Linux での共有ライブラリの作成方法

１．　共有ライブラリの作成

C ソースファイルの開発においては、共有ライブラリにすることを意識する必要
はありません。
またコンパイルに関しては、オブジェクトファイルを PIC 形式にしておきます。PIC 形式は
gcc 特有のオプションです。このオプションを付加すると出力されるコードが位置非依存
型になります。速度の問題を考慮しないなら、単なるおまじないと考えれば良いでしょう。
ライブラリ化するとき、スタティックライブラリを作成する場合には ar コマンドを使いました。
共有ライブラリを作成する場合は、もう一度 gcc を使います。-shared オプションを使うこと
により、共有ライブラリが出力されます。

Makefile の例。

```
OBJ=oid.o hudp.o    xxxget.o
libxx.so.1.0 : $(OBJ)
        gcc -shared -o libxx.so.1.0 $(OBJ)
descr : main.o
        gcc -g -o    descr main.o -lsocket -lnsl -ldl
.c.o :
        gcc -O -g -c -fPIC -o $*.o $<
```

つまり、コンパイル時には、

```
gcc -O -fPIC -o../obj/test.o  test.c
```

の如く、PIC コードを作成する事がポイントです。

リンク時は、

```
gcc -shared -o../bin/libxx.so.1.0 ../obj/*.o
```

の如く、-shared オプションを指定して gcc でライブラリ化する事がポイントです。

libxx.so.1.0 ができたら、シンボリックリンク libxx.so.1 を作成しておきます。
共有ライブラリは慣用的に libxx.so.x.y という形式の名前をとります。このなかで、
拡張子、ｘ、ｙはそれぞれ共有ライブラリの major 版番号、minor 版番号とします。
共有ライブラリを使用する側からは、単に libxx.so.x を使用させます。もし、後ほど

共有ライブラリのを改版させなければならない場合、minor 番号だけインクリメントした
ファイルをリリースし、シンボリックリンク libxx.so.1 を新版の共有ライブラリ
に付け替えます。こうすることにより、共有ライブラリを改版しても呼び出し側
をリコンパイルする必要はなくなります。呼び出し側も改版しなければならないような
重大な改版の場合には、major 番号をインクリメントします。
$ln -s libxx.so.1.0 libxx.so.1

出来上がった共有ライブラリの内容を確認します。

$nm libxx.so.1

というコマンドを使用します。

２．　共有ライブラリの使用

共有ライブラリ libxx.so.1 の中の xxx_get を呼び出すプログラムを作成します。
概要は、まず dlopen(3X)を用いて指定の共有ライブラリをロードします。
次に dlsym(3X)を使って、関数の開始アドレスを取得し、その関数を呼び出します。
これらのシステムコールのプロトタイプは<dlfcn.h>にあり、
ライブラリは libld.a です。前章にある Makefile のなかで、descr というコマンドを
make している行で、リンクオプションに-ldl が付加されています。

以下に示すプログラムは、XXX を用いてＬＡＮ機器の名前とリブートしてからの経過時間
を取得し、表示するプログラムです。

```
#include <dlfcn.h>//ダイナミックローディング用ヘッダファイルです
#include <stdio.h>
#include "hxxx.h"
#define SOFILE          "./libxx.so.1"//共有ライブラリのパス名を指定しています
#define XXX_GET         "xxx_get"//共有ライブラリの呼び出す関数を指定しています

static void *libxxx;
static int (*func)();
```

```
static void usage(char *cmd)
{
        printf("usage :%s ip    xxx-community\n",cmd);
}

void main(int argc,char **argv)
{
        extern int dump_flag;
        OREC       *porec[3];
        OREC       orec[2];
        char       descr[2048];
        int        uptime,ret,reason;
        if(argc != 3 && argc != 4){
                usage(argv[0]);
                exit(1);
        }

        libxxx = dlopen(SOFILE,RTLD_LAZY);//ここでライブラリファイルをオープンします
        if(libxxx==NULL){
                fprintf(stderr,"cannot load %s\n",SOFILE);
                exit(1);
        }
        func = dlsym(libxxx,XXX_GET);//関数へのポインタを取得します
        if(func==NULL){
                fprintf(stderr,"cannot find %s in %s\n",XXX_GET,SOFILE);
                exit(1);
        }
        porec[0]=&orec[0];
        sprintf(orec[0].oid,"43.6.1.2.1.1.1.0");
        orec[0].type=STRING;
        orec[0].data=descr;
        porec[1]=&orec[1];
        sprintf(orec[1].oid,"43.6.1.2.1.1.3.0");
        orec[1].type=NUMERIC;
        orec[1].data=&uptime;
        porec[2]=NULL;
        ret = (*func)(argv[1],argv[2],porec,&reason);//ライブラリ関数を呼び出します
        if(ret != 0 ){
                printf("xxx error ret=%d    reason=%d\n",
                        ret,reason);
```

```
        }else{
                printf("%s\n%d\n",descr,uptime);
        }
}
```