Pale INF583 – Operating Systems

2010–2011

- The exam lasts 3 hours.
- All documents are authorized.
- Programs must be commented and every answer must be justified

Exercise 1

Question 1

Describe the mathematical function implemented by the following program, and describe (shortly) the role of each instruction or declaration sequence immediately following a comment of the form `// C?`. You may simply write the number of the comment, followed by your explanation. For example:

C0: declaration of the constants for the number of threads and the size of the vectors.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

// C0
#define NUMTHRDS 4
#define VECLEN 100

// C1
typedef struct {
  double      *a;
  double      *b;
  double     sum;
  int     veclen;
} DOTDATA;

// C2
DOTDATA dotstr;

// C3
pthread_t callThd[NUMTHRDS];

// C4
pthread_mutex_t mutexsum;

// C5
void *dotprod(void *arg) {
  int i, start, end, len;
  long offset;
  double mysum, *x, *y;
  offset = (long)arg;

  // C6
  len = dotstr.veclen;
  start = offset*len;
  end   = start + len;
  x = dotstr.a;
  y = dotstr.b;

  // C7
  mysum = 0;
  for (i=start; i<end ; i++)
    mysum += (x[i] * y[i]);

  // C8
  pthread_mutex_lock (&mutexsum);
  dotstr.sum += mysum;
  pthread_mutex_unlock (&mutexsum);

  pthread_exit((void*)0);
}

int main (int argc, char *argv[]) {
  long i;
  double *a, *b;
  void *status;
  pthread_attr_t attr;

  // C9
  a = (double*) malloc (NUMTHRDS*VECLEN*sizeof(double));
  b = (double*) malloc (NUMTHRDS*VECLEN*sizeof(double));

  for (i=0; i<VECLEN*NUMTHRDS; i++) {
    a[i]=1;
    b[i]=a[i];
  }

  dotstr.veclen = VECLEN;
  dotstr.a = a;
  dotstr.b = b;
  dotstr.sum=0;

  pthread_mutex_init(&mutexsum, NULL);

  // C10
  for(i=0;i<NUMTHRDS;i++)
    pthread_create(&callThd[i], &attr, dotprod, (void *)i);

  // C11
  for(i=0;i<NUMTHRDS;i++)
    pthread_join(callThd[i], &status);

  // C12
  printf ("Sum =  %f \n", dotstr.sum);

  free (a);
  free (b);

  pthread_mutex_destroy(&mutexsum);

  pthread_exit(NULL);
}
```

Question 2

Why does the accumulation take place in the private variable `mysum` and not in the field `dotstr.sum` shared among all threads?

Question 3

Propose two methods to improve performance through the elimination of the mutex, and briefly explain the associated changes on the program in both cases.

Exercise 2

This is a multiple choice questionnaire about C and the system. Write down the number of the selected answer.

Grading: 1 point per correct answer, 0 points if no answer, -0.5 points per incorrect answer. The grade between -2 et 4 is then scaled according to the number of points allocated to the exercise.

1. What does the following C statement represent?

   ```
   int *p(void*);
   ```

   (1) a function pointer without arguments

   (2) a function pointer with an argument of any pointer type

   (3) a function without arguments returning a pointer to int

   (4) a function with an argument of any pointer type returning a pointer to int

2. What is the value of `x` after executing the following statements?

   ```
   int x=1, y=2; int *p=&y; int **q=&p;
   *q=&x; **q+=*p;
   ```

   (1) 1 (2) 2 (3) 3 (4) 4

3. What is the purpose of the following C code?

   ```
   #include <stdio.h>
   int main (int argc, char *argv[]) {
     char buf[100];
     FILE *fd=fopen ("toto", "r");
     read (fileno(fd), buf, 100);
   }
   ```

   (1) it opens a file named `toto`, read only, and reads 100 bytes

   (2) it opens a file named `toto`, read/write, and reads 100 bytes

   (3) nothing, the syntax is incorrect

   (4) nothing sensible, the argument types are incorrect

4. When a process using more than half of the free memory calls the `fork()` system call, what outcome is *not* expected to happen?

   (1) the process is killed instantly to free some memory

   (2) another process is killed instantly or later

   (3) the system call returns -1 and `errno` is set to ENOMEM (out of memory)

   (4) fork returns as normal

Exercice 3

We would like to build a local instant messaging system. A directory contains several text files, each one containing the conversations of one of the chat rooms hosted by the messaging server.

A “client” registered to a chat room is a process capable of writing and reading conversations in the file corresponding to this room. In this exercise, we assume that there is only one active room, and that the associated file is called “salon.txt”.

A client writes into “salon.txt” when the user wishes to contribute some text to the conversation. The client tests in an endless loop whether “salon.txt” has been modified by other clients.

- if not, it does nothing;
- if so, it reads the last characters added to the file and prints them to the standard output.

Question 4

This operating mode requires a concurrency issue to be resolved. Which one? How to solve it?

Question 5

The client tests the “salon.txt” file in an endless loop. Write a C function implementing this waiting loop, with the appropriate system call.

Question 6

Explain in a few sentences (maximum) what is *active waiting*, and give a different example from the one of the previous waiting loop.

Question 7

We would like to set up a passive waiting method instead.

The proposed method is based on the SIGUSR1 signal, to detect when a message has been posted to the chat room. Reception of this signal by a client means that “salon.txt” has been modified by another client.

Which function should be used to react to SIGUSR1 in the client? Is a multithreaded program needed? Why?

Question 8

Modify the preceding function and waiting loop to implement this detection mechanism for signal SIGUSR1.

Question 9

Which precautions are required in the client when it is reading the user’s input from the standard input?

Question 10

We now study the emission of signal SIGUSR1.

Who is responsible for emitting SIGUSR1 to signal the modification of “salon.txt”? What is the main difficulty when using signal SIGUSR1 to signal a modification of “salon.txt”?

Question 11

How can each client obtain and update the list of clients to which SIGUSR1 should be delivered? Detail the mechanism (no need to write a C program).

Question 12

The client is not yet capable of printing incomming messages while it is waiting on the standard input (when the user is typing a new message). Propose a solution based on multiple processes allowing incomming messages to be printed during the interactive input. Describe your solution without a C program (text, picture).

Question 13

To improve the previous solution, you will make sure that the child processes reach a clean termination state when the main client process ends with a call to `exit()` or if it is killed by a catchable/maskable signal (such as SIGINT, associated to Ctrl-C for example).

Question 14

Propose a way for multiple users connected through the Internet may use the messaging system in a secure way, although the program does not use any socket.

Question 15

How to “start to secure” the messaging system, i.e., to autorize only specific users to view or contribute to specific rooms? Give an example. What are the limitations of such an approach?