
  - => Plus simple, plus déterministe.. mais rudimentaire.

Ecos : création de Thread

- Création:
  - `cyg_thread_create(prio, fn, arg, "name", stack, stacksize, &thHandle, &thDesc);`
  - `void fn(cyg_addrword_t data) {...}`
- Thread créée en état « suspendu »
  - `cyg_thread_resume (thHandle);`
- Pas de tests de débordement de pile, pas d'extension automatique de pile

eCos Thread services

```
void cyg_thread_yield(void);

void cyg_thread_delay(cyg_tick_count_t
delay);

void cyg_thread_suspend(cyg_handle_t
thread);

void cyg_thread_resume(cyg_handle_t
thread);

void cyg_thread_release(cyg_handle_t
thread);
```

eCos Thread services

```
cyg_thread_self
cyg_thread_idle_thread
cyg_thread_get_stack_base
cyg_thread_get_stack_size
cyg_thread_measure_stack_usage
cyg_thread_get_next
cyg_thread_get_info
cyg_thread_get_id
cyg_thread_find
```

eCos Thread services

cyg_thread_exit

cyg_thread_kill

cyg_thread_delete

cyg_thread_get_priority

cyg_thread_get_current_priority

cyg_thread_set_priority

Ordonnancement

- MultiLevel Queue
  - Priorités fixes 0 à 31
  - Round Robin
- BitMap
  - Priorités fixes 0 à 31
  - 1 thread max par niveau de priorité

eCos Synchronisation

- Mutex
  - Non récursifs, héritage de priorité optionnel
- Sémaphores
- Events
- Compteurs
- Mailboxes
- Alarmes, horloges