

# Première séance machine

## Notions de C, compilation, débogage

Ensimag 1A Apprentissage, Logiciel de Base

2010

### 1 Premiers pas en C

Dans votre éditeur de texte préféré, tapez (ou recopiez ...) le programme suivant (dans un fichier `hello.c`) :

```
#include <stdio.h>

int main (void) {
    int age;
    puts("Bonjour,");
    puts("Quel age avez vous ?");
    scanf ("%d", &age);
    printf("Vous avez %d ans.\n", age);
    return 0;
}
```

(la fonction `puts` affiche une chaîne à l'écran, puis passe à la ligne. La fonction `printf` affiche la chaîne donnée en argument, dans laquelle on remplace les `"%d"` par les arguments suivant la chaîne)

Compilez-le, puis exécutez-le :

```
telesun> gcc hello.c -o hello
telesun> ./hello
[...]
```

### 2 Exploration de la chaîne de compilation

#### 2.1 Compilation

Il s'est en fait passé beaucoup de choses dans les deux commandes précédentes. La première étape est la compilation proprement dite, on peut arrêter `gcc` après cette étape avec l'option `-S` :

```
telesun> gcc -S hello.c
```

Ceci va générer un fichier `hello.s`. Ouvrez-le dans un éditeur de texte. Vous avez devant vous un programme en langage d'assemblage. On trouve dedans :

- Des définitions de données (les chaînes qui apparaissent dans le programme)
- Des instructions. En particulier, on reconnaît les appels à la fonction `puts`, qui sont sous la forme `call puts`, entourées d'incantations magiques.

- Des étiquettes, par exemple, `main` :

A ce stade, le programme ne dit rien sur l'implémentation des fonctions `puts`, `printf` et `scanf`. On se contente de supposer qu'elles existent (on cherchera leur implémentation plus tard, c'est l'édition de liens).

## 2.2 Assemblage

Le langage d'assemblage décrit précisément chaque instruction à exécuter, mais elles ne sont pas dans un format directement exécutable par le processeur. L'étape suivante est d'assembler les instructions, on peut le faire avec la commande :

```
telesun> gcc -c hello.s
```

Ce qui va créer un fichier `hello.o`. Essayez de l'ouvrir dans un éditeur de texte pour vous convaincre que c'est du binaire illisible. On reconnaît quand même quelques chaînes de caractères. On peut afficher le fichier de manière un peu raisonnable avec des commandes comme :

```
telesun> hexdump hello.o
[...]
telesun> hexdump -c hello.o
[...]
```

Mais des outils spécialisés peuvent aussi extraire quelques données intéressantes du fichier. Par exemple, la commande `nm` liste les symboles définis ou utilisés dans le fichier (l'équivalent des étiquettes, mais après assemblage) :

```
$ nm hello.o
00000000 T main
          U printf
          U puts
          U scanf
```

On voit que le symbole `main` est défini («T» veut dire «Text», on verra plus tard pourquoi). En revanche, les symboles `printf`, `puts` et `scanf` sont toujours indéfinis («U» pour «Undefined»).

La traduction du langage d'assemblage vers le binaire garde la quasi-totalité des informations du fichiers assembleur, mais change de représentation. On peut retrouver le texte assembleur avec un désassembleur, par exemple :

```
$ obdjump -d hello.o
```

## 2.3 Édition de lien

L'étape suivante est surtout intéressante lorsqu'on fait de la compilation séparée, mais il se passe aussi des choses avec un seul fichier : on «prépare» le fichier, pour qu'il soit prêt à être exécuté :

```
telesun> gcc hello.o -o hello
```