解决c++ - Creating a library file in makefile and compiling after that

itPublisher 分享于 2017-03-30

My problem is fairly easy but I just don't know how to solve it. I know how to compile and library and link against it if I'm not using a make file because then I can just call ar separately and everything goes right.

Anyway I'm using a petsc library and I'm using a makefile what they provided:

```
CFLAGS           =
FFLAGS           =
CPPFLAGS         =
FPPFLAGS         =
LOCDIR           = /home/user/.../.../    # Working folder
EXAMPLESC        = main.cpp class.cpp         #.cpp file names here
EXAMPLESF        =
#MANSEC           = Mat I don't know what this is but it seems to work
without it.

include ${PETSC_DIR}/conf/variables
include ${PETSC_DIR}/conf/rules

myProgram: main.o class.o  chkopts
    -${CLINKER}  -o myProgram main.o class.o ${PETSC_MAT_LIB}
    ${RM} main.o class.o

include ${PETSC_DIR}/conf/test
```

ARFLAGS will be -rv as a default so where should I provide such a information as

```
ar -rv libclassdll.a class.o
```

and where should I add -L./-lclassdll ?

I'm quite a rookie with makefiles so that's why I'm a bit lost here :<

I tried to change the line to

```
myProgram: main.o class.o  chkopts
    -${CLINKER}  -o myProgram main.o class.o ${AR} libclassdll.a
class.o ${PETSC_MAT_LIB}
    ${RM} main.o class.o
```

and then my compiling command seems to be mpicxx -o myProgram main.o class.o /usr/bin/ar/ libclassdll.a class.o -L (a lot of linking here) and at least it says: g++ classdll.a no such a file or dir.

So it doesn't generate even a lib file for me. So any ideas will be really appreciated.

A new problem when I uploaded makefile on the different machine, my current makefile looks like this

```
LibMyClass.so: MyClass.o  chkopts
    -${CLINKER}  -
 d -Wl,-soname,${SONAME} -o ${VERS}   *.o  ${PETSC_MAT_LIB}

    mv ${VERS} ${LIBADD}
```


相关内容推荐

1 微信公众号文章采集,并发布到WordPress

最新文章

1 opencv_contrib based on CMake全过程
2 WebRTC编译系统之gn和ninja
3 Boost.Asio c++ 网络编程翻译（17）
4 libevent evhttp学习——http客户端
5 c++之STL(13) STL 算法 - 查找算法（2） search_n() search_n(b, e, c, v) search_n (b, e, c, v, p)
6 c++之STL(13) STL 算法 - 查找算法（1）

That works on one machine but other machine gives following error

```
/usr/bin/ld: MyClass.o: relocation R_X86_64_32S against `.rodata' can
not be used when making a
 d object; recompile with -fPIC
MyClass.o: could not read symbols: Bad value
```

I did change the paths of course but I guess that indicates other kind of problem because even if I type "g++ -
d -Wl,-soname,libmyclass.so.1 -o libmyclass.so.1.0 MyClass.o" or "g++ -fPIC -
... " I'll get the same error.

answer 1 >>解决方法

Ideally you should construct the library first, then use it, just as you would "by hand".

To construct (or update) the library, you need a rule something like this:

```
libclassdll.a: class.o
    ar -rv libclassdll.a class.o
```

Or more concisely, like this:

```
libclassdll.a: class.o
    ar $(ARFLAGS) $@ $^
```

Then the rule for `myProgram` becomes:

```
# Assuming CLINKER is something civilized, like gcc
myProgram: main.o libclassdll.a  chkopts
    -${CLINKER} -o myProgram main.o -L. -lclassdll ${PETSC_MAT_LIB}
```

or better:

```
myProgram: main.o libclassdll.a  chkopts
    -${CLINKER} -o $@ $< -L. -lclassdll ${PETSC_MAT_LIB}
```

So in your makefile, you would replace

```
myProgram: main.o class.o  chkopts
    -${CLINKER}  -o myProgram main.o class.o ${PETSC_MAT_LIB}
    ${RM} main.o class.o
```

with

```
myProgram: main.o libclassdll.a  chkopts
    -${CLINKER} -o $@ $< -L. -lclassdll ${PETSC_MAT_LIB}

libclassdll.a: class.o
    ar $(ARFLAGS) $@ $^
```

There are other refinements you can make, but that should be enough for now.

用户评论

游客

写点什么...

发表