JAPLA 研究会資料 2011/2/26

J と DLL について
－小さな解説とコメント－

西川　利男

J から DLL を呼んだり、利用することができる。最近では、須田祐司氏の「J 言語事典」の 14 章として、Pascal を用いた例が紹介されている。　DLL は Windows のいろいろなアプリケーションで非常に広く使われている手法である。筆者も以前から興味を持っていたが、J の Lab を元にした小さな解説と個人的なコメントを報告する。

DLL(Dynamic Link Library)とは、メインのプログラムから呼ばれる一種のサブプログラムであるが、いろいろなプログラムから実行時に動的に呼ばれることを目的として、汎用的に作られたプログラムファイルである。ふつう、ファイルの拡張子として dll となっている。Windows では Shared Library File とも呼ばれる。

アプリケーションなど、大きなプログラムではメインプログラムからいろいろなサブプログラムを呼ぶようになっているものも多い。そのためのサブプログラムが複数のモジュール・ファイルに分かれていても、ひとつのプログラミング言語だけで出来ているときはそれほどの分かりにくさはない。ところが、これらが異なった言語で作られている場合には、両方の言語について精通することはもちろんであるが、とくに引き渡すデータの型、数などの整合性をあわせることが肝要になる。

J の DLL

J の DLL に関しては User Manual の“DLL and Memory Management”にその記載がある。

また、J の Lab として

① DLL: Writing and Using a DLL……J で作った簡単な DLL のやり取り

② DLL: Using System DLLs (file examples) … Win32API を使ったファイル処理

の２つの解説がある。

ここでは、①の Lab の最初の部分を元に、なるべく平易に筆者なりの解説を行う。。

まず、require ‘dll’とすると、15!:0 で定義される J のプリミティブを、cd として、いろいろな機能が有効になる。

J の system¥examples¥dllwrite に Lab を動かす各種ファイルがはいっている。

dlltest.cpp … C++で書かれたプログラムファイル

dlltest.def … Export 用の参照ファイル

dlltest.dll … 上の２つで作られる DLL ファイル

この小解説で必要な部分は C++による次の定義である。

```
void __stdcall incval() {++val;}
int  __stdcall getval() {return val;}
```

これで分かるように、C++で書かれたこれら２つの関数定義 incval と getval とは、システムコールとして、それぞれ値を１増やす、その値を返すという処理を行い、結果を返すものである。なお、C では慣用で関数と呼んでいるが、実際は引数なしなので手続きである。

なお、C++のプログラム dlltest.cpp のリスティングは最後にあげた。

これらを起動するにはつぎのようにする。

```
  open 'system¥examples¥dllwrite¥dlltest.cpp'
```

つぎにＪでDLLを実行するには、つぎの書式で行う。

```
   'filename   procedure          declaration'          cd  parameters
    ファイル名  関数名（手続き名） 引数、戻り値の型など  cd   右引数
```

Ｊとしての左引数はすべて文字列で記述する。 ファイル名はドライブ名から省略せずフルネームで記す。これはＪの命令ではあるが、Ｃの慣用にあわせてある。

現在の実験のために、ファイル名は長いので名詞DLLFILで置き換える。

```
   DLLFIL =: ''"e:¥j402¥system¥examples¥dllwrite¥dlltest.dll"'
```

さて、それでは実験をしてみよう。

```
   (DLLFIL,'incval n') cd ''
+-+
|0|
+-+
```

手続きincvalでは１増やす処理操作を行う。指定nにより結果はなしで、ボックスで0が返される。

つぎの実験をする。

```
   (DLLFIL,'getval i') cd ''
+-+
|1|
+-+
```

手続きgetvalでは今の処理結果を受け取る。指定iにより整数値としてボックスで1が返される。

再び、手続きincvalをやってみる。

```
   (DLLFIL,'incval n') cd ''
+-+
|0|
+-+
```

そして、手続きgetvalを行う。

```
   (DLLFIL,'getval i') cd ''
+-+
|2|
+-+
```

さらに１増えて、今度は２が返された。

このようにして、ＪからC++で書かれたDLLが実行できたのである。

ここで、cdの左引数のdeclarationは、Ｃ言語の書式（int, floatなどデータ型）に合わせた文字列で与える。 したがって、これに合わないときはエラーになり、ときにはメモリやシステムのクラッシュとなることもある。J User ManualやLabの中ではこの注意を何回も警告している。

Ｊの以前のUser Manualでは、例えばWindows開始のチャイム音などはDLLで出来ているので、簡単に実験できるとしているが、筆者は試していない。

筆者の個人的な感想を述べれば、通常のＪユーザであれば、ＪのDLL機能はほとんど必要を感じないと思う。 Labにあるファイル処理などは、通常のfilesライブラリで安全に行えばすむことで、DLLを使ってあえて危険を冒す必要もない。

C++プログラム dlltest.cpp のリスティング

```
// lab "DLL: writing and using a DLL"
#include <windows.h>
BOOL APIENTRY DllMain(HANDLE hInst, DWORD reason, LPVOID){ return 1;}

int val=0; char data[]="static data"; // globals

// __stdcall - windows API DLL calling convention
void  __stdcall incval(){++val;}
int   __stdcall getval(){return val;}
short __stdcall incs(short s){return ++s;}
char  __stdcall incc(char c){return ++c;}
int   __stdcall inci(int i){return ++i;}
float __stdcall incf(float f){return ++f;}
double__stdcall incd(double d){return ++d;}
int   __stdcall addi(int x, int y){return x+y;}

int   __stdcall addvi(int n, int* p){ int r=0;
      for(int i=0; i<n; ++i) r+=*p++; return r;}

void  __stdcall incvi(int n, int* p){
      for(int i=0; i<n; ++i) *p++ +=1;}

void  __stdcall incvs(int n, short* p){
      for(int i=0; i<n; ++i) *p++ +=1;}

void  __stdcall getdata(char* c) {strcpy(c,data);}
char* __stdcall getdatap(){return data;}
void  __stdcall getdatapx(char* *p){ *p = data;}

// __cdecl - default C calling convention
// __cdecl is the alternate (+ cd flag) calling convention
int   altinci(int i){return ++i;}
```