# *Comment concevoir un environnement applicatif ouvert*

**SIG Prospective et Management**
**17 Octobre 1991**



*Bernard Ourghanlian*

# Sommaire

- Pourquoi un environnement applicatif ouvert ?
- Qu'est-ce qu'un système ouvert ?
- Qu'est-ce qu'un environnement applicatif ouvert ?
- Comment concevoir un environnement applicatif ouvert ?
- Un exemple : comment développer une application portable

# Pourquoi un environnement applicatif ouvert ?

## ***Le Monde change...***

- Les frontières s'écroulent
- L'économie de marché semble s'imposer
- Pour s'adapter au changement, le diriger et non le subir, les entreprises :
  - Se réorganisent, se séparent, s'allient, font des acquisitions,...

# Pourquoi un environnement applicatif ouvert ?

## ***Le Monde Informatique change...***

- La puissance d'un *Mainframe* arrive sur le bureau
- Le réseau est partout
- La tendance vers le *Downsizing* semble irréversible

# Vue de l'utilisateur

DEC

IBM

HP

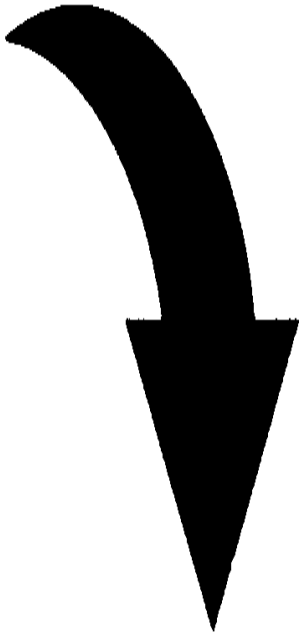
VMS

BULL

MVS

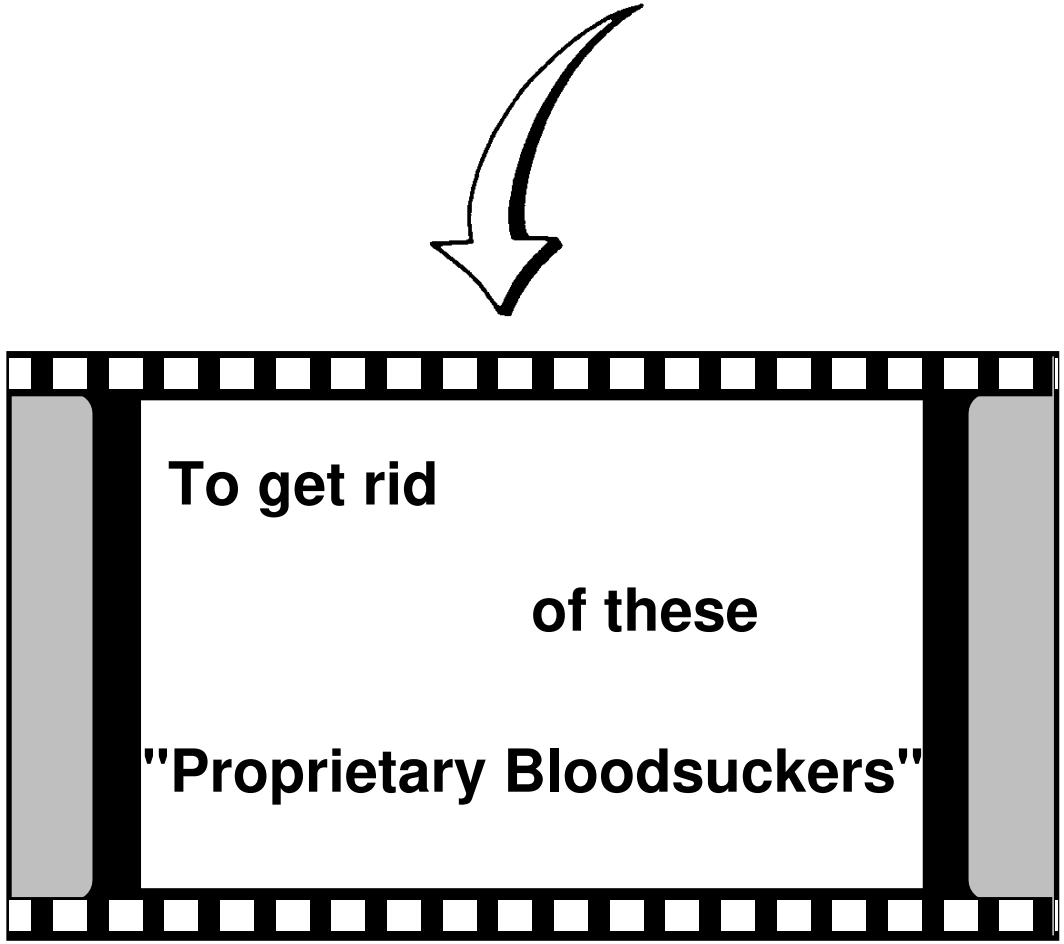
MPE

GCOS



Utilisateur

# Pourquoi un environnement applicatif ouvert ?

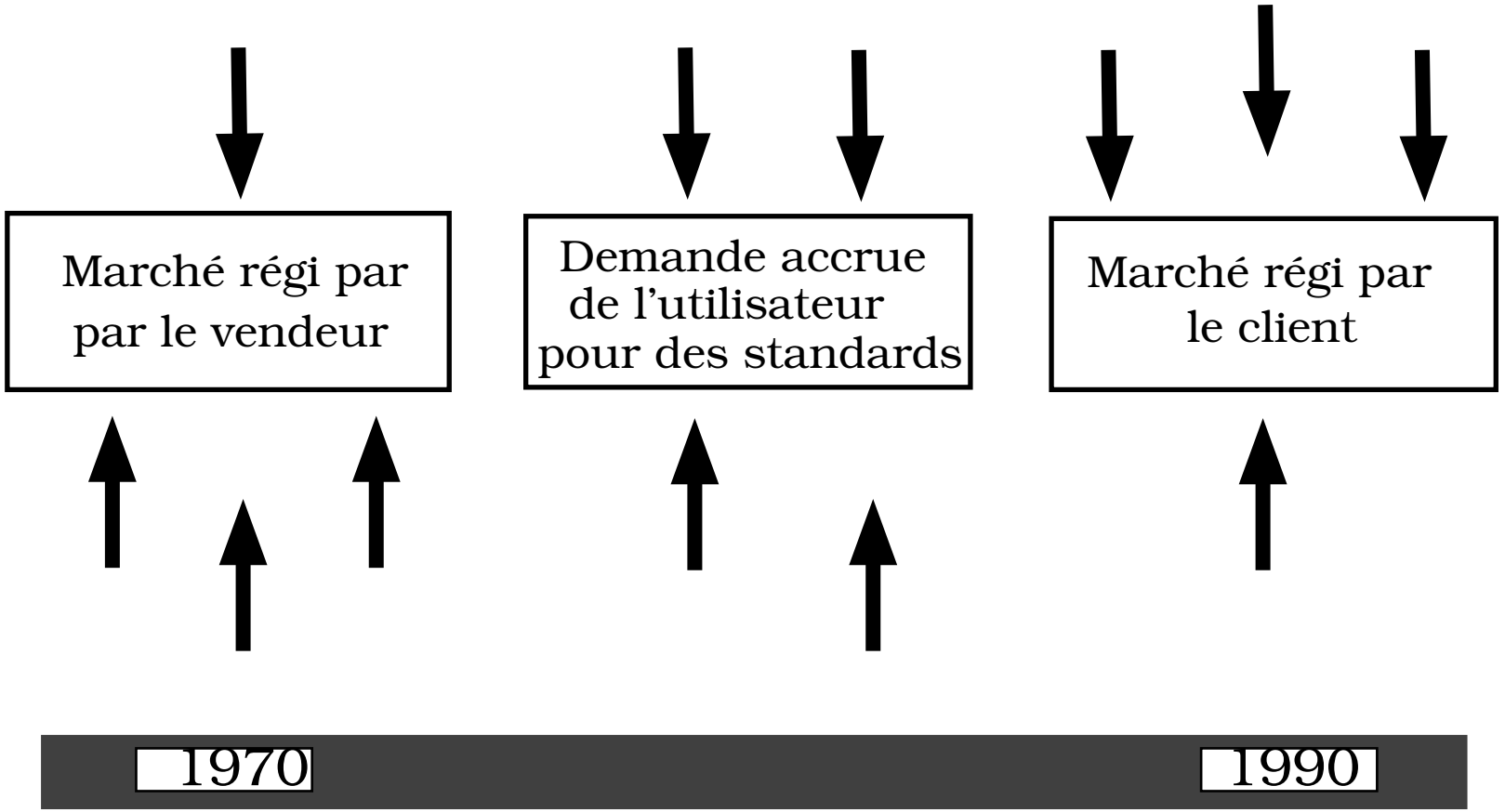


# Quand les systèmes propriétaires étaient la seule option possible

- Les utilisateurs et les entreprises étaient enchaînés au constructeur
- Profiter d'une nouvelle technologie signifiait :
  - — Jetter le précédent *hardware*
  - — Réécrire le *software*
  - — Reformer le personnel voire embaucher

- Un problème possible chez le constructeur signifiait :
  - La mise en péril de sa société
  - La mise en péril éventuelle de sa carrière...

# L'industrie informatique est en profonde mutation



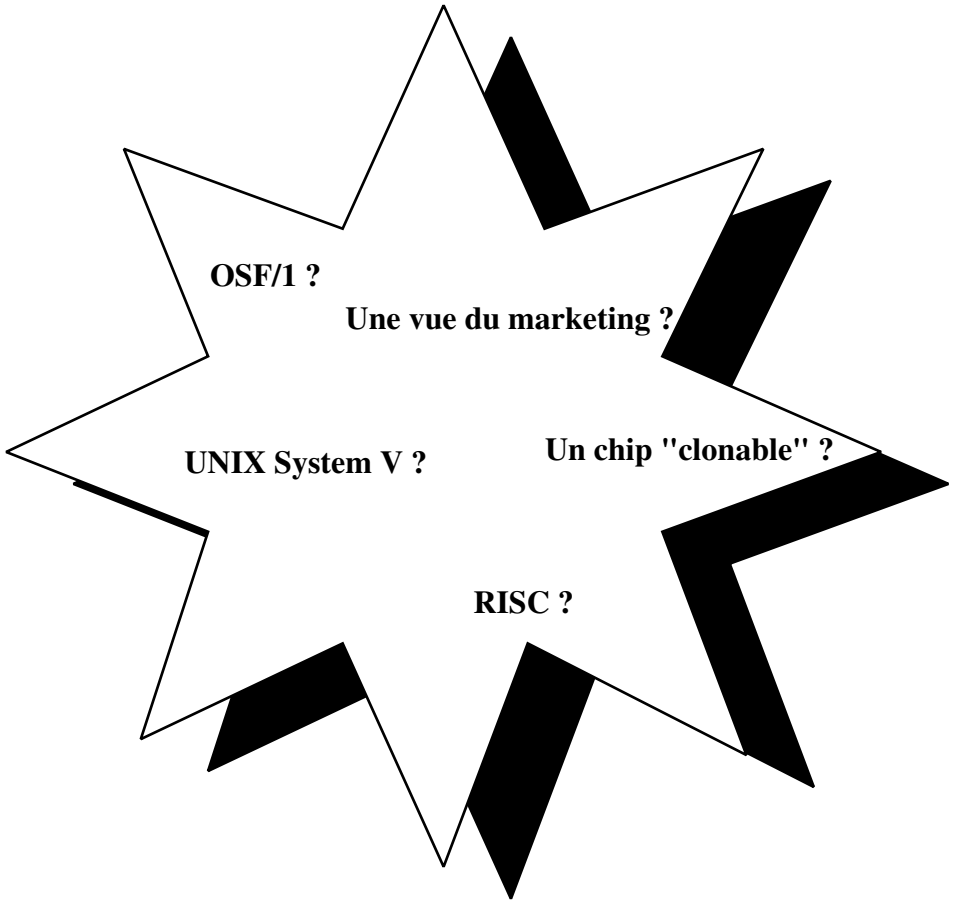
# L'industrie informatique est en profonde mutation

Les utilisateurs se précipitent donc sur les "systèmes ouverts" qui sont perçus comme une panacée

Les gains potentiels attendus de ces "systèmes ouverts" sont :

Le contrôle des coûts
La flexibilité
La possibilité d'une revanche...
....

# Qu'est-ce qu'un système ouvert ?

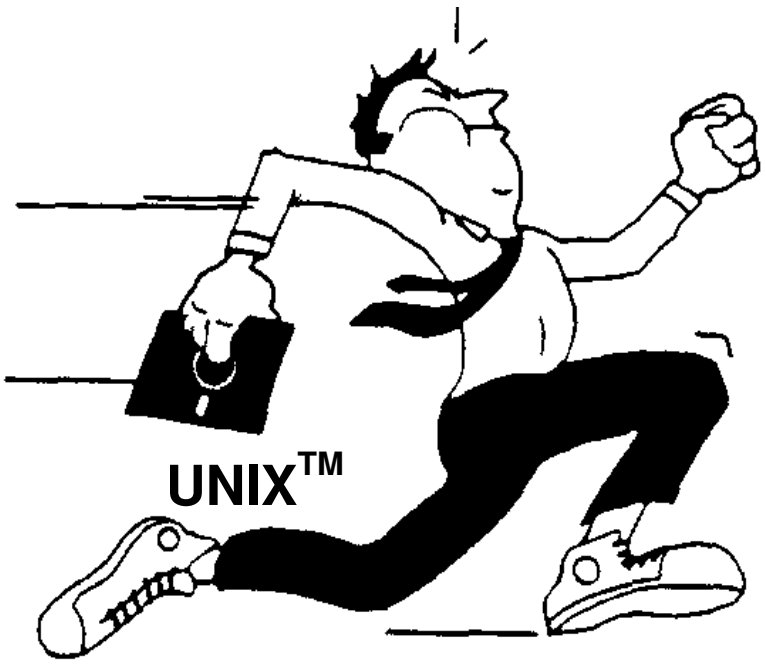


# Qu'est-ce qu'un système ouvert ?

Nouveau

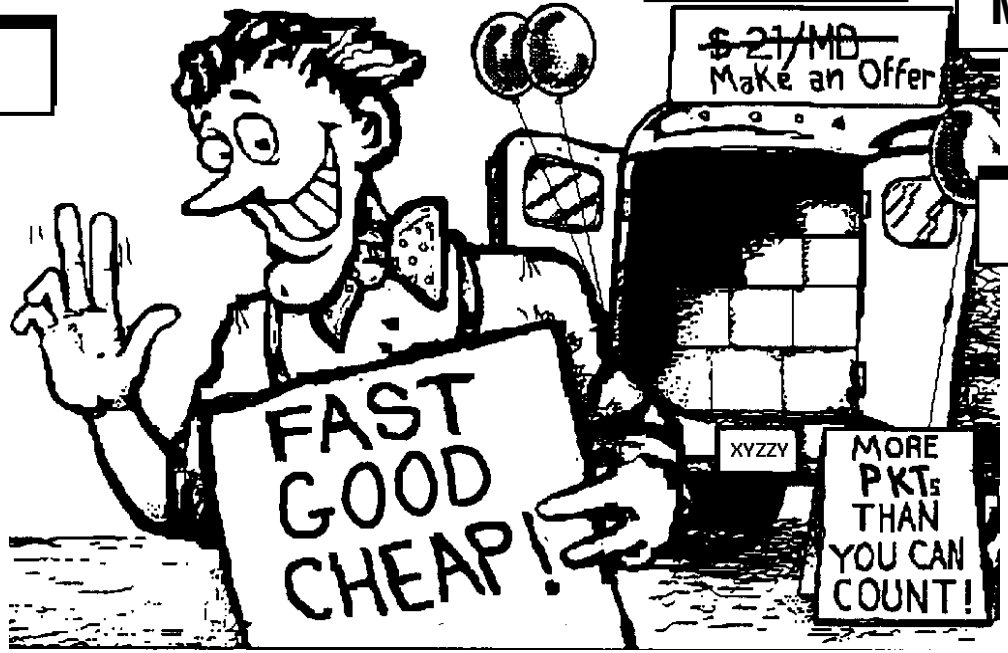
Promotion

UNIX™



# Qu'est-ce qu'un système ouvert ?

System III
Coherent
Regulus
Convex
AIX
UNIX-DG
AUX
System V
Hub
BSD
Minix
For:Pro
IDRIS
Ultrix
SUN-OS
RTU
Accent
Xenix
Primix



# Qu'est-ce qu'un système ouvert ?

*Réponse* :

**Pourquoi se battre sur une définition ?**
**Il en existe une :**

- **Stable**
- **Indépendante des constructeurs**
- **Adoptée par l'ISO**
- **Adaptable**
- **Protégeant l'investissement**

# Qu'est-ce qu'un système ouvert ?

*Réponse* :

**POSIX**

# Qu'est-ce qu'un système ouvert ?

## Pourquoi POSIX ?

**Parce que :**

- **Indépendant vendeurs/architectures**
- **Adopté par la communauté internationale :**

| | |
|---|---|
| **Open Software Foundation** | **X/Open** |
| **UNIX International** | **Uniforum** |
| **European Commission** | **NIST** |
| **ISO** | |

# Qu'est-ce qu'un système ouvert ?

## Pourquoi POSIX ?

**Parce que :**

**"...the various [UNIX] systems have diverged, and the need for standards has heightened."**

**"Your guide to POSIX", copyright 1989 by Uniforum**

# Définition d'un système ouvert par l'IEEE

"A system that implements sufficient /open specifications for interfaces, services and supporting formats to enable properly engineered applications software:

- To be ported with minimal changes across a wide range of systems
- To interoperate with other applications on local and remote systems
- To interact with users in a style that facilitates user portability"

# Définition d'un système ouvert par l'IEEE

Des *Specifications ouvertes* sont définies comme :

"...public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards."

IEEE

# P1003.0

# Qu'est-ce qu'un environnement applicatif ouvert ?

- C'est un environnement complet qui prend en compte l'aspect système MAIS aussi :
  - L'accès aux données
  - L'accès aux documents
  - L'accès au réseau
  - L'interface homme-machine
  - Les outils logiciels

# Comment concevoir un environnement applicatif ouvert ?

- Une des meilleures façon de concevoir un environnement applicatif ouvert est de le faire à l'aide d'une équipe pluridisciplinaire
- 7 étapes : les étapes permettant de comprendre ce qu'est un environnement applicatif ouvert
- Les profils applicatifs : un outil pour répondre aux besoins des applications

- Il peut être nécessaire de gérer et d'assumer son indépendance en rejoignant :
  - L'OSF
  - L'X/Open
  - L'IEEE

# Comment concevoir un environnement applicatif ouvert ?

## Étape 1 : Identifier les objectifs *business*

- Prendre en compte le problème : ne pas prendre en compte la technologie à mettre en oeuvre pour résoudre ce problème
- Quels sont les facteurs importants :
  - Portabilité
  - Interoperopérabilité

— Investissement humain

— Performances

— Disponibilité du système

— ....

# Comment concevoir un environnement applicatif ouvert ?

## Étape 2 : Identifier les besoins applicatifs

- Que faut-il faire pour couvrir les besoins applicatifs ?
- Quels sont ces besoins ?
- Quelles sont les contraintes ? :
  - Portabilité de l'application
  - Interaction homme-machine

— Interopérabilité

— Accès aux données

— Gestion

— ....

# Comment concevoir un environnement applicatif ouvert ?

Étape 3 : Identifier les normes et standards permettant de couvrir ces besoins

- C'est l'étape de création du profil applicatif
- Quels standards et normes sont disponibles ?
- Quels en sont les possibilités ?
- Si ces normes et standards ne répondent pas aux besoins, que faut-il faire ?

# Comment concevoir un environnement applicatif ouvert ?

## Étape 4 : Acquérir et/ou construire ce profil applicatif

- L'acquisition d'un système n'est pas suffisant
- Il faut "construire" son profil applicatif
- Un environnement applicatif ouvert ne s'achète pas toujours... : il se construit...

# Comment concevoir un environnement applicatif ouvert ?

## Étape 5 : Vérifier les caractéristiques d'ouverture du système créé

- S'assurer que l'application remplit bien sa mission en terme d'ouverture
- Documenter l'application

# Comment concevoir un environnement applicatif ouvert ?

## Étape 6 : S'assurer que les objectifs *business* sont satisfaits

- S'assurer que les objectifs sont atteints et que l'application répond bien au problème
- Ajuster la solution si celle-ci ne répond pas correctement au problème

# Comment concevoir un environnement applicatif ouvert ?

## Étape 7 : Recommencer...

- L'environnement étant très changeant, il est probable qu'il faudra recommencer ce processus
- Mais si ce processus a été correctement effectué, l'évolution sera beaucoup plus facile...

# Comment concevoir un environnement applicatif ouvert ?

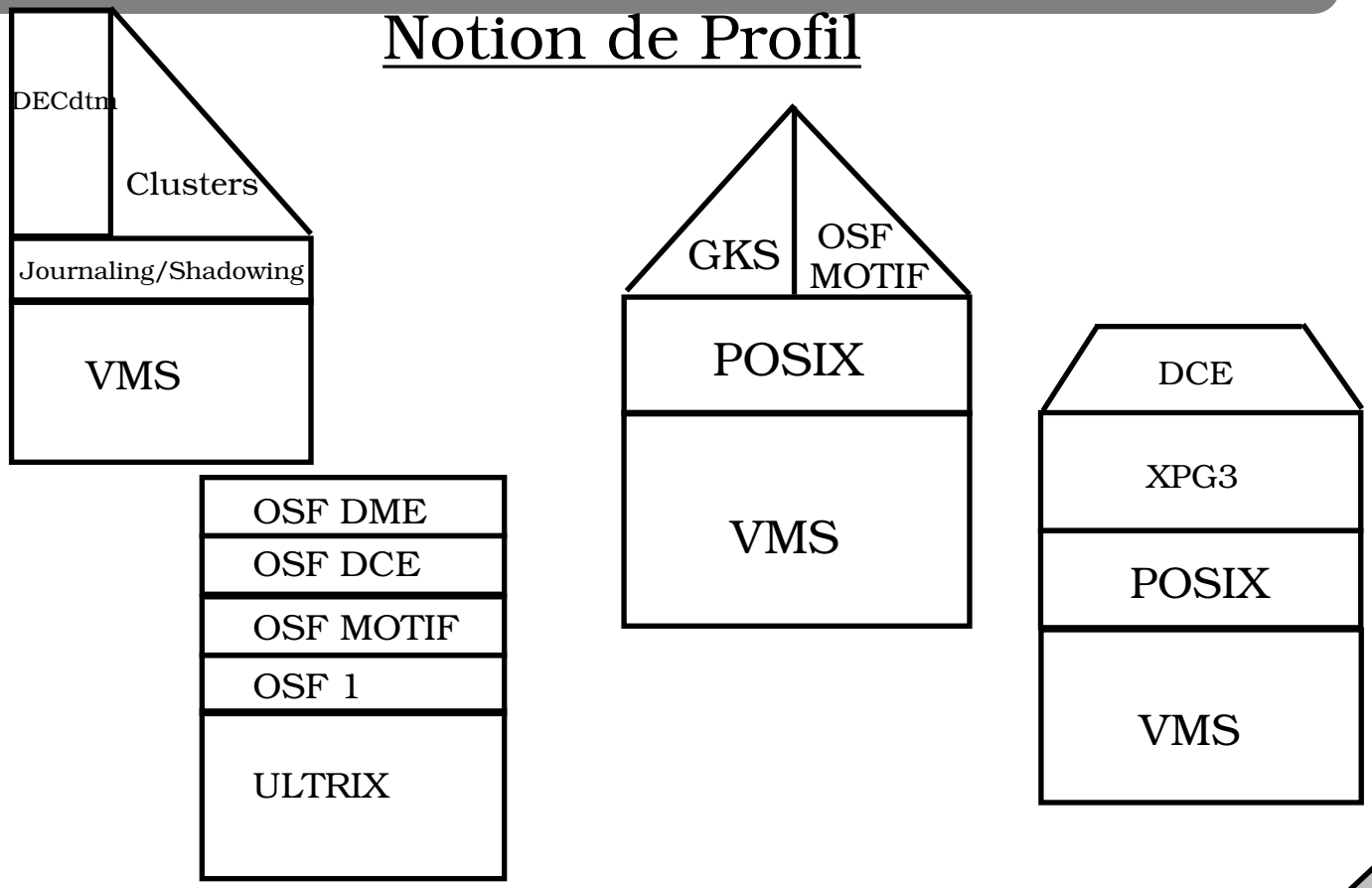
## Notion de Profil

Un Profil est:

Un ensemble de normes et standards, comprenant options et parametètres, nécessaire pour supporter les nécessités fonctionnnelles d'un domaine spécifique d'applications

# Comment concevoir un environnement applicatif ouvert ?

## Notion de Profil



# Exemple : comment développer une application portable ?

*Qu'est-ce que veut dire portable ?*

- On peut considérer 2 types de portabilité :
  - La portabilité applicative
  - La portabilité des utilisateurs

- Une application est dite portable quand le même code source peut être *complié*, peut être soumis à l'éditeur de lien et peut s'exécuter correctement sur plus d'une plate-forme

- Le degré de portabilité peut être mesuré comme une fonction du nombre de modifications qu'il faut faire subir au code source : moins il faut faire de modifications pour porter son code sur une autre plateforme, plus celui-ci est portable

- La "vraie" portabilité est difficile voire impossible à atteindre

- Une application est considérée comme permettant la "portabilité des utilisateurs" quand un utilisateur final peut utiliser cette application sur plus d'une plateforme de façon aussi transparente que possible

- Ceci veut dire que l'interface homme-machine est identique et donc que les connaissances de l'utilisateur sont portables sur la nouvelle plateforme

# Exemple : comment développer une application portable ?

*Portabilité : quels sont les avantages ?*

- Le code source est identique sur toutes les plateformes : une seule version de logiciel à écrire/maintenir : ceci permet de réduire les coûts initiaux de développement et aussi les coûts de maintenance

- L'utilisateur final n'a pas à réapprendre à utiliser l'application : ceci permet de réduire les coûts de formation

- L'entreprise peut utiliser  au mieux son matériel : il n'y a pas de duplication de *hard-ware* pour les différentes applications (traite-ment de texte sur une machine et comptabilité sur une autre) car les mêmes applications tournent sur tous les *hardwares* disponibles : l'entreprise peut disposer du bon matériel au bon endroit sans prendre en con-sidération l'application devant tourner sur celui-ci

- En raison des changements très rapides de technologie, les entreprises peuvent acquérir de nouvelles plateformes sans besoin d'investir pour porter les applications existan-tes

- Il devient possible de développer de nouvelles applications avec moins d'effort car les applications existantes ont été conçues et développées pour être portables et peuvent donc être réutilisées facilement

- Les programmeurs peuvent travailler sur différentes plateformes sans besoin de formation supplémentaire

# Exemple : comment développer une application portable ?

*Portabilité : qu'est-ce qui est disponible aujourd'hui ?*

- Des langages de programmation qui existent sur plusieurs plateformes (COBOL, PASCAL, Ada, FORTRAN) mais qui présentent des extensions spécifiques à la plateforme

- Des environnement *runtime* (SQL *services*) qui peuvent être une façon d'être indépendant de la plateforme  mais ne suffisent pas

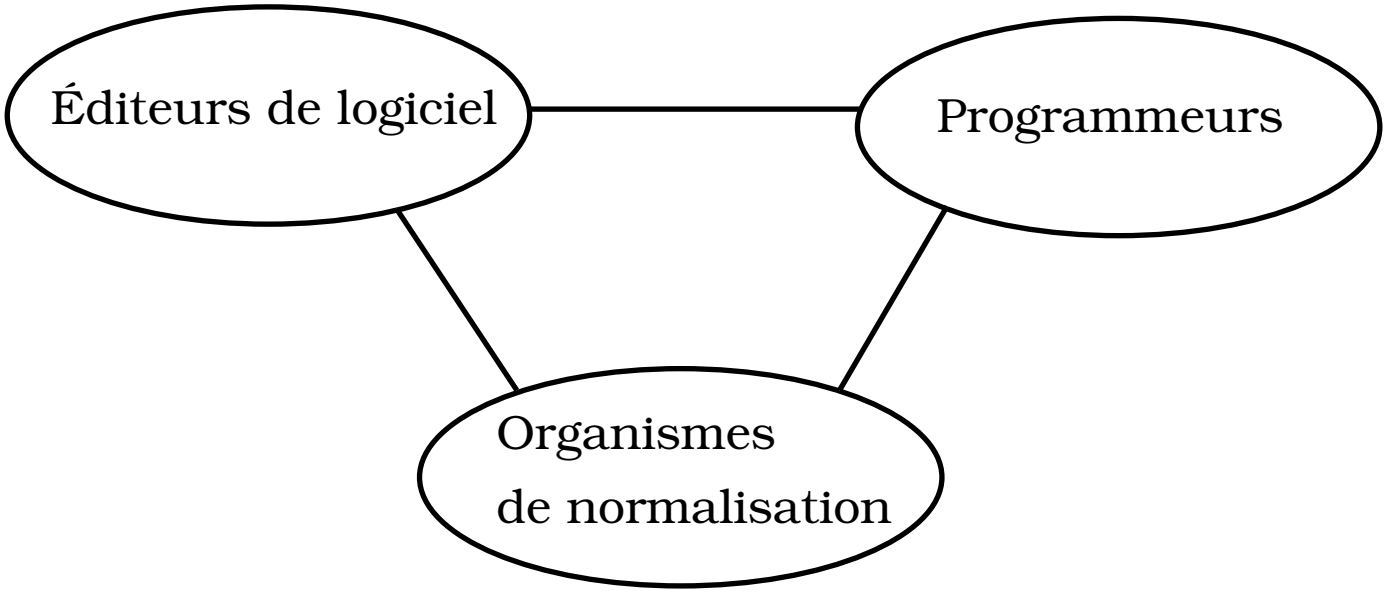
en général car n'offrant pas de services système ou d'interface homme-machine portable

- UNIX (tm) est semblable sur chacune des implantations mais il y a plus de 60 implantations majeurs d'UNIX, chacune ayant ses spécificités

## Que faut-il donc faire ?

# Exemple : comment développer une application portable ?

***Portabilité : le principe des systèmes ouverts***



# Exemple : comment développer une application portable ?

*Portabilité : le principe des systèmes ouverts*

- Les organismes de normalisation tels que l'IEEE, l'ISO développent des spécifications fonctionnelles et des règles de syntaxe pour des produits standardisés

- Les éditeurs de logiciel (comme DIGITAL) produisent du logiciel qui est en accord avec les spécifications fonctionnelles définies par les organismes de normalisation

- Les programmeurs utilisent les spécifications fonctionnelles définies par les organismes de normalisation autour des produits logiciels mis à  leur disposition par les éditeurs de logiciel pour écrire des applications

- Le principe des systèmes ouverts est donc le suivant : les  organismes de normalisations créent les normes et les standards, les éditeurs de logiciels informatique offrent des produits conformes aux spécifications et les programmeurs écrivent des applications utilisant les normes avec la bonne syntaxe; le résultat est une application qui est portable sur les différents systèmes qui supportent les mêmes spécifications

- Par exemple, un code source qui adhère à POSIX 1003.1, Motif et TCP/IP peut être compilé et s'exécuter sur une VAXstation tournant VMS (avec VMS POSIX), une DECstation tournant ULTRIX et une SPARCstation tournant Sun/OS

# Exemple : comment développer une application portable ?

*Portabilité : le problème clé*

**Déterminer l'environnement cible**

- C'est la clé d'une conception réussie d'application pour laquelle l'ojectif est la portabilité

- Une application est "réellement" portable quand elle n'utilise que l'ensemble des normes et standards disponibles sur l'ensemble des plateformes ciblées

- Cet ensemble de normes et standards disponibles sur ces plateformes est le "profil" de son application

- Par exemple, si l'on désire cibler des plateformes tournant VMS, ULTRIX, SUN/OS, POSIX.1, Motif et TCP/IP et POSIX peut être utilisé pour définir son profil

# Exemple : comment développer une application portable ?

*Portabilité : le langage de programmation*

- Connaitre l'environnement cible est la clé : le programmeur ne peut utiliser que les langages disponibles sur la plateforme cible
- Utiliser des langages normalisés (par exemple le C ANSI). Toutefois, toutes les implantations des langages normalisés ne sont pas toujours complètes

- Utiliser quand c'est possible un seul langage : une application utilisant plusieurs langages de programmation peut ne pas être portable en raison des limitations d'appels inter-langages

# Exemple : comment développer une application portable ?

*Portabilité : la méthode de programmation*

- La modularité est essentielle : il faut isoler les aspects architecturels, les dépendances du système et les fonctions et données de l'environnement logiciel et matériel (par exemple les fonctions d'entrées-sorties)

- La meilleure façon d'obtenir une application portable est de planifier son développement : avant d'écrire la première ligne de code, il

faut décider quel langage, quelles interfaces vers le système d'exploitation et quelles interfaces homme-machine utiliser. Quand c'est possible, il faut éviter d'utiliser des fonctions ou des services spécifiques à une plateforme ou un système d'exploitation donné; s'il s'avère nécessaire d'utiliser ces fonctions, il faut les isoler dans des modules applicatifs spécifiques

- Il faut se souvenir que la portabilité n'est pas gratuite : augmenter la portabilité signifie en général diminuer les performances et les possibilités. Quand on conçoit l'application, il faut évaluer les coûts et les bénéfices de la conception d'une application portable : per-

formances, possibilités, temps de développement, temps de maintenance

- De façon générale, plus une application est portable plus le prix à payer en terme de performances est élevé : il est possible de concevoir une application complètement portable dans un premier temps et d'en améliorer les performances pour des plateformes spécifiques dans un deuxième temps

# Exemple : comment développer une application portable ?

*Portabilité : considérations réseau*

- Connaitre l'environnement cible est la clé : il faut connaitre les possibilités et les contraintes de l'environnement cible
- Utiliser uniquement des protocoles de communication standardisés
- Utiliser uniquement des interfaces d'appel système standardisés ou, si celles-ci ne sont

pas disponibles sur tous les environnements cible, définir et utiliser sa propre interface de façon à minimiser la ré-écriture lors du portage de l'application sur une autre plateforme

- Si l'on désire échanger des données entre des applications sur le réseau, utiliser un protocole d'échange de données strandardisé ou si celui-ci n'est pas disponible, définir et utiliser son propre protocole de façon à être indépendant du système

# Exemple : comment développer une application portable ?

*Portabilité : considérations interface homme-machine*

- Concevoir l'interface homme-machine pour être aussi semblable que possible sur toutes les plateformes cibles : les masques d'écran doivent être semblables, la saisie des données doit s'effectuer dans le même ordre, les touches de fonctions doivent être identiques : ceci correspond au suivi d'un guide de style

- Utiliser des interfaces homme-machine standardisées (Motif, FIMS)

# Exemple : comment développer une application portable ?

*Portabilité : le développement*

- Tester son code sur toutes les plateformes cible aussi fréquemment que possible
- Gérer ses sources avec méthode : garder ses sources sur un seul système !!!
- Gérer et transférer ses sources entre les systèmes en utilisant des outils réseau standard

- Établir et maintenir des normes de lisibilité du code source

# Exemple : comment développer une application portable ?

*Portabilité : le développement en langage C*

- Types de données :
  - — Toujours considérer la portabilité et l'indépendance du *hardware* lors du choix des types de données
  - — S'il n'est pas possible d'éviter des déclarations ou des types de données spécifiques à un système, utiliser une macro pour cacher

ces différences : lors du portage, seule la définition de la macro est à changer

— Ne pas s'appuyer sur les conversions de *floats* en *ints* car les règles d'arrondi peuvent varier de système en système

— Ne pas utiliser de constantes multi-caractères : ceci est très important si l'on désire créer un logiciel multi-national

— Ne pas utiliser de variables de type *char* pour y stocker des données non caractères

— Ne pas supposer qu'une *union* de deux *shorts* est la même chose qu'un *long*

— Ne pas utiliser de noms globaux qui ne diffèrent que par majuscule/minuscule : cela peut poser des problèmes à l'éditeur de lien

- Structures et pointeurs :

  — Ne jamais utiliser la conversion implicite

  — Éviter de déclarer des tableaux de taille non définie dans une structure

  — Passer et retourner un pointeur vers une structure plutôt que la structure elle-même

  — Quand une structure est envoyée sur le réseau, coder cette structure champ par champ (la même structure peut avoir dif-

férentes implantations sur différentes plate-formes)

— Attention au fait que les différentes plate-formes peuvent avoir différents *character sets* (ASCII, EBCDIC)

— Attention au problème de stockage de données ("Endian", taille d'un *byte*, *character set*)

— Ne pas utiliser de pointeurs de type *char* pour pointer vers des structures complexes et accéder des champs spécifiques

— Initialiser les pointeurs avant de les utiliser !!

— Ne pas supposer que les pointeurs sont signés

— Ne pas traiter les pointeurs comme des entiers et réciproquement

- Fichiers et *file system* :

— Ne pas supposer que les structures de fichiers sont orientées enregistrement !

— Ne pas supposer que les structures de fichiers sont orientées *stream*

— Ne pas supposer qu'il y a des *terminateurs* de ligne

— Ne pas supposer qu'il y a un *buffer cache*

# Exemple : comment développer une application portable ?

*Portabilité : résumé*

- Avant décrire l'application :
  - — Bien connaître sa cible
  - — Définir ses objectifs de portabilité
- Pendant la conception de l'application :
  - — Bien connaître sa cible

— Prendre en compte les compromis possibles portabilité-performance en isolant les parties du logiciel faisant appel à une fonctionnalité spécifique d'une plateforme

- Pendant l'écriture de l'application :

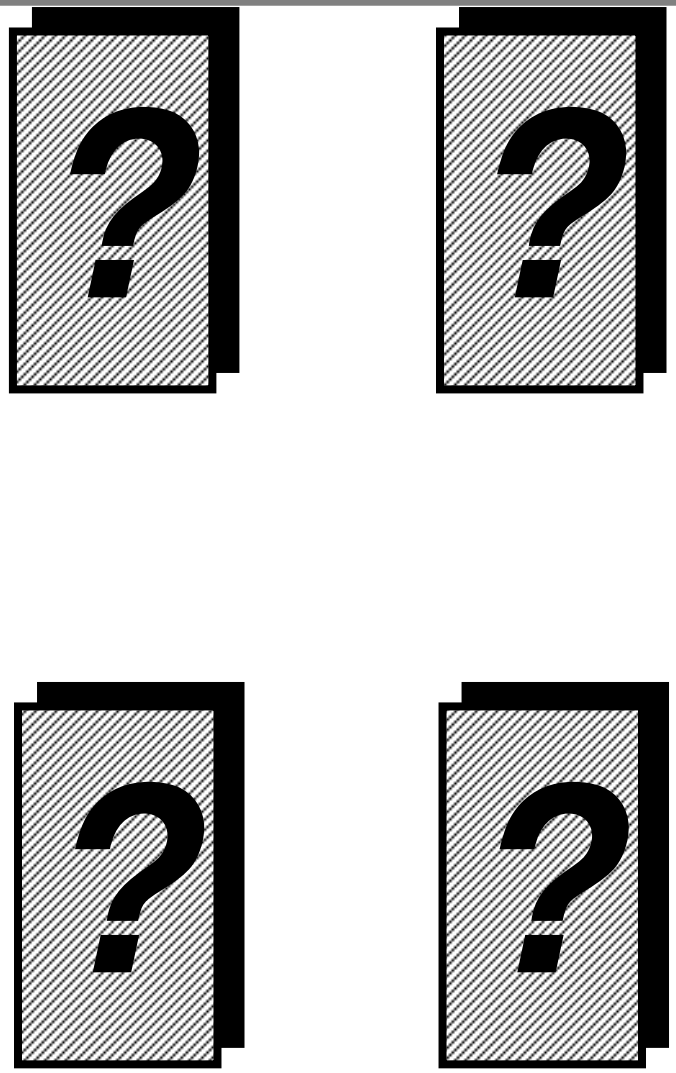
— Penser en terme de portabilité

— Tester souvent

# Exemple : comment développer une application portable ?

* Développer une application portable est une science, pas un art

* Cette science peut s'apprendre par la pratique

* Il faut connaitre la cible de son développement

# Questions