Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux dirigés 2 : affectation.

1 Un premier programme C

Programme C

```
1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3
4  /* Declaration des constantes et types utilisateurs */
5
6  /* Declaration des fonctions utilisateurs */
7
8  /* Fonction principale */
9  int main()
10 {
11     /* Declaration et initialisation des variables */
12     int x = 5;
13     int y;
14
15     y = 2;
16     x = y;
17
18     /* valeur fonction */
19     return EXIT_SUCCESS;
20 }
21
22 /* Definitions des fonctions utilisateurs */
```

Traduction

```
1   valeur 2 r0
2   ecriture r0 11
3   lecture 11 r0
4   ecriture r0 10
5   stop
6
7
8
9
10  5
11  ?
```

FIGURE 1 – Un programme C et sa traduction machine

Ligne	x	y	sortie
initialisation	5	?	
15		2	
16	2		
19	renvoie `EXIT_SUCCESS`		

(a) Trace en C

Instructions	Cycles	CP	r0	10	11
Initialisation	0	1	?	5	?
`valeur 2 r0`	1	2	2		
`ecriture r0 11`	2	3			2
`lecture 11 r0`	3	4	2		
`ecriture r0 10`	4	5		2	
`stop`	5	6			

(b) Trace amil

FIGURE 2 – Traces

Voici, figure 1, un premier programme C. Le langage C sera présenté au prochain cours. Pour le moment voici ce que vous avez à savoir.

Un programme en langage C est donné sous forme de texte, *le source*, et il doit passer par une étape de traduction avant de pouvoir être exécuté par le processeur. Nous donnons ici une traduction sous forme d'instructions amil. Il s'agit d'un artifice pédagogique, la traduction réelle en code binaire exécutable est plus compliquée. Par analogie avec la musique, le source est la partition, et le fichier exécutable est le morceau musical (codé sur le support adapté au système de lecture : un fichier mp3, un CD, etc.). La traduction est effectuée par un ensemble de programmes (et non par des musiciens), le source doit donc obéir à des règles syntaxiques précises.

Les textes entre `/*` et `*/` sont des *commentaires*, ils ne feront pas partie du programme exécutable, ils servent aux humains qui manipulent les programmes. Les commentaires du programme figure 1 vous serviront au cours du semestre à structurer tous vos programmes C.

Tout programme C comporte une fonction principale, le `main()`, qui sert de point d'entrée au programme. Cette fonction doit se terminer par l'instruction `return EXIT_SUCCESS`. En renvoyant cette valeur, le `main` signale au système d'exploitation la terminaison correcte du programme.

Une *variable* est un symbole (habituellement un nom) qui renvoie à une position de mémoire dont le contenu peut prendre successivement différentes valeurs pendant l'exécution d'un programme [1].

L'instruction `int x = 5` déclare une variable `x` et fixe sa valeur initiale à 5. Le mot clé `int` signifie que cette variable contiendra un entier. Dans le code amil correspondant `x` renvoie à l'adresse `10` où se trouve intialement la valeur `5`.

L'instruction `int y` déclare une variable entière `y` sans l'initialiser. L'effet de cette déclaration est de réserver un espace mémoire pour y stocker un entier.

Le signe égal (`=`) a un sens bien particulier, il dénote une *affectation*. L'objectif de ce TD est de bien comprendre l'affectation. La partie à gauche du signe égal doit désigner un espace mémoire, c'est typiquement une variable. La partie à droite du signe égal est une expression dont la valeur sera évaluée et écrite à l'adresse à laquelle renvoie la partie gauche. Par exemple, `y = 2` a été traduit en code machine par une instruction évaluant l'expression `2` dans un registre (ici `valeur 2 r0`), et par une instruction d'écriture de la valeur trouvée dans la mémoire réservée à `y`. Une variable s'évalue comme sa valeur (celle contenue dans la mémoire correspondante, au moment de l'évaluation).

1.1 Questions

1. Quel espace mémoire a été reservé pour `y` dans le code amil ?
2. Comment a été traduite l'instruction d'affectation `x = y` en amil ?
3. Si il y avait `y = x + 2`, ligne 15 dans le programme C, à la place de `y = 2`, quel serait le code amil correspondant ?

2 Échange de valeurs

2.1 Introduction au problème

Nous avons deux tableaux anciens, chacun accroché à un clou, et un troisième clou, libre, sur le mur d'une exposition. Pour des critères esthétiques, nous voulons changer de place nos deux tableaux sans les mettre par terre, car cela risquerait d'abîmer nos précieuses toiles... Comment faire ?

2.2 Échange des valeurs de deux variables en C

1. Écrire un programme C qui déclare et initialise deux variables entières `x` et `y` et effectue la permutation de ces deux valeurs. Commencer par écrire un algorithme, à l'aide de

1. `http://fr.wikipedia.org/wiki/Variable`

FIGURE 3 – Échanger les deux tableaux en utilisant le clou libre...

phrases telles que « Copier la valeur de la variable ... dans la variable ...», en vous inspirant de la question précédente.

2. Traduire ce programme C en un programme amil. On supposera que les deux variables sont stockées aux adresses `10` et `11`.
3. Exécuter ces programmes sur un exemple (faire les traces).
4. (Facultatif). Donner d'autres solutions en assembleur à ce problème (la permutation des contenus des adresses `10` et `11`).
5. (Facultatif). Mêmes questions que précédemment mais pour faire une permutation circulaire de 3 valeurs.

Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux pratiques 2 : premiers pas en langage C

Vous allez mettre tous vos programmes écrits dans ce TP dans le répertoire TP2 :

1. À partir du début de votre arborescence, créez le répertoire TP2 : `mkdir TP2`
2. Allez dans ce répertoire pour y mettre des fichiers : `cd TP2`
3. Créez un nouveau fichier source pour le langage C : `gedit bonjour.c &`

Ce premier programme devra afficher `Bonjour !`, vous le composerez en recopiant le programme donné en exemple dans le TD, sauf les lignes 12 à 16. Gardez tous les commentaires vous en aurez besoin plus tard. Votre éditeur de texte vous assistera en colorisant automatiquement le code saisi.

Pour réaliser l'affichage vous utiliserez l'instruction :
`printf("Bonjour !\n");`, dans votre fonction principale. Le `printf` est une *fonctionnalité supplémentaire* d'entrée sortie, pour qu'il fonctionne il faut insérer la ligne suivante après la ligne 2 :

```
#include <stdio.h> /* pour printf */
```

Le `\n` représente un saut de ligne.

4. Après avoir fini d'écrire votre programme, enregistrez le.
5. Créez un programme exécutable à partir de votre fichier source, cette étape, dite de compilation, sera expliquée au prochain cours : `gcc -Wall bonjour.c -o bonjour.exe`
6. Quand l'étape précédente a réussi, exécutez le programme pour vérifier qu'il fonctionne : `bonjour.exe` (ou `./bonjour.exe`).

Vous répéterez ces trois dernières étapes (écrire/sauvegarder/compiler exécuter), très souvent ce semestre et le suivant. Exercez-vous à tout faire avec les raccourcis clavier *sans utiliser la souris*.

7. Modifiez le programme de manière à ce qu'il affiche votre prénom après le «Bonjour» (compiler/exécuter).