Dépt info – 1ere année Cours d'introduction à l'informatique
4 décembre 2002

Contrôle long Corrigé

Durée : 3h. Aucun document ni calculatrice autorisé.
Répondez aux questions posées et pas à autre chose (pour éviter de perdre du temps), rédigez clairement et lisiblement. Et bon courage !

Questions de cours (4 pts)

1. Où sont physiquement les barrettes mémoires dans l'ordinateur ? Quelle manipulation fait-on pour rajouter de la mémoire ? Quel est le mot français pour "ram" ?
Sur la carte mère – on ajoute ou remplace une ou des barrettes enfichée dans les prises – ram = mémoire vive

2. Qu'est-ce que sauver les données sur lesquelles on travaille (que se passe-t-il physiquement dans la machine) ? Qu'est-ce que charger des données ?
Sauver les données = les copier de la mémoire vive sur un disque (plus généralement, sur une *mémoire de masse*, c'est à dire une mémoire capable de stocker de grandes quantités d'information et qui ne s'efface pas quand on coupe le courant – autre ex une bande) ;
Charger les données = les copier du disque dans la mémoire vive

3. Donnez 8 commandes Unix parmi les 10 :
   a. afficher le nom du répertoire courant : **pwd**
   b. changer de répertoire courant pour aller dans _nouv_rep_ **cd nouv_rep**
   c. afficher le contenu du répertoire courant, y compris les fichiers cachés **ls –a**
   d. créer un nouveau répertoire **mkdir nouv_rep**
   e. détruire un fichier **rm fich_**
   f. déplacer un fichier **mv fich_**
   g. afficher l'aide sur une commande Unix ou une fonction C **man *commande***
   h. voir le contenu d'un fichier **cat fich_** ou **more fich_**
   i. voir ses premières lignes **head fich_**
   j. voir ses dernières lignes **tail fich_**

Exercice 1 (4 pts)

Quand on exécute le programme suivant :

```
main() {
    long int x=3, y=7, z=4, litab[ 3] ;
    printf("x est à %u ; y est à %u ; z est à %u ; litab est à %u\n",
                                &x, &y, &z, litab);
    /* Endroit pour ajouter du code*/
}
```

il affiche :

```
x est à 1567892 ; y est à 1567884 ; z est à 1567876 ; litab est à 1567852
```

a. Quelle est sur cette machine la taille d'un entier long (expliquer votre réponse).

8 octets (distance entre 2 variables successives)

b. On ajoute à l'endroit indiqué le code :

```
litab[ 0]  = litab[ 4]  ;
```

Indiquez si le compilateur C indiquera une erreur en lisant cette instruction. Expliquez (si votre réponse est oui, expliquez quelle règle a permis de la détecter ; si votre réponse est non, expliquez quelle difficulté empêche d'ajouter dans le compilateur une règle de détection de cette erreur).

non – dans ce cas, il n'y a aucune difficulté pour détecter l'erreur à la compilation, mais ce n'est pas le cas général. Le C admet que les indices soient calculés (ex : tab[i+3], tab[f(i)]). Il faudrait donc pouvoir dire à la compilation (en lisant le programme sans l'avoir exécuté) si i+3 ou f(i) par exemple vont dépasser les limites du tableau. Dans cette écriture, f(i) peut très bien être une fonction de bibliothèque qui a été programmée par quelqu'un d'autre. Donc on ne peut pas en général savoir s'il y aura un dépassement des limites avant d'essayer (avant l'exécution) ; une vérification qui ne marche que dans un petit pourcentage de cas n'a pas paru très utile aux concepteurs du C.

c. On ajoute encore

```
*(litab + 3) = 10 ;
printf("x vaut %d, y vaut %d, z vaut %d, litab contient dans l'ordre {%d, %d, %d}\n",
                                    x, y, z, litab[ 0] , litab[ 1] ,litab[ 2] ) ;
```

Calculez l'adresse à laquelle l'instruction de la première ligne écrit.

Donnez l'affichage fait par le printf.

***(litab+3) est à l'adresse de z (`1567852 + 3 x 8`)**

L'affichage est (les '?' désignent des valeurs aléatoires) :

x vaut 3, y vaut 7, z vaut 10, listab contient dans l'ordre (7, ?, ?) [puisque listab n'est pas initialisé]

Exercice 2 (5 pts)

Dans l'institut informatique de Villejoyeuse, les 300 étudiants disposent de trois salles machines sous linux (les salles "terre", "lune" et "soleil"). Les noms des machines sont formés avec leur nom de salle et une lettre.

Toutes les machines de la même salle sont organisées de la même façon (elles montent toutes les mêmes disques distants). Chaque étudiant a une boite aux lettres (en abrégé une bal) qui est dans /var/mail. M. Lambda se logue successivement dans deux salles sur une machine. Quand il vient de se loguer il tape (ces deux commandes donnent le même résultat dans toutes les salles)

```
terre_a>pwd
/home/users/etud/1A/groupe6/lambda.
terre_a> whereis emacs
/usr/local/bin/emacs
```

Il essaye ensuite la commande uname –sr qui donne le nom du système d'exploitation. Il obtient selon la machine :

```
terre_a>uname -sr          | lune_c>uname -sr
Linux Mandrake 7.3         | Linux RedHat 8.1
```

Quand il lance la commande `df`, il obtient :

```
terre_a>df -k
Filesystem            1K-blocks     Used          Avail         Capacity      Mounted on
/dev/s3d2t4           97663043      429192        6337112       35%            /
progMd:/exec/SE       23489556      18643004      4846552       79%           /usr
data:/users/etud      61427954      42136897      19291057      69%       /home/users/etud
mail:/users/etud/bal  274247        204333        42490         83%           /var/mail

lune_c> df -k
Filesystem            1K-blocks     Used          Avail         Capacity      Mounted on
/dev/ibm70g6s4        9766152       1506108       8260044       15%           /
progRH:/exec/SE       25609350      14650468      10891382      76%           /usr
data:/users/etud      61427954      42136897      19291057      69%      /home/users/etud
mail:/users/etud/bal  274247        204333        42490         83%           /var/mail
```

(used = utilisé ; Available = libre ; Capacity = taux d'utilisation)

1. Indiquez pour chacune des deux machines le nombre de disques auxquels elle accède et quel répertoire est monté sur chaque disque. Précisez lesquels sont des disques locaux et lesquels sont des disques distants.

Chaque machine a un disque local et 3 disques distants. Les disques distants sont ceux dont le file system commence par un nom de machine. Par ex pour terre_a, /usr est monté sur un disque de la machine progMd, disque dont le chemin d'accès sur progMd est /exec/SE (et dont le chemin d'accès sur terre_a est /usr !)

2. Sur quelle machine est en fait le répertoire de login de Mr Lambda ? Donnez le chemin d'accès complet sur cette machine du répertoire en question. D'après vous, les utilisateurs retrouvent-ils leurs données quand ils changent de salle ?

Il est sur la machine data, dans le répertoire /users/etud/1A/groupe6/lambda. Comme le répertoire de login est dans toutes les salles un sous répertoire de /home/users/etud, que ce dernier est partout monté au même endroit, et qu'aucun de ses sous-répertoire n'est monté, le répertoire de login est physiquement le même dans toutes les salles.

3. Mêmes questions pour le fichier de courrier électronique qui sur terre_a ou lune_c s'appelle /var/mail/lambda

Il est sur la machine mail, et s'appelle /users/etud/bal/lambda. Il est aussi accessible de partout

4. De combien d'espace disque chaque étudiant dispose-t-il en moyenne pour stocker ses données ? De combien dispose-t-il pour sa boite aux lettres ? Expliquez ce qui se passe sur le disque quand on envoie à chaque étudiant un message avec une circulaire de 50 Ko attachée.

Données : 60 Go pr 300 etud → 200Mo par etud. Mail : 274 Mo en tout, soit moins d'un méga par étud. La circulaire occupe à elle seule 300 fois 50 Ko = 15 Mo sur le disque mail (plus de 5% de l'espace disque)

5. Sur quelle machine est en fait le logiciel emacs utilisé par terre_a ? par lune_c ? Sachant que tous les utilitaires système d'une machine donnée sont dans /usr, expliquez comment sont organisés les utilitaires systèmes.

Emacs est pour terre_a sur progMD, pour lune_c sur progRH. Chaque salle a un serveur de logiciel différent, adapté au système utilisé dans cette salle, et qui sert les utilitaires. (c'est une structure assez classique, qui augmente un peu le trafic réseau et les temps de réponse, mais facilite la maintenance).

Exercice 3 (7 pts)

Dans les débuts de l'informatique (il y a au moins 30 ans), certaines des grosses machines les plus employées avaient des mots machine de 9 bits. Au lieu de mesurer et d'utiliser la mémoire par octets comme nous le faisons encore maintenant, sur ces machines on la mesurait et on l'utilisait par nonnets, c'est à dire par paquets de 9 bits. Deux exemples de nonnets sont (110010101_b) et (000111001_b). Le code ascii n'était encore qu'un code de caractères parmi d'autres, et les codages entiers n'étaient pas normalisés.

1. Combien de nonnets différents y a-t-il ? Cela suffit-il pour coder les caractères ?
   512 – c'est largement assez puisqu'en ascii étendu on a 256 caractères
2. Supposons un code caractère sur un nonnet. Quand on met en mémoire ou sur disque un texte de 1000 caractères, combien occupe-t-il de place (en bits). Comparez avec la place occupée par le même texte codé en ascii. A votre avis, quel problème se posait à ces anciens informaticiens quand ils construisaient un code de caractère de taille un nonnet, sachant que le matériel (mémoire, disques,…) coûtait très cher ?
   9000 bits au lieu de 8000 en ascii étendu, 7000 en ascii pur. Ce codage consomme de 12,5% à 14% de ressources en plus, ce qui coûtait cher
3. On s'intéresse maintenant aux entiers. On veut d'abord choisir leur taille. **Tous les calculs seront faits en kilo, mega, etc.** Indiquez le nombre d'entiers qu'on peut représenter si l'on utilise pour cela 2, 3 ou 4 nonnets. Faites un tableau indiquant dans chacun des trois cas le plus grand entier non signé, le plus grand et le plus petit entier signé.
   $2^{18} = 2^8 \times 2^{10} = 256 \times 1k$, etc

	2	**3**	**4**
nmbre d'entiers	**256 K**	**128 M**	**64 G**
+ gd non signé	**256 K -1**	**128 M -1**	**64 G -1**
+ gd signé	**128 K –1**	**64M –1**	**32G –1**
+ ptt signe	**-128K**	**-64M**	**-32G**

4. Comme pour les octets, l'écriture binaire des nonnets est mal commode. Expliquez pourquoi la décomposition de cette écriture en groupes de 4 bits à l'aide de la représentation hexadécimale ne convient pas non plus (aide : si vous ne voyez pas, prenez un ou deux exemples au hasard d'entiers sur 2 nonnets et convertissez-les en hexadécimal).
   Expliquez pourquoi la décomposition en groupes de 3 bits convient mieux. Quel est la base de la représentation associée ? quels sont son plus petit et son plus grand chiffre ?
   Vous venez d'inventer la représentation octale. Donnez la représentation octale de l'entier non signé sur deux nonnets ($110010101\ 000111001_b$). Donnez la méthode de traduction de

la représentation binaire en représentation octale et la méthode de traduction de la représentation octale en représentation binaire.

Parce que 9 n'est pas un multiple de 4, et donc si l'on traduit 2 nonnets en hexa, on ne retrouve pas la juxtaposition des traductions de chacun dans le code. En octal, pas de pb. La base est huit, le plus grand chiffre est 7, le plus petit 0. Ex : 625071.
Traduction : de binaire en octal, on fait des paquets de 3 bits (en partant de la droite si la longueur est quelconque) et on les remplace par le chiffre octal correspondant ; d'octal en binaire, on traduit les chiffres un par un.

5. Le langage C a été inventé à cette époque[1], et les caractères peuvent encore y être représentés en octal : \ooo représente le caractère dont le code ascii est, en octal, ooo (3 chiffres obligatoires. Les guillemets gardent le même rôle: on écrit char c = '\ooo' comme on écrivait déjà char c = 'a' ou char c = '\n').
   On se limite au code ascii pur (non étendu). Donnez la plus grande représentation utilisée.
   Donnez la représentation de A (= 41h), celle du a (=61h) et celle du 0 (=30h).
   En déduire la représentation du Z et celle du 9.

La plus grande représentation utilisée en ascii pur correspond au code 127_d soit en C \177. On a aussi 'A' =='\101 'a' == '\141 '0'=='\060' 'Z'=='\132' '9'=='\071'

[1] Sur des machines 6 bits plutôt que 9 bits. Mais les questions 1 et 2 auraient été plus compliquées…