# 33. 扩展 DPDK

本章描述了开发者如何通过扩展DPDK来提供一个新的库、目标文件或者支持新的开发板。

## 33.1. 示例：添加新的库 libfoo

要添加新的库到DPDK，按照如下操作：

1. 添加新的配置选项：

for f in config/\\*; do \

echo CONFIG\_RTE\_LIBFOO=y >> $f; done

1. 创建新的源码目录：

mkdir ${RTE\_SDK}/lib/libfoo

touch ${RTE\_SDK}/lib/libfoo/foo.c

touch ${RTE\_SDK}/lib/libfoo/foo.h

1. 源码添加 foo() 函数。

函数定义于 foo.c:

void foo(void){

}

函数声明于 foo.h:

extern void foo(void);

1. 更新文件 lib/Makefile:

vi ${RTE\_SDK}/lib/Makefile# add:# DIRS-$(CONFIG\_RTE\_LIBFOO) += libfoo

1. 为新的库创建新的 Makefile，如派生自 mempool Makefile，进行修改：

cp ${RTE\_SDK}/lib/librte\_mempool/Makefile ${RTE\_SDK}/lib/libfoo/

vi ${RTE\_SDK}/lib/libfoo/Makefile# replace:# librte\_mempool -> libfoo# rte\_mempool -> foo

更新文件 mk/DPDK.app.mk，添加 -lfoo 选项到 LDLIBS 变量中。 链接DPDK应用程序时会自动添加此标志。

添加此新库之后，重新构建DPDK (此处仅显示这个特殊的部分)：

cd ${RTE\_SDK}

make config T=x86\_64-native-linuxapp-gcc

make

1. 检测这个库被正确安装了：

ls build/lib

ls build/include

### 33.1.1. 示例：在测试用例中使用新库 libfoo

测试应用程序用于验证DPDK的所有功能。 一旦添加了一个库，应该在测试用例程序中添加一个用例。

* 新的测试文件 test\_foo.c 被添加，包含头文件 foo.h 并调用 foo() 函数。 当测试通过时，test\_foo() 函数需要返回0。
* 为了处理新的测试用例，Makefile, test.h 和 commands.c 必须同时更新。
* 测试报告生成：autotest.py 是一个脚本，用于生成文件 ${RTE\_SDK}/doc/rst/test\_report/autotests 目录中指定的测试用例报告。 如果libfoo处于新的测试家族，链接 ${RTE\_SDK}/doc/rst/test\_report/test\_report.rst 需要更新。
* 重新构建DPDK库，添加新的测试应用程序：

cd ${RTE\_SDK}

make config T=x86\_64-native-linuxapp-gcc

make