

TP n° 1 : Exécution de programmes C : quelques problèmes.

0. Introduction

Les objectifs de ces TP sont les suivants :

- se familiariser avec l'utilisation du compilateur C d'Intel (ICC) sous Linux, notamment pour mesurer de manière relativement précise les temps d'exécution de programmes.
- Observer l'impact de certaines options de compilation
- Mettre en évidence un certain de conséquences de l'architecture matérielle sur les résultats obtenus, soit en terme de précision des résultats (cas des exercices 1 et 2 sur les nombres flottants), soit en terme de temps d'exécution (autres exercices) en fonction des options de compilation et/ou des choix de programmation au niveau du code C

Ce TP est à effectuer sur des PC équipés de processeurs Intel (Pentium 4 de préférence).

On demande un compte rendu de TP par binôme à rendre en TD la semaine du 10 au 14 Octobre.

On donnera les résultats obtenus sans chercher à fournir les explications des résultats (le but des cours et des TD/TP du module info313 est justement de permettre d'expliquer les résultats obtenus).

1. Exercice 1 : calcul en nombres flottants

Soit le programme C suivant :

```
main() {
int i ;
float S ;
S = 1000.0 ;
for (i=0; i<10000; i++)
    S=S+0.1;
printf(...)
}
```

Quel est la valeur finale de S ?

Même question en remplaçant les `float` par des `double`.

Avec les `float` (uniquement), on ajoutera l'option de compilation `-f32`

2. Exercice 2 : calcul en nombres flottants

Soit un programme C qui calcule les solutions d'une équation du second degré : $a.x^2+b.x+c=0$

Exécuter le programme pour les valeurs

$a = 0.3$

$b = 2.1$

$c = 3.675$

en utilisant d'abord des `float`, puis des `double`.

Quels sont les résultats obtenus ?

Avec les `float` (uniquement), on ajoutera l'option de compilation `-f32`.

3. Exercice 3 : calcul de la somme des N premiers entiers

On utilisera les deux procédures suivantes (récursive et itérative) permettant de calculer la somme des N premiers entiers :

```
int sumrec (int x){
    return (x <1 ? 0: x+sumrec(x-1));}

int sumite(int x){
    int i, tmp=0;
    for (i=1; i<=x; i++) tmp+=i;
    return tmp;}
```

Soit un programme C qui calcule la somme des N premiers entiers utilisant successivement chacune des procédures.

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N) pour chacune des versions pour N=10, N=50, N=100, N=500

Avec l'option de compilation -O0

Avec l'option de compilation -O2 et

- Avec l'option de compilation -xW
- Sans l'option de compilation -xW

4. Exercice 4 : copie de tableaux

Soit un programme C qui effectue la copie d'un tableau N x N dans un autre en utilisant successivement les deux algorithmes suivants :

```
Version ij
for (i=0;i<N;i++)
    for (j=0;j<N;j++)
        Y[i][j]=X[i][j];
```

```
Version ji
for (j=0;j<N;j++)
    for (i=0;i<N;i++)
        Y[i][j]=X[i][j];
```

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N*N) pour chacune des versions pour N=10, N=50, N=100, N=500.

Avec l'option de compilation -O2 et

- Avec l'option de compilation -xW
- Sans l'option de compilation -xW

5. Exercice 5 : Création d'un tableau contenant la suite des entiers de 0 à N

Soient les 3 fonctions suivantes permettant d'écrire la suite des entiers de 0 à N dans un tableau (T[i] = i).

```
void foo(int x){
    int i;
    Z[0]=0;
    for (i=1; i<=x; i++) Z[i]=Z[i-1]+1;}
```

```
void fou(int x){
    int i;
    Z[0]=0;
    for (i=1; i<=x; i++) Z[i]=Z[0]+i;}
```

```
void fov(int x){
    int i;
```

```
for (i=0; i<=x; i++) Z[i]=i;}
```

Donner le temps d'exécution/élément (c'est-à-dire le temps d'exécution total divisé par N) pour chacune des versions pour N=10, N=50, N=100, N=500

Avec l'option de compilation `-O2` et

- Avec l'option de compilation `-xW`
- Sans l'option de compilation `-xW`

6. Annexe : Mesures de temps

Les mesures de temps sous Linux sont données par les fonctions suivantes, qui donnent le nombre de cycles d'horloge.

Pour obtenir les temps d'exécution, on exécute les programmes un certain nombre de fois, et on fait la moyenne des temps obtenus en enlevant les valeurs « aberrantes » (très supérieures aux autres).

```
double dtime();
long long readTSC ();

long long readTSC ()
{
    long long t;
    asm volatile (".byte 0x0f,0x31" : "=A" (t));

    return t;
}
double dtime()
{
    return (double) readTSC();
}
```

Mesure du temps d'exécution

```
double t1,t2 ;//déclaration des variables

t1 = dtime();

//Partie du programme dont on mesure le temps d'exécution.

t2 = dtime();

dt = t2-t1; // Nombre de cycles d'horloge processeur
```

7. Annexe : utilisation de ICC

Documentation :

http://www.intel.com/software/products/compilers/clin/docs/main_cls/index.htm

Les programmes à utiliser sont disponibles dans la page Web du L313

<http://www.lri.fr/~de/ArchiL3-0506.htm>