Compte Rendu APP1

EL-BOUCH ISMAIL JIANG Yilun MAMADOU DIALLO

client-tutoriel.c

```c
#include "client.h"
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

#define MAXMSG MAXREP

int main()
{
    char reponse[MAXREP];
    char message[MAXMSG];

    // Affiche les échanges avec le serveur (false pour désactiver)
    mode_debug(true);

    puts("Bienvenue dans cette introduction à AppoLab !\n"
         "AppoLab est un serveur d'exercices algorithmiques que vous allez devoir \n"
         "utiliser pour vos APPs. Je vais vous guider pas à pas pour que vous \n"
         "puissiez vous débrouiller tout·e seul·e.");

    puts("Le client va maintenant tenter de se connecter automatiquement au serveur \n"
         "AppoLab. Il vous faut bien entendu pour cela une connection internet.\n"
         "(En cas de problème, modifiez le fichier client-introduction.c pour \n"
         "utiliser le port 443 au lieu de 9999 sur la ligne 'connexion'.)");

    // Connexion au serveur AppoLab
    connexion("im2ag-appolab.u-ga.fr", 9999);
    /* connexion("im2ag-appolab.u-ga.fr", 443); */

    puts("Si tout va bien, vous devez avoir reçu le message de bienvenue d'AppoLab.\n"
         "Si non, arrêtez ce programme (avec Ctrl-C) et demandez de l'aide à un·e enseignant·e.\n"
         "(Si vous avez des difficultés à lire certains messages, modifiez les couleurs dans client.h et recompilez.)");

    puts("Comme indiqué, commencez par vous loguer avec l'identifiant et le mot de \n"
         "passe qui vous ont été fournis. Pour les étudiants d'INF301, le login est\n"
         "votre numéro d'étudiant·e, et le mot de passe votre nom en majuscule.\n"
         "Entrez les au clavier ainsi :\n"
         "login 12345678 \"MOT DE PASSE\"");
```

```
40
41      strcpy(message, "login 12100255 JIANG");
42      envoyer_recevoir(message, reponse);
43
44      puts("Bravo, vous venez de vous identifier auprès du serveur !\n"
45           "Comme vous pouvez le voir ce programme trace tout ce que vous envoyez au
        \n"
46           "serveur sur les lignes commençant par <<<envoi<<<, et tout ce que répond
        le \n"
47           "serveur sur des lignes commençant par >>>recu >>>.");
48
49      puts("Vous êtes maintenant prêt·e à lancer le premier exercice qui se nomme
        'tutoriel'.\n"
50           "Lancez le grâce à la commande 'load' ainsi :\n"
51           "load tutoriel");
52
53      strcpy(message, "load tutoriel");
54      envoyer_recevoir(message, reponse);
55
56      puts("Voilà, vous venez de lancer votre premier exercice...\n"
57           "Suivez les consignes de l'exercice maintenant.");
58
59      strcpy(message, "help");
60      envoyer_recevoir(message, reponse);
61
62      strcpy(message, "start");
63      envoyer_recevoir(message, reponse);
64
65      strcpy(message, "oui");
66      envoyer_recevoir(message, reponse);
67
68      strcpy(message, "4");
69      envoyer_recevoir(message, reponse);
70
71      strcpy(message, "blanc");
72      envoyer_recevoir(message, reponse);
73
74      strcpy(message, "Pincemoi");
75      envoyer_recevoir(message, reponse);
76
77      strcpy(message, "tutoriel");
78      envoyer_recevoir(message, reponse);
79
80      attendre();
81
82      return 0;
83  }
84
```

projetX.c

```c
#include "client.h"
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

#define MAXMSG MAXREP

int main()
{

    char *serveur = "im2ag-appolab.u-ga.fr";
    int port = 9999;

    char reponse[MAXREP];
    char message[MAXMSG];

    mode_debug(true);

    puts("Bienvenue dans le client interactif d'AppoLab");
    puts("Connection à AppoLab dans le client interactif d'AppoLab ...");

    // Connexion au serveur AppoLab
    connexion(serveur, port);

    strcpy(message, "login 12100255 JIANG");
    envoyer_recevoir(message, reponse);

    strcpy(message, "load projetX");
    envoyer_recevoir(message, reponse);

    strcpy(message, "help");
    envoyer_recevoir(message, reponse);

    strcpy(message, "depart");
    envoyer_recevoir(message, reponse);

    strcpy(message, "veni vidi vici");
    envoyer_recevoir(message, reponse);

    strcpy(message, "load crypteMove");
    envoyer_recevoir(message, reponse);

    return 0;
}

```

crypteMove.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>
#include "client.h"

#define MAXMSG MAXREP

int algoCrypteMove(char TXT[], char ENC[]){
    int lenTXT = strlen(TXT);
    // Contient la position du caractère à ajouter dans le tableau encodé
    int avancementENC = 0;
    // Contient la position de la première lettre dans le tableau TXT
    int position_start = 0;
    // Contient la position de la dernière lettre dans le tableau TXT
    int position_finish = lenTXT - 1;

    char X;

    // Tant qu'il reste des lettres dans le texte à chiffrer
    while (position_finish ⩾ position_start)
    {
        // Concatène la lettre première lettre de TXT à ENC
        ENC[avancementENC] = TXT[position_start];
        // Pour ne plus prendre en compte le caractère ajouté à ENC
        position_start++;
        // Calcul de X, à partir de la lettre concaténée à ENC
        X = ENC[avancementENC] % 8;
        // Ajoute 1 à la taille du texte Encodé pour pouvoir concaténer au bonne endroit les futurs lettres
        avancementENC++;

        if (X > 0 && ((position_finish - position_start) ⩾ X)){
            // nb cases disponible avant = position_start
            // Si assez de place pour stocker la fin de la chaine au début du tableau
            if(position_start ⩾ (position_finish - (position_start + X - 1))){
                // Initialisation compteur j, pour déplacer les élément de fin de chaine de caractère au début
                int j = position_start + X;
                // Calcul nouvelle position start de la chaine de caractère
                position_start-=position_finish - (position_start + X - 1);
                // Calcul la nouvelle position final de la chaine de caractère
                position_finish = j - 1;
                // Boucle pour placer les X caractères finaux au début de la chaine
                for (int i = position_start; i < (position_finish - X + 1); i++)
                {
                    TXT[i] = TXT[j];
                    j++;
                }
            }
```

```
            // Si assez de place pour stocker à la fin du tableau
            else if(MAXMSG ⩾ position_finish + X){
                int j = position_start;
                position_start += X;
                position_finish += X;
                for (int i = position_finish - X + 1; i ⩽ position_finish; i++){
                    TXT[i] = TXT[j];
                    j++;
                }
            }
            // Si de la place null part renvoie 1
            else{
                return 1;
            }
        }
    }
    return 0;
}

void crypteMove(char TXT[], char ENC[]){
    if(algoCrypteMove(TXT, ENC) == 1)
        printf("Le texte est de trop grande taille ... Il n a pas pu etre
dechiffre\n");
}

int main(){
    char reponse[MAXREP];
    char message[MAXMSG];

    // Affiche les échanges avec le serveur (false pour désactiver)
    mode_debug(true);

    connexion("im2ag-appolab.u-ga.fr", 9999);
    envoyer_recevoir("login 12100255 JIANG", reponse);

    envoyer_recevoir("load crypteMove", reponse);
    envoyer_recevoir("help", reponse);
    crypteMove(reponse, message);
    envoyer_recevoir("start", reponse);
    envoyer_recevoir(message, reponse);
    envoyer_recevoir("exit", reponse);

    deconnexion();

    return 0;
}
```

BayOfPigs.c

```
#include "client.h"
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>
```

```c

#define MAXMSG MAXREP

void inverse(char *TXT, int x)
{
    char k[MAXREP];
    int j = 0;
    int TXTlen = strlen(TXT);
    for (int l = TXTlen - x; l ⩽ TXTlen; l++)
    {
        k[l] = TXT[j];
        j++;
    }
    j = 0;
    for (int a = x; a < TXTlen; a++)
    {
        k[a] = TXT[j];
        j++;
    }
    k[TXTlen] = '\0';
    strcpy(TXT, k);
}

void elem(char *TXT, char *ENC)
{
    char k[MAXREP];
    int ENClen = strlen(ENC);
    k[0] = ENC[ENClen - 1];
    ENC[ENClen - 1] = '\0';
    for (unsigned long j = 0; j ⩽ strlen(TXT); j++)
    {
        k[j + 1] = TXT[j];
    }

    strcpy(TXT, k);
}

int main()
{

    char reponse[MAXREP]; // pour stocker la réponse du serveur
    // char message[MAXMSG]; // pour stocker le message à envoyer au serveur
    //  Affiche les échanges avec le serveur (false pour désactiver)
    mode_debug(true);

    // Connexion au serveur AppoLab
    connexion("im2ag-appolab.u-ga.fr", 9999);
    // utilisez le port 443 en cas de problème sur le 9999
    /* connexion("im2ag-appolab.u-ga.fr", 443); */

    // Remplacez <identifiant> et <tab de passe> ci dessous.

    envoyer_recevoir("login 12100255 JIANG", reponse);
    printf("%s\n", reponse);
    envoyer_recevoir("load BayOfPigs", reponse);
    envoyer_recevoir("help", reponse);
```

```
    envoyer_recevoir("start", reponse);
    envoyer_recevoir("Par otuam eriet", reponse);

    char TXT[MAXMSG];
    char ENC[MAXMSG];

    TXT[0] = '\0';
    strcpy(ENC, reponse);
    int ENClen = strlen(ENC);
    int TXTlen = strlen(TXT);
    int d = ENC[ENClen - 1];
    int i = d % 8;

    while (strlen(ENC) ≠ 0)
    {
        TXTlen = strlen(TXT);
        if (i ⩽ TXTlen)
        {
            inverse(TXT, i);
        }
        elem(TXT, ENC);
        ENClen = strlen(ENC);
        d = ENC[ENClen - 1];
        i = i % 8;
    }

    envoyer_recevoir(TXT, reponse);
    return 0;
}

```

crypteSeq.c

```c
#include "client.h"
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

#define MAXMSG MAXREP

void crypteSequence(char *message, char *messageDecrypte)
{
    char sequence[10000];
    int x, y;
    int l;
    int indiceCase, longueurMessage, longueurSequence, longueurTableau,
indicesMessage;

    indicesMessage = 0;
    l = message[indicesMessage];
    longueurSequence = 0;
    longueurTableau = 0;
    sequence[longueurSequence] = l;
```

```
    longueurSequence++;
    messageDecrypte[longueurTableau] = l;
    longueurTableau++;
    longueurMessage = strlen(message);
    int booleanTemp;

    for (y = 0; y < longueurMessage; y++)
    {
        indicesMessage++;
        l = message[indicesMessage];
        booleanTemp = 1; // si n'existe pas

        for (x = 0; x < longueurSequence; x++)
        {
            if (l == sequence[x]) // si existe
            {
                booleanTemp = 0;
                indiceCase = x; // on stocke num de sa case
            }
        }
        if (booleanTemp)
        { // if la premiere fois que cette lettre (a) apparait, on va l'ajtxt_decryer à la fin nde sequence et de messageDecrypte
            sequence[longueurSequence] = l;
            longueurSequence++;
            messageDecrypte[longueurTableau] = l;
            longueurTableau++;
        }
        else
        { // si la lettre existe (a) deja en sequence, on prend lettre qui la precede et l'ajtxt_decrye à fin messageDecrypte, puis on la rend au debut de sequence
            if (indiceCase != 0)
            {
                messageDecrypte[longueurTableau] = sequence[indiceCase - 1];
                longueurTableau++;
            }

            else
            {
                messageDecrypte[longueurTableau] = sequence[longueurSequence - 1];
                longueurTableau++;
            }

            if (indiceCase != longueurSequence - 1)
            { // si a n'existe pas à la fin de sequence
                for (x = indiceCase; x < longueurSequence - 1; x++)
                {
                    sequence[x] = sequence[x + 1];
                }
                sequence[longueurSequence - 1] = l;
            }
        }
    }
}
```

```
73
74  void decrypteSequence(char *rep, char *messageDecrypte)
75  {
76
77      int a, b;
78      int longueurMessage, longueurSequence, longueurTableau;
79      char sequence[10000];
80
81      int indicesMessage = 83;
82      char caractre = rep[indicesMessage];
83
84      longueurSequence = 0;
85      longueurTableau = 0;
86      sequence[longueurSequence] = caractre;
87      longueurSequence++;
88      messageDecrypte[longueurTableau] = caractre;
89      longueurTableau++;
90      longueurMessage = strlen(rep);
91      int indiceCase;
92      int booleanTemp;
93      for (b = 83; b < longueurMessage; b++)
94      {
95          indicesMessage++;
96          caractre = rep[indicesMessage];
97          booleanTemp = 1;
98          for (a = 0; a < longueurSequence; a++)
99          {
100             if (caractre == sequence[a])
101             {
102                 booleanTemp = 0;
103                 indiceCase = a;
104             }
105         }
106         if (booleanTemp)
107         {
108             sequence[longueurSequence] = caractre;
109             longueurSequence++;
110             messageDecrypte[longueurTableau] = caractre;
111             longueurTableau++;
112         }
113         else
114         {
115             if (indiceCase != longueurSequence - 1)
116             {
117                 messageDecrypte[longueurTableau] = sequence[indiceCase + 1];
118                 longueurTableau++;
119             }
120             else
121             {
122                 messageDecrypte[longueurTableau] = sequence[0];
123                 longueurTableau++;
124             }
125
126             if (indiceCase != longueurSequence - 1)
127             {
128                 if (indiceCase + 1 != longueurSequence - 1)
```

```
                {

                    indiceCase++;
                    caractre = sequence[indiceCase];
                    for (a = indiceCase; a < longueurSequence - 1; a++)
                    {
                        sequence[a] = sequence[a + 1];
                    }
                    sequence[longueurSequence - 1] = caractre;
                }
            }
            else
            {
                caractre = sequence[0];
                for (a = 0; a < longueurSequence - 1; a++)
                {
                    sequence[a] = sequence[a + 1];
                }
                sequence[longueurSequence - 1] = caractre;
            }
        }
    }
}

void decrypteMove(char *rep, char *msg)
{
    int x, y;
    int len, a;
    char tableau[MAXMSG];
    int mod, indicesMessage;
    char txt[MAXREP];

    len = strlen(rep);
    for (x = 0; x < len; x++)
    {
        txt[x] = rep[x];
    }

    a = len;
    indicesMessage = len - 1;

    for (y = 0; y < a; y++)
    {
        mod = txt[indicesMessage] % 8; // mod de dernier caractere de text à
decrypter
        len = strlen(tableau);

        if (mod =≠ 0 && mod < len)
        {
            for (x = 0; x < mod; x++)
            {

                tableau[len] = tableau[x]; // pour x=mod, prendre les x premiers
elements et les deplacer à la fin
                len = strlen(tableau);
            }
```

```
183
184             for (x = 0; x < len; x++)
185             {
186                 if (x < len - mod)
187                 {
188                     tableau[x] = tableau[x + mod];
189                 }
190                 else
191                 {
192                     tableau[x] = '\0';
193                 }
194             }
195         }
196         len = strlen(tableau);
197         tableau[len] = txt[indicesMessage];
198         len = strlen(tableau);
199         indicesMessage--;
200     }
201     for (x = 0; x < len; x++)
202     {
203         msg[x] = tableau[len - 1 - x];
204     }
205 }
206
207 int main()
208 {
209     char rep[MAXREP]; // pour stocker la rÃ©ponse du serveur
210     char msg[MAXMSG]; // pour stocker le msg Ã  envoyer au serveur
211     char messageDecrypte[MAXREP];
212     char messageDecrypte2[MAXREP];
213
214     mode_debug(true);
215
216     connexion("im2ag-appolab.u-ga.fr", 9999);
217     envoyer_recevoir("login 12100255 JIANG", rep);
218     envoyer_recevoir("load crypteSeq", rep);
219     envoyer_recevoir("Alice, nous avons besoin de toi. Decode mon msg au plus
    vite. Bob", rep);
220     envoyer_recevoir("start", rep);
221
222     decrypteMove(rep, msg);         // decryptage par methode crypteMove
223     crypteSequence(msg, messageDecrypte); // encrypter par methode cryptesequence
224     envoyer_recevoir(messageDecrypte, rep);
225     decrypteSequence(rep, messageDecrypte2); // decryptage par methode de
    cryptesequence
226     envoyer_recevoir(messageDecrypte2, rep);
227
228     printf("Fin de la connection au serveur\n");
229     return 0;
230 }
231
```

Northwoods.c

```c
#include "client.h"
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

#define MAXMSG MAXREP

void crypteSeq(char *message, char *messageDecrypte)
{
  char sequence[10000];
  int x, y;
  int l;
  int indiceCase, longueurMessage, longueurSequence, longueurTableau,
indicesMessage;

  indicesMessage = 0;
  l = message[indicesMessage];
  longueurSequence = 0;
  longueurTableau = 0;
  sequence[longueurSequence] = l;
  longueurSequence++;
  messageDecrypte[longueurTableau] = l;
  longueurTableau++;
  longueurMessage = strlen(message);
  int booleanTemp;

  for (y = 0; y < longueurMessage; y++)
  {
    indicesMessage++;
    l = message[indicesMessage];
    booleanTemp = 1;

    for (x = 0; x < longueurSequence; x++)
    {
      if (l == sequence[x])
      {
        booleanTemp = 0;
        indiceCase = x;
      }
    }
    if (booleanTemp)
    {
      sequence[longueurSequence] = l;
      longueurSequence++;
      messageDecrypte[longueurTableau] = l;
      longueurTableau++;
    }
    else
    {
      if (indiceCase != 0)
      {
        messageDecrypte[longueurTableau] = sequence[indiceCase - 1];
```

```
        longueurTableau++;
      }

      else
      {
        messageDecrypte[longueurTableau] = sequence[longueurSequence - 1];
        longueurTableau++;
      }

      if (indiceCase != longueurSequence - 1)
      {
        for (x = indiceCase; x < longueurSequence - 1; x++)
        {
          sequence[x] = sequence[x + 1];
        }
        sequence[longueurSequence - 1] = l;
      }
    }
  }
}

void decrypteSequence(char *rep, char *messageDecrypte)
{

  int x, y;
  char carac;
  int indiceCase, booleanTemp, longueurMessage, longueurSequence,
longueurTableau, indiceMessage;
  char sequence[10000];

  indiceMessage = 0;
  carac = rep[indiceMessage];
  longueurSequence = 0;
  longueurTableau = 0;
  sequence[longueurSequence] = carac;
  longueurSequence++;
  messageDecrypte[longueurTableau] = carac;
  longueurTableau++;
  longueurMessage = strlen(rep);

  for (y = 0; y < longueurMessage; y++)
  {
    indiceMessage++;
    carac = rep[indiceMessage];
    booleanTemp = 1;
    for (x = 0; x < longueurSequence; x++)
    {
      if (carac == sequence[x])
      {
        booleanTemp = 0;
        indiceCase = x;
      }
    }
    if (booleanTemp)
    {
      sequence[longueurSequence] = carac;
```

```
      longueurSequence++;
      messageDecrypte[longueurTableau] = carac;
      longueurTableau++;
    }
    else
    {
      if (indiceCase ≠ longueurSequence - 1)
      {
        messageDecrypte[longueurTableau] = sequence[indiceCase + 1];
        longueurTableau++;
      }
      else
      {
        messageDecrypte[longueurTableau] = sequence[0];
        longueurTableau++;
      }
      if (indiceCase ≠ longueurSequence - 1)
      {
        if (indiceCase + 1 ≠ longueurSequence - 1)
        {
          indiceCase++;
          carac = sequence[indiceCase];
          for (x = indiceCase; x < longueurSequence - 1; x++)
          {
            sequence[x] = sequence[x + 1];
          }
          sequence[longueurSequence - 1] = carac;
        }
      }
      else
      {
        carac = sequence[0];
        for (x = 0; x < longueurSequence - 1; x++)
        {
          sequence[x] = sequence[x + 1];
        }
        sequence[longueurSequence - 1] = carac;
      }
    }
  }
}

int main()
{
  char rep[MAXREP];
  int longueur;
  int x;
  char mot_de_pass[100];
  char messageDecrypte[MAXREP];
  char messageDecrypte2[MAXREP];
  char mot_de_pass2[200] = "There will be no Nineteen Eighty-Four";
  char tableau[MAXREP];

  mode_debug(true);

  connexion("im2ag-appolab.u-ga.fr", 9999);
```

```
  envoyer_recevoir("login 12100255 JIANG", rep);
  envoyer_recevoir("load Northwoods", rep);
  envoyer_recevoir("ok", rep);
  envoyer_recevoir("start", rep);
  envoyer_recevoir("hasta la victoria siempre", rep);
  decrypteSequence(rep, messageDecrypte);

  longueur = strlen(messageDecrypte);
  for (x = longueur - 29; x < longueur - 9; x++)
  {
    mot_de_pass[x - longueur + 29] = messageDecrypte[x];
  }

  envoyer_recevoir(mot_de_pass, rep);
  crypteSeq(mot_de_pass2, messageDecrypte2);
  envoyer_recevoir(messageDecrypte2, rep);
  longueur = strlen(rep);
  for (x = 24; x < longueur; x++)
  {
    tableau[x - 24] = rep[x];
  }

  decrypteSequence(tableau, messageDecrypte);
  envoyer_recevoir(messageDecrypte, rep);

  return 0;
}

```

Lien du projet

https://github.com/Marshellson/INF301/tree/main/APP1