NOM
Prénom

17 janvier 2008

Bases des systèmes informatiques
Interrogation 2

Exercice 1 : code détecteur d'erreur

0	1	2	3	4	5	6	7	8

On se donne un code acceptant des mots binaires de 9 bits. On appellera les bits par leur position, où le premier bit est appelé 0 et le dernier 8 (cf. ci-dessus). On suppose que chaque bit dont la position est une puissance de 2 est un bit de parité de tous les bits à sa gauche (i.e. de position inférieure), bits de parité inclus.

1. Combien ce code a-t-il de bits de contrôle ? Lesquels ?

   4 : bits # 1 ($=2^0$), 2 ($=2^1$), 4 ($=2^2$), 8 ($=2^3$)

2. Combien de mots valides accepte-t-il ?

   Puisque les 4 bits de contrôle sont fixés par les autres bits, le code accepte 2^5 mots.

3. Quelle est la distance de Hamming de ce code ?

   En inversant un bit unique sur les bits de données (e.g. 000000000 et 110000000) on obtient une distance de 2. Toute inversion d'un bit de données changeant au moins un bit de contrôle, c'est la distance minimale.

4. Y a-t-il des bits à valeur constante, pour les mots valides ? Si oui, lesquels et pour quelle valeur ?

   Le bit #2 a pour valeur 0, dans tout mot valide. En effet, si le mot est valide, la parité des deux premiers bits est toujours paire.

5. Pour chacun des mots proposés, mettez une croix dans la colonne de droite s'il est valide, encerclez la ou les erreurs, sinon (on supposera, sauf impossibilité, que les bits de contrôles sont corrects) ?

1 1 0 0 0 0 0 0 0	×
1 1 1 0 0 0 0 0 0	
1 1 1 0 1 0 0 0 0	
1 1 0 1 1 0 0 0 0	×
1 1 0 0 0 1 0 0 1	×
1 1 0 0 0 1 1 1 1	×
1 1 0 1 1 1 1 1 1	×
1 1 0 1 1 0 1 1 1	

T.S.V.P

Exercice 2 : compter en assembleur

Soit le programme suivant :

```
      MOV   AX,[100]   ;  lire le contenu de l'adresse 100 dans le registre AX
      MOV   BX,0       ;  initialiser BX à 0
L1 :  ADD   BX,1       ;  ligne avec étiquette 'L1', incrémenter BX de 1
      CMP   BX,AX      ;  comparer BX à AX
      JLE   L1         ;  si BX ⩽ AX, retourner à la ligne L1 sinon continuer
      MOV   [100],BX   ;  écrire le contenu de BX à l'adresse mémoire 100
      INT   3          ;  stop
```

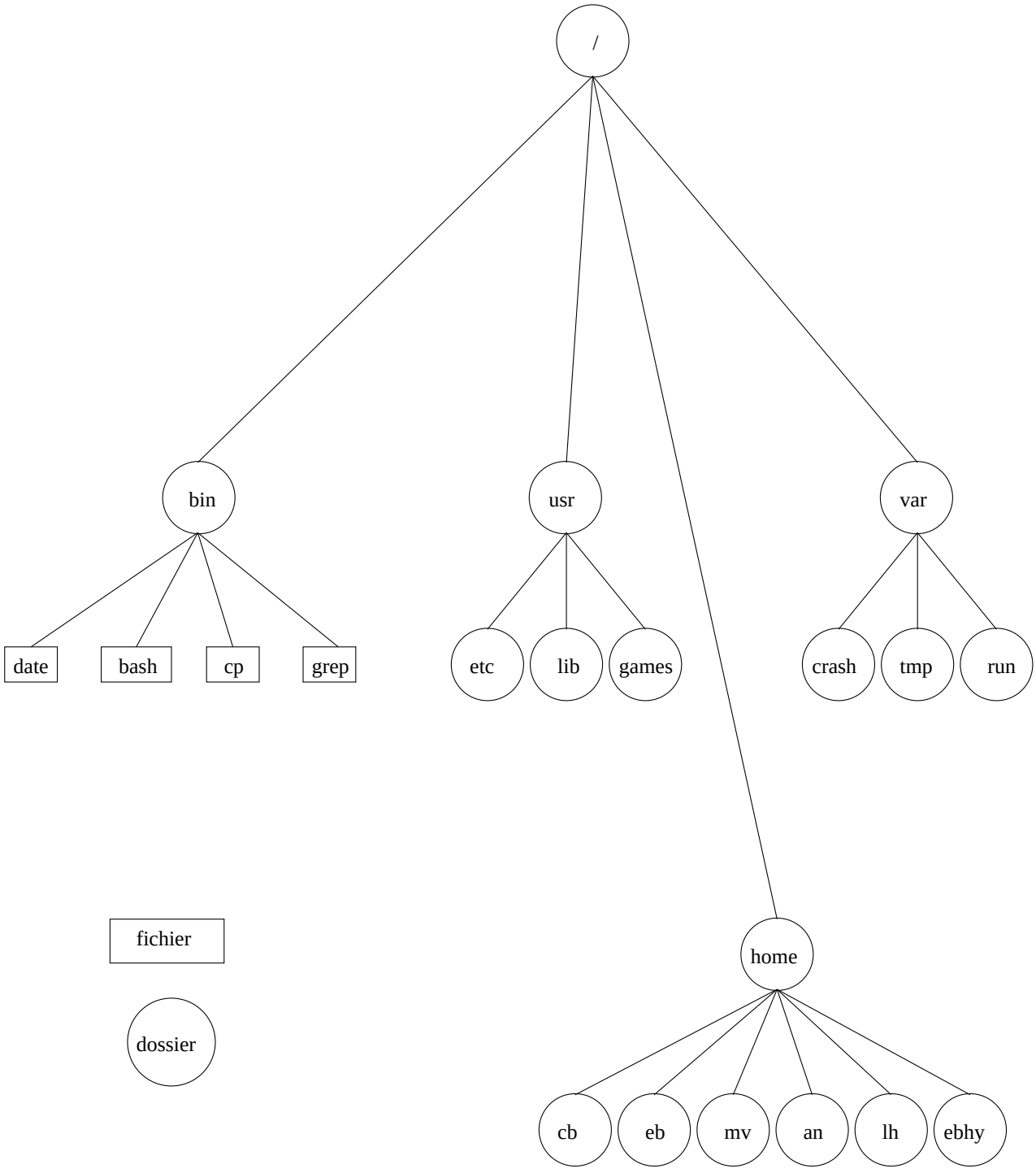
1. Que fait ce programme ?

Tant que la variable temporaire est inférieure ou égale à la valeur à l'adresse mémoire 100, il l'incrémente de 1. Ensuite il range le résultat au même endroit.
Il fait donc [100]+=1.

2. Écrire un programme – s'inspirant de l'exemple ou non – qui additionne les nombres impairs inférieurs à celui contenu à l'adresse 200 et écrit le résultat à l'adresse 100.

```
      MOV   AX,[200]   ;  lire le contenu de l'adresse 200 dans le registre AX
      MOV   BX,0       ;  initialiser BX à 0
      MOV   CX,1       ;  initialiser CX à 1
L1 :  ADD   BX,CX      ;  incrémenter BX de la valeur courante de CX
      ADD   CX,2       ;  augmenter CX de 2 pour parcourir les nombre impairs
      CMP   CX,AX      ;  comparer CX à AX
      JLE   L1         ;  si CX ⩽ AX, boucler sinon continuer
      MOV   [100],BX   ;  écrire le contenu de BX à l'adresse mémoire 100
      INT   3          ;  fin
```

Exercice 3 : Fichiers sous UNIX



On suppose que le dossier courant est la racine et que chacun reconnaîtra son répertoire personnel, à ses initiales (demander, sinon).

T.S.V.P

1. Sans changer de dossier, donner la commande permettant de copier le fichier ‘bash’ vers votre répertoire personnel.

```
cp bin/bash ~/bash
```

2. Donner la commande permettant d’afficher le contenu du répertoire bin.

```
ls bin
```

3. Donner la commande permettant d’aller dans votre répertoire personnel

```
cd /home/cb                          chemin absolu, ou
cd ./home/cb ⇔ cd home/cb            chemin relatif, ou
cd ~                                 racourci
```

4. Donner la commande permettant créer un répertoire nommé ‘shell’

```
mkdir shell
```

5. Donner la commande permettant de créer, dans le répertoire ‘shell’, un fichier texte – qu’on appellera ‘liste.txt’ – contenant le résultat de la commande ‘ls’ appliquée aux répertoires ‘usr’ et ‘var’

```
ls /usr > shell/liste.txt; ls ../../var >> ./shell/liste.txt
```