Université de Bretagne Occidentale

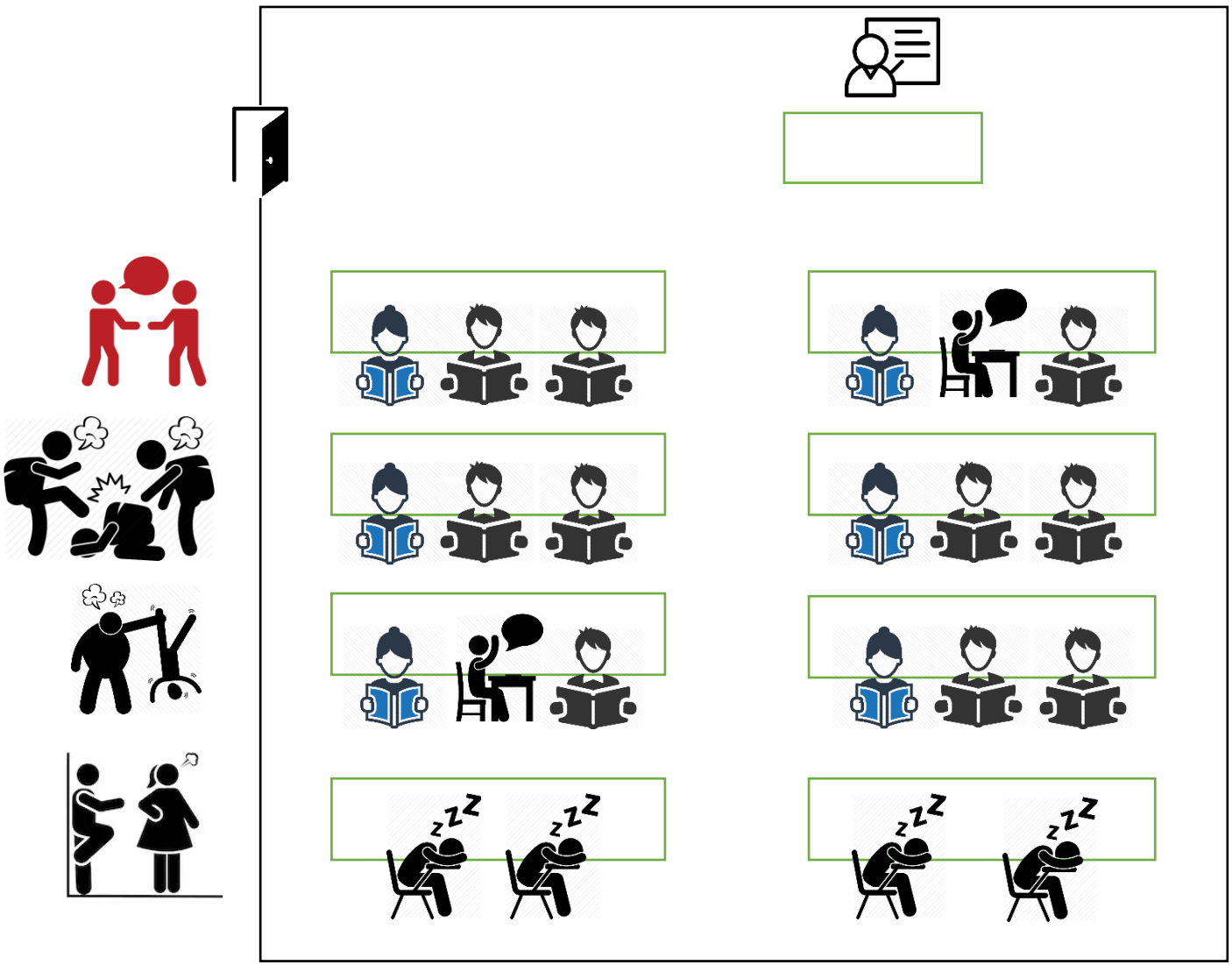
Systèmes d‘Exploitation pour l‘Embarqué

Introduction

Hai Nam TRAN (https://lab-sticc.univ-brest.fr/~htran/)
Université de Bretagne Occidentale – M2 LSE

Le CM est inspiré du cours précédemment dispensé par Jalil Boukhobza

Aménagement



Ce cours

- **Contient 24 séances de 2h**
  - **Des cours : 8 séances**
  - **Des TPs : 16 séances**
  - **Pas de TDs**
- **Evaluation**
  - **Contrôle continue (1/2)**
    - Plusieurs comptes rendus de TP
  - **Examen final (1/2)**

Contenu du cours

- **2 intervenants**
  - **Hai Nam TRAN (hai-nam.tran@univ-brest.fr)**
  - **Frank SINGHOFF (frank.singhoff@univ-brest.fr)**

Contenu du cours

- **[HNT]**
  - **Revoir la plupart les fonctionnalités des OS en se focalisant sur ce qui est spécifique à l'embarqué**
  - **TP**
    - Etude approfondie du système Linux (une partie)
    - Compilation de noyau Linux pour l'embarqué
      - Manipulation des outils permettant de le faire
- **[FS]**
  - **Système d'exploitation temps-réel : RTEMS, POK**
  - **Architectures partitionnées, TSP/ARINC 653**

Plan

1. Qu’est ce qu’un système embarqué
2. Deux types de performance : Latence vs Débit
3. AspenCore Embedded Markets Study
4. Les architectures des systèmes d’exploitation

Plan

1. **Qu’est ce qu’un système embarqué**
2. **Deux types de performance : Latence vs Débit**
3. **AspenCore Embedded Markets Study**
4. **Les architectures des systèmes d’exploitation**

Qu’est ce qu’un système embarqué ?

- **Ébauche d’une définition**
  - **C’est un système électronique et informatique autonome qui est dédié à une tâche particulière et contenue dans un système englobant.**
    - Il n’est « généralement » pas programmable
  - **Pas d’E/S standards**
  - **Matériel et application intimement liés**
  - **Logiciel enfoui … noyé dans le matériel … pas facilement discernable comme dans un PC.**
- **Ils sont partout !!!**
  - **Radio/réveil**
  - **Machine à café**
  - **Télévision/télécommande**
  - **Moyen de transport**
  - **Fenêtre, sonnette, moniteur**

Qu’est ce qu’un système embarqué ?

- **Caractéristiques principales d’un système embarqué**
  - **Système principalement numérique**
  - **Met généralement en œuvre un processeur**
  - **Exécute une application logicielle dédiée précise**
  - **Il n’a pas réellement de clavier standard et l’affichage est limité**
  - **Ce ne sont pas des PC, mais des architectures similaires (x86) basse consommation (l’une des contraintes …)**

Quelques propriétés des systèmes embarqués

Simple

Interface spécifique

Fiable

Sécurisé

Maintenable

Optimisé

Cible

Logiciel/matériel embarqué

- **Logiciel embarqué : programme/application utilisé dans un équipement et complètement intégré.**
- **Système embarqué : Matériel(s) + logiciel(s) (+ OS)**
  - **2 types de systèmes embarqués (UNE classification)**
    - **Systèmes embarqués destinés à l'utilisateur (high-end)** : généralement une version dégradée d'un OS existant (ex: Linux). Ex: routeurs, smartphone, etc.
    - **Systèmes embarqués profondément enfouis** : peu de fonctions, très petite empreinte mémoire, généralement construit "from scratch". Appareil photo numérique, radio, réveil, etc.
  - **Différences avec les « machines normales »**
    - Prix (production de masse)
    - Performance
    - Consommation (contrainte de consommation → batterie)
    - Simplifier l'architecture
    - Réduire la vitesse d'horloge
    - Réduire l'utilisation mémoire

Système d'exploitation pour l'embarqué ?

- **Les systèmes d'exploitation permettent**
  - **De gérer les ressources matérielles en assurant leurs partages entre les différents utilisateurs/applications**
    - Ressources matérielles : Processeur, RAM, HDD, Camera, GPS
    - Utilisateurs/applications : Navigateur Web, Thunderbird, Maps, Photos
  - **De présenter une interface homogène et générique (en abstrayant la complexité matérielle) mieux adaptée aux utilisateurs/développeurs**
    - Utilisateurs : Android Phones
    - Développeurs

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
```

Système d'exploitation pour l'embarqué ?

- **Pourquoi un système d'exploitation**
  - **Affranchir le développeur de bien connaître le matériel → gain en temps de développement**
    - Les applications doivent avoir un accès aux services de l'OS via des APIs (réutilisabilité du code, interopérabilité, portabilité, maintenance aisée)
  - **Possibilité de bénéficier des mêmes avancées technologiques que les applications classiques (TCP/IP, HTTP, etc.)**
  - **Environnement de développement plus performant**
  - **Réduire time-to-market**

Système d'exploitation pour l'embarqué ?

- **Pourquoi un système d'exploitation pour l'embarqué ?**
  - **Un appareil complexe : smartphone, PC**
    - Grande taille de mémoire, microcontrôleur haute fréquence, grande espace de stockage

  → facile à installer un système d'exploitation

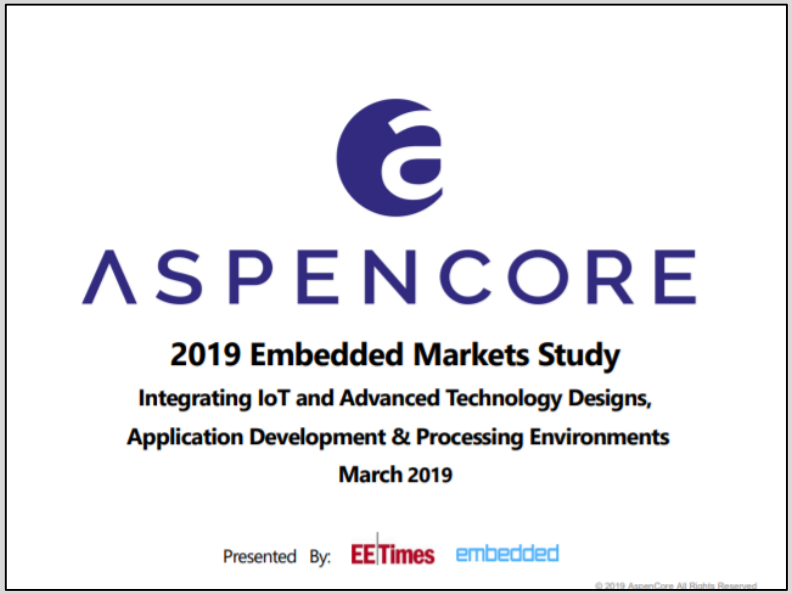
  - **Et pour les appareils petits + ressources limitées ?**

Différences avec les « machines normales »

- Prix (production de masse)
- Performance
- Consommation (contrainte de consommation → batterie)
- Simplifier l'architecture
- Réduire la vitesse d'horloge
- Réduire l'utilisation mémoire

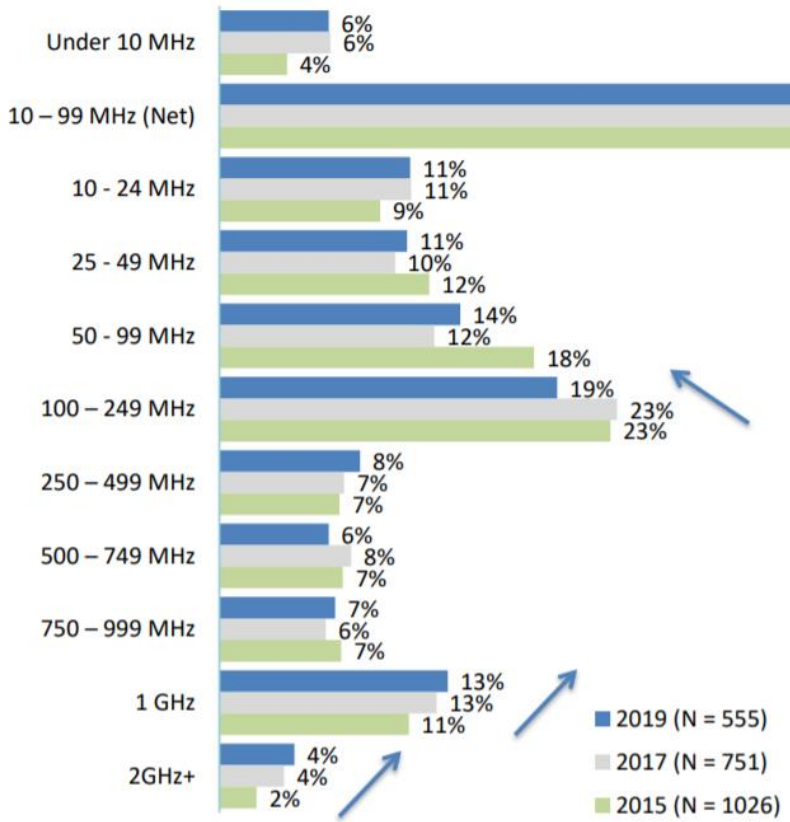
Idée reçue numéro 01

- La fréquence d'horloge se mesure en GHz de nos jours !, en dessous c'est bon pour le placard ! Autant faire du calcul mental tiens !



Quelle est la fréquence d'horloge du processeur utilisé dans votre projet embarqué actuel ?

My current embedded project's main processor clock rate is:



The **average** processor clock rate was:
462 MHz in 2019
445 MHz in 2017
397 MHz in 2015
428 MHz in 2014

EETimes embedded

2019 Embedded Markets Study

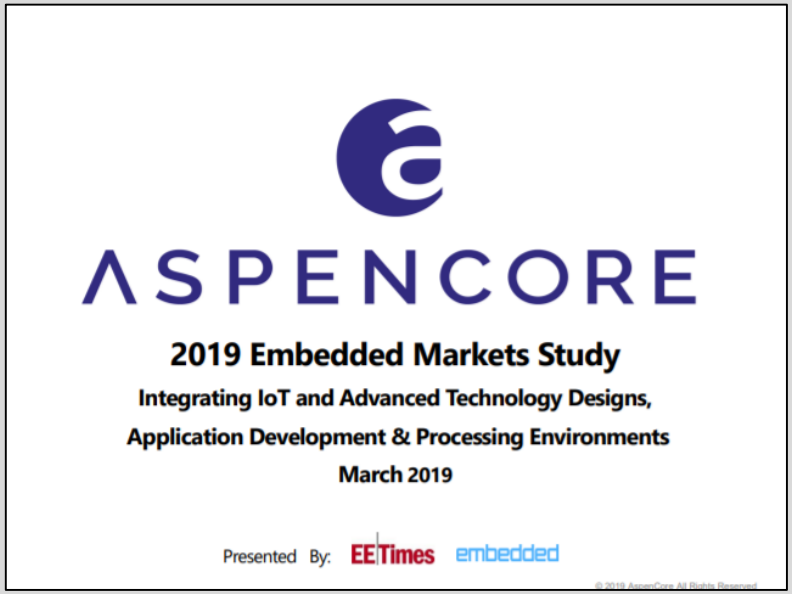
Quiz

- **Quelle est la différence principale entre un système d'exploitation comme MS Windows et un système d'exploitation pour l'embarqué**
  - A. Le système d'exploitation pour l'embarqué nécessite moins de ressources
  - B. Le système d'exploitation pour l'embarqué a une gamme plus large de support matériel
  - C. Le système d'exploitation pour l'embarqué est moins cher
  - D. Le système d'exploitation pour l'embarqué est plus simple et plus petit
- **Correction : A, D**

Système d'exploitation pour l'embarqué (2) – real time

- "En informatique temps réel, le comportement correct d'un système dépend, non seulement des résultats logiques des traitements, mais aussi du temps auquel les résultats sont produits" J. Stankovic

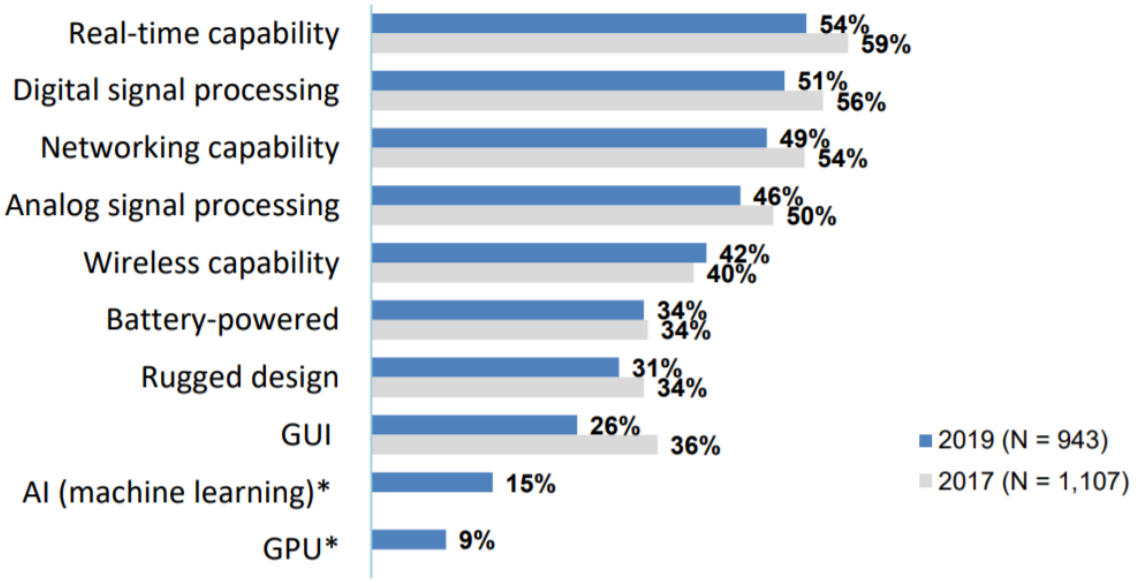
- **Objectifs**
  - **Déterminisme logique** : les mêmes entrées appliquées au système produisent les mêmes résultats
  - **Déterminisme temporel** : respect des contraintes temporelles (ex : échéance)
  - **Fiabilité** : le système répond à des contraintes de disponibilité (fiabilité du logiciel et du matériel)
  - **Système prédictible** : on cherche à déterminer a priori si le système va répondre aux exigences temporelles
  - Un système temps réel (STR) n'est pas un système "qui va vite" mais un système qui satisfait à des contraintes temporelles
- **Exemples**
  - Voiture autonomes, systèmes de contrôle de vol



Lesquelles des fonctionnalités suivantes sont incluses dans votre projet embarqué ?

Système d'exploitation pour l'embarqué (2) – real time

Which of the following capabilities are included in your current embedded project?



*AI and GPU were added in 2019.

EETimes embedded 2019 Embedded Markets Study

Système d'exploitation pour l'embarqué (2) – real time

- **Exemples de grandeur (I. Demeure and C. Bonnet)**
  - La milliseconde pour les systèmes radar
  - La seconde pour les systèmes de visualisation humain
  - Quelques heures pour le contrôle de production impliquant des réactions chimiques
  - 24 heures pour les prévisions météo
  - Plusieurs mois ou années pour les systèmes de navigation de sonde spatiale

Système d'exploitation pour l'embarqué (2) – real time

- **Garantie de service = niveau de respect des contraintes. Garanties *déterministes*, *probabilistes* ou "*best effort*"**
  - **Contrôle de processus sans (ou à faible) contrainte temporelle → systèmes à temps partagé**
    - Garantir le partage équitable du temps et des ressources
  - **Contrôle de processus avec contrainte temps réel → systèmes temps réel**
    - Garantir les temps de réponse
    - ***Systèmes à contraintes souples/molles***: systèmes acceptant des variations minimes de temps de réponse (systèmes multimédias)
    - ***Systèmes à contraintes dures ou critiques***: gestion stricte du temps pour conserver l'intégrité du système (déterminisme logique et temporel et fiabilité)

« Delay »

- ***Définition*: différence entre le moment où une tâche doit débuter (ou finir) et le moment où elle débute réellement.**
- **Elles sont dues**
  - **Aux propriétés temporelles des processeurs, des bus mémoire et d'autres périphériques**
  - **Aux propriétés des politiques d'ordonnancement**
  - **À la *préemptibilité* du noyau**
  - **À la charge du système**
  - **Au changement de contexte**

« Delay »

- **Tâches introduisant de « *l'indeterminisme* » temporel**
  - **Accès disque : technologie mécanique, géométrie différente d'un disque à l'autre**
  - **Accès au réseau : retransmissions en cas d'erreur**
  - **Résolution basse du *timer***
  - **Pilotes de périphérique non temps réel : utilisation d'attente active et de période de sommeil peu précise**
  - **Allocation et gestion de la mémoire : mémoire virtuelle /swap non prédictible**
  - **Système de fichiers virtuel/*proc* : tout ce qui se passe dans le système → création à la volée**

Récapitulatif

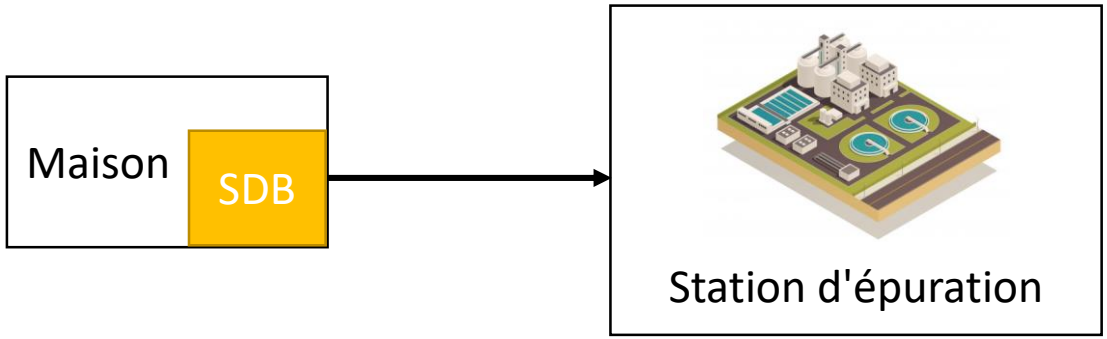
Critères	Temps partagé	Temps réel
But	Maximiser la capacité de traitement (débit) & utilisation des ressources	Etre prévisible (garantir les temps de réponse)
Temps de réponse	Bon en moyenne	Bon dans le pire des cas / moyenne non importante
Comportement à la charge	Confortable à l'utilisateur	Stabilité et respect des contraintes de temps

Plan

1. **Qu’est ce qu’un système embarqué**
2. **Deux types de performance : Latence vs Débit**
3. **AspenCore Embedded Markets Study**
4. **Les architectures des systèmes d’exploitation**

Deux types de performance

- **Latence (Latency) vs Débit (Throughput)**
  - **Exemple**



  - **La station peut traiter 4 baignoires d'eau / second**
  - **Combien de temps pour traiter l'eau sale de ma baignoire ?**
    - A : ¼ second
    - B : 4 seconds
    - C : 24 heures
    - D : Besoin plus d'information

Le CM est inspiré du cours dispensé par David Black-Schaffer

Deux types de performance

- **Problème avec la question**
  - **La question concerne la latence**
  - **4 baignoire d'eau / second → Débit**
- **Réponse : 24 heures**
  - **Pourquoi ?**
    - A : Très rapide
    - B : Très lente
    - C : 345,600 en parallèle
    - D : Pas possible
  - **Réponse : 345,600 en parallèle**

Définition

- **Latence**
  - **Combien de temps (le "delay") il faut pour livrer un objet.**
  - **Temps absolu pour chaque réponse**
    - 24 h pour traiter une baignoire
  - **La rapidité de chaque réponse est importante**
- **Débit**
  - **Combien d'objets peut être livré sur une période de temps**
  - **Temps moyen pour chaque réponse**
    - 1 baignoire / ¼ second
  - **La réponse globale par temps est importante**

- **La station d'épuration : quantité d'eau qu'ils peuvent traiter pour toute la ville.**

Exemple : Vérification orthographique

- **2 solutions**
  - **A : 10,000 mots vérifiés dans 10s**
  - **B : 10 mots vérifiés dans 0,1s**
- **Latence (à partir du moment vous appuyez "Start")**
  - **A : 10s pour trouver tous les mots mal orthographiés**
  - **B : 0,1s pour trouver les premiers mots mal orthographiés**
- **Débit**
  - **A : 1000 mots/s (avg)**
  - **B : 100 mots/s (avg)**
- **Meilleure solution ?**
  - **Pensez à utiliser MS Word, voulez-vous attendre 10 secondes pour que la vérification orthographique démarre**

Exemple : Annulation d'écho

- **Ce qui est mieux ?**
  - A : 1000ms d'audio traité en 100ms
  - B : 10ms d'audio traité en 5ms
  - C : Besoin plus d'information
- **Latence**
  - A : 100ms pour le premier son
  - B : 5ms pour le premier son → Real-time chat, Skype
- **Débit**
  - A : 10x real-time → Montage vidéo ou film
  - B : 2x real-time

Latence (Latency) vs Débit (Throughput)

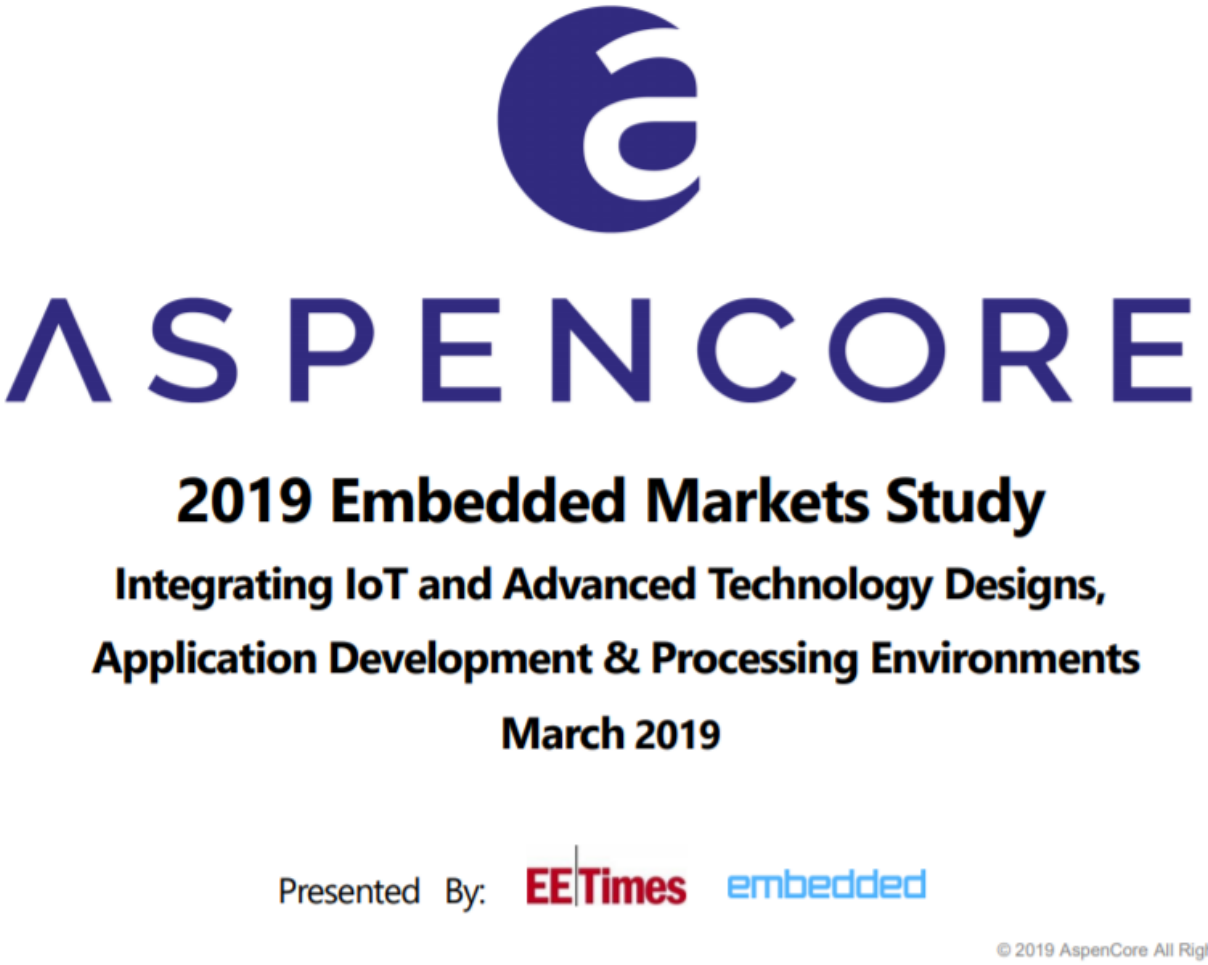
- **Dépend de ce que vous faites**
  - **Traitements par lots (pixels, audio/video processing) : Débit**
  - **Actions individuelles (clicks, real-time audio) : Latence**
- **Pourquoi nous devons savoir ça ?**
  - **Historiquement, tous les processeurs ont été focalisés sur la latence (performance = temps du premier résultat) → ++ MHz**
  - **Aujourd'hui, les processeurs se concentrent sur le débit (performance = résultats/time) → ++ Cœurs**
- **Vous voulez que votre code s'exécute vite**
  - **Besoin de comprendre la différence**

Plan

1. Qu’est ce qu’un système embarqué
2. Deux types de performance : Latence vs Débit
3. AspenCore Embedded Markets Study
4. Les architectures des systèmes d’exploitation

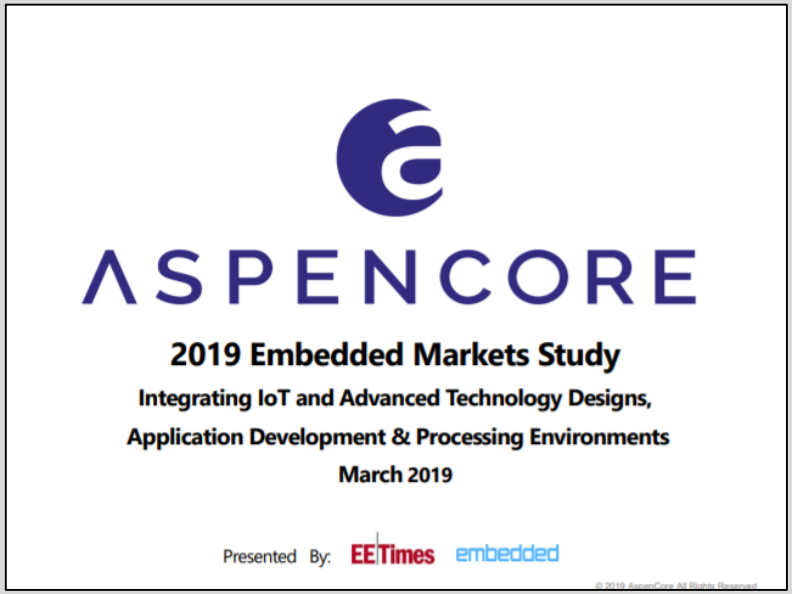
Quelques chiffres sur le domaine de l'embarqué

https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf



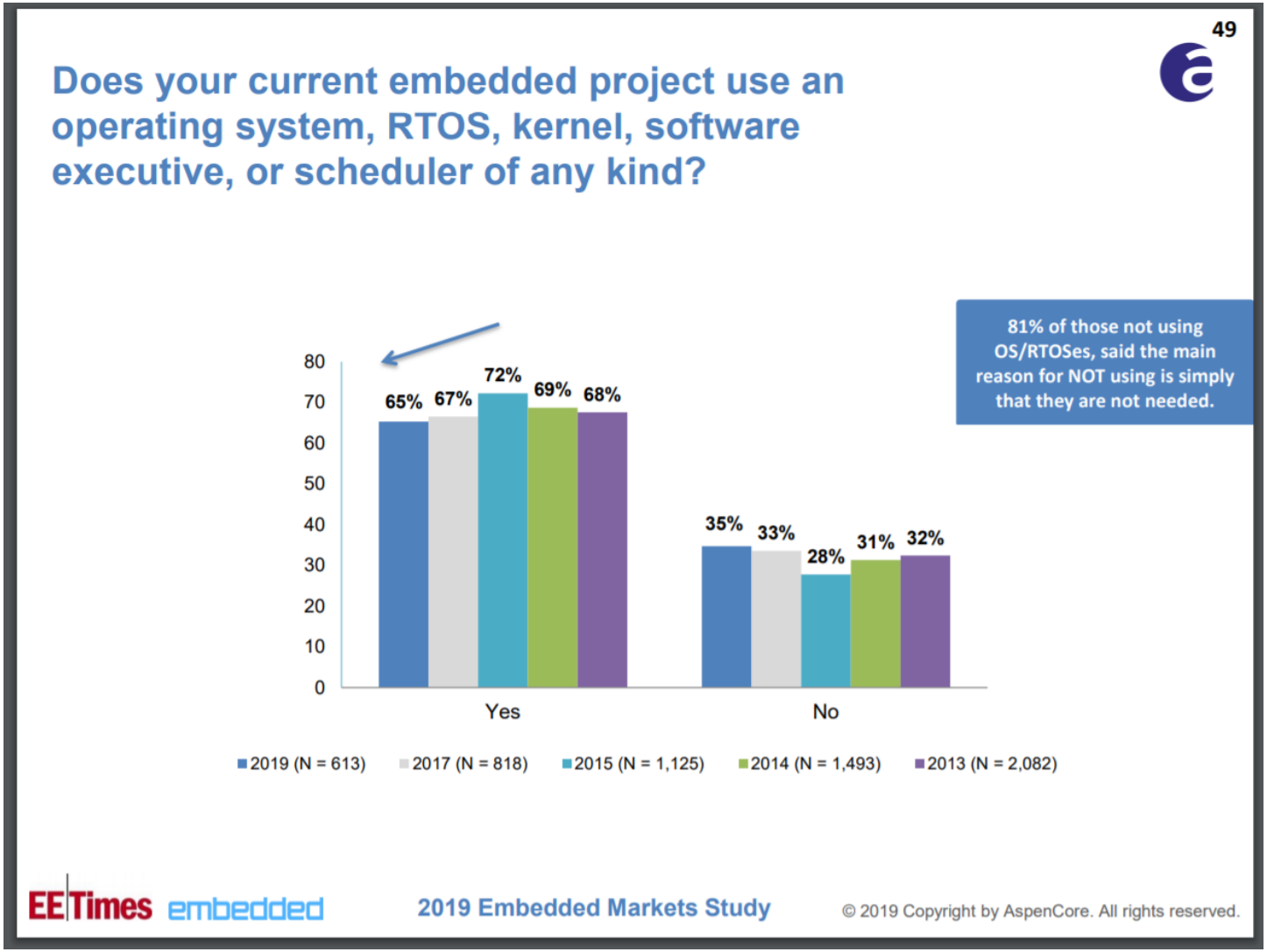
Embedded Markets Study

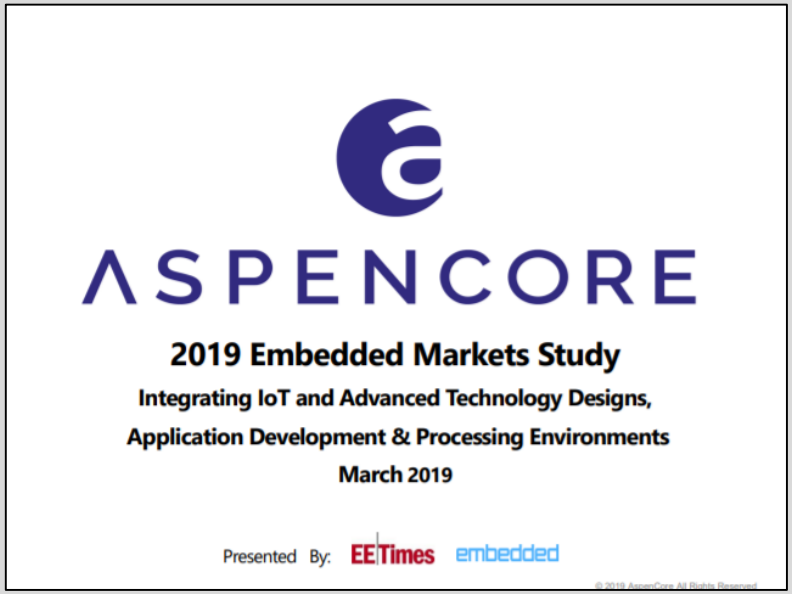
- **Enquête faite par AspenCore, EETimes and Embedded.com**
  - **Objective : présenter les résultats de l'étude complète sur les marchés embarqués dans le monde**
  - **Participants : abonnés EETimes et Embedded.com et entreprises liées à AspenCore**
    - America
    - Asia Pacific (APAC)
    - Europe, the Middle East and Africa (EMEA)
  - **Les données de cette étude sont projetables à 95% de confiance avec un intervalle de confiance de +/- 3,15%**



Votre projet actuel utilise-t-il un RTOS, un noyau, ou un ordonnanceur ?

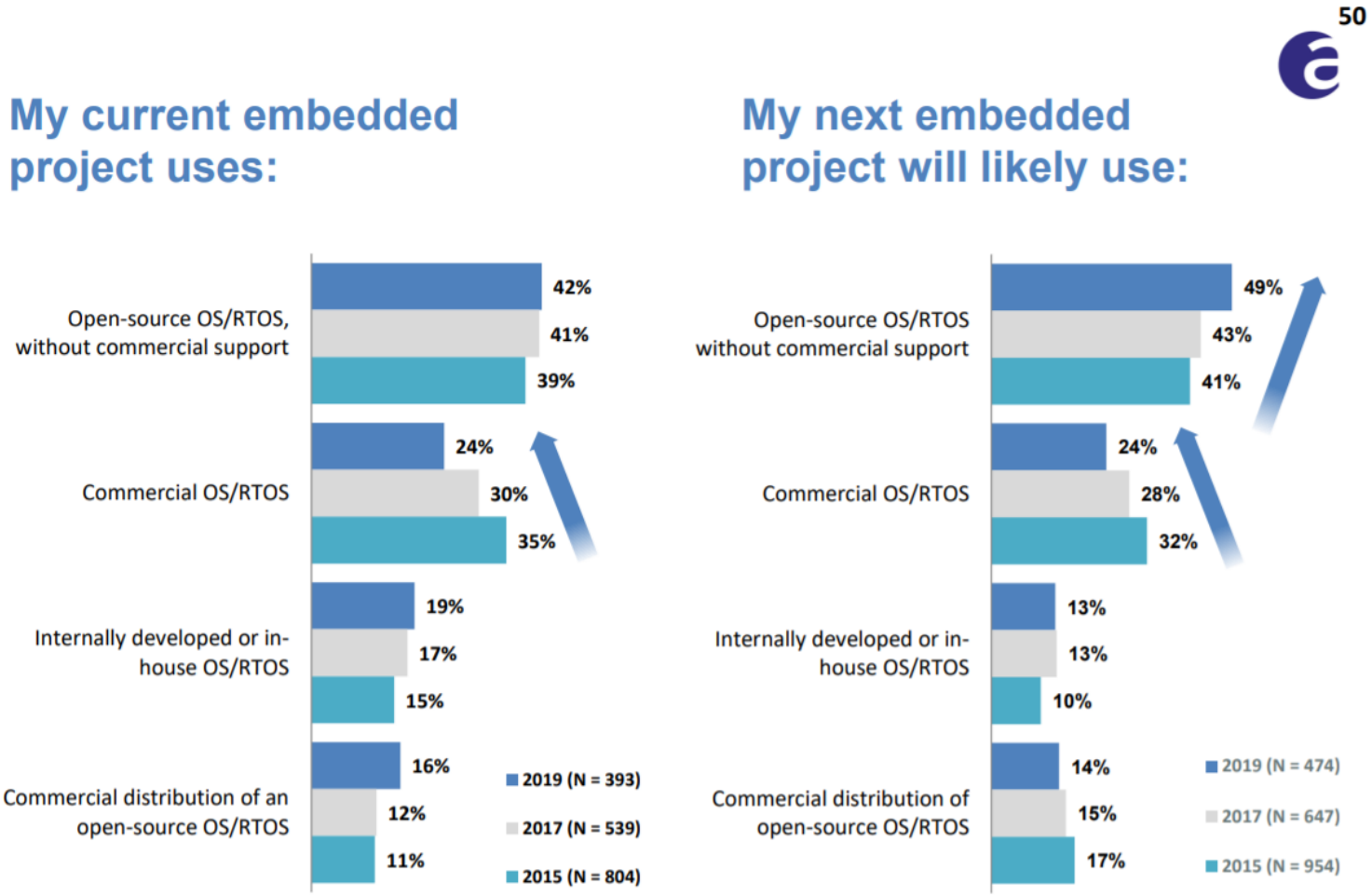
Y a un « OS » ?


49
Does your current embedded project use an operating system, RTOS, kernel, software executive, or scheduler of any kind?
81% of those not using OS/RTOSes, said the main reason for NOT using is simply that they are not needed.
80
70
60
50
40
30
20
10
0
65%
67%
72%
69%
68%
35%
33%
28%
31%
32%
Yes
No
2019 (N = 613)
2017 (N = 818)
2015 (N = 1,125)
2014 (N = 1,493)
2013 (N = 2,082)
EETimes embedded
2019 Embedded Markets Study
© 2019 Copyright by AspenCore. All rights reserved.

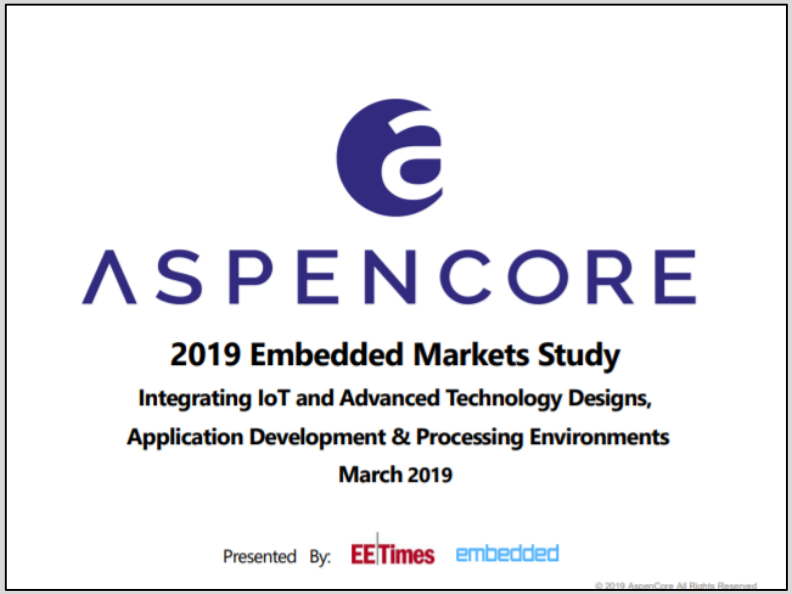

ASPENCORE
2019 Embedded Markets Study
Integrating IoT and Advanced Technology Designs,
Application Development & Processing Environments
March 2019
Presented By: EETimes embedded

Quel «OS» ?

Quel « OS » ?



EETimes embedded 2019 Embedded Markets Study

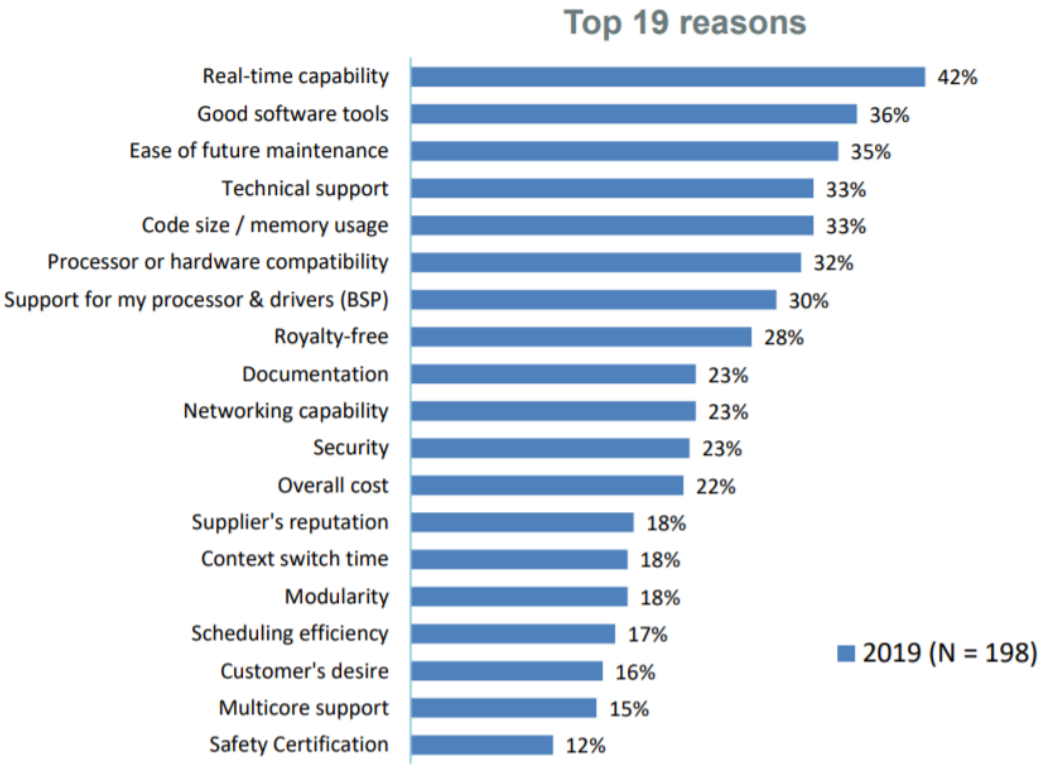


Quels facteurs ont le plus influencé votre décision d'utiliser un système d'exploitation commercial ?

Quel « OS » ?

51

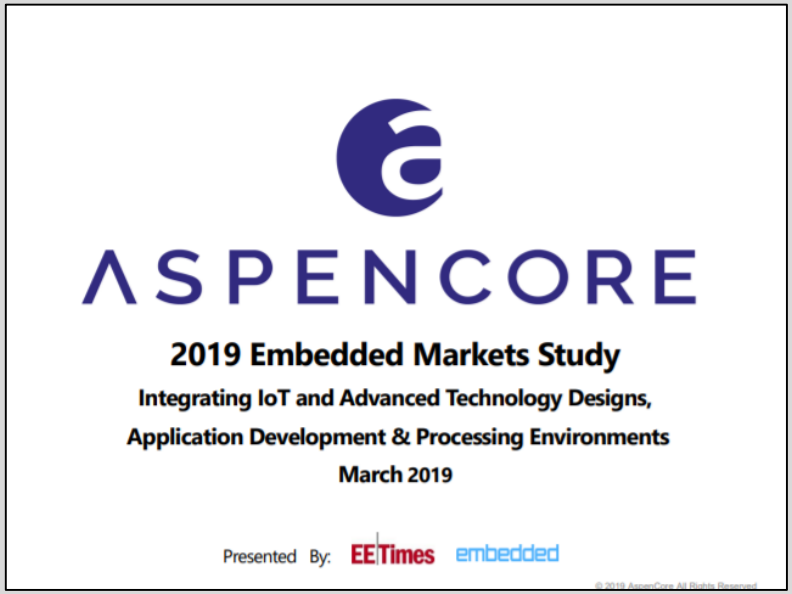
Which factors most influenced your decision to use a commercial operating system?



2019

Base = Those who currently use a "Commercial" OS/RTOS

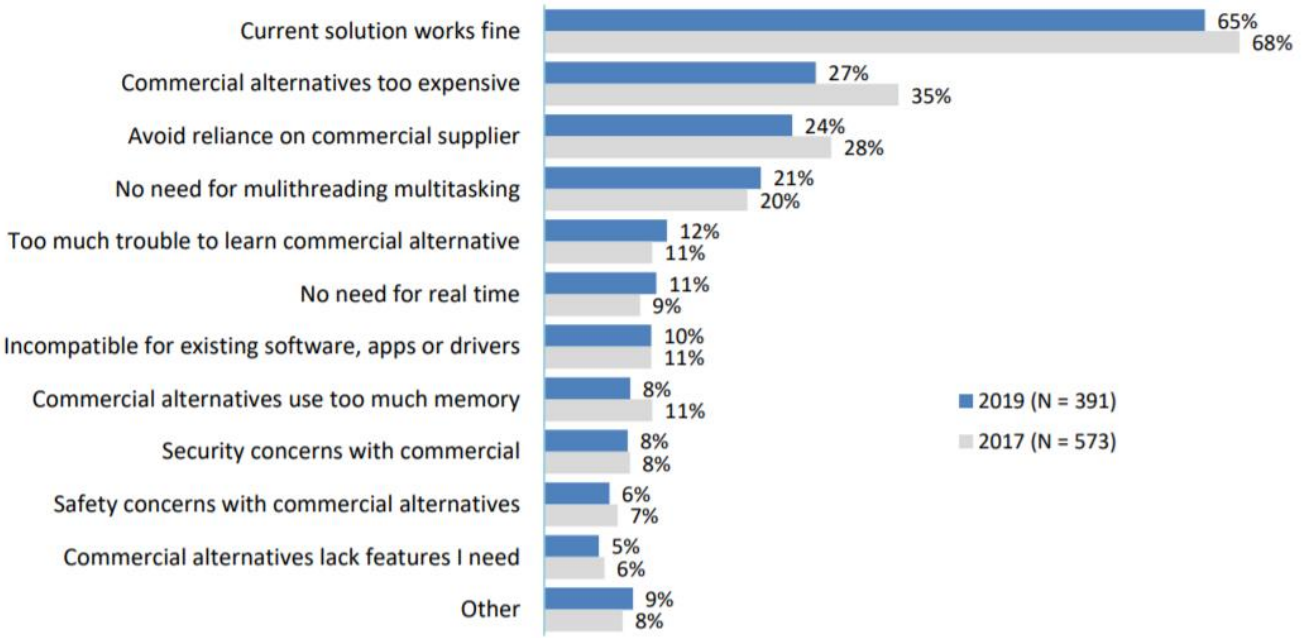
EETimes embedded 2019 Embedded Markets Study



Quelles sont vos raisons pour ne pas utiliser un système d'exploitation commercial ?

Quel « OS » ?

What are your reasons for not using a commercial operating system?

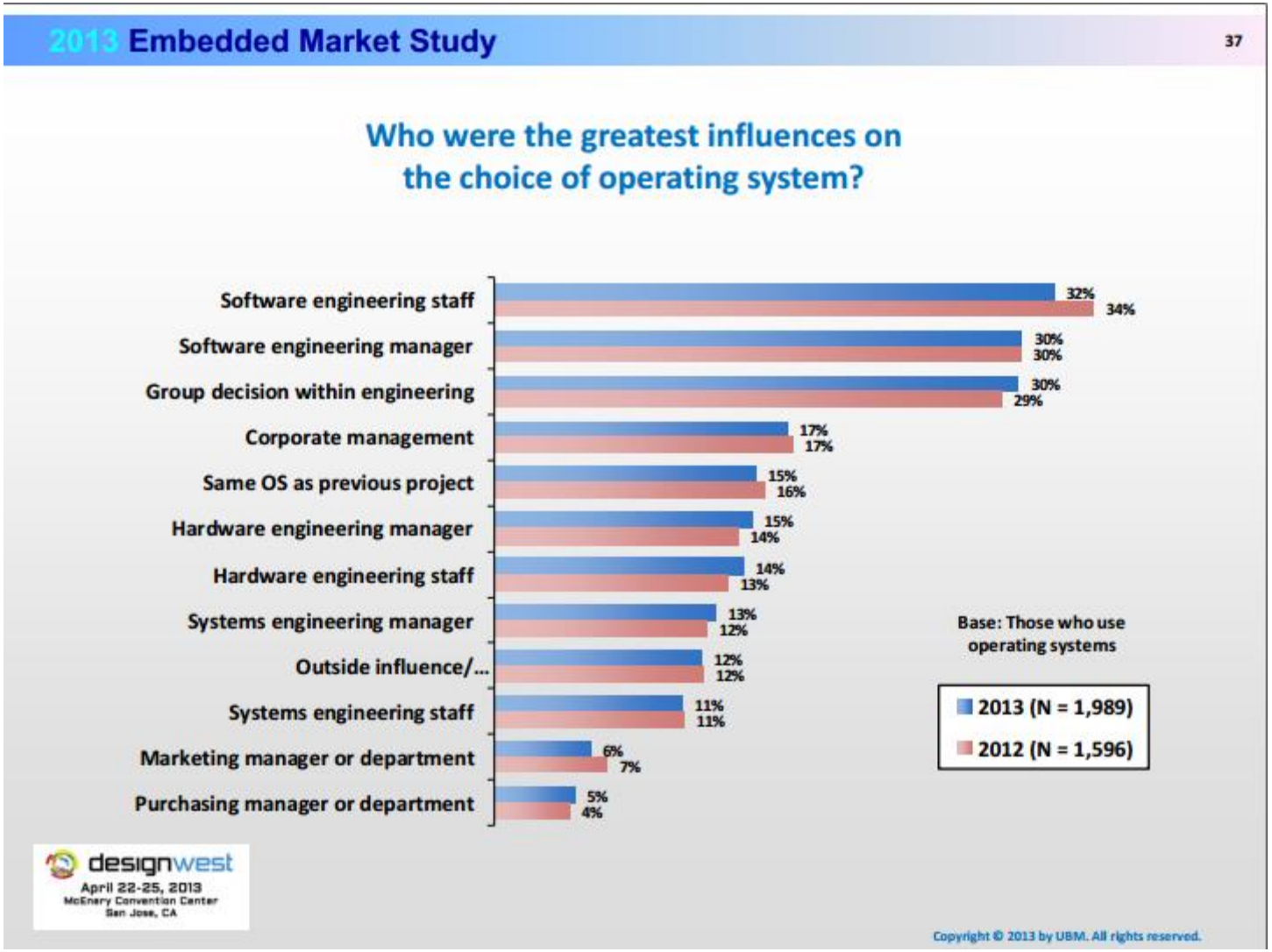


Base = Those who do not currently use a "Commercial" OS/RTOS

EETimes embedded 2019 Embedded Markets Study

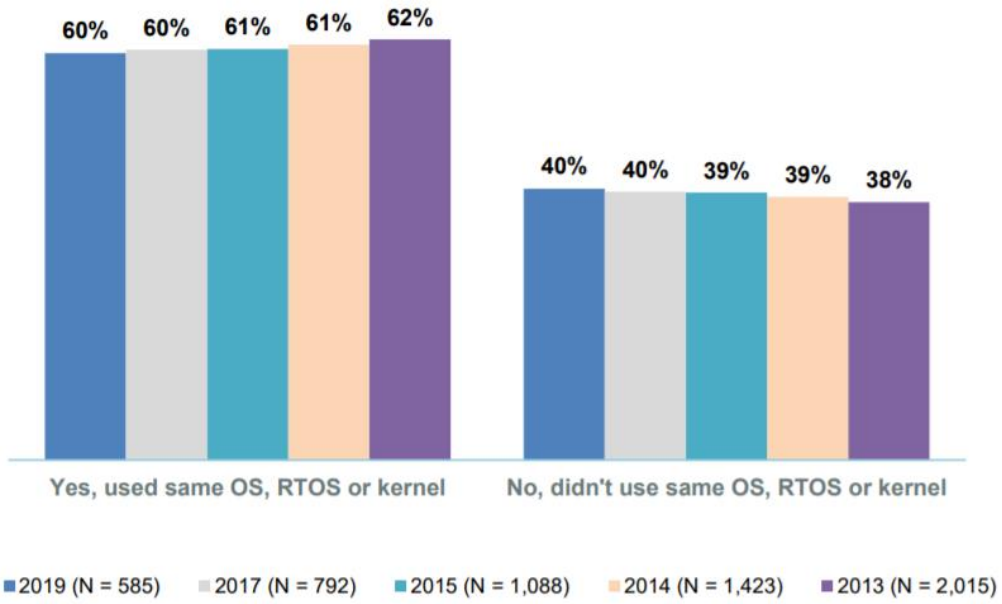
Qui ont eu le plus d'influence sur le choix du système d'exploitation ?

Quel « OS » ?


2013 Embedded Market Study
37
Who were the greatest influences on the choice of operating system?
Software engineering staff
32%
34%
Software engineering manager
30%
30%
Group decision within engineering
30%
29%
Corporate management
17%
17%
Same OS as previous project
15%
16%
Hardware engineering manager
15%
14%
Hardware engineering staff
14%
13%
Systems engineering manager
13%
12%
Outside influence/...
12%
12%
Systems engineering staff
11%
11%
Marketing manager or department
6%
7%
Purchasing manager or department
5%
4%
Base: Those who use operating systems
2013 (N = 1,989)
2012 (N = 1,596)
designwest
April 22-25, 2013
McEnery Convention Center
San Jose, CA
Copyright © 2013 by UBM. All rights reserved.

Quel « OS » ?

Did you use the same operating system, RTOS, or kernel as in your previous project?



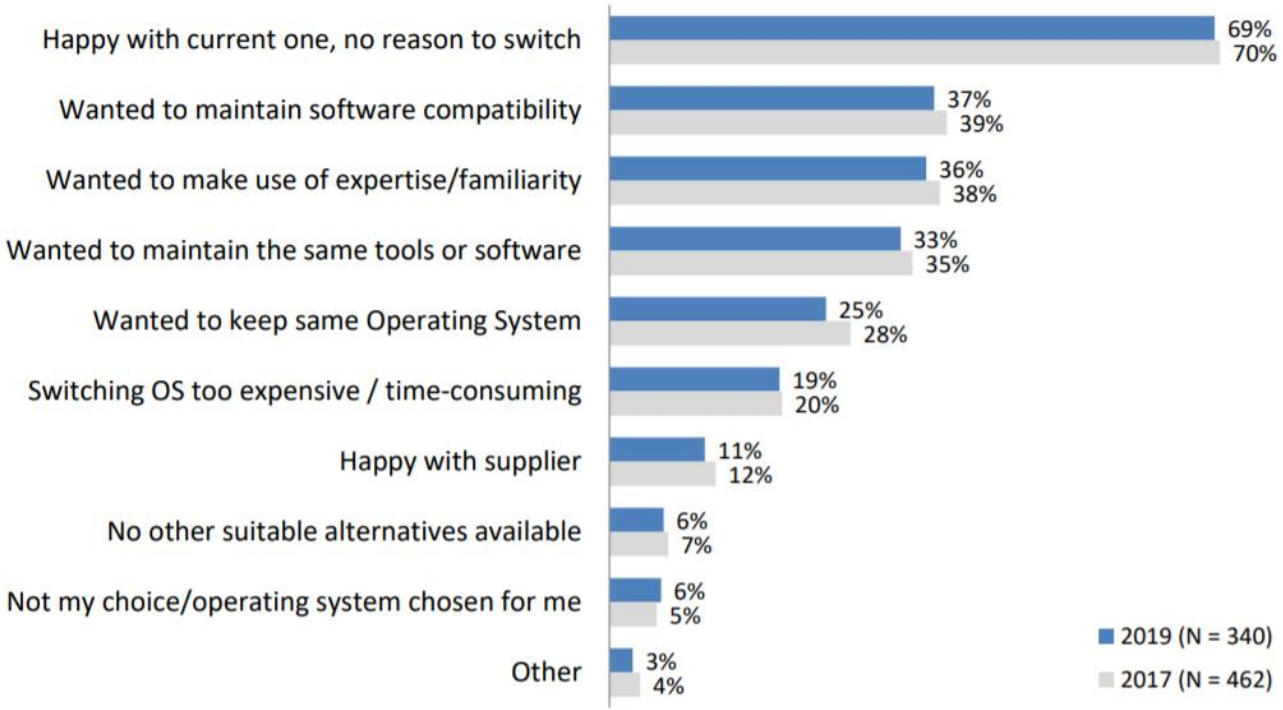
Base: Those who use operating systems

EETimes embedded

2019 Embedded Markets Study

Quel « OS » ?

Why did you use the same operating system?



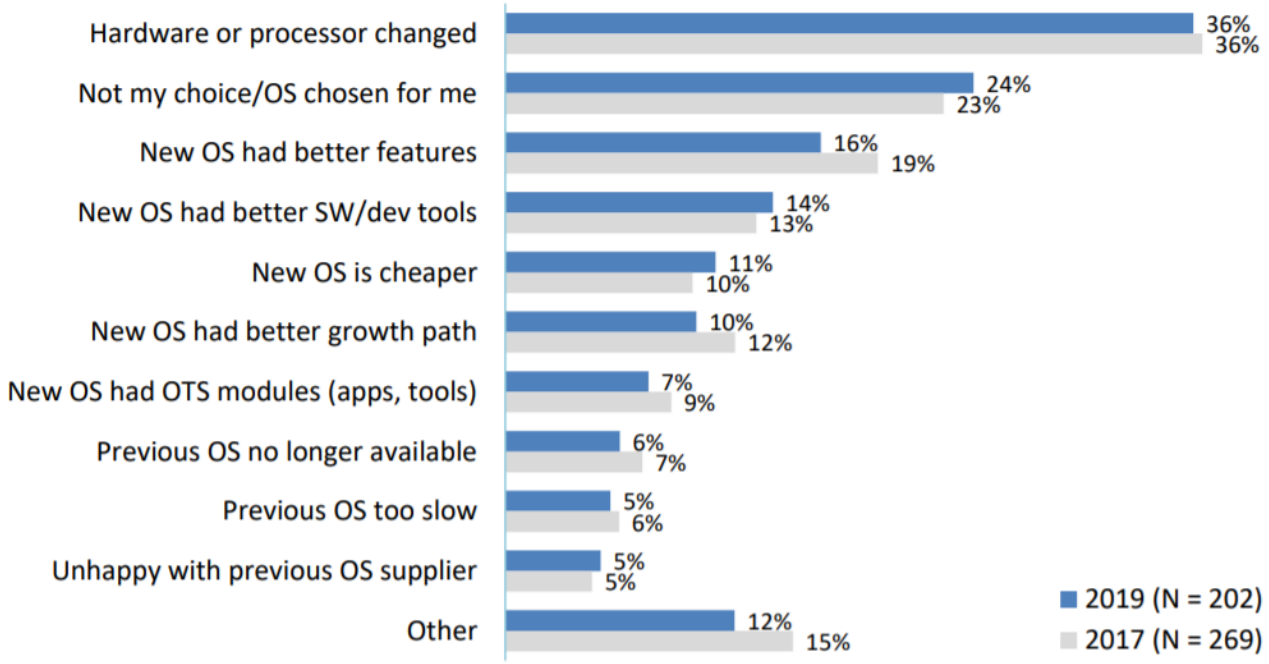
Base = Those who are using the same operating system as in previous project

EETimes embedded 2019 Embedded Markets Study

Quel « OS » ?

55

Why did you switch operating systems?

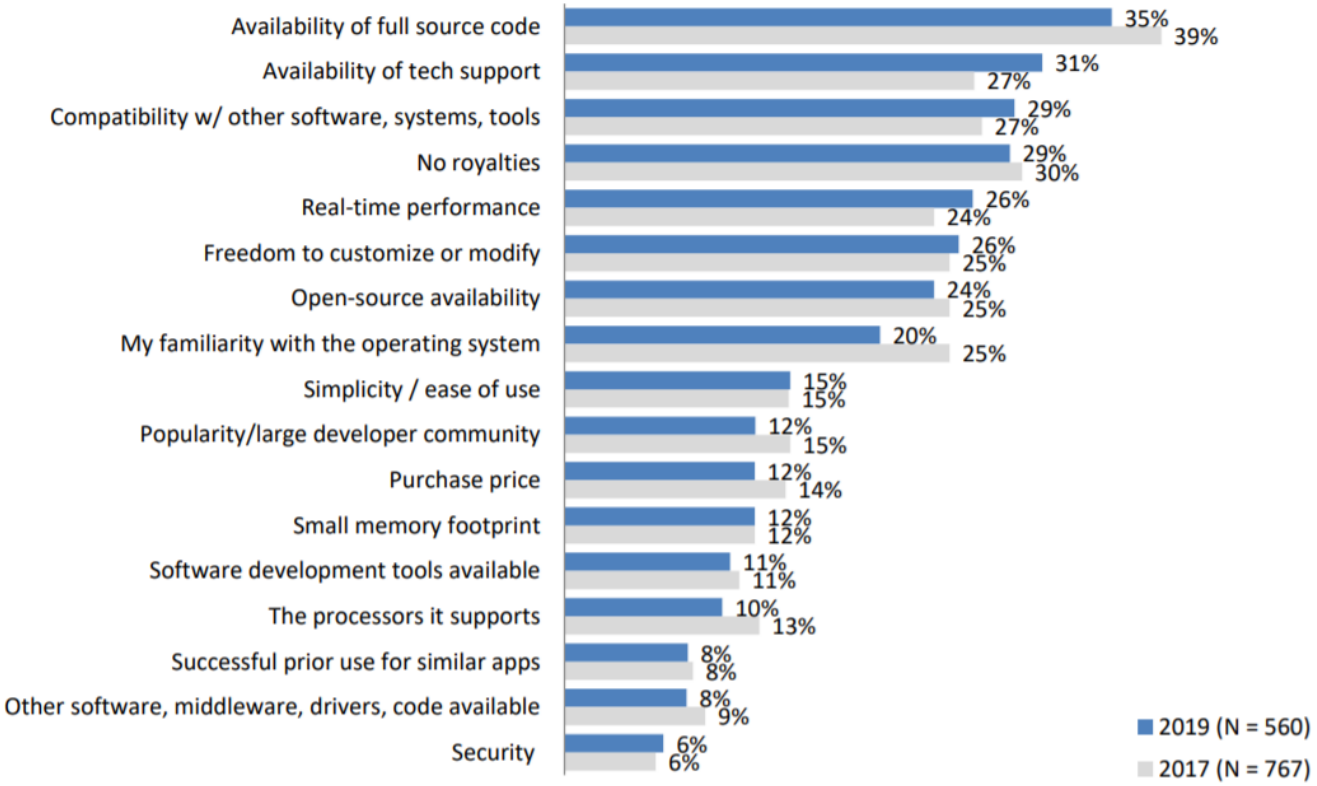


EETimes embedded 2019 Embedded Markets Study

Quel « OS » ?

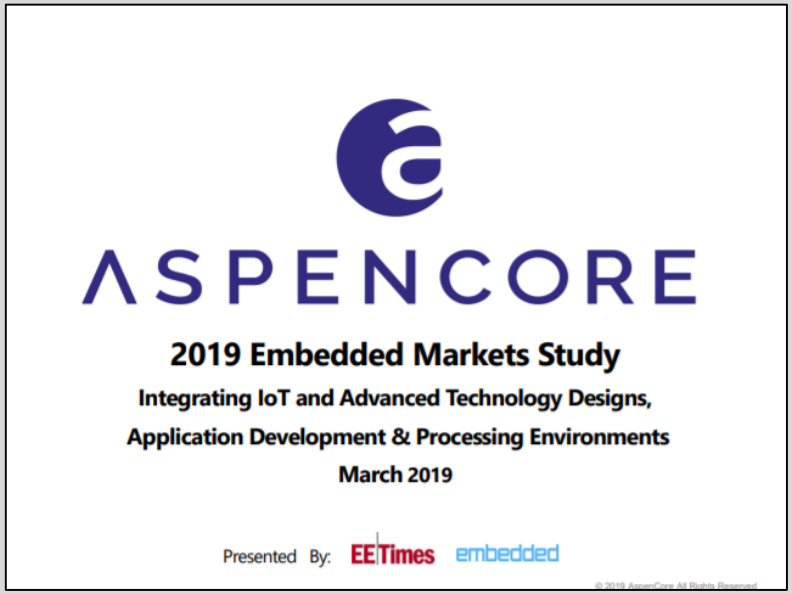
56

What are the most important factors in choosing an operating system?



Base: Currently using an operating system

EETimes embedded 2019 Embedded Markets Study



Veuillez sélectionner tous les systèmes d'exploitation que vous utilisez actuellement

Idée reçue numéro 02

- Il doit y avoir peu d'entreprises développant des systèmes d'exploitation pour l'embarqué qui se partagent le marché dont ANDROID!, c'est sûr !

Tendance des (RT)OS … (2006)

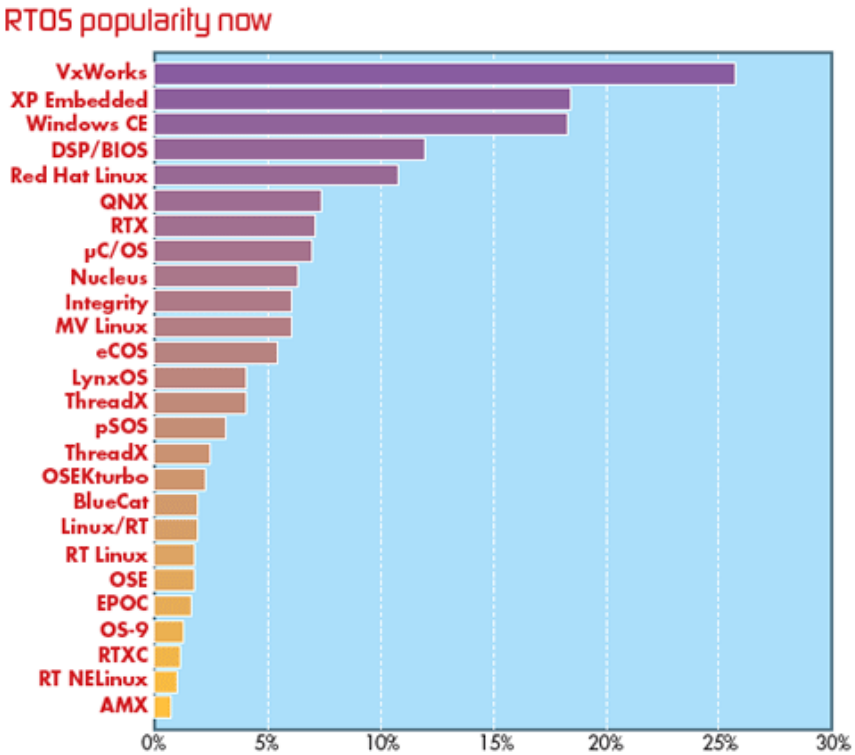


Figure 6

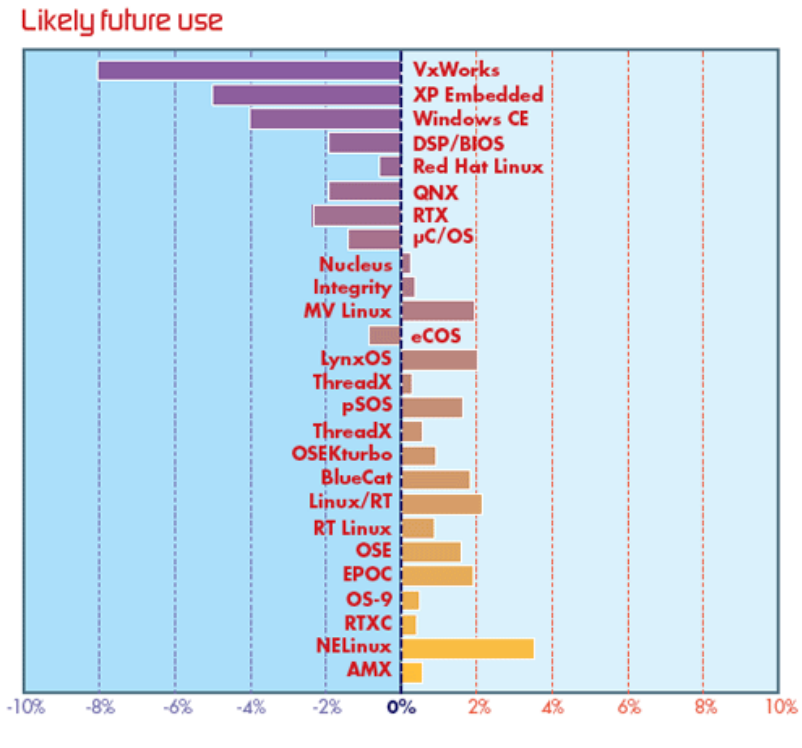
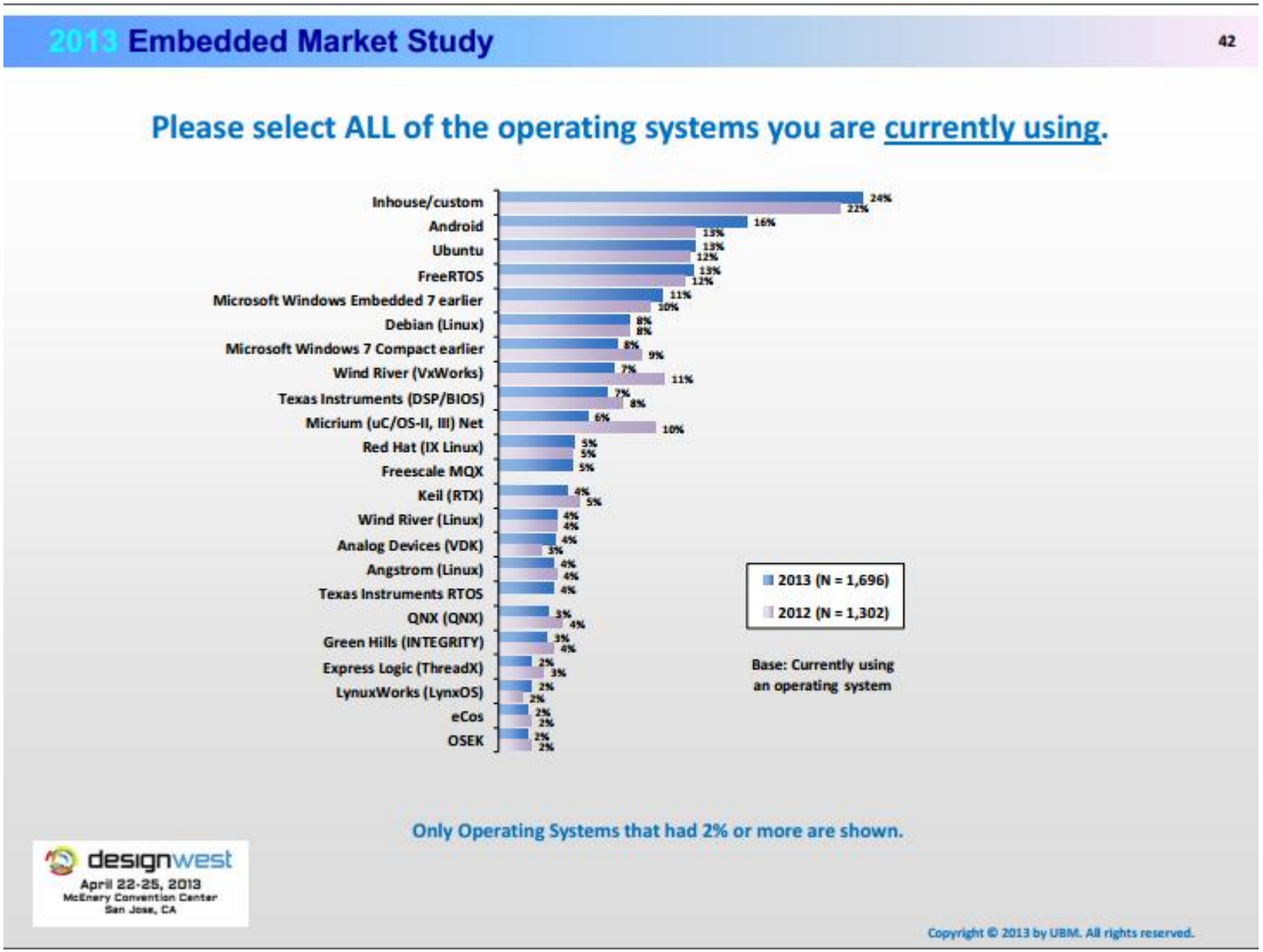


Figure 7

Source: Operating systems on the rise, Jim Turley
Embedded Systems Design
(06/21/06, 09:00:00 AM EDT)
www.eetimes.com/discussion/other/4025674/Operating-systems-on-the-rise

Android 1.0, September 23, 2008

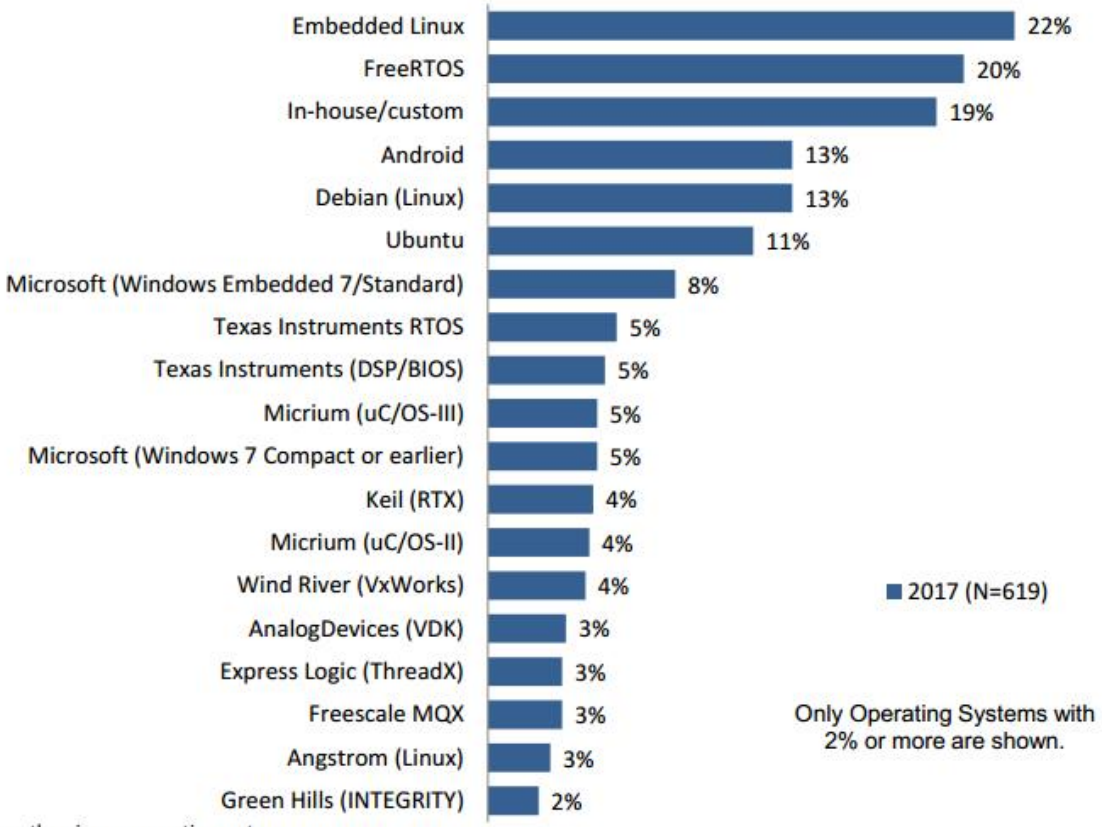
Tendance des (RT)OS … (2013)



Tendance des (RT)OS … (2017)

ASPENCORE

Please select ALL of the operating systems you are currently using.



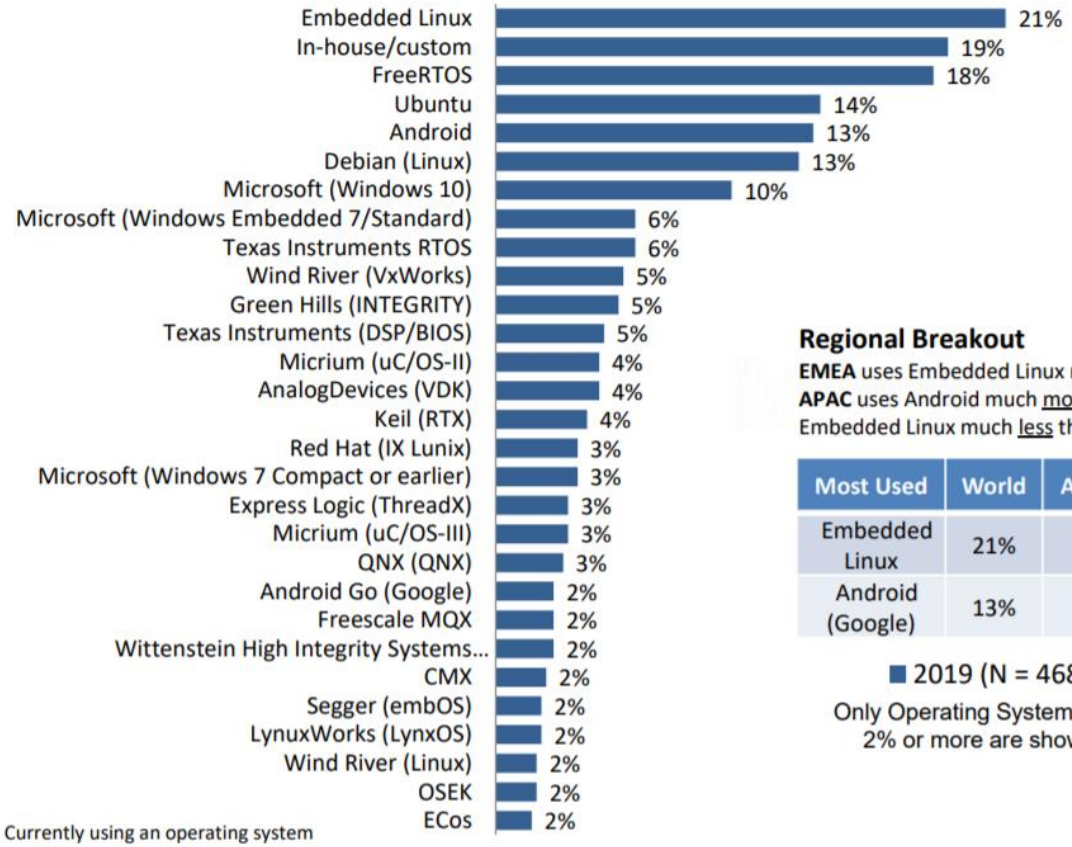
Base: Currently using an operating system

EETimes embedded 2017 Embedded Markets Study

Tendance des (RT)OS … (2019)

57

Please select ALL of the operating systems you are currently using.



Base: Currently using an operating system

Regional Breakout

EMEA uses Embedded Linux much more than other regions.
APAC uses Android much more than other regions and uses Embedded Linux much less that others.

Most Used	World	Americas	EMEA	APAC
Embedded Linux	21%	21%	30%	15%
Android (Google)	13%	9%	14%	27%

2019 (N = 468)

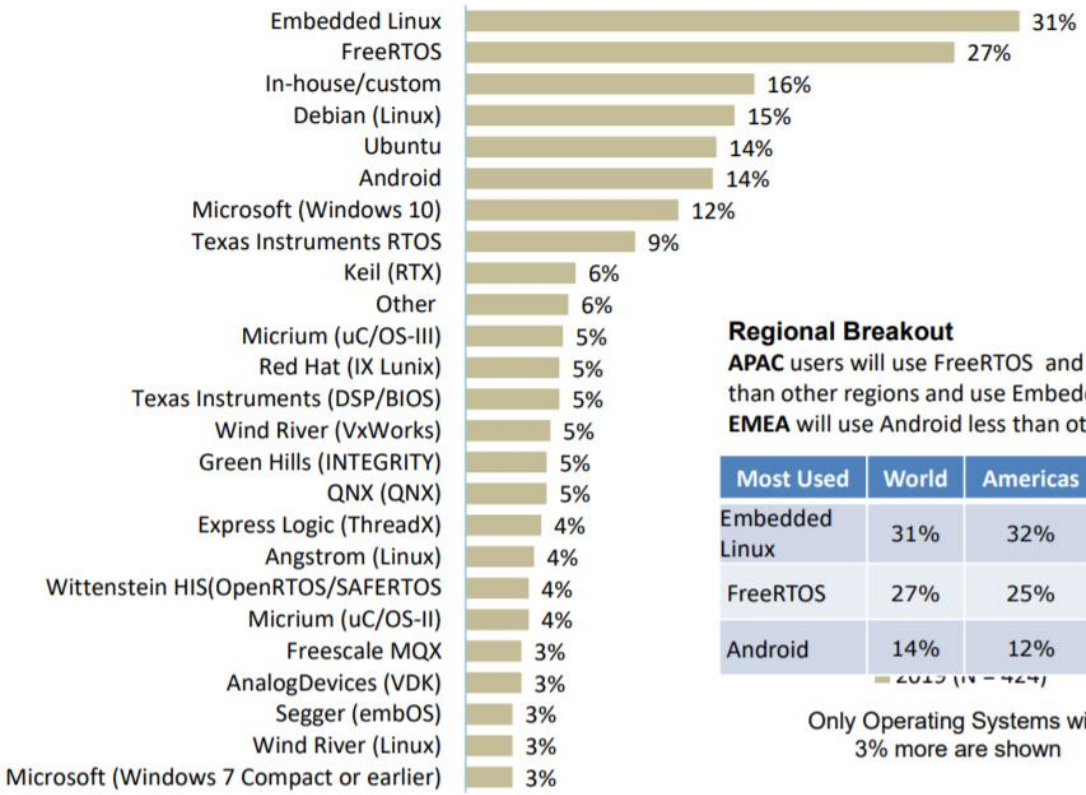
Only Operating Systems with 2% or more are shown.

EETimes embedded 2019 Embedded Markets Study

Tendance des (RT)OS … (2019)

58

Please select ALL of the operating systems you are considering using in the next 12 months.



Regional Breakout

APAC users will use FreeRTOS and Android much more than other regions and use Embedded Linux much less.
EMEA will use Android less than other regions.

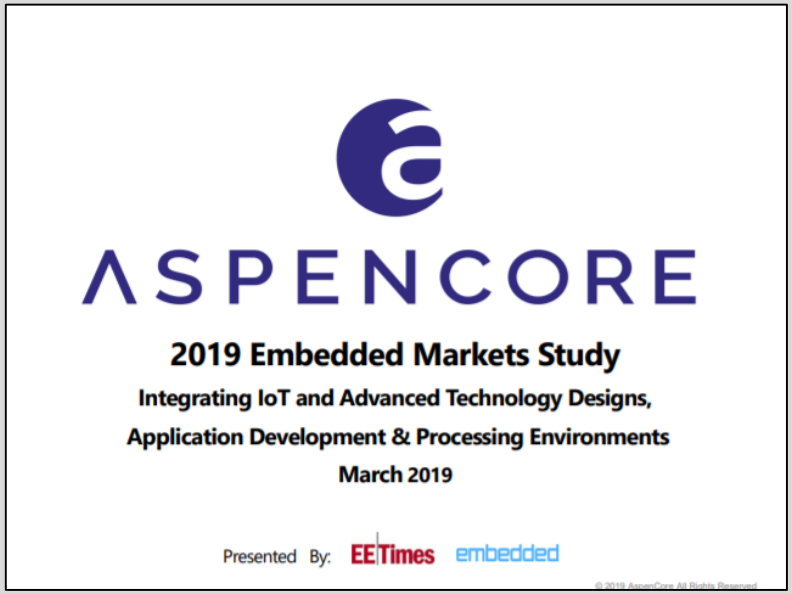
Most Used	World	Americas	EMEA	APAC
Embedded Linux	31%	32%	31%	26%
FreeRTOS	27%	25%	24%	37%
Android	14%	12%	10%	26%

2019 (N = 424)

Only Operating Systems with 3% more are shown

Base: Those who are considering an operating system in any project in the next 12 months

EETimes embedded 2019 Embedded Markets Study

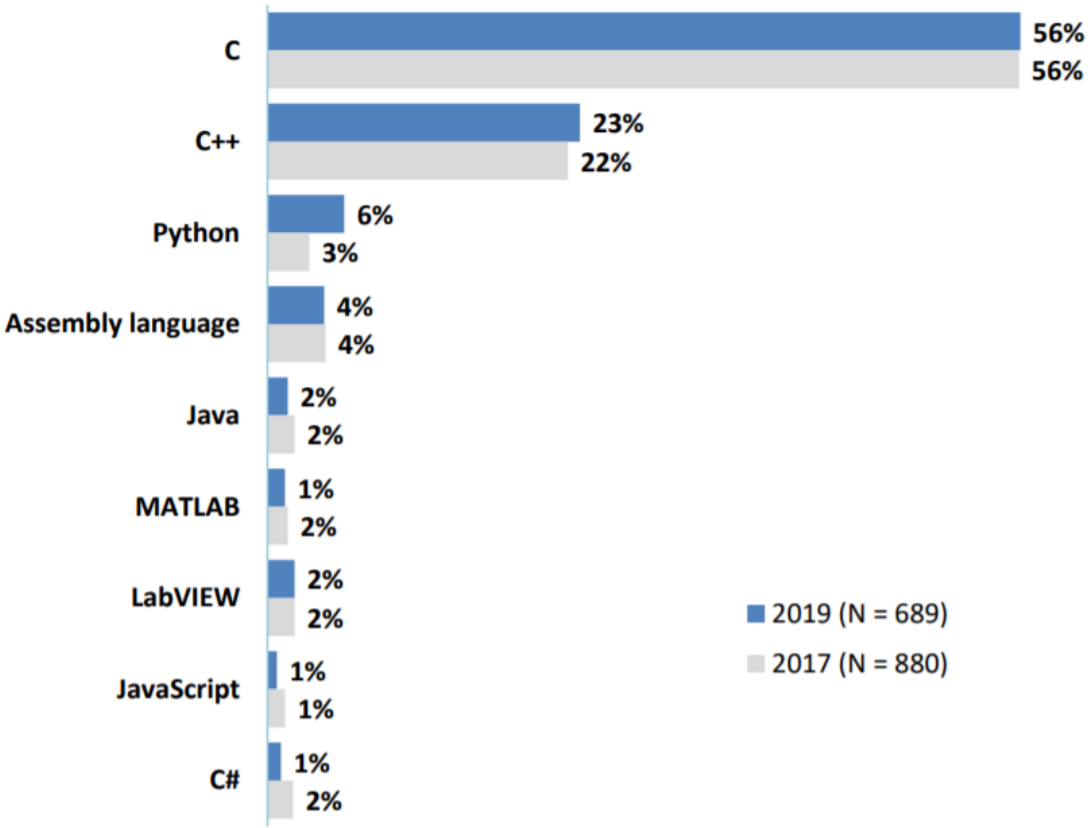


Quel est le langage de programmation de votre projet embarqué actuel ?

Idée reçue numéro 03

- Il faut apprendre Python et Java, … le C ? pfff c'est mort ! L'assembleur !! … c'est pour les musées, ça !!!

My *current* embedded project is programmed mostly in:



EETimes embedded 2019 Embedded Markets Study

Conclusion

- **OS/RTOS usage : 65%**
- **OS/RTOS utilisés**
  - **Embedded Linux (21%), Inhouse (19%), FreeRTOS (18%).**
    - Europe, the Middle East and Africa - Embedded Linux (30%)
    - Asia Pacific - Android (27%)
- **OS/RTOS considérant**
  - **Embedded Linux (31%), FreeRTOS (27%), Inhouse (16%)**
- **OS embarqués : très utilisés, très critiques, aspects, temps réel**

Plan

1. Qu’est ce qu’un système embarqué
2. Deux types de performance : Latence vs Débit
3. AspenCore Embedded Markets Study
4. Les architectures des systèmes d’exploitation

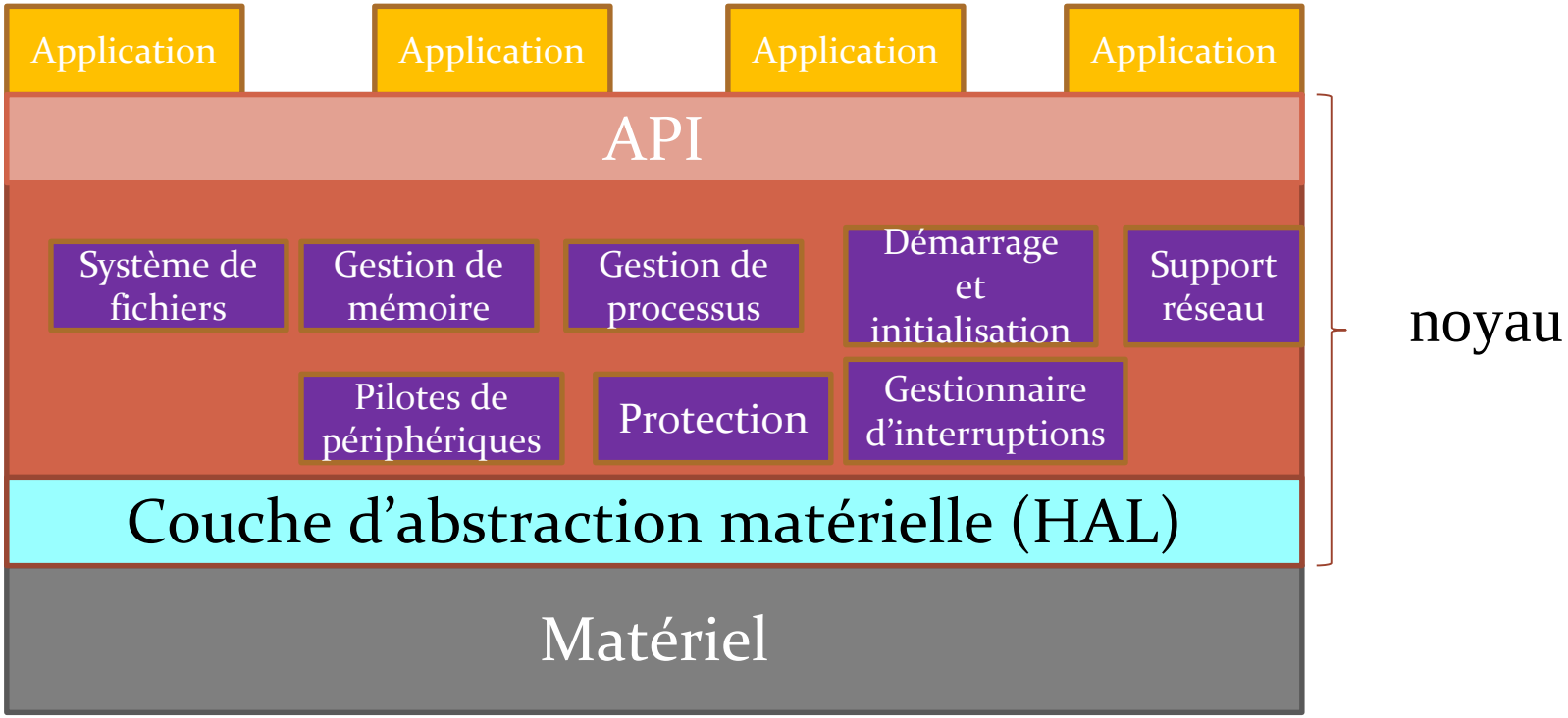
Les architectures des systèmes d'exploitation

- No OS (bare metal)
- OS Monolithique
  - FreeBSD, Linux, SunOS
- OS Micronoyau
  - Free RTOS, PikeOS, QNX
- OS Hybride
  - OS X, Windows NT kernel (MS Windows NT, 2000 → 10)
- Machine virtuelle

Les architectures des systèmes d'exploitation

- **OS Monolithique (plus ancien):**
  - **Simple/ne consomme pas beaucoup de ressources**
  - **Convient aux « petits systèmes » ou quelques portions de systèmes temps réel complexe**
  - **OS → entièrement en mode privilégié**
  - **L'application utilise un appel système pour accéder aux services de l'OS → procédure exécutée**
  - **Impossible de mettre à jour l'application « à chaud » (remplacement + reboot)**

OS Monolithique (exemple: UNIX)



OS Monolithique

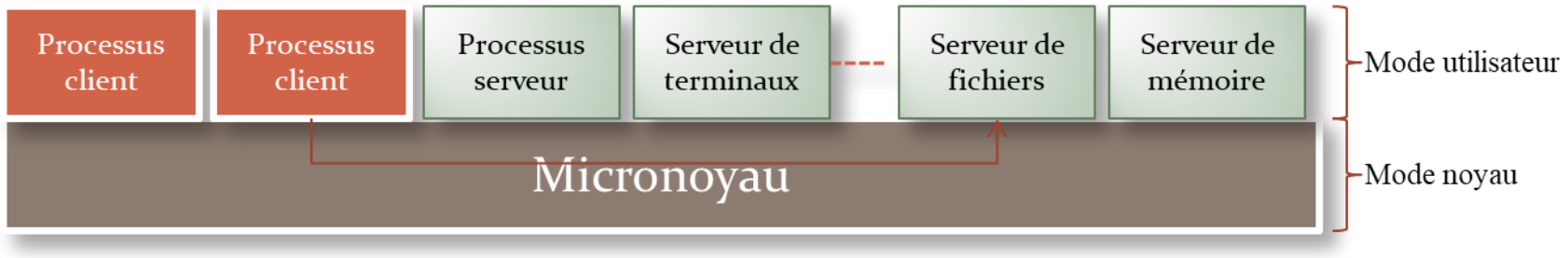
- **Avantages**
  - **De meilleures performances**
  - **Vite développé**
  - **Évolution: chargement dynamique (et donc sélectif) des modules**
- **Inconvénients**
  - **Extension difficile**
    - Code non modulaire
  - **Très complexe**
  - **Code massif**
  - **Plus c'est gros, moins c'est performant !**
  - **Nid de bugs**
  - **Peu fiable (un bug → redémarrage)**
  - **Premières versions à chargement statique → 400 périphériques supportés → 400 périphériques chargés au démarrage !!**

Les architectures des systèmes d'exploitation

- **OS Micronoyau**
  - **Déplace plusieurs fonctions de l'OS vers des « processus serveurs » s'exécutant en mode utilisateur → réduction au maximum de la taille du code privilégié.**
  - **Gérer les communications entre applications et serveurs pour**
    - Renforcer la politique de sécurité
    - Permettre l'exécution de fonctions système (accès aux registres d'E/S, etc.).
  - **Fiabilité augmentée : si un processus serveur « crash », le système continue à fonctionner et il est possible de relancer ce service sans redémarrer**
  - **Modèle facilement étendu à des systèmes distribués (efficacité?).**

Les architectures des systèmes d'exploitation

- **OS Micronoyau**

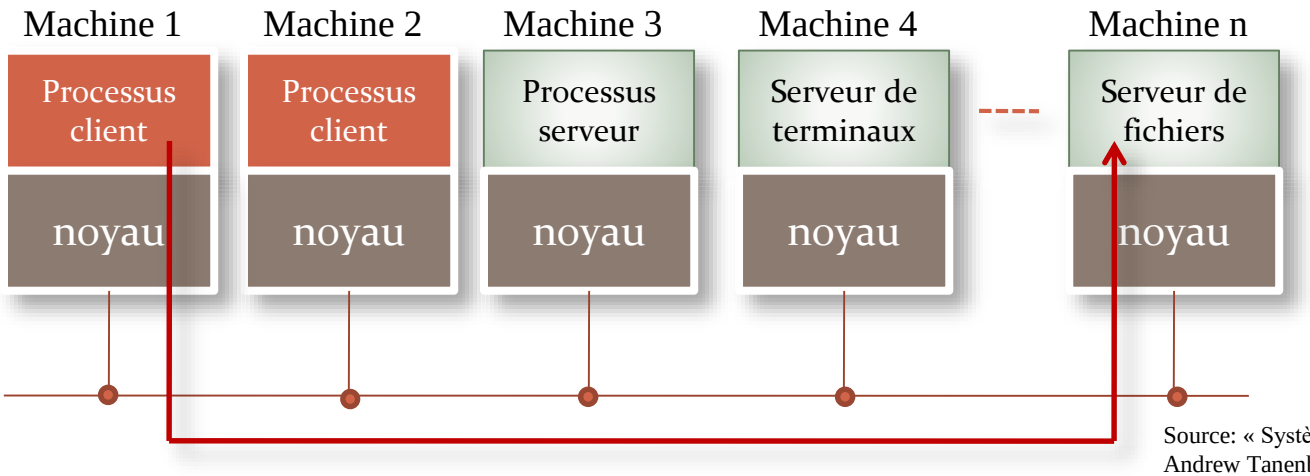


Source: « Systèmes d'exploitation », Andrew Tanenbaum, 2ème édition, Pearson Education 2001

- **Le noyau gèrent les communications entre clients et serveurs.**
- **Certains services sont impossibles à exécuter en mode utilisateur (pilotes de périphériques d'E/S):**
  - Garder certains processus serveur critiques en mode noyau
  - Garder une partie du mécanisme en mode noyau en laissant le choix des politiques aux serveurs en mode utilisateur.

Les architectures des systèmes d'exploitation

- **OS Micronoyau**



Source: « Systèmes d'exploitation », Andrew Tanenbaum, 2ème édition, Pearson Education 2001

- **Si le client communique avec le serveur par envoi de messages, il lui importe peu que le serveur soit local ou distant, le résultat (logique) est le même d'où l'adaptabilité aux systèmes distribués**

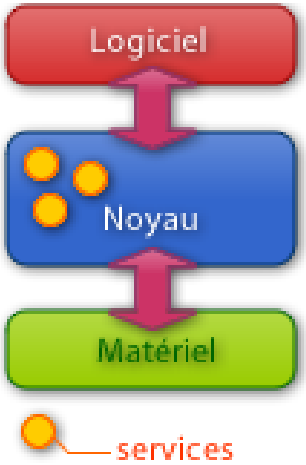
OS Micronoyau

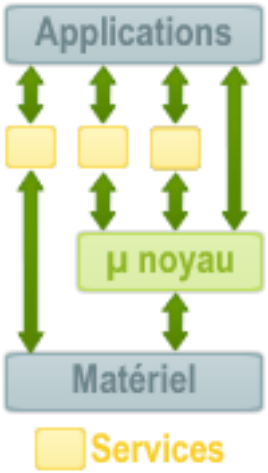
- **Avantages**
  - **Extensibilité**
  - **Minimise le code du noyau → "trusted computing base" petit**
  - **Sécurité**
    - Un serveur (mode utilisateur) crashe, il sera le seul à redémarrer
  - **Fiabilité**
    - Micronoyau: code plus petit → moins de bugs
- **Inconvénients**
  - **Souvent tenté de rajouter des choses dans le noyau (vu qu'il est petit…)**
  - **Mauvaises performances**
  - **Requière beaucoup de prudence lors de la conception**

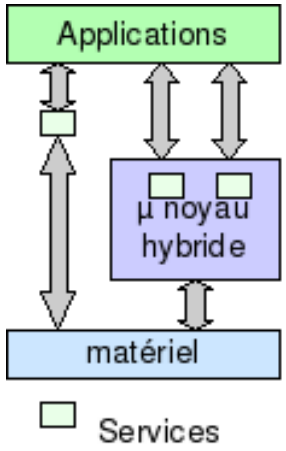
OS Hybride

- **OS Hybride**
  - **Un hybride entre une architecture de noyau monolithique et un micro-noyau**
    - Les concepteurs d'un noyau hybride peuvent décider de garder plusieurs composants à l'intérieur du noyau et d'autres à l'extérieur
    - "*Au début des années 1990 les développeurs et concepteurs se sont aperçus des faiblesses des premiers micro-noyaux, certains réintégrèrent diverses fonctionnalités non principales dans le noyau*"

Récapitulatif







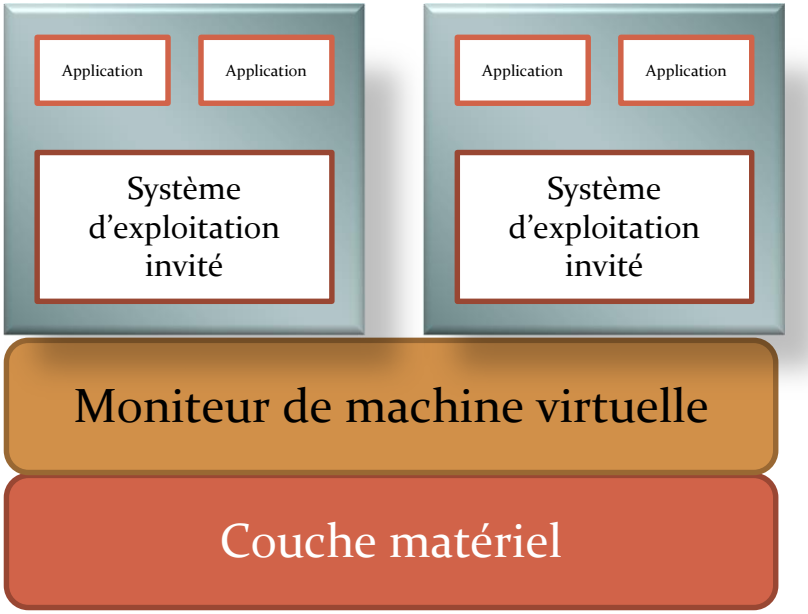
http://www.berkeley-software.wikibis.com/noyau_de_systeme_d_exploitation.php

Les architectures des systèmes d'exploitation

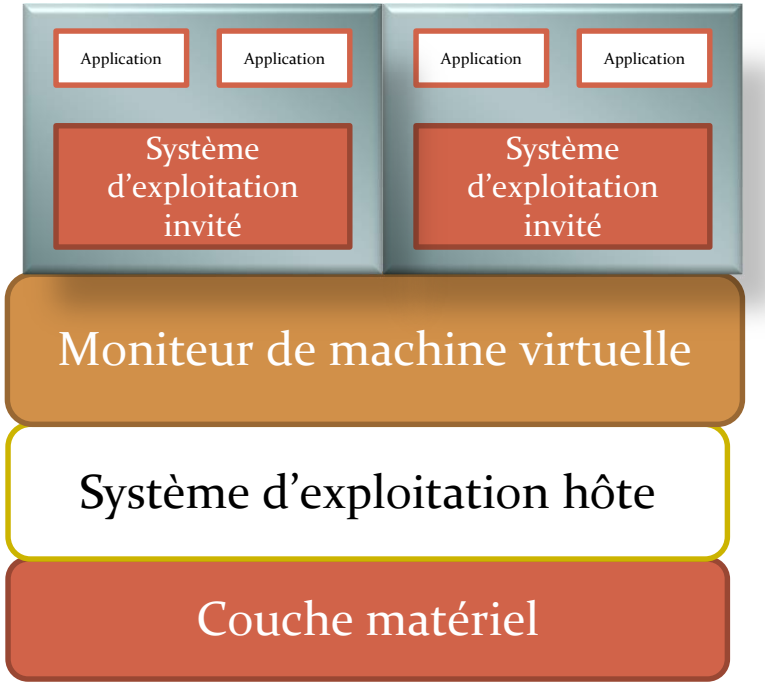
- **Machine virtuelle**
  - **L'OS doit remplir 2 fonctions**
    - Multi programmation → moniteur de machine virtuelle
      - Mode privilégié (exécution)
      - Plusieurs processeurs virtuels
    - Services système → système invité
      - Un ou plusieurs OS « invités » qui s'exécutent sur les processeurs virtuels et fournissent les services système
  - **Le moniteur de machine virtuelle (hyperviseur) intercepte les instructions privilégiées envoyées par l'OS invité, les vérifie (politique de sécurité) et les exécute sur l'OS hôte**
  - **Les interruptions sont aussi interceptées par le moniteur de la MV**

Machine virtuelle

- **2 types**
  - **MV native (type 1)**
  - **MV invité (type 2)**
  - **Exemple de ce type d'OS: XEN, VMWare, IBM' VM/370, VirtualBox, etc.**



VM native



VM invité

Machine virtuelle

- **Avantages**
  - **Permet l'exécution de plusieurs OS sur une seule machine**
  - **Permet une bonne portabilité des applications**
  - **Une protection complète (code exécute en mode privilégié complètement géré)**
  - **Bon environnement de développement (dev. système en mode utilisateur…)**
- **Inconvénients**
  - **Gros problème de performances (plusieurs couches)**
  - **Manque de flexibilité**

Logiciel libre et systèmes embarqués

- **Contrairement aux logiciels classiques, les (plusieurs) logiciels embarqués ont (généralement) une durée de vie particulièrement longue**
  - **Important de faire évoluer le logiciel indépendamment des aléas économiques.**
- **Contraintes des systèmes propriétaires**
  - **Sociétés de taille moyenne → ont du mal à suivre l'évolution technologique**
  - **Outils de développement sont moins accessibles, la compétence est donc plus chère à obtenir.**

Logiciel libre et systèmes embarqués

- **Logiciels libres: plusieurs critères (disponibles sur www.opensource.org), les principaux sont**
  - **La disponibilité du code source**
  - **La possibilité de réaliser des travaux dérivés**
  - **La redistribution sans *royalties***
- **Contrainte majeure : "SAV" sur le long terme + obligation de redistribuer le code**