INF 2005 Programmation orientée objet avec C++

Module 5 - Solutions

1.

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace std;
double *s_ltab;
long ltab_n;
void Sommation(double b)
{
  double l1b = 1.0 / log(b);
  double s = 1.0;
  for (long k=0; k<ltab_n; ++k)
  {
    s_ltab[k] = log(1.0+s) * l1b; // == log_b(1+1/2^k)
    s *= 0.5;
  }
}
```

2.

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace std;
int main()
{
  float a, b, c;
  cout << "Entrer les coefficients de l'equation:";
  cin >> a >> b >> c;
  if (a == 0)
  {
    cout << "votre équation n'est pas du second degré: a == 0 \n";
    return 0;
  }
  cout << "l'équation est: " << a << " x^2 + " << b << " x + " <<
c << " = 0 \n";
  double d, xl, x2;
  d = b*b - 4*a*c;// the discriminant
  if (d < 0)
  {
    cout << "cette équation n'a pas de solution réelle: d < 0\n";
    return 0;
    xl = (-b + sqrt(d))/(2*a);
    x2 = (-b - sqrt (d)) / (2*a);
    cout << "les solutions sont: " << xl << "," << x2 <<endl;
  }
}
```

3.

Le client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <iostream>
#define MAX_LINE 100
#define LINE_ARRAY_SIZE (MAX_LINE+3)
#define NEWLINE "\n"
using namespace std;

int main()
{

  int socketDescriptor;
  int numRead;
  unsigned short int serverPort;
  struct sockaddr_in serverAddress;
  struct hostent *hostInfo;
  struct timeval timeVal;
  fd_set readSet;
  char buf[LINE_ARRAY_SIZE], c;

  cout << "entrer le nom du serveur ou son adresse IP: ";
  memset(buf, 0x0, LINE_ARRAY_SIZE); // Zero out the buffer.
  cin.get(buf, MAX_LINE, '\n');

// gethostbyname() takes a host name or ip address in "numbers and
// dots" notation, and returns a pointer to a hostent structure,
// which we'll need later. It's not important for us what this
// structure is actually composed of.

  hostInfo = gethostbyname(buf);
  if (hostInfo == NULL) {

    cout << "probleme pour connaitre le host: " << buf << "\n";
    exit(1);

  }

  cout << "Entrer le numero de port du serveur: ";
  cin >> serverPort;
  cin.get(c);

// Creation de socket

  socketDescriptor = socket(PF_INET, SOCK_DGRAM, 0);
  if (socketDescriptor < 0) {

    cerr << "ne peut pas créer de socket\n";
    exit(1);

  }

  serverAddress.sin_family = hostInfo->h_addrtype;
  memcpy((char *) &serverAddress.sin_addr.s_addr,
  hostInfo->h_addr_list[0], hostInfo->h_length);
  serverAddress.sin_port = htons(serverPort);
  cout << "\nEnter some lines, and the server will modify them
and\n";
  cout << "send them back. When you are done, enter a line
with\n";
  cout << "just a dot, and nothing else.\n";
  cout << "If a line is more than " << MAX_LINE << " characters,
then\n";
  cout << "only the first " << MAX_LINE << " characters will be
used.\n\n";
  cout << "Input: ";
  memset(buf, 0x0, LINE_ARRAY_SIZE); // Zero out the buffer.
  cin.get(buf, MAX_LINE, '\n');
  while (cin.get(c) && c != '\n');
  // Stop when the user inputs a line with just a dot.
  while (strcmp(buf, ".")) {

  // Send the line to the server.
    if (sendto(socketDescriptor, buf, strlen(buf), 0,

    (struct sockaddr *) &serverAddress,
     sizeof(serverAddress)) < 0) {

       cerr << "cannot send data ";
       close(socketDescriptor);
       exit(1);

   }

// wait until answer comes back, for up to 1 second

  FD_ZERO(&readSet);
  FD_SET(socketDescriptor, &readSet);
  timeVal.tv_sec = 1;
  timeVal.tv_usec = 0;

  if (select(socketDescriptor+1, &readSet, NULL, NULL, &timeVal))
{

  // Read the modified line back from the server.
  memset(buf, 0x0, LINE_ARRAY_SIZE); // Zero out the buffer.
  numRead = recv(socketDescriptor, buf, MAX_LINE, 0);

    if (numRead < 0) {

      cerr << "pas de reponse du serveur?";
      close(socketDescriptor);
      exit(1);

     }

    cout << "modification: " << buf << "\n";

    }

    else {

      cout << "** le serveur ne reponds pas dans une seconde.\n";

    }

  cout << "Input: ";
  memset(buf, 0x0, LINE_ARRAY_SIZE);
  cin.get(buf, MAX_LINE, '\n');
  while (cin.get(c) && c != '\n')

  ;
  }
  close(socketDescriptor);
  return 0;
}
```

Le serveur

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <iostream>
#include <arpa/inet.h>
#define MAX_MSG 100
using namespace std;

int main()
{
  int listenSocket, i;
  unsigned short int listenPort;
  socklen_t clientAddressLength;
  struct sockaddr_in clientAddress, serverAddress;
  char line[(MAX_MSG+1)];
  cout << "entrer le numero de port d'écoute (between 1500 and
65000): ";
  cin >> listenPort;

// Create socket for listening for client connection requests.

  listenSocket = socket(AF_INET, SOCK_DGRAM, 0);

  if (listenSocket < 0) {

  cerr << "ne peut pas créer une socket d'écoute";
  exit(1);

}

  serverAddress.sin_family = PF_INET;
  serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
  serverAddress.sin_port = htons(listenPort);

  if (bind(listenSocket,

  (struct sockaddr *) &serverAddress,

  sizeof(serverAddress)) < 0) {

    cerr << "ne peut pas lier la socket";
    exit(1);
    }

  listen(listenSocket, 5);
  cout << "attente de requete sur le port ouvert " << listenPort
<< "\n";

  while (1) {

    clientAddressLength = sizeof(clientAddress);
    memset(line, 0x0, (MAX_MSG+1));

    if (recvfrom(listenSocket, line, MAX_MSG, 0,

    (struct sockaddr *) &clientAddress,

    &clientAddressLength) < 0) {

      cerr << " I/O Problemes";
      exit(1);

    }

  cout << " from " << inet_ntoa(clientAddress.sin_addr);

  // Show the client's port number.

  cout << ":" << ntohs(clientAddress.sin_port) << "\n";

  // Show the line

  cout << " Received: " << line << "\n";

  // Convert line to upper case.

  for (i = 0; line[i] != '\0'; i++)

    line[i] = toupper(line[i]);

  if (sendto(listenSocket, line, strlen(line) + 1, 0,

  (struct sockaddr *) &clientAddress,

  sizeof(clientAddress)) < 0)

    cerr << "Error: ne peut pas modifier les données";
    memset(line, 0x0, (MAX_MSG+1)); // set line to all zeroes

  }
}
```

4.

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

void *print_message_function( void *ptr );

main()

{

  pthread_t thread1, thread2;
  char *message1 = "Thread 1";
  char *message2 = "Thread 2";
  int iret1, iret2;

// Création de deux threads qui vont afficher chacun leur message

  iret1 = pthread_create( &thread1, NULL, print_message_function,
(void*)
  message1);
  iret2 = pthread_create( &thread2, NULL, print_message_function,
(void*)
  message2);

  // On attend que les threads soient prêts

  pthread_join( thread1, NULL);
  pthread_join( thread2, NULL);

  // On affiche la valeur de retour des threads
  cout << "Thread 1 returns: \n" << iret1;
  cout << "Thread 2 returns: \n" << iret2;
  exit(0);

}

void *print_message_function( void *ptr )

{

  char *message;
  message = (char *) ptr;
  cout << "%s \n" << message;

}
```