Le protocole SMTP

par Benjamin Roux (Retour aux articles)

Date de publication : 12/02/2007

Dernière mise à jour :

Cet article va vous présenter le protocole SMTP.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

1 - Introduction

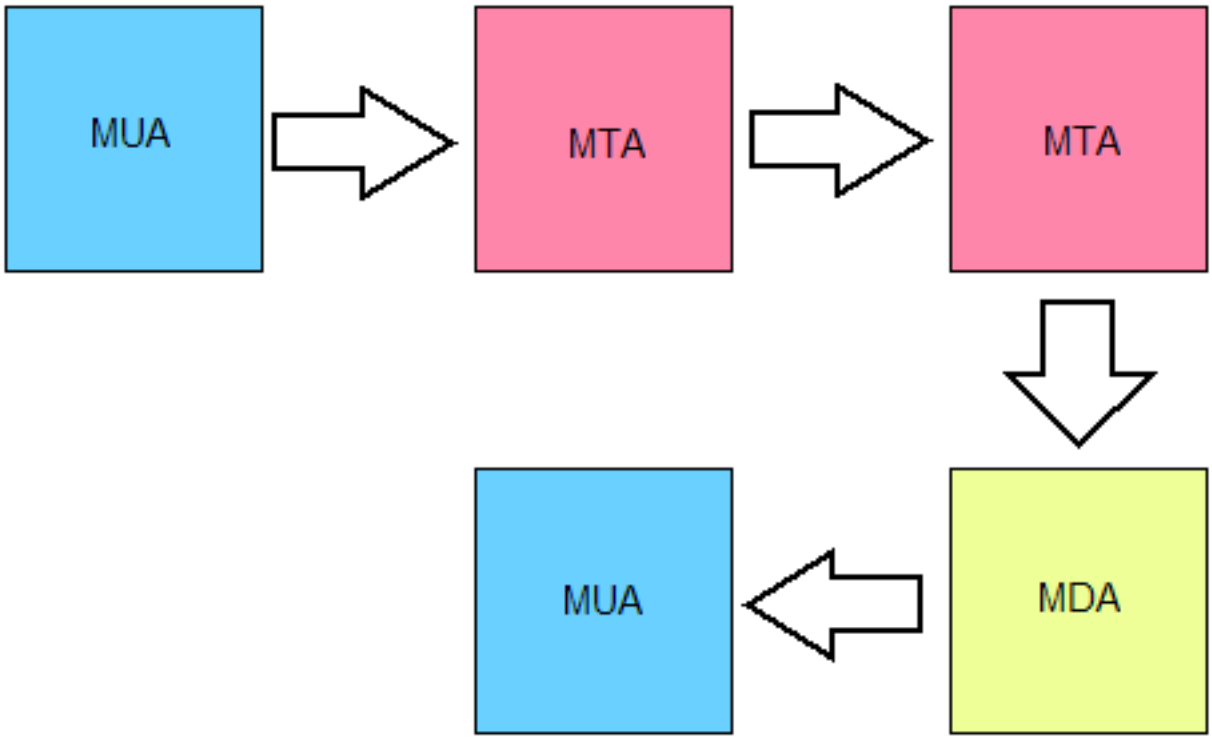
L'une des applications les plus utilisées d'Internet, est sans nul doute l'envoi de mails. Comme vous le savez sûrement, l'envoi de mail est régi par le protocole SMTP, qui signifie **Simple Mail Transfert Protocol**.
Mais savez vous exactement comment ce protocole fonctionne ? C'est ce que nous allons voir dans cet article.

*Avec l'essor de ce protocole, une extension a été créée, l'**Extended SMTP (ESMTP)**. Il n'y a pas de grande différence entre les 2.*
*Il devient de plus en plus difficile de trouver un serveur ne supportant pas l'**ESMTP**. En conséquence tous les exemples suivants seront sur un serveur supportant **ESMTP**.*

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

2 - Cheminement d'un email

Voici le cheminement classique d'un email.



Explication :

- **MUA** : Mail User Agent, c'est le client mail utilisé pour écrire le mail, le reçevoir (Outlook, Thunderbird...).
- **MTA** : Mail Transfert Agent, c'est l'agent qui va transférer votre mail vers le MTA de votre destinataire.
- **MDA** : Mail Delivery Agent, c'est l'agent chargé de délivrer le mail à votre destinaire

Une fois votre mail parti depuis votre **MUA**, il est envoyé à votre **MTA**, qui est chargé de le délivrer au **MTA** de votre destinataire. Il se peut que ces derniers soit les mêmes, et il se peut aussi que le mail transite par plusieurs **MTA** avant d'arriver au dernier. Les serveurs **MTA** communiquent entre eux par le protocole **SMTP**, d'où leur nom de **serveur SMTP**.
Une fois sur le dernier **MTA**, ce dernier le transmet au **MDA**, qui stocke le mail avant que le destinataire ne vienne le relever.

À l'image du monde réel, le **MTA** représente le bureau de poste qui est chargé de reçevoir tous les courriers, et d'ensuite les envoyer là où il faut, et si besoin est à un autre bureau de poste.
Le **MDA** représente la boîte aux lettres du client, où le courrier est stocké tant que le client n'est pas venu le relever.

Les serveurs **MDA** sont bien entendus protégés par des logins, mots de passes, pour éviter que tout le monde ne viennent lire le courrier de tout le monde. La récupération des messages se fait par les protocoles POP3/IMAP4 que nous verrons dans un prochain article.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

3 - Procédures SMTP

Voici les différentes procédures du protocole SMTP, de la simple ouverture de connexion, à l'envoi de mail. Nous allons voir les commandes associées à ces actions.

3.1 - Code de retour

À chaque envoi de commande, le serveur nous renverra un code de retour, selon la réussite ou l'échec de la commande.

Voici les différents codes de retour et leur signification.

- *211* État système, ou réponse d'aide système
- *214* Message d'aide [Informations sur l'utilisation d'un récepteur ou signification d'une commande non standard particulière ; utile seulement pour un utilisateur humain]
- *220* <domaine> Service disponible
- *221* <domaine> Canal de transmission en cours de fermeture
- *250* Action de messagerie effectuée, succès
- *251* Utilisateur non local ; réémission vers <route-directe> (avec relais automatique)
- *354* Début du corps du message ; arrêt par <CRLF>.<CRLF>
- *421* <domaine> Service non disponible, canal en fermeture [Réponse à émettre sur tous les canaux lorsque le système exécute une séquence d'arrêt]
- *450* Action non effectuée : boîte aux lettres non disponible [Ex. : boîte aux lettres occupée]
- *451* Action arrêtée : erreur de traitement
- *452* Action non effectuée : manque de ressources système
- *500* Erreur de syntaxe, commande non reconnue [y compris des erreurs de type "ligne de commande trop longue"]
- *501* Erreur de syntaxe dans les paramètres ou arguments
- *502* Commande non implémentée
- *503* Mauvaise séquence de commandes
- *504* Paramètre de commande non implémenté
- *550* Action non effectuée : boîte-aux-lettres non disponible [Ex : boîte aux lettres non trouvée, pas d'accès]
- *551* Utilisateur non local ; essayer <route-directe> (sans relais automatique)
- *552* Action annulée : manque de ressources de stockage
- *553* Action non effectuée : nom de boîte-aux-lettres non autorisée [Ex : erreur de syntaxe dans le nom de boîte]
- *554* Transaction échouée

3.2 - Ouverture et fermeture de liaison

Une fois le connexion au serveur effectuée, une vérification s'impose pour savoir si le client demandeur parle bien au serveur demandé.
Les 2 commandes suivantes sont donc utilisées à l'établissement et à la fermeture de la connexion.

```
HELO <domain> <CRLF>
```

Le domaine *domain* correspond à votre nom de domaine. Si vous n'en avez pas (ce qui est le cas de la plupart des internautes) vous pouvez mettre ce que vous voulez.
En réalité, le domaine importe peu, cette commande est juste là pour tester la relation.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

*Dans le protocole ESMTP (Extended SMTP), il faut envoyer **EHLO***

```
QUIT <CRLF>
```

Comme vous l'aurez deviné, **QUIT** quitte la connexion.

***<CRLF>** correspond à un "Carriage return" et à un "Life Feed" qui signifient respectivement "Retour Chariot" et "Nouvelle Ligne". En quelque sorte, cela correspond à un appui sur la touche **Entrée** de votre clavier.*

Exemple d'ouverture de connexion :

```
220 mwinf2344.orange.fr ESMTP ABO *************************
EHLO skyrunner.home
250-mwinf2344.orange.fr
250-PIPELINING
250-SIZE 10485760
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250 8BITMIME
QUIT
221 Bye
```

Vous noterez qu'il s'agit d'un serveur ESMTP.

3.3 - Emission de courrier

Une transaction SMTP se déroule en 3 étapes.

- La première donne l'identificateur de la transaction.
- La deuxième donne les destinataires.
- La troisième donne le contenu message.

La première étape se fait *via* cette commande

```
MAIL FROM:<adresse_source> <CRLF>
```

Si l'émetteur accepte la commande, il renvoie 250. *adresse_source* est l'adresse utilisée en cas de rapport d'erreur à transmettre (mail non délivré).

La deuxième étape est la liste des destinataires, via cette commande

```
RCPT TO:<dest> <CRLF>
```

Cette commande renvoie aussi 250, en cas de succès. Elle peut être répétée autant de fois que nécessaire, pour les différents destinataires, que ce soient des destinataires en copie ou copie cachée.

La troisième étape est donc la saisie et l'envoi du message via cette commande

```
DATA <CRLF>
```

Après cette commande, le récepteur SMTP renvoi 354.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Toutes les lignes saisies à la suite seront donc dans le message. Pour terminer la saisie du message, il faut taper une ligne avec un point unique.
Si vous voulez envoyer dans votre message une ligne, avec seulement un point, il faut en taper deux.

Après la saisie du message, le récepteur SMTP renvoi 250.

C'est aussi dans cette saisie que s'effectue l'envoi de l'en-tête du message (Date, Subject, From...), tout doit être spécifié manuellement. L'en-tête doit être spécifiée au début, et une ligne vide doit séparer l'en-tête, du corps du message. La structure d'un message, appartient à elle seule à une RFC, la RFC822. Si vous voulez plus d'infos sur cette dernière c'est par là **RFC822** ou par là pour les anglophobes **RFC822**

Voici un exemple basique d'en-tête de message

```
Date: Fry, 29 Dec 06 14:36:00 +0100 GMT
From: "aaa" <aaa@aaa.com>
Subject: Test SMTP
To: "bbb" <bbb@bbb.com>
Cc: "xxx" <xxx@xxx.com>, "yyy" <yyy@yyy.com>
Bcc: "zzz" <zzz@zzz.com>
```

En-tête

- **Date** : date du message (possibilité d'antidater vos mails...). Si aucune date n'est spécifiée, le premier MTA s'en charge.
- **From** : indique l'expéditeur.
- **To** : indique les destinataires.
- **Cc** : indique les destinataires en copie.
- **Bcc** : indique les destinataires en copie cachée.
- **Subject** : sujet du message.

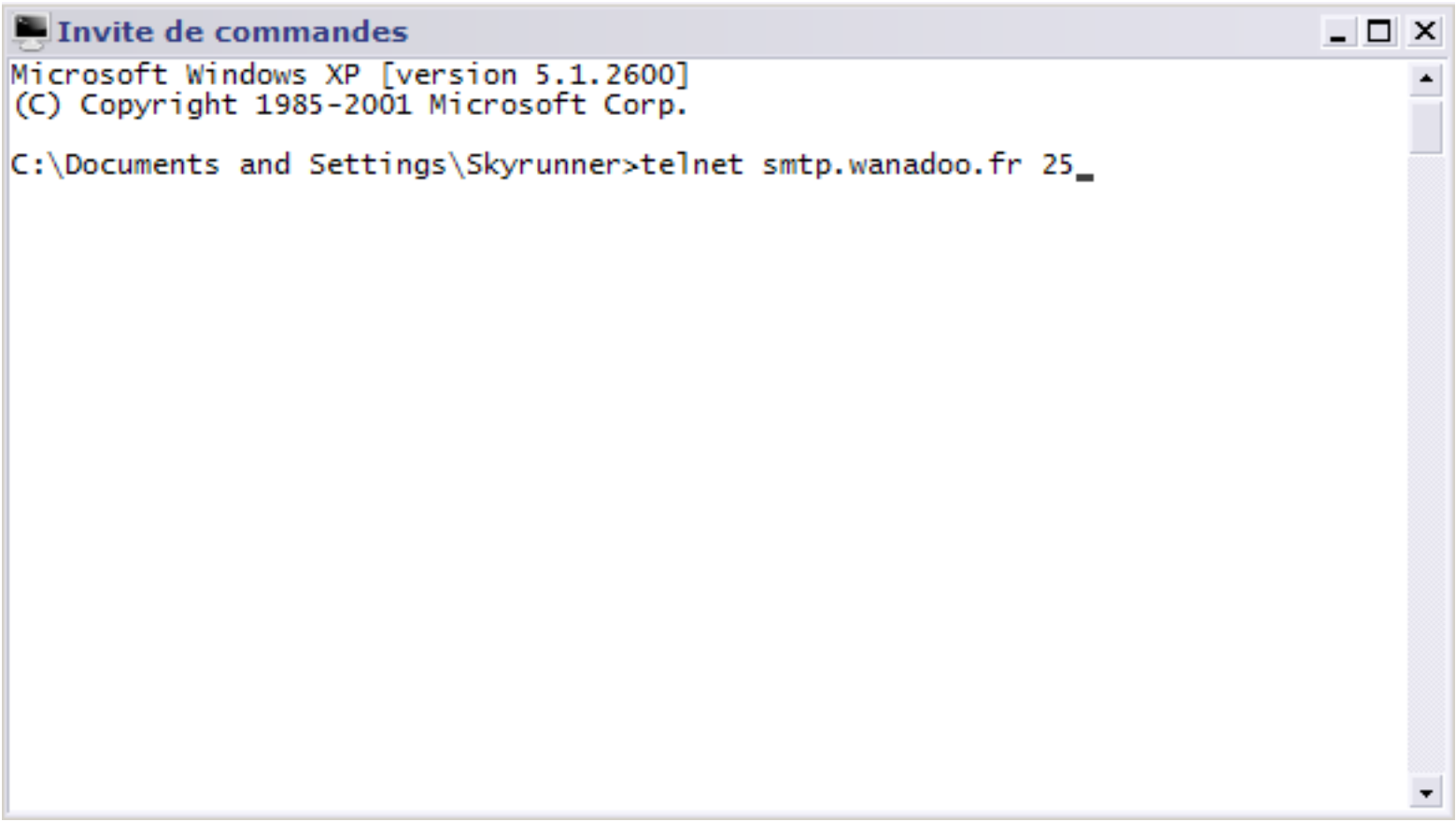
Pour plus d'informations sur l'en-tête des messages, je vous renvoie à la RFC **RFC822** ou **RFC822**.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

4 - Exemple via telnet

Le protocole SMTP à l'instar de HTTP, est un protocole basé sur le modèle client serveur, en mode texte.
Pour envoyer un mail, on peut donc facilement se connecter au serveur SMTP de son FAI, via **telnet**.
Le port de SMTP est le port 25.

Si vous voulez refaire l'exemple ci-dessous, attention en utilisant telnet. En effet, lorsque vous tapez un caractère, il est directement envoyé, donc si vous vous trompez dans une commande, il faut la retaper.

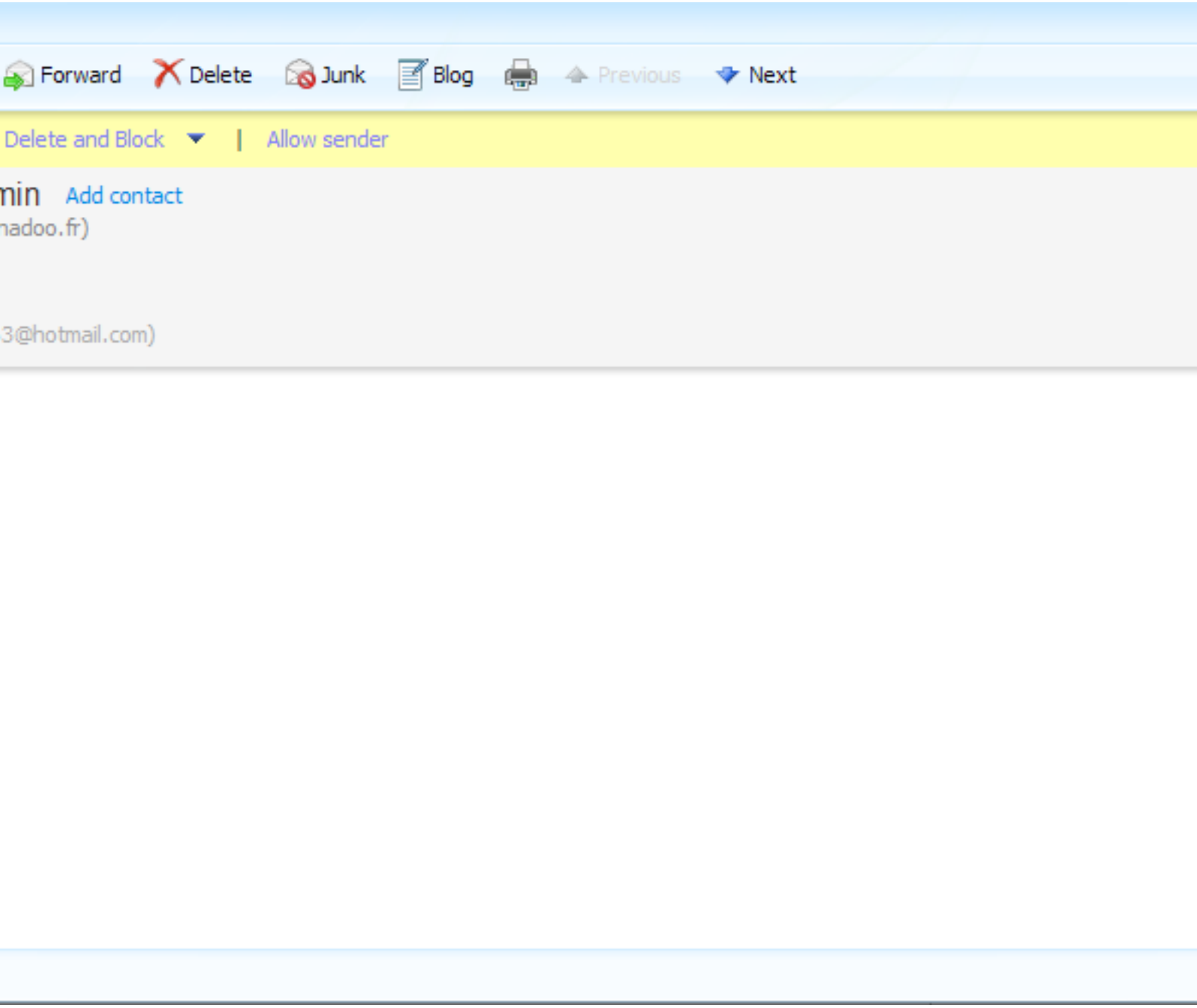


Simple connexion à smtp.wanadoo.fr sur le port 25.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Envoi d'un mail, comme décrit dans la norme de SMTP, avec les bonnes commandes.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Le mail que j'ai reçu quelques secondes plus tard. Tout est bien rempli, y compris l'en tête (juste un petit problème d'heure à cause de l'heure d'hiver).

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

5 - Exemple en C

5.1 - Description

Voici un petit code en C, compilable sous Windows et Linux.
Il permet l'envoi d'un e-mail, en se basant sur un fichier texte passé en argument.

Le fichier texte doit respecter ce format :

```
serveur smtp
domaine
adresse source
adresse(s) destination(s), séparées par des virgules sans espaces
En-tête du message

Message
```

Voici donc un exemple de fichier.

```
smtp.wanadoo.fr
skyrunner.home
xxx@wanadoo.fr
yyy@hotmail.com
From: "xxx" <xxx@wanadoo.fr>
To: "yyy" <yyy@hotmail.com>
Subject: Test smtp

Bonjour,
Test pour le programme C.
```

5.2 - Code source

Voici le projet Code::Blocks : **Code source client SMTP**

client_smtp.h

```
#ifndef CLIENT_SMTP_H
#define CLIENT_SMTP_H

#ifdef WIN32

#include <winsock2.h>

#elif defined (linux)

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h> /* close */
#include <netdb.h> /* gethostbyname */
#define INVALID_SOCKET -1
#define SOCKET_ERROR -1
#define closesocket(s) close(s)
typedef int SOCKET;
typedef struct sockaddr_in SOCKADDR_IN;
typedef struct sockaddr SOCKADDR;
typedef struct in_addr IN_ADDR;

#else

#error not defined for this platform
```


Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

client_smtp.h

```
#endif

#define EHLO        "EHLO"
#define MAIL_FROM   "MAIL FROM:"
#define RCPT_TO     "RCPT TO:"
#define DATA        "DATA"
#define END_DATA    "."
#define QUIT        "QUIT"
#define PORT        25
#define BUF_SIZE    1024
#define CRLF        "\r\n"

static void init(void);
static void end(void);
static void app(const char *filename);
static void read_server(SOCKET sock, char *buffer);
static void write_server(SOCKET sock, char *buffer);
static void send_message(FILE *file, SOCKET sock, char *buffer);
static void get_line(FILE *file, char *buffer);
static void get_and_send_rcpts(SOCKET sock, FILE *file);

#endif /* guard */
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

#include "client_smtp.h"

static void init(void)
{
#ifdef WIN32
    WSADATA wsa;
    int err = WSAStartup(MAKEWORD(2, 2), &wsa);
    if(err < 0)
    {
        puts("WSAStartup failed !");
        exit(EXIT_FAILURE);
    }
#endif
}

static void end(void)
{
#ifdef WIN32
    WSACleanup();
#endif
}

static void app(const char *filename)
{
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    SOCKADDR_IN sin;
    struct hostent *hostinfo;
    char command[BUF_SIZE];
    char buffer[BUF_SIZE];
    FILE *file = NULL;

    file = fopen(filename, "r");

    if(file == NULL)
    {
        perror("fopen()");
        exit(errno);
    }

    if(sock == INVALID_SOCKET)
```


Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

main.c

```
    {
        perror("socket()");
        exit(errno);
    }

    get_line(file, buffer);
    hostinfo = gethostbyname(buffer);
    if (hostinfo == NULL)
    {
        fprintf (stderr, "Unknown host %s.\n", buffer);
        exit(EXIT_FAILURE);
    }

    sin.sin_addr = *(IN_ADDR *) hostinfo->h_addr;
    sin.sin_port = htons(PORT);
    sin.sin_family = AF_INET;

    if(connect(sock,(SOCKADDR *) &sin, sizeof(SOCKADDR)) == SOCKET_ERROR)
    {
        perror("connect()");
        exit(errno);
    }

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    /* send EHLO */
    get_line(file, buffer);
    sprintf(command, "%s %s%s", EHLO, buffer, CRLF);
    write_server(sock, command);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    /* send MAIL FROM */
    get_line(file, buffer);
    sprintf(command, "%s<%s>%s", MAIL_FROM, buffer, CRLF);
    write_server(sock, command);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    /* send RCTP TO */
    get_and_send_rcpts(sock, file);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    /* send DATA */
    sprintf(command, "%s%s", DATA, CRLF);
    write_server(sock, command);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    /* send body of the message */
    send_message(file, sock, buffer);
    fclose(file), file = NULL;

    /* send the mark of the end of the message */
    sprintf(command, "%s%s%s", END_DATA, CRLF, CRLF);
    write_server(sock, command);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);
```


Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

main.c

```
    /* send QUIT */
    sprintf(command, "%s%s", QUIT, CRLF);
    write_server(sock, command);

    /* read response */
    read_server(sock, buffer);
    puts(buffer);

    closesocket(sock);
}

static void read_server(SOCKET sock, char *buffer)
{
    char *p;
    int n = 0;

    if((n = recv(sock, buffer, BUF_SIZE - 1, 0)) < 0)
    {
        perror("recv()");
        exit(errno);
    }

    if(n == 0)
    {
       printf("Connection lost\n");
       exit(EXIT_FAILURE);
    }

    buffer[n] = 0;

    p = strstr(buffer, CRLF);

    if(p != NULL)
    {
        *p = 0;
    }
}

static void write_server(SOCKET sock, char *buffer)
{
    if(send(sock, buffer, strlen(buffer), 0) < 0)
    {
        perror("send()");
        exit(errno);
    }
}

static void get_line(FILE *file, char *buffer)
{
    if(fgets(buffer, BUF_SIZE, file) != NULL)
    {
        char *p;

        p = strchr(buffer, '\n');

        if(p != NULL)
        {
            *p = 0;
        }
        else
        {
            int c;

            while((c = fgetc(file)) != '\n' && c != EOF)
            {
            }
        }
    }
    else if(!feof(file))
    {
```


Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

main.c

```
        puts("Error read file !");
    }
}

static void get_and_send_rcpts(SOCKET sock, FILE *file)
{
    char buffer[BUF_SIZE];
    char *rcpt = NULL;

    if(fgets(buffer, BUF_SIZE, file) != NULL)
    {
        char *p;

        p = strchr(buffer, '\n');

        if(p != NULL)
        {
            *p = 0;
        }
        else
        {
            int c;

            while((c = fgetc(file)) != '\n' && c != EOF)
            {
            }
        }
    }
    else if(!feof(file))
    {
        puts("Error read file !");
    }

    rcpt = strtok(buffer, ",");

    while(rcpt != NULL)
    {
        char command[BUF_SIZE];
        sprintf(command, "%s<%s>%s", RCPT_TO, rcpt, CRLF);
        write_server(sock, command);

        rcpt = strtok(NULL, ",");
    }
}

static void send_message(FILE *file, SOCKET sock, char *buffer)
{
    char message[BUF_SIZE];

    while(fgets(buffer, BUF_SIZE, file) != NULL)
    {
        char *p = strchr(buffer, '\n');
        if(p != NULL)
        {
            *p = 0;
            sprintf(message, "%s%s", buffer, CRLF);
            write_server(sock, message);
        }
        else
        {
            int c;

            write_server(sock, buffer);

            while ((c = fgetc(file)) != '\n' && c != EOF)
            {
                char ch = c;
                write_server(sock, &ch);
            }

            write_server(sock, CRLF);
```

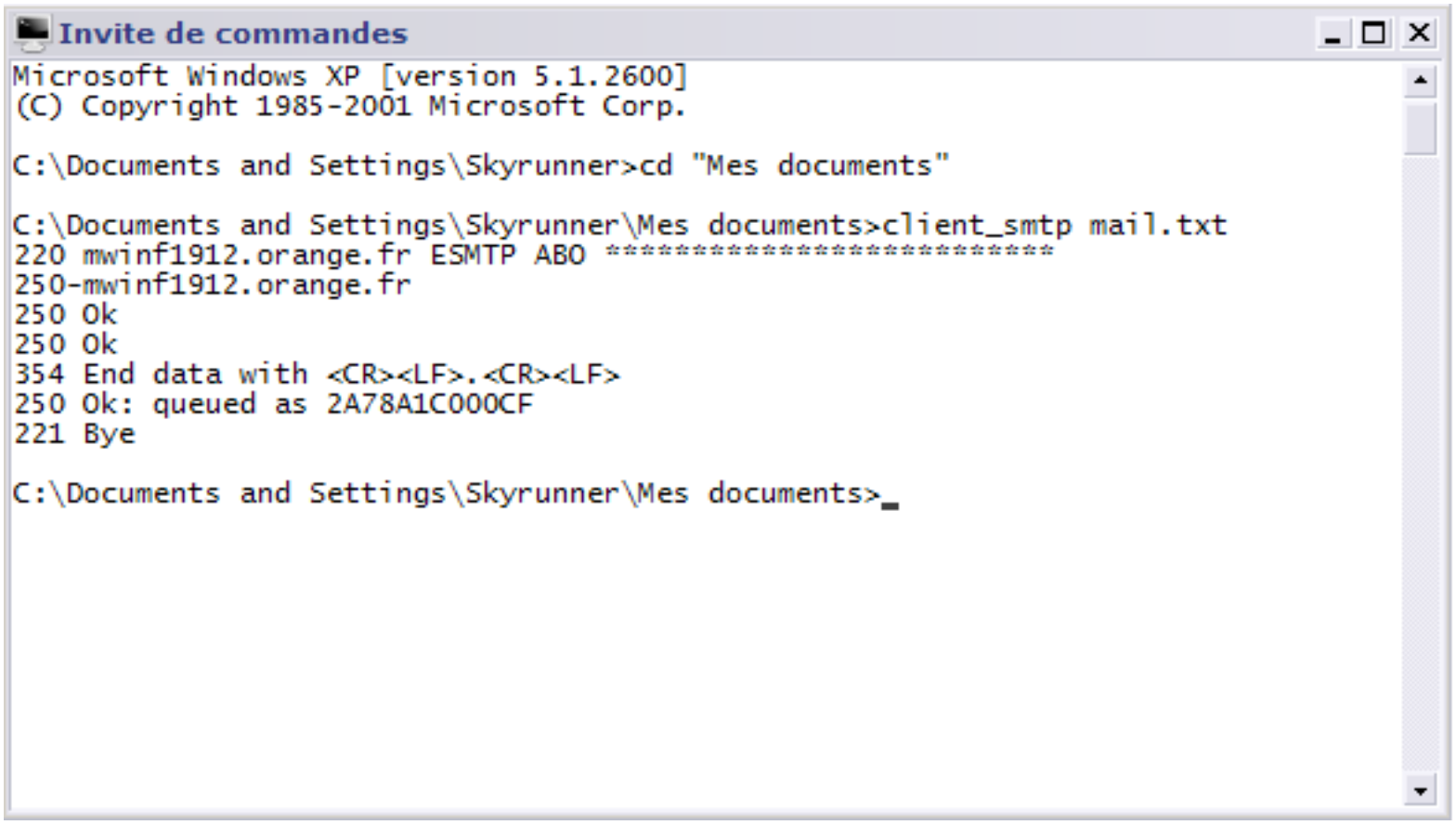

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

main.c

```
        }
    }

    if(!feof(file))
    {
        puts("Error read file !");
        exit(EXIT_FAILURE);
    }
}

int main(int argc, char **argv)
{
    if(argc < 2)
    {
        printf("Usage : %s [filename]\n", argv[0]);
        return EXIT_FAILURE;
    }

    init();

    app(argv[1]);

    end();

    return EXIT_SUCCESS;
}
```

5.3 - Résultat

Et voici la sortie du programme.



Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

6 - Conclusion

Vous venez donc de voir le protocole SMTP, qui est, comme vous avez pu le remarquer, un protocole extrêmement simple.

Vous pouvez donc dès maintenant faire vos propres programmes pour envoyer des e-mails.

Nous verrons prochainement le protocole POP, dans un prochain article.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

7 - Remerciements

Je tiens à remercier Yogui pour sa relecture et ses corrections.

Copyright © 2007 - Benjamin Roux. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.