ROB4 - INFORMATIQUE SYSTEME

11 Décembre 2014
Devoir sur table : 70 min
Ordinateurs, calculatrices, téléphones et documents interdits.
Répondre directement sur le sujet.

Ex.	max points	questions	note
1	3	questions	
2	4	applications	
3	4	programmation	
4	3	programmation	
5	6	programmation	
		total	

Ex 1 : Questions de cours

- Écrire votre nom et prénom dans un commentaire en C.

- Soit : `int main(int argc, char** argv)`. Que vaut `argv[0]` ?

- Donner un exemple de ce qui est stocké dans la pile (stack) et dans le tas (heap)
  - Pile :
  - Tas :

– Allouer et libérer dynamiquement un tableau de 8 entiers.

– Soit `int* (*ptr)[];`. Décrire par une phrase ce qu'est `ptr` :

– Considérons le modèle à quatre états d'un processus. Décrire ces états :
  – Running (Élu) :
  – Ready (Prêt) :
  – Waiting (Bloqué) :
  – Zombie (Zombie) :

– Que fait la famille de fonction `exec*` (détailler la réponse) ?

– Quelles sont les différences entre un thread et un processus ?

– Définir un type `pf_t` représentant un pointeur vers une fonction prenant en argument deux entiers et qui retourne un double, puis déclarer un tel pointeur.

– Dans le cadres des entrées/sorties, quelles sont les différences entre un accès direct et un accès séquentiel ?

– Donner des exemples de ce que peut être un stream (ou flux) ?

– En Bash, comment rediriger la sortie `stdout` d'un programme `mon_prog` vers un fichier `sortie.txt` ?

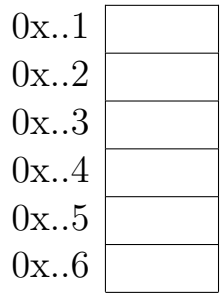
– Quel est le résultat de `34 >> 3` ?

Ex 2 : Application directe

Question 1

Après la portion de code ci-dessous, spécifiez le contenu de la mémoire (1e colonne : laissez blanches les cases indéterminées) ainsi que ce que donnerait un `printf` de certaines expressions (2e colonne) : valeurs, adresses ou erreurs.

```
1  int b = 5;
2  int v[] = { 0, 2, 3};
```



`printf("...", ...);`

`b`		`&b`	
`*b`		`&(b+1)`	
`*v + 1`		`&b+1`	
`*(v + 1)`		`++b`	
`&(&v)`		`**b`	

Question 2

Implémenter la fonction `void swap(int* a, int* b)` qui permute deux entiers.

Question 3

Déclarer deux entiers (2 et 3 par exemple) et donner l'instruction qui permet de les permuter.

Question 4

Dessiner de manière claire l'arbre des processus associé au code suivant :

```c
#include <stdlib.h>
#include <sys/types.h>

int main(int argc, char** argv){
    pid_t pid;
    int i;
    fork();
    pid = fork();
    if(pid == 0){
        fork();
    }
    else{
        for(i=0; i<2; i++){
            pid = fork();
            if(pid == 0){
                break;
            }
        }
    }

    return 0;
}
```

–question 4–

Ex 3 : Programmation 1

Question 1

Écrire un programme qui lit sur l'entrée standard, et affiche sur la sortie standard ce qui est lu. Le programme a un argument optionnel `--lc` . S'il est présent, le programme affichera ce qui est lu précédé du numéro de la ligne. Indice : La fonction `fgets` s'arrête de lire lorsqu'un saut de ligne ou une fin de fichier est lu. Il ne faut pas oublier les includes.

Question 2

Écrire le Makefile permettant de générer l'exécutable `lignes` (et le nettoyage) en supposant que le code de la question précédente est dans le fichier `lignes.c`.

Ex 4 : Programmation 2

Écrire un programme générant deux processus fils. Le premier processus mute en `ls`, le second en `who`. Il faut attendre la fin des deux fils avant de retourner.

Ex 5 : Programmation 3

Question 1

Il faut veiller à respecter les points suivants :
- Programme Thread-Safe
- Pas de mémoire perdue
- Gestion des erreurs
- Code compilable
- Pas de warning (variables inutiles, ou non inialisées avant évaluation. . .)

Soit l'entête `prod.h` suivant :

```
1  #ifndef PROD_H
2  #define PROD_H
3
4  #include <pthread.h>
5  #include <sys/types.h>
6
7  typedef struct{
8      pthread_mutex_t* jeton;
9      int val;
10     double sleep_p_plus;
11     double sleep_p_moins;
12 } param_t;
13
14 void* p_plus(void* arg);
15 void* p_moins(void* arg);
16
17 #endif /* PROD_H */
```

Implémenter les fonction `p_plus` et `p_moins` (dans un supposé `prod.c`).
- `p_plus` prend en argument un `param_t`, ajoute 2 à `val`, attend `sleep_p_plus` puis affiche la valeur courante de `val` tant que `val` (ou une copie de `val`) est inférieur à 10,
- `p_moins` prend en argument un `param_t`, soustrait 1 à `val` attend `sleep_p_moins` puis affiche la valeur courante de `val` tant que `val` (ou une copie de `val`) est supérieur à -10,

– question 1 –

Question 2

Implémenter dans un supposé `prod_main.c` une fonction `main`. Le programme prend en argument la valeur de départ, le temps de pause de `p_plus` et le temps de pause de `p_moins` et crée deux threads. Le premier appelle `p_plus` et le second `p_moins`. Le programme affiche la valeur finale avant de retourner.

– question 2 –

Question 3

Écrire le Makefile permettant de générer l'exécutable `prod` (sans oublier la règle `clean`).