

CS | VB, VB.NET | ASP.NET, ASP | C, C++ | ColdFusion | PHP | Javascript | Delphi | Flash | Java | Graphisme | Irc | Assembleur | C# | Mobilité | SQL | Foxpro | Python | Windev | Snippets | IT Pros



## SERVEUR PROXY HTTP (WIN32)

Pour plus de détails sur ce code, les commentaires et tout ce dont vous pouvez avoir besoin de savoir pour le faire fonctionner, veuillez consulter cette page : [SERVEUR PROXY HTTP \(WIN32\)](#)

## Source

```

1.  #define WIN32_LEAN_AND_MEAN
2.
3.  #include <windows.h>
4.  #include <winsock2.h>
5.  #include <stdlib.h>
6.  #include <string.h>
7.
8.  #define HTTP_400_BEGIN "HTTP/1.1 200 OK\r\nServer: mProxy/1.0\r\nConnection: clo
9.  #define HTTP_400_END "</pre></blockquote><hr><div align=right>mProxy by <a href=
10.
11. #define HTTP_404_BEGIN "HTTP/1.1 200 OK\r\nServer: mProxy/1.0\r\nConnection: clo
12. #define HTTP_404_END "</pre></blockquote><hr><div align=right>mProxy by <a href=
13.
14. #define HTTP_501_BEGIN "HTTP/1.1 200 OK\r\nServer: mProxy/1.0\r\nConnection: clo
15. #define HTTP_501_END "</pre></blockquote><hr><div align=right>mProxy by <a href=
16.
17. void PutStr(char** r, char* s) {
18.     int x = strlen(s) + 1;
19.     int n = (*r) ? strlen(*r) : 0;
20.     *r = (char*) realloc(*r, x + n);
21.     memcpy(*r + n, s, x);
22. }
23.
24. int main(int argc, char* argv[]) {
25.     WSADATA WSADATA;
26.     SOCKADDR_IN ServerSock;
27.     int ServerSockSize = sizeof(ServerSock);
28.     SOCKET Server = INVALID_SOCKET;
29.     SOCKADDR_IN ClientSock;
30.     int ClientSockSize = sizeof(ClientSock);
31.     SOCKET Client = INVALID_SOCKET;
32.     WSStartup(MAKEWORD(2, 2), &WSADATA);
33.     memset(&ServerSock, 0, ServerSockSize);
34.     ServerSock.sin_family = AF_INET;
35.     ServerSock.sin_port = htons(8080);
36.     ServerSock.sin_addr.s_addr = htonl(INADDR_ANY);
37.     Server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
38.     bind(Server, (SOCKADDR*) &ServerSock, ServerSockSize);
39.     listen(Server, SOMAXCONN);
40.     while (1) {
41.         int i = 0;
42.         int n = 0;
43.         char c = 0;
44.         char* p = NULL;
45.         char* r = NULL;
46.         char* cv = NULL;
47.         int cl = 0;
48.         memset(&ClientSock, 0, ClientSockSize);
49.         Client = accept(Server, (SOCKADDR*) &ClientSock, &ClientSockSize);
50.         while (recv(Client, &c, 1, 0) > 0) {
51.             r = (char*) realloc(r, i + 1);
52.             r[i++] = c;
53.             if (c == '\\r' || c == '\\n') {
54.                 if (++n == 4) break;

```

```

55.     } else n = 0;
56. }
57. r = (char*) realloc(r, i + 1);
58. r[i] = '\0';
59. p = strstr(r, "Content-Length: ");
60. if (p) {
61.     int i = 0;
62.     p += 16;
63.     while (*p != '\r' && *p != '\n') {
64.         cv = (char*) realloc(cv, i + 1);
65.         cv[i++] = *p;
66.         p++;
67.     }
68.     cl = atoi(cv);
69.     free(cv);
70.     cv = (char*) malloc(cl + 1);
71.     recv(Client, cv, cl, 0);
72.     cv[cl] = '\0';
73. }
74. if (r[0] != 'G' || r[1] != 'E' || r[2] != 'T' || r[3] != ' ') {
75.     send(Client, HTTP_501_BEGIN, strlen(HTTP_501_BEGIN), 0);
76.     send(Client, r, i, 0);
77.     if (cv) send(Client, cv, cl, 0);
78.     send(Client, HTTP_501_END, strlen(HTTP_501_END), 0);
79. } else {
80.     char* p = strstr(r, " HTTP/");
81.     if (p[6] != '1' || p[7] != '.' || p[8] != '1') {
82.         send(Client, HTTP_400_BEGIN, strlen(HTTP_400_BEGIN), 0);
83.         send(Client, r, i, 0);
84.         if (cv) send(Client, cv, cl, 0);
85.         send(Client, HTTP_400_END, strlen(HTTP_400_END), 0);
86.     } else {
87.         char* p = r;
88.         char* host = NULL;
89.         char* port = NULL;
90.         char* location = NULL;
91.         HOSTENT* mProxyAddr = NULL;
92.         while (p[0] != 'H' || p[1] != 'o' || p[2] != 's' || p[3] != 't' || p[4]
93.             p += 6;
94.             host = p;
95.             while (*p != ':' && *p != '\r' && *p != '\n') p++;
96.             if (*p == ':') {
97.                 *p = '\0';
98.                 port = p + 1;
99.                 while (*p != '\r' && *p != '\n') p++;
100.            } else port = "80";
101.            *p = '\0';
102.            p = strstr(r, "://");
103.            location = (p) ? p + 3 : r + 4;
104.            while (*location != '/') location++;
105.            *(strstr(location, " HTTP/")) = '\0';
106.            mProxyAddr = gethostbyname(host);
107.            if (mProxyAddr) {
108.                SOCKADDR_IN mProxySock;
109.                int mProxySockSize = sizeof(mProxySock);
110.                SOCKET mProxy = INVALID_SOCKET;
111.                memset(&mProxySock, 0, mProxySockSize);
112.                memcpy(&mProxySock.sin_addr.s_addr, mProxyAddr->h_addr, mProxyAddr->h_
113.                mProxySock.sin_port = htons(atoi(port));
114.                mProxySock.sin_family = AF_INET;
115.                mProxy = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
116.                if (connect(mProxy, (SOCKADDR*) &mProxySock, mProxySockSize)) {
117.                    send(Client, HTTP_404_BEGIN, strlen(HTTP_404_BEGIN), 0);
118.                    send(Client, host, strlen(host), 0);
119.                    send(Client, ":", 1, 0);
120.                    send(Client, port, strlen(port), 0);

```

```

121.         send(Client, HTTP_404_END, strlen(HTTP_404_END), 0);
122.     } else {
123.         char* r = NULL;
124.         PutStr(&r, "GET ");
125.         PutStr(&r, location);
126.         PutStr(&r, " HTTP/1.1");
127.         PutStr(&r, "\r\n");
128.         PutStr(&r, "User-Agent: mProxy/1.0");
129.         PutStr(&r, "\r\n");
130.         PutStr(&r, "Host: ");
131.         PutStr(&r, host);
132.         PutStr(&r, ":");
133.         PutStr(&r, port);
134.         PutStr(&r, "\r\n");
135.         PutStr(&r, "Connection: close");
136.         PutStr(&r, "\r\n");
137.         PutStr(&r, "\r\n");
138.         send(mProxy, r, strlen(r), 0);
139.         free(r);
140.         while (recv(mProxy, &c, 1, 0) > 0) send(Client, &c, 1, 0);
141.         shutdown(mProxy, SD_BOTH);
142.         closesocket(mProxy);
143.     }
144. } else {
145.     send(Client, HTTP_404_BEGIN, strlen(HTTP_404_BEGIN), 0);
146.     send(Client, host, strlen(host), 0);
147.     send(Client, ":", 1, 0);
148.     send(Client, port, strlen(port), 0);
149.     send(Client, HTTP_404_END, strlen(HTTP_404_END), 0);
150. }
151. }
152. }
153. shutdown(Client, SD_BOTH);
154. closesocket(Client);
155. free(r);
156. }
157. }

```



Dev réalisé par **Nicolas SOREL (Nix)** avec l'aide de : **Cyril DURAND, Bidou, Nurgle**  
 Base de données maintenue par **Christian**, Design réalisé par **Guillaume**, Mascotte réalisée par **N. SOREL (Nix)** & **KDO-Conception**  
 CPPFrance.com© Toute reproduction même partielle est interdite sauf accord écrit du Webmaster  
 CodeS-SourceS.com© est une marque déposée tous droits réservés  
 Hébergement de **Serveur dédié** - **Téléphonie VOIP** - **Comparer les prix** - **Envoyer des ecards gratuit** - **Icones**  
 Temps d'exécution de la page : 0,0156-3sec



Certaines images présentes sur le site (notamment certains avatars) sont issues des collections **IconShock**, donc si vous souhaitez utiliser ces icons vous devez les acheter, ne les copiez pas et ne utilisez pas dans vos sites et applications sans les avoir commandé.