Introduction à l'utilisation de Linux dans les systèmes embarqués et temps réel

Eurogiciel

Agence de Rennes
22 rue Rigourdière - 35510 Cesson Sévigné

26 janvier 2011

4 Processus d'amorçage
- Présentation
- Améliorations au niveau du noyau
- Analyse de la séquence de démarrage du noyau
- Analyse de la séquence de démarrage des processus

5 Capacités temps réel
- Description du temps réel
- Classification
- Notions élémentaires
- Charge CPU et ordonnancement
- Disciplines d'ordonnancement sous Linux
- Modes de préemption
- Technique à co-noyau

6 Aspects juridiques
- Logiciels libres
- Logiciels "open source"
- Notion de Copyleft
- Différentes licences
- Validité juridique de la licence GPL
- Exemples concrets

7 Conclusion

Caractéristiques

- Combinaison d'électronique et d'informatique, de matériel et de logiciel
- Exécution d'une fonction spécifique et dédiée
- Système enfoui au cœur d'un équipement
- Difficultés d'accès
- Ressources restreintes
- Contraintes techniques et environnementales
- Contrainte de durée de vie et de maintenabilité

Principaux champs d'application

- Calcul généraliste
- Contrôle de systèmes asservis
- Traitement de signal
- Réseaux et télécommunications

Première définition

"*Les logiciels embarqués (ou enfouis) sont destinés au pilotage de systèmes embarqués informatiques et électroniques autonomes ayant de très fortes interactions avec leur environnement.*"

Autre définition

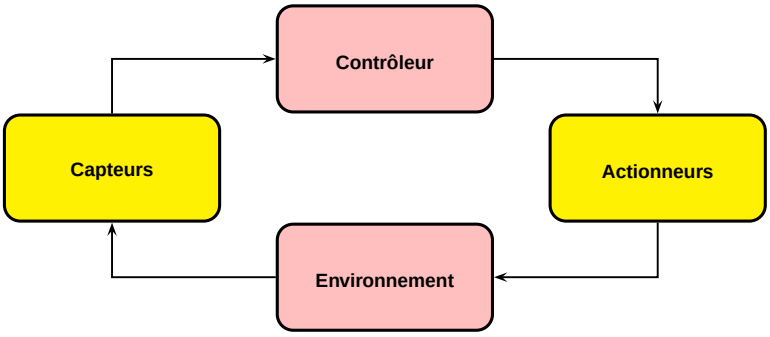
"*Un système embarqué est un équipement qui réalise une fonction dédiée ou qui est conçu pour une utilisation avec une application logicielle embarquée spécifique.*"

Éléments constitutifs courants

- Microprocesseur, microcontrôleur ou DSP
- Coprocesseurs
- Périphériques d'entrées-sorties
- Convertisseurs A/N et N/A
- Clavier et IHM minimalistes
- FPGA ou ASIC

Interactions avec l'environnement

- Lecture de capteurs
- Commande d'actionneurs



2 Présentation de Linux

- Notions élémentaires
- Modes d'exécution
- Projet GNU
- Solution pour l'embarqué

Différents supports d'exécution

- Noyau
- Exécutif
- Système d'exploitaion

Noyau

- Structure minimaliste
- Éxécution performante
- Ordonnancement des tâches
- Communication et synchronisation entre les tâches
- Gestion de l'horloge "temps réel"
- Gestion des interruptions matérielles
- Liaison directe avec le matériel

Exécutif

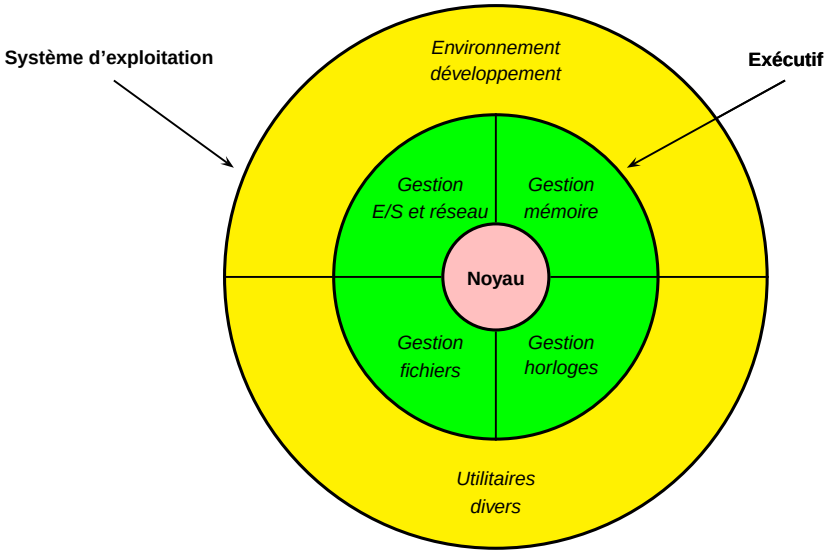
- Exploitation d'un noyau
- Gestion de la mémoire
- Gestion des entrées-sorties et des accès réseau
- Gestion de fichiers et d'horloges
- Accès direct aux primitives du noyau

Système d'exploitation

- Couche logicielle basée sur un exécutif
- Intégration éventuelle d'un environnement de développement
- Mise à disposition de plusieurs utilitaires
- Accès sécurisé aux primitives du noyau

Structure d'un système d'exploitation

- Empilement de couches logicielles



Distinction Linux et GNU/Linux

- Linux est un noyau “augmenté”
- GNU/Linux est un système d'exploitation de type Unix utilisant Linux
- Un système Linux embarqué est une forme d'exécutif

Historique de Linux (1/2)

- 1991 : Écriture du noyau Linux à partir de zéro (from scratch) en 6 mois par Linus TORVALDS dans sa chambre à l'université d'Helsinki pour son PC 80386
- 1991 : Linus TORVALDS distribue son noyau sur Internet et des programmeurs du monde entier le rejoignent pour contribuer au code et aux tests
- 1992 : Plus de 100 développeurs travaillent sur le noyau

Historique de Linux (2/2)

- 1992 : Distribution de Linux sous licence GNU GPL
- 1994 : Sortie de Linux 1.0
- 1994 : Création de la société Red Hat par Bob YOUNG et Marc EWING créant ainsi un nouveau modèle économique basé sur une technologie "open source"
- 1995 : Utilisation progessive de GNU/Linux et des logiciels libres pour les serveurs Internet
- 2001 : Investissement par IBM d'un milliard de dollars dans Linux
- 2002 : Adoption massive de GNU/Linux dans de nombreux secteurs de l'industrie
- 2010 : Environ 1200 personnes contribuent au développement du noyau

Mode basique de numérotation des versions selon trois champs

- Indice majeur
- Indice mineur
- Indice de révision

Quatrième champ depuis la version 2.6.11 (mars 2005)

- Incrémentation en fonction de la correction de bogues
- Pas de nouvelle fonctionnalité

Versions actuelles

- 2.6.36 publiée le 20 octobre 2010
- 2.6.35.8 publiée le 29 octobre 2010

Caractéristiques (1/2)

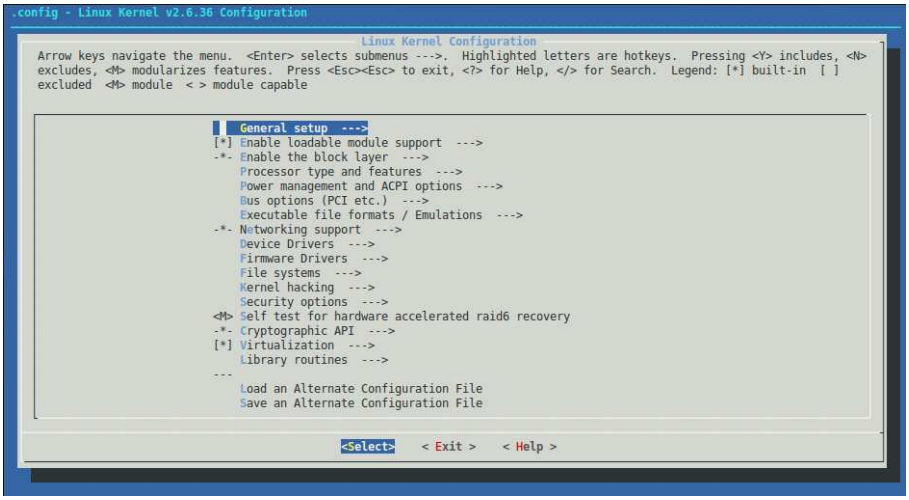
- Structure monolithique
- Pilotes pouvant être compilés comme modules et chargés ou déchargés “à chaud”
- Exécution des pilotes de périphériques en mode/espace noyau
- Exécution de la plupart des interfaces graphiques en mode/espace utilisateur
- Support des plateformes multiprocesseurs (SMP) et de machines de toutes tailles (scalabilité)
- Large portabilité et support matériel étendu

Caractéristiques (2/2)

- Développement essentiellement en langage C avec une légère couche en assembleur
- Structure 32 bits minimale des processeurs avec ou sans MMU (Memory Management Unit)
- Mécanisme de préemption (2.6.x)
- Conformité aux standards et interopérabilité POSIX
- Stabilité et fiabilité

Configuration et compilation d'un noyau Linux

- Placement des sources dans `\usr\src\`
- Configuration (ex. : `make menuconfig`)
- Compilation et installation par la séquence `make && make modules_install && make install`



Exemples de mise en œuvre

- Cluster Jaguar (1.75 PFlops)
- PicoTux

Partitionnement de l'espace mémoire

- Deux régions distinctes

Espace utilisateur

- Mode restreint assurant la protection des processus
- Impossibilité pour un processus d'accéder à une zone mémoire d'un autre processus
- Espace d'exécution des programmes applicatifs
- Programmation principalement à l'aide des fonctions de la bibliothèque standard

Espace noyau

- Utilisation large du matériel et notamment de la mémoire
- Accès aux services du noyau
- Espace d'exécution de la plupart des pilotes de périphériques et des applications temps réel
- Programmation à l'aide de l'API noyau

Interactions avec le noyau depuis l'espace utilisateur

- Utilisation des appels système depuis l'espace utilisateur

Description

- Acronyme récursif signifiant "GNU is Not Unix"
- Projet initié en 1983 par Richard STALLMAN suite à un désaccord avec le principe de la licence BSD (Berkeley Software Distribution)
- Projet de système d'exploitation composé exclusivement de logiciels libres et basé sur les concepts d'Unix
- Création en 1985 de la FSF (Free Software Foundation) par Richard STALLMAN

Adoption du noyau Linux

- Prévision du développement d'un noyau (Hurd)
- Disponibilité des logiciels du projet GNU par la publication du noyau Linux (sous licence GPL en 1992)
- Ensemble des distributions Linux marquées par l'empreinte du projet GNU (licences, etc.) et appellation GNU/Linux défendue par Richard STALLMAN

Nombreux outils

- Utilitaires de base : Bash, Grep, Tar, Cpio, Emacs, Wget, ...
- Développement : GCC, GDB, Make, Binutils, Glibc, ...
- Outils et environnements graphiques : Gimp, GTK+, GNOME, ...
- Etc.

Contraintes des solutions propriétaires

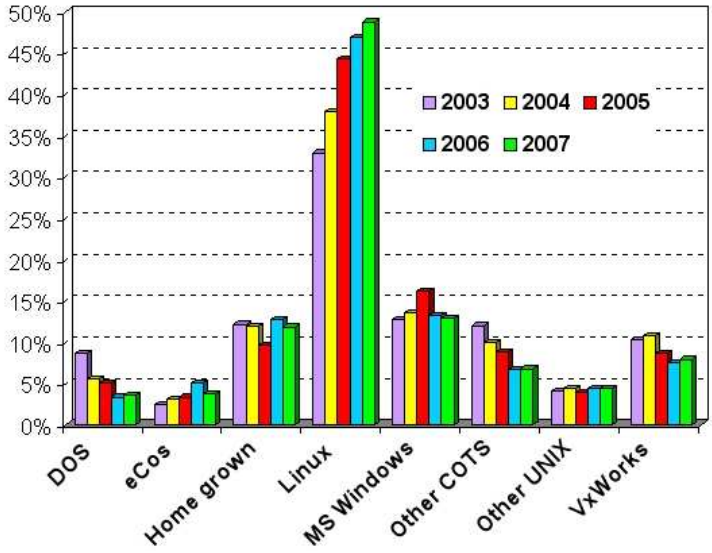
- Pas ou peu de compatibilité entre les solutions
- Outils de développement coûteux et figés
- Dépendance d'un éditeur et redevances souvent élevées
- Problèmes de pérennité
- Portabilité réduite

Intérêt de Linux

- Fort développement de Linux et des logiciels libres
- Alternative très sérieuse du prototypage au produit fini
- Support élargi et très large documentation
- Code ouvert et coûts moindres

Positionnement sur le marché

- Solution la plus utilisée



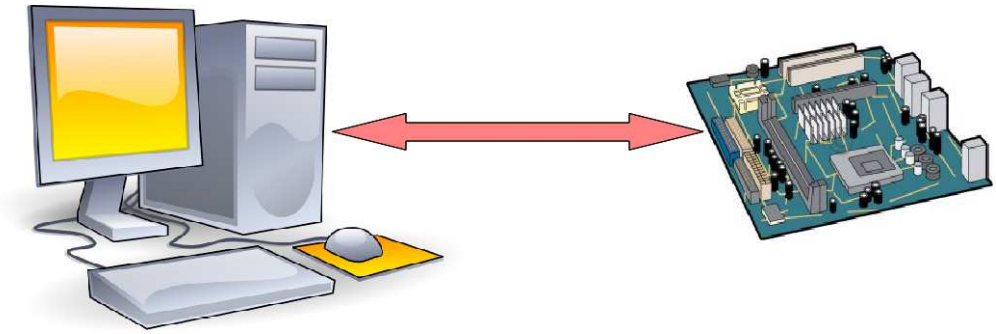
(Source : www.linuxfordevices.com)

3 Outils pour l'embarqué

- Plateforme de développement
- Réduction de taille et d'empreinte mémoire
- Choix du chargeur d'amorçage
- Production d'une distribution
- Outils de débogage, de profilage et d'émulation

Atelier

- Architectures différentes entre la machine hôte et la cible
- Communication via une liaison Ethernet, RS232, USB, etc.
- Chargement à l'aide d'une interface JTAG (IEEE 1149.1)



Machine hôte

- Éditeur
- Chaîne de développement croisé (compilateur, éditeur de liens, débogueur, outils de profilage, etc.)
- Simulateur et émulateur

Cible

- Chargeur d'amorçage (bootloader)
- Noyau
- Système de fichiers racine (rootFS)

Bibliothèque des fonctions standards

- GLibc peu adaptée aux systèmes embarqués

Alternatives disponibles

Bibliothèque	Taille	Fonctionnalités et compatibilité
EGlibc	-	+ +
μClibc	+	+
DietLibc	+ +	-
KLibc	+ + +	- -

Commandes utilisateur

- Coreutils peu adapté aux systèmes embarqués

Alternatives disponibles

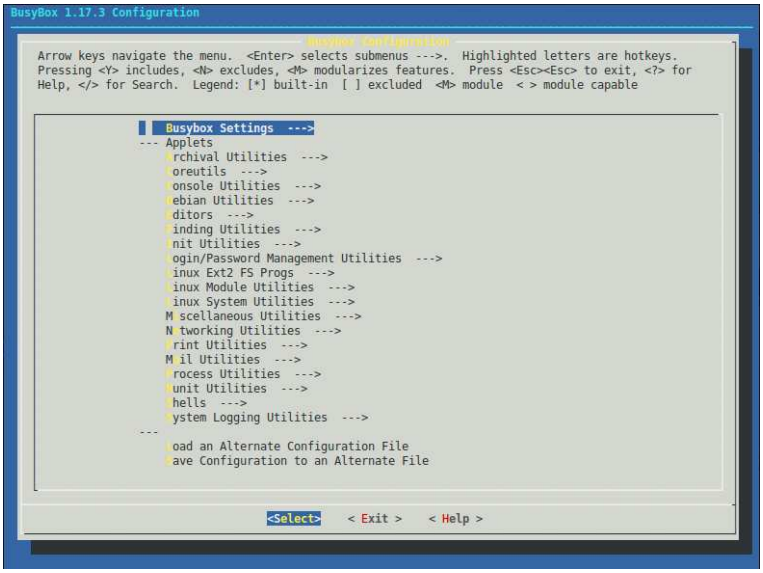
Bibliothèque	Taille	Fonctionnalités et compatibilité
Busybox	+ +	-
Embutils	+	- -

Busybox (1/2)

- "Couteau suisse" pour Linux embarqué
- Agglomération des commandes Unix essentielles en un seul exécutable
- Utilisation avec les bibliothèques Glibc et μClibc
- Possibilité de réduction en fonction des besoins
- Mise à disposition d'outils complémentaires (SSH, SMTP, NTP, etc.)

Busybox (2/2)

- Configuration via un menu graphique



Rôle essentiel

- Lancement du noyau
- Spécification du système de fichiers racine

Solutions disponibles

- U-Boot
- ARMBoot
- RedBoot
- OpenBIOS
- FreeBIOS
- PXELinux
- EtherBoot
- Etc.

Solutions disponibles

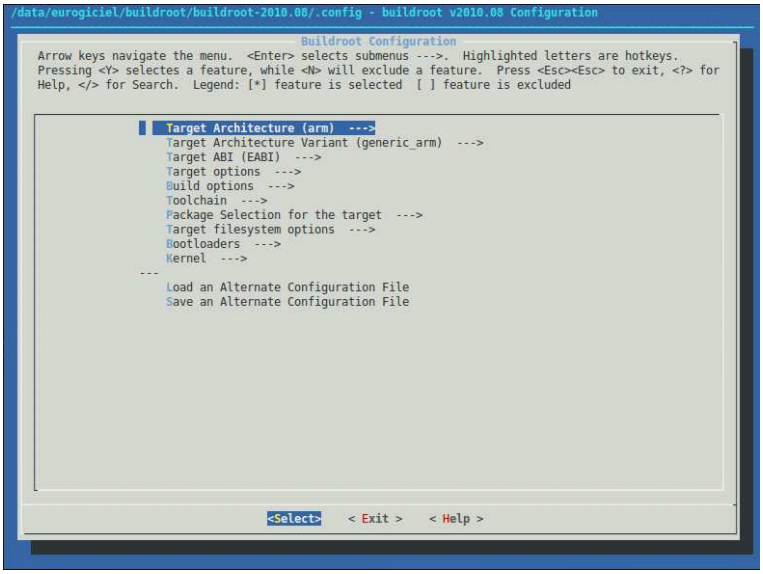
- Buildroot
- OpenEmbedded
- ElinOS
- Timesys
- MontaVista
- OpenWRT
- LTIB
- PTXdist
- Linux From Scratch
- Etc.

Buildroot (1/2)

- Génération complète d'un système embarqué
  - Chaîne de compilation croisée
  - Images du chargeur d'amorçage, du noyau et du système de fichiers racine
- Sélection des outils et de la bibliothèque standard
  - Busybox
  - Glibc, μClibc, EGlibc
  - Etc.
- Choix du système de fichiers
  - JFFS2
  - UbiFS
  - Ext2
  - CramFS
  - Etc.
- Possibilité d'extension
  - Intégration de paquets et automatisation de tâches

Buildroot (2/2)

- Configuration via un menu graphique



GDB

- Débogeur du projet GNU
- Variantes pour le débogage du noyau (KGDB) et sur cible (GDBserver)
- Disponibilité d'une interface graphique (DDD)
- Compilation par GCC avec l'option "-g"

Strace

- Outil de trace des appels système réalisés par un programme

Dtrace

- Outil de détection de problèmes en temps réel aux niveaux noyau et applicatif

Gprof

- Profileur de code
- Compilation par GCC avec l'option "-pg"

SystemTAP

- Outil d'analyse de l'exécution d'un système GNU/Linux
- Exploitation d'un langage de script
- Compilation d'un script après validation et chargement comme module du noyau
- Mesure de performances des appels de fonctions
- Fonctionnement en modes utilisateur et noyau
- Disponibilité d'une interface graphique (SystemTAP GUI)

Valgrind

- Boîte à outils de mise au point, de déverminage et de profilage
- Analyse dynamique de programmes en cours d'exécution
- Support des plateformes x86, PPC et ARM
- Disponibilité de plusieurs interfaces graphiques
- Memcheck : Détection des problèmes mémoire dans les programmes C/C++
- Cachegrind : Optimisation du cache
- Callgrind : Extension à Cachegrind fournissant des informations sur les graphes d'appels
- Massif : Optimisation de la pile
- Helgrind : Détection des problèmes de concurrence dans les programmes multi-tâches

QEMU

- Utilitaire d'émulation
- Bon niveau de performances
- Support de nombreuses architectures
- Ajout relativement aisé de nouveaux matériels
- Connexion avec GDB entre la machine hôte et la cible émulée

KVM

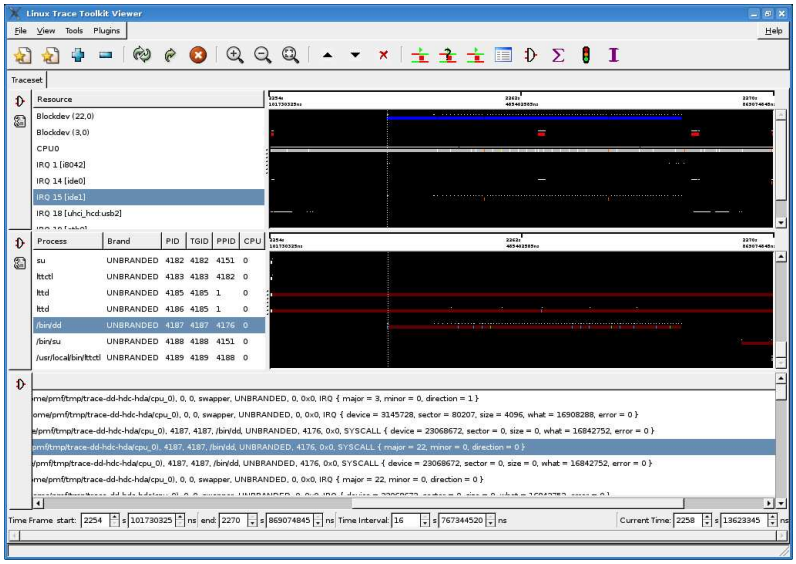
- "Fork" de QEMU
- Machine virtuelle fonctionnant sur une architecture x86 disposant des technologies Intel VT et AMD SVM
- Intégration au noyau depuis la version 2.6.20

LTTng (1/2)

- Utilitaire de trace, d'analyse de performances et de débogage au niveau noyau
- Logiciel intrusif dans le noyau (patch)
- Contrainte de lien avec la version du noyau
- Supervision des appels systèmes, des interruptions, des allocations mémoire, de l'ordonnanceur, etc.
- Contrôle des opérations de trace par LTT-Control
- Portage de LTTng pour les exécutions en mode utilisateur (UST)

LTTng (2/2)

- Utilitaire LTTV pour la visualisation graphique



4 Processus d'amorçage
- Présentation
- Améliorations au niveau du noyau
- Analyse de la séquence de démarrage du noyau
- Analyse de la séquence de démarrage des processus

Constat

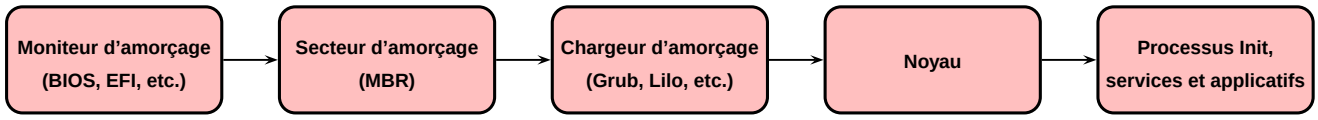
- Démarrage d'un système Linux "standard" en plusieurs dizaines de secondes
- Démarrage d'un système Linux optimisé en moins d'une seconde

Deux niveaux d'optimisation complémentaires

- Amélioration de l'initialisation du noyau
- Gestion adaptée du démarrage des processus

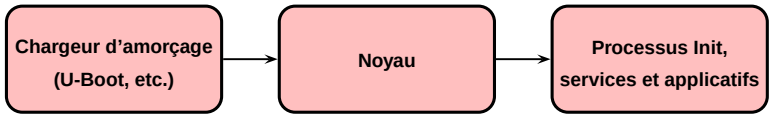
Séquence de démarrage standard

- Exécution du moniteur d'amorçage
- Lecture du secteur d'amorçage
- Lancement du chargeur d'amorçage
- Démarrage du noyau
- Exécution du processus Init



Séquence de démarrage "embarqué"

- Lancement du chargeur d'amorçage
- Démarrage du noyau
- Exécution du processus Init



Horadatage des messages

- Configuration du noyau avec l'option `CONFIG_PRINTK_TIMES`
- Activation par passage du paramètre `printk.time=1` au noyau
- Activation/désactivation dynamique par la ligne de commande avec `echo [1/0] > /sys/module/printk/parameters/printk_time`

```
[49357.810982] ata1.00: ACPI cmd ef/03:0c:00:00:00:a0 (SET FEATURES) filtered out
[49357.810986] ata1.00: ACPI cmd ef/03:22:00:00:00:a0 (SET FEATURES) filtered out
[49357.850587] ata1.00: configured for MWDMA2
[49357.870062] usb 5-2: reset low speed USB device using uhci_hcd and address 2
[49358.183111] PM: resume of drv:usb dev:5-2 complete after 844.694 msecs
[49358.183118] PM: resume of drv:usbhid dev:5-2:1.0 complete after 844.686 msecs
[49358.183124] PM: resume of drv: dev:ep_81 complete after 581.138 msecs
[49358.370035] [drm:atom_op_jump] *ERROR* atombios stuck in loop for more than 1sec aborting
[49358.370039] [drm:atom_execute_table_locked] *ERROR* atombios stuck executing E952 (len 86, WS 4, PS 0) @ 0xE985
[49359.190062] PM: resume of drv:radeon dev:0000:01:00.0 complete after 1857.197 msecs
[49359.960066] ata3: SATA link up 3.0 Gbps (SStatus 123 SControl 300)
[49359.964358] ata3.00: ACPI cmd f5/00:00:00:00:00:a0 (SECURITY FREEZE LOCK) filtered out
[49359.964361] ata3.00: ACPI cmd b1/c1:00:00:00:00:a0 (DEVICE CONFIGURATION OVERLAY) filtered out
[49359.964453] ata3.00: ACPI cmd c6/00:10:00:00:00:a0 (SET MULTIPLE MODE) succeeded
[49359.964456] ata3.00: ACPI cmd ef/10:03:00:00:00:a0 (SET FEATURES) filtered out
[49359.974610] ata3.00: ACPI cmd f5/00:00:00:00:00:a0 (SECURITY FREEZE LOCK) filtered out
[49359.974613] ata3.00: ACPI cmd b1/c1:00:00:00:00:a0 (DEVICE CONFIGURATION OVERLAY) filtered out
[49359.974708] ata3.00: ACPI cmd c6/00:10:00:00:00:a0 (SET MULTIPLE MODE) succeeded
[49359.974711] ata3.00: ACPI cmd ef/10:03:00:00:00:a0 (SET FEATURES) filtered out
[49359.980498] ata3.00: configured for UDMA/133
[49360.011734] ata3.00: configured for UDMA/133
[49360.011737] ata3: EH complete
[49360.022023] PM: resume of drv:sd dev:2:0:0:0 complete after 2683.574 msecs
```


Techniques de base

- Suppression des pilotes et fonctions inutiles
- Suppression des fonctions de détection automatique
- Choix d'un système de fichiers performant et adapté
- Configuration en module de tout pilote inutilisé au moment du démarrage
- Recopie du noyau de la mémoire flash vers la mémoire RAM par DMA

Chargement différé de modules

- Patch du noyau
- Changement de la fonction `module_init()` en `deferred_module_init()`
- Activation ultérieure des modules concernés par la ligne de commande `echo 1 > /proc/deferred_initcalls`

Suppression des messages vers la console

- Écriture vers la console très pénalisante
- Affichage vers la console pas toujours nécesaire
- Gains souvent importants (30 à 50%)
- Désactivation des appels à la fonction `printk()` par passage du paramètre `quiet` au noyau

Calibration des délais

- Utilisation des fonctions `mdelay()` et `udelay()`
- Prise en compte de la granularité d'horloge de 1 ms (1000 Hz) ou 10 ms (100 Hz)
- Calcul de la variable `lpj` sur 25 quantums de temps (“jiffies”)
- Délai nécessaire au calcul de 25 ms ou 250 ms
- Valeur constante de `lpj` sur un système donné
- Récupération de la valeur de `lpj` dans les messages par passage du paramètre `loglevel=8` au noyau
- Passage du paramètre `lpj=xxxxxx` au noyau

Synchronisation de l'horloge système

- Exactitude de l'heure système pas toujours nécessaire
- Mise à jour de l'horloge système à chaque démarrage à partir de l'horloge temps réel (RTC)
- Délai en fonction de l'architecture et de la résolution de la RTC
- Suppression de la synchronisation par un patch
- Mise en œuvre possible d'une synchronisation via une horloge externe (ex. : NTP)

Accélération du redémarrage

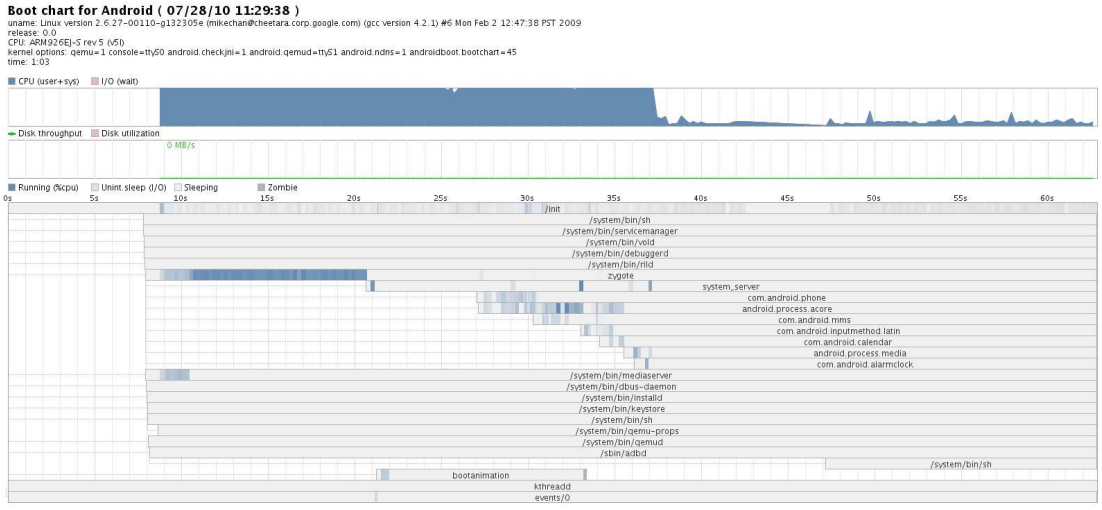
- Utilisation de l'outil Kexec
- Lancement d'une seconde exécution du noyau
- Pas d'exécution de la phase d'amorçage
- Utilisation courante lors de la mise à jour d'un micrologiciel

Utilisation de la trace

- Configuration du noyau avec l'option `CONFIG_BOOT_TRACER`
- Activation par passage des paramètres `initcall_debug` et `printk.time=1` au noyau
- Génération d'images au format SVG par la ligne de commande avec `dmesg | perl scripts/bootgraph.pl > output.svg`
- Disponibilité depuis la version 2.6.28

Utilisation de l'outil Bootchart

- Éxécution d'un script lors de la phase d'initialisation
- Lecture des informations dans le système de fichiers `/proc`
- Production d'un graphique



5 Capacités temps réel
- Description du temps réel
- Classification
- Notions élémentaires
- Charge CPU et ordonnancement
- Disciplines d'ordonnancement sous Linux
- Modes de préemption
- Technique à co-noyau

Caractéristiques

- Respect de contraintes temporelles
- Correction d'un traitement conjointement basée sur l'exactitude logique et le délai d'obtention du résultat
- Pas nécessairement rapide, mais réactif
- Prévisible, fiable, déterministe logiquement et temporellement
- Contexte multitâche

Première définition

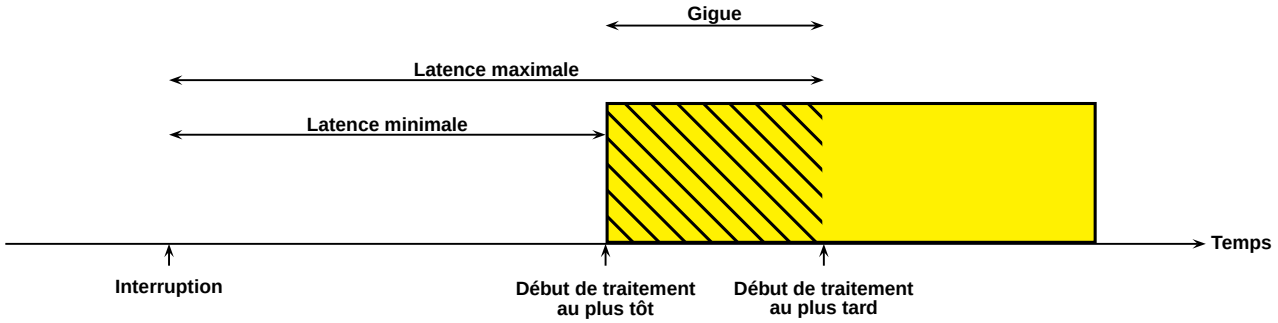
"*Un système est dit temps réel lorsque la correction de son comportement ne dépend pas uniquement des résultats logiques de son exécution, mais aussi des instants physiques auxquels ces résultats sont obtenus.*"

Autre définition

"*Un système informatique temps réel peut être défini comme un système qui contrôle un environnement en recevant des données, en les traitant et en produisant une action de façon suffisamment rapide pour intervenir sur le comportement de l'environnement à ce moment-là.*"

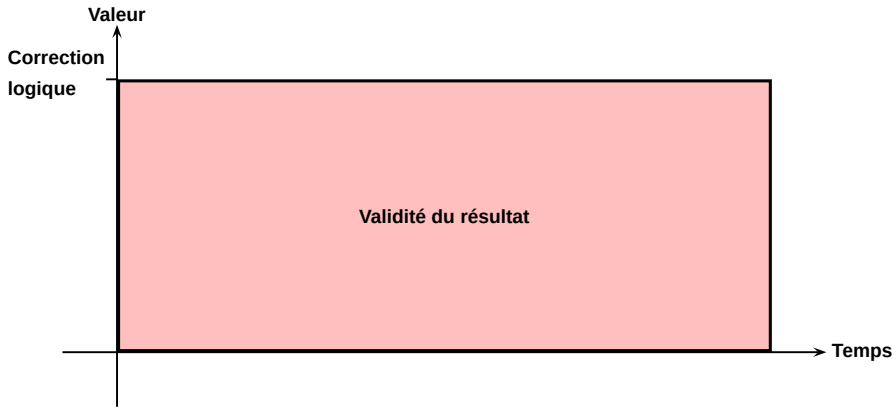
Objectifs visés

- Respect d'échéances lâches ou fermes pour l'exécution des tâches
- Minimisation de la latence
- Minimisation de la gigue



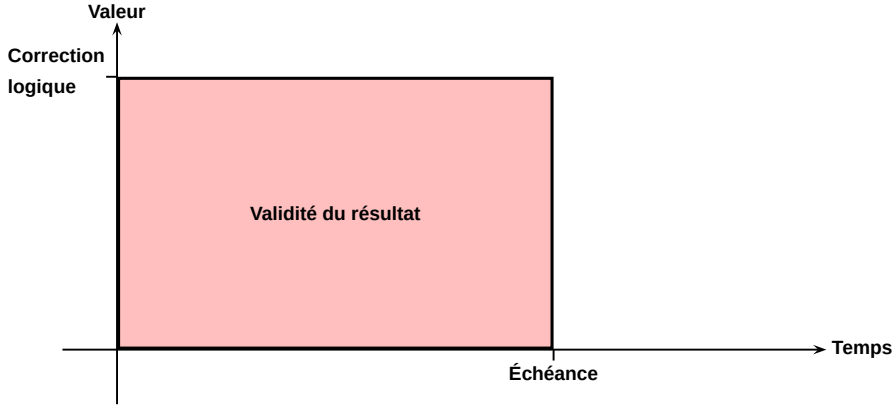
Temps partagé

- Validité d'un résultat uniquement basée sur sa correction logique
- Pas de considération de la date d'obtention du résultat



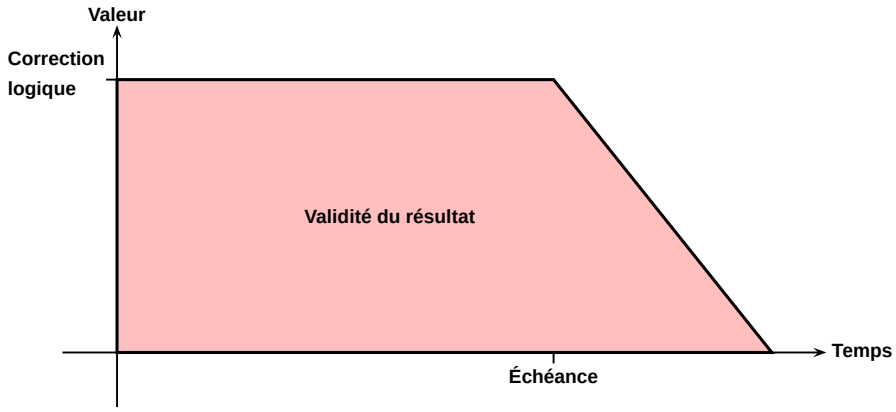
Temps réel dur

- Criticité du résultat
- Tout retard d'obtention du résultat génère une faute temporelle



Temps réel mou

- Tolérance de certains dépassement d'échéances
- Validité partielle du résultat au-delà de l'échéance



Description d'une tâche

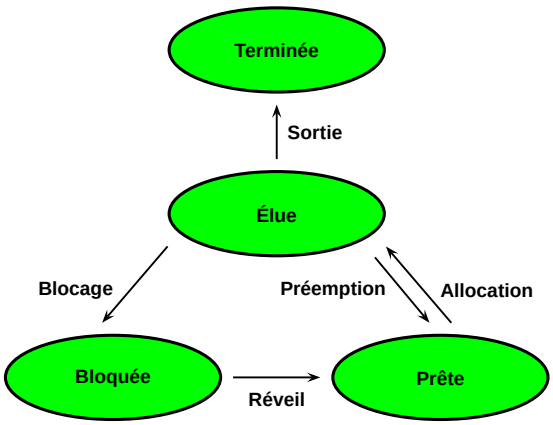
- Suite d'instructions agissant sur un ensemble de données dans un contexte d'exécution défini
- Traitement nécessitant l'allocation d'une ressource (processeur)
- Déclinaison sous la forme d'un processus ou d'un thread
- Caractérisation essentielle par une capacité (durée d'exécution maximale)

Différents types de tâches

- Périodique
  - Activation à intervalles réguliers
  - Lecture de capteurs, opération de scrutation, traitement cyclique, etc.
- Sporadique
  - Temps minimal entre deux activations
  - Manifestation attendue, mais incertaine, d'un événement ou d'une action, etc.
- Apériodique
  - Aucune connaissance des dates d'activation
  - Déclenchement d'une alarme, manifestation subite d'un événement ou d'une action, etc.

Différents états d'une tâche

- Élue
- Bloquée
- Prête
- Terminée

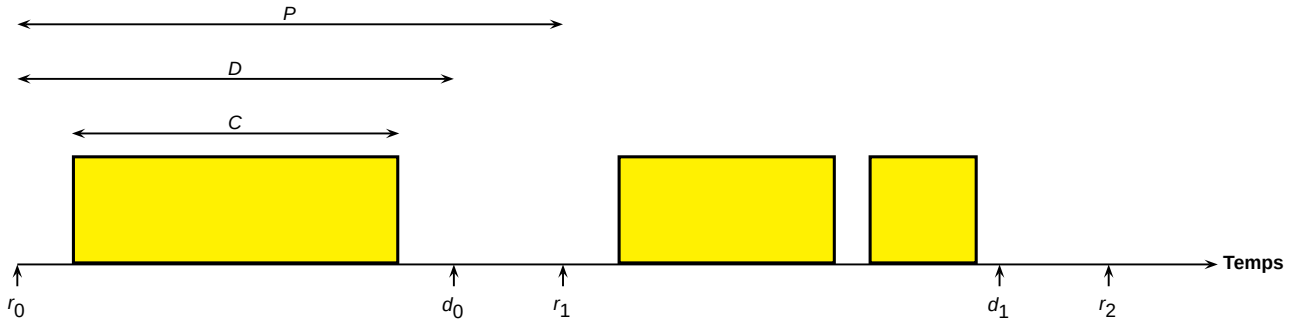


Contraintes potentielles sur les tâches

- Précédence
- Accès à une ressource en exclusion mutuelle

Modèle canonique d'une tâche périodique

- r_i : Date de réveil de la $i^{\text{ème}}$ instance
- d_i : Date d'échéance de la $i^{\text{ème}}$ instance
- C : Capacité
- D : Délai critique
- P : Période
- Cas d'échéance sur requête lorsque $D = P$



Objectifs visés

- Répartition de la ressource processeur entre les différentes tâches
- Choix de la tâche courante (exclusion mutuelle et politique d'arbitrage)
- Mise en œuvre d'une discipline garantissant le respect des contraintes de temps pour l'ensemble du jeu de tâches
- Utilisation optimale de la ressource processeur

Facteur d'utilisation du processeur

- Évaluation par la période et la capacité des tâches

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \text{avec} \quad U \leq 1 \ \ (100\%)$$

- Exemple

$$\left.\begin{array}{l} T_1 : C_1 = 5, P_1 = 25 \\ T_2 : C_2 = 6, P_2 = 20 \\ T_3 : C_3 = 3, P_3 = 15 \end{array}\right\} \quad U = 0.7 \quad (70\%)$$

Classification des disciplines d'ordonnancement

- Préemptif/non préemptif
- En ligne/hors ligne
- Priorités statiques/priorités dynamiques
- Oisif/non oisif
- Mono-processeur/multi-processeurs

Disciplines d'ordonnancement classiques

- Premier arrivé-premier servi (FIFO)
- Tourniquet (RR)

Rate monotonic (1/9)

- Introduction en 1973
- Ordonnancement préemptif hors ligne à priorités statiques
- Considération de tâches indépendantes et périodiques de capacité connues
- Affectation des priorités aux tâches de manière inversement proportionnelle à la période
- Priorité la plus élevée à la tâche la plus fréquente

Rate monotonic (2/9)

- Existence d'une borne sur le taux d'utilisation du processeur (condition suffisante)

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \text{avec} \quad U \leq n(2^{\frac{1}{n}} - 1)$$

- Toujours ordonnançable lorsque $U \leq 0.693$ (69.3%)

Rate monotonic (3/9)

- Premier exemple

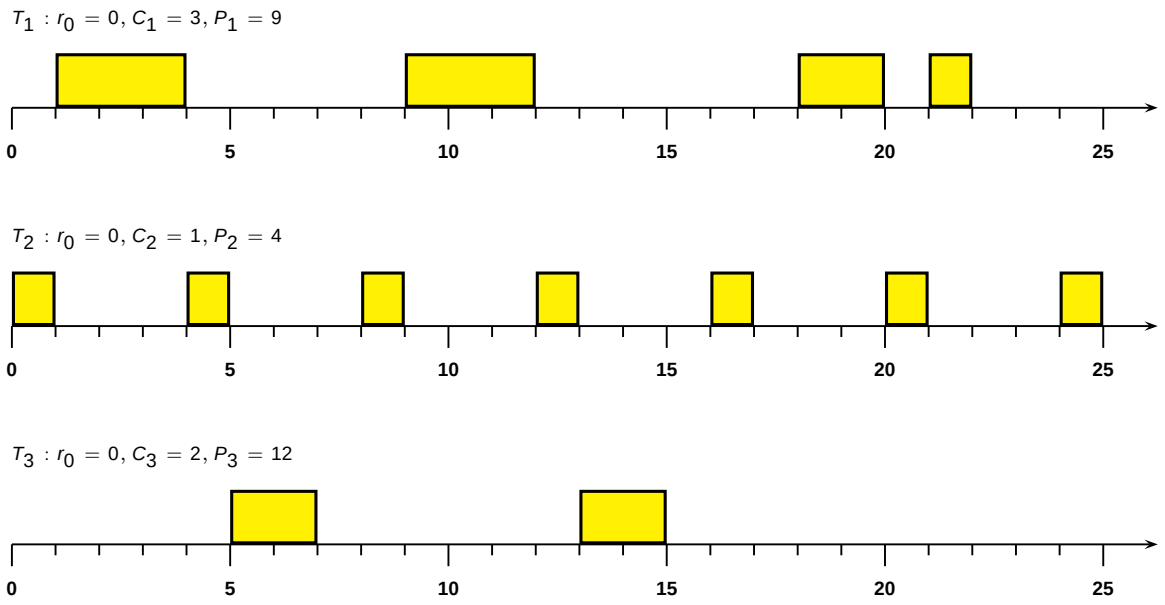
$$\left.\begin{array}{l} T_1 : r_0 = 0, C_1 = 3, P_1 = 9 \\ T_2 : r_0 = 0, C_2 = 1, P_2 = 4 \\ T_3 : r_0 = 0, C_3 = 2, P_3 = 12 \end{array}\right\} \quad U = 0.75 \quad (75\%)$$

- Condition suffisante vérifiée

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \leq \quad n(2^{\frac{1}{n}} - 1) = 0.78$$

Rate monotonic (4/9)

- Vérification graphique



Rate monotonic (5/9)

- Second exemple

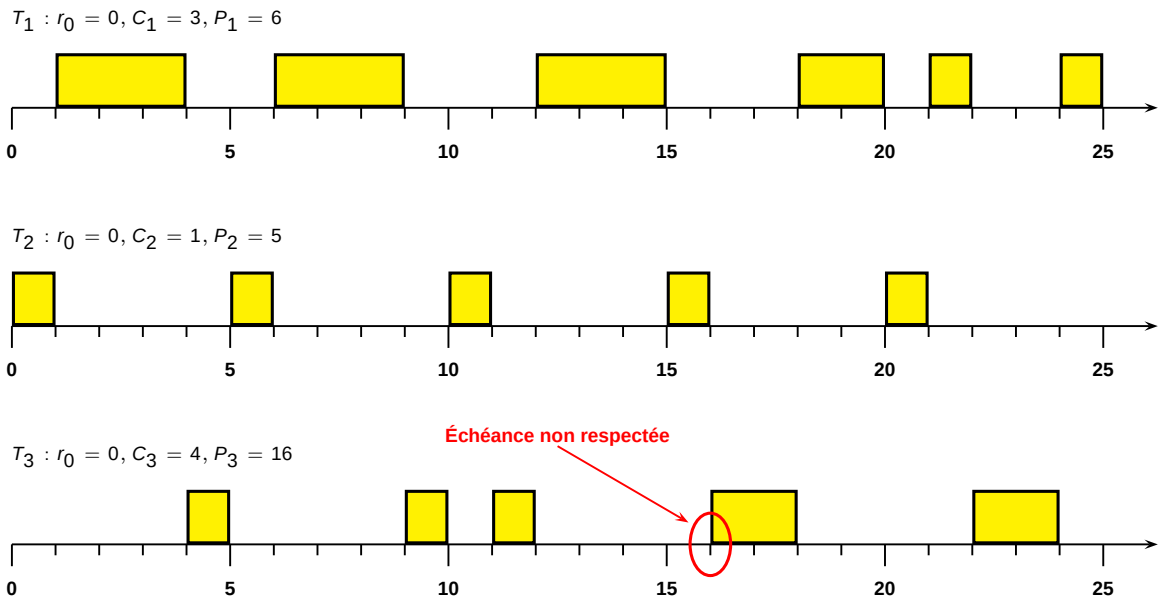
$$\left.\begin{array}{l} T_1 : r_0 = 0, C_1 = 3, P_1 = 6 \\ T_2 : r_0 = 0, C_2 = 1, P_2 = 5 \\ T_3 : r_0 = 0, C_3 = 4, P_3 = 16 \end{array}\right\} \quad U = 0.95 \quad (95\%)$$

- Condition suffisante non vérifiée

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \leq \quad n(2^{\frac{1}{n}} - 1) = 0.78$$

Rate monotonic (6/9)

- Vérification graphique



Rate monotonic (7/9)

- Troisième exemple

$$\left.\begin{array}{l} T_1 : r_0 = 0, C_1 = 2, P_1 = 6 \\ T_2 : r_0 = 0, C_2 = 1, P_2 = 3 \\ T_3 : r_0 = 0, C_3 = 4, P_3 = 12 \end{array}\right\} \quad U = 1 \quad (100\%)$$

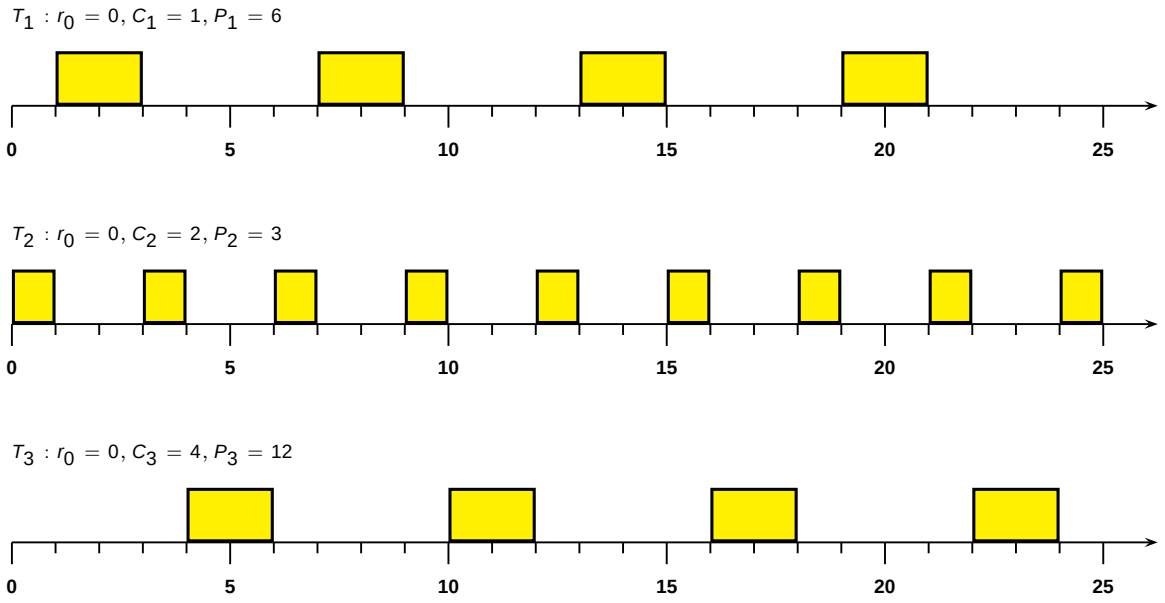
- Condition suffisante non vérifiée

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad > \quad n(2^{\frac{1}{n}} - 1) = 0.78$$

- Configuration harmonique

Rate monotonic (8/9)

- Vérification graphique



Rate monotonic (9/9)

- Utilisation limitée aux tâches à échéance sur requête
- Possibilité d'obtention d'une charge CPU totale en cas de configuration harmonique
- Borne théorique pouvant souvent être dépassée et repoussée à 88%

Deadline monotonic (1/2)

- Ordonnancement préemptif hors ligne à priorités statiques
- Considération de tâches indépendantes et périodiques de capacité connues
- Affectation des priorités aux tâches en fonction de leur délai critique
- Priorité la plus élevée à la tâche dont le délai critique est le plus court
- Performances équivalentes à RM pour les tâches à échéance sur requête
- Performances meilleures que RM pour les configurations quelconques

Deadline monotonic (2/2)

- Condition suffisante d'ordonnançabilité

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} \quad \text{avec} \quad U \leq n(2^{\frac{1}{n}} - 1)$$

- Toujours ordonnançable lorsque $U \leq 0.693$ (69.3%)
- Possibilité de dépassement de la borne $n(2^{\frac{1}{n}} - 1)$ dans certains cas

Earliest deadline first (1/4)

- Ordonnancement préemptif en ligne à priorités dynamiques
- Considération de tâches indépendantes et périodiques de capacités connues
- Affectation dynamique des priorités aux tâches de manière croissante et inversement proportionnelle au délai restant avant échéance
- Priorité la plus élevée à la tâche dont l'échéance est la plus proche

Earliest deadline first (2/4)

- Évaluation du taux d'utilisation du processeur pour des tâches à échéance sur requête (condition nécessaire et suffisante)

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \text{avec} \quad U \leq 1$$

- Évaluation du taux d'utilisation du processeur pour des tâches quelconques (condition suffisante)

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} \quad \text{avec} \quad U \leq 1$$

Earliest deadline first (3/4)

- Exemple

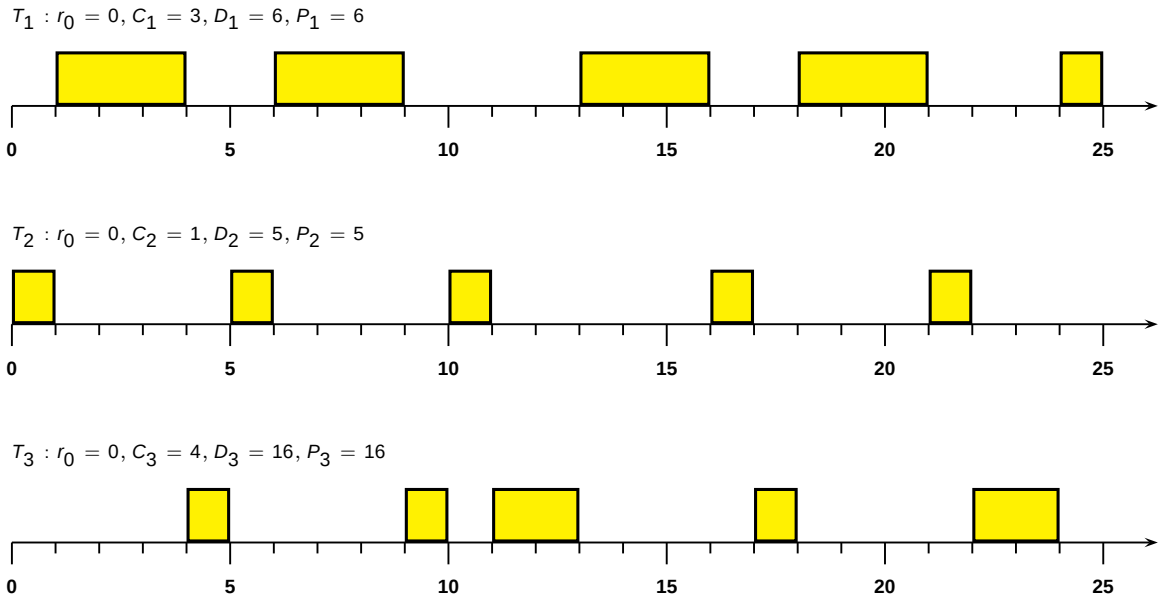
$$\left.\begin{array}{l} T_1 : r_0 = 0, C_1 = 3, D_1 = 6, P_1 = 6 \\ T_2 : r_0 = 0, C_2 = 1, D_2 = 5, P_2 = 5 \\ T_3 : r_0 = 0, C_3 = 4, D_3 = 16, P_3 = 16 \end{array}\right\} \quad U = 0.95 \quad (95\%)$$

- Condition nécessaire et suffisante vérifiée

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad \leq \quad 1$$

Earliest deadline first (4/4)

- Vérification graphique

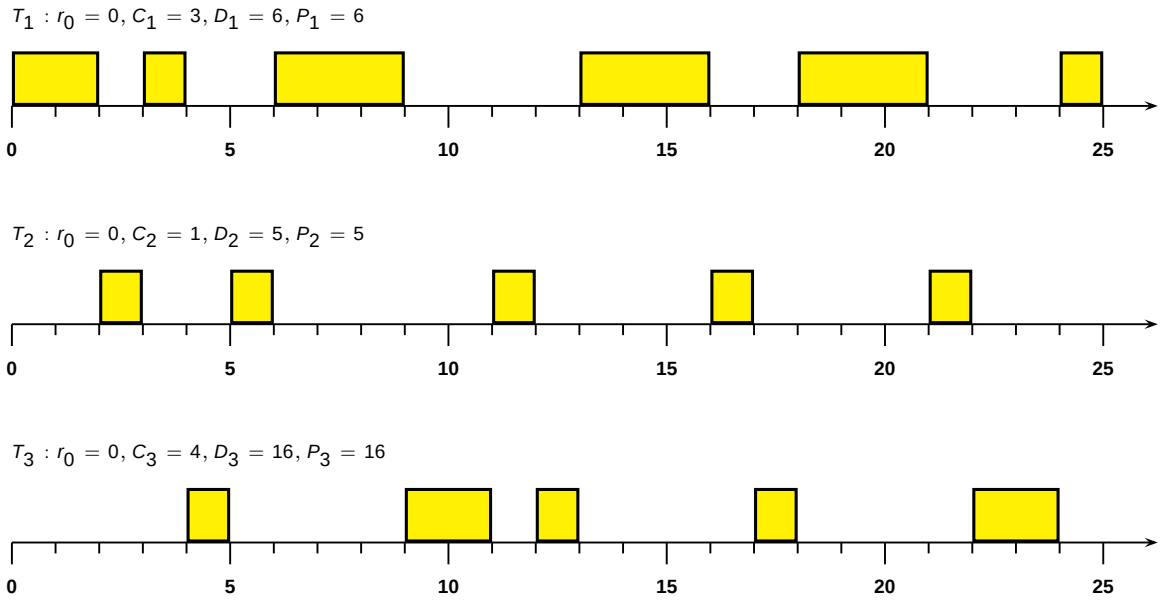


Least laxity first (1/2)

- Ordonnancement préemptif en ligne à priorités dynamiques
- Considération de tâches indépendantes et périodiques de capacités connues
- Affectation des priorités aux tâches en fonction de leur laxité dynamique
- Priorité la plus élevée à la tâche dont la marge de temps restant avant échéance est la plus réduite
- Conditions d'ordonnançabilité identiques à EDF
- Changements de contexte généralement plus fréquents que pour EDF

Least laxity first (2/2)

- Vérification graphique (même jeu de tâches que pour EDF)



Premier arrivé-premier servi (SCHED_FIFO)

- Gestion temps réel des processus
- Niveaux de priorité de 1 à 99
- File d'attente de processus pour chaque niveau de priorité
- Exécution du premier processus de la file de plus haute priorité
- Pas de préemption dans une même file d'attente
- Préemption d'une file sur une autre moins prioritaire

Tourniquet (SCHED_RR)

- Gestion temps réel des processus
- Niveaux de priorité de 1 à 99
- File d'attente de processus pour chaque niveau de priorité
- Exécutions successives des processus de la file de plus haute priorité
- Préemption d'une file sur une autre moins prioritaire

Gestion en temps partagé (SCHED_OTHER)

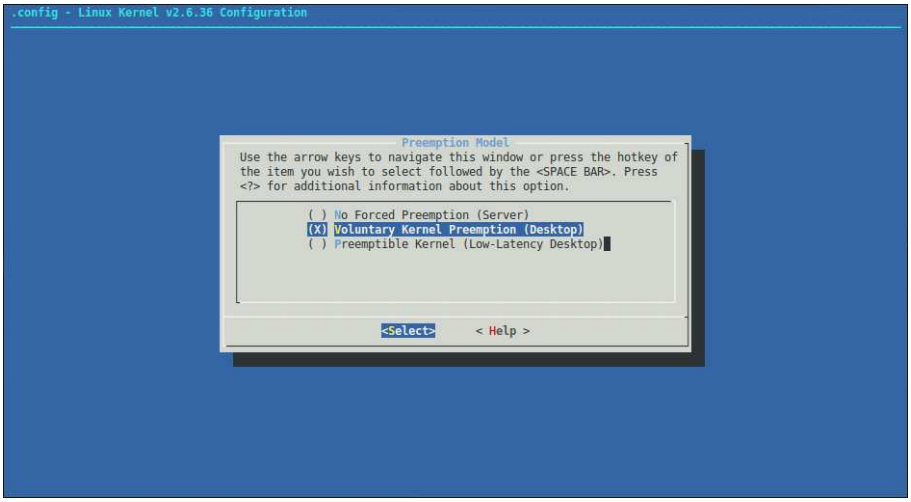
- Gestion par défaut des processus
- Priorité dynamique d'un processus prêt à être exécuté déterminée par incrémentation à partir du niveau statique de courtoisie (commande `nice`) et du temps d'attente écoulé
- Priorité statique de niveau 0
- Préemption par les disciplines SCHED_FIFO et SCHED_RR

Gestion en temps partagé (SCHED_BATCH)

- Depuis Linux 2.6.16
- Considération d'un processus exigeant en temps CPU
- Application d'une pénalité pour les traitements par lot (non interactifs)

Configuration du noyau

- Réduction du temps de latence
- Trois solutions standards disponibles



Pas de préemption (PREEMPT_NONE)

- Pas de préemption du code s'exécutant en mode noyau
- Limitation des changements de contexte

Préemption volontaire (PREEMPT_VOLUNTARY)

- Ajout de points explicites de préemption dans le noyau
- Intégration basée sur les appels à la fonction `might_sleep`
- Limitation de placement des points de préemption à des zones déterminées du noyau

Préemption complète (PREEMPT)

- Préemption totale du noyau
- Réaction aux événements interactifs

Intégration du patch PREEMPT_RT

- Ajout d'un quatrième mode de préemption au noyau
- Extension de l'ordonnanceur standard
- Temps réel dur
- Réduction du temps de latence moyen à 20/30 microsecondes
- Temps de latence maximal entre 100 et 500 microsecondes

Caractéristiques

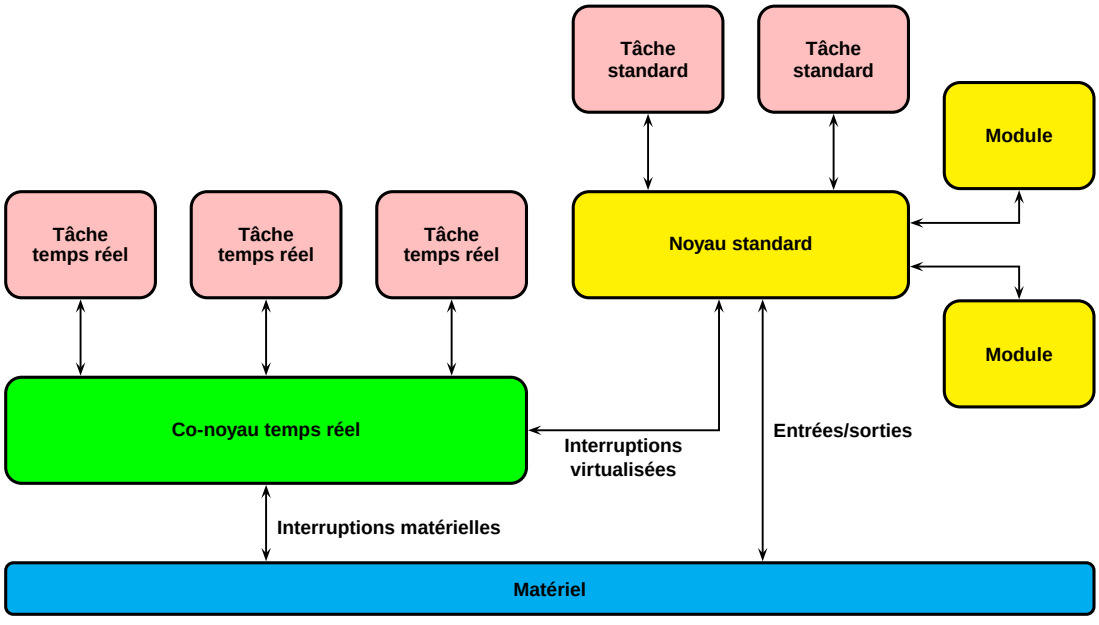
- Ajout d'un co-noyau pour la gestion temps réel
- Intégration d'un sous-système temps réel dans un module noyau
- Virtualisation des interruptions
- Routage prioritaire des interruptions vers le co-noyau
- Utilisation d'un ordonnanceur temps réel spécifique
- Gestion du noyau standard comme une tâche de bas niveau de priorité
- Temps réel dur
- Temps de latence maximal de 15/20 microsecondes

Principales solutions

- RTLinux
- RTAI
- Xenomai

Structure générale

- Intégration du co-noyau



6 Aspects juridiques
- Logiciels libres
- Logiciels "open source"
- Notion de Copyleft
- Différentes licences
- Validité juridique de la licence GPL
- Exemples concrets

Définition

"*Un logiciel libre est un logiciel qui peut être utilisé, copié, étudié, modifié et redistribué sans restriction majeure autre que la mise à disposition du code source.*"

Libertés fondamentales selon la FSF

- Liberté d'exécuter le programme pour tous les usages
- Liberté d'étudier le fonctionnement du programme et de l'adapter (accès au code source requis)
- Liberté de redistribuer des copies
- Liberté d'améliorer le programme et de publier les améliorations pour en faire profiter toute la communauté (accès au code source requis)

Définition

"*Un logiciel "open source" est un logiciel dont la licence respecte notamment la possibilité de libre redistribution, d'accès au code source et de travaux dérivés.*"

Critères selon l'OSI

- Redistribution libre et gratuite du programme
- Livraison du code source
- Respect de la licence d'origine pour les travaux dérivés
- Préservation de l'intégrité du code source de l'auteur
- Aucune discrimination envers des personnes ou des groupes
- Aucune discrimination envers des domaines d'application
- Pas besoin de se conformer à des termes de licences complémentaires
- Pas de licence spécifique à un produit
- Pas de contamination des autres programmes conjointement distribués
- Neutralité vis-à-vis de la technologie utilisée

Définitions

"*Le Copyleft est la possiblité de copier, d'utiliser, d'étudier, de modifier et de distribuer un logiciel dans la mesure où cette possibilité est préservée.*"

Caractéristiques

- Opposition à la notion de Copyright
- Conservation du statut libre pour un logiciel dérivé
- Garantie que le code libre le restera dans toutes ses modifications
- Impossibilité de distribution d'un logiciel propriétaire incorporant du code sous licence avec Copyleft

GPL (General Public Licence)

- Licence avec Copyleft
- Publication par Richard STALLMAN et Eben MOGLEN pour la définition des conditions légales de distribution des logiciels libres du projet GNU
- Placement sous licence GPL des programmes GPL modifiés ou dérivés ainsi que les logiciels intégrant du code de programmes GPL même lié statiquement ou dynamiquement (application uniquement aux logiciels distribués)
- Clause spéciale pour GCC (permission de compiler un programme sans placement sous GPL)
- Application des libertés fondamentales de la FSF

LGPL (Lesser General Public Licence)

- Licence avec Copyleft publiée par la FSF
- Version limitée, ou amoindrie, de la licence GPL pour permettre à certains logiciels libres de pénétrer certains domaines n'autorisant pas le choix d'une publication entièrement libre
- Permission d'intégrer sans contrainte une partie de code non modifiée d'un programme sous licence LGPL (placement sous licence LGPL si le code est modifié)
- Autorisation de lier statiquement ou dynamiquement un programme à une bibliothèque sous licence LGPL sans contrainte de licence ou de distribution de code source
- Autorisation de lier un programme sous licence LGPL à une bibliothèque non LGPL sans pour autant révoquer la licence

MIT ou X11 (Massachusetts Institute of Technology)

- Licence sans Copyleft
- Autorisation donnée à toute personne recevant un logiciel sous licence MIT/X11 le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et de changer sa licence
- Seule obligation de mettre le nom des auteurs avec la notice copyright

BSD (Berkeley Software Distribution)

- Licence sans Copyleft
- Conditions aujourd'hui identiques à la licence MIT/X11, mais la licence BSD contenait jusqu'en 1999 une clause publicitaire maintenant disparue

CeCILL (CEA CNRS INRIA Logiciel Libre)

- Licences francophones
- Version principale avec Copyleft et compatible avec la licence GPL
- Garantie du respect des principes du logiciel libre, à savoir le libre accès au code source, la libre utilisation, la libre modification et la libre redistribution
- Garantie du respect du droit français aux créateurs et aux utilisateurs de logiciels libres en termes de responsabilité cicvile et de propriété intellectuelle
- Versions B, compatible avec les licences BSD et X11 (sans Copyleft) et C, compatible avec la licence LGPL (avec Copyleft)

Cas Free

- Procès en cours (contre FSF France) pour non distribution aux utilisateurs finaux (abonnés) du code GPL utilisé
- Objet du procès lié à la définition de la notion de distribution
- Pas d'achat de l'équipement ("box") par l'utilisateur final (appartenance à Free)
- Facturation exceptionnelle du client en cas de non renvoi après résiliation

Cas Skype

- Téléphone WSKP100 commercialisé, mais non fabriqué par Skype
- Utilisation d'une version modifiée du noyau Linux pour le firmware
- Non respect de la distribution du code source aux utilisateurs finaux

Cas Edu4

- Reconnaissance, le 16 septembre 2009, par la cour d'appel de Paris de la culpabilité de la société Edu4 de ne pas avoir fourni à son client (AFPA) le code source d'une version modifiée du logiciel libre VNC utilisé par le prestataire
- Reconnaissance à l'utilisateur du logiciel le droit de faire respecter les termes de la licence GPL
- Pas d'intervention de l'auteur au procès du logiciel

Cas #1

- Développement d'un produit sous licence quelconque (libre ou non) et utilisation d'une librairie sous licence GPL
- Notion ambiguë de liaison des librairies, mais constat des situations suivantes :
  - Contamination si la librairie est liée statiquement au produit
  - Pas de contamination si la librairie est appelée dynamiquement par le produit
- Prise en compte spécifique de ce cas par la licence LGPL

Cas #2

- Introduction par copié/collé de morceaux de code sous licence GPL dans un logiciel développé par une entreprise
- Contamination du code par la licence GPL et obligation de fourniture du code source au client

Cas #3

- Développement d'un produit sous licence GPL, car dérivé d'un projet à l'origine sous licence GPL (ex. : personnalisation d'un noyau), facturation et livraison du code source à un client
- Revente ou cession possible par le client avec fourniture du code source
- Perte systématique d'exclusivité lors de la revente ou cession d'un produit sous licence Copyleft

7 Conclusion

Bilan

- Linux est une alternative sérieuse pour l'embarqué
- Existence d'une large communauté de développeurs et d'utilisateurs
- Disponibilité d'une documentation étendue
- Possibilités pour le temps réel
- Vigilance quant aux licences

Questions

?