Cours de compilation
TD numéro 1

Jean Méhat

24 septembre 2015

On partira avec deux fichiers ; l'un contient une fonction `pgcd` que nous verrons dans un chapitre ultérieur et l'autre une fonction `main` qui sert à appeler cette fonction. Le fichier `pgcd.c` contient

```
/* pgcd.c
 */
int
pgcd(int a, int b){
        int t;
        while(a != 0){
                if (a < b){
                        t = b;
                        b = a;
                        a = t;
                }
                a -= b;
        }
        return b;
}
```

Le fichier `main.c` contient

```
/* main.c
 */
# include <stdio.h>

int
main(int ac, char * av[]){
        if (ac < 3){
                fprintf(stderr, "usage: %s N M\n", av[0]);
                return 1;
         }
        printf("%d\n", pgcd(atoi(av[1]), atoi(av[2])));
        return 0;
}
```

Produire un exécutable et l'exécuter avec les commandes

```
$ gcc main.c pgcd.c
$ a.out 91 28
```

On obtient un message d'erreur si on essaye de produire un exécutable à partir d'un seul de ces fichiers avec

```
$ gcc pgcd.c
$ gcc main.c
```

Question 1 : Quel sont les messages d'erreur ? À quelle étape de la compilation sont-ils produits ?
Compiler chaque fichier indépendamment pour obtenir un fichier .o

```
$ gcc -c main.c
$ gcc -c pgcd.c
```

Examiner avec la commande `nm` la table des symboles des fichiers `.o` produits par ces commandes. Dans la sortie de `nm`, chaque symbole occupe une ligne ; si le symbole est marqué comme `U`, c'est qu'il reste à définir ; s'il est marqué comme `T` c'est qu'il s'agit de code à exécuter et on voit sa valeur.
Question 2 : Quels sont les symboles indéfinis dans chacun des fichiers ?
Combiner les deux fichiers `.o` pour avoir un exécutable et l'exécuter avec

```
$ gcc main.o pgcd.o
$ a.out 85 68
```

Question 3 : Qu'imprime la commande `a.out 85 68` ?
La commande `nm` permet aussi d'examiner la table des symboles du fichier exécutable.
Question 4 : Quels sont les symboles indéfinis dans le fichier `a.out` obtenu précédemment ?
Question 5 : A quelles adresses se trouvent les fonctions `main` et `pgcd` ?
Sauver le fichier a.out sous le nom `dyn.out` et compiler le programme sous la forme d'un programme qui utilise les bibliothèques statiques sous le nom `stat.out` avec

```
$ mv a.out dyn.out
$ gcc -static pgcd.c main.c -o stat.out
```

Question 6 : Quelles sont les tailles respectives de `dyn.out` et de `stat.out` ?
Question 7 : Le `T` majuscule dans la sortie de `nm` permet à peu près d'identifier les fonctions. Combien y a-t-il de fonctions présentes dans chacun des deux exécutables ?
Question 8 : La commande `ldd` permet de déterminer les librairies dynamiques nécessaires à un programme. Indiquer les librairies nécessaires pour les deux fichiers exécutables.
Question 9 : Recompiler les programmes avec l'option `-verbose` de Gcc de manière à voir toutes les étapes de la production de l'exécutable.
On va maintenant refaire toutes les étapes en appelant chaque commande directement au lieu de passer par Gcc.
On peut appeler le préprocesseur soit avec la commande `cpp` (comme *C PreProcessor* ; il y a parfois un problème avec certains systèmes où le compilateur C++ porte ce nom) ou avec l'option `-E` de Gcc.
Faire passer le préprocesseur C sur chacun des fichiers C, placer la sortie dans des fichiers nommés `x` et `y`.
Question 10 : Combien les fichiers `x` et `y` comptent-ils de lignes ? D'où viennent ces lignes ?
On peut trouver le compilateur Gcc proprement dit, avec l'analyse lexicale, l'analyse syntaxique et la génération de code dans un fichier qui porte le nom `cc1`. On peut le trouver avec l'une des deux commandes

```
$ find / -name cc1 -print
$ locate */cc1
```

Question 11 : Où se trouve le fichier `cc1` sur votre ordinateur ?
Le programme `cc1` peut lire le programme sur son entrée standard et place dans ce cas la sortie dans un fichier qui porte le nom `gccdump`. On peut donc compiler la sortie de l'étape précédente avec

```
$ cc1 < x
$ mv gccdump.s x2
$ cc1 < y
$ mv gccdump.s y2
```

Question 12 : Que contiennent les fichiers `x2` et `y2` ?
Il faut ensuite faire traiter le contenu des fichiers `x2` et `y2` par la commande `as`, qui place sa sortie dans un fichier nommé `a.out`. On peut donc faire

```
$ as x2
$ mv a.out x3
$ as y2
$ mv a.out y3
```

Question 13 : Que contiennent les fichiers `x3` et `y3` ?
Il est aussi possible de pratiquer l'édition de lien directement sans passer par la commande gcc mais ça devient franchement pénible. On revient donc au programme principal pour la dernière étape.

```
$ mv x3 x.o
$ mv y3 y.o
$ gcc x.o y.o
```

Question 14 : Examiner le fichier `a.out` produit avec la commande `readelf` (pas de correction).
Question 15 : Trouver sur votre ordinateur tous les fichiers qui portent un nom du type `crt*.o`.
Question 16 : Trouver sur votre machine la librairie C standard statique (qui s'appelle `libc.a`).
Question 17 : Examiner le contenu de `libc.a` avec la commande `ar t` (pas de correction).
Question 18 : Extraire, de nouveau avec `ar`, le fichier `printf.o` de `libc.a`. Combien occupe-t-il d'octets ?