Examen de système d'exploitation
Semestre décalé
Département d'informatique IUT Villetaneuse
lundi 23 janvier 2006, 14h-17h

Remarques : tous les documents de cours, td/tp sont autorisés. Le barème est indicatif.

1- Commandes unix – 6 pts

	Commandes	*Questions / Affirmations*	*Réponse / Justification*	*Points attribués*
1.1	`$ gcc -O3 essai.c`	Le code exécutable généré n'est pas optimisé	Vrai ou Faux	1
1.2	`$ file a.out` `a.out: Mach-O executable ppc`	Le processeur sur lequel on travaille a une architecture x86-32 bits.	Vrai ou Faux	1
1.3	`$ gcc -S essai.c`	On transforme le fichier objet en exécutable « Static »	Vrai ou Faux	1
1.4		Le compilateur gcc, l'interface graphique X11 ne peuvent pas tourner dans un environnement Windows.	Vrai ou Faux	1
1.5		Quelle est la commande que l'on applique sur un exécutable pour lister les fonctions que cet exécutable contient ?		1
1.6		L'ordonnanceur du système Linux ne passe jamais la main au processus de numéro 13	Vrai ou Faux	1/2
1.7	`$ ps aux | grep 13`	Affichage des informations sur le processus numéro 13 uniquement	Vrai ou Faux	1/2

2- Langage de commande bash - 4 pts

Dans le devoir à la maison et dans l'exercice vu en contrôle court, vous avez programmé un script bash qui prend en paramètre une fonction trigonométrique, une borne inférieure, une borne supérieure et un pas d'incrément et vous avez affiché dans une fenêtre la représentation graphique de la fonction au moyen de l'utilitaire `tput`.
On veut mieux contrôler l'affichage en ramenant tous les points de la courbe dans un cadre délimité par les coordonnées (0,0) et (24,79) – car on fait un affichage dans une fenêtre texte de 25 lignes et 80 colonnes.

1. Modifiez votre script bash précédent pour trouver les valeurs minimales et maximales de votre fonction y=f(x)
2. Calculer deux facteurs de contraction/dilatation (un pour x et un pour y) qui seront appliqués sur les points (x, y) de la courbe afin que l'affichage se fasse dans le cadre (0,0) et (24,79).

3- Fork - 3pts

Ajoutez à l'endroit indiqué dans le code ci dessous des instructions de gestion de processus nécessaires pour provoquer l'affichage de « Bonjour » exactement 4^N fois. Vous ne pouvez pas toucher au code ailleurs qu'à l'endroit indiqué (sauf si vous voulez ajouter des déclarations de variables), vous ne pouvez plus ajouter d'instruction `printf`. Vous devez réaliser la tâche avec des `fork()`, peut être avec des `wait()`. Donnez toutes les justifications nécessaires.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define N 3

int
main(int argc, char *argv[])
{
      int i, pid1, pid2;

      for (i = 0; i < N; i++) {
            /* debut des ajouts */

            /* fin des ajouts   */
      }

      printf("Bonjour\n");
      return 0;
}
```

Note : il existe une solution à ce problème qui s'exprime en 14 caractères.

4- Mémoire partagée, Tubes et Fork – 7 pts

Le but de cet exercice est d'afficher les nombres premiers compris entre 2 et 100. Nous verrons deux méthodes différentes.

4.1- La première méthode utilise la mémoire partagée et la commande `fork()`. On commence par placer dans un tableau partagé (en utilisant `mmap()` par exemple) tous les nombres impairs (le nombre premier 2 est traité à part) :

père	fils	père	fils	père	fils	...	père
3	5	7	9	11	13	...	99

Ensuite on traite deux cases consécutives du tableau : T[i] est traité par le père et T[i+1] par un fils créé pour la circonstance et pour i convenablement choisi. Pour chaque case, les processus vérifient que le contenu de la case n'est divisible par aucun nombre non nul contenu dans les cases précédentes.

Si nous trouvons un diviseur dans les cases précédentes, la valeur de T[i] (respectivement T[i+1]) deviendra 0. Le père attend la terminaison de son fils pour passer au prochain couple de cases.

Implémentez cette méthode en langage C avec de la mémoire partagée.

4.2- La deuxième méthode utilise les tubes et les processus créés par l'instruction `fork()`. Ici un père et un fils s'échangent une liste de taille décroissante d'entiers potentiellement premiers. Au départ le père considère les nombres impairs entre 3 et 99. Il affiche 3 puis passe à son fils (via un tube) les entiers impairs qui ne sont pas divisibles par 3. Cette liste commence par 5. Le fils affiche 5 puis renvoie à son père les entiers impairs qui ne sont pas divisibles par 5. Le père reçoit en premier l'entier *7* qu'il affiche puis il renvoie à son fils les entiers impairs qui ne sont pas divisibles par *7* etc

1. Par quel moyen technique allez-vous pouvoir contrôler l'arrêt de l'algorithme donné ci dessus ? Décrivez-le.
2. Ecrire le code C qui effectue le premier aller-retour entre le père et le fils et qui utilise en particulier des tubes.
3. Généralisez votre code précédent pour traiter le problème dans son ensemble.