

Architecture des ordinateurs et systèmes d'exploitation

TP 4: Manipulation de bits

Marcel Bosc

Christophe Dehlinger
Benoît Meister

Arnaud Giersch
Nicolas Passat

Mathieu Haeefele

Le but de ce TP est de manipuler des données bit à bit. En C, les opérateurs bit à bit sont les suivants :

Fonction	Opérateur
et	&
ou	
ou exclusif	^

Fonction	Opérateur
complément à 1	~
décalage (gauche et droit)	<< et >>

1. Cryptage simple

On désire crypter un texte. On fait pour cela un ou exclusif entre un code choisi à l'avance et chacun des caractères du texte. Écrire un programme C cryptant ainsi un texte lu sur l'entrée standard, et affichant le résultat sur la sortie standard.

Que se passe-t-il si on crypte deux fois de suite un texte avec le même code ?

2. Compter le nombre de bits

- Écrire des fonctions en C permettant, à partir d'un mot de 32 bits¹ (i) d'en extraire le nième bit, (ii) de mettre le nième bit à 1, et (iii) de mettre le nième bit à 0.
- Écrire une fonction (simple) comptant le nombre de bits à 1 dans un mot de 32 bits.
- Voici un autre algorithme permettant de trouver le même résultat :
 - Le mot de 32 bits peut être vu comme 32 champs de 1 bit, chacun contenant le nombre de bits à 1 dans le champ.
 - On additionne les champs de 1 bit par paire, et on stocke le résultat dans les deux bits utilisés. Chaque champ de 2 bits contient le nombre de bits à 1 dans ces 2 bits du mot de départ.
 - On additionne ensuite de la même manière les champs de 2 bits pour avoir un résultat sur 4 bits.
 - On continue ainsi jusqu'à ce que le résultat « occupe » les 32 bits. Le nombre ainsi formé est le nombre de bits à 1 dans le mot d'origine.

Exemple, avec le mot 01111000011111010101100100010110 :

0|1|1|1|1|0|0|0|0|1|1|1|1|1|0|1|0|1|0|1|1|0|0|1|0|0|0|1|0|1|1|0

on additionne les champs de 1 bit,

0 1|1 0|0 1|0 0|0 1|1 0|1 0|0 1|0 1|0 1|0 1|0 0|0 1|0 1|0 1

on additionne les champs de 2 bits,

00 1|1|000 1|001 1|001 1|001 0|001 0|000 1|001 0

¹On utilisera le type C `unsigned int`.

on additionne les champs de 4 bits,

```
00000100|00000110|00000100|00000011
```

on additionne les champs de 8 bits,

```
0000000000001010|0000000000000111
```

on additionne les champs de 16 bits,

```
0000000000000000000000000000010001
```

Le résultat, 17, est bien le nombre de bits à 1 dans le mot de départ.

Suggestion : utiliser des décalage et des masques pour aligner les champs à additionner.

3. Bit de parité

Un bit de parité est un bit ajouté à un mot, de telle manière que le nombre total de bits à 1 soit pair. Cela sert par exemple à vérifier qu'un mot n'a pas été altéré lors d'une communication.

a) Écrire une fonction pour positionner le 31^e bit d'un mot de manière à ce que la parité soit respectée.

b) Écrire une fonction qui vérifie la parité d'un mot.

En faire deux versions, une pour chacun des algorithmes de comptage de l'exercice précédent. Comparer (avec la commande `time (1)`) le temps d'exécution pour tester 10^8 mots.

4. Compression

On désire compresser du texte ne contenant que des lettres (A-Z) et des espaces.

a) Combien de bits faut-il au minimum pour pouvoir coder tous les caractères ? Combien de caractères peut-on ainsi coder dans 32 bits ?

b) Écrire un programme C lisant un texte sur l'entrée standard, le compressant, et écrivant le résultat sur la sortie standard.

Écrire un programme permettant de faire l'opération inverse (décompresser).

c) À quoi faut-il faire attention si on transfère un fichier ainsi compressé entre deux machines (par exemple entre `ada` et votre PC) ?