Tutoriel programmation C sous Code Blocks

Installation de code Blocks sous windows

Aller sur :
www.codeblocks.org/downloads
Puis clicker sur « Download the binary release »
Puis télécharger **codeblocks-17.12mingw-setup.exe**
Cette version n'est peut-être plus disponible. Choisissez la dernière version « mingw » car elle permet d'être au plus près des outils gnu pour Linux : compilateur gcc et debugger gdb.

Exécuter ce programme, suivez les instructions, et laissez les options par défaut.

Notre premier programme

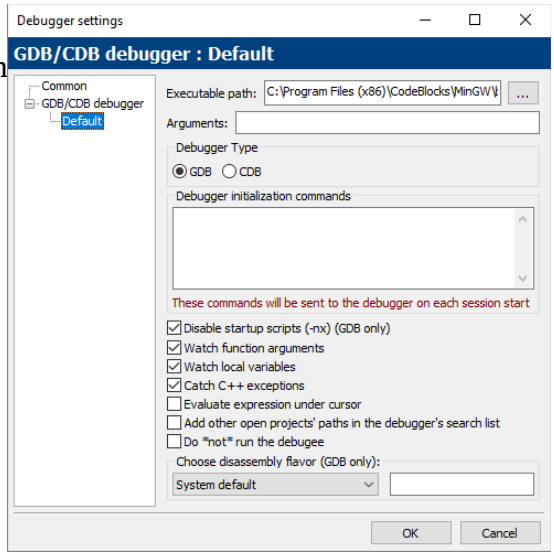
Lancez codeBlocks
clickez sur « **File → New Project** »
puis double-cliqez sur « **Console Application** »
puis « **Next** »
puis choisir « **C** » puis « **Next** »
puis choisir un **nom** à votre projet et choisir le **répertoire** où il sera sauvegarder. Puis cliquer sur « **Next** »
Choisissez « **GNU GCC Compiler** » pour le champ « compiler ».
Laissez les cases « create Debug » et « create Release » **cochées**.
Cliquer sur « **Finish** »

Dans l'onglet « Projects » (sur la gauche de l'écran) double-cliquer sur « **sources** », puis sur « **main.c** ».

Pour exécuter l'exemple, juste cliquer sur le triangle vert 

Débogage

Pour Débugger avec code Blocks, il faut préciser que l'on utilise GDB :
Sur la barre de menu, cliquer sur « **settings → Debugger** » puis double-cliquer sur **GDB/CDB** puis sur **default**. Dans le champs « Executable Path » parcourez vos dossiers afin de trouver le fichier « **gdb32.exe** » qui se trouve très probablement dans :
« C:\Program Files (x86)\CodeBlocks\MinGW\bin\gdb32.exe »



Puis cliquer sur « **GDB** » et enfin « OK »

Pour mettre un point d'arrêt sur une ligne du code

source: faites click droit sur le numéro de la ligne puis « add breakpoint ».
Pour exécuter en mode déboggage, cliquer sur le triangle rouge : 
A côté vous trouverez les boutons pour :

- debugger jusqu'à une certaine ligne
- exécuter une instruction
- entrer dans le code d'une fonction
- …

Pour voir les variables, cliquer sur « **Debug → Debugging windows → Watches** »

Tutoriel Programmation réseau avec Code Blocks sous windows

Par rapport à la programmation sous Linux, il faut ajouter quelques fonctions Windows supplémentaires pour initialiser l'API socket : WSAStartup( ) et WSACleanup ( ).
Vous trouverez ci-dessous un programme exemple d'utilisation de ces 2 fonctions. Ce programme est un serveur d'écho. Il renvoie au client ce que ce dernier lui envoie.

De plus, il faut ajouter la bibliothèque libws2_32.a.
Pour cela, aller cliquer « **Project → Build Option... → Linker settings** » puis cliquer sur « **Debug** » et « **Add** » puis parcourir le répertoire d'installation de codeBlocks pour ajouter libws2_32.a
Ce fichier se trouvera probablement dans :
C:\Program Files (x86)\CodeBlocks\MinGW\lib\libws2_32.a
Cliquer maintenant sur« **Release** » et refaire de même.

Pour tester le programme, exécuter le, puis lancer **teraterm**, et sélectionner
service : « **telnet** »
host : « **127.0.0.1** »
port : « **3000** »
puis entrer des mots sur teraterm qui s'afficheront côté serveur et côté client.
Voici un exemple de code serveur :

```
// @TITLE : Serveur d'echo sur le port passé en argument
// @BRIEF : Répète ce qu'il reçoit et l'affiche. La connexion s'arrete après réception de bye.
// @AUTHOR : Ph Lefebvre - ENSI de Caen

#include <errno.h>                // biblio des erreurs
#include <winsock2.h>              // bibliothèque socket
#include <stdio.h>                // biblio pour les E/S.
#include <stdlib.h>
#include <unistd.h>
#define PAQUET_LEN 64

int main( int argc, char **argv) {
       int er, lgr;                    // gestion des erreurs
       int s;                          // socket d'ecoute du port
       int sd;                         // socket de dialogue avec le client.
       char paquet[PAQUET_LEN];              // paquet recu / envoye

       struct sockaddr_in adTo, adFrom;    // adresse au format internet

       int ladd=sizeof (struct sockaddr_in);  // longueur de l'adresse de reception
       int port = 3000 ;                                          // port de connexion

       WSADATA WSAData;
    WSAStartup(MAKEWORD(2,0), &WSAData); // préparation de la bibliothèque socket

       /* On ouvre la socket  Internet en mode connecte. 6 est le numéro de TCP , cf. le
fichier /etc/protocols*/
       if ((s = socket(AF_INET, SOCK_STREAM, 6)) < 0) {
              perror("\nERREUR socket\n"); exit(-1);
       }

       adTo.sin_family = AF_INET;           // adresses de type internet
       adTo.sin_port = htons (port);        // numero de port du serveur
       adTo.sin_addr.s_addr = INADDR_ANY;   // accepte les connexions de n'importe quelle
interface ethernet, wifi...

       /* On attache la socket au port d'écoute */
       er = bind (s, (struct sockaddr *) &adTo, sizeof (struct sockaddr_in));
       if (er < 0) {
              perror ("bind : "); exit(-1);
       }
```

```
  /* On fixe le nombre maximum de clients simultanes. Ici 1 seul*/
       er = listen (s, 1);
       if (er < 0) {
              perror ("listen : "); exit(-1);
       }

    sd = accept( s, (struct sockaddr *) &adFrom, &ladd);             /* attente d'une connexion. La
fonction accept est boquante et crée une socket de dialogue si un client se connecte.*/
    printf ("1 client !\n");
    do {
              lgr = recv( sd, paquet, PAQUET_LEN, 0 );     // on lit au maximum 64 octets venant
du client.
              if (lgr == 0 ) {printf("Deconnexion par le client !\n\n") ; exit(0) ;}
              if (lgr < 0 ) {perror ("pb recv") ; exit(-1) ;}
              paquet[lgr]= '\0';                                          // le paquet reçu n'est
pas une chaîne de caractères => ajout de "\0" à la fin
              printf ("paquet de longueur %d reçu : %s\n",lgr,paquet);
              send (sd, paquet, lgr, 0);          // réémission du paquet recu.
       }
       while ( (strncmp ((char *)paquet, "bye", 3)));      // on compare les 3 premiers octets
reçus a "bye"
    printf (" Mon client s'est deconnecte.\n");
       close (sd);                          // fermeture de la socket de dialogue
    close (s);                                             // fermeture de la socket d'écoute
    WSACleanup();              // on libère les ressources windows utilisées par les sokets
    return(0) ;
}
```