Université Paul Sabatier
Feuille 1 : Environnement de travail et premiers programmes C

1 Préléminaires

Nous commençons par présenter brièvement l'environnement de travail et la procédure de compilation des programmes.

1.1 Procédure de connexion

Le système d'exploitation que nous utilisons pour les TP est Linux Fedora.

Pour se connecter (≪logger≫) sous linux, deux modes sont possibles : le mode console (texte), et le mode graphique (avec des fenêtres).
Par défaut, il y a 6 consoles et un mode graphique. Pour aller du mode graphique à la console, taper CTRL+ALT+Fi où i varie de 1 à 6. Pour repasser en mode graphique, taper CTRL+ALT+F7 ou ALT+F7. Pour passer d'une console à l'autre, taper CTRL+ALT+Fi où i est la console où vous désirez aller.

Nous nous connectons en mode graphique.

1.2 Déconnexion

Pour se déconnecter, il faut choisir l'item déconnexion du menu "Fedora" en bas à gauche de l'écran, puis choisir le bouton éteindre l'ordinateur. A partir de là **on ne touche plus à rien**, la machine s'éteint seule sans action sur le bouton marche/arrêt.

2 La compilation d'un programme C

Nous utiliserons le compilateur gcc de Gnu. Le comportement de ce compilateur peut être modifié à l'aide d'options. Nous utiliserons SYSTEMATIQUEMENT l'option *-Wall* ou l'option *pedantic* pour rendre la compilation stricte et faciliter la recherche d'erreurs.

La compilation se fait en deux phases :

La compilation propement dite qui consiste à traduire le code C en code objet (exécutable) en gardant les références symboliques.

L'édition de liens qui consiste à remplacer dans le code objet les références symboliques (sous programmes, itérations, etc.) par les adresses réelles dans ce code.

Des erreurs peuvent survenir dans l'une ou l'autre de ces phases. Les erreurs de compilation sont anoncées par un message et un numéro de ligne précédés des mots :

error (l'erreur est fatale) ou *warning* (l'erreur n'empêche pas la génération de code objet). Les erreurs d'édition de liens sont annoncées par un message précédé des lettres *ld* :

La compilation se fait dans une fenêtre terminal à l'aide de la commande :

```
gcc -Wall <nom_du_fichier.c>
```

par exemple :

```
gcc -Wall hello.c
```

S'il n'y a pas d'erreur le code exécutable correspondant est nommé *a.out*. Pour lui donner un autre nom on doit utiliser l'option -o (pour output)

```
gcc -Wall hello.c -o hello
```

3 Programmation en C

Nous n'introduisons ici que le strict minimum des diverses notions et instructions abordées.

3.1 Structure d'un programme C

Un programme C est constitué :
- D'instructions au préprocesseur C (*#define*, *#include*)
- De procédures ou fonctions
- D'une procédure ou fonction ayant le statut de programme principal et appelée *main*.

Le minimum vital pour un programme C consiste en une procédure *main*. Les instructions correspondant à cette procédure sont délimitées par une marque de début : "{", et une marque de fin : "}". Toute instruction est terminée par un point virgule (";"). Toute utilisation de variable nécessite une déclaration au moyen d'un type prédéfini.

3.2 Les entrées/sorties (E/S)

3.2.1 Saisie

scanf (≺format≻,liste d'adresses) ;

Spécificateur	type des arguments	format de lecture
d	int	décimal
u	unsigned int	décimal
f	float	virgule flottante
lf	double	virgule flottante
c	char	un seul caractère

float x ; **int** n ; **char** c='A' ;

Instruction	Saisie	Résultat	Commentaire
scanf("%c",&c) ;	P↩	c = 'P'	le premier caractère tapé est stocké
scanf("%c",&c) ;		c = '↩'	la machine prend le CR sans attendre !
scanf("%f",&x) ;	1658.32↩	x=1658.32	
scanf("%d",&n) ;	132↩	n=132	
scanf("%d",&n) ;	13␣2↩	n=13	Le␣`est considéré comme une fin de saisie`
scanf("%d",&n) ;		n=2	la machine prend le 2 resté en attente dans le buffer

Nous allons considérer pour l'instant que le caractère "&" est obligatoire. Les TD et TP sur le passage de paramètres éclaiciront les choses dans le futur.

3.2.2 Affichage

printf (≺format≻,liste d'expressions) ;

format peut contenir :

des caractères ordinaires simplement recopiés

des spécificateurs de conversion :

Spécificateur	type des arguments	format de sortie
d,i	int	décimal
u	unsigned int	décimal
f	float,double	virgule flottante
e,E	float,double	notation exponentielle
c	char,int	un seul caractère

Attention, l'affichage n'est effectif qu'aprés un retour à la ligne : \n

float x=1.23 ; **int** n=3 ; **char** c='P' ;

Instruction	Sortie	Commentaires
printf("%c\n",c) ;	P↩	
printf("code de'%c'=%d",c,c) ;	code de 'P'=80	80 est le code ASCII de 'P'
printf("%d\n",n) ;	3↩	
printf("%4d\n",n) ;	3↩	affichage sur 4 caractères cadrage à droite
printf("%-4d\n",n) ;	3↩	affichage sur 4 caractères cadrage à gauche
printf("Reel=%f\n",x) ;	Reel=1.230000↩	par défaut 6 décimales
printf("Reel=%3.1f\n",x) ;	Reel=1.2↩	3 caractères en tout dont 1 décimale
printf("%+-10.1e",x) ;	+1.2e+00	10 caractères en tout dont 1 décimale

4 Exercices de programmation : la sélection

◇ Exercice **1** : Ecrire un programme qui demande à l'utilisateur un caractère, et qui affiche son code ASCII

◇ Exercice **2** : Ecrire un programme qui lit un entier au clavier puis qui affiche son carré, sa racine carrée et son exponentielle. Les deux derniers résultats sont des réels, ils seront affichés

avec une précision de 10^{-4}.
Vous utiliserez les fonctions de la librairie math.h de prototype :

double sqrt (**double**)

double exp (**double**)

Exemple d'exécution :
Entrer un entier n : 5

5*5=25 ; racine(5)=2.2361 ; exp(n)=1.484132 E2

Remarque : en C, on peut convertir une valeur d'un type à un autre en mettant le nouveau type entre parenthèses avant la valeur.Par exemple, si x et y sont de type float, on peut ecrire : x=(**float**) sqrt ((**double**) y) ;

Penser à modifier la ligne de compilation pour indiquer à la machine que la librairie math va être utilisée :

gcc *nom_source.c* **-Wall -o** *nom_executable_sans_extension* **-lm**

◇ Exercice **3** : Ecrire un programme qui demande un entier compris entre 0 et 127 et qui affiche la nature du caractère correspondant :

$\{0 \cdots 31\} \cup \{127\}$: caractères de contrôle non affichables

$\{65 \cdots 90\}$: alphabet latin majuscule

$\{97 \cdots 122\}$: alphabet latin minuscule

$\{48 \cdots 57\}$: chiffres $0 \cdots 9$

sinon : divers caractères affichables : ponctuation,{,} $\cdots$

Exemple d'exécution :
Entrer un entier compris entre 0 et 127 : 68
C'est le caractère 'D' de l'alphabet latin majuscule.

◇ Exercice **4** : Le calendrier grégorien est le calendrier actuellement utilisé dans la majeure partie du monde. Conçu pour corriger la dérive séculaire du calendrier julien, sa dénomination porte le nom de son instigateur Grégoire XIII, pape de 1572 à 1585. Son point de départ, l'an 1, est une estimation de la naissance de Jésus.
Depuis l'instauration du calendrier grégorien, sont bissextiles, les années : divisibles par 4 mais non divisibles par 100 ou divisibles par 400

Ecrire un programme qui détermine si une année saisie au clavier est bissextile.

◇ Exercice **5** : Exécutez le programme suivant, que se passe t-il ?

```
#include <stdio.h>

int main()
{
int a=10;
while (a!=0)
{
```

```
a=a*a ;
printf("%d\n",a) ;
}
return (0) ;
}
```

Recherchez dans */usr/include/* le fichier : limits.h. Expliquez le comportement du programme précédent par la valeur de

`__INT_MAX__`

◇ Exercice **6** : Ecrivez un programme de résolution d'une équation du second degré à coefficients réels saisis par l'utilisateur. Discuter en fonction des coefficients : $ax^2 + bx + c$ (envisager tous les cas, par exemple si $a = b = 0,\ c \neq 0$, il n'y a pas de solutions...)

version 1 : solutions réelles (vu en TD)

version 2 : solutions complexes

version 3 : ajouter une boucle demandant à l'utilisateur s'il veut résoudre une nouvelle équation ou s'arrêter.