TD n° 1 :

Construction

- Exécutable
- Bibliothèques :
  Statique & Dynamique

Compilation se fait en plusieurs phases

- Compilation pour avoir un fichier objet à partir de prog.c:

```
gcc -c prog.c → prog.o
```

- Compilation pour avoir un fichier objet + Worning :

```
gcc -c -Wall prog.c → prog.o
```

- Compilation pour avoir un fichier objet + Worning + Info pour déboguer (ex. gdb) :

```
gcc -c -g -Wall prog.c → prog.o
```

Compilation se fait en plusieurs phases

- Pré compilation à partir de prog.c pour avoir le fichier : prog.i

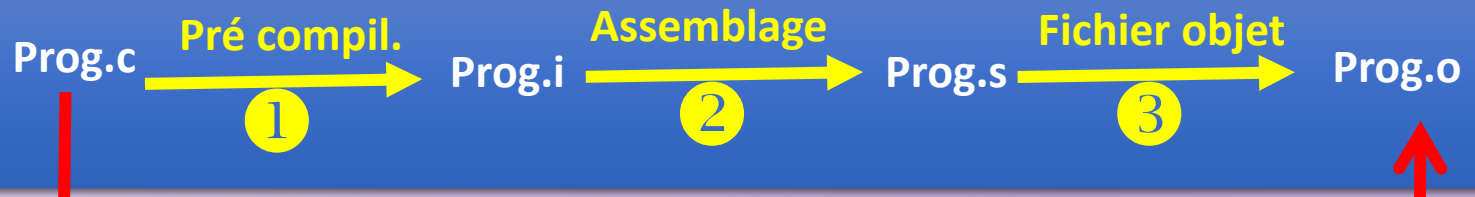
1 gcc -E -o prog.i prog.c → prog.i

- Compilation à partir de prog.i pour avoir un fichier assembleur : prog.s

2 gcc -S -o prog.s prog.i → prog.s

- Compilation à partir de prog.s pour avoir un fichier objet : prog.o

3 gcc -c -o prog.o prog.s → prog.o



gcc -c prog.c → prog.o

Compilation & déboguer

- Déboguer : Afficher code + données de prog.o

```
objdump -s prog.o
```

- Déboguer : Donner assembleur du code de prog.o

```
objdump -d prog.o
```

- Déboguer : Affiche la table des symboles

```
objdump -t prog.o
```

Edition de liens

- Edition de liens pour avoir l'exécutable prog à partir de prog.o

```
gcc  prog.o  -o  prog
```

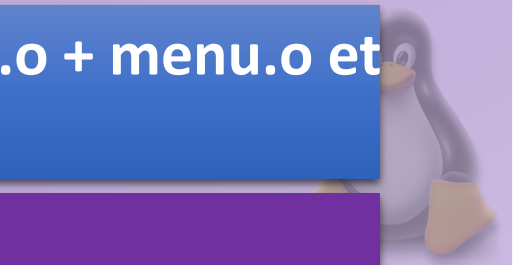
- Edition de liens pour avoir l'exécutable prog à partir de prog.o et menu.o

```
gcc  prog.o  menu.o  -o  prog
```

- Edition de liens pour avoir l'exécutable prog à partir de prog.o + menu.o et la bibliothèque libmath.so

```
gcc  prog.o  menu.o  /usr/lib/libmath.so  -o  prog
```

- Exécution : prog ou ./prog



Edition de liens

- Utiliser -L pour le chemin de la bibliothèque

```
gcc  prog.o  menu.o  -L /usr/lib libmath.so -o  prog
```

- Si le nom de la bibliothèque est libxxx .a ou libxxx .so Utiliser –l pour indiquer le nom de la bibliothèque xxx :

```
gcc  prog.o  menu.o  -lxxx  -o  prog
```

- Le nom de la bibliothèque mathématique est libm.a ou libm.so Utiliser -l pour indiquer le nom de la bibliothèque m :

```
gcc  prog.o  menu.o  -lm  -o  prog
```

Exemple de création de bibliothèques statique et dynamiques

Le dossier **src** contient les fichiers suivants :

pile.h contient le type et les spécifications fonctionnelles de la pile.

pile.c contient les fonctions de gestion de la pile.

file.h contient le type et les spécifications fonctionnelles de la file.

file.c contient les fonctions de gestion de la file.

essai.c inclus pile.h et file.h

Deux dossiers vides :

biblios pour stoker la bibliothèque statique.

bibliod pour stoker la bibliothèque dynamique.

bibliothèque statique nommée libpf.a partir de pile.o et file.o

- Création de fichiers objets : pile.o et file.o

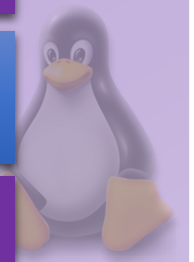
```
gcc   -c   -Wall   pile.c   →   pile.o
gcc   -c   -Wall   file.c   →   file.o
```

- Création de bibliothèque statique libpf.a à partir de pile.o et file.o

```
ar -r libpf.a pile.o file.o
```

- Déplacement de la bibliothèque libpf.a dans ./biblios

```
mv libpf.a ./biblios/libpf.a
```



Compilation & éditions de liens de essai.c

- Compilation de essai.c

gcc -c -Wall essai.c → essai.o

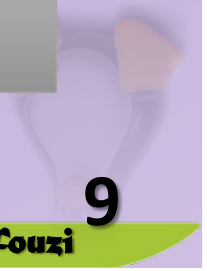
- Edition de liens de essai.o et libpf.a

gcc essai.o ./biblios/libpf.a -o essai → essai

- Edition de liens de essai.o et libpf.a avec utilisation de : –L et -l

gcc essai.o -L./biblios -lpf -o essai → essai:

- Exécution : essai ou ./essai



bibliothèque dynamique nommée libpf.so partir de pile.o et file.o

- Création de fichiers objets : pile.o et file.o

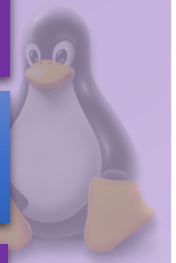
```
gcc  -Wall  -c  -fPIC  pile.c  →  pile.o
gcc  -Wall  -c  -fPIC  file.c  →  file.o
```

- Création de bibliothèque dynamique libpf.so à partir de pile.o et file.o

```
gcc  -shared  -fPIC  pile.o  file.o  -o  libpf.so
```

- Déplacement de la bibliothèque libpf.so dans ./bibliod

```
mv  libpf.so  ./bibliod/libpf.so
```



Compilation & éditions de liens de essai.c

- Compilation de essai.c

gcc -c -Wall -fPIC essai.c → essai.o

- Edition de liens de essai.o et libpf.so

gcc -fPIC essai.o ./bibliod/libpf.so -o essai → essai

- Edition de liens de essai.o et libpf.so avec utilisation de : –L et -l

gcc -fPIC essai.o -L./bibliod -lpf -o essai → essai:

- Que faut-il faut ajouter dans l'environnement ?

```
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:./bibliod:
export LD_LIBRARY_PATH
```

- Exécution : essai ou ./essai

Bibliothèque statique ou dynamique

Une bibliothèque statique (d'extension .a) est une bibliothèque qui sera intégrée à l'exécutable lors de la compilation.

- **Autonome**
- **Se comporte comme un fichier objet**
- **L'exécutable est volumineux**
- **Chargée plusieurs fois en mémoire**

Une bibliothèque dynamiques (-.so Sharing Object : objet partagé sous linux) Ou (.dll Dynamics Link Library sous Windows)

- **Non intégrée à l'exécutable lors de l'édition de liens**
- **Elle est chargée une seule fois en mémoire**
- **L'exécutable est plus léger par rapport à la bibliothèque statique**
- **On peut (sous conditions) la mettre à jour sans recompiler le programme**
- **Elle se charge une seule fois en mémoire**
- **La version dynamique va provoquer l'inclusion de quelques instructions pour charger la bibliothèque**
- **L'option -fPIC (Position Independent Code) compile sans indiquer d'adresse mémoire dans le code**

Les erreurs communes

- On ne compile JAMAIS un .h : gcc -c menu.h
  C'est aussi stupide et incohérent

- Un .h est un fichier d'inclusion. CE N'EST PAS UNE BIBLIOTHEQUE !

- L'éditeur de liens n'a jamais besoin d'accéder aux fichiers d'inclusion nécessaires à la compilation

  gcc main.o menu.o menu.h -o prog :
  mettre menu.h va faire planter l'édition de liens

- Une bibliothèque .a ou .so est une collection de fichiers objet .o
  CE N'EST PAS UN INCLUDE !

- Le compilateur n'a jamais besoin d'accéder aux bibliothèques contenant des fonctions nécessaires à l'exécution

  gcc -c prog.c -lm : le -lm est totalement inutile

Exemple de création de bibliothèques statique et dynamique
Il faut le programmer

Le dossier **src** contient les fichiers suivants :

op.h contient les spécifications fonctionnelles des 5 opérations arithmétiques et le calcule de la racine carré d'un réel.

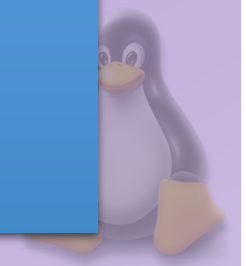
op.c contient Entête et corps de chaque fonction du fichier op.h

essai.c **inclus** **op.h** **et** teste toutes les fonctions de op.h

Deux dossiers vides :

biblios pour stoker la bibliothèque statique.

bibliod pour stoker la bibliothèque dynamique.



Exemple de création de bibliothèques statique et dynamique
Il faut le programmer

Voici le contenu du fichier op.h : il manque le type de retour et le (s) type (s) de (s) paramètre (s)

```
???     add ( ??? )      // Fonction : Addition de 2 entiers et retourne un entier
???     sous ( ??? )     // Fonction : Soustraction de 2 entiers et retourne un entier
???     mul ( ??? )      // Fonction : Multiplication de 2 entierset retourne un entier
???     div ( ??? )      // Fonction : Division de 2 entiers et retourne un entier
???     res ( ??? )      // Fonction : Reste de la division de 2 entiers et retourne un entier
???     racine ( ??? )   // Fonction : Racine carré d'un réel et retourne un réel
```

1/ Donner le texte de : op.h et op.c
2/ Construire une bibliothèque statique nommée libop.a sous biblios
3/ Construire une bibliothèque dynamique nommée libop.so sous bibliod
4/ Donner le texte de : essai.c
5/ Construire et tester l'exécutable (sans –L et –l) nommé essai1 en fonction de libop.a
6/ Construire et tester l'exécutable (avec –L et –l) nommé essai11 en fonction de libop.a
7/ Construire et tester l'exécutable (sans –L et –l) nommé essai2 en fonction de libop.so
8/ Construire et tester l'exécutable (avec –L et –l) nommé essai22 en fonction de libop.so

Notes :

A vous de reproduire sur machine la solution de l'exemple sans faire copier/coller

FIN TD n°1