Système de Numération

TP-1, Module I41, Licence Informatique
université du sud Toulon-Var

Fevrier 2007

1 Ligne de commande

Lorsque vous lancez un programme, en tapant `./nom_du_prog` sur la ligne de commande, vous pouvez en même temps rajouter derrière le nom de l'exécutable une ou plusieurs chaînes de caractères que l'on appelle *paramètres* du programme. Par exemple :

```
$ ./a.out toto 31
```

Ici, $ symbolise le prompt de l'utilisateur. La chaine `toto` est le premier paramètre, et `31` le second. Pour pouvoir récupérer la valeur de ces paramètres à l'intérieur de votre programme C vous devez déclarer la fonction `main` de la façon suivante:

```
int main(int argc, char *argv[])
```

L'entier `argc` contient le nombre de paramètres présents plus un. Le tableau `argv` est un tableau de chaînes de caractères, `argv[i]` contenant le $i^{\text{ème}}$ paramètre du programme, `argv[0]` est le nom du programme lui-même. A titre d'exemple, exécutez le programme suivant (en oubliant pas de rajouter des paramètres sur la ligne de commande):

```
#include <stdio.h>
int main(int argc, char *argv[])
{ int i;
  printf("Il y a %d paramtres:\n",argc);
  for (i = 0; i < argc; i++)
     printf("%s\n",argv[i]);
}
```

N'oubliez pas que le tableau `argv` récupère les paramètres sous forme de chaînes de caractères, si l'un d'entre eux est un entier que vous désirez manipuler en tant que tel dans votre programme vous devez le convertir avec la fonction `atoi` (ascii to integer). A titre d'exemple, exécutez le programme suivant qui affiche la valeur du premier paramètre en base hexadécimale :

```
#include <stdio.h>
int main(int argc, char *argv[])
{ int z;
  z = atoi(argv[1]);
  printf("%d --> %x \n",z, z);
}
```

Exercice 1 *Écrire un programme* `ajoute` *qui calcule puis affiche la somme d'une liste d'entiers passée par la ligne de commande.*

```
$ ./ajoute 15 3 12 5
35
```

2 Numération et Énumération

Exercice 2 *Soit* $(a_n \dots a_2 a_1)_b$ *la décompositon en base* b *d'un entier* z*, on appelle* b*-poids de* z *le nombre de composantes* a_i *non nulles. Par exemple le* 3*-poids de* 571 *est* 4*; En effet,* $571 = (210011)_3$*. Ecrire le programme* `poids` *acceptant deux paramètres, un entier* b *et un entier* z*, qui calcule le* b*-poids de* z*.*

```
$ ./poids 3 571
4
```

Exercice 3 *Utiliser le principe de décomposition pour écrire le programme* `enuplet` *acceptant deux paramètres, un entier* m *et un entier* q *pour énumerer tous les* m*-ulets* q*-aire. Modifier votre programme pour qu'il rapporte le nombre total* $N(q, m)$ *de réductions modulo* q *effectuées lors de l'énumération. Faire une représentation graphique avec* `gnuplot` *de la fonction* $\log N(5, m)$*. Proposer une formule!*

Exercice 4 *Utiliser les opérateurs de décalage (*`<< >>`*), et l'opérateur logique* `&` *pour ecrire le programme* `binaire` *qui décompose un entier* z *en base* 2*. Le programme doit afficher les symboles de la décomposition en base en commançant par le chiffre significatif.*

```
$ ./binaire 155
1 0 0 1 1 0 1 1
```

Indication: Vous allez d'abord devoir trouver la plus grande puissance de 2 *inférieure ou égale à* z.