Systèmes d'exploitation embarqués et temps-réel

Chapitre 1 : Introduction

Aimen Bouchhima

2020-2021

Plan

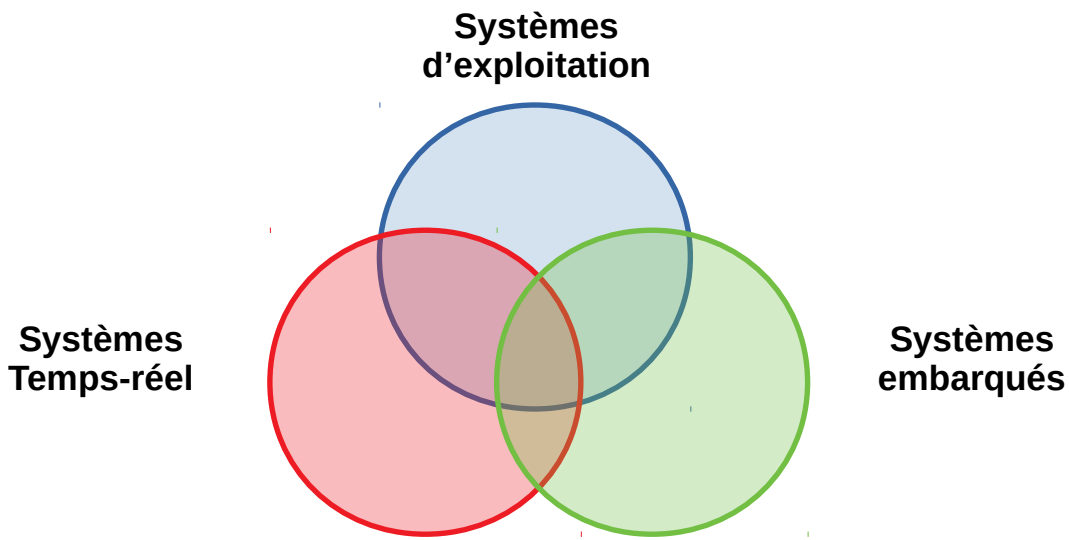
Information utiles

- Quelques références :
  - Sam Siewert, John Pratt, “Real-time embedded components and systems, with linux and RTOS”, Mercury Learning and Information, 2016
  - William Stallings, “Operating Systems. Internals and Sesign Principles”, Seventh Edition, Prentice Hall, 2012
- Cours intégré
- Prérequis
  - Systèmes d'exploitation
  - Systèmes embarqués
  - Programmation C

Définition

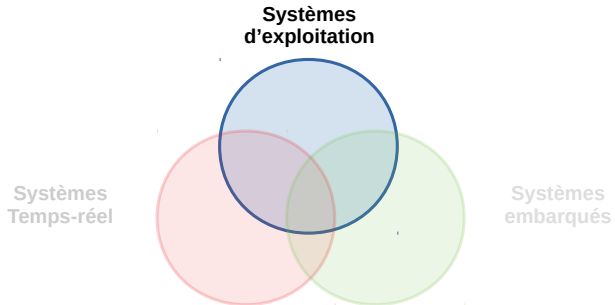
Un système d'eploitation embarqué et temps-réel est un :

- Système d'exploitation
- Destiné pour une cible (matériel) embarquée
- Offrant des garanties de temps de réponse



Système d'exploitation

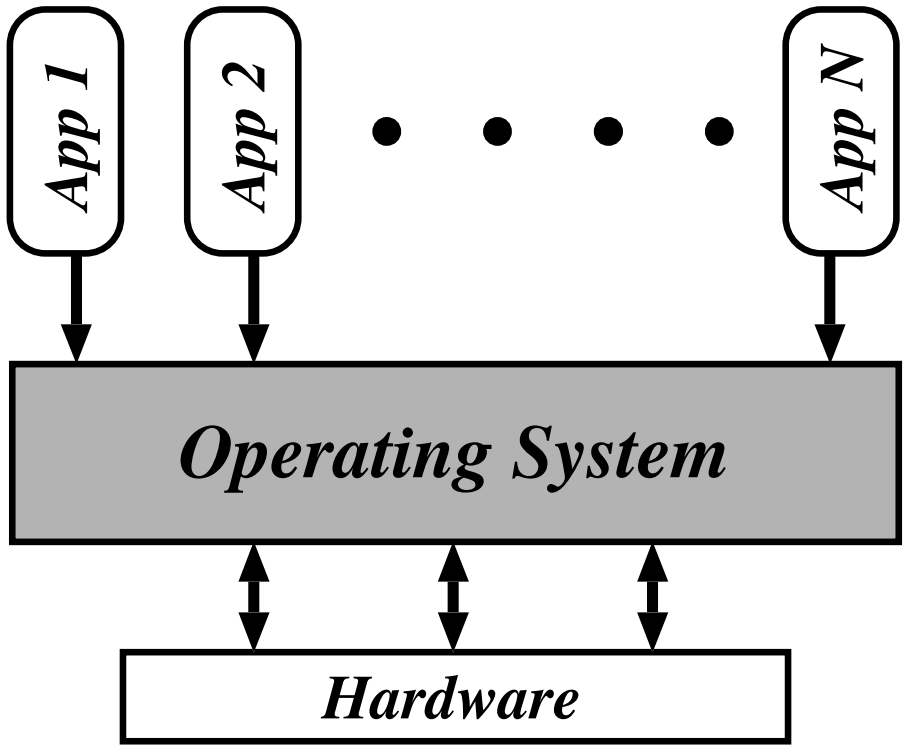
Un système d'exploitation (OS) est un ensemble de programmes permettant de gérér les ressources matérielles de la machine en vue de fournir des services pour les logiciels applicatifs. un OS peut être donc vu sous deux angles:



- Un fournisseur de services : il exporte une interface de programmation (Application programmer Interface ou API)
- Un gestionnaire de ressources : il implémente des politiques pour gérér efficacement les ressources (limitées)
  - Ordonancement (CPU)
  - Allocation/Protection mémoire, mémoire virtuelle (Mémoire)
  - Accès au périphériques, système de fichier virtuel (Périphériques)

Système d'exploitation

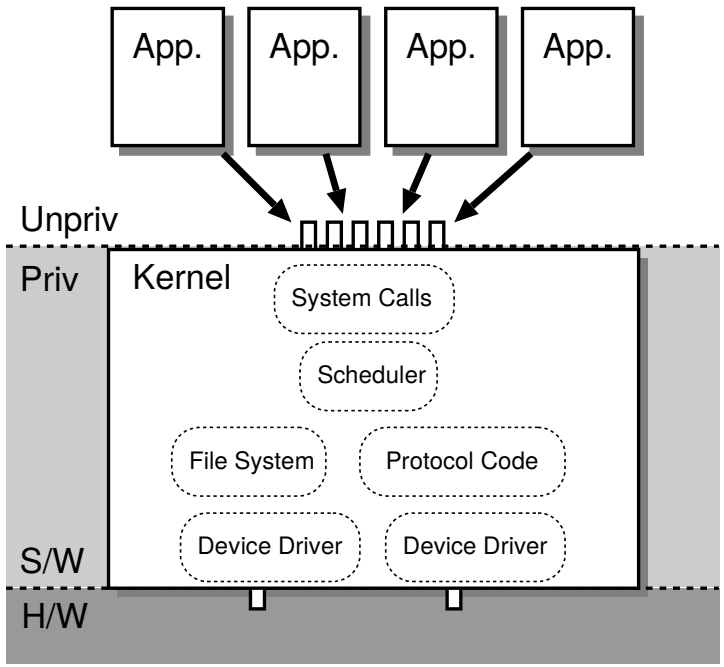
Schéma conceptuel d'un système d'exploitation



Architecture d'un système d'explotation

Système d'exploitation à base de noyau (kernel)

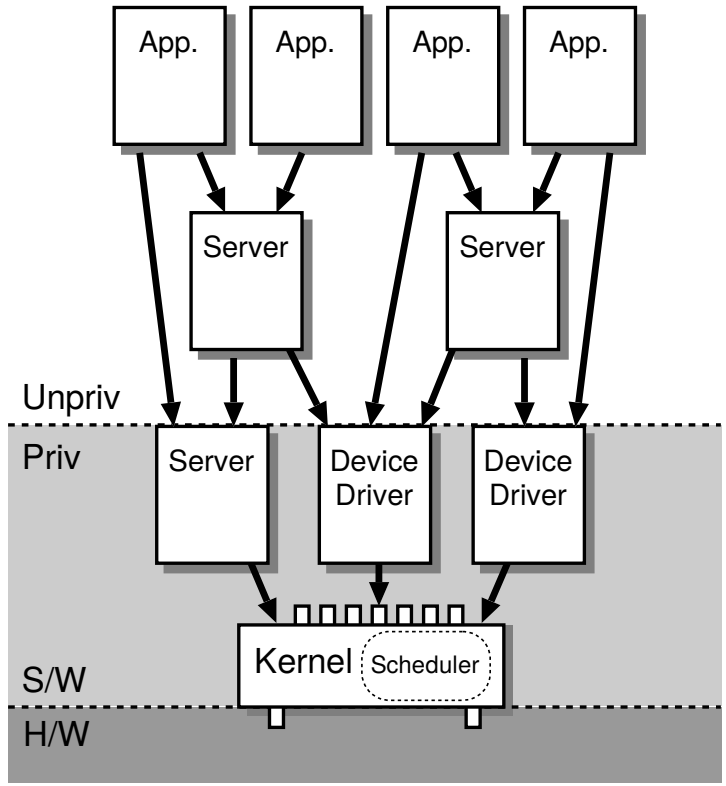
- Appelé aussi OS monolithique
- Tous les services de l'OS sont implémentés dans l'espace privilégié
- l'application utilise ces services via des appels système (System calls) implémentés par des interruptions logicielles
- Exemples : Linux, Windows



Architecture d'un système d'explotation

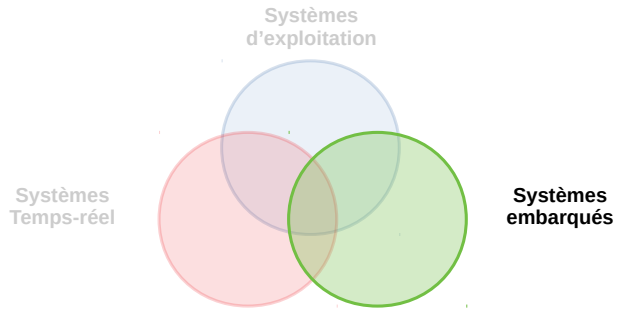
Système d'exploitation à base de micro-noyau (microkernel)

- Conserver dans l'espace privilégié le minimum de services
- Les autres services sont implémentés dans l'espace non-privilégé, généralement sous forme de processus serveurs
- Plus sécurisé, mais pose un problème de performance
- Exemples : Minix 3, Mach 3, L4, HarmonyOS (Huawei)



Systèmes Embarqués

Un système embarqué est défini comme un système électronique et informatique enfouis dans un système physique plus large, et spécialisé dans une tâche bien précise

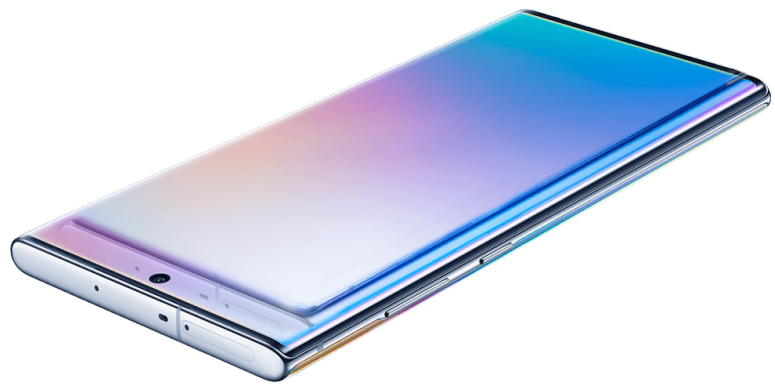


- Contraintes de coût, de consommation d'énergie, de temps de réponse, de sûreté, ...
- Développement important durant les trois dernières décennies grâce à la loi de Moore
  - Systèmes sur puce (System on Chip SoC)
  - Systèmes logicels/matériel pour un compromis performance/flexibilité

Système embarqué : ressources limitées ?

Un système embarqué est classiquement défini comme un système à ressources (matérielles) limitées. Bien qu'elle reste valable pour une large gamme de produits embarqués, d'autres produits échappent aujourd'hui à cette définition

- RAM : 12GB
- Stockage : 512GB
- CPU : Octacore Quallcomm Sapdragon (up to 2.8GHz)



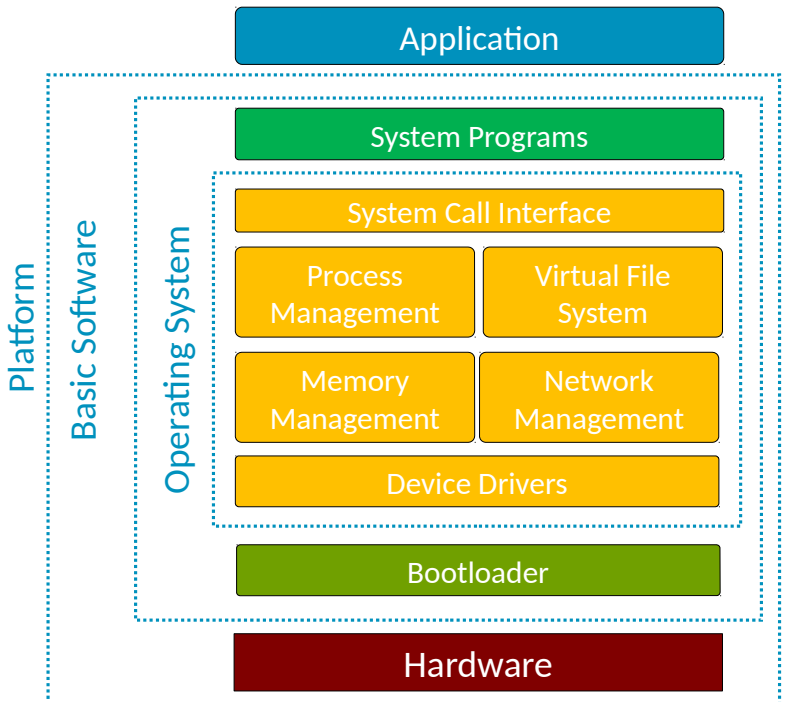
Architecture matérielle d'un SoC

L'architecture matérielle d'un système sur puce (SoC) est composée d'au moins un sous-système processeur (CPU + mémoire) et d'un ensemble de composants matériels (intellectual properties ou IP). Vue la large gamme de produits embarquées, le nombre et les caractéristique de ces composants varient considérablement

- Famille des micro-contrôleurs : ressources matérielles limités, très faible consommation d'énergie.
- Famille des SoC application : ressources CPU et mémoire plus importantes, présence d'accélérateurs matériel (exemple GPU)

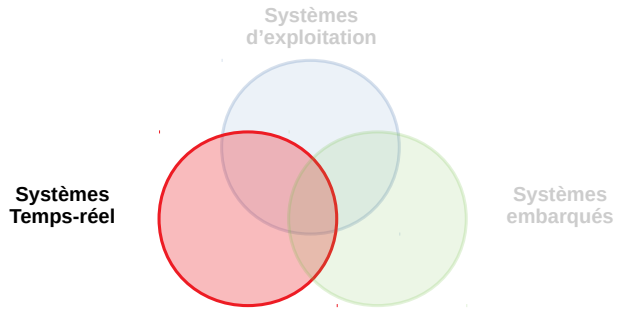
Composants logiciels/matériels

- Plateforme : Ensemble de composants logiciels et matériels qui exécutent une application
- Programmes système : Utilitaires (applications) systèmes pour accèder aux services de l'OS
  - Exemple : ls, grep, ...
- Bootloader : programme de démarrage, initialise le matériel et charge l'OS à partir du périphérique de stockage de masse.



Définition d'un système temps-réel

Les systèmes informatiques temps réel se différencient des autres systèmes informatiques par la prise en compte de contraintes temporelles dont le respect est aussi important que l'exactitude du résultat, autrement dit le système ne doit pas simplement délivrer des résultats exacts, il doit les délivrer dans des délais imposés.

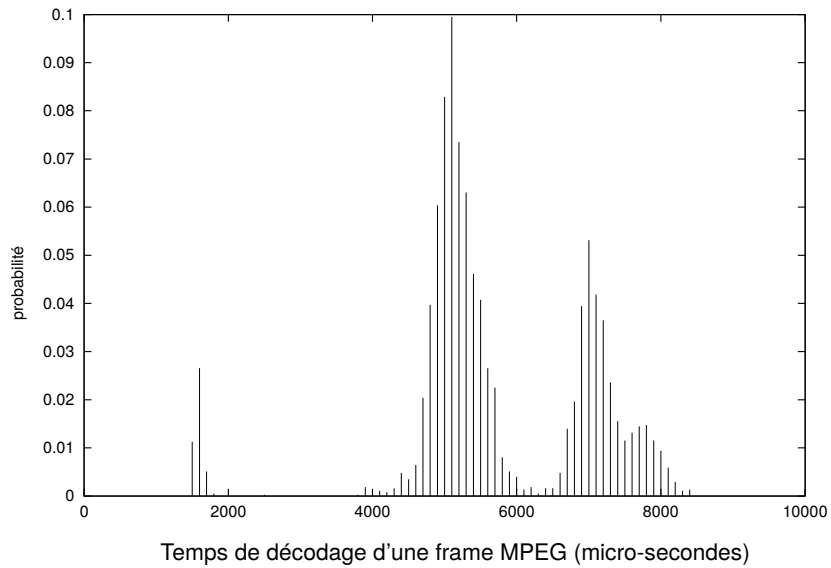


- Un résultat exact produit en retard est un résultat faux
- Contraintes temporelles modélisées par la notion de délai ou échéance. Par exemple, le système doit:
  - réagir à un évènement venant de l'environnement dans un délai précis
  - répéter une action périodiquement avec une période fixe

Déterminisme et temps-réel

Pour pouvoir respecter les contarintes temporelles, un système temps-réel doit être déterministe dans le sens ou le temps pris pour effectuer un certain traitement (action) doit être borné et connu à l'avance. L'indétermisme peut être lié à plusieurs sources

- Les données en entrée
- Etat du système d'exploitation (présence d'autres tâches concurrentes)
- Aspects liés au matériel (effets des caches, pipeline processeur, translation d'adresse)



1

Débit vs temps-réel

Les systèmes informatiques classiques et les systèmes temps-réel ont des objectifs différents

- Les systèmes informatiques classiques (appelés aussi systèmes à usage général) sont aptimisé pour "le cas le plus courant". Donc, en moyenne, ils sont rapides dans le sens où le nombre de tâches effectuées par seconde (le débit) est élevé
- Dans les systèmes temps-réel on résonne plutôt sur le "pire cas" (worst case) pour garantir le respect de la contrainte temporelle.

Choix de l'OS embarqué et temps-réel

- Le choix de l'OS doit tenir compte de :
  - La nature temps-réel du système : temps-réel strict (dûr) vs temps-réel souple
  - La nature de la cible matérielle : micro-contrôleurs vs SoC application
- Les deux aspects ne sont pas orthogonaux : souvent, les micro-contrôleurs offrent des caractéristiques temporelles plus adaptées au temps-réel strict (pas de caches, latence minimale des interruption, pipeline CPU simple ...)

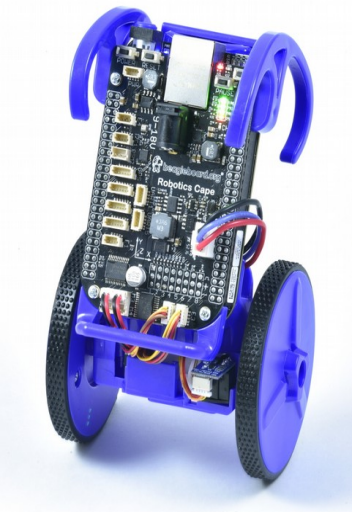
Choix de l'OS embarqué et temps-réel

- Dans la pratique, les combinaisons suivantes sont utilisées:
  - (temps-réel strict/souple + microcontrôleur) : on utilise des petits systèmes d'exploitation temps-réel (RTOS)
    - FreeRTOS(Amazon), RTX(ARM), Riot, eCos,...
  - (temps-réel souple + SoC application) : on utilise des systèmes d'exploitation modèrnes
    - Linux embarqué, Windows CE, iOS, ...

Pourquoi Linux Embarqué

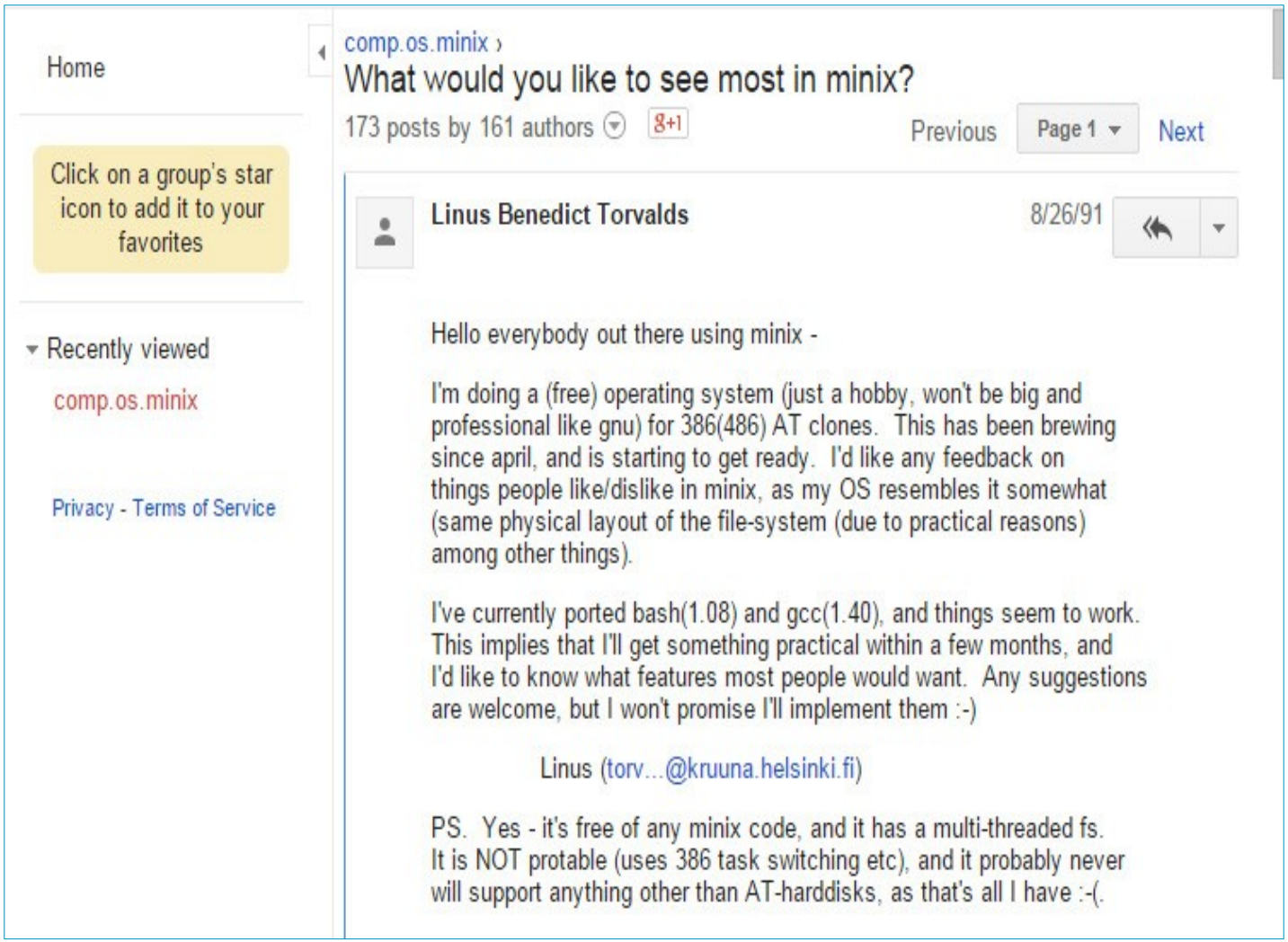
- Open Source (GNU General Public License v2.0 : GPLv2)
  - Le code source peut être étudié et adapté
- Communauté riche et engagée
  - Entreprises, Développeurs libres, Universités, ...
- Supporte une large gamme de processeurs/machines
  - ARM, x86, PowerPC, SPARC, ...
- Supporté par un écosystème très large de produits logiciels (open source)
  - Applications, bibliothèques, middleware, bootloader, ...

Linux embarqué dans ...



L'origine de Linux

Tout a commencé par ce post à comp.os.minix, le 26 Aout 1991



Home

Click on a group's star icon to add it to your favorites

Recently viewed

comp.os.minix

Privacy - Terms of Service

comp.os.minix ›

What would you like to see most in minix?

173 posts by 161 authors

Previous Page 1 Next

Linus Benedict Torvalds 8/26/91

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.