


# XENOMAI 2, temps réel dur pour Linux


P. Gerum / P. Ficheux

RMLL 2006




Formation Linux Airbus

1



## Quelques mots sur OW/OS4I/PF ?

- Créée en sept 2001 dans le sillage de 2 grands groupes: THALES et SCHNEIDER Electric
- Vocation: industrialiser les composants open source
- Domaines: informatique industrielle, infrastructures, portails applicatifs
- Entité spécifique en cours de mise en place: OS4I (Open Source for Industries)
- PF: utilise les logiciels libres depuis 1989, auteur de l'ouvrage « Linux embarqué », CTO OS4I
- PG: auteur de XENOMAI 2, consultant expert OS4I



Formation Linux Airbus

2



## Sommaire

- Tour d'horizon
  - Linux 2.6 standard preemptible
  - Linux 2.6 + patch PREEMPT\_RT
  - Linux + co-noyau / hybrides
    - RTLinux
    - RTAI
    - Xenomai
- Synthèse comparative des solutions



## Linux 2.6 standard preemptible

- Solution clé en main
  - distributions généralistes ou spécialisées
  - pas de patch externe requis
- Disponible sur de nombreuses architectures
  - option de preemption non stable sur toutes
- Bonnes performances globales
  - surcoût tolérable lié à la préemptibilité noyau
- Stabilité et maturité
  - premiers travaux sur la preemption noyau > 5 ans





## Linux 2.6 standard preemptible caractéristiques (2/2)

- Fenêtres de masquage d'interruptions
  - en général courtes, mais non bornées
  - zones masquées non preemptibles
- Tâches preemptibles en mode utilisateur ET noyau
  - verrou de preemption noyau
  - zones verrouillées non preemptibles, parfois longues (ms)
- Gestion de la priorité temps-réel
  - classe d'ordonnancement POSIX SCHED\_FIFO
  - 99 niveaux de priorité temps-réel
  - ordonnanceur O(1)
  - pas de prévention des inversions de priorité



## Linux 2.6 standard préemptible fonctionnement

- Preemption noyau active, sauf :
  - dans les sections critiques SMP (spinlock)
  - sur contexte d'interruption
- *Variantes de préemption*
  - *PREEMPT\_VOLUNTARY*, reposant sur des points de préemption explicitement insérés dans le code
    - *couverture plus faible, mais moins d'artefacts de synchronisation*
  - *PREEMPT\_DESKTOP*, reposant sur des points de préemption implicitement insérés en sortie des verrous SMP.
    - *couverture plus large, mais plus d'artefacts de synchronisation*





## Linux 2.6 standard preemptible utilisation et performances

- Pas de changements majeurs / version non-préemptible
  - écriture de code noyau
  - écriture de code applicatif
- Légère amélioration de la latence moyenne
- Pas de réduction significative de la latence maximale
- Moins d'amplitude dans les pics de latence
- Système non temps-réel au sens strict



## Linux 2.6 PREEMPT\_RT

- Patch externe récent sur base 2.6.x (-rt)  
[people.redhat.com/mingo/realtime-preempt](http://people.redhat.com/mingo/realtime-preempt)
- Disponible sur un plus faible nombre d'architectures
  - x86
  - moins testé sur x86\_64
  - encore instable sur ppc, arm, mips, ia64
- En développement, testé essentiellement sur x86





## Linux 2.6 PREEMPT\_RT utilisation et performances

- Changements significatifs du code noyau
  - sémantique de verrouillage des sections critiques
  - inspection nécessaire de la totalité du code (drivers compris)
- Pas d'impact sur le code utilisateur
  - modèle programmatique totalement standard
- Impact du coût de la préemption significatif (20-100%)
  - selon le type de charge
  - selon la puissance CPU disponible
- Temps de latence moyen (x86 - PIII) < 100 microsecondes



Temps de latence maximum encore indéfini



## Linux 2.6 PREEMPT\_RT utilisation et performances

- Temps de latence moyen (x86 - PIII) < 100 microsecondes
- Temps de latence maximum encore indéfini
  - tend globalement vers moins de 100 us (x86 - PIII)
  - pics pathologiques encore constatés vers 400 us (SMP/x86)





## Linux + co-noyau / hybrides caractéristiques

- Ajout d'un co-noyau pour la gestion temps-réel
  - sous-système temps-réel intégré dans un module noyau
  - patch de virtualisation des interruptions
- Différents modèles programmatiques
  - Kernel uniquement (RTLinux/GPL, [www.rtlinux-gpl.org](http://www.rtlinux-gpl.org))
  - Kernel & user-space, semi-intégration Linux (RTAI, [www.rtai.org](http://www.rtai.org))
  - Kernel & user-space, intégration Linux complète (Xenomai, [www.xenomai.org](http://www.xenomai.org))



## Linux + co-noyau / hybrides fonctionnement

- Séparation entre le composant temps-réel et Linux
  - ordonnanceur temps-réel spécifique
  - pas de dépendance sur les sections critiques Linux
- Virtualisation de la gestion d'interruptions Linux
  - routage prioritaire des IRQs vers le co-noyau
  - activation de l'ordonnanceur temps-réel
- Linux comme tâche *idle* du co-noyau





## Linux + co-noyau / hybrides utilisation et performances

- Changement minimal sur le noyau Linux
  - patch de virtualisation d'interruptions (ADEOS)
  - pas d'impact sur l'écriture de code noyau classique
- Impact sur l'écriture de code temps-réel
  - utilisation d'APIs fournies par le co-noyau
  - notion de domaine d'exécution (temps-réel / normal)
- Garanties temps-réel fortes
  - ordonnanceur spécifique indépendant
  - sous-système temps-réel bien délimité
- Latence maximale < 20 microsecondes



## Xenomai

- Coeur de RTOS « abstrait »
  - émulation d'APIs propriétaires (VxWorks, pSOS, VRTX, uITRON)
  - personnalité POSIX
  - personnalité native
- *Architectures supportées*
  - *arm, blackfin, x86, ia64, powerpc32, powerpc64*
  - *simulateur événementiel*
- Modèle programmatique
  - espace kernel
  - espace utilisateur, fortement intégré au modèle standard





## Synthèse comparative des solutions

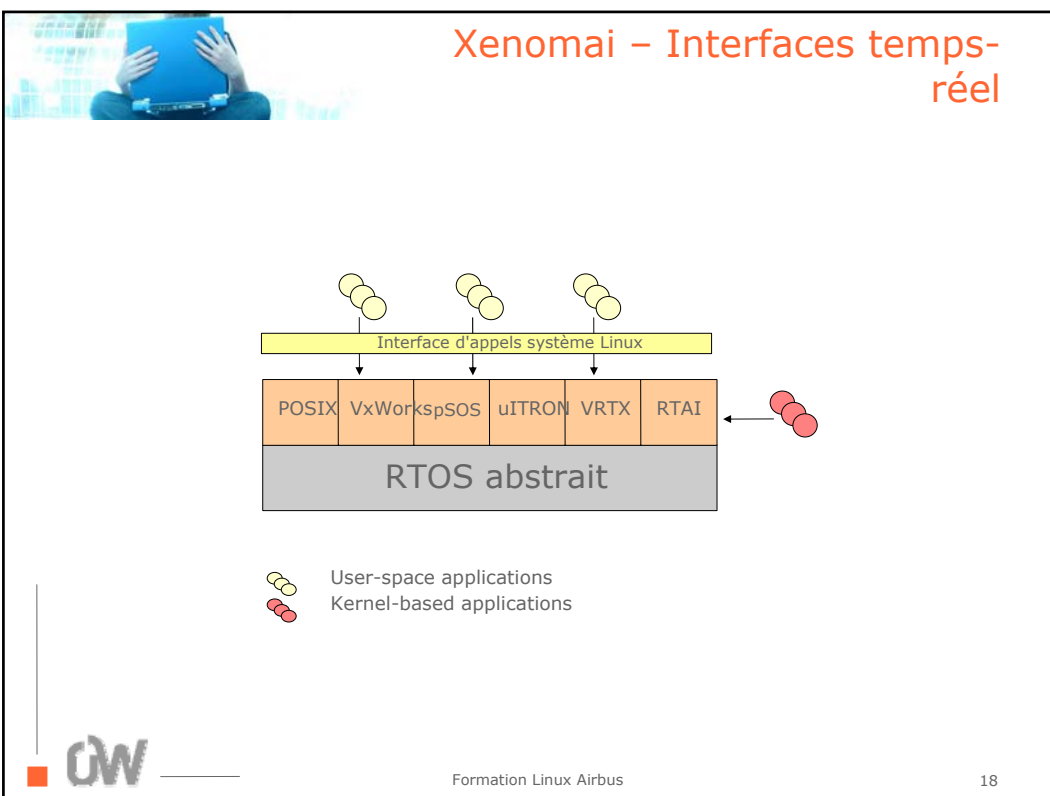
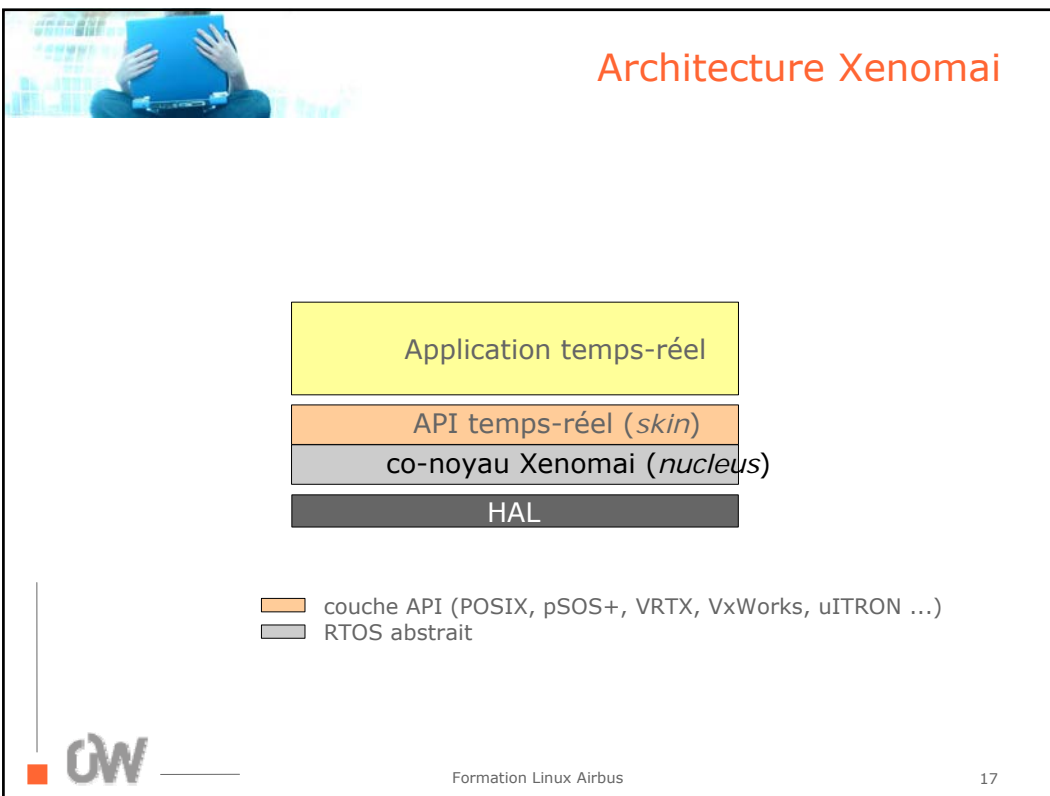
	Linux préemptible	Linux PREEMPT_RT	RTLinux	RTAI	Xenomai	
Type	preemption native	preemption native	co-noyau	co-noyau	co-noyau intégré	
Architectures maintenues	toutes	arm, ppc, i386, x86_64, mips *	<i>i386</i>	<i>i386, arm</i>	<i>arm, blackfin, i386, ia64, ppc 32/64</i>	
API	noyau, POSIX standard	noyau, POSIX standard	POSIX PSE51 espace noyau	spécifique noyau ou user	multiples, communes noyau/user	
Latence moyenne	HZ(1-10 ms)	10 -20 us	< 5 us	< 5 us	< 5us	
Latence max	$\infty$	70-400? us	< 20 us	< 20 us	< 20 us	
Etat	stable	développement actif	stable, maintenance éteinte	stable, évolutions chaotiques	stable, développement actif	

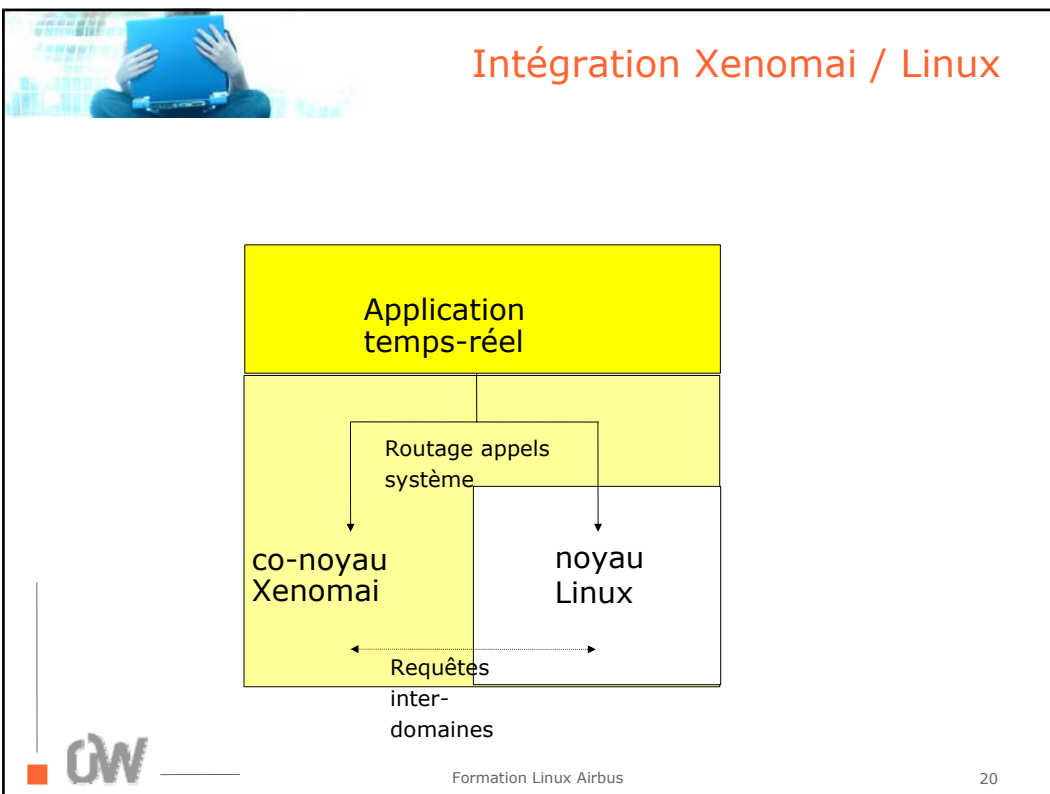
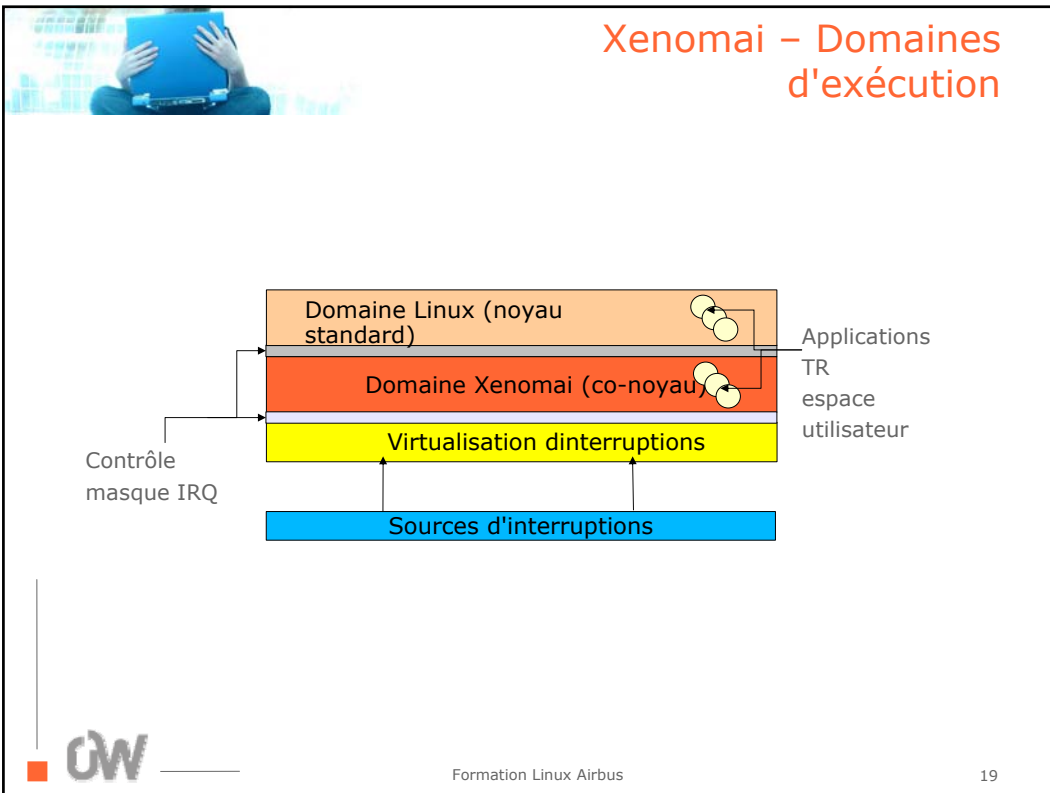


## Focus sur Xenomai

- Xenomai
  - Linux 2.4 et 2.6 (x86, ppc32, ppc64, ia64, blackfin, arm)
  - Support temps-réel faible latence espace utilisateur
  - Technologie type *nanokernel*
  - Emulation de RTOS traditionnels
- v2.2-rc2, publié en Avril 2006
  - v2.2 prévue en Mai 2006









## Approches orthogonales

- PREEMPT\_RT (préemption native)
  - Accès aux pilotes Linux standard en mode temps-réel
  - Base noyau preemptible la plus large
  - Modèle programmatique totalement natif
- Xenomai (préemption externe)
  - Surcharge CPU minimale de l'apport temps-réel
  - Déterminisme fort et latences faibles
  - Isolation du code Linux potentiellement non-déterministe
  - Modèle programmatique quasiment natif



## Support POSIX / Xenomai

- Périmètre POSIX avec garantie temps-réel
  - Xenomai : Sous-ensemble 1003.1b
  - PREEMPT\_RT : 1003.1b standard (en voie d'adaptation)
- Ré-implémentation POSIX (captage)
  - bibliothèque d'interface POSIX/Xenomai
  - exécution standard des appels non-ré-implémentés
- Routage dynamique d'appels système
  - Interposition sur tous les appels système
  - Gestion des migrations entre domaines



## Migration de domaine / Xenomai

Appel POSIX standard

**Domaine Linux**

**Domaine Xenomai**

Service POSIX Xenomai

→ Migration d'appel système  
 — Exécution tâche temps-réel

Formation Linux Airbus

23

## POSIX renforcé / Xenomai

Interface standard d'appels système

Services noyau Linux

API  
POSIX  
co-noyau  
Xenomai

← Applications espace noyau

Application espace utilisateur  
 Applications espace noyau

Formation Linux Airbus

24



## Conclusion

- Spectre applicatif temps-réel
  - pas de solution Linux « tout-en-un »
- Exigences variables
  - d'une application à l'autre
  - d'une tâche à l'autre dans une même application
  - besoin d'accès aux services Linux standard
- Modèle programmatique classique
  - intégration transparente du support temps-réel
  - protection mémoire
  - accès aux outils standard (ex. GDB)

