Programmation Avancée en Langage C

Examen Module I22, Licence Informatique, USTV.

Mai 2010

Le sujet est composé 10 questions à traiter en moins de 90 minutes. Aucun document n'est autorisé. Vous êtes invités à remettre une copie claire, concise, sans rature ni surcharge. Vous répondrez directement sur ces feuilles. La note finale tiendra compte de la présentation générale de la copie et de la rapidité d'exécution.

1. Commande. Donner 7 commandes particulièrement utiles aux programmeurs en langage C sous **linux** :

commande	description

2. Option de compilation. Au cours des séances de travaux pratiques, nous avons utilisé un compilateur. Préciser lequel en donnant 5 des options fréquemment utilisées.

compilateur	
option	description

3. Flux standard. Préciser les 3 flux standards associés à un processus.

flux	description

- Quelle commande permet de rediriger la sortie standard de **prog.exe** vers un fichier **fic.txt** ?

- Quelle commande permet compter le nombre de lignes de **prog.c** contenant le motif **float** ?

4. Erreur d'exécution.

```
1 int main( void )
2 {
3 int x = 0;
4 int *y = NULL;
5
6 return 0;
7 }
```

Quelle instruction suffirait-il d'insérer en ligne 5 pour obtenir un programme se terminant mal :

instruction	terminaison
	Erreur de segmentation
	Exception en point flottant
	Débordement de pile

5. Débogage. L'exécution **./prog.exe 123** se termine par une erreur de segmentation. Comment procéder avec **gdb** pour localiser la ligne d'erreur.

6. Lien avec le système. Dans un programme en langage **C**, une fonction particulière doit faire le lien entre le système et le processus.

1. Laquelle ?

2. Préciser le prototype adéquate

3. Préciser la nature des paramètres transmis par le système.

4. Préciser le rôle de la valeur de retour de cette fonction.

5. Quelle fonction de la librairie **unistd** permet de gérer les options placées sur la ligne de commande.

6. Makefile.

```
CF= -Wall -g
all : prog.exe
prog.exe : prog.c
	gcc $(CF) prog.c -o prog.exe
demo : prog.exe
	./prog.exe 123
```

On utilise le fichier **makefile** pour gérer la compilation du programme de **prog.c**.

1. Quelle commande faut-il lancer pour obtenir la compilation du programme **prog.c**.

2. Quelle commande faut-il lancer pour exécuter la démonstration associée à la cible **demo** ?

7. Chaine de caractères. On rappelle que le code ASCII de la lettre A est 65, et celui du caractère de nouvelle ligne est 10.

1. Comment est représentée la chaine "$HELLO\backslash n$" en langage **C**?

2. Ecrire une fonction **int strlen(char * str)** qui calcule et renvoie la longueur de la chaine **str**.

8. Mémoire.

```
char *x;
int main ( void )
{
char *y = NULL;
  x = (char*) malloc(1);
return 0;
}
```

La mémoire associée à un programme est divisée en 4 zones. Préciser le nom et le rôle de ces

zones. Dans quelle zone sont situés les adresses référencées par : **x**, **&x**, ***x**, **y**, **&y**, ***y**, **&main**, avant le retour de **main**.

zone	description	référence
hors zone	ne référence pas une adresse, adresse invalide	

```
int r;
void proc(int x,int y,int *z)
{ int t;
        t = x + y;
       *z = t;
}
int main ( void )
{
   proc(1, 2, &r);
}
```

Représenter le cadre de pile de la procédure **proc** en cours d'exécution sachant que l'adresse de r est $0x9000ABCD$.

adresse	valeur	commentaire
0xBF00 0000	sauvegarde	BP appelant
0xBF00 0004	main+x	adresse de retour
0xBF00 0008	1	paramètre x

10. Flux. L'objectif du programme (1) est de convertir un fichier texte en majuscule. Les deux noms de fichiers sont passés par la ligne de commande. Comment compléter les lignes marquées (//) pour obtenir un programme correct.

ligne	instruction
	déclaration
7	
	contrôle des arguments
8	
	ouvertures des fichiers
12	
13	
	boucle
23	
	fin
26	
27	

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <ctype.h>
 4
 5 int main (int argc, char *argv[] )
 6 { int car;
 7   //
 8   if (    ){ //
 9     fprintf(stderr, "\nerreur : arg");
10     exit( 1 );
11   }
12   src = //
13   dst = //
14   if ( src == NULL ) {
15     fprintf(stderr, "\nerreur : nom");
16     exit( 1 );
17   }
18   while ( ! feof( src) ) {
19     car = fgetc( src );
20     if ( car != EOF ) {
21       if ( isalpha( car ) )
22         car = toupper( car );
23     //
24     }
25   }
26   fclose( src );//
27                 //
28   return 0;
29 }
```

Figure 1: Source **flux.c**