

Le coût des déficiences et des défauts en informatique est élevé :

- erreurs propagées par un programme utilisateur,
- erreurs dans le SE,
- pannes matérielles,
- malveillances, etc...

Le coût de toute protection est également élevé. D'où la recherche d'un compromis privilégiant des objectifs stratégiques.

Le degré de protection dépend de la protection des informations manipulées, du degré de confiance que l'on peut accorder aux logiciels, dont le SE.

Un logiciel fiable :

- satisfait à ses spécifications,
  - résiste à un environnement imprévu :
    - soit en le signalant,
    - soit en le corrigeant
- en évitant la propagation des erreurs.

## 1. CONTROLE D'ACCES AUX RESSOURCES PAR UN PROCESSUS

Il s'effectue à deux niveaux :

- niveau 1 (logique) : en développant un modèle de protection : des règles spécifiant quels accès sont autorisés et quels accès sont interdits
- niveau 2 (physique) : en mettant en oeuvre des mécanismes de protection pour appliquer le modèle

## 2. MODELE DE PROTECTION PAR MATRICE D'ACCES

Le SE comporte :

- des entités actives ou sujets : les processus
- des entités accessibles ou objets, représentants de *types abstraits*. Ce sont :
  - les ressources matérielles : UC, mémoire centrale, entrées/sorties, etc...
  - les ressources logicielles : fichiers, programmes, sémaphores, etc...

Un modèle de protection doit répondre à la question : *Quels sujets ont accès à quels objets ? De quelle façon (modalités d'accès) ?* Etant entendu qu'il convient de prévoir une modalité d'accès par type abstrait.

**Définition :** On appelle **droit d'accès** tout couple (nom d'objet, modalités d'accès à cet objet). Par exemple (fichier f1 , lire) ou (processus P0 , appeler)

Le modèle de protection fixe le droit d'accès de chaque processus à chaque instant. L'ensemble de ces droits pour un processus est appelé **domaine de protection du processus**.

Exemple :  $D = \{ (\text{fichier f1 , ouvrir}) , (\text{fichier f1 , fermer}) , (\text{fichier f1 , lire}) , (\text{fichier f1 , écrire}) , (\text{éditeur , appeler}) , (\text{console 5 , lire}) , (\text{imprimante , afficher}) \}$

Un domaine de protection peut se représenter par une **matrice d'accès**.

Exemple :

		objets				
		fichier 1	segment 1	segment 2	processus 1	éditeur
sujets	processus 1	L	exec	L/E		entrer
	processus 2	L/E				entrer
	processus 3		L/E , exec		entrer	entrer

↓  
domaine de protection dprocessus 1

## **2.1 Domaine de protection restreint**

Un processus peut être décomposé en tâches, avec des droits d'accès différents selon les tâches. Donc, la matrice doit pouvoir évoluer dynamiquement :

- pour ajouter un droit d'accès à la case (i , j)
- pour retrancher un droit d'accès à la case (i , j)

Par exemple, si on ajoute (**domaine j , entrer**) en ligne i, le processus qui s'exécute dans le domaine de protection i passe au domaine de protection j.

Exemple :

Le degré de protection d'un SE est celui de son maillon le plus faible ("principe de la porte dérobée"). Un procédé de contrôle simple, mais permanent est préférable à un procédé complexe, mais pas systématiquement utilisé. De plus, il faut moduler les contrôles en fonction des utilisateurs. Pour ces raisons, tout type abstrait de domaine de protection doit être modulaire.

## **2.2 Problème du cheval de Troie**

Un programme peut profiter du domaine de protection de l'utilisateur pour consulter ou copier ou modifier des données auxquelles il ne devrait pas avoir accès. **Il est donc important d'exécuter tout programme dans le domaine de protection le plus réduit.**

## **2.3 Problème du confinement**

Un programme, exécuté par un utilisateur qui n'est pas son propriétaire, peut retenir une information confidentielle : **il faut donc confiner l'information.**

Les mécanismes de contrôle rassemblés dans la matrice d'accès peuvent être classés en trois catégories :

- accès hiérarchiques
- listes d'accès
- capacités

## **3. ACCES HIERARCHIQUES**

On peut hiérarchiser les accès en deux modes :

- le mode système ou maître : par exemple tout ce qui concerne la création ou la destruction des processus, l'accès aux tables du SE, les instructions d'entrée-sortie. Le domaine de protection est vaste : tous les droits du SE (processus système, super utilisateur(s))

- le mode utilisateur ou esclave : le domaine de protection est celui d'un sous-ensemble d'instructions autorisées et utilisées

Souvent, un bit du mot d'état de la machine indique quel est le mode de protection pour la mise en œuvre des contrôles.

Toutefois, il n'est pas facile de mettre en œuvre le *principe du moindre privilège* énoncé ci-dessus. Une solution peut être d'organiser des domaines de protection en anneaux concentriques. L'anneau central n° 0 correspond au maximum de privilèges. La mémoire centrale est segmentée avec une partition par anneau (par domaine de protection).

Le cas de 2 anneaux correspond à la situation simple maître/esclave.

Lorsqu'un processus s'exécute dans le domaine  $D_i$  ( $i = 0, 1, 2, \dots$ ), il peut accéder à tout segment de mémoire centrale de niveau n°  $k$  ( $k \neq i$ , mais jamais  $k < i$ ). C'est en contradiction avec le principe de moindre privilège.

Une solution élégante à ce problème a été proposée pour la 1ère fois pour la gamme des microprocesseurs INTEL 80286 en 1984 : il s'agit de la technique des guichets (**gates**). Un processus n'accède à un module de niveau inférieur que par un guichet facilement contrôlable.

**Définition :** un guichet est un descripteur, associé à un numéro d'anneau, qui pointe vers un segment de mémoire de n° d'anneau inférieur.

**Règles :** - tout processus qui peut accéder à un guichet accède au segment pointé par le guichet  
- il y a 2 solutions, selon les systèmes, pour répondre à la question : "*dans quel domaine de protection travaille-t-on dans l'anneau appelé ?* "

**solution 1 ou principe de conformité :** un processus de niveau k ( $k < i$ ) qui accède à un guichet pointant sur le niveau i s'exécute dans l'anneau i. Il n'a pas accès à la mémoire de niveau k, son propre niveau !

**solution 2 ou restriction de privilèges :** on associe temporairement au segment de mémoire de niveau k le numéro d'anneau i pour qu'un processus de niveau k puisse accéder à cet anneau quand il est exécuté au niveau i

## 4. LISTES D'ACCES (authorized list)

Chaque colonne de la matrice d'accès est structurée en une liste chaînée de protections associée à un objet. C'est le propriétaire de l'objet qui définit la liste.

Lorsqu'un processus veut accéder à un objet, le SE vérifie dans la liste que l'accès est permis.

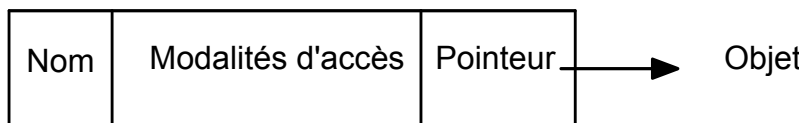
La technique UNIX des bits de protection `rwX ....` s'apparente à cette solution.

Toutefois, un problème se pose : il n'est pas immédiat de modifier un domaine de protection, puisqu'il faut pour cela explorer *toutes* les listes. Cette solution n'est donc pas bien adaptée à la mise en œuvre du principe de moindre privilège.

## 5. CAPACITES (capability)

Une capacité est une structure à 3 champs :

- nom d'un objet
- modalités d'accès à cet objet
- pointeur sur l'objet



**Règles :** - une capacité est créée par un processus particulier du SE

- les modalités d'accès sont le seul des 3 champs d'une capacité dont la modification est autorisée

- un sujet peut transmettre la copie d'une capacité à un autre sujet, éventuellement avec des modalités d'accès réduites

- lorsqu'un sujet crée un objet, le SE lui alloue une capacité associée à l'objet

Une capacité correspond à un **découpage en lignes** de la matrice d'accès. Le domaine de protection d'un processus est formé de la liste des capacités (*C-liste*) qu'il possède à un instant donné.

Exemple :

Pour un processus, changer de domaine de protection peut consister à changer de C-liste.

### **5.1 Domaines de protection restreints**

Une solution élégante consiste à faire de chaque C-liste utilisée un objet n'admettant que la seule modalité d'accès *entrer*. Pour accéder à cet objet, donc pour entrer dans le domaine de protection décrit par la C-liste, un sujet doit posséder une capacité d'entrée pour cet objet.

Cette méthode permet de bien réaliser le principe du moindre privilège. Ainsi, dans la figure ci-dessous, lorsqu'un processus appelle une procédure, il doit posséder une capacité lui permettant d'accéder à la C-liste de la procédure.

**Règles :** - après l'appel à une procédure, le domaine de protection du processus devient celui de la C-liste de la procédure

- une fois que le processus est sorti de la procédure, son ancien domaine de protection est restauré

### **5.2 Protection des capacités**

Elle est normalement effectuée au niveau de l'objet, c'est à dire du matériel. Deux solutions sont envisageables :

**- solution 1 : étiquetage**

Dans chaque mot mémoire, un bit indique si le mot est une capacité. Ainsi, il n'y aura pas de manipulation illicite des droits d'accès d'une capacité.

**- solution 2 :partition de la mémoire**

Chaque segment de mémoire contient soit des capacités, soit des données. Il existe des registres spécifiques pour la manipulation des capacités, d'autres pour les données.

**Règles** : les processus système accèdent aux registres capacité et aux segments capacité. Les données ne peuvent être copiées que dans des segments ou des registres de données.

Un problème se pose : la **révocation d'une capacité**.

Les capacités transmises à d'autres sujets sont disséminées. C'est extrêmement incommode pour supprimer une capacité.

La solution généralement adoptée est la suivante : le créateur d'un objet de nom O ne transmet pas à un autre sujet une copie de la capacité de O, mais il transmet une capacité pour l'objet O' qui pointe sur O. Comme la capacité de O n'est pas dupliquée, son éventuelle suppression est simple : pour révoquer, on supprime le pointeur de O' sur O.

### **Bibliographie**

J. BEAUQUIER, B. BERARD, Systèmes d'exploitation, Edisciences  
(plusieurs figures du chapitre sont issues de cet ouvrage).