Systèmes Embarqués Portables

TELECOM Nancy 3A Apprentissage / 2A LE

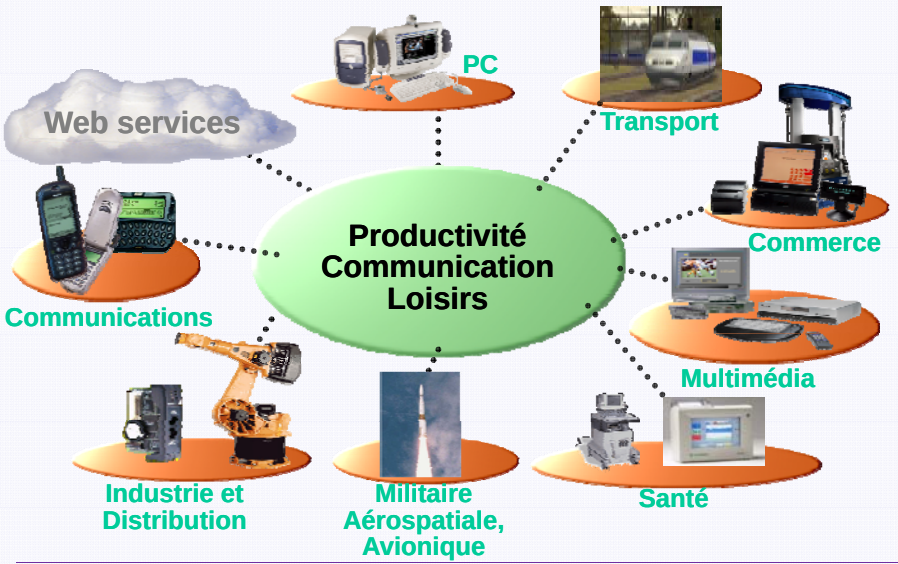
Vincent Bombardier
(MdC HC 61ème Section)

Centre de Recherche en Automatique de Nancy -UMR CNRS 7039-
Département : Ingénierie des Systèmes Eco-Techniques
Projet: Systèmes Intelligents Ambiants

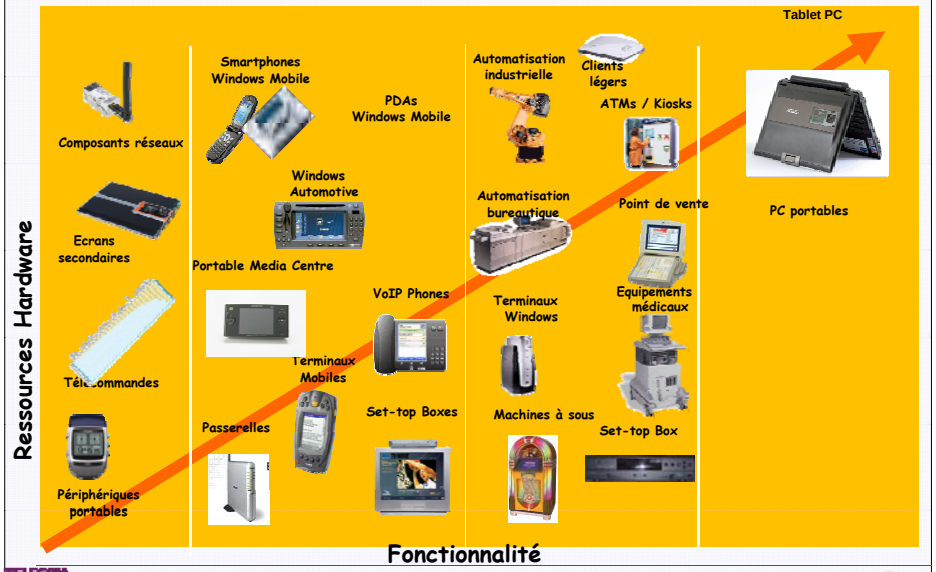
Présentation des Systèmes Embarqués Portables

- **Objectifs du Module**
  - 6h CM, 8h TD, 8hTP : HP iPAQ 614c
- **Présentation**
  - Définitions, Contraintes
- **Mécanismes Fondamentaux**
  - S.E. Multitaches, Synchronisation
- **Windows NT**
  - Historique, Noyau, Gestion mémoire, Threads, Sémaphores, GDI
- **Windows CE 6.0**
  - Plateform Builder, Structure du noyau
- **Windows Mobile 6.0**
  - Spécificités, Environnement de Développement

Présentation : *Un Monde de périphériques*



Présentation : *Evolution*



Présentation : *Définitions*

➢**Système Embarqué (embedded system):**

- Un système embarqué est un système numérique qui intègre des logiciels et des matériels conjointement et spécifiquement pour assurer des fonctionnalités. Il est autonome et ne possède pas forcément de périphériques d'E/S standard.
- An "embedded system" is any computer system or computing device that performs a dedicated function or is designed for use with a specific embedded software application.
- Embedded systems may use a ROM-based operating system or they may use a disk-based system, like a PC. But an embedded system is not usable as a commercially viable substitute for general purpose computers or devices.

➢**Embarqué dans quoi?:**

- Dans les matériels informatiques enfouis pour le militaire (missiles, s/s marins), la science (appareils de mesures, satellites,…) et le grand public (voitures, GSM, HiFi, PDA,…)

Systèmes Embarqués Portables ????

Présentation : *Enjeux Economiques*

- **Les Systèmes Embarqués sont d'une importance stratégique pour l'économie**
  - Facteur d'innovation et de différentiation : nouvelles fonctionnalités et nouveaux services dans les produits existants, nouveaux produits et services
  - Source principale de valeur ajoutée : surtout les logiciels embarqués
  - Amélioration de la compétitivité
- **Forte croissance du secteur – taux annuel prévu pour la période 2009-2014 *(source: Venture Development Corp.)***
  - 7,3% des développeurs de matériels et de logiciels pour systèmes embarqués
  - 8,7% du nombre des développeurs de logiciels
  - 18,9% du marché mondial des OS pour systèmes embarqués
    - **1 billion dollars !!**

Présentation : *Besoins des SE*

- Intégration massive de composants embarqués répartis - systèmes de systèmes communicants pour bâtir la société de l'information :
  - *téléphones cellulaires – PDA – machines programmables – automobile - appareils médicaux – photo/vidéo/hifi – électroménager – avionique - spatial – jouets*
- Outils théoriques et pratiques pour l'intégration permettant de prendre en compte tous ces critères à la fois *(recherche d'optimalité, par rapport au marché ciblé)*
  - *Fonctionnalité*
  - *Qualité : Robustesse et Performances*
  - *Contraintes matérielles (poids, encombrement, ..)*
  - *Coût global, consommation électrique*
  - *Délai de mise sur le marché*
- **Construire des systèmes de fonctionnalité et qualité déterminée et garantie, à coût acceptable, est un défi technologique et scientifique majeur.**

Présentation : *Caractéristiques*

➢**Différents types de Système d'Exploitation:**

- Basic (microcontroleur 1.4 billion en 2012 dont 45% 8bits) :
  - IHM réduite IMC privilégiée (TR)
- Evolué (microprocesseur) :
  - IHM Développée (interfaces graphiques)

➢**Différentes contraintes:**

- Machines et logiciels = partie intégrante du matériel??
- Ergonomie - Packaging
- Facilité d'utilisation
- Sureté de fonctionnement et de conception
- Fiabilité
- Contraintes temporelles « dures »

Présentation : *Architecture du SE*

➢Blocs fonctionnels pas forcement toujours présents:

Présentation : *Contraintes du SE*

Secteur d'activité	Contraintes
Équipements scientifiques	Performances, fiabilité, coût
Équipements militaires et aérospatiaux	Performances, fiabilité, pérennité, intégration
Transports	Fiabilité, coût, interactivité
Informatique industrielle	Fiabilité, coûts, pérennité
Matériel de bureau	Performances, coûts, standardisation
Réseau et téléconnunications	Performances, fiabilité, intégration
Électronique grand public	Performances, coûts, design / intégration

Présentation : *Contraintes du SE*

Besoin	Minimum	Petit	Moyen	Haut de gamme	PC embarqué	Embarqué haute disponibilité
Taille RAM	<0,1Mo	0,1 – 1Mo	2 – 8Mo	8 – 32Mo	> 64Mo	>xMo
Taille ROM / Flash	0,1 – 0,5Mo	0,5 – 2Mo	2 – 8Mo	8 – 32Mo	> 64Mo	>xMo
Processeurs	Microcontrôleurs (entre 4 et 20 Mhz) Mcore Coldfire ARM		Hitachi SH X86 PowerPC			Pentium PowerPC
Vitesse	16 – 80Mhz		50 – 150Mhz	100 – 400Mhz	600 – 3Ghz	>3Ghz
Exemple d'applications	Cartes sans contact, GSM		Télécommunications (Modem routeur ADSL)	PDA, lecteur / enregistreur mp3	Équipement réseau, imprimante	Routeur haute performance, commutateur téléphonique

Présentation : *Contraintes du SE*

➢**Contraintes Matérielles:**

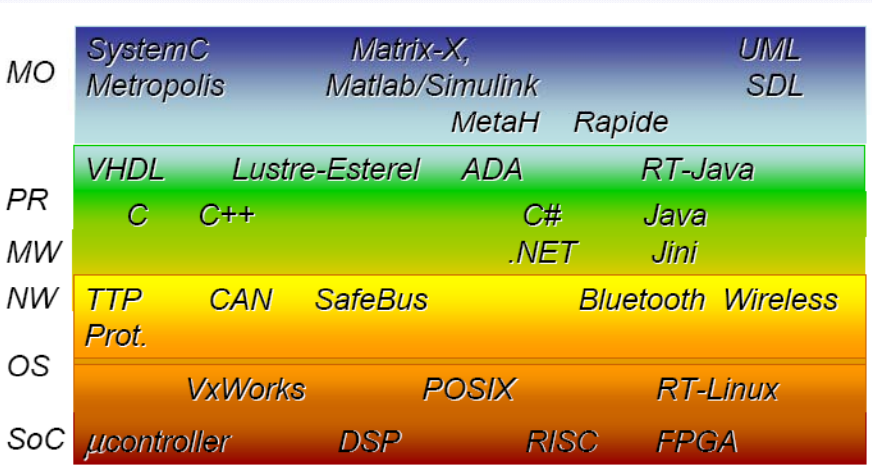
- Poids - Encombrement
- Cout global
- Gestion dédiée et minimaliste de :
  - Mémoire
  - Processeur
  - Alimentation

➢**Contraintes de développement:**

- Absence d'outils conventionnels
- Fort couplage à l'électronique embarquée
- Conception orientée objet
- Algorithmes tenant compte des spécificités du matériel (pas de MMU)
- Fiabilité - Performance
- Délai de mise sur le marché

Présentation : *Contraintes du SE*

Absence d'environnement intégré:



Présentation : *Contraintes du SE*

➢ **Un « bon » OS embarqué:**

- Modulaire,
- Evolutif,
- Configurable,
- Faible encombrement mémoire,
- Processeur supporté,
- Drivers, …

➢ **Un « bon » OS Temps réel :**

- Multiprocessus (multi threads)
- Préemptif
- Gestion des priorités (pas de « deadline driven OS»)
- Mécanismes de communication et synchronisation prévisibles
- Priorités inhérentes, …

SEP vs SETR

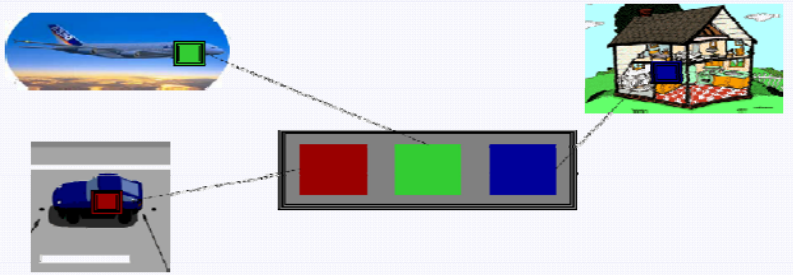
Présentation : *Approche Centrée Système*

➢ Aspects :
- *Techniques* : Conception **conjointe** (Matériel, Logiciel, Environnement)
- *Économiques* : Optimisation par rapport au marché , entre coût et qualité (efficacité, flexibilité)
- *Multi-compétence* : Combinaison de compétences en informatique, automatique, électronique, ergonomie. => FORMATION

➢ Les systèmes complexes sont construits comme les cathédrales au moyen-age :
- Pas de théorie unifiée pour les logiciels et les propriétés dynamiques de leur exécution sur un matériel donné.
- Construction des systèmes complexes par « patch » successifs.
- Explosion des coûts de validation !

Présentation : *Défis à relever: Hétérogénéité*

Mécanismes de communication, vitesse de fonctionnement, granularité des calculs, variété des médias…
- Construire des systèmes complexes par intégration de composants hétérogènes

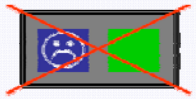


- Normes pour garantir l'interopérabilité
- Modèle unifié supportant l'intégration hétérogène
  - *(parallélisme, temps, gestion des ressources, performances …)*

Présentation : *Défis à relever: Complexité*

L'effort de validation augmente exponentiellement avec le nombre de composants intégrés.

- Remplacer les méthodes de validation a posteriori par des méthodes de validation incrémentale.
- Développer des outils théoriques et techniques pour la validation incrémentale
- **Composabilité** : Préserver la fonctionnalité et la qualité des composants le long de l'intégration

- **Compositionalité** : Inférer les propriétés d'un systèmes à partir des propriétés de ses composants

Présentation : *Défis à relever: Intelligence*

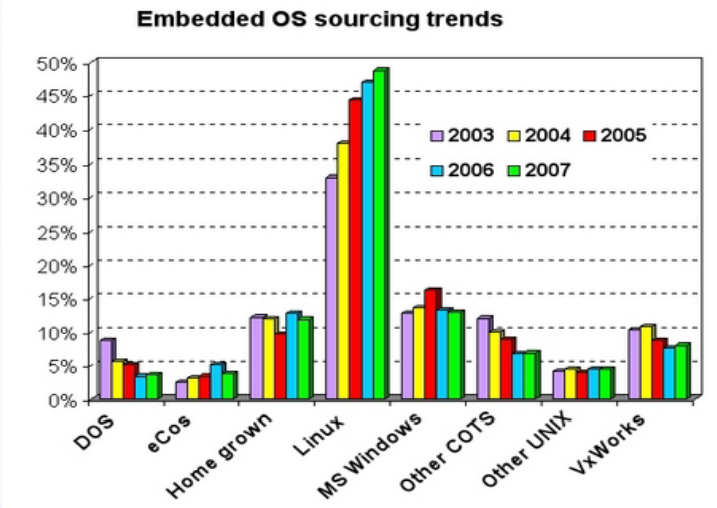
- Moyen d'améliorer la qualité (robustesse et la performance) des systèmes :
  - Réflexivité: capacité d'analyser, d'auto-diagnostiquer son état
  - Adaptabilité: capacité d'adapter (auto-organiser+planifier) son comportement en fonction d'objectifs de robustesse et de performance
- Deux propriétés de robustesse :
  - Sûreté : résistance aux mauvais fonctionnements (pannes, erreurs) et aux aléas de l'environnement
  - Sécurité : résistance aux attaques et actes de malveillance

Développer des outils théoriques et pratiques pour l'ingénierie des systèmes intelligents

Présentation : *Les pionniers de l'Embarqué*

- **Windows (CE, XP Embedded, Automotive, Mobile)**
- **Linux Commercial (Amirix Embedded Linux, Coollogic Coollinux, Coventive Xlinux, Esfia RedBlue Linux, KYZO Pizza Box Linux Embedix - lineo, Blue Cat – Lynx, Hard Hat – Montavista)**
- **Linux Open source (GNU, Embedded Debian Project, ETLinux, µCLinux, µLinux)**
- **WindRiver System (WxWorks)**
- **QNX Software**
- **MicroWare (OS9)**
- **Palm OS**
- **Symbian OS**

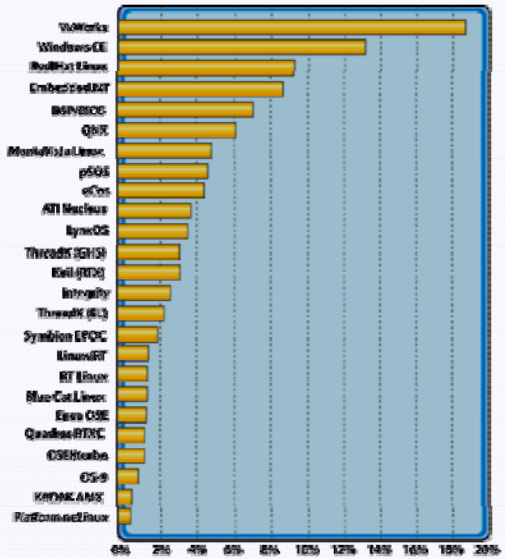

Présentation : *Choix Windows*



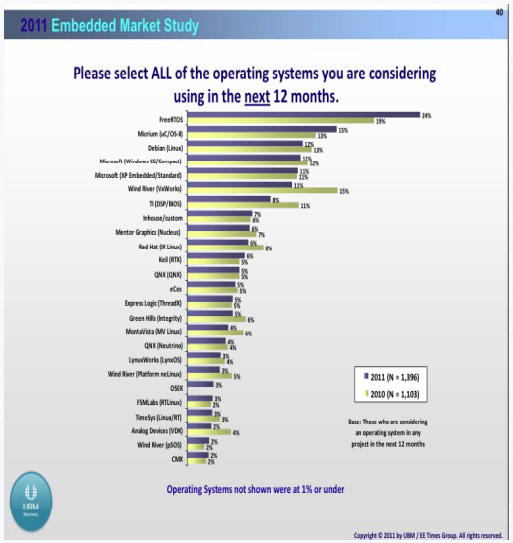
Snapshot of the embedded Linux market -- April, 2007

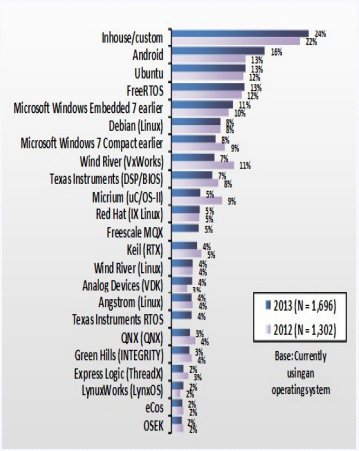
Présentation : *Répartition des OS en 2005*

- Windows Mobile pour SEP
- OS « constructeur » pour SETR (Spectrum Digital)

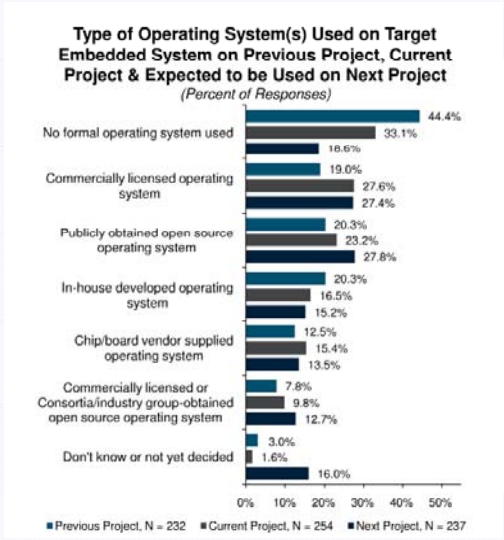


Présentation : *Répartition des OS en 2011-2013*

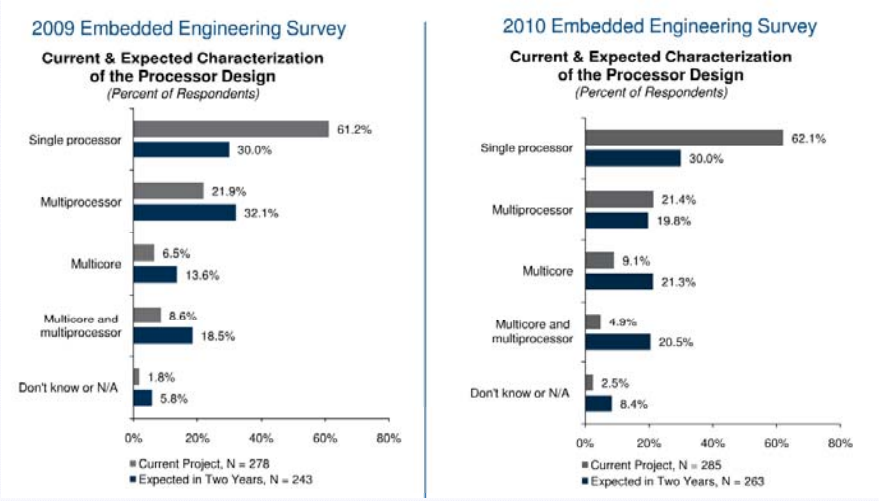




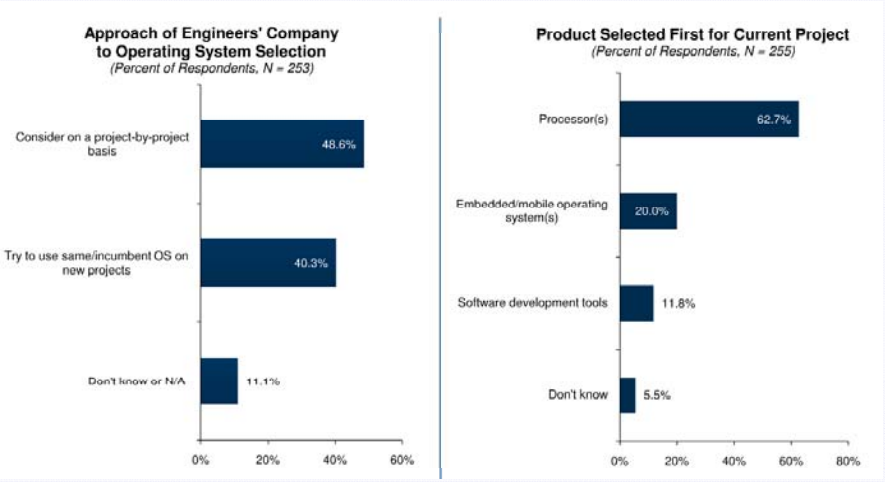
Présentation : *Choix d'un OS*



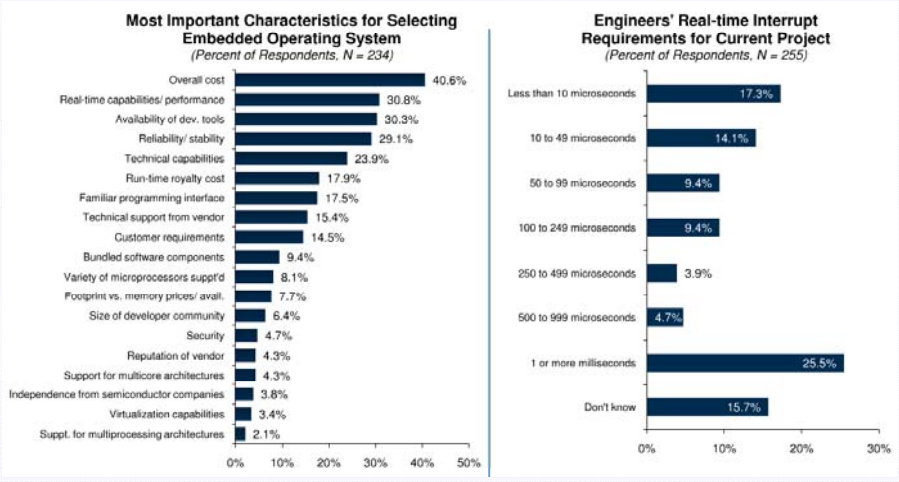
Présentation : *Choix d'un OS*



Présentation : *Choix d'un OS*



Présentation : *Choix d'un OS*



Présentation : VxWoks

- **VxWorks : OS commercial temps réel développé par Wind River Systems.**
- **Idée principale:**
  - use monolithic kernel to schedule user tasks according to user defined priorities. Maximize kernel timing predictability.
  - Gives the users maximal control.
  - A dedicated real time system, not intended as a general purpose OS.
- **Caractéristiques:**
  - Lacks many modern os features that interfere with real time performance (flat memory model, no paging).
  - Scheduling is done using a preemptive priority driven approach, priorities are chosen arbitrarily by the user (0-255).
  - Priorities can be changed by the user at runtime but this is discouraged.
  - A user can lock a task so that it can't be preempted even by higher priority tasks or interrupts.
  - This allows the use of the fixed priority response time analysis to check schedulability offline.
  - Is resource sharing aware and has a priority inheritance built in.
  - Optimizations in implementation of the context switch and the return from interrupts.
  - The kernel never disables NMI (non-maskable interrupts) so they are always available to the user.

Présentation : VxWoks

- **LIMITATIONS:**
  - Ne propose pas les primitives d'un OS "moderne".
  - Le respect des échéances est du ressort du concepteur informatique.
  - Ne supporte pas les API "modernes" (petite partie de POSIX).
  - Le modèle de mémoire retenu est source de fragmentation mémoire.
  - Offre un maximum de contrôle, mais peu de services
- **OS dédié aux applications Temps Réel.**

Présentation : *Choix Windows*

- **Les points forts**
  - noyau bien conçu et cohérent,
  - intégration et offre globale,
  - .NET pour les applications de base (RAD),
  - .NET pour le Web,
  - masse critique largement dépassée.
- **Les inconvénients**
  - technologies complexes,
  - dépendance forte envers Microsoft.
  - développement coûteux,
  - technologies non maîtrisées,
  - bogues incontournables,
  - déploiement difficile,
  - prise en compte de la sécurité

Présentation : *Choix Windows*

- Les solutions
  - définition stricte du périmètre fonctionnel,
  - définition stricte de l'environnement cible,
  - formation technique des développeurs,
  - isolation des applications et des réseaux,
  - expertise externe,
  - ne pas sous-estimer les charges.

Présentation : *Choix Windows*

Total Cost of Development Analyzed

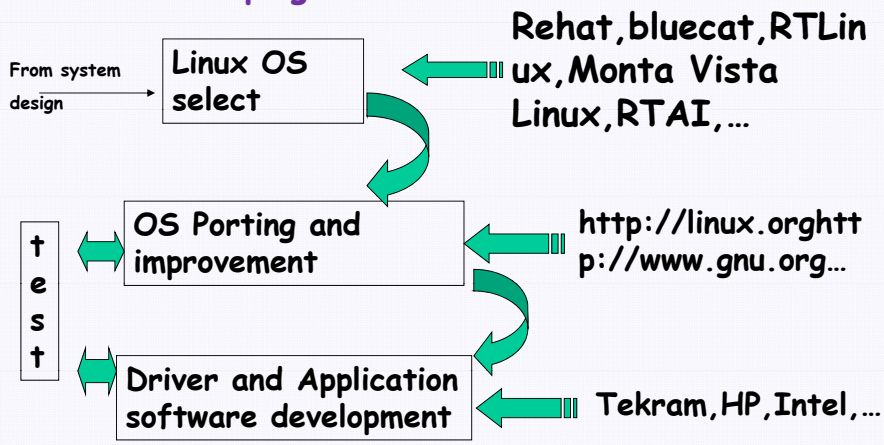
An Evaluation of the Costs Associated with Embedded Linux Development as Compared with Commercial RTOSes and Non-Commercial Linux **(December 2007)**

A comprehensive cost estimation framework for evaluating embedded development platforms (Juillet 2003)

Jerry Krasner, Ph.D., MBA
Embedded Market Forecasters
American Technology International, Inc.

Présentation : *Choix Windows*

Linux Developing Process



Présentation : *Choix Windows*

- **Comparative Total Cost of Development and Associated Costs Summary**

Data Results	Windows CE .NET	Windows XP Embedded	Embedded Linux
Total Cost of Development			
Total Time to Market, months	**8.2**	**8.0**	**14.3**
Software Engineers/Project, #	**8.3**	**7.3**	**14.2**
Development Man Months (mm)	**68.1**	**58.4**	**203.1**
Cost/mm of developer's time	**$7,500**	**$7,500**	**$7,500**
Total Cost of Development	**$510,450**	**$438,000**	**$1,522,950**
Comparative Savings (relative to Linux)	**66.5%**	**71.2%**	**0%**
Associated Costs			
Average Tool Costs (single developer seat)	**$995**		**$3,899**
Average Maintenance Costs (per developer)	**$0 for 1st 5 years.**		**$740-10,000/year**
Average Support Costs	**1st two support incidents free. $99 per web support incident. $245 per phone support incident**		**$500-2500/year**
Example 5 Year Associated Costs	**$6,660**		**$56,697**

Mécanismes Fondamentaux d'un Système d'Exploitation

- **Interfaçage des E/S**
- **Notion de Tâche**
- **Etat et changement d'Etat d'une tâche**
- **Ordonnancement**
- **Exclusion Mutuelle, Synchronisation, Communication**

Mécanismes Fondamentaux : *Interfaçages des E/S*

- Gestion des E/S (blocage, synchro, …)
- Interaction par Scrutation Cyclique (polling) :
  - Méthode « type Automate Programmable Industriel »
    - Boucle infinie sur le Cycle:
      - » Lire Capteurs tant que info non disponible
      - » Acquérir Données d'entrée
      - » Traiter les données
      - » Démarrer les réactions
      - » Vérifier les Actionneurs jusqu 'à actions effectuées
- Interaction par interruptions :
  - Scrutation des interruptions
    - Carto - mémoire, DMA, …
      - » ***Pb: contrôle des temps de latence de E/S***
  - Traitement et Gestion des interruptions
    - Imbrication, driver, …
      - » ***Pb: Réduire les durées aléatoires***
  - Réduire les temps de réponses aux interruptions

⇒ **Systèmes Multitâches :**

Diviser une application en tâches importantes (noyau) et tâches moins importantes (utilisateur)

Mécanismes Fondamentaux : *Notion de tâches*

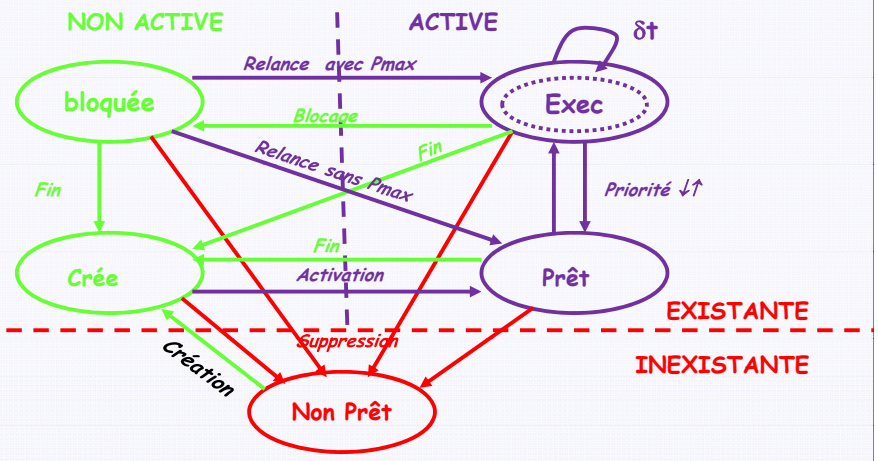
- **Définition :**
  - Une tâche (process) ou processus est définie comme une entité dynamique, par opposition à un programme qui est essentiellement statique.
  - Une tâche possède un contexte et peut évoluer dans le temps. Un aspect cyclique, des notions de progression et d'actions sont liés à la notion de tâche.
- **Contexte d'une tâche :**
  - Ensemble d'informations qui peuvent être consultées ou modifiées par la tâche : la tâche possède un processeur virtuel.
  - Tache complète (Processus) ou Tache « légère » (Thread)
  - Le contexte complet (lourd) est constitué de :
    - contexte du Proc. (C.O. , Registres, Flag, …)
    - contexte mémoire indépendant* (Code, données, pile, …)
    - ensemble d'attributs* :
      - » Identif du processus
      - » Priorité,
      - » Droits.

(* : Pas pour Thread)

⇒ PAS DE NOTION DE TEMPS

Mécanismes Fondamentaux : *Etats d'une tâche*

- 5 états ou groupes d'états d'une tâche :



Mécanismes Fondamentaux : *Changements d'Etat*

- Etats d'une tâche:
  - État Non créé (inexistante)
  - État Dormant (Crée, Inactif, Sommeil)
  - État Prêt (Actif, Veille)
  - État Exécuté
    - Proc Réel
    - Proc Virtuel
  - État Bloqué (Suspendu)
- Changement d'état:
  - Création
  - Activation
  - Relance
    - Avec Pmax
    - Sans Pmax
  - Fin
  - Suppression

Mécanismes Fondamentaux : *Politiques d'ordonnancement*

- Ordonnanceur (Scheduler)
  - Permet de choisir la tâche à attribuer au processeur
  - Permet le passage d'une tâche à une autre
  - Contraintes :
    - Garantir à chaque tâche un tps d'allocation donné
    - Respecter un ordre de priorité
    - Respecter un temps de réponse donné
    - Annuler de manière préemptive une tâche
- Commutation de tâche (Dispatcher)
  - Sauvegarde du contexte de la tâche en cours
  - Mise de la tâche en file d'attente
  - Requête à l'ordonnanceur pour élire la tâche suivante
  - Restitution de contexte de la nouvelle tâche.
- Critères de choix de l'ordonnanceur
  - Efficacité : E= (T-t) / T
  - Temps de réponse
  - Impartialité
  - Débit
  - Tps de réponse aux interruptions

Mécanismes Fondamentaux : *Politiques d'ordonnancement*

- Stratégies Classiques :
  - Circulaire (coopératif : Windows 95)
  - Préemptif (Windows NT)
- Technique FIFO
- Technique « Short Job First »
- FIFO avec priorité
- Ordonnancement Préemptif
  - Round Robin Simple
    - Méthode de recyclage : Time Sharing
    - Système préemptif
    - Choix du Quantum (efficacité)
  - Tourniquet à files d'attente multiples
  - Tourniquet avec recyclage des priorités
    - Respect du déterminisme
    - Priorité flottante calculée à partir de :
      - temps CPU consommé
      - temps passé en file d'attente
      - caractéristiques des ressources utilisées

⇒ CONTRAINTES DE TEMPS ?

Mécanismes Fondamentaux : *Outils classiques*

- **Exclusion Mutuelle**
  - Masquage des interruptions
    - Fn Système
  - Attente Active
    - Pas adapté
  - Verrous - Sémaphores
    - Fn Bloquantes, Files d 'attente FIFO
  - Moniteurs
    - Systèmes ou Langages orientés Objet (AML2, ADA)
- **Synchronisation**
  - Directe - indirecte
  - Par événements (signaux) mémorisés ou non
  - Par sémaphores
  - Type RdV
- **Communication**
  - Par variables communes (mémoire partagée)
  - Par messages (Bal, Tubes, Sémaphores à mess.)

Un système : *le noyau Windows NT*

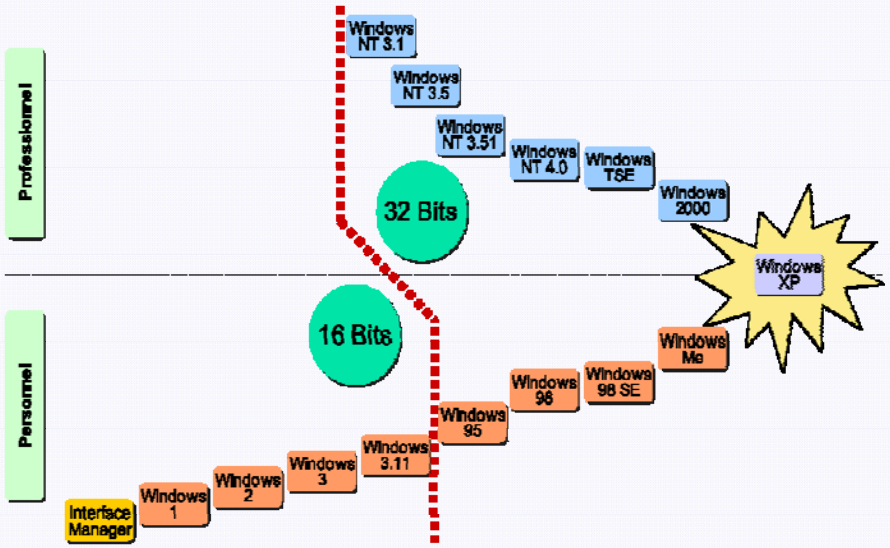
- **Historique**
- **Le noyau**
  - **Architecture**
  - **Services**
  - **Gestion du temps**
- **Gestion mémoire**
- **Interface Graphique**

Noyau Windows NT : *Historique*

1972 : *Bill Gates et Paul Allen* fondent la société **Traf-O-Data** (future Microsoft)

- **Novembre 1983** : **Microsoft Word 1.0** pour MS/DOS.
- **Avril 1984** : **Microsoft** présente **Interface Manager** (**Windows**),
- **Mai 1985** : **Microsoft** présente **Microsoft Windows 1.0,**
- **Novembre 1985** : **Microsoft Windows 1.0** sur le marché pour 100 $.
- **Microsoft 2.0 fut présenté en 1987.**
- **Fin 1988**, Bill Gates débauche Dave Cutler (DEC), pour créer NT OS/2, ou OS/2 3.0 ou encore Portable OS/2, qui donnera naissance à Windows NT.
- **1990 : Microsoft** sort **Windows 3.0.** (succès commercial).
  Rupture définitive avec IBM
- **Octobre 1991: Microsoft** annonce **Windows NT** (OS Nouvelle Technologie)
- **Juillet 1993 : Sortie de Windows NT 3.1**
- **Septembre 1994 : Sortie de Windows NT 3.5**
- **1996 : Sortie de Windows NT 4** (internet + mode noyau)
- **Février 2000 : Sortie de Windows 2000** (USB, DFS, ADS, Mode sans échec,…)
- **Septembre 2001 : Sortie de Windows eXPerience** (FireWire, WiFi, …)
- **2003 : Sortie de Windows .NET 2003** (version serveur de XP)

Noyau Windows NT : *Historique*



Noyau Windows NT : *Historique*

- .Janvier 2007 : Sortie de Windows Vista (64 bits)
- .Février 2008 : Sortie de Windows Server 2008
- · Octobre 2009 : Sortie de Windows 7 / Server 2008 R2
- · 2012 : Sortie de Windows Server 2012
- .Octobre 2012 : Sortie de Windows 8 / Windows RT (ARM 7)
- .Octobre 2013 : Sortie de Windows 8.1

- **Evolution des performances :**

Système	SMP	RAM	VM
NT4	2	128 Mo	4 Go
2000	2 - 32	4 Go	4 – 16 Go
XP	2 – 32	4 – 16 Go	4 Go – 16 To
.Net 2003	4 - 64	4 – 512 Go	4 Go – 16 To
Server 2012	64	4 To	….

Noyau Windows NT : *Le cahier des charges*

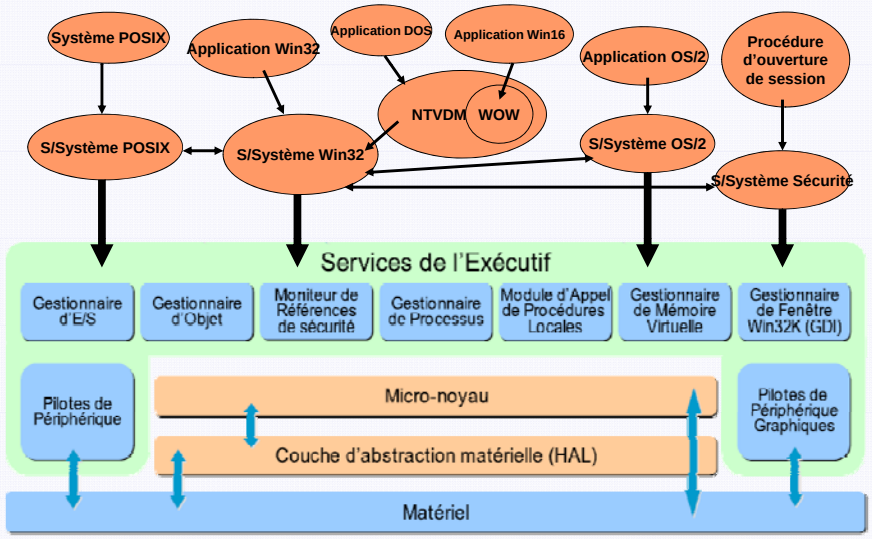
Microsoft voulait créer un OS :

- Evolutif
- Portable
- Fiable
- Compatible
- Sécurisé
- Performant

Noyau Windows NT : *Architecture en général*

- Objet, objet, objet
- monolithique pas trop, plutôt modulaire
- Préemptif et multi-tâches
- Multi-processeurs (SMP)
- Architecture Client/Serveur
- Adressage 32 bits → 4Go d'espace mémoire
- Support de plusieurs systèmes de fichiers
- Ecrit en C et C++ principalement
- Services réseau intégrés

Noyau Windows NT : *Architecture*



Noyau Windows NT : *Les sous-systèmes d'environnement*

- VDMs (Virtual DOS Machines) MS-DOS et VDM Win16 n'en sont pas
- Sous-système OS/2
- Sous-système Posix
- Sous-système Win32
- L'entête du fichier définit dans quel environnement le programme doit être exécuté
- Fournit un environnement d'exécution pour les applications

Noyau Windows NT : *VDM MS-DOS*

- C'est une application Win32 émulant un ordinateur x86 (80386 et +).
- Instructions x86 émulées par la « Instruction Execution Unit »
- Services d'interruptions ROMBIOS fournis par le module d'émulation MS-DOS
- Services d'interruptions 21 de MS-DOS fournis par le module d'émulation MS-DOS
- Les « Virtual Device Drivers » (VDD) émulent un hardware virtuel (écran, clavier,...)
- Une VDM par application avec son propre espace d'adressage
- Nombre de VDM DOS illimité
- La VDM offre plus de mémoire qu'il aurait été possible d 'avoir en DOS normal

Noyau Windows NT : *VDM Win16*

- Un sous-système Win16 multi-threadé coopératif
- WOW : Win16 On Win32 :
  - fait la conversion 16 bits - 32 bits et inversement (thunking)
  - fait le lien avec le sous-système Win32
- Applications Win16 non préemptives entre elles, mais le système reste préemptif
- Toutes les applications Win16 tournent dans le même environnement

Noyau Windows NT : *Sous-système OS/2*

- Interface de présentation de NT à l'origine
- OS/2 1.x → mode caractère
- Tourne uniquement sur des processeurs à base Intel
- Les applications OS/2 tournent dans leur propre espace d'adressage mémoire en mode multi-tâche préemptif
- Quelques supports réseau

Noyau Windows NT : *Sous-système Posix*

- Portable Operating System Interface
- Conforme à Posix 1 (IEEE Std 1003.1-1988)
- Fonctions réseau et système non Posix
- Noms de fichiers Posix avec NTFS
- Les applis Posix tournent dans leur propre espace d'adressage mémoire en mode multi-tâche préemptif

Noyau Windows NT : *Norme POSIX 1*

- POSIX.1, Services centraux (inclut le standard *ANSI C*) (IEEE Std 1003.1-1988):
  - la création et le contrôle des procesus
  - les gestions des signaux interprocessus
  - les exceptions des nombres flottants (gestion du *FPU*)
  - les violations de segmentation
  - les instructions illégales
  - les erreurs de bus
  - les timers
  - les opérations sur les fichiers et les dossiers
  - les tubes
  - la librairie standard de C
  - les entrées-sorties et le contrôle des ports
- POSIX.1b, extension pour le temps réel (IEEE Std 1003.1b-1993) :
  - l'ordonnancement
  - les signaux en temps réel
  - les horloges et les timers
  - les sémaphores
  - le passage de messages
  - la mémoire partagée
  - les entrées-sorties synchrones et les *asynchrones*
  - les outils de verrouillage de la mémoire
- POSIX.1c, extension sur les processus légers (les threads) (IEEE Std 1003.1c-1995) :
  - la création, le contrôle et la suppression des thread
  - l'ordonnancement des thread
  - la synchronisation des thread
  - l'interception des signaux (*Signal Handling*)

Noyau Windows NT : *Sous-système Win32*

- Tous les sous-environnements vus précédemment passent par Win32 : ils traduisent leurs API en API Win32
- Gère l'affichage, le clavier et la souris
- Le gestionnaire de fenêtres (USER) et l 'interface graphique (GDI) sont en fait dans le NT Executive

Noyau Windows NT : *H.A.L.(Hardware Abstraction Layer)*

- Interface entre le matériel et le reste du système d'exploitation
- Intel, MIPS, PowerPC, Alpha
- Interface SMP (Symmetric Multi Processing)
  - jusqu'à 4 processeurs et même 32 (dev. spécifique)
- Chaque processeur physique est représenté comme un processeur virtuel
- Accessible que par NT Executive

Noyau Windows NT : *Kernel (Noyau ou micro-noyau)*

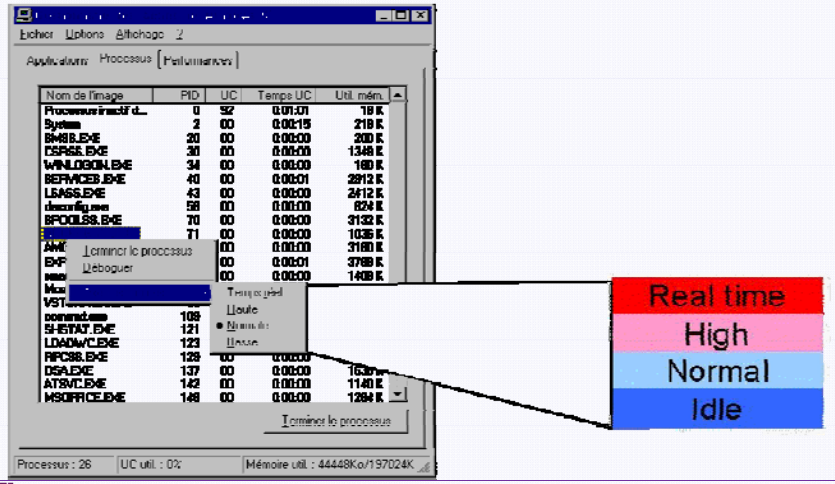
- Gère le(s) microprocesseur(s)
- Distribue et planifie les threads sur les processeurs virtuels (en fonction des priorités)
- Un noyau tourne sur chaque processeur
- Ne peut être swappé ni préempté
- Gère les interruptions système
- Traite les exceptions processeur
- Parties critiques écrites en assembleur
- Le reste des tâches est délégué aux services de NT Executive

Noyau Windows NT : *Ordonnancement*

- Système d'exploitation multitâche préemptif.
- Round Robin avec Priorités Multiples
- Règles de préemption:
  - Fin d'un TimeSlice.
  - Arrivée d'un Thread de priorité supérieure.
  - Accès à un périphérique
- 31 Niveaux de priorité:
  - 1 – 15 Utilisateurs.
  - 16 – 31 Administrateurs.

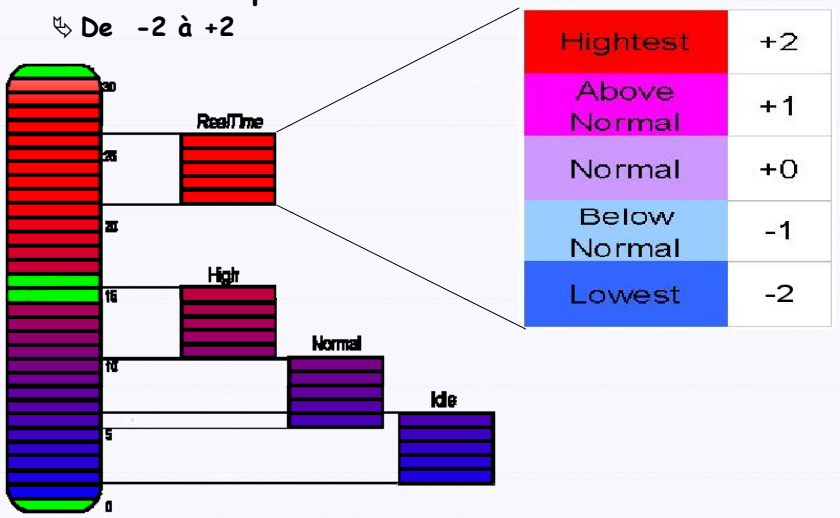
Noyau Windows NT : *Ordonnancement*

- 4 classes de priorités :



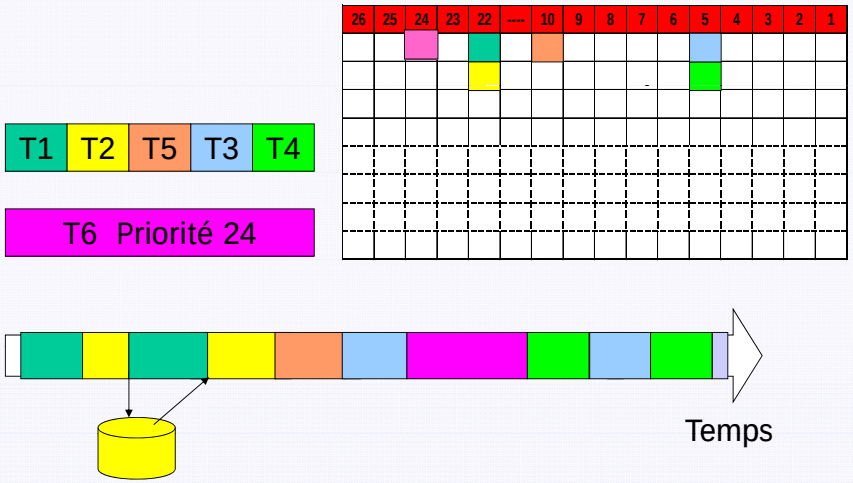
Noyau Windows NT : *Ordonnancement*

- 5 Niveaux de priorités:
  - De -2 à +2

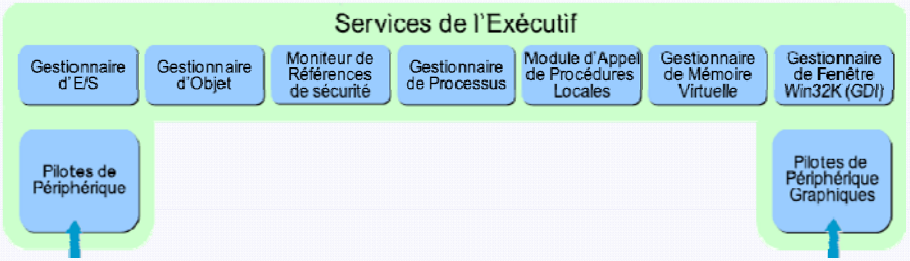


Noyau Windows NT : *Scheduler*

- Il décide de l'exécution des Threads



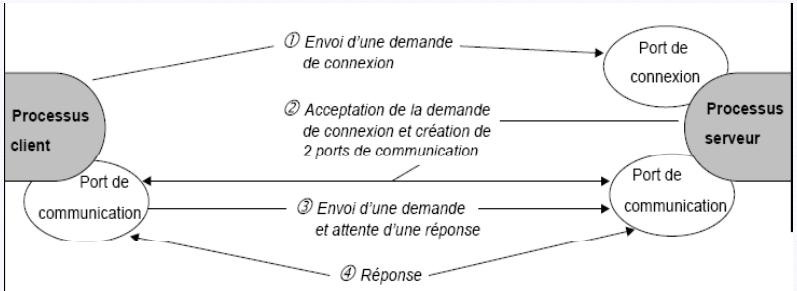
Noyau Windows NT : *Services du NT executive*



- **Ce sont des programmes qui fonctionnent en mode noyau**
- **Fournissent des services de base aux sous-systèmes d'environnement**

Noyau Windows NT : *Appel de procédure locale (LPC)*

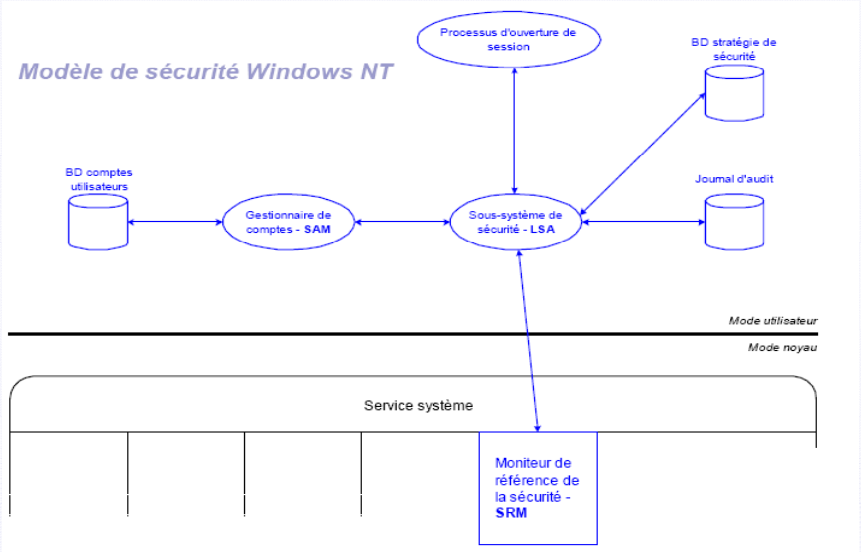
- **Permet à deux threads appartenant à deux processus différents de communiquer**
- **Deux processus voulant dialoguer passent obligatoirement par le LPC : modèle Client / Serveur**
- **Les dialogues entre sous-systèmes d'environnement Win32 et Posix par exemple passent par des LPC**



Noyau Windows NT : *Sécurité*

- ***Sous-système de sécurité – LSA (Local Security Authority)***
  - Gère et applique la stratégie de sécurité de l'ordinateur local (définie par l'administrateur dans la base de données "stratégie de sécurité")
  - Offre un service interactif de validation des utilisateurs
  - Génère les jetons d'accès
- **Moniteur de référence de la sécurité – SRM (Security Reference Monitor)**
  - Gère la sécurité sur la machine locale
  - Sécurise l'accès aux objets du système
  - Compare les droits de l'utilisateur et l'ACL de l'objet : calcule les droits à positionner dans le handle

Noyau Windows NT : *Sécurité*



Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- Coordonne les entrées-sorties du système
- Conception par couches
- Gère les systèmes de fichiers
- Gère les redirecteurs réseaux
- Gère le cache disque
- Gère la communication entre drivers

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***Les différents types de périphériques***
  - **Classiques :**
    - Mémoire de masse :(DD, CDROM, disquette, bande).
      - ✓ périphérique IDE / périphérique SCSI /SATA
    - Réseaux (réseaux locaux).
    - Vidéo (écran) graphique.
    - Imprimantes
    - Multimédia (microphone, webcam).
    - Standard / parallèle / série / souris / clavier.
    - Autres (lecteur de cartes, lecteur code barre, lecteur optique, scanner, digitaliseur).
  - **« Industriels » :**
    - Parallèle / série, TOR, Analogique 0-10V , -10+10 , couplée …
    - Acquisition d'images, traitement d'images,
    - Communication, téléphonie (GPS, GPRS, ..)
    - Capteur exotique (X, scintillateur, laser, peau sensible, ...)

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***Accès à partir de l 'adresse du premier registre***
- ***Dépend de l 'Espace d'adresse où se trouvent les registres***
- ***Généralement : 3 types de registres***
  - Commande (contrôle)
  - Etat
  - Donnée (tampon)
- ***Mécanismes d'accès au périphérique :***
  - **Instructions spéciales (implicite)**
    - Petit calculateur : pas d'adresse de registre → micro-contrôleur.
  - **Cartographie mémoire**
    - Registre dans espace des E/S ⇒ port (accès par inp / outp)
      Limité à 64 Ko sur 80x86 et +.
    - Registres mappés dans mémoire centrale (accès par mov)
    - Double accès (tampon vidéo + registre de commande).

Mécanisme de transfert de données

➢ *E/S programmées (PIO)*

- Adresse vue comme un mot mémoire
  - registre de contrôle du périphérique.
- Utilisent le processeur pour l'initialisation du transfert :
  - émission/réception
  - gestion du tampon
  - décompte d'octets transférés
  - interruption signale chaque transfert

pilote ou circuit

⇒ *périphérique lent ; peu de données*

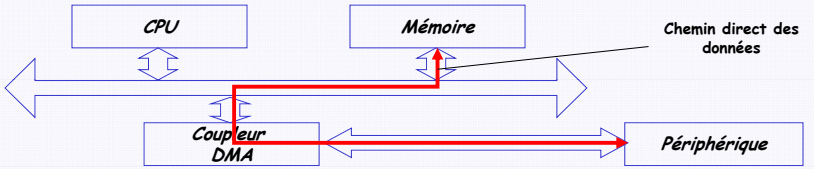
➢ *DMA (Direct Memory Access)*

- Composant :
  processeur d'E/S : coupleur DMA (1950).
- Objectif :
  Décharger la CPU d'une partie du travail de lecture/écriture des E/S
  Gestion du périphérique (transfert mémoire ⌦ E/S).
- Principe :
  Le coupleur DMA est initialisé avec une valeur d'adresse mémoire et un compteur de transfert :
  - à chaque transfert, le CDMA incrémente l'adresse et décrémente le compteur de transfert ;
  - à la fin des transferts, génère une interruption.
- Travail de la CPU :

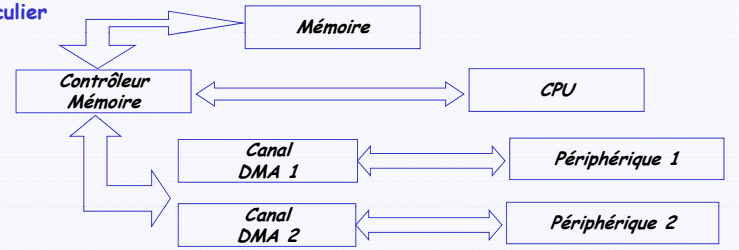
initialise le CDMA avec registre d'adresse et décompte + mot contrôle (L/E + …).

➢ *Différents types de CDMA :*

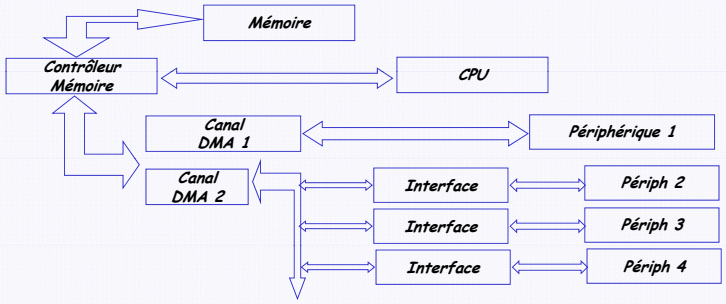
- coupleur : lien direct entre le périphérique et la RAM



- canal DMA : plusieurs liens (généralement 4) attribués à un périphérique particulier



- canal multiplexé : plusieurs liens attribués à plusieurs périphériques en même temps



➢ *DMA bus master: DMA propre au périphérique*
(indépendant CPU)

➢ *DMA esclave ou système: DMA sur carte mère système,*
(périphérique demande 1 canal).

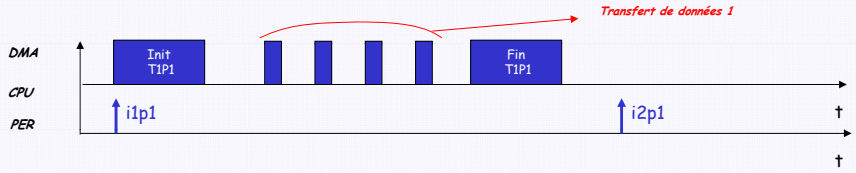
➢ *Partage mémoire DMA :*

➢ *Mémoire centrale : passe par Bus S.I*
- Attention ⇒ allocation avant transfert

➢ *Mémoire sur carte : vue depuis mémoire centrale*
- Remarque : cela revient à ajouter de la mémoire dans le S.I.

➢ *Problème de partage de données :*
- vol de cycle : suspend le processeur pendant quelques cycles (juste R/W) ;
- mode bloqué : verrouille l'accès à la mémoire (par objet logiciel) ou niveau matériel par blocage de la CPU le temps du transfert (driver gère le partage) ;
- mode Rafale (Burst) :
  - DMA alloué au périphérique seul si MUX,
  - transfert en une seule fois de la totalité des données (périphérique rapide à haut débit)

➢ *Séquencement :*



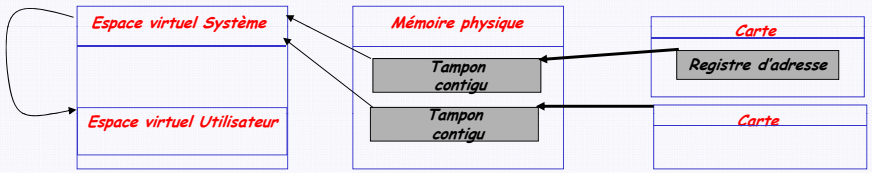
Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***E/S tamponnées (bufferisées)***
  - Principe : Périphérique lit ou écrit dans des variables intermédiaires, indépendantes de l'espace virtuel du processus utilisateur. Il y a recopie de cette zone dans l'espace de l'utilisateur à la fin du transfert (si entrée).
  - Pourquoi : Résoudre le problème posé par l'adressage virtuel : en direct I/O, les variables du processus doivent être dans des pages présentes en mémoire et verrouillées pour éviter les fautes de pages.
  - Difficultés : On ne peut pas écrire à des adresses physiques.
  - Solution : On alloue une zone tampon dans un espace mémoire non paginé, réservé par le système (voir schéma).
  - Inconvénients :
    - temps perdu par recopie,
    - mémoire occupée importante

    réservé au périphérique lent avec peu de données si tampon en mémoire centrale.
  - Autre solution : tampon sur disque ⇒ SPOOL imprimante (Simultaneous Peripheral Output Off Line).
  - Conclusion : Dans les deux cas, le périphérique apparaît aussi rapide que la vitesse de transfert du support utilisé (mémoire centrale ou de masse).
  - Remarque : Le système et le processus partagent l'accès au buffer ⇒ E.M ⇒ accès asynchrone

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***Plage d 'adresses privées en mémoire physique dans la zone adressable de la CPU :***
  - *Registres mappés en mémoire ou ROM interne*
  - *Bloc mémoire tampon temporaire (cartes vidéo ou réseau)*
- ***Le périphérique à besoin d'une plage d'adresses particulière***
  - *(carte video : bloc de 128Ko en OxA0000)*
- ***Le périphérique possède un registre contenant l'adresse de base de sa mémoire dédiée.***



- ***Dans les trois cas, le problème à résoudre reste que la carte gère des adresses physiques et que les systèmes évolués ne fournissent généralement que des adresses virtuelles***
  - *le pilote de périphérique doit mapper la mémoire physique dans l'espace virtuel du système pour la rendre atteignable par les processus.*

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***Autoconfiguration (Plug and Play)***
- ***Objectifs:***
  - *Reconnaissance automatique du matériel,*
  - *Configuration automatique du Système Info.*
- ***Problème:***
  - *Affectation des ports, IRQ et canaux DMA.*
- ***Généralement:***
  - *Configuration hard par switch ou cavalier sur la carte mère ou sur le périphérique,*
  - *Configuration soft par programmation (plus souple d'utilisation).*
- ***Pré-requis:***
  - *Identification du périphérique et liste des ressources :*
    - *ID Fabricant,*
    - *ID type de périphérique,*
    - *Espaces E/S nécessaire,*
    - *Interruption(s) nécessaire(s),*
    - *Canaux DMA,*
    - *Mémoire nécessaire.*
  - *Reconfiguration dynamique du périphérique.*
  - *Notification de changement au système.*

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- ***Le système de gestion des E/S doit permettre au système d'exploitation de réaliser plusieurs tâches :***
  - *Assurer une réponse rapide aux E/S (vecteurs d'interruption),*
  - *Offrir un ensemble de services suffisants pour répondre aux demandes de tous les programmes,*
  - *Gérer la base de données des E/S (exécution de plusieurs centaines d'E/S simultanées),*
  - *Prendre en charge l'asynchronisme et offrir aux programmes utilisateurs des fonctions de resynchronisation,*
  - *Assurer l'indépendance logiciel-matériel en introduisant la notion de conducteur d'interface (driver).*
- ***Problèmes***
  - *Plusieurs E/S sont en cours simultanément (en parallèle).*
  - *Le système doit prendre en compte la totalité de la gestion des E/S.*
- ***Solutions :***
  - *Stocker les différents paramètres relatifs à une interruption en fonction du type de périphérique :*
  - *A partir de l'interruption, le système d'exploitation retrouve le périphérique, puis son type, dans une table appelée "descripteur de périphérique".*
  - *A partir de ce descripteur, le système d'exploitation construit des paquets d'informations concernant l'E/S.*

Noyau Windows NT : *Gestionnaire d'entrée-sortie*

- *Notion de base de données d'E/S : Contenu des paquets d'E/S*
  - *Lien avec le périphérique (I/O Request Packet),*
  - *Adresse d'une zone tampon contenant les données,*
  - *Longueur et le type de l'E/S,*
  - *Adresse du programme concerné par l'E/S,*
  - *Adresse des drapeaux,*
  - *Adresse des sous-programmes d'interruption.*
- *Les informations, descripteur de périphérique + paquet, constituent la base de données d'E/S.*

Maximum d'indépendance entre le logiciel et le matériel.

- *Solutions :*
  - *Le système d'exploitation est le seul à connaître l'adresse physique du périphérique dans la machine.*
  - *L'utilisateur ne "voit" qu'un nom ou numéro logique.*
  - *Le système maintient une correspondance entre ce nom logique et le numéro de périphérique.*
  - *Le système d'exploitation dirige les requêtes d'E/S vers un programme spécial : le conducteur d'interface ou driver.*

Noyau Windows NT : *Drivers (XP)*

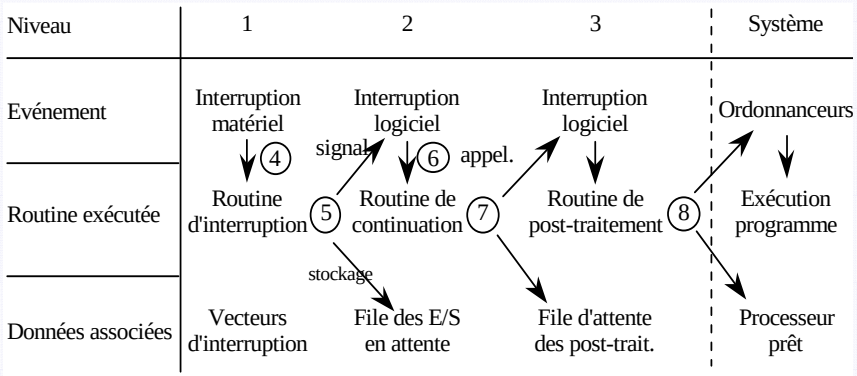
- Ce programme contient toutes les particularités du périphérique
  - (adresse fn, adresses des mots de contrôle, flag à positionner, à tester).
- Le conducteur d'interface est une suite de sous-programmes qui s'enchaînent.
- De plus, pour pouvoir traiter plusieurs E/S simultanées, certaines de ces routines doivent pouvoir s'appeler elles-mêmes
  - interrompues par une routine qui provoque sa propre exécution: réentrance.
- Deux types de drivers
  - Kernel Mode pour la performance,
  - User Mode pour la robustesse.
- Effort minime pour porter les drivers existants vers XP puis Vista, Seven
  - 1 à 3 jours de travail par driver,
  - La structure générale des drivers ne change pas,
  - Les changements principaux résident dans la manière dont les drivers accèdent à la mémoire des applications clientes,
  - Plus d'appels à SetKMode ou SetProcPermissions.

Noyau Windows NT : *Drivers (XP)*

- **Software Development Kit (SDK) :**
  - Outil de développement permettant de réaliser des applications utilisant des fonctions systèmes incluses dans les bibliothèques du SDK.
- **Driver Development Kit (DDK) :**
  - Outil de développement permettant de réaliser des drivers sous Windows XP.

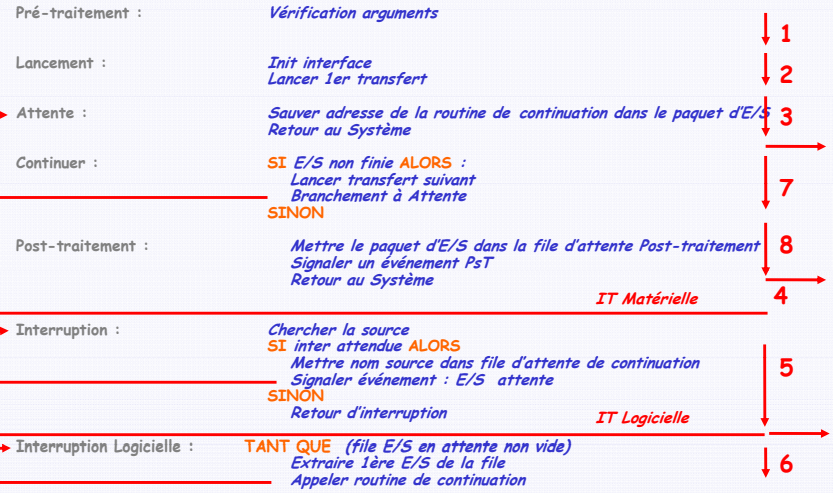
Noyau Windows NT : *Drivers (XP)*

- Schéma général d'un driver: (routines ré-entrantes)



Noyau Windows NT : *Drivers (XP)*

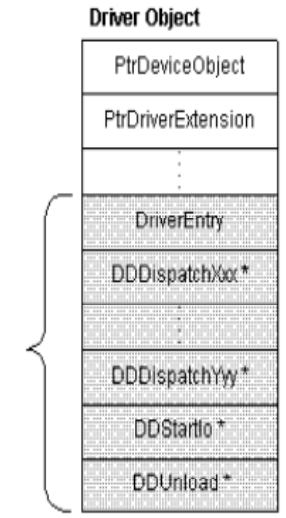
➢ Schéma général d'un driver: (routines ré-entrantes)



Noyau Windows NT : *Drivers (XP)*

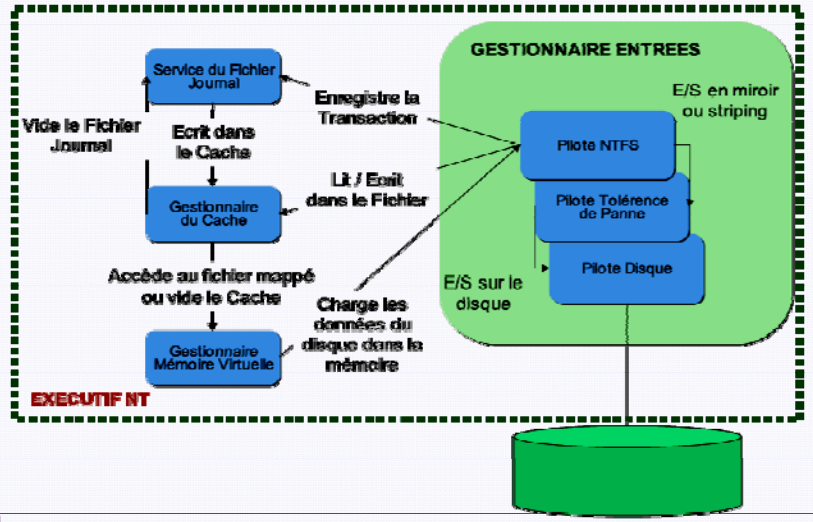
➢ Structure générique:

- PtrDeviceObject :
  - Informations sur les caractéristiques et l'état du périphérique.
- PtrDriverExtension :
  - Pointeur vers une zone mémoire définie par l'utilisateur
- DriverEntry :
  - Initialisation du driver.
- PA_Dispatch :
  - Appelée lorsqu'un programme essaye d'avoir un handle sur le périphérique.
- PA_DispatchIoctl :
  - Gère les appels de fonction définis par le programmeur du driver en fonction du matériel qu'il supporte.
- PA_UnloadDriver :
  - Effectue le nettoyage des objets restants en mémoire, libération des handles



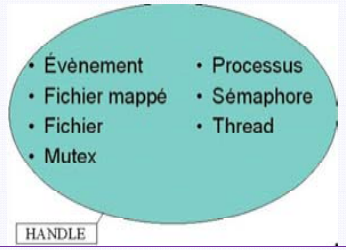
Noyau Windows NT : *Gestionnaire d'entrée-sortie*

➢ Systèmes de Gestion de fichiers

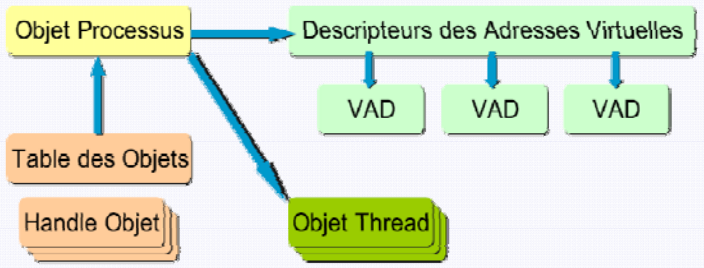


Noyau Windows NT : *Gestionnaire d'objets*

- Sous NT, les ressources de l'OS sont représentés par des objets :
  - Processus, Thread
  - Périphériques, Sémaphores, Mutex, Evenement
- Un objet possède une ACL (Acces Control List)
- Le gestionnaire d'objet crée, modifie et supprime ces objets. Supprime aussi les objets orphelins.
- Il fournit les handles
- Il gère les ressources consommées par chaque objet



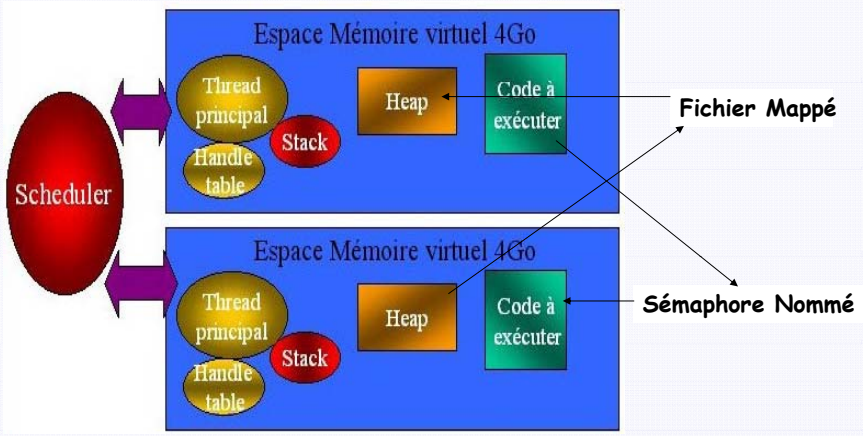
Noyau Windows NT : *Gestionnaire de process*



- **Crée, supprime et modifie l'état des processus et des threads**
- **Renseigne sur l'état des processus et des threads**
- **Ne cadence pas les threads**

Noyau Windows NT : *Gestionnaire de process*

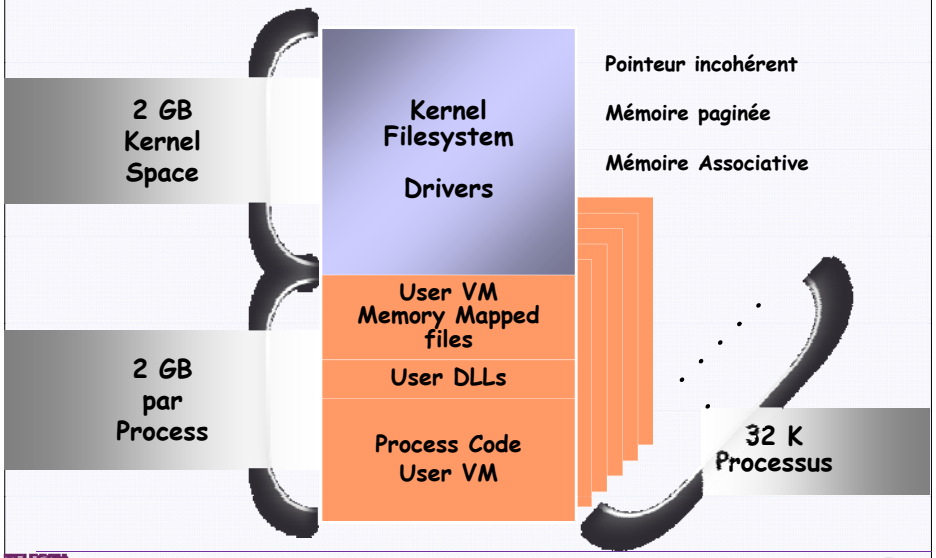
- **Multi Thread** ***Synchronisation - Communication*** **:**



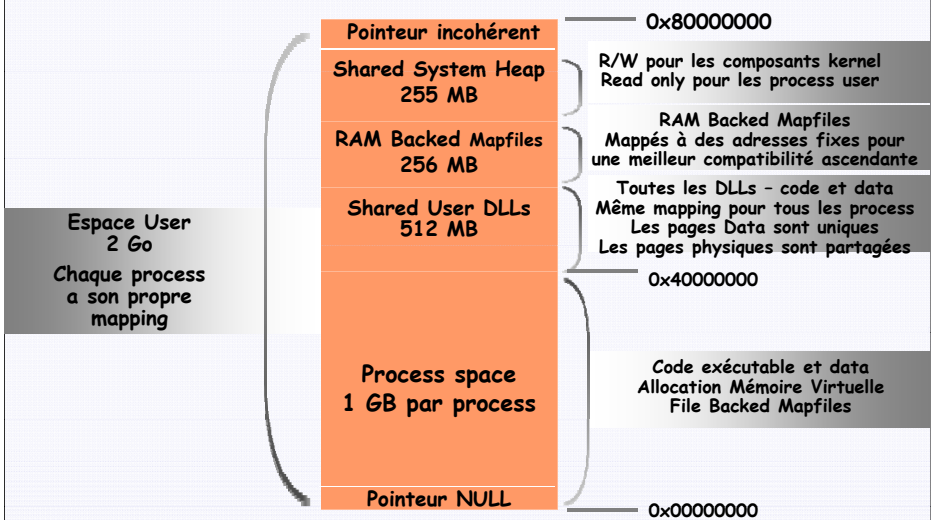
Noyau Windows NT : *Architecture Mémoire*

- Gère une mémoire linéaire sur 32 bits
- Affecte à chaque processus 4Go d'espace mémoire
  - 2Go pour le système
  - 2Go pour l'application
- Mémoire gérée par Pagination (4 ou 8Ko) allouée à la demande
- Gère le swap (pagefile.sys)

Noyau Windows NT : *Architecture Mémoire*

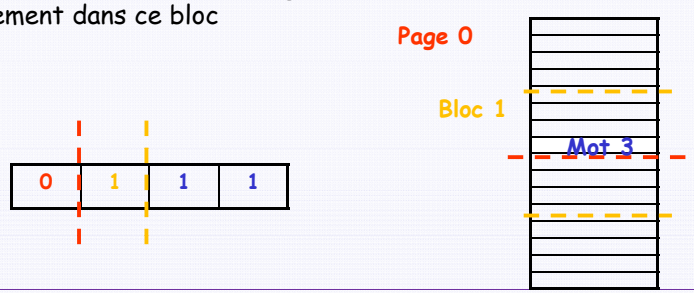


Noyau Windows NT : *Architecture Mémoire*



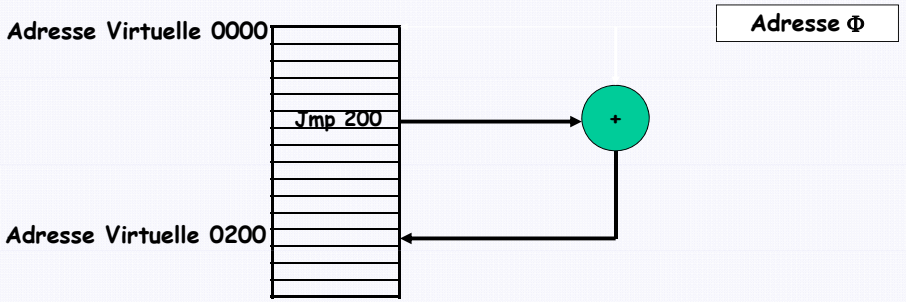
Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- Adressage Virtuel :
  - Gestion facilitée de la mémoire
  - Programmes relogeables et translatables
  - Le lien avec l'adresse physique est calculé lors de l'accès à la zone mémoire
- Principe de découpage des adresses (split and join):
  - Une adresse binaire : assemblage d'une numéro de bloc et d'un déplacement dans ce bloc



Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- **Adressage Basé : mécanisme le plus simple**
  - Adresses du programme relatives à l'adresse 0
  - Adresse d'implantation du programme en mémoire physique dans le registre de base
  - Calcul fait par la MMU



Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

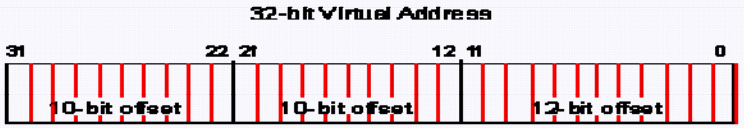
- **Adressage Basé : Inconvénients**
  - Fragmentation de l'espace physique
  - Difficultés de compactage
  - Difficulté d'allocation (blocs de taille différente)

⇒ **PAGINATION**

- **Principe :**
  - Espace virtuel découpé en page de taille fixe
  - Espace physique découpé en bloc de même taille
  - Mécanisme : Table de Pages Contient des Informations:
    - Etats (6 s/s Windows: Valide, Modifiée, libre, …)
    - Protection
    - Adresse physique ou auxiliaire

Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- **Le VMM utilise une table de gestion des pages à deux niveaux :**
  - **Page directory ou répertoire des pages,**
  - **Page table ou table des pages.**

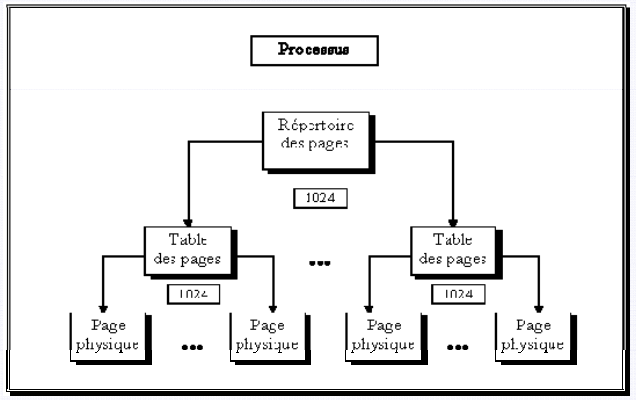


- **1024 PDE contenant 1024 PTE de 4096 octets**

  ⇒ 4 Go

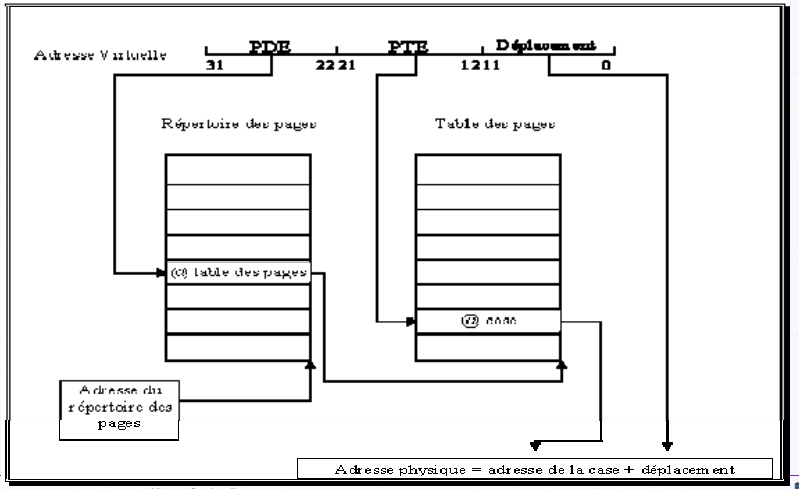
Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- **Structure Arborescente:**
  - PTE : 32 bits (5 : Protection; 20 : Adresse; 4: Pagefile; 3: Etat
  - Problème : Taille de cette structure en RAM!



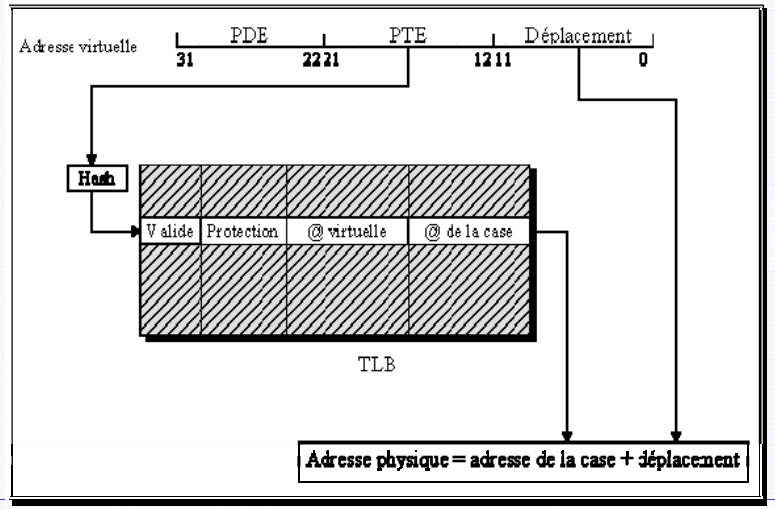
Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- **Traduction d'adresses:**
  - Problème : Temps de translation (accès mémoire supp.)



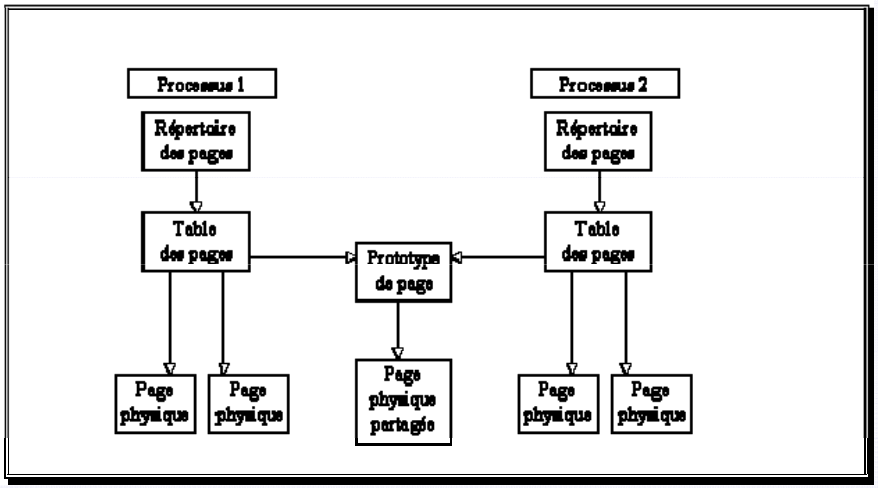
Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

- **Mémoire associative: Translation Lookaside Buffer (TLB)**



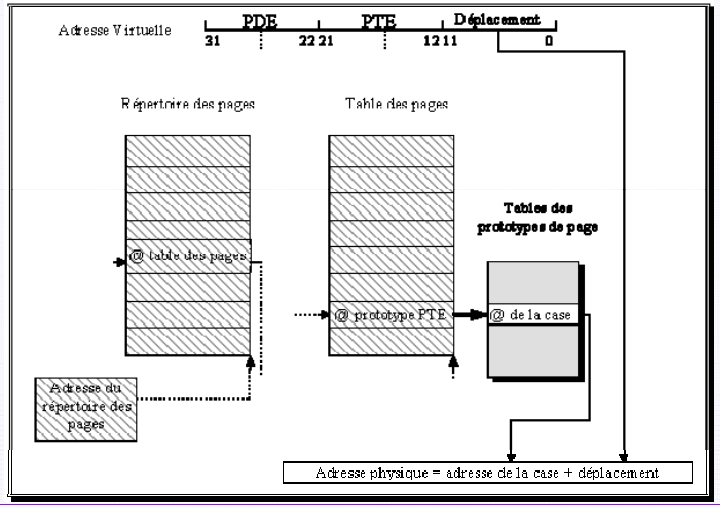
Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

➢ **Partage de données: Table de prototypes de page**



Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

➢ **Partage de données: Table de prototypes de page**



Noyau Windows NT : *Gestionnaire de mémoire virtuelle (VMM)*

➢ **Politique de chargement :**
- Pré-paging ou Pre-Fetch (Super-Fetch depuis Vista)

➢ **Politique de remplacement :**
- Faute de pages !
- FIFO ou LRU

➢ **Segmentation + Pagination**
- **Codes, Données, Pile**

➢ **Trashing**

➢ **Swap**

Noyau Windows NT : *Gestionnaire d'interface graphique (GDI)*

➢ **Windows fournit un "quasi-standard" d'interface graphique et permet donc à un utilisateur de "prendre en main" très rapidement une application et son environnement.**

➢ **Notons l'importance mais aussi la complexité du développement d'interfaces graphiques.**
- **Windows est une interface complexe et touffue qui propose selon le mode de comptage entre 750 et 1250 appels de fonctions possibles (API natif de Windows -API = Application Programming Interface-);**
- **L'API n'est pas spécialement conçue pour le C++ (antérieure à C++ et l'approche objet)**

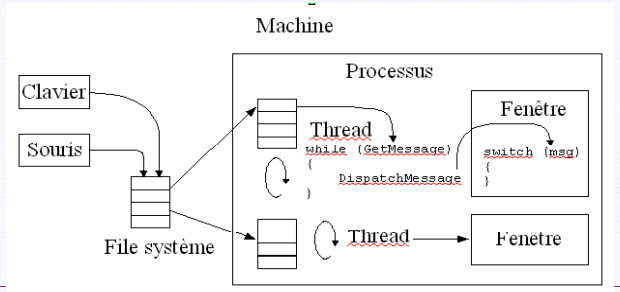
➢ **La programmation d'application Windows directement avec les fonctions de l'API natif est une tâche ardue, aussi les concepteurs d'environnement de programmation ont développé différents outils (bibliothèques, ...) qui permettent de simplifier ce travail.**
- **La bibliothèque MFC (Microsoft Foundation Classes) de Visual C++ est une bibliothèque de classes d'objets qui encapsule dans des objets de plus haut niveau les fonctionnalités de l'API natif.**

Programmation événementielle:

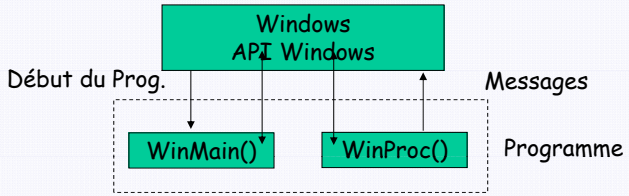
- Un évènement est une action réalisée par l'utilisateur sur l'interface
  - Windows doit être capable de récupérer les informations liées à l'évènement
  - Transmettre cette information à l'application concernée pour déclencher le traitement de cet évènement
- Programmation avec une boucle infinie de gestion de messages
  - Les messages sont lus et dispatchés un à un,
  - La notion de fenêtre est théoriquement facultative.
- Au plus une file de messages par thread
  - la file de messages est créée à la demande,
  - PeekMessage peut forcer sa création

Les files de messages

- Une seule file de messages système reçoit les messages clavier, souris, etc.
- Une file de message par thread
- les messages sont postés en mode FIFO,
- Certains messages sont exécutés en synchrone (gestion du focus, du curseur, activation des fenêtres...).



- **Sous sa forme la plus réduite un programme Windows, écrit uniquement avec l'API Windows comporte, 2 fonctions:**
  - La fonction WinMain() qui est l'équivalent de la fonction main() d'un programme console.
  - La fonction WinProc() que Windows appelle pour traiter les messages destinés à l'application
- **Ces deux fonctions composent un programme complet mais elles ne sont pas directement liées. C'est Windows qui gère les appels à ces deux fonctions.**



Fonction de gestion des messages: WinMain()

```
while (Getmessage(&msg, NULL, 0,0))
{       TranslateMessage( &msg);
        DispatchMessage(&msg);
}
```

- **Cette boucle fonctionne tant que les messages reçus ne sont pas un message de terminaison de l'application:**
  - **GetMessage extrait un message de la file d'attente,**
  - **TranslateMessage convertit le message extrait, si nécessaire**
  - **DispatchMessage demande à Windows d'appeler la fonction WinProc() dans l'application pour traiter le message**

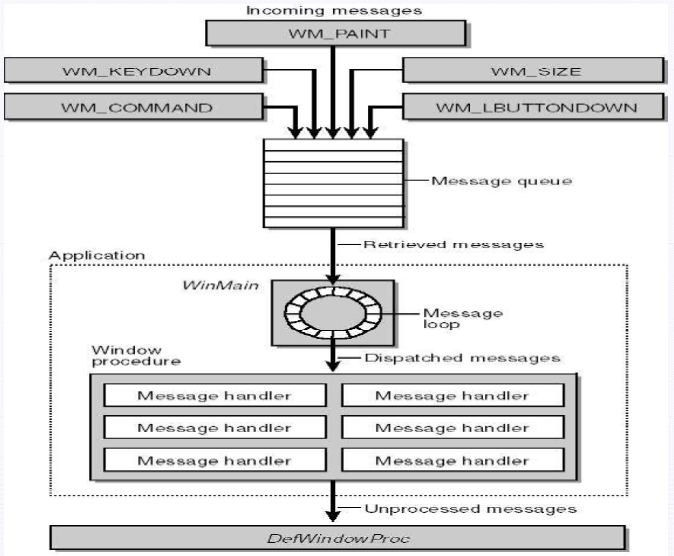
Fonction de traitement des messages: WinProc()

- **Le décodage des messages est effectué à l'aide d'une instruction conditionnelle multiple**

```
switch (message)
{       case WM_LBUTTONUP : ……..; break;
        case WM_PAINT : ……..; break;
        ……..
        default : …….
}
```

Taxonomie des messages

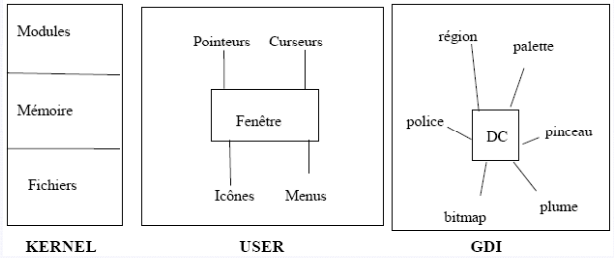
- Il existe environ 250 types de messages prédéfinis dans Windows. On peut distinguer 8 catégories principales de messages dont:
  - les messages hardware : entrées d'informations avec le clavier ou la souris
    - WM_LBUTTONDOWN, WM_LBUTTONUP,
    - WM_CHAR, WM_KEYDOWN
  - les messages de maintenance des fenêtres :
    - notification, demande d'action (WM_CLOSE, WM_PAINT, ...)
  - les messages de maintenance de l'interface utilisateur concernant les menus
    - WM_COMMAND,WM_INITMENU, ...
  - les messages de terminaison (WM_QUIT, ...)
  - les messages privés (pour fenêtres spécifiques : boîtes à liste, bouton, ...)
  - les messages de notification concernant les ressources système
    - WM_TIMECHANGE,...



Windows repose sur trois bibliothèques :

KERNEL, USER, GDI

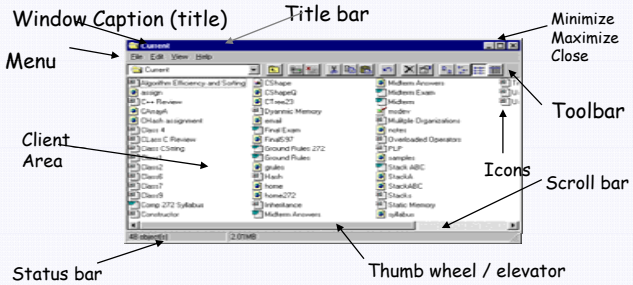
KERNEL gère le support à bas niveau:
- des fichiers d'entrées/sorties,
- de la mémoire,
- du chargement des programmes,
- des opérations classiques d'un OS



- USER et GDI font appel aux services de KERNEL
  - USER s'occupe de la création et de la gestion des objets d'interface utilisateur (fenêtres, menus, boîtes de dialogues, curseurs, icônes, …)

l'objet primordial est la fenêtre

- GDI : Graphical Device Interface
  - assure l'indépendance vis-à-vis des périphériques
  - gère les dessins de fenêtres, les menus, les boîtes de dialogue, etc...
- **USER utilise les services de GDI pour les tracés de ses objets**



Structure de l'API

- **Non polymorphique et hétéroclite**
  - pas de type de type HANDLE unifié,
  - création spécifique des objets,
  - méthodes spécifiques de manipulation,
  - destruction spécifique.
- **API à état externe**
  - moins de paramètres,
  - notion de Device Context.
    - **Le Device Contexte contient tous les paramètres courants de l'appelant**
    - **Il doit être fourni à quasiment toutes les fonctions GDI**

La manipulation des DC est lourde et omniprésente


```
typedef struct _WNDCLASS {
    UINT      style;
    WNDPROC   lpfnWndProc;
    int       cbClsExtra;
    int       cbWndExtra;
    HINSTANCE hInstance;
    HICON     hIcon;
    HCURSOR   hCursor;
    HBRUSH    hbrBackground;
    LPCTSTR   lpszMenuName;
    LPCTSTR   lpszClassName;
} WNDCLASS, *PWNDCLASS;
```

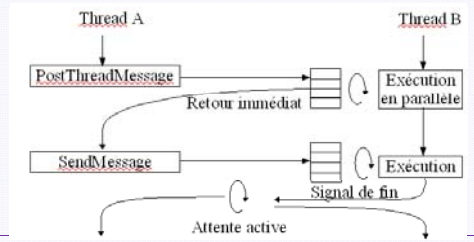
```
HWND CreateWindow(
  LPCTSTR lpClassName,
  LPCTSTR lpWindowName,
  DWORD dwStyle,
  int x,
  int y,
  int nWidth,
  int nHeight,
  HWND hWndParent,
  HMENU hMenu,
  HINSTANCE hInstance,
  LPVOID lpParam
);
```


Notion de ressources

- **Les ressources centralisent les données hors du code :**
  - **Facilitent les traductions,**
    - **prototypage facilité.**
  - **Elles peuvent contenir :**
    - **des boîtes de dialogue,**
    - **des chaînes de messages,**
    - **des accélérateurs, etc.**
  - **Adaptées aux applications formulaires**
    - **conception et codage facilités,**
    - **applications statiques et peu souples.**
  - **Gestion des ressources multilingues complexe**
    - **pas de langue par défaut,**
    - **chargement dynamique de DLL de ressources.**
  - **Facteur de plantages nombreux**
    - **gestion des ressources de sous-systèmes complexes,**
    - **nécessité de retrouver le module tout le temps.**

Programmation dangereuse

- **Les procédures de fenêtres doivent être réentrantes**
  - **la gestion de l'état du programme est technique,**
  - **les données globales et variables membres doivent être protégées (perturbe gravement les langages objets).**
- **Les communications interthread sont risquées**
  - **SendMessage est synchrone et sujette à interblocages,**
  - **Marshalling non réalisé pour les messages utilisateurs,**
  - **PostMessage/PostThreadMessage requis,**
  - **Appels de méthodes OLE non contrôlables**

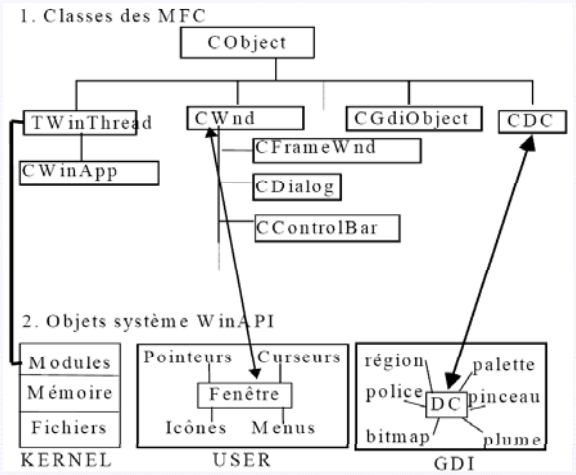


Les MFC (Microsoft Foundation Classes)

- **Encapsulation C++ de l'API :**
  - **encapsulation quasi directe des concepts GDI,**
  - **API encore une fois très vaste.**
- **Jeu de macros :**
  - **déclaration des messages,**
  - **déclaration utilitaires.**

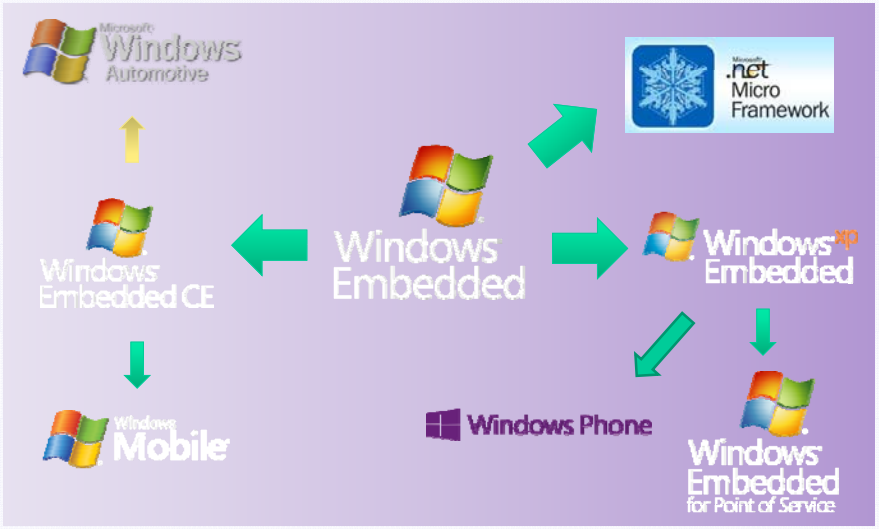
Permettent d'éviter beaucoup de frais

Les Relations entre les MFC et l'API Windows



- CObject
  - CCmdTarget
    - CWinThread
      - **CWinApp**
    - CWnd
      - **CFrameWnd**
        - CMDIChidWnd
        - CMDIFrameWnd
      - **CView**
      - CDialog
    - **CDocument**
    - CDocTemplate
      - CMultiDocTemplate
      - CSingleTemplate
    - CGdiObject
    - CDC
      - CClientDC
      - CPaintDC

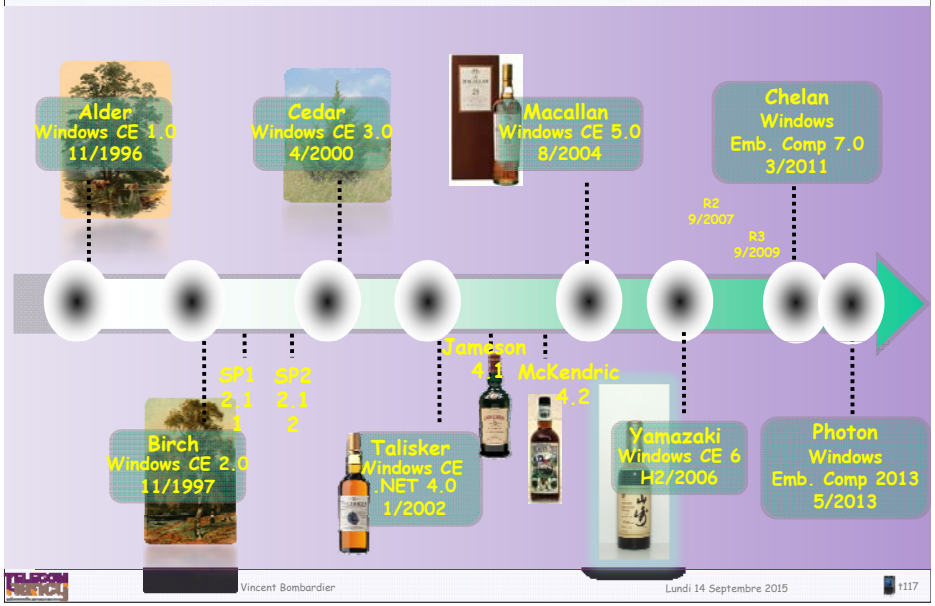
Windows® Embedded CE 6.0 R2



Choisir une plateforme Windows® Embedded

	.net Micro Framework	Windows Embedded CE	Windows XP Embedded
Examples	Noeuds de capteurs, écrans Sideshow, équipements médicaux, télécommandes	GPS, Handhelds, PDAs, Automotive, Set Top Boxes	Clients légers, ATMs, Kiosques
Features	Connecté, Petit, "portable", IHM Graphique	Connecté, IHM Graphique, Serveur, Browser, RAS, DirectX	Performance de type PC, Réseau PC
Taille empreinte	200-400Ko Code .Net inclus	300Ko+ sans .NET CF 12Mo+ avec .NET CF	40Mo +
Alimentation	Très basse cosommation	Basse consommation	Alimentation PC
CPU	ARM7, ARM9 sans MMU	X86, MIPS, SH4, ARM, avec MMU	X86
Temps réel	Pas de temps réel dur	Temps réel dur	Temps réel dur avec solutions tierces parties
.Net vs. code natif	Natif : interop uniquement, Managed : .NET MF	Natif : de base, Managed : .NET CF	Natif : de base, Managed : .NET Framework

Windows CE : *Historique*



Windows EMBEDDED CE

- **Windows Embedded CE 6.0 n'est PAS**
  - **Windows Mobile 6.0**

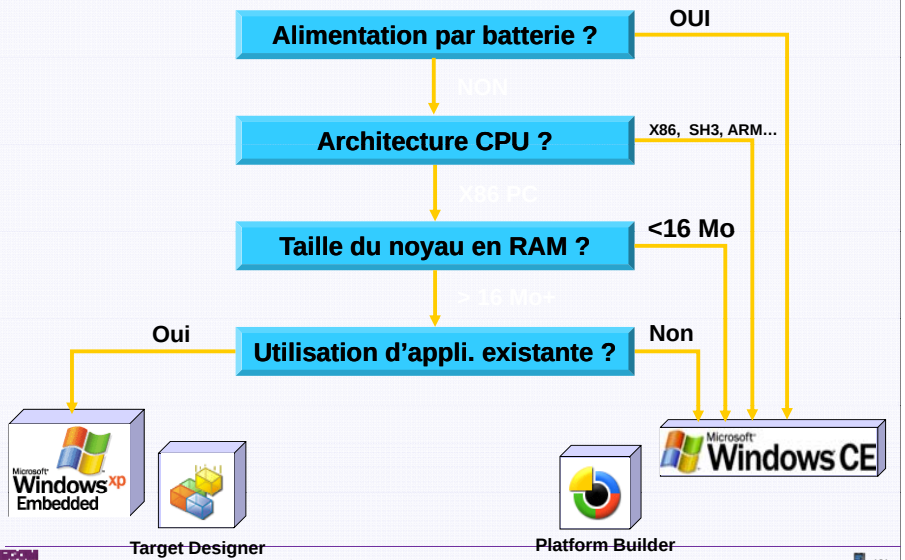
- **Windows Embedded CE 6.0 c'est…**
  - Un OS 32-bit, temps-réel, multitâche
  - Modulaire
    - Disponible sous la forme d'un ensemble de composants
    - On utilise Visual Studio 2005 et le plug-in Platform Builder pour configurer et générer l'image
  - Modulaire
    - La taille de l'empreinte dépend des fonctionnalités choisies
  - Supporte une large variété de CPUs
    - x86, ARM, MIPS and SH4
  - **Et sera le cœur de Windows Mobile …**

Windows EMBEDDED CE

- OS modulaire et compact
- Connectivité
- Capacitées temps-réel
- Support multimédia et multilangues
- Outils de développement
- Sources disponibles

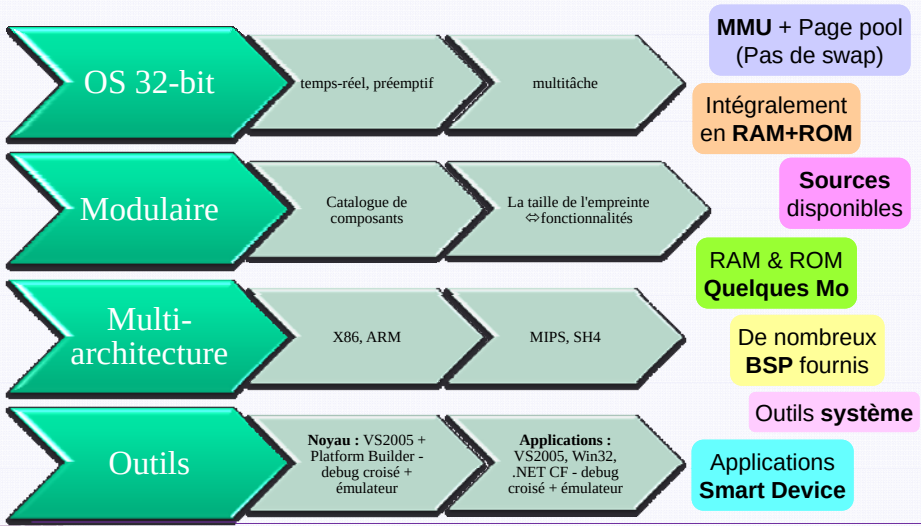
La famille Windows

Choix d'un OS embarqué Windows

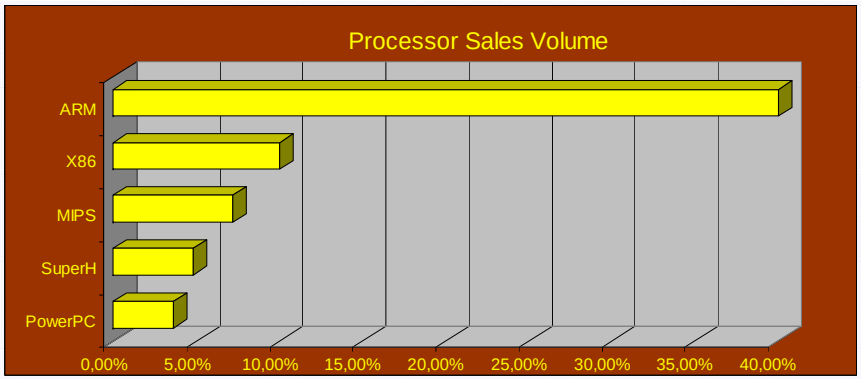


Windows CE dans l'industrie

Caractéristiques Windows CE



Ventes annuelles 2008 32/64 bits



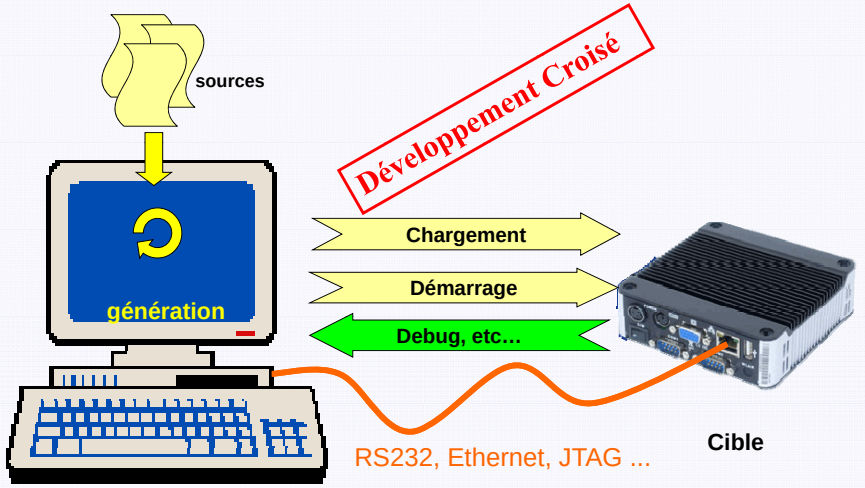
Based on 2002 sales data

Particularités de CE

- Utilise une « Base de Registre »
  - Paramètres du système (idem. /etc/… dans UNIX)
  - Paramètres des Applications
- Module d'interface graphique intégré
  - Graphic and Windowing Environment Subsystem (GWES)
- Chargement des périphériques (Device manager)
- Modules de Communication
  - NDIS 5.1, IrDA, Bluetooth, WiFi, Ethernet, TCP/IP, etc…
- Applications intégrées
  - Browser, Media Player
  - Servers HTTP, FTP, TELNET, etc.
  - Shells, Viewers, SQL server …




Développement de l'OS



CPU Supportés

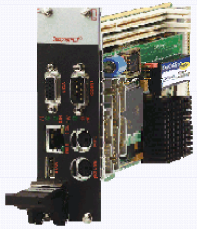
Architecture	CPU
ARM	freescale i.MX
	SMDK
	Texas Instruments MAP
	Etc.
MIPS	NEC Vr4122
	NEC Vr5432
SH Renesas Everywhere you imagine	SH4-7750
	SH3-7729 (CE 4.x)
x86	intel Pentium M Atom
	AMD-Geode, VIA, DM&P

Windows CE : *Les outils*

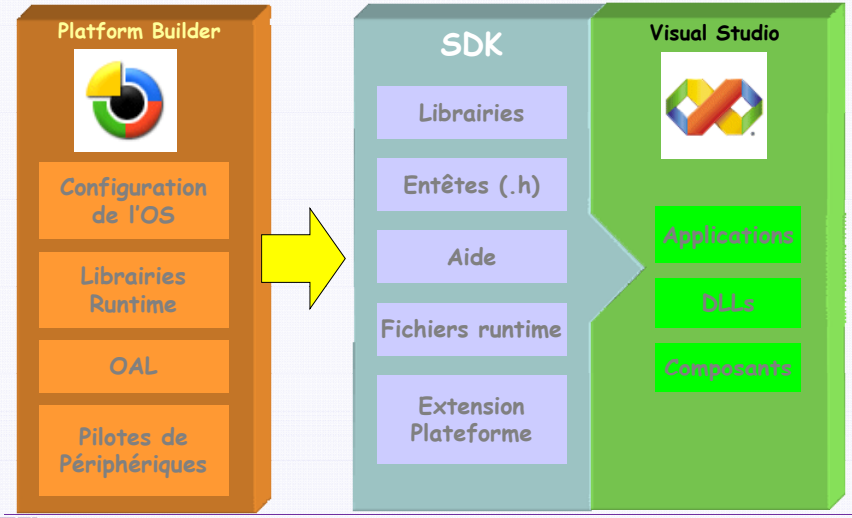
ETAPE 1
Fabrication du système
Board Support Package (BSP)

ETAPE 2
Développement applicatif
Code Natif (Win32)
Code Managed

Microsoft Visual Studio 2005 Microsoft .net




Software Developpement Kit

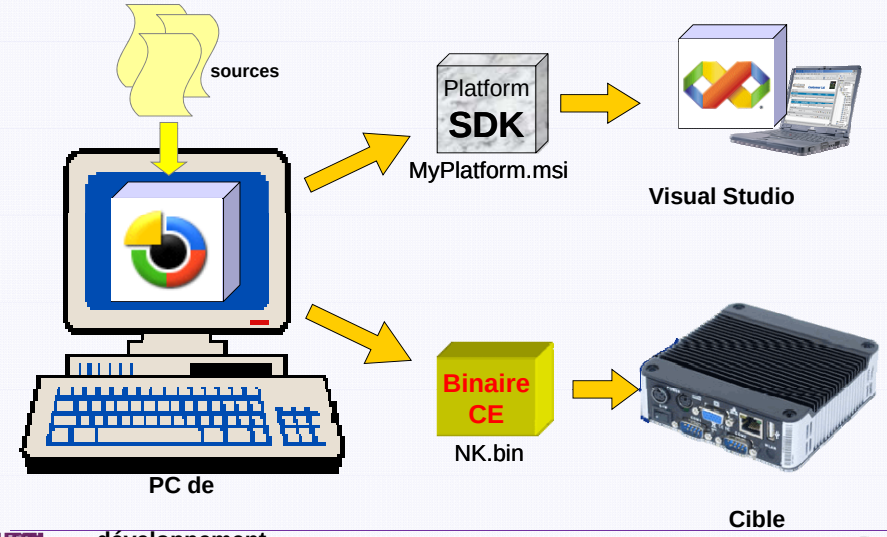


Platerforme Builder

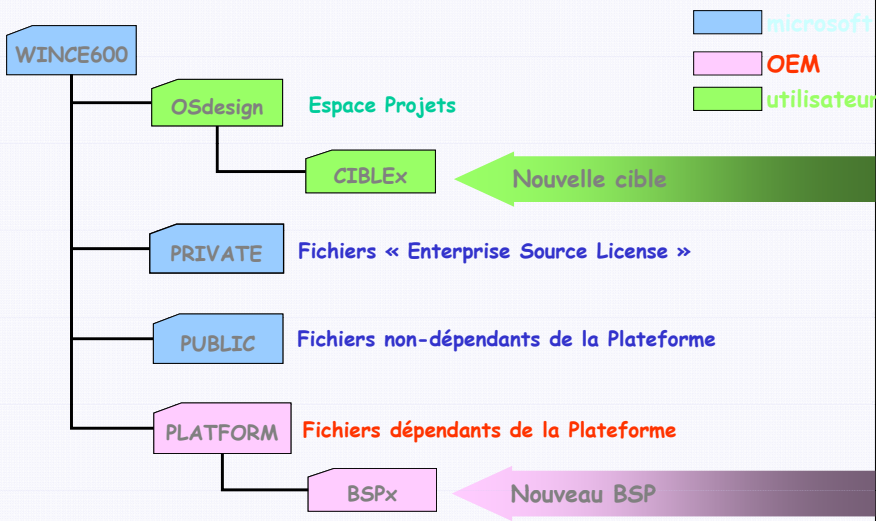
- **Créateur de systèmes d'exploitation**
  - Générateur de noyaux
  - Producteur de SDK

- **Code du noyau CE**
  - Composants du noyau
  - Pilotes de périphériques
  - Piles de protocoles
  - Shell et Applications
  - …

Ce que produit Platform Builder

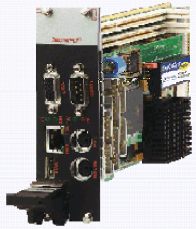


Sources Windows CE

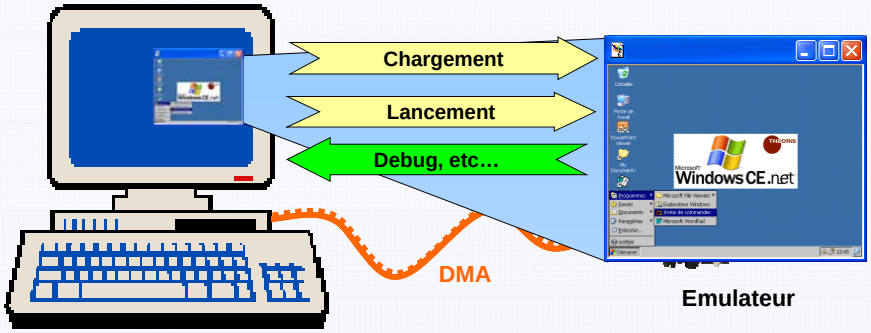


Exemple développement (complet)

ETAPE 1
Fabrication du système



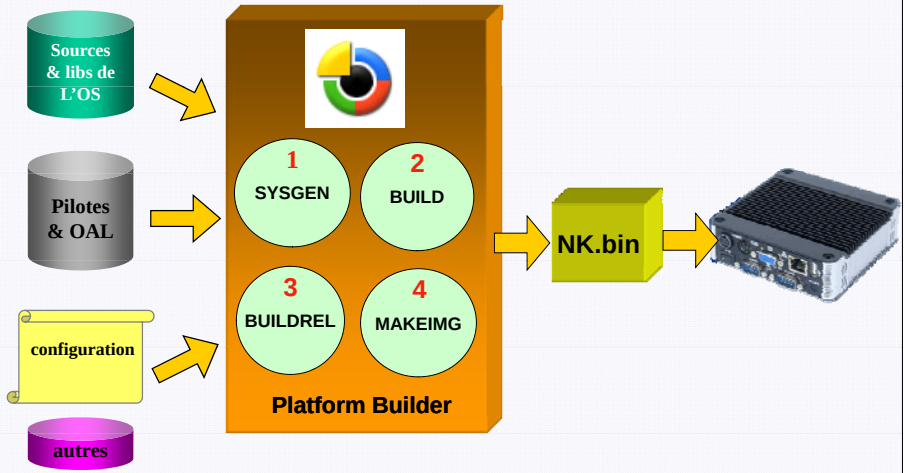
Développement sans cible



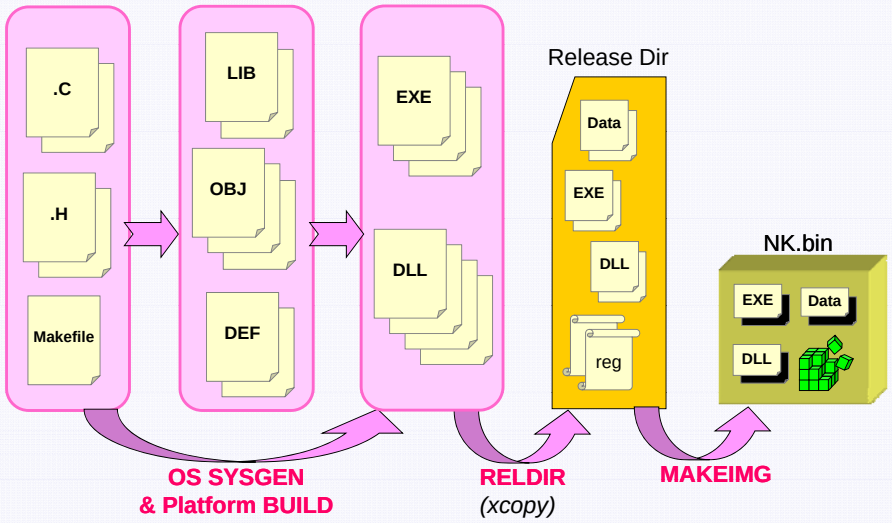
Emulateur ARMV4I

- Cible virtuelle sur station
  - PocketPC
  - SmartPhone
  - Propriétaire
- Développement sans cible
  - Emulation du code machine (ARM)
  - Se comporte comme une cible (outils croisés)
  - Fourni avec son SDK
- Avantages
  - Gain de temps
  - Gain de matériel

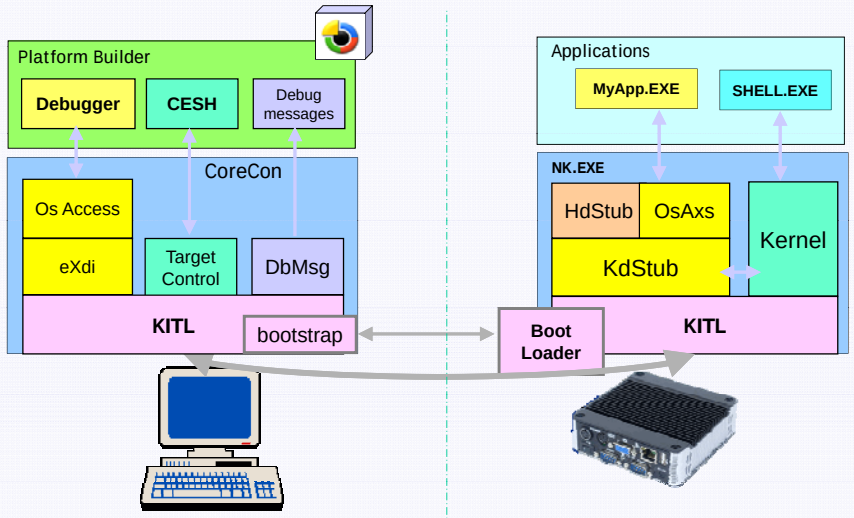
Les étapes du "Build"



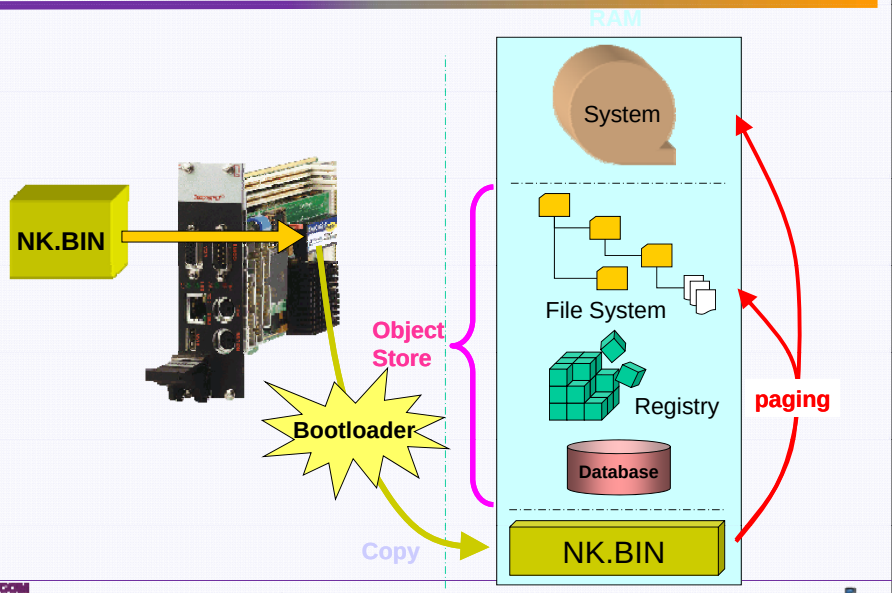
Détail du processus de Build



Connectivité du noyau

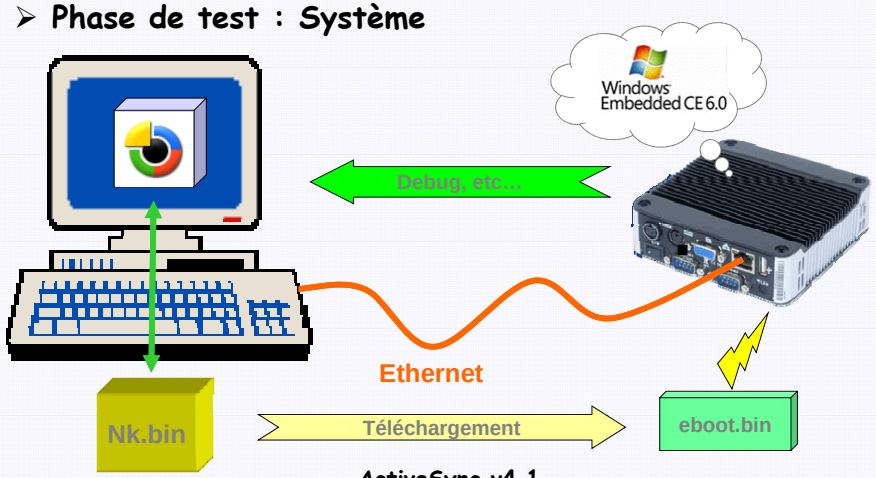


Démarrage de Windows CE

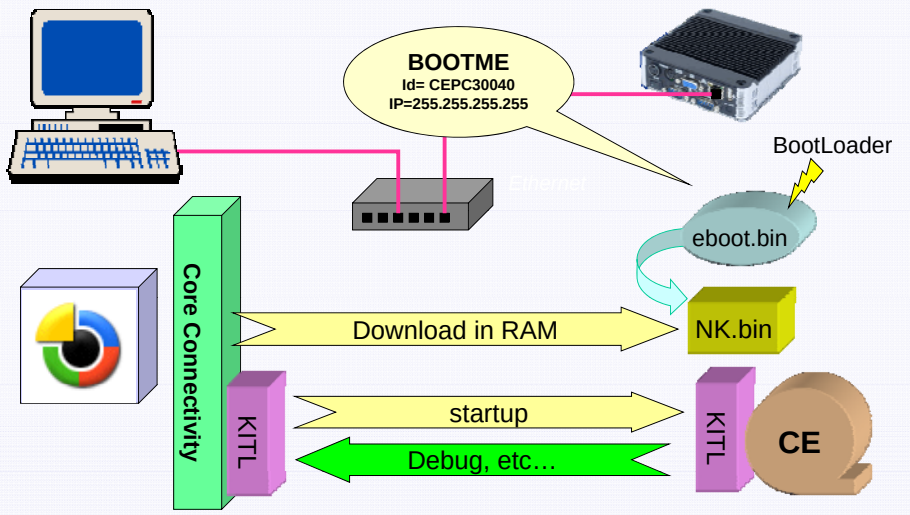


Exemple développement (complet)

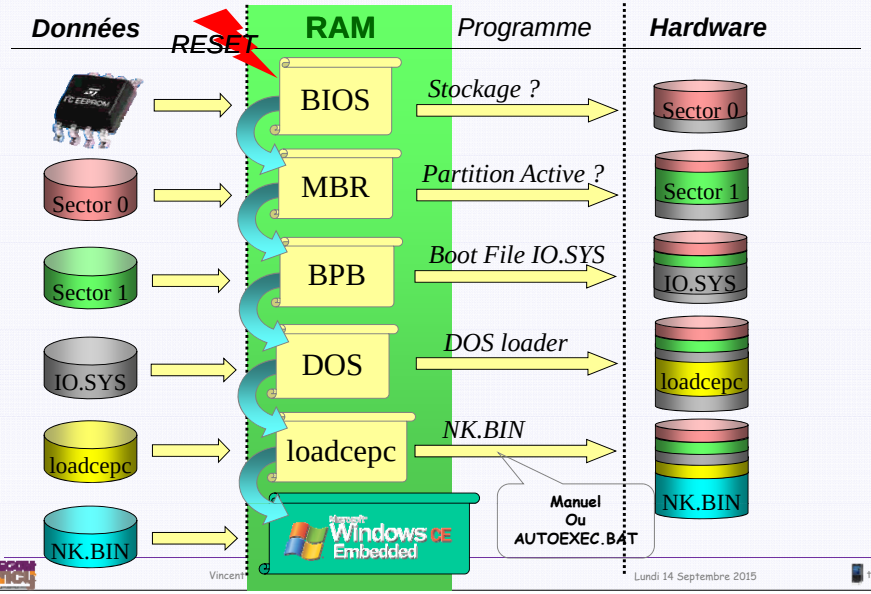
➢ Phase de test : Système



Utilisation du BOOTME



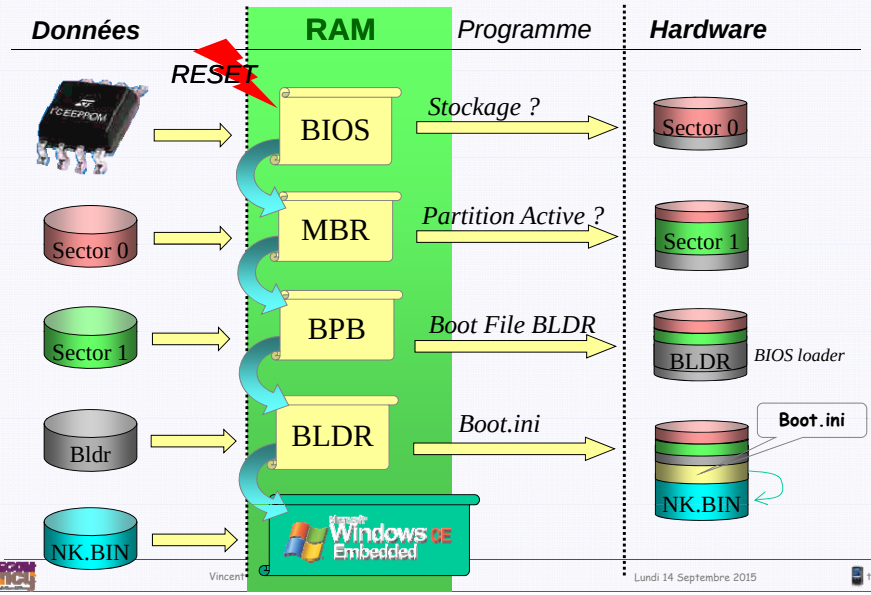
Démarrage CEPC – DOS Loader



Média Bootable – DOS Loader

- Fichiers de boot dans
  - PLATFORM/CEPC/SRC/BOOTLOADER/DOS
  - Préparer un média bootable DOS (IO.sys)
- DOS LOADER *(répertoire BOOTDISK)*
  - Copier sur le média
    - Loadcepc.exe
    - Eboot.bin
    - Config.sys
    - Autoexec.bat
    - NK.bin *(depuis FlatRelseaseDir)*

Démarrage CEPC – BIOS Loader



Média Bootable – BIOS Loader

- Fichiers de boot dans
  - PLATFORM/CEPC/SRC/BOOTLOADER/ BIOSLOADER/DISKIMAGE/SETUPDISK
  - Il faut deux médias, un (M1) sera préparé pour le BIOSLOADER, un autre (M2) sert à la préparation
- BIOS Loader
  - Rendre le média M2 bootable
  - Copier le contenu de SETUPDISK sur M2
  - Insérer M1 et M2 puis démarrer sur M2 (setup)
  - Lancer MKDISK avec pour cible le volume de M1
  - Copier NK.bin & Splash.bmx sur M1

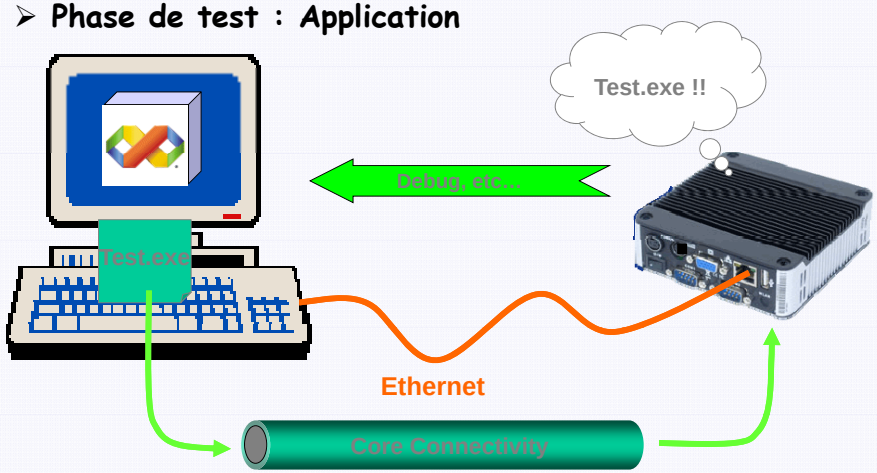
Exemple développement (complet)

ETAPE 2
Développement applicatif

Code Natif (Win32)
Code Managed

Windows CE : *Exemple développement (complet)*

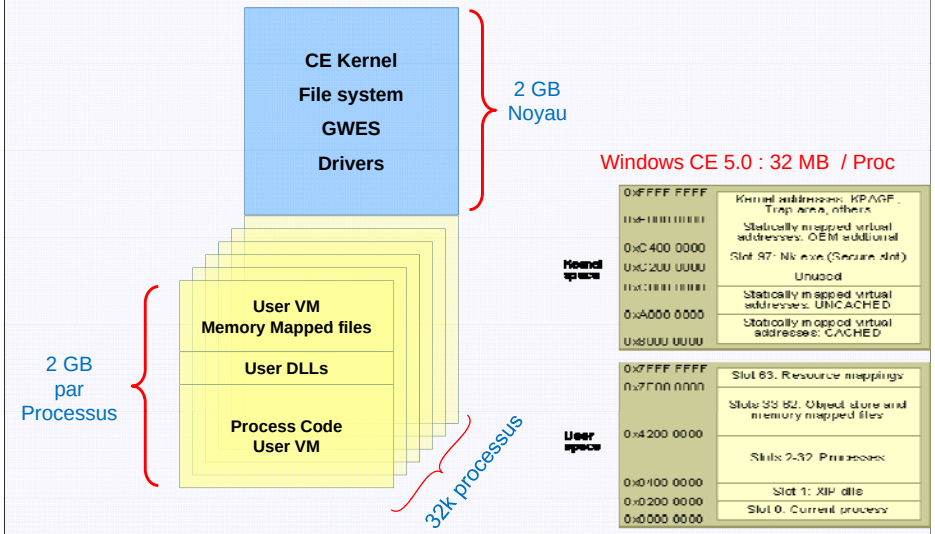
- **Phase de test : Application**



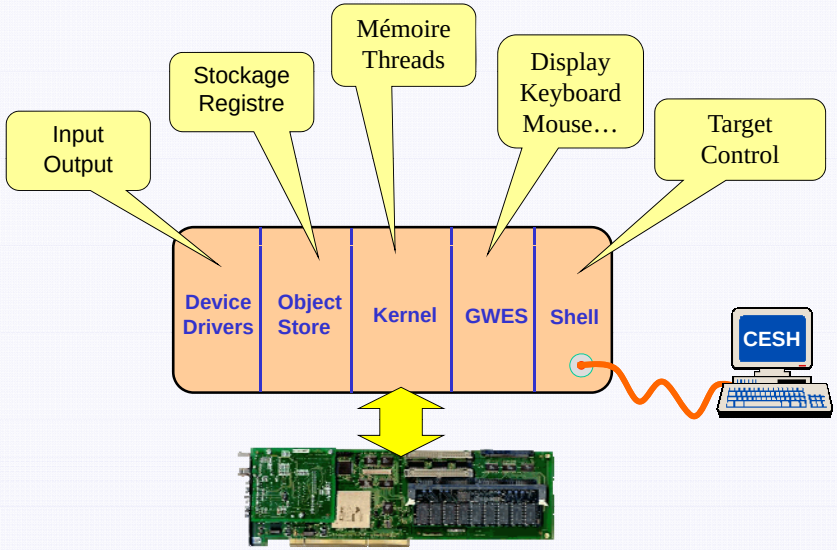
Le Noyau

- **Encapsulation du Noyau**
  - **Les composants "critiques" de l' OS ont été déplacés dans l'espace kernel**
- **Amélioration des performances du système**
- **Amélioration de la sécurité et de la robustesse**
- **Compatibilité ascendante étendue**

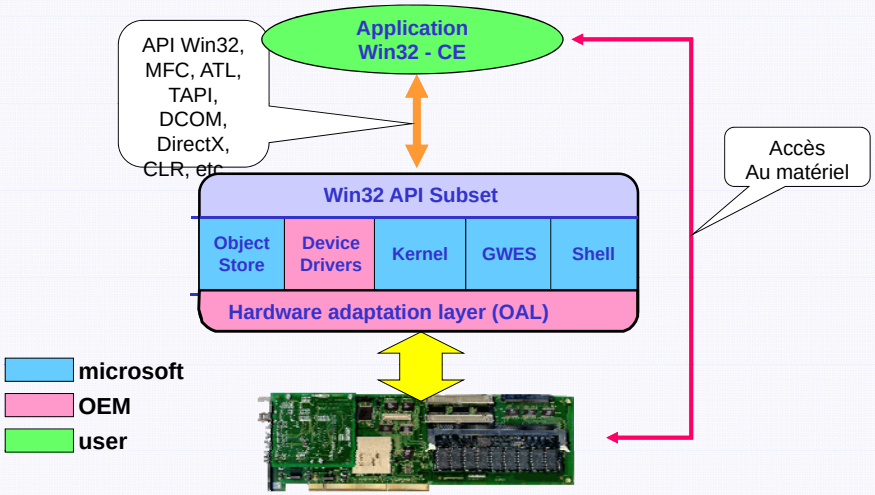
La mémoire depuis CE 6.00 : 4 GB



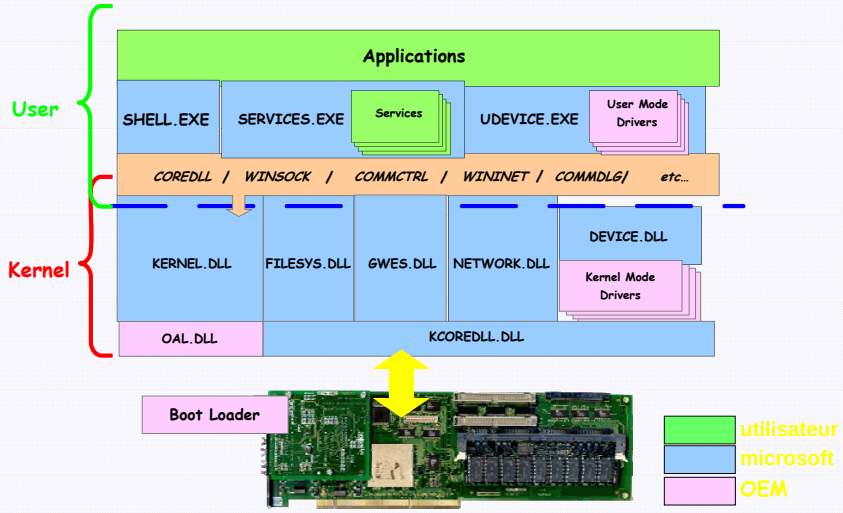
Les services de CE



Architecture Générale



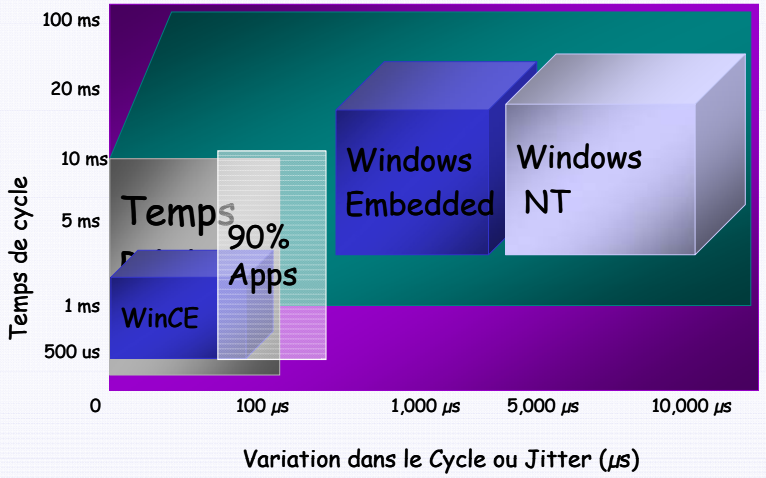
Architecture CE 6.0



BSPs proposés pour CE 6.0

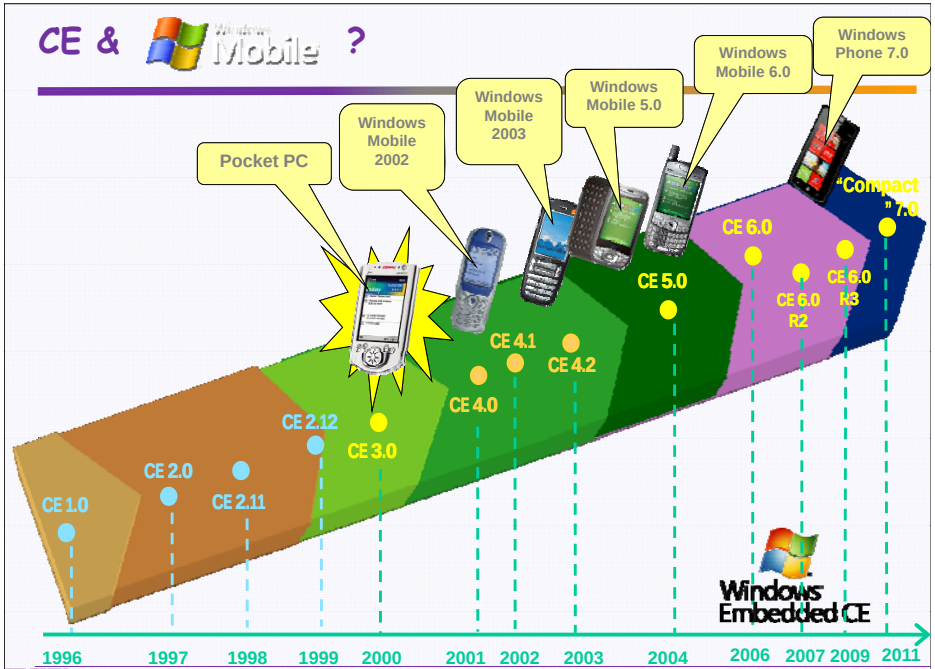
Famille	BSP	Kernel	Sources inclus ?
ARM	Intel Mainstone III (C-Step)	ARMv4i	Oui
	Device Emulator	ARMv4i	Oui
	Board	ARMv4i	Oui
	TI OMAP 2420 (ARM v6 core)	ARMv4i	Oui
MIPS	NEC Rockhopper SG2 Vr5500	MIPSII & II_FP, MIPSIV & IV_FP	Oui
SH4	Renesas Aspen	SH4	Oui
x86	x86 (CEPC)	X86	Oui

Le temps-réel défini par l' OMAC*



*OMAC : Open, Modular Architecture Control

CE & Windows Mobile ?



Windows Mobile : *Historique*

- Windows Phone 8.0 – Apollo - 2013 (base: Windows Embedded 8)
- Windows Phone 7.5 – 2011 / 7.8 - 2013
- Windows Phone 7 - Mango – 2010
- Windows Mobile 6.5 - 2008
- Windows Mobile 6.0 – Crossbow – 2007;
- Windows Mobile 5.0 – Magneto – May 10, 2005;
- Windows Mobile 2003 Second Edition –March 24, 2005;
- Windows Mobile 2003 – Ozone – June 23, 2003;
- Pocket PC 2002 – Merlin – October 4, 2001;
- Pocket PC 2000 (Palm-size PC v2) – Rapier – April 19 2000;
- Palm PC v1.0/1.2 – Gryphon/Wyvern – January 8, 1998 (3 months later Palm PC was renamed to Palm-size PC).

Windows Mobile : *Fonctionnalités*

- **Main innovations in Windows Mobile 2005:**
  - The single-handed control of the device, not involving the touch screen.
  - Greatly improved support of QWERTY-keyboards
  - The support of applications for Windows Mobile 2003 for Smartphone. The key devices that will illustrate this approach this year are Asus p515, HTC Blizzard.
  - Brand new memory organization, data safety doesn't depend on battery charge level
  - Possibility to add photos and assign personal melodies to the contacts.
  - Greatly improved Word Mobile (it keeps formats while editing, shows pictures); to a smaller extent - Excel Mobile. New application added - PowerPoint Mobile.
  - Built-in API for GPS and the camera.
  - Integrated localization

Windows Mobile : *Fonctionnalités*

- **Major changes introduced in Windows Mobile 6.0:**
  - Revamped outlook, refined icons;
  - Totally re-mastered sound theme;
  - Increased interface speed;
  - Bundled IP-telephony (VoIP, SIP standard);
  - Smart Dial 2.0, fast search through contacts, call history; removable on-screen virtual keyboard;
  - Any tune can be assigned to any event;
  - Support for shortcuts in Outlook Mobile;
  - Full-fledged meetings arrangement system;
  - Support for HTML in messages;
  - Smart Filter – handy and fast search through messages;
  - Numerous MS Exchange 2007-tailored changes;
  - Integration with Windows Live online-services;
  - Marketplace – purchase, download, and install directly from the device;
  - Encryption – bundled memory card encryption

Windows Mobile : *Fonctionnalités*

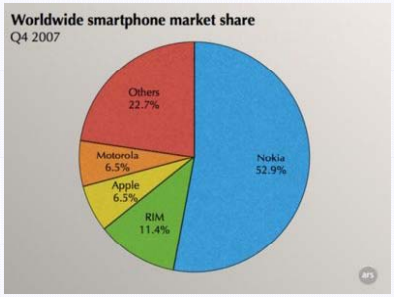
- **Evolutions Majeures pour Windows Mobile 6.5:**
  - Internet Explorer Mobile 6.0
  - La librairie graphique GAPI abandonnée;

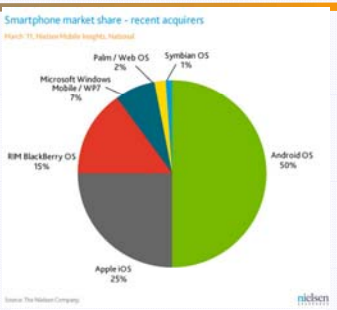
- **WinPhone 7 – 7.5 - 8:**
  - Navigation au doigt ;
  - Clavier virtuel adapté à une manipulation tactile ;
  - Gestion d'un accéléromètre;
  - Utilisation du capteur photo numérique pour détecter les déplacements;
  - Internet Explorer Mobile 8.0
  - Silverlight - Flash

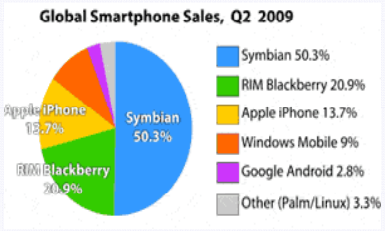
  - La création de sujet de conversation dans les courriels et SMS.
  - Prise en charge des contacts Twitter et linkedln

  - Internet Explorer Mobile 9.0

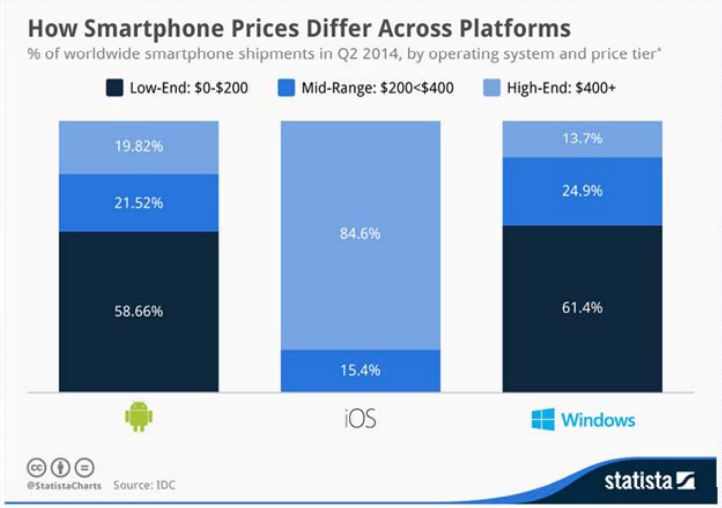
Windows Mobile : *Part de marché*







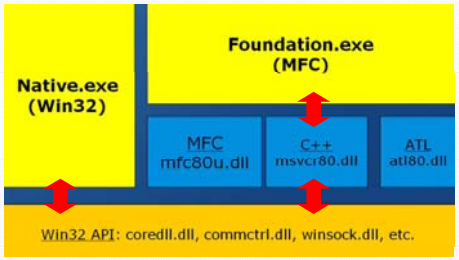
Windows Mobile : *Part de marché*

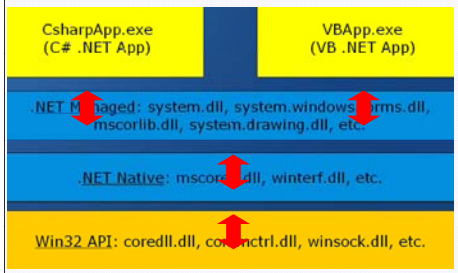


Outils de développement

	Application Programming Interface (API)	Programming Language	Tool
NATIVE CODE	Windows 32-Bit API (Win32)	C C++	Visual Studio 2008
NATIVE CODE	Microsoft Foundation Class (MFC) Library, ver 8.0	C++	Visual Studio 2008
MANAGED CODE	.NET Compact Framework (ver 1.0 or ver 2.0)	C# Visual Basic .NET	Visual Studio 2008

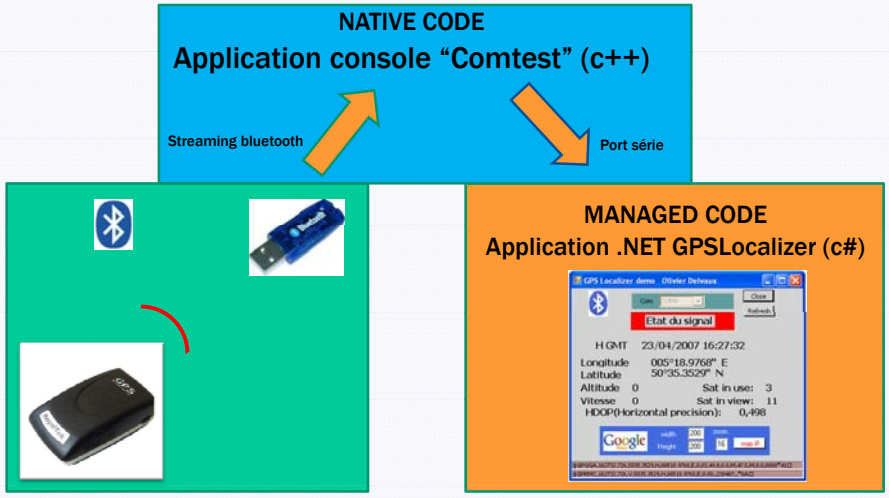
Outils de développement : WIN32 API / MFC / .NET





- **Native code (c/c++)**
  - \+ Pas de Framework (gain en termes de taille et de performance).
  - \+ Run-time = OS
  - \- Portabilité des codes sources.
- **Managed code (c#,vb .net)**
  - \+ Finition esthétique, fonctionnalité de base.
  - \+ Gestion de la mémoire automatique.
  - \+ Portabilité des exécutables.
  - \- Run-time
    - CF 1.0 – 2MB
    - CF 2.0 – 5MB

Exemple : Port série virtuel



Windows Mobile : *Cible ESIAL*

HP iPAQ 614 Business Navigator

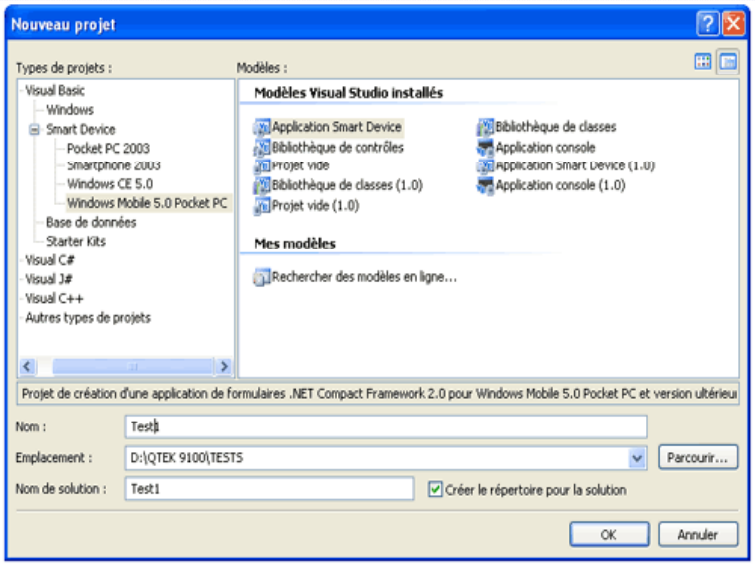
Windows Mobile : *Aide au développement*

http://msdn.microsoft.com/en-us/library/bb158522.aspx

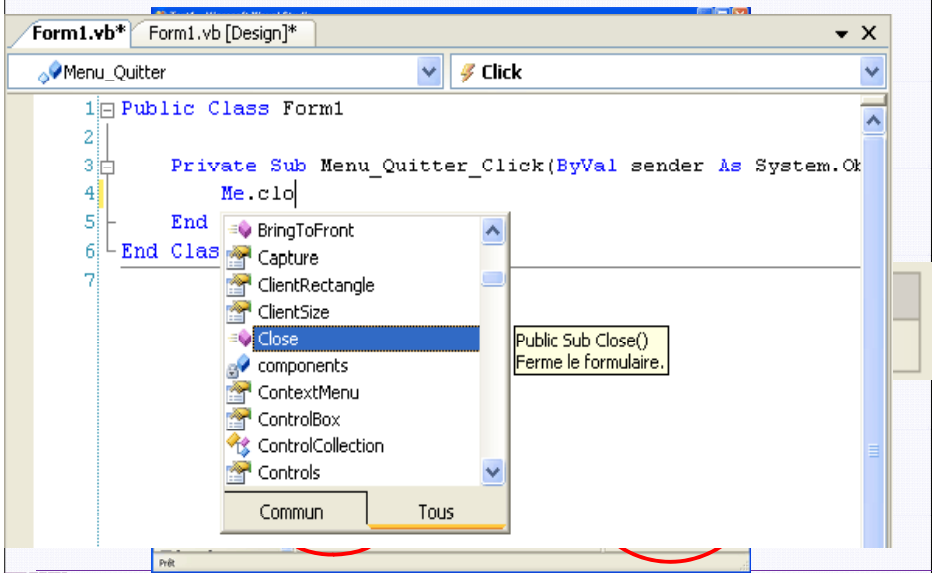
Windows Mobile : *Création du projet*

- Visu
- Win
- Ren

Lance

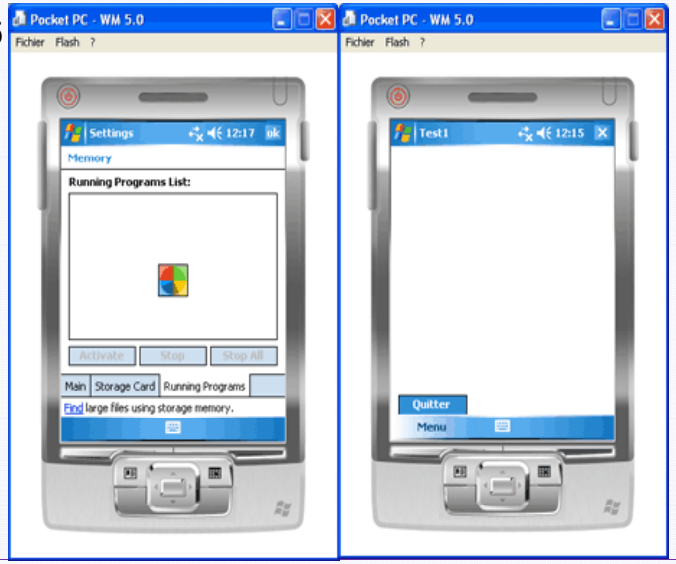


Windows Mobile : *Modification du Menu*



Windows Mobile : *Emulation*

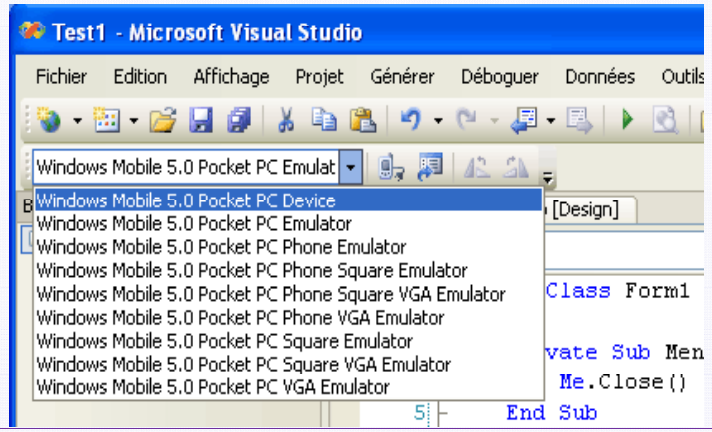
➢ **Touche F5**



Windows Mobile : *Déploiement*

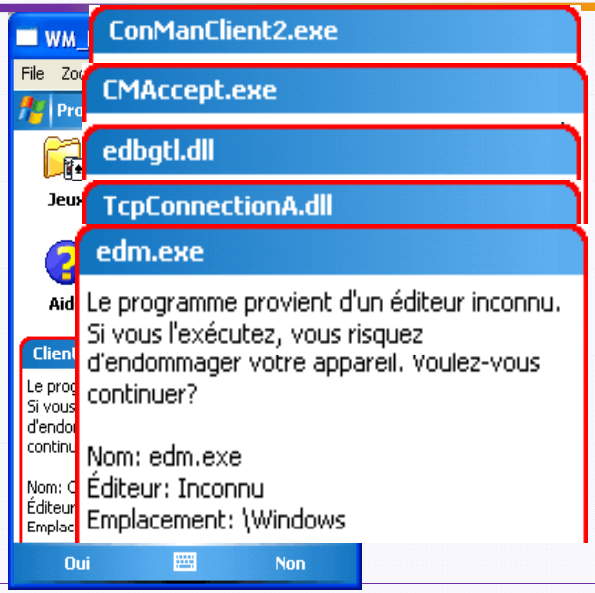
➢ **Raccorder le pocket au PC via l'USB, ActiveSync doit démarrer...**

➢ **Sélectionner "Windows Mobile 5.0 Pocket Device" dans la zone des connecteurs :**

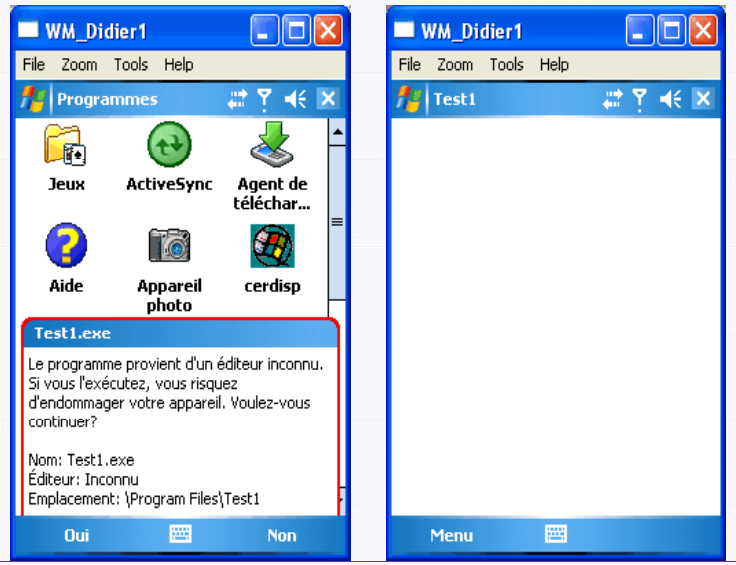


Windows Mobile : *Déploiement*

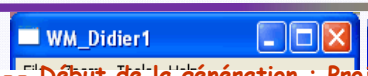
➢ **Touche F5**



Windows Mobile : *Exécution*



Windows Mobile : *Résultats*




```
------ Début de la génération : Projet : Test1, Configuration : Debug Any
CPU ------
Test1 -> D:\QTEK 9100\TESTS\Test1\Test1\bin\Debug\Test1.exe
------ Début du déploiement : Projet : Test1, Configuration : Debug Any
CPU ------
Déploiement de 'C:\Program Files\Microsoft Visual Studio
8\SmartDevices\SDK\CompactFramework\2.0\v2.0\windowsce\wce500\armv4i
\NETCFv2.wm.ARMV4I.cab'
Déploiement de 'D:\QTEK 9100\TESTS\Test1\Test1\bin\Debug\Test1.exe'
Déploiement de 'C:\Program Files\Microsoft Visual Studio
8\SmartDevices\SDK\CompactFramework\2.0\v2.0\windowsce\diagnostics\Sys
tem_SR_fr_wm.cab'
========== Génération : 1 a réussi ou est à jour, 0 a échoué, 0 a été
ignoré ==========
========== Déploiement : 1 a réussi, 0 a échoué, 0 a été ignoré
==========
```