

組み込み SQL の始め方 (C 言語編)

組み込み SQL の始め方 (C 言語編)

初版

日本アイ・ビー・エム株式会社
ソフトウェア事業部
ソフトウェア・テクニカルサポート
DM 技術部 開発技術支援グループ

目次

1. はじめに	5
2. 組み込み SQL アプリケーションを作成するプロセス.....	6
3. ステップ 1	7
4. ステップ 2	9
5. ステップ 3	12
6. ステップ 4	13
7. makefile にまとめる.....	14
8. まとめ	15

更新履歴

初版 : 2002/07/29

この資料の対象とする読者

この資料の対象者は、DB2 UDB 上で C 言語を使って

- ・ 組み込み SQL アプリケーションをこれから学ぼうとする人
- ・ 組み込み SQL アプリケーションのコンパイル方法がよくわからない人

を想定しています。

なお、コマンド・ウィンドウや、Microsoft VisualC++を経験している事を前提としています。

この資料は、次の開発環境を想定して説明しています。

- ・ Microsoft Windows 2000 Professional SP2
- ・ Microsoft Visual C++6.0 SP5
- ・ DB2 UDB EE V7.2 (Windows 版)

(*) 記載の会社名と製品名はそれぞれ各社の登録商標または商標です。

1. はじめに

C 言語を使った組み込み SQL を学ぶ時は、DB2 の導入先の `sqllib\samples\c` 配下にあるプログラミング例を参考すると大変便利です。

Windows の場合、Microsoft VisualC++などの環境があれば、簡単にサンプルプログラムをビルドして実行する事ができます。

例えば、サンプルプログラム `openftch.sql` をビルドする時は、コマンド・ウィンドウ上で

```
D:\sql\lib\samples\c> nmake openftch
```

を実行します。

ビルドが成功すると `openftch.exe` プログラムができあがります。そして、動作を確認する事ができます。

動作を確認する事はできましたが、`nmake` を使ったビルド方法に不慣れな場合、いざ、自分のプログラムをビルドする場合、開発環境の構築に戸惑いを感じる事も多いと思います。

その原因には次の2つが考えられるでしょう。

`nmake` を使ったビルドは、`makefile` に、ビルド方法が記述している為、Microsoft VisualC++などの IDE の環境に慣れてしまっている方は、`makefile` を読む事に慣れていない。

この `makefile` 内で、さらに `embprep` というバッチファイルも使用しているので、組み込み SQL アプリケーションを作成するプロセスを理解していないとさらに読むのが難しく感じる。

そこで、本資料では、2つの原因を解消する為、`makefile` で行っている事を分解し、その内容と組み込み SQL アプリケーションを作成するプロセスを対応付けて説明していきます。

この内容を理解できれば、自分で開発環境を整える事ができるはずです。

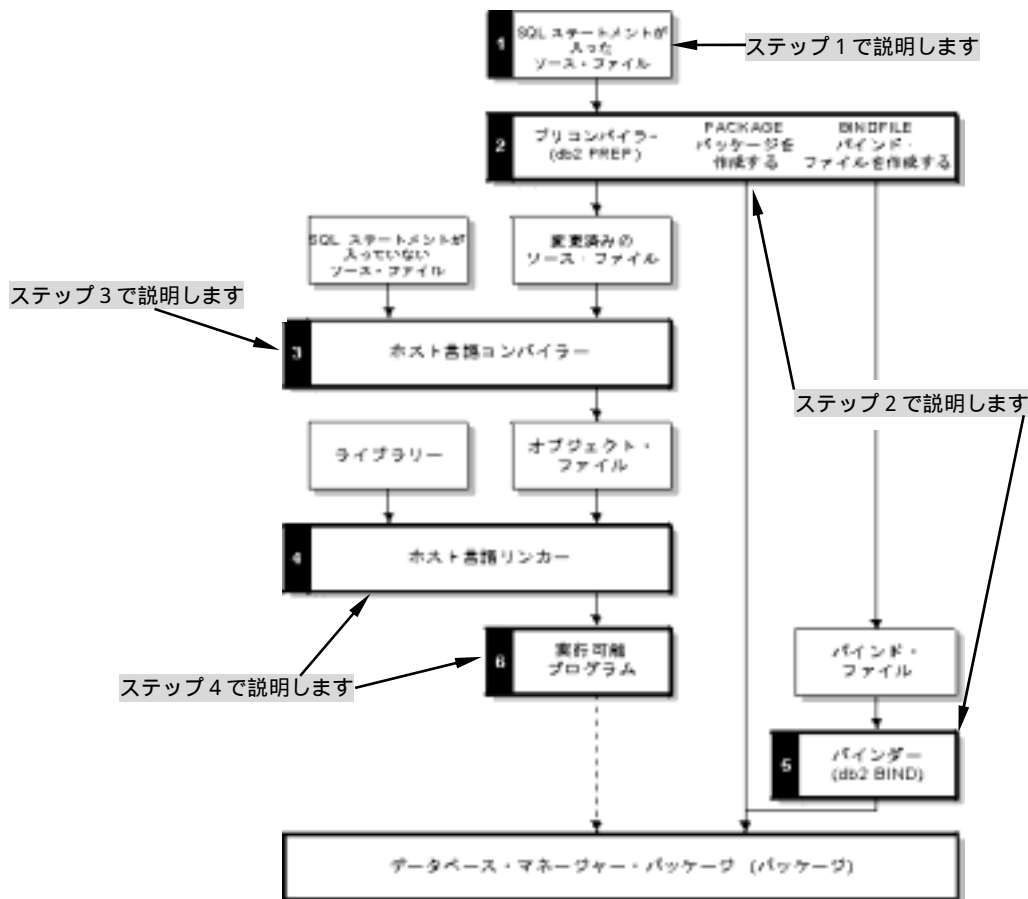
この資料では、

- ・組み込み SQL アプリケーションを作成するプロセスで行っている事を理解する事
- ・組み込み SQL アプリケーションの開発環境を整える事ができるようになる事を目的としています。

2. 組み込み SQL アプリケーションを作成するプロセス

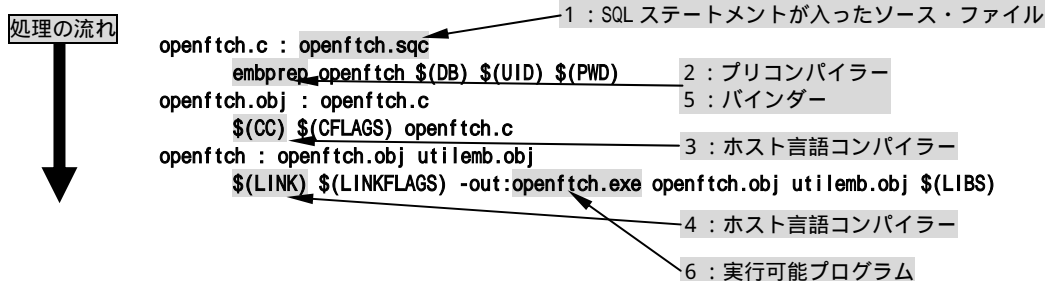
組み込み SQL アプリケーションを作成するプロセスは次の通りです。

(「アプリケーション開発の手引き」から抜粋)



このプロセスと makefile を対応させます。

(openftch の場合。sqllib¥samples¥c¥makefile から 558 行目付近を抜粋)



上の図からわかる通り、プリコンパイルとバインドには、embprep というバッチファイルを使っています。この資料では、embprep の内容も見ていきます。

3. ステップ 1

『 1 . SQL ステートメントが入ったソースファイル』に対応します。

この資料で使用するソース (ファイル名 : myopnfch.sqc) は、openfch.sqc サンプルソースからユーティリティ関数を使用している部分 (EMB_SQL_CHECK などのマクロ) を省いて作成しました。

myopnfch.sqc の内容

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
EXEC SQL INCLUDE SQLCA;

int main(int argc, char *argv[]) {

    EXEC SQL BEGIN DECLARE SECTION;
    char  pname[10];
    short dept;
    char  userid[9];
    char  passwd[19];
    EXEC SQL END DECLARE SECTION;

    if (argc == 1) {
        EXEC SQL CONNECT TO sample;      接続
    }
    else if (argc == 3) {
        strcpy (userid, argv[1]);
        strcpy (passwd, argv[2]);
        EXEC SQL CONNECT TO sample USER :userid USING :passwd;
    }
    else {
        printf ("%nUSAGE: MyOpnFch [userid passwd]%n%n");
        return 1;
    } /* endif */

    EXEC SQL DECLARE c1 CURSOR FOR
        SELECT name, dept FROM staff WHERE job='Mgr'
        FOR UPDATE OF job;              カーソル宣言

    EXEC SQL OPEN c1;                   カーソルオープン

    do {
        EXEC SQL FETCH c1 INTO :pname, :dept;   データの取り出し
        if (SQLCODE != 0) break;

        if (dept > 40) {
            printf ("%n-10.10s in dept. %2d will be demoted to Clerk%n", pname, dept );
            EXEC SQL UPDATE staff SET job = 'Clerk' WHERE CURRENT OF c1;   カレント行の UPDATE
        } else {
            printf ("%n-10.10s in dept. %2d will be DELETED!%n", pname, dept);
            EXEC SQL DELETE FROM staff WHERE CURRENT OF c1;   カレント行を DELETE
        } /* endif */
    } while ( 1 );
}
```

組み込み SQL の始め方 (C 言語編)

```
EXEC SQL CLOSE c1;   カーソルクローズ

EXEC SQL ROLLBACK;   テストなので ROLLBACK
printf( "%n\n second thought -- changes rolled back.%n" );

EXEC SQL CONNECT RESET;   切断
return 0;
}
/* end of program : MyOpnFch.SQC */
```

作業の前に・・・

これからの作業は、すべて「コマンド・ウィンドウ(DB2 CLP)」を利用します。



コマンド・ウィンドウを起動後、myopnfch.sqc のある場所に移動します。
myopnfch.sqc が D:\MyOpnFch フォルダに配置されている場合の例です。

例)

```
D:\SQLLIB\BIN> cd \MyOpnFch
```

4. ステップ 2

『2 . プリコンパイラ (プリコンパイル)』と『5 . バインダー (バインド)』に対応します。

ここでは、3つの作業を行います。

myopnfch.sqc ファイルから C 言語で書かれたソース (myopnfch.c) を作成する。

myopnfch.sqc ファイルからバインド・ファイル (myopnfch.bnd) を作成する。

バインド・ファイルからパッケージを作成する。

パッケージは、そのソースファイル内の SQL 文を実行するために必要な情報を含んでいるので、組み込み SQL アプリケーション実行時に必要になります。

バインド・ファイルにはパッケージ作成に必要な情報が含まれています。

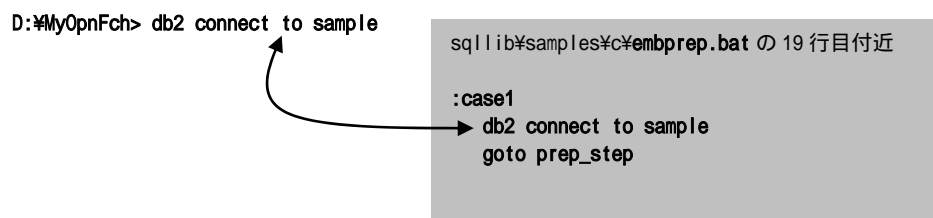
この作業は、プリコンパイル、この作業はバインドで行います。

これらの作業に、embprep バッチファイルを使っていたことは先に触れました。ステップ 2 では、embprep の内容と比較します。

プリコンパイルを行うには、まず、データベースに接続する必要があります。

embprep バッチファイル内では、19 行目付近に相当します。

```
D:\MyOpnFch> db2 connect to sample
```

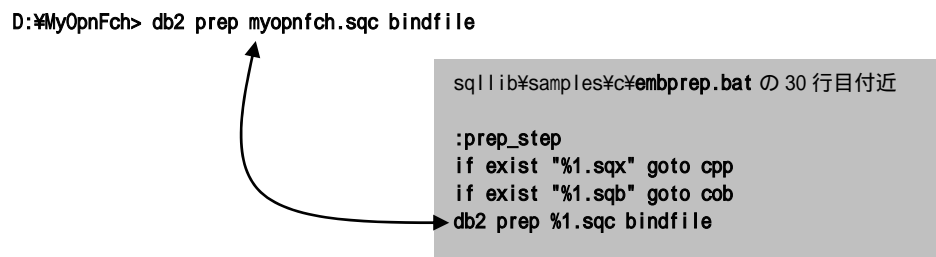


```
sql lib %samples%c %embprep.bat の 19 行目付近
:case1
db2 connect to sample
goto prep_step
```

次はプリコンパイルです。プリコンパイルは prep コマンドを使用します。

embprep バッチファイル内の 30 行目付近に相当します。

```
D:\MyOpnFch> db2 prep myopnfch.sqc bindfile
```



```
sql lib %samples%c %embprep.bat の 30 行目付近
:prep_step
if exist "%1.sqx" goto cpp
if exist "%1.sqb" goto cob
db2 prep %1.sqc bindfile
```

組み込み SQL の始め方 (C 言語編)

myopnfch.c (C 言語のソースファイル) と myopnfch.bnd (バインド・ファイル) が出来ている事を確認してください。

```
D:\MyOpnFch> dir myopnfch.*  
  
D:\MyOpenFtch のディレクトリ  
  
2002/07/01 19:24          4,741 myopnfch.bnd ← バインド・ファイル  
2002/07/01 19:24          6,347 myopnfch.c   ← ソース・ファイル  
2002/07/01 17:14          1,482 myopnfch.sqc
```

myopnfch.c を開くと、EXEC SQL の部分がすべてコメントになり C 言語のソースになっていることがわかります。

myopnfch.c の一部分から

```
/* ← コメント  
EXEC SQL CONNECT TO sample USER :userid USING :passwd;  
*/  
  
{  
#line 25 "myopnfch.sqc" ← C 言語に変換されている  
    sqlastrt(sqla_program_id, &sqla_rtinfo, &sqlca);  
#line 25 "myopnfch.sqc"  
    sqlaaloc(2,3,2,0L);  
    {  
        struct sqla_setd_list sql_setdlist[3];  
#line 25 "myopnfch.sqc"  
        sql_setdlist[0].sqltype = 460; sql_setdlist[0].sqlllen = 7;
```

次は、myopnfch.bnd ファイルを使ってバインド処理です。

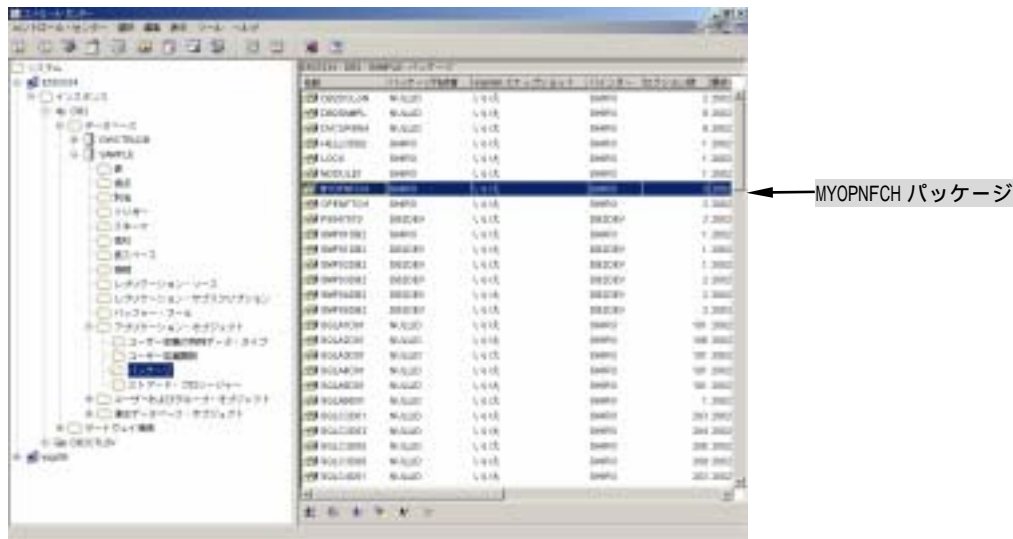
embprep バッチファイル内の 41 行目に相当します。

```
D:\MyOpnFch> db2 bind myopnfch.bnd  
  
sql lib\samples\c\embprep.bat の 41 行目付近  
:bind_step  
db2 bind %1.bnd  
db2 connect reset
```

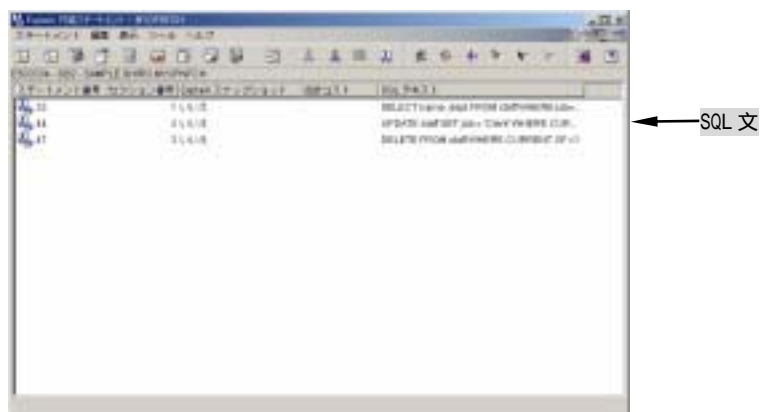
バインド処理が完了するとパッケージできあがります。

コントロールセンターから確認する事ができます。

組み込み SQL の始め方 (C 言語編)



また、MYOPNFCH を選択して「Explain 可能ステートメントを表示」を選択すると myopnfch.sqc での SQL 文を確認する事ができます。(静的 SQL の場合)



プリコンパイルの処理が終わったので接続を解除します。
embprep バッチファイル内の 42 行目に相当します。

```
D:\MyOpnFch > db2 connect reset
```

sql lib%samples%c%embprep.bat の 42 行目付近

```
:bind_step  
db2 bind %1.bnd  
db2 connect reset
```

5. ステップ 3

『3 . ホスト言語コンパイラ』に対応します。

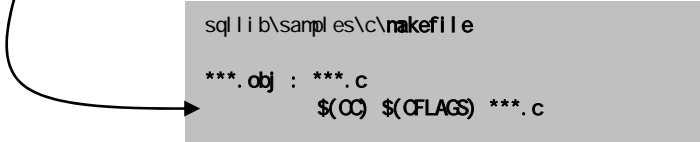
myopnfch.c ができあがったので、ここからは C 言語でアプリケーションを作成する時の手順と同じです。

myopnfch.c をコンパイルします。

makefile の \$(CC) \$(CFLAGS) *.c の部分に相当します。

インクルードに、%DB2PATH%\include を含ませている事に注意してください。%DB2PATH%は DB2 をインストールしたディレクトリになります。

```
D:\MyOpnFch> cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 -I%DB2PATH%\include myopnfch.c
```



```
sql lib\samples\c\makefile では  
***.obj : ***.c  
        $(CC) $(CFLAGS) ***.c
```

コンパイルが完了すると myopnfch.obj (オブジェクト・ファイル) ができあがります。

```
D:\MyOpnFch> dir myopnfch.*
```

```
D:\MyOpenFch のディレクトリ
```

2002/07/01 19:24	4,741 myopnfch.bnd	
2002/07/01 19:24	6,347 myopnfch.c	
2002/07/01 19:24	4,382 myopnfch.obj	← オブジェクト・ファイル
2002/07/01 17:14	1,482 myopnfch.sqc	

6. ステップ 4

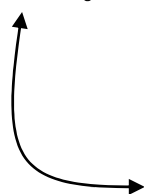
『4. ホスト言語リンカー』と『6. 実行可能プログラム』に対応します。

オブジェクト(myopnfch.obj)が作成後は、リンクです。

makefile の `$(LINK) $(LINKFLAGS) -out:***.exe ***.obj $(LIBS)` の部分に相当します。

db2api.lib もリンクしています。

```
D:\MyOpnFch> link -debug:full -debugtype:cv -out:myopnfch.exe myopnfch.obj %DB2PATH%\lib\db2api.lib
```



```
sql lib\samples\c\makefile では  
***.obj : ***.c  
    $(CC) $(CFLAGS) ***.c  
*** : ***.obj  
    $(LINK) $(LINKFLAGS) -out:***.exe ***.obj $(LIBS)
```

以上のステップで、実行ファイル myopnfch.exe が完成しました。

7. makefile にまとめる

ここで、ステップ 1 からステップ 4 までの入力をもう一度、掲載します。

```
D:\MyOpnFch> db2 connect to sample
D:\MyOpnFch> db2 prep myopnfch.sqc bindfile
D:\MyOpnFch> db2 bind myopnfch.bnd
D:\MyOpnFch> db2 connect reset
D:\MyOpnFch> cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 -I"%DB2PATH%\include" myopnfch.c
D:\MyOpnFch> link -debug:full -debugtype:cv -out:myopnfch.exe myopnfch.obj %DB2PATH%\lib\db2api.lib
```

DB2 に接続して、embprep を使った作業。
プリコンパイルとバインド処理

通常の C 言語での開発作業
コンパイルとリンク。

毎回、これらをコマンドラインから入力するのは大変なので、これらを makefile にまとめてみます。

ファイル名は myopnfch.mk ファイルです。

myopnfch.mk の内容

```
DB=sample
UID=
PWD=

CC=cl
LINK=link
CFLAGS=-Zi -Od -c -W2 -D_X86_=1 -DWIN32 -I"%DB2PATH%\include"
LINKFLAGS=-debug:full -debugtype:cv
LIBS="%DB2PATH%\lib\db2api.lib"

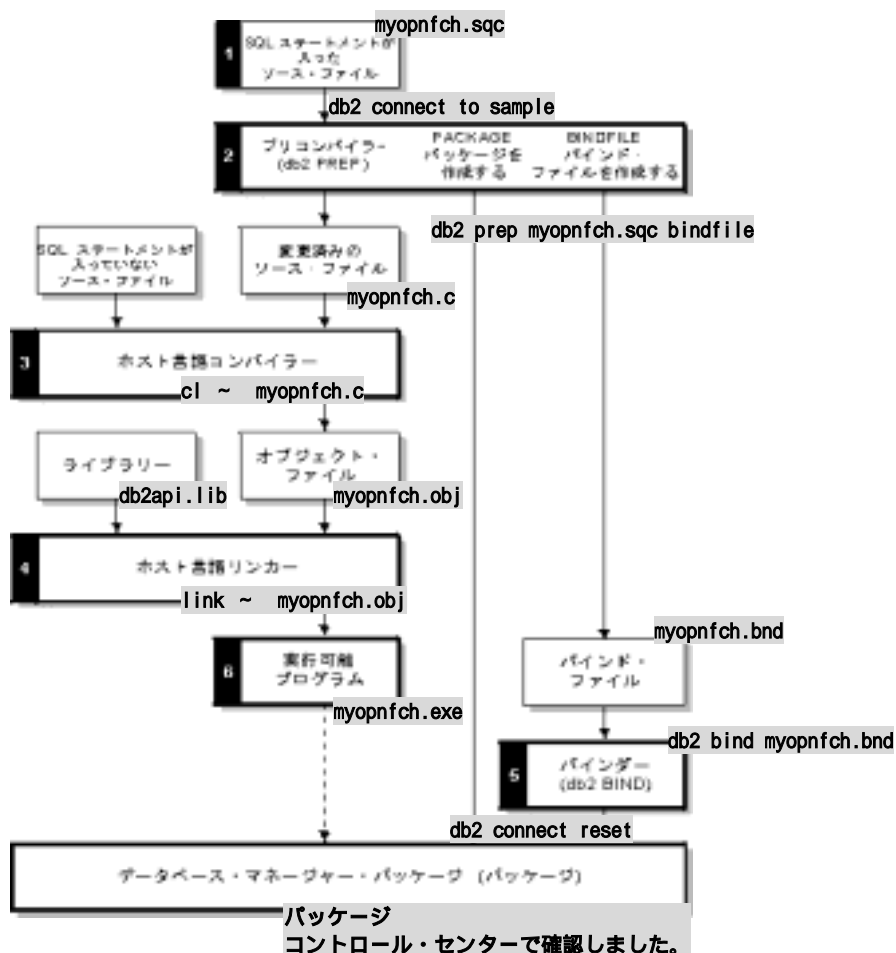
myopnfch.c : myopnfch.sqc
    %DB2PATH%\samples\c\embprep myopnfch $(DB) $(UID) $(PWD)
myopnfch.obj : myopnfch.c
    $(CC) $(CFLAGS) myopnfch.c
myopnfch : myopnfch.obj
    $(LINK) $(LINKFLAGS) -out:myopnfch.exe myopnfch.obj $(LIBS)
```

この makefile を使ってビルドする場合は、次のように行います。

```
D:\MyOpnFch> nmake -f myopnfch.mk myopnfch
```

8. まとめ

これまで説明した内容と、組み込み SQL アプリケーションを作成するプロセスを合わせると次のようになります。



この図から、DB2 に接続して行う作業（プリコンパイルとバインド）と、通常の C 言語での開発作業（コンパイルとリンク）が区別して理解できると思います。

DB2 にはプリコンパイル時に使用できる embprep ファイル、ユーティリティに使用できる utilemb が付属しているので、これらをうまく使って、効率よく開発を行いましょう。

詳しい説明は、「アプリケーション構築の手引き」等を参考にしてください。

<http://www-6.ibm.com/jp/software/data/developer/library/manual/db2online/index.htm>