Programmation Système Win32

Cours pour programmeurs C/UNIX

Les DLLs

Jean-Yves Tigli, tigli@essi.fr, 2017

Plan

1. Concept de DLL
2. Terminologie : différents types de librairies
3. Les avantages de l'utilisation des DLL.
4. Différents types de liens avec une DLL.
    - 4.1 Création de la DLL
    - 4.2 Lien implicite à la compilation
    - 4.3 Lien implicite à l'édition de lien
    - 4.4 Lien implicite à l'exécution
    - 4.5 Lien explicite
5. Organisation de la mémoire réservée à un processus dans Win32 avec les DLL associées.

1. Concept de DLL

- Une DLL est un fichier contenant des données, des fonctions compilées, des ressources.
- Elle est relogeable dans la mémoire virtuelle attribuée au processus appelant.
- Une DLL exporte des fonctions, des variables (DLL Win32), des types (classes).
- Le programme client importe ces fonctions.
- Windows contrôle l'adéquation entre les importations et exportations quand il charge la DLL.
-

2. Terminologie : différents type de librairies

- Librairie "objet" :
  - c'est un fichier dont l'extension est .LIB qui contient du code qui sera ajouté à celui du programme EXE au moment de l'édition de lien (de façon statique).
  - Par exemple pour MS Visual C++, la librairie de run-time "C" LIBC.LIB est statiquement linkée avec le reste du code.
- Librairie "import" :
  - aussi un fichier d'extension .LIB qui ne contient pas de code, mais sera utilisé par le linker pour résoudre les appels aux fonctions externes.
  - Elles donnent au linker les informations nécessaires pour construire des tables pour le lien dynamique (lors de l'exécution).
- Ces deux types de librairies sont utilisées au moment de l'édition de lien (lors de la création d'un exécutable) et plus jamais après.

2. Terminologie : différents type de librairies

- Librairie DLL doit être présente sur le disque au moment avant que le programme appelant soit exécuté.
- Windows recherche le fichier DLL selon la séquence suivante :
  - dans le répertoire contenant le programme EXE
  - dans le répertoire courant
  - dans le répertoire WINDOWS\SYSTEM
  - dans le répertoire WINDOWS
  - dans les répertoires accessibles sous la commande PATH de l'environnement MS-DOS

3. Les Avantages de la DLL

- **Lien dynamique :**
  - Une DLL n'est pas un fichier directement exécutable.
  - Ce sont des fichiers séparés contenant des fonctions appelées par d'autres programmes ou DLL.
  - Par rapport au lien statique crée par le linker au moment de la création de l'exécutable, un lien dynamique entre l'appelant et la DLL appelée se fait au moment de l'exécution.
  - Certaines librairies ne contiennent pas de code, mais uniquement des ressources (par exemples, des fontes).
- **Chargement automatique par l'OS :**
  - Bien que certaines DLL peuvent avoir l'extension .EXE ou .FON, l'extension générale d'un fichier DLL est .DLL.
  - Les librairies DLL peuvent être partagées par plusieurs processus

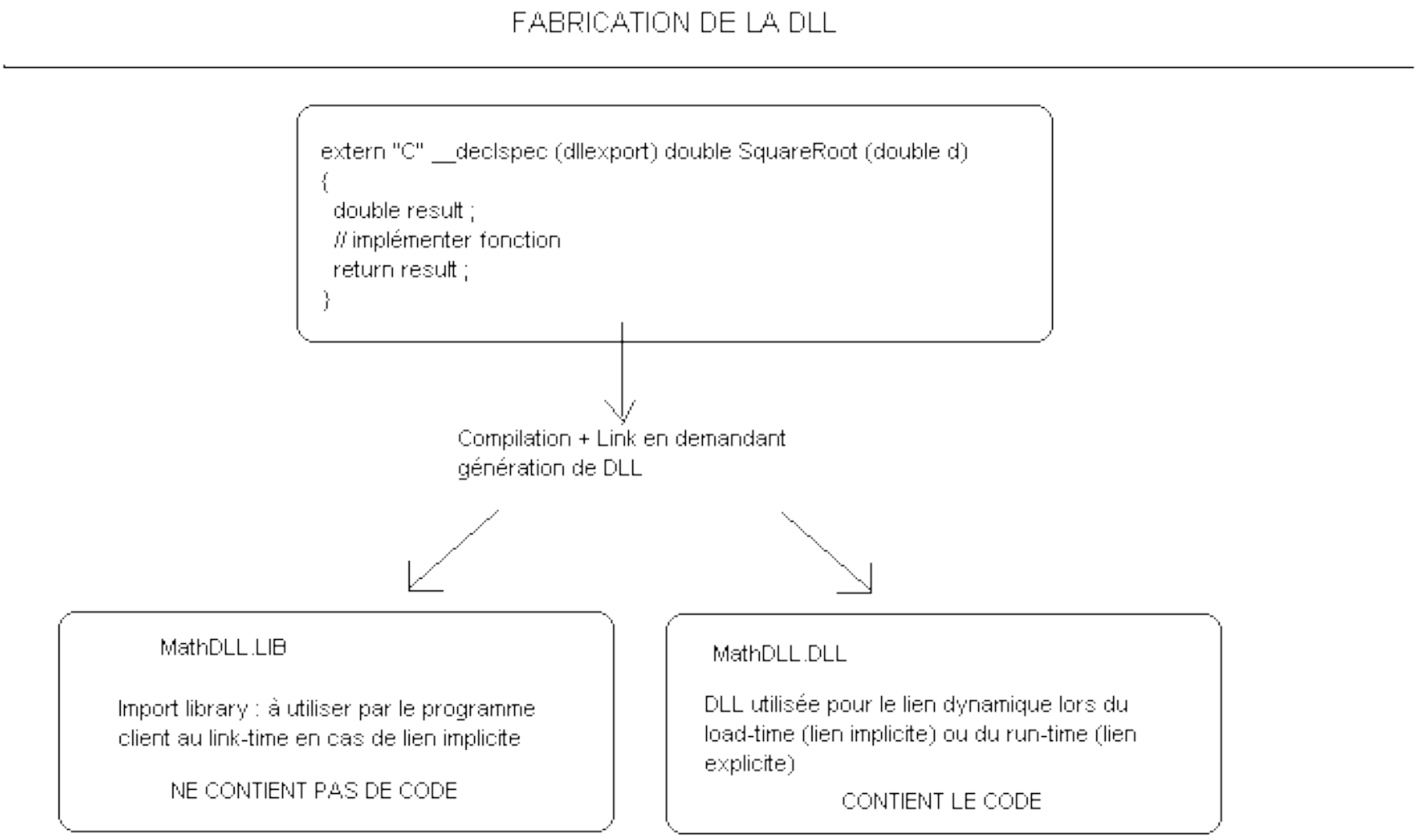
3. Les Avantages de la DLL

- **Modularité :**
  - Il est très utile de faire appel à la technique des DLL lorsqu'on veut écrire du logiciel modulaire.
  - Les classes d'une application sont modulaires jusqu'au moment du build, tandis que les DLL sont modulaires au run-time.
  - De plus, au lieu de bâtir de grandes applications lourdes à compiler et à linker, il est préférable de faire de plus petites DLL testées individuellement.
  - Un autre avantage par rapport aux librairies statiques est que si plusieurs programmes utilisent la même DLL, les modifications apportées à cette DLL seront reportées à tous les programmes utilisateurs sans refaire d'édition de lien (à condition que les interfaces n'aient pas changé).

4. Différentes types de Liens : implicites et explicites

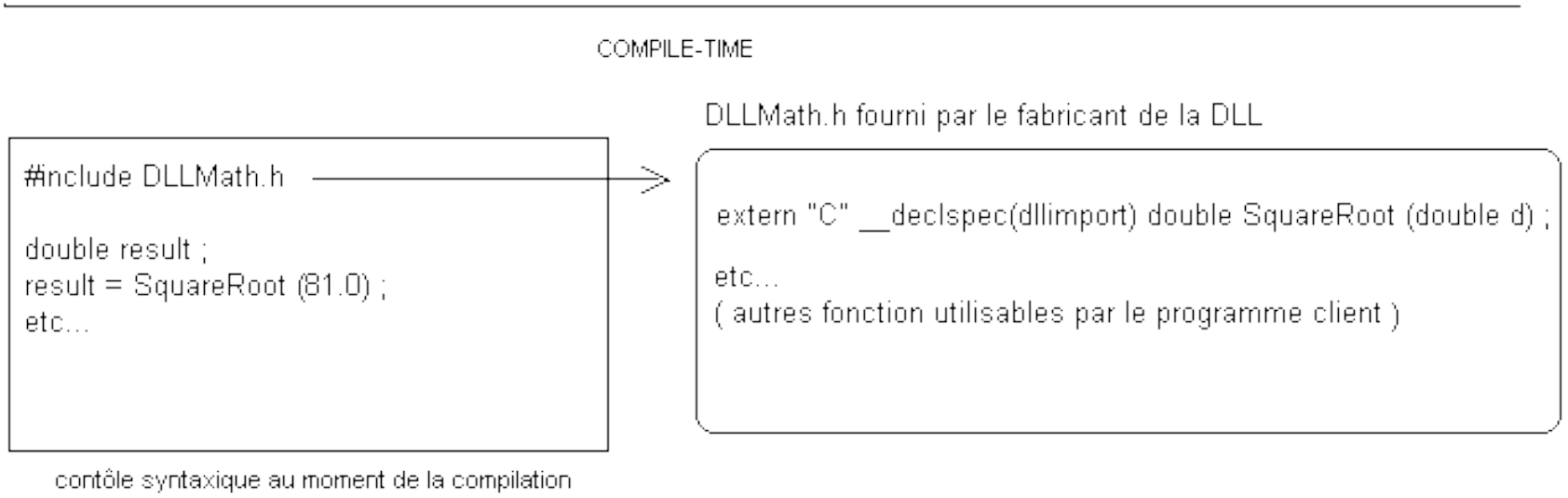
- Un exécutable peut faire le lien (ou charger) une DLL de 2 façons :
  - Lien implicite (appelé aussi chargement statique [static load] ou lien dynamique au moment du chargement [load-time dynamic linking])
  - Lien explicite (appelé aussi chargement dynamique [dynamic load] ou lien dynamique au moment de l'exécution [run-time dynamic linking])
- **Avec le lien implicite**, l'exécutable qui utilise la librairie doit faire un lien au moment du build avec une "import library" donnée par le fabricant de la DLL. L'OS charge la DLL lorsque l'exécutable qui l'utilise est chargé). Le programme client appelle les fonctions exportées comme si les fonctions étaient contenues dans l'exécutable.
- **Avec le lien explicite**, l'exécutable client de la DLL doit appeler des fonctions pour explicitement charger et décharger la DLL au moment choisi et pour accéder aux fonctions exportées. L'exécutable client doit appeler les fonctions exportées au travers d'un pointeur.
- Un exécutable peut utiliser la même DLL en invoquant l'une des 2 méthodes.

4.1 Création de la DLL

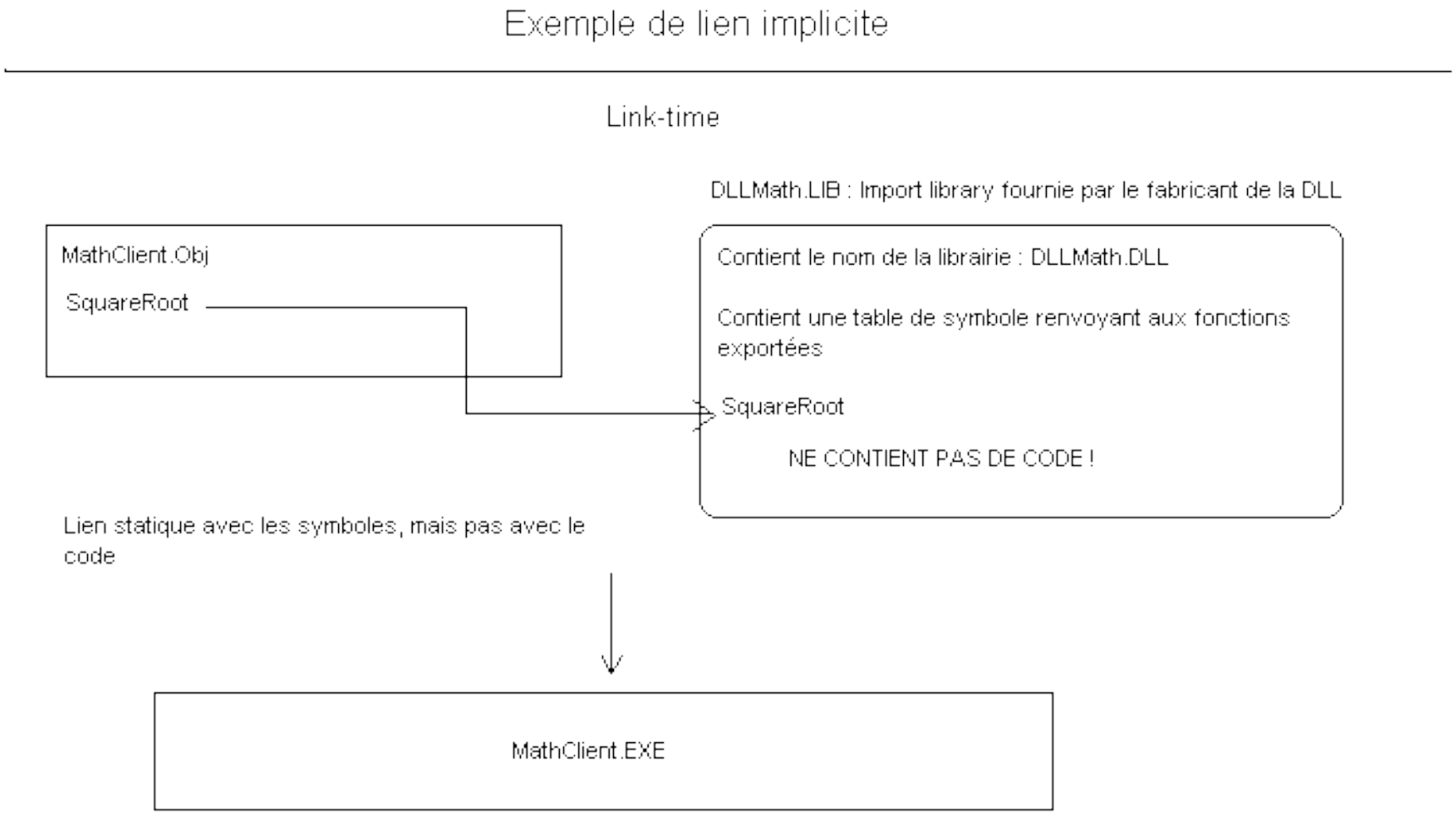


4.2 Lien implicite à la compilation

Exemple de lien implicite

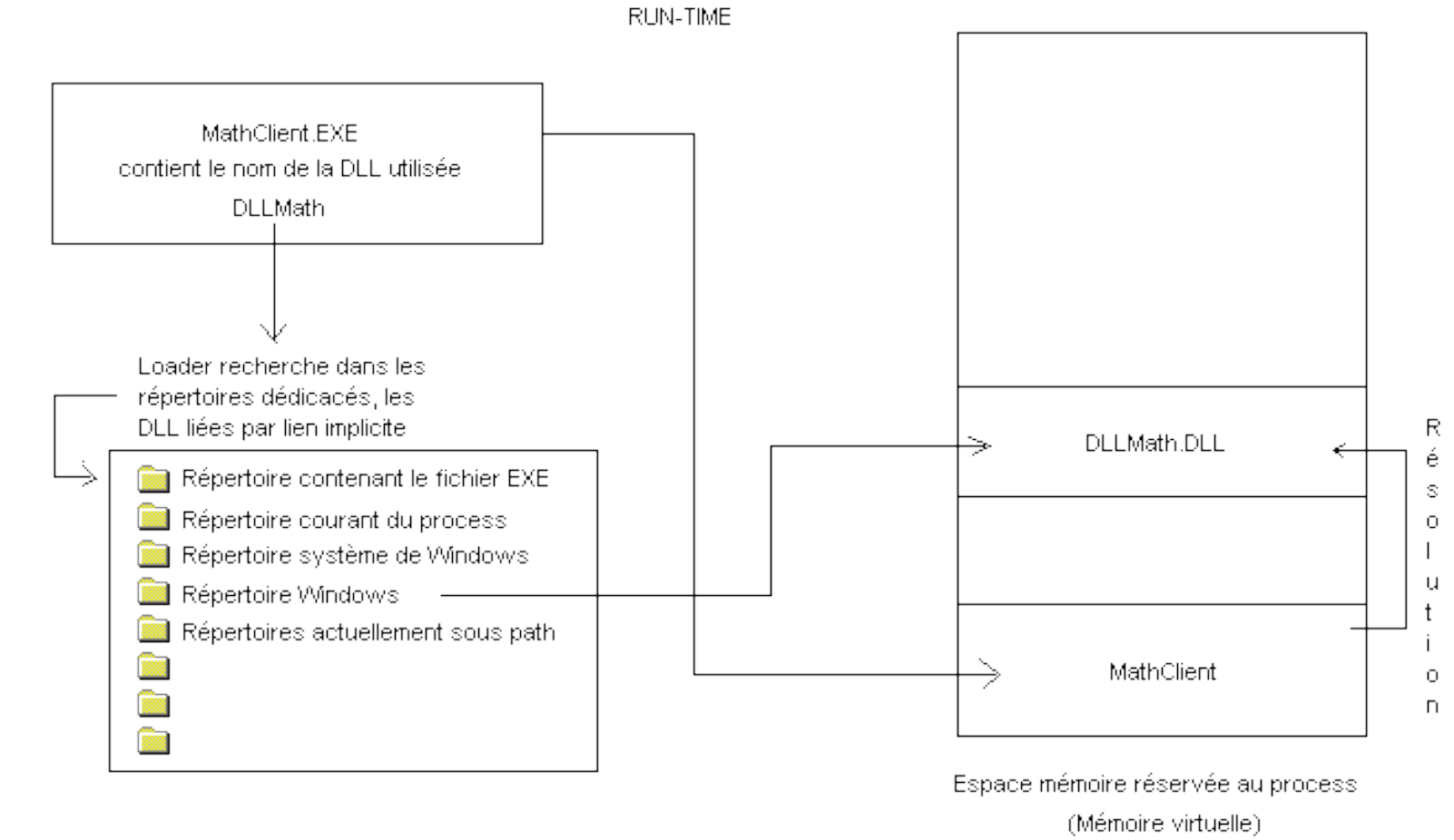


4.3 Lien implicite à l'édition de lien



4.4 Lien implicite à l'exécution

Exemple de lien implicite



Les liens dynamiques sont résolus au moment du chargement du programme client (load-time dynamic linking)

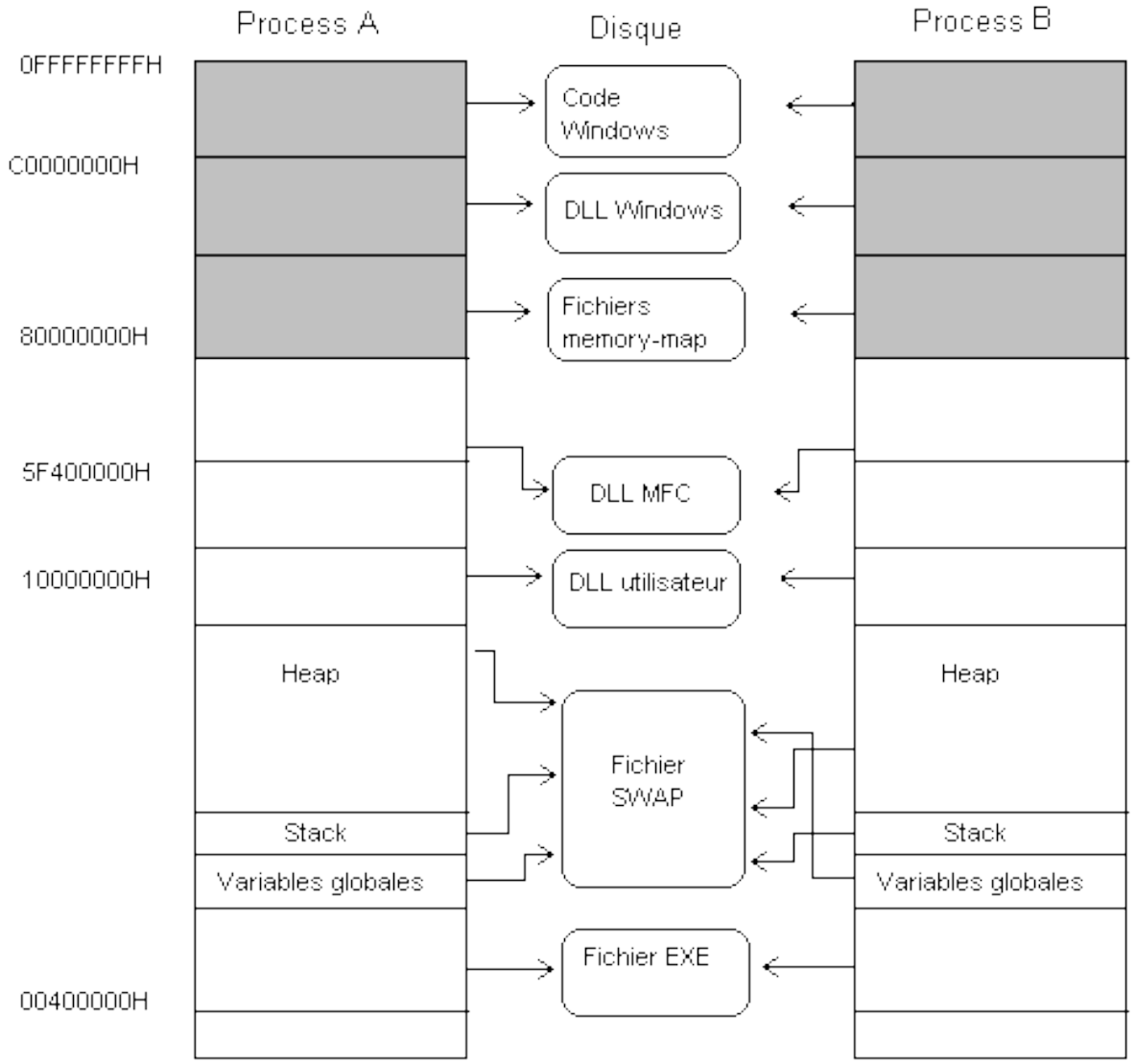
4.5 Lien explicite

- Exemple de lien explicite:
- Le programme client fait le lien explicite de la manière suivante :

```
typedef double (SQRTPROC) (double) ;
HINSTANCE hInstance ;
SQRTPROC * SquareRoot ;

// Charge la librairie à ce moment, on pwut spécifier un chemin d'accès
// complet, autrement recherche dans les répertoires dédicacés.
VERIFY (hInstance = :: LoadLibrary("DLLMath.DLL")) ;

// Fait le lien dynamique avec la procedure qui nous interesse
VERIFY (SquareRoot = (SQRTPROC *) :: GetProcAddress
(hInstance,"SquareRoot")) ;

// utilise la procedure via son pointeur
double d = *SquareRoot (81.0) ;
```

- Au moment de la compilation : Aucun contrôle syntaxique
- Au moment de l'édition de lien : rien de spécial, pas besoin de "import library"
- Au moment du chargement du programme client : la DLL n'est pas chargée.
- Au moment de l'appel de ::LoadLibrary. La DLL est recherchée par le loader, chargée dans l'espace mémoire virtuel attribué au process.

5. Processus et Mémoire Win32



- Si un programme est lancé deux fois et s'exécute simultanément, il y a deux processus séparés.
- Un processus possède sa mémoire, des références à des fichiers (handle), des ressources système.

5. Processus et Mémoire Win32

- Espace adressable d'un processus :
  - Chaque processus reçoit un espace virtuel de 4 Gbytes RAM.
  - Ex. Windows 95 : les 2 Gbytes du bas (0 ..7FFFFFFFH) sont privés au processus et contiennent l'image du fichier EXE, variables globales (lecture/écriture), stack, heap, et les DLL non systèmes. Les 2 Gbytes du haut (80000000H..0FFFFFFFFH) sont réservés au système et sont partagés par tous les processus où sont mappés : Noyau Windows, VxD, DLL systèmes etc...

5. Processus et Mémoire Win32

- L'adresse virtuelle est transformée en addresse physique.
  - La mémoire physique est divisée en pages de 4Kbytes.
  - Chaque processus possède son propre répertoire de table de pages qui sera indexée par les 10 bits de poids fort de l'adresse virtuelle.
  - Ceci permet de pointer (sur 20 bits) sur une table de pages que l'on va indexer avec les 10 bits de poids moyen de l'adresse virtuelle.
  - Ceci donne un pointeur (20 bits) sur une page de mémoire physique de 4 KBytes que l'on va indexer avec les 12 bits de poids faible de l'adresse virtuelle.
  - Chaque page est marquée dans la table des pages comme :
    - présente en RAM ou non
    - read-write (pages de data) ou read only (pages de code)

5. Processus et Mémoire Win32

- Toutes les pages ne sont pas forcément chargées en RAM.
- Le manager de mémoire Win32 détermine en fonction de la nécessité d'allouer de la RAM pour d'autres processus de libérer des pages pas utilisées.
- Celle-ci seront sauvées sur le swap file (voir schéma) pour les pages read-write.
- Par contre, les pages de code ne seront pas sauvées dans le swap file, elles seront rechargées en RAM depuis le fichier .EXE ou .DLL lorsque que cela sera nécessaire.

♦ La quantité totale de mémoire libre est déterminée par la somme de mémoire RAM physique libre et de la mémoire libre accessible sur le disque dur pour swapper les pages read/write (stack, heap, data).

♦ Les pages de codes peuvent être partagées par les process, ceci est particulièrement utiles pour les DLL.

♦ Par contre, les pages read/write ne se jamais partagées entre les processus.