一、关于使用 gcc 动态库和静态库。

1.静态库：

程序编译时会被连接到目标代码中。

程序运行时不需要静态库。

2.动态库

程序编译时不会被连接到目标代码中。

程序运行时需要动态库载入。

（一）、实例

（1）.创建一个项目

（2）.输入代码

```
#ifndef HELLO_H

#define HELLO_H

void hello(const char *name);

#endif//HELLO_H
```

转为 c：

```
#include<stdio.h>

void hello(const char *name)

{
printf(“Hello %s\n”,name);

}
```

主函数 main.c：

```
#include"hello.h"

int main()
```

```
{
hello(“everyone”);

return 0;

}
```

（3）.使用 gcc 编译得到.o 文件

```
gcc -c hello.c
```

（二）、使用静态库

为创建静态库所使用的工具：ar

静态库文件的命名规范：以 lib 作为前缀,是.a 文件

创建静态库：

```
ar -crv libmyhello.a hello.o
```

使用静态库：

```
gcc -o hello main.c -L. -lmyhello

gcc main.c libmyhello.a -o hello
```

生成 main.o gcc -c main.c

生成可执行文件 gcc -o hello main.c libmyhello.a

（三）、使用动态库

为创建动态库所使用的工具：gcc

动态库文件命名规范：以 lib 作为前缀,是.so 文件

使用动态库：

```
gcc -o hello main.c -L. -lmyhello
```

（四）、库的实例使用

1.R1.c 代码为：

```
#include<stdio.h>
void print1(int arg)
{
printf(“R1 print arg:%d\n”,arg);
}
```

R2.c 代码为：

```
#include<stdio.h>
```

```
void print2(char *arg)
{
printf(“R2 printf arg:%s\n”,arg);
}
```
R.h 为：

```
#ifndef A_H
#define A_H
void print1(int);
void print2(char *);
#endif
```
test.c:

```
#include<stdio.h>
#include"A.h"
int main()
{
print1(1);
print2(“test”);
return 0;
}
```
在程序中使用静态库：

```
ar crv libfile.a A1.o A2.o
gcc -o test test.c libfile.a
```
动态库使用：

```
gcc -shared -fPIC -o libfile.so A1.o A2.o
gcc -o test test.c libfile.so
```

二、OpenCV 的使用：

1.准备使用环境：

打开虚拟机终端，输入：

```
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
```
2.安装：

配置:cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local …

build:sudo make

安装：sudo make install

执行 sudo gedit /etc/ld.so.conf.d/opencv.conf 命令后打开一个空白的文件 在文件中添加 /usr/local/lib

保存回到命令行界面,执行如下命令使得刚才的配置路径生效

```
sudo ldconfig
```
配置 bash：sudo gedit /etc/bash.bashrc

在文件末尾添加：

PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig

Export PKG_CONFIG_PATH

配置生效：

root

查看 OpenCV 版本：

pkg-config --modversion opencv

编写一个打开图片进行特效显示的代码 test1.cpp:

编写程序 test1.cpp:

```
#include <opencv2/highgui.hpp>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main(int argc, char** argv)
{
CvPoint center;
double scale = -3;

IplImage* image = cvLoadImage("shenzao.jpg");
argc == 2? cvLoadImage(argv[1]) : 0;

cvShowImage("Image", image);


if (!image) return -1;       center = cvPoint(image->width / 2, image->height / 2);
for (int i = 0;i<image->height;i++)
      for (int j = 0;j<image->width;j++) {
            double dx = (double)(j - center.x) / center.x;
            double dy = (double)(i - center.y) / center.y;
            double weight = exp((dx*dx + dy*dy)*scale);
            uchar* ptr = &CV_IMAGE_ELEM(image, uchar, i, j * 3);
            ptr[0] = cvRound(ptr[0] * weight);
            ptr[1] = cvRound(ptr[1] * weight);
            ptr[2] = cvRound(ptr[2] * weight);
      }

Mat src;Mat dst;
src = cvarrToMat(image);
cv::imwrite("test.png", src);

cvNamedWindow("test",1);          imshow("test", src);
 cvWaitKey();
```

```
 return 0;
}
```

然后执行以下命令： g++ test1.cpp -o test1 pkg-config --cflags --libs opencv

执行以下命令：./test1：

打开的图片：

练习使用 opencv 库编写打开摄像头压缩视频的程序。

1.准备一个.mp4 格式的文件

2.创建一个 test2.cpp 文件。

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main()
{
VideoCapture capture("Palette.mp4");

while(1){
	Mat frame;
	capture >> frame;
	if(frame.empty())
		break;
	imshow("读取视频帧",frame);
	waitKey(30);
}
system("pause");
return 0;
```

3.编译

```
g++ test2.cpp -o test2 pkg-config --cflags --libs opencv
```

结果：