COMPLIATION

Plan

1. Compilation
2. Préprocesseur
3. Edition de liens
4. Création d'une bibliothèque

Compilation

Processus en 4 étapes, à partir de `nom.c`

1. Préprocesseur : `cpp` produit un fichier `nom.i`;
2. Compilateur : `ccom` produit un fichier `nom.s`;
3. Assembleur : `as` produit un *module objet* `nom.o`;
4. Editeur de lien : `ld` produit un *fichier exécutable*, par défaut `a.out`.

Module objet directement par l'option `-c`:

```
cc -c add.c mult.c
```

produit `add.o` et `mult.o`.

Fichier exécutable de nom `xyz` directement par

```
cc -o xyz add.c mult.c
```

Autres options

`-I` répertoires pour `include`;

`-L` répertoires de bibliothèques;

`-g` pour le débogueur;

`-D` pour définir des pseudo-variables.

Préprocesseur

Rôle:

- macro-définitions: `#define`
- inclusion : `#include`

```
#include <nom.h>
```

Les fichiers inclus sont cherchés

1. dans les répertoires spécifiées dans l'option `-I`
2. dans le répertoire `/usr/include`

```
#include "nom.h"
```

Les fichiers inclus sont cherchés

1. dans le répertoire de travail;
2. dans les répertoires spécifiées dans l'option `-I`;
3. dans le répertoire `/usr/include`.

sauf en cas de chemin absolu:

```
#include "$HOME/include/nom.h"
```

- Pour éviter des inclusions multiples :

```
/* nom.h */
#ifndef NOM_H
#define NOM_H
...
#endif
```

- Usage de l'option `-D`

```
-Dindent
-Dident=valeur
```

Le préprocesseur s'exécute comme si on avait écrit:

```
#define indent
#define ident valeur
```

- Utilisé pour compiler certaines parties:

```
cc -DIBM nom.c
```

Le morceau suivant *est* compilé:

```
#ifdef IBM
...
#endif
```

Edition de liens

Rôle : Création d'un programme exécutable à partir de

- modules objet;
- bibliothèques.

Résolution des références : associer la définition d'une fonction ou variable à une déclaration externe, entre modules ou bibliothèques.

Les *bibliothèques* sont dans `/lib` et `/usr/lib` avec suffixes `.a` ou `.so` notamment.

```
$ ls /usr/lib
libc.a
libm.a
libX11.a
...
```

Utilisation de la bibliothèque `lib`*nom*`.a` par

`-l`*nom*

cherchée dans `/lib` et `/usr/lib`. On spécifie un autre répertoire avec `-L`.
La bibliothèque `libc.a` est toujours incluse.

```
cc -o couleur couleur.c -lm -lX11
```

Edition de lien statique : extrait le code d'une bibliothèque et le recopie dans le programme.

Edition de lien dynamique : localise le code dans la bibliothèque et mémorise sa place; le chargement est fait à l'exécution.
Deux variantes existent :

- liaison immédiate : chargement au lancement.
- liaison différée : chargement à l'appel.

Création d'une bibliothèque

ar *clé nom-archive noms-objet*

La clé prend notamment les valeurs :

`r` remplace des modules et ajoute à la fin;

`v` mode "verbeux";

`t` liste les modules contenus dans l'archive.

Exemple:

```
$ ar t /usr/lib/libm.a
...
acos.o
acosh.o
asin.o
atanh.o
ceil.o
floor.h
...
```

Exemple

1.— Fichier en-tête pour les entiers de Gauss:

```
/* gauss.h */
  typedef struct {
    int re;
    int im;
  } gauss;
  gauss add(gauss,gauss);
  gauss mult(gauss,gauss);
```

2.— Deux fichiers implémentant respectivement l'addition et la multiplication:

```
/* addg.c */
  #include "gauss.h"
  gauss add(gauss a, gauss b) {
    gauss c;
    c.re = a.re + b.re;
    c.im = a.im + b.im;
    return c;
  }
```

et

```
/* multg.c */
  #include "gauss.h"
  gauss mult(gauss a, gauss b) {
    gauss c;
    c.re = a.re * b.re - a.im * b.im;
    c.im = a.re *b.im+ a.im *b.re;
    return c;
  }
```

3.— Compilation *séparée* des deux fichiers:

```
$ gcc -c addg.c
$ gcc -c multg.c
```

4.— Création de la bibliothèque

```
$ ar rv libgauss.a addg.o multg.o
a - addg.o
a - multg.o
$
```

5.— Un fichier de calcul sur les entiers de Gauss:

```
/* calcul.c */
  #include <stdio.h>
  #include "gauss.h"

  main ()
  {
    gauss a = {2, 3};
    gauss b = {-2, 1};
    gauss c;

    c = add (a, b);
    printf ("(%d,%d) + (%d,%d) = (%d,%d)\n",
      a.re, a.im, b.re, b.im, c.re, c.im);
    c = mult (a, b);
    printf ("(%d,%d) * (%d,%d) = (%d,%d)\n",
      a.re, a.im, b.re, b.im, c.re, c.im);
  }
```

6.— Compilation *avec* la bibliothèque **gauss**; il faut spécifier que cette bibliothèque se trouve dans le répertoire courant.

```
$ gcc -o calcul calcul.c -L. -lgauss
```

7.– Exécution

```
$ calcul
(2,3) + (-2,1) = (0,4)
(2,3) * (-2,1) = (-7,-4)
```