(1条消息) linux将c++程序制作成.deb_hongge_smile的博客-CSDN博客

有的时候,我们需要将自己的程序(动态库或者可执行程序)打包成一个deb包,将deb包拷贝到其他的电脑,安装就可以使用包里面包含的函数或者可执行程序了.
linux将C++城西制作deb包,通过查找资料,找到了下面的几种方法.

方法1 使用cmake

C++ 工程大部分都是用 CMake 配置编译， 而 CPack 是 CMake 内置的工具，支持打包成多种格式的安装包。因为是 CMake 的内置工具，所以使用的方式也是通过在 CMakeLists.txt 配置参数，就能达到我们的需求。使用起来很方便，容易上手。

下面举一个简单的例子, 一个c++工程的目录结构如下：

```
.
└── cmake
    ├── CMakeLists.txt
    ├── include
    │   └── add_num.h
    ├── README.md
    └── src
        ├── add_num.cpp
        └── main.cpp
```

其中每个文件的内容如下：

include/add_num.h:

```
class A2DD
{
  int gx;
  int gy;

public:
  A2DD(int x,int y);
  int getSum();
};
```

- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

src/add_num.cpp

```
A2DD::A2DD(int x,int y)
{
  gx = x;
  gy = y;
}

int A2DD::getSum()
{
  return gx + gy;
}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12

src/main.cpp

```
using namespace std;

int main(int argc, char *argv[]){
   A2DD a(1, 2);
   int res = a.getSum();
   cout << res << endl;
   return 0;
}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

- 9
- 10
- 11

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.12.2)

project (addnum)

add_library(addnum STATIC src/add_num.cpp)
target_include_directories( addnum
    PUBLIC
        "${CMAKE_CURRENT_SOURCE_DIR}/include"
)

add_executable(addnumapp src/main.cpp)
target_link_libraries(addnumapp addnum)

install(TARGETS addnumapp
    COMPONENT linapp
    RUNTIME DESTINATION "/home/"
    LIBRARY DESTINATION "/home/"
    DESTINATION "/home/"
)

SET(CPACK_GENERATOR "DEB")
SET(CPACK_DEBIAN_PACKAGE_MAINTAINER "KK")
INCLUDE(CPack)
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23

CMakeLists.txt内容说明:

- 第1行指定了我们正在使用的CMake版本。这是必需的说明。
- 第3行-我们告诉CMake，我们想在此目录中创建一个项目，并将其称

为“ addnum”。

- 第5行-我们告诉CMake使用文件src / add_num.cpp创建一个名为“ addnum”的静态库。add_library()命令在CMake世界中创建我们所谓的“目标”。这里的目标是一个静态库。
- 第6–9行—我们告诉CMake使用包含目录中的头文件作为目标“ addnum”。CMAKE_CURRENT_SOURCE_DIR是一个CMake变量，它提供了我们在其中调用目录的当前路径。
- 第11行-我们告诉CMake使用src/main.cpp创建另一个目标“ addnumapp”。但是，这次我们告诉它使其成为可执行文件（与平台无关）而不是库。
- 第12行-我们刚刚创建的可执行文件使用了在第5行中创建的库。因此，我们告诉CMake将库“ addnum”链接到可执行目标“ addnumapp”。
- 此时，您应该能够运行CMake并获取可以从命令行运行并执行它的内置可执行文件。到目前为止，通过运行“ g ++ main.cpp”，我们已经完全完成了以前的工作。
- CMake自动选择系统上安装的编译器。因此，您不必担心显式调用编译器。
- 第14-19行-我们使用install()命令告诉CMake在安装时要做什么。我们告诉命令我们要用于安装程序的目标（“ addnumapp”）。我们还告诉目标位置我们想要安装的位置，在这种情况下，它是“ home/”目录。RUNTIME和LIBRARY目标指定了在我们有其他库的情况下希望放置库的目标。
- 第21-23行-CPack是CMake使用的打包工具。这些说明是不言自明的。

生成**deb**安装包

```
mkdir build && cd build
cmake ..
cmake --build .
cpack
```

- 1
- 2
- 3
- 4

执行完上述命令以后，在build目录下生成了addnum-0.1.1-Linux.deb。

打包完成后，我们可以通过以下命令查看一下这个deb软件包的信息和内容:

```
dpkg-deb -I <deb包文件>
dpkg -c <deb包文件>
```

- 1
- 2

附录： **deb**包的简单使用

```
sudo dpkg -i <deb包文件>
sudo dpkg -l
sudo dpkg -P <包名称>
```

- 1
- 2
- 3

cpack的其他的参数说明：

参考cmake官网：https://cmake.org/cmake/help/latest/module/CPack.html

```
set(_VERSION_MAJOR 1)
set(_VERSION_MINOR 2)
set(_VERSION_PATCH 3)
set(CPACK_PACKAGE_VERSION_MAJOR "${_VERSION_MAJOR}")
set(CPACK_PACKAGE_VERSION_MINOR "${_VERSION_MINOR}")
set(CPACK_PACKAGE_VERSION_PATCH "${_VERSION_PATCH}")

set(CPACK_DEBIAN_PACKAGE_NAME "mork_printer_project_demo")

set(CPACK_DEBIAN_PACKAGE_ARCHITECTURE "amd64")



set(CPACK_PACKAGE_CONTACT "supporter@gmail.com")

set(CPACK_DEBIAN_PACKAGE_MAINTAINER "members of support@gmail.cn")

INSTALL(TARGETS LinkLibraryDemo
        RUNTIME DESTINATION /tmp/cmake_demo


        )
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

- 22
- 23

此方法打包的优缺点：

优点：操作简单

缺点：如果打包的工程依赖了第三方库，则这些三方库需要以静态库(static)的形式链接到项目中去，否则，如果将deb包安装在另外一台不包含这些库的电脑上，可执行程序将无法运行。(或者可以在目标电脑上安装相应的库文件)

方法2 使用dpkg-deb

dpkg-deb命令是Debian Linux下的软件包管理工具，它可以对软件包执行打包和解包操作以及提供软件包信息。

下面举一个简单的例子, 一个c++工程add_num的目录结构如下(和上面的例子一样)：

```
.
└── cmake
    ├── CMakeLists.txt
    ├── include
    │   └── add_num.h
    ├── README.md
    └── src
        ├── add_num.cpp
        └── main.cpp
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

每个文件的内容同上。

```
mkdir build && cd build
cmake ../
make -j4
```

```
cd ../
mkdir DEBIAN
echo "Package: add_num
Version: 1.0
Section: custom
Priority: optional
Architecture: all
Essential: no
```

```
Installed-Size: 1024
Maintainer: linuxconfig.org
Description: Print linuxconfig.org on the screen" >DEBIAN/control
mkdir -p ./usr/bin
cp ./build/addnumapp ./usr/bin/
rm -rf build
cd ../
dpkg-deb --build add_num
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22

此时在当前目录下会生成add_num.deb。

查看deb包信息, dpkg -I add_num.deb

```
new Debian package, version 2.0.
size 18838636 bytes: control archive=348 bytes.
    199 bytes,     9 lines      control
Package: linuxconfig
Version: 1.0
Section: custom
Priority: optional
Architecture: all
Essential: no
Installed-Size: 1024
Maintainer: linuxconfig.org
Description: Print linuxconfig.org on the screen
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

- 11
- 12

dpkg -c add_num.deb:

```
drwxr-xr-x kfr/kfr           0 2020-10-29 18:11 ./
-rw-r--r-- kfr/kfr         278 2020-10-29 18:10 ./CMakeLists.txt
-rw-r--r-- kfr/kfr          41 2020-10-29 18:10 ./README.md
drwxr-xr-x kfr/kfr           0 2020-10-29 18:10 ./include/
-rw-r--r-- kfr/kfr          81 2020-10-29 18:10 ./include/add_num.h
drwxr-xr-x kfr/kfr           0 2020-10-29 18:10 ./src/
-rw-r--r-- kfr/kfr         111 2020-10-29 18:10 ./src/add_num.cpp
-rw-r--r-- kfr/kfr         177 2020-10-29 18:10 ./src/main.cpp
drwxr-xr-x kfr/kfr           0 2020-10-29 18:10 ./usr/
drwxr-xr-x kfr/kfr           0 2020-10-29 18:10 ./usr/bin/
-rwxr-xr-x kfr/kfr        9080 2020-10-29 18:10 ./usr/bin/addnumapp
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

此方法打包的优缺点：

优点：操作简单

缺点：如果打包的工程依赖了第三方库，则需要将依赖的第三方库和deb文件一起拷贝到目标电脑上，否则可执行文件找不到依赖库。

解决办法： 在**DEBIAN**文件夹同级目录下新建一个**lib**文件夹，将可执行程序依赖的动态库都拷贝到**lib**文件夹中去。

参考链接

Creating debian packages — CMake： https://blog.usejournal.com/creating-debian-packages-cmake-e519a0186e87

cpack官方教程：https://cmake.org/cmake/help/latest/module/CPack.html

dpkg-deb： https://linuxconfig.org/easy-way-to-create-a-debian-package-and-local-package-repository

dpkg命令大全：https://man.linuxde.net/dpkg

debmake使用：https://www.debian.org/doc/manuals/debmake-doc/apa.en.html
debian包介绍:https://wiki.debian.org/Packaging/Intro?action=show&redirect=IntroDebianPackaging
cpack链接:https://cmake.org/cmake/help/latest/module/CPack.html

cpack使用介绍:https://zhuanlan.zhihu.com/p/141956373
cpack:https://juejin.im/post/6844903558551175175
cpack制作deb:https://blog.csdn.net/cedian0443/article/details/106456999/
Making a Debian Package: https://people.debian.org/~jaldhar/make_package1.html