Cours IFT6815 - Automne 2005

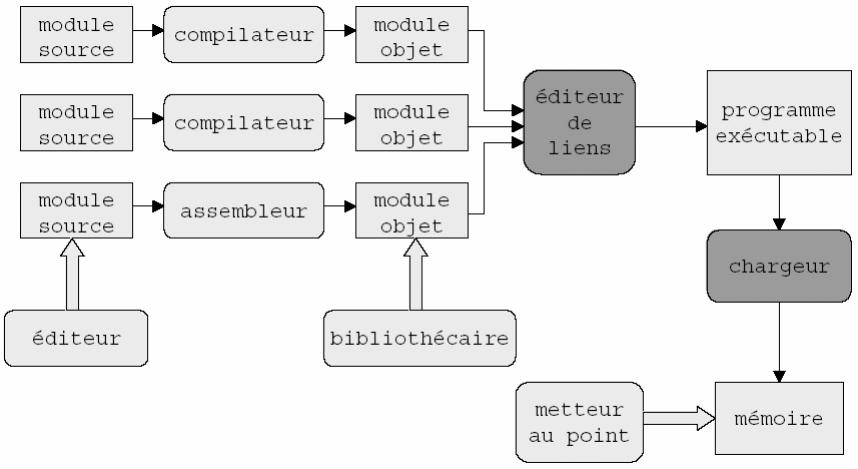
Architecture et systèmes d'exploitation

9. L'édition de liens et le chargement

Professeur:
Victor Ostromoukhov

9. L'édition de liens et le chargement

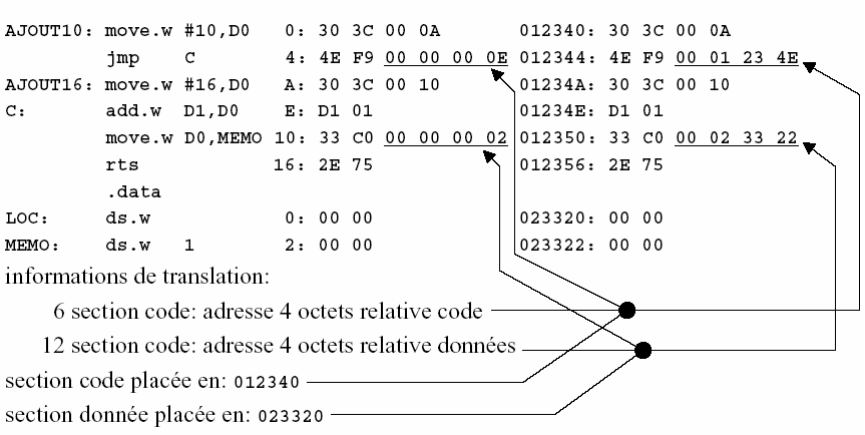
L'édition de liens et le chargement



Notion de module translatable (1)

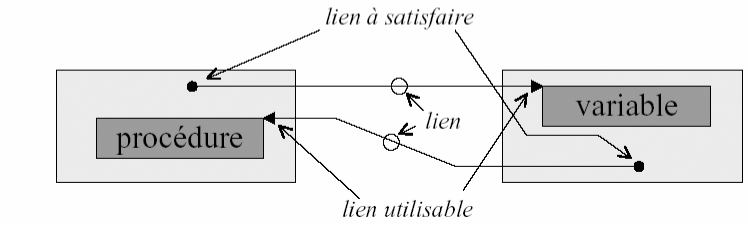
- Emplacements disjoints des modules à l'exécution
- Architecture matérielle (Multics, iAPX432, etc.)
  - adresse = couple <n, d>
    - n désigne l'espace
    - d le déplacement dans l'espace
- En général => pouvoir mettre les modules n'importe où
  - sections différentes: code, données constantes, variables
  - indiquer la nature de la section
  - indiquer les emplacements contenant des adresses relatives à une section
    => ajouter à chacun de ces emplacements l'adresse de début de la section correspondante

Notion de module translatable (2)



Notion de lien

- intermédiaire: *nom* = identificateur du texte source
- catégorie des objets (implicite ou explicite):
  - *internes* => aucun lien
  - *exportés* ou *publics* => lien utilisable
  - *importés* ou *externes* => lien à satisfaire



Exemples d'expression du lien (1)

```
// module a

int         a_global_var1;
float       a_global_var2;
char    *a_global_var3 = "module a";
extern int      b_global_var1;
extern double proc_b1(double par1, double par2);

int main()
{
  b_global_var1 = 10;
  a_global_var2 = b_global_var1 / 5.;
  a_global_var2 = proc_b1((double)a_global_var2, 10.);
  return (0);
} // main

double proc_a1(double par1, double par2)
{
  int a1_var1;
  double a1_var2;
  a1_var1 = par1;
  a1_var2 = a1_var1 * par2 / a_global_var2;
  a1_var2 = proc_a2(a1_var1);
  return(a1_var2);
}

int proc_a2(int par1)
{
  int a2_var1;

  a2_var1 = par1 * a_global_var1;
  return(a2_var1);
}
```

```
// module b

int         b_global_var1;
float       b_global_var2;
extern  float a_global_var2;
extern  double proc_a1(double par1, double
par2);

double proc_b1(double par1, double par2)
{
  int b1_var1 = 5;
  double b1_var2 = 2.;

  b1_var2 = proc_a1(1., b1_var1) *
            par2 / a_global_var2;
  return(b1_var2);
}
```

Exemples d'expression du lien (2)

```
// module a

int         a_global_var1;
float       a_global_var2;
char    *a_global_var3 = "module a";
extern int      b_global_var1;
extern double proc_b1(double par1, double par2);

int main()
{
  b_global_var1 = 10;
  a_global_var2 = b_global_var1 / 5.;
  a_global_var2 = proc_b1((double)a_global_var2, 10.);
  return (0);
} // main

double proc_a1(double par1, double par2)
{
  int a1_var1;
  double a1_var2;
  a1_var1 = par1;
  a1_var2 = a1_var1 * par2 / a_global_var2;
  a1_var2 = proc_a2(a1_var1);
  return(a1_var2);
}

int proc_a2(int par1)
{
  int a2_var1;

  a2_var1 = par1 * a_global_var1;
  return(a2_var1);
}
```

```
> cc -c module_a.c

> strings -a module_a.o
01.01
module a
GCC: (GNU) 2.95.3 20010315 (release)
.symtab
.strtab
.shstrtab
.text
.rel.text
.data
.rel.data
.bss
.note
.rodata
.comment
module_a.c
gcc2_compiled.
a_global_var3
main
b_global_var1
a_global_var2
proc_b1
proc_a1
proc_a2
a_global_var1
>
```

Exemples d'expression du lien (3)

```
// module b

int         b_global_var1;
float       b_global_var2;
extern  float a_global_var2;
extern  double proc_a1(double par1, double
par2);

double proc_b1(double par1, double par2)
{
  int b1_var1 = 5;
  double b1_var2 = 2.;

  b1_var2 = proc_a1(1., b1_var1) *
            par2 / a_global_var2;
  return(b1_var2);
}
```

```
> cc -c module_b.c

> strings -a module_b.o
01.01
GCC: (GNU) 2.95.3 20010315 (release)
.symtab
.strtab
.shstrtab
.text
.rel.text
.data
.bss
.note
.rodata
.comment
module_b.c
gcc2_compiled.
proc_b1
proc_a1
a_global_var2
b_global_var1
b_global_var2
```

Exemples d'expression du lien (4)

```
> cc -o prog module_a.o module_b.o
>

> strings prog
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
__deregister_frame_info
_IO_stdin_used
__libc_start_main
__register_frame_info
GLIBC_2.0
PTRh
module a
>
```

Édition simple (liste de modules)

- Trois étapes
  - placement des sections provenant des modules
    - regroupement selon leur nature (protection):
      - code -> exécution
      - constantes -> lecture
      - variables -> lecture/écriture

    => définition des adresses d'implantations des sections

    ad (section n+1) := ad (section n) + taille (section n)
  - construction de la table des liens utilisables (transformation adresse)
  - construction du programme
    - translation du module
    - résolution des liens à satisfaire

      si pas de lien utilisable pour un lien à satisfaire => non résolu

      anomalie mais non erreur

fusion

Construction de la table des liens

- LU_M liens utilisables du module M
- LAS_M liens utilisables du module M
- $LU_M \cap LU_{M'}$ doubles définitions

rencontre d'un lien utilisable dans M, déjà à l'état utilisable dans la table

- $LAS_L = \bigcup_{M \in L} LAS_M - \bigcup_{M \in L} LU_M$ liens non résolus dans L

état à_satisfaire à la fin

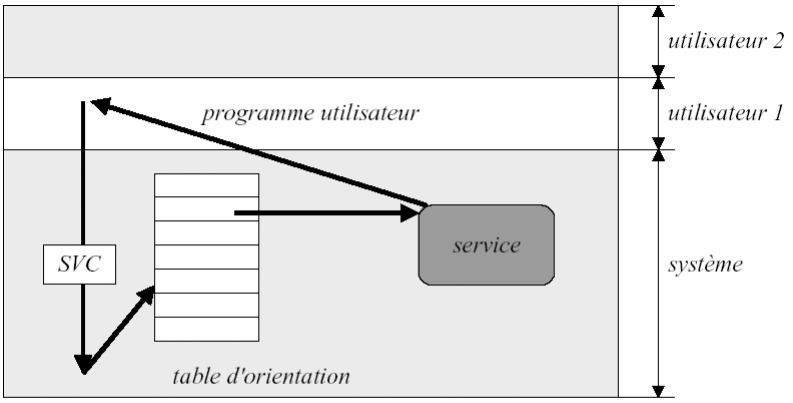
- étape 2: construire la table de tous les liens, avec leur état

nom	état	valeur
A	à_satisfaire	
B	utilisable	3281

Bibliothèque de modules (nature)

- bibliothèque de calcul
  - sous programme standard: sinus, cosinus, racine_carrée, etc.
  - sous programmes liés à un langage
- bibliothèque d'interface système
  - sous programmes exécutant les appels systèmes
  - préparent les paramètres, et exécutent l'appel
  - appel système traité différemment d'un appel de sous programme:
    - contrôles incontournables
    - pas d'établissement de liaison par l'éditeur de liens (indépendance)
    - instruction particulière permettant le changement de mode
- bibliothèque d'utilisateur
  - sous programmes pour des besoins spécifiques

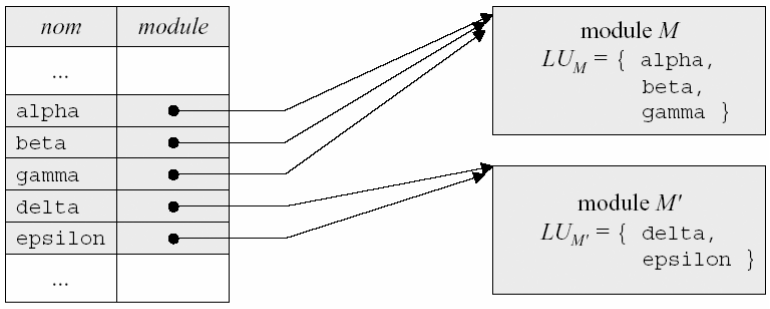
L'appel système



changement mode maître-esclave
contrôle des paramètres
pas de liaisons par l'EdL

INT sur IAPX ?86
TRAP sur M68K

Bibliothèque de modules (structure)



en général, on a: $\forall M, M' \in B, LU_M \bigcap LU_{M'} = \varnothing$

utilisation d'une liste de bibliothèque $B_1, B_2, \ldots, B_p$

Adjonction de modules (1)

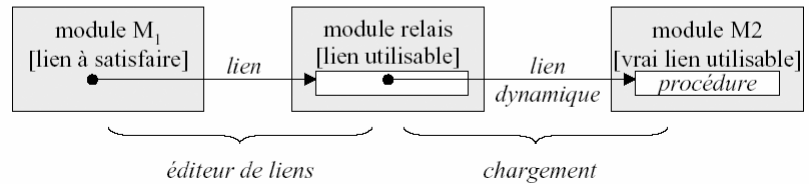
- Pour chaque lien à satisfaire de la table,
  - parcourir $B_1, B_2, \ldots, B_p$ et y rechercher le lien
    - si trouvé, ajouter le module à la liste, et traiter les liens
    - sinon, mettre son état à inexistant
- Nécessité de structure pour l'efficacité de la recherche
- À la fin, $LIN = LAS_L - \bigcup_{M \in L, B_i} LU_M$

Adjonction de modules (2)

- Parcourir les $B_1, B_2, \ldots, B_p$
  - pour chacune, rechercher les liens à satisfaire
    - si trouvé, ajouter le module à la liste, et traiter les liens
    - sinon, mettre son état à inexistant
  - en fin de bibliothèque, remettre inexistant -> à_satisfaire
- À la fin, il est possible que $LAS_L \bigcap \left(\bigcup_{M \in B_i} LU_M \right) \neq \varnothing$
- Ordre à respecter
  - entre bibliothèques:
    L à satisfaire dans M ∈ B_i, utilisable dans M' ∈ B_j => i<j
  - entre les modules d'une même bibliothèque si parcours séquentiel:
    L à satisfaire dans M ∈ B, utilisable dans M' ∈ B => M avant M'

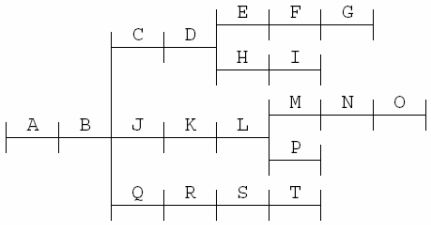
Édition de liens dynamique

- Retarder l'édition de liens jusqu'au chargement ou au moment de l'utilisation
- Permet le partage des modules entre plusieurs programmes
- Soit au niveau module, soit au niveau bibliothèque



Notion de recouvrement

- Programmes de grande taille
  - Mémoire virtuelle de grande taille
    => le système met en mémoire réelle ce qui est nécessaire
  - recouvrement (*overlay*) géré par l'utilisateur
    - regroupement des modules en parties pouvant se recouvrir
    - édition de liens avec recouvrement
    - instructions spécifiques de chargement de ces parties.



Références croisées

- But: savoir qui utilise quoi et où
  - mise au point: limiter la recherche des modules en cause
  - maintenance: quels modules sont concernés par un changement
  - amélioration du découpage en modules
  - organisation des bibliothèques

nom lien	module qui exporte	modules et emplacements qui importent
alpha	M1	M2(10, 50, 86), M3(26)
beta	M2	M1(104), M4(34)
gamma	M3	M4(246, 642)
delta	M4	M5(68,94),M6(4),M7(456,682,624)

Chargement

- Édition de liens => programme exécutable
- Chargement = mise en mémoire et lancement
  - lecture en emplacement fixe => trop restrictive
  - avec translation => mise en mémoire n'importe où
    - informations de translation (traducteurs -> éditeur de liens)
  - adresse de lancement = adresse 1ère instruction à exécuter
    - traducteur reconnaît le programme principal et fournit l'adresse
    - éditeur de lien la transforme en adresse dans programme
  - environnement du programme exécutable => à construire
    - programme autonome = machine nue
    - programme sur machine abstraite = construire la machine

      faire l'édition de liens dynamique, partage de sous programmes, ...

Conclusion

- translation => permettre l'exécution d'un module n'importe où
- lien => information permettant à un module d'accéder à un autre
- édition de liens => 2 phases
  - construction de la table des liens
  - remplacement des liens à satisfaire et construction du programme
- bibliothèque => moyen de compléter une liste de modules
  - sans structure -> ordre des modules est important
  - avec structure -> accès par lien utilisable
- édition de liens dynamique => partager les bibliothèques
- recouvrement => diminuer l'encombrement total du programme
- références croisées => savoir *qui utilise quoi*
- chargement => mise en mémoire, machine abstraite, lancement

Résumé

+ La translation d'un module consiste à modifier son contenu pour qu'il puisse s'exécuter à un endroit différent de celui pour lequel il était prévu initialement.

+ Un lien est l'information qui permet à un module d'accéder à un objet appartenant à un autre module. Il est désigné par un nom. On appelle lien à satisfaire la partie du lien située chez l'accédant, et lien utilisable celle située chez l'accédé.

+ L'expression de la nature d'un lien dépend du langage de programmation et du traducteur. Elle peut prendre des formes variées.

+ L'édition de liens se fait en deux phases. La première construit la table de l'ensembles des liens trouvés dans les modules à relier. La seconde construit le programme exécutable en translatant les modules et en remplaçant les liens à satisfaire par la valeur du lien utilisable correspondant.

+ Entre les deux phases, l'éditeur de liens peut compléter la liste des modules à relier avec des modules provenant de bibliothèques, en tenant compte des liens restant à satisfaire.

+ Une bibliothèque peut être une simple collection de modules parcourue séquentiellement par l'éditeur de liens, mais l'ordre des modules est important. Elle peut être structurée de façon à savoir rapidement quel module, s'il existe, contient un lien donné comme lien utilisable.

+ Une bibliothèque regroupe en général des modules de fonctionnalité voisine. On peut avoir, par exemple, une bibliothèque de calcul, une bibliothèque d'interface système, et des bibliothèques spécifiques aux utilisateurs.

+ Le recouvrement est une technique qui permet de diminuer l'encombrement total en mémoire centrale d'un programme, en décidant que des parties du programme peuvent se trouver au même endroit, car elles ne sont pas utiles au même moment pour la bonne exécution du programme.

+ L'édition des références croisées consiste à éditer une table qui, pour chaque lien trouvé dans une liste de module, donne le module dans lequel il est utilisable, et ceux dans lesquels il est à satisfaire. Elle peut être obtenue en sous-produit de l'édition de liens, ou par un outil spécifique. Elle est utile pour la mise au point et pour la maintenance.

+ Le chargement est l'opération qui consiste à mettre en mémoire centrale un programme exécutable, avec une éventuelle translation, et à en lancer l'exécution. Il consiste également à construire la machine abstraite demandée par le programme.