université de BORDEAUX

Collège Sciences et technologies

Année	2019-2020	**Type**	Examen
Licence	Informatique		
Code UE	4TIN503EX	**Épreuve**	Programmation Système
Date	20/12/2019	**Documents**	Non autorisés
Début	11h30	**Durée**	1h30

La plupart des questions peuvent être traitées même si vous n'avez pas répondu aux précédentes. *À la fin du sujet, vous trouverez un memento vous rappelant la syntaxe de quelques fonctions utiles.*

1 Question de cours

Décrivez ce que produit le programme suivant sur la sortie standard. Expliquez précisément.

```c
int main (int argc, char *argv[])
{
  printf ("Avant\n");
  execl ("/bin/echo", "echo", "milieu", NULL);
  printf ("Après\n");
  return 0;
}
```

Et si on retire les `'\n'` qui terminent les chaines en paramètre à `printf` ?

2 Processus légers (*threads*)

On dispose d'un module de manipulation d'images, similaire à celui vu en TP. On s'intéresse tout particulièrement à la fonction `compute_histogram` qui calcule, pour chaque valeur de luminance comprise entre 0 et 255, le nombre de pixels lui correspondent. Voici le code de cette fonction :

```c
unsigned int image [HEIGHT][WIDTH]; // on suppose ces dimensions très grandes
unsigned int histogram [256];

// la fonction luminance renvoie une valeur entre 0 et 255
unsigned char luminance (unsigned color);

void compute_histogram (void)
{
  memset (histogram, 0, 256 * sizeof(int));

  for (int y = 0; y < HEIGHT; y++)
    for (int x = 0; x < WIDTH; x++)
      histogram [luminance (image [y][x])] ++;
}
```

Le code de la fonction `luminance` n'est pas utile ici. En revanche on sait que le temps de calcul ne dépend pas de la valeur du paramètre. On souhaite accélérer ce calcul sur une machine disposant de 16 cœurs au moyen de *threads*.

Question 1 Donnez une version parallèle de la fonction `compute_histogram`. Expliquez votre stratégie de répartition du travail entre les *threads*. Dans cette première version, on ne se préoccupe pas des accès mémoire concurrents.

Question 2 Quelle est l'instruction dans le code qui pose problème si elle est exécutée par plusieurs threads simultanément ? Pourquoi ? Le cas échéant, donnez une solution simple permettant de corriger le problème et indiquez comment il faut modifier le code.

Question 3 On souhaite maintenant une version performante du code. Dans l'état actuel, discutez des performances obtenues en fonction de la durée du calcul de la luminance.

Proposez une solution dans laquelle les *threads* calculent réellement en parallèle sans se gêner durant les itérations. Donnez le code complet.

3 Processus et tubes

On souhaite disposer d'un programme `as_input` qui prenne en premier argument un nom de commande, suivi d'une liste de noms de fichiers, et exécute la commande de manière à ce que le contenu successif de tous les fichiers lui soit envoyé sur l'entrée standard. Ainsi, l'exécution de `as_input cmd f1 f2 f3` est équivalente à la commande shell `cat f1 f2 f3 | cmd`.

Donnez le code du programme `as_input`. Vous pouvez utiliser la commande existante `cat` pour vous aider.

4 Signaux

On dispose d'une fonction `afficher_image (int n)` permettant de rafraîchir une animation à l'écran en affichant la n-ième image graphique d'un sprite. On a écrit une première version permettant d'appeler périodiquement cette fonction ci-dessous :

```c
volatile int i = 0;

void handler (int sig)
{
  i++; alarm (2);
}

int main ()
{
  sigaction(SIGALRM, handler); // syntaxe simplifiée

  alarm (2);
  while (1) {
    afficher_image (i);
    pause ();
  }
  return 0;
}
```

Question 1 Que fait le code précédent (expliquez votre réponse) ?

1. Il affiche l'image 0, puis le programme se termine
2. Il affiche immédiatement l'image 0, puis les images successives (1, 2, etc.) avec deux secondes entre chaque
3. Il affiche l'image 0 après deux secondes, puis les images successives (1, 2, etc.) avec deux secondes entre chaque

Question 2 Que se passe-t-il si l'exécution de la fonction `afficher_image` dure légèrement plus de deux secondes ?

1. Les images sont toutes affichées, mais sont espacées de plus de deux secondes
2. Seules les images paires s'affichent, à raison d'une toutes les quatre secondes
3. La fonction `afficher_image` est appelée toutes les deux secondes, mais aucune image n'a le temps d'être affichée complètement

Expliquez votre réponse.

Question 3 Modifiez le code précédent pour que, lorsque la durée d'affichage de l'image i excède deux secondes, l'exécution soit interrompue pour démarrer celle de l'image $i+1$ à temps, c'est-à-dire deux secondes après le début de l'affichage de la précédente.

Question 4 On souhaite désormais ne plus interrompre un affichage dont la durée excède deux secondes. Dans ce cas, on enchaînera sur l'image $i+2$. Par ailleurs, si la durée de l'affichage s'apprête à dépasser quatre secondes, on doit l'interrompre immédiatement pour enchaîner sur l'image $i+2$. Indiquez les changements à apporter au code précédent.

Memento

```c
int pipe(int fildes[2]);
int dup2(int oldfd, int newfd);
int sigsetjmp(sigjmp_buf env, int savemask);
void siglongjmp(sigjmp_buf env, int val);
int execlp(const char *file, const char *arg0, ...);
int execvp(const char *file, char *const argv[]);
pid_t waitpid(pid_t pid, int *stat_loc, int options);
int sigaction(int sig, void (*_handler)(int)); // Syntaxe simplifiée vue en co
int sigprocmask(int how, sigset_t *set, sigset_t *oset);
int sigsuspend(const sigset_t *sigmask);
```