第一步，首先安装node.js,安装包为node-v10.13.0-x64.msi,安装过程：

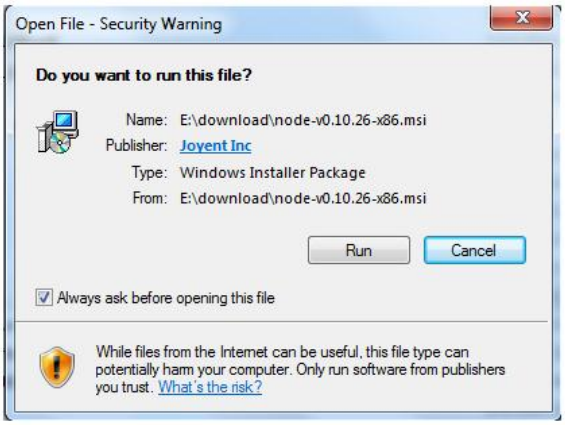
1、Windows 安装包(.msi)

32 位安装包下载地址：https://nodejs.org/dist/v4.4.3/node-v4.4.3-x86.msi

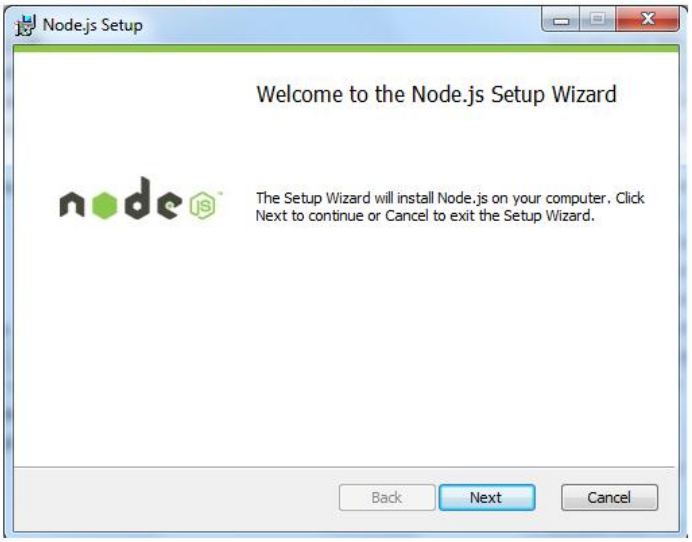
64 位安装包下载地址：https://nodejs.org/dist/v4.4.3/node-v4.4.3-x64.msi

本文实例以 v0.10.26 版本为例，其他版本类似， 安装步骤：

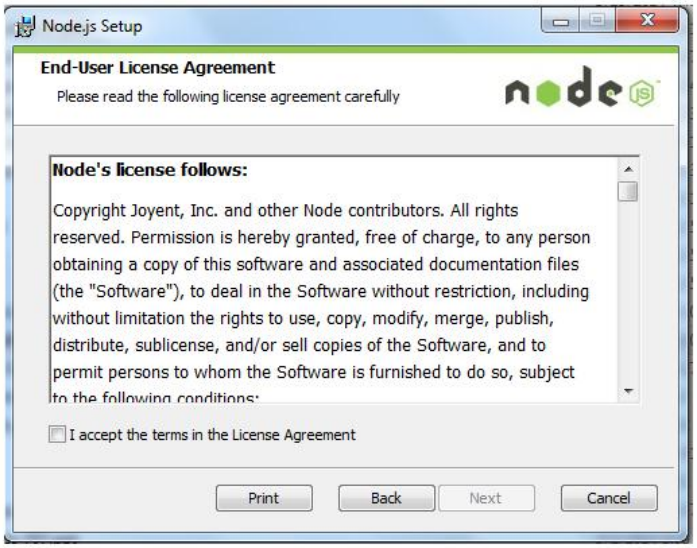
步骤 1：双击下载后的安装包 **v0.10.26**，如下所示：



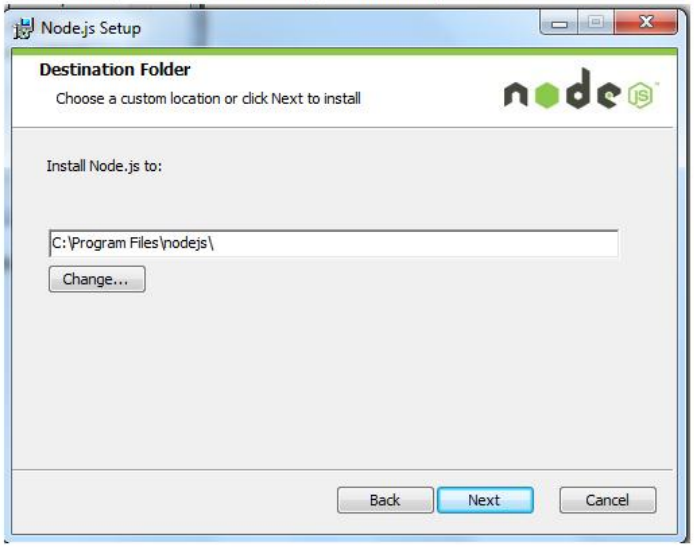
步骤 2：点击以上的Run(运行)，将出现如下界面：



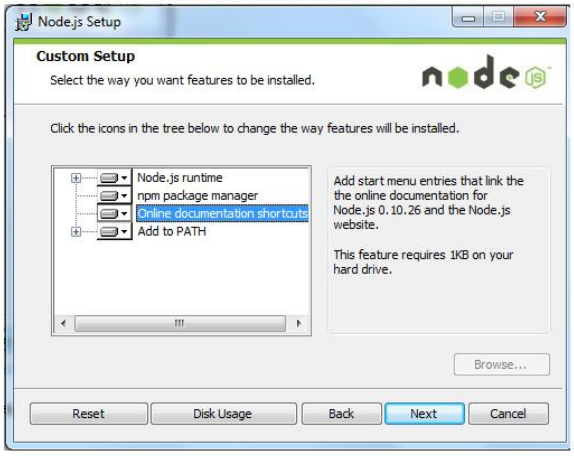
步骤 3 : 勾选接受协议选项，点击 next（下一步）按钮：



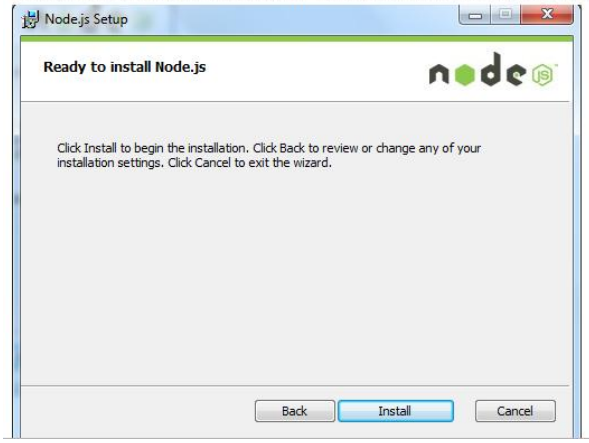
步骤 4 : Node.js默认安装目录为 "C:\Program Files\nodejs\"，你可以修改目录，并点击 next（下一步）：



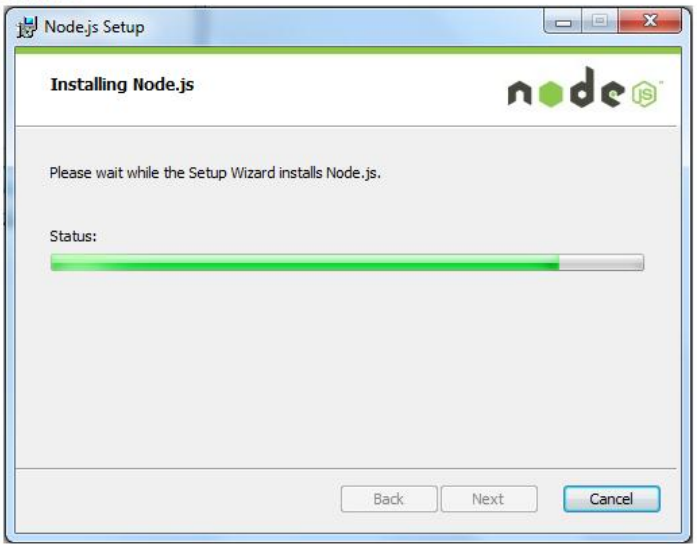
步骤 5 : 点击树形图标来选择你需要的安装模式，然后点击下一步 next（下一步）



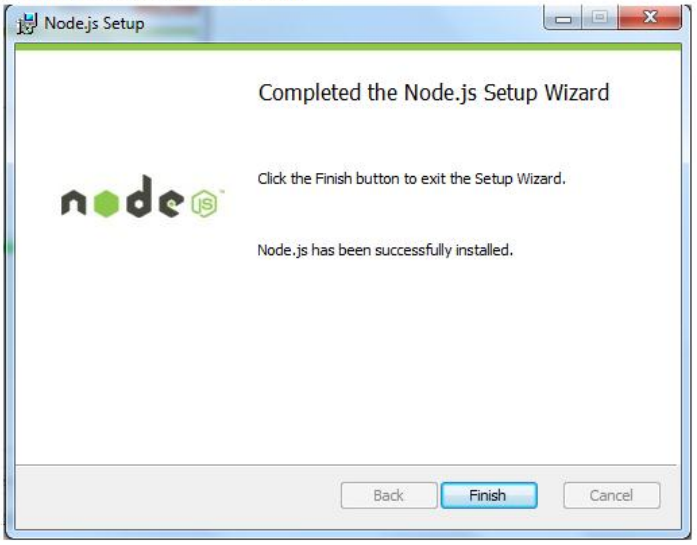
步骤 6 :点击 Install（安装） 开始安装Node.js。你也可以点击 Back（返回）来修改先前的配置。 然后并点击 next（下一步）：



安装过程：



点击 Finish（完成）按钮退出安装向导。



检测PATH环境变量是否配置了Node.js，点击开始=》运行=》输入"cmd" => 输入命令"path"，输出如下结果：

```
PATH=C:\oraclexe\app\oracle\product\10.2.0\server\bin;C:\Windows\system32;
C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;
c:\python32\python;C:\MinGW\bin;C:\Program Files\GTK2-Runtime\lib;
C:\Program Files\MySQL\MySQL Server 5.5\bin;C:\Program Files\nodejs\;
C:\Users\rg\AppData\Roaming\npm
```

我们可以看到环境变量中已经包含了C:\Program Files\nodejs\

检查Node.js版本

Windows上安装过程结束

第二步，安装express框架，在客户端搭建服务器框架，安装过程如下：

windows下安装express

在你的windows上已经安装了node.js的基础上再安装express

第一部分:安装express

第一步:执行 npm install -g express-generator

note:必须安装这个,不然创建express项目的时候会提示express命令没有找到

第二步:执行 npm install -g express

第三步:执行 express -V

note:'V'是大写的,如果express安装成功会显示版本号

第二部分:创建一个express项目

第一步:执行 express youProjectName

note:youProjectName是你的项目的名称,按照自己的要求选择合适的项目名称

第二步 :进入你的项目:cd youProjectName

第三步:在你的项目的目录下执行 npm install

第四步:启动你的项目,执行 npm start

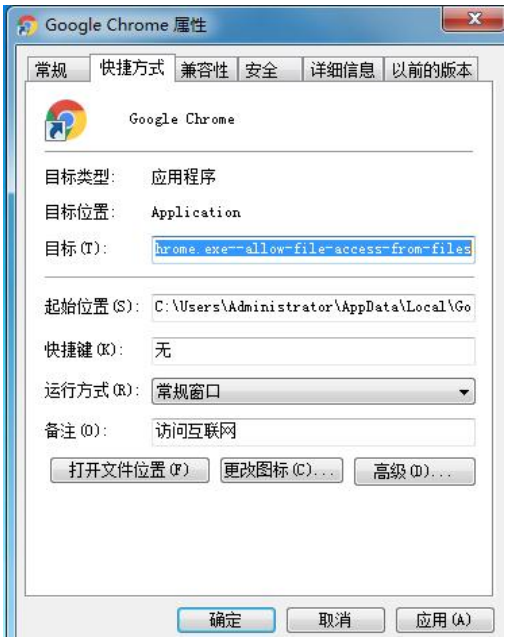
第三部分:在浏览器中访问你的项目

打开你的浏览器,在地址栏中输入:http://127.0.0.1:3000

然后你会看到:express的欢迎信息

实际中，首先在F盘下创建了faceverity的目录，然后在faceverity目录下创建express项目。
其中ser.js就是所做的客户端的服务器。

第三步，就是对浏览器的要求



首先在浏览器的属性中添加--allow-file-access-from-files

然后添加两个插件Allow-Control-Allow-Origin:*1.0.3和CORS0.1.2，从而实现跨域访问（首先应该安装谷歌访问助手）

第四步，安装nginx服务器

Windows机器配置:
Windows7旗舰版 64位
Intel(R) Core(TM)i5-2520 CPU @2.50GHz 2.50 GHz
内存: 4GB

1. 下载 nginx 1.7.11.3 Gryphon
下载链接: http://nginx-win.ecsds.eu/download/nginx 1.7.11.3 Gryphon.zip
下载完成后解压;
将解压后的目录名:
   nginx 1.7.11.3 Gryphon
改成:
   nginx-1.7.11.3-Gryphon

2. 下载服务器状态检查程序 stat.xsl
https://github.com/arut/nginx-rtmp-module/

将nginx-rtmp-module-master.zip解压后复制到目录:nginx-1.7.11.3-Gryphon下,
保证stat.xls的目录为:
nginx-1.7.11.3-Gryphon\nginx-rtmp-module\stat.xsl

3. 配置文件 conf\nginx-win-rtmp.conf 内容如下:

```
#user  nobody;
# multiple workers works !
worker_processes  2;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;

events {
      worker_connections  8192;
      # max value 32768, nginx recycling connections+registry optimization =
      #    this.value * 20 = max concurrent connections currently tested with one worker
      #    C1000K should be possible depending there is enough ram/cpu power
      # multi_accept on;
}

rtmp {
      server {
            listen 1935;
            chunk_size 4000;
            application live {
                  live on;
            }
      }
}

http {
      #include       /nginx/conf/naxsi_core.rules;
```

```
    include          mime.types;
    default_type   application/octet-stream;

    #log_format   main   '$remote_addr:$remote_port - $remote_user [$time_local] "$request" '
    #                    '$status $body_bytes_sent "$http_referer" '
    #                    '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log   logs/access.log   main;

#     # loadbalancing PHP
#     upstream myLoadBalancer {
#           server 127.0.0.1:9001 weight=1 fail_timeout=5;
#           server 127.0.0.1:9002 weight=1 fail_timeout=5;
#           server 127.0.0.1:9003 weight=1 fail_timeout=5;
#           server 127.0.0.1:9004 weight=1 fail_timeout=5;
#           server 127.0.0.1:9005 weight=1 fail_timeout=5;
#           server 127.0.0.1:9006 weight=1 fail_timeout=5;
#           server 127.0.0.1:9007 weight=1 fail_timeout=5;
#           server 127.0.0.1:9008 weight=1 fail_timeout=5;
#           server 127.0.0.1:9009 weight=1 fail_timeout=5;
#           server 127.0.0.1:9010 weight=1 fail_timeout=5;
#           least_conn;
#     }

    sendfile           off;
    #tcp_nopush       on;

    server_names_hash_bucket_size 128;

## Start: Timeouts ##
    client_body_timeout    10;
    client_header_timeout 10;
    keepalive_timeout      30;
    send_timeout              10;
    keepalive_requests     10;
## End: Timeouts ##

    #gzip   on;

    server {
        listen          80;
        server_name   localhost;


        location /stat {
            rtmp_stat all;
            rtmp_stat_stylesheet stat.xsl;
        }
        location /stat.xsl {
            root nginx-rtmp-module/;
        }
        location /control {
```

```
            rtmp_control all;
        }

        #charset koi8-r;
        #access_log   logs/host.access.log   main;

        ## Caching Static Files, put before first location
        #location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
        #      expires 14d;
        #      add_header Vary Accept-Encoding;
        #}

# For Naxsi remove the single # line for learn mode, or the ## lines for full WAF mode
        location / {
            #include      /nginx/conf/mysite.rules; # see also http block naxsi include line
            ##SecRulesEnabled;
          ##DeniedUrl "/RequestDenied";
          ##CheckRule "$SQL >= 8" BLOCK;
          ##CheckRule "$RFI >= 8" BLOCK;
          ##CheckRule "$TRAVERSAL >= 4" BLOCK;
          ##CheckRule "$XSS >= 8" BLOCK;
            root    html;
            index   index.html index.htm;
        }

# For Naxsi remove the ## lines for full WAF mode, redirect location block used by naxsi
        ##location /RequestDenied {
        ##      return 412;
        ##}

## Lua examples !
#             location /robots.txt {
#                rewrite_by_lua '
#                   if ngx.var.http_host ~= "localhost" then
#                      return ngx.exec("/robots_disallow.txt");
#                   end
#                ';
#             }

        #error_page   404                    /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page    500 502 503 504   /50x.html;
        location = /50x.html {
            root    html;
        }

        # proxy the PHP scripts to Apache listening on 127.0.0.1:80
        #
        #location ~ \.php$ {
        #      proxy_pass    http://127.0.0.1;
```

```
        #}

        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        #location ~ \.php$ {
        #       root           html;
        #       fastcgi_pass   127.0.0.1:9000; # single backend process
        #       fastcgi_pass   myLoadBalancer; # or multiple, see example above
        #       fastcgi_index  index.php;
        #       fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        #       include        fastcgi_params;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny  all;
        #}
    }

    # another virtual host using mix of IP-, name-, and port-based configuration
    #
    #server {
    #       listen          8000;
    #       listen          somename:8080;
    #       server_name   somename   alias   another.alias;

    #       location / {
    #           root    html;
    #           index   index.html index.htm;
    #       }
    #}

    # HTTPS server
    #
    #server {
    #       listen          443 ssl spdy;
    #       server_name   localhost;

    #       ssl                         on;
    #       ssl_certificate        cert.pem;
    #       ssl_certificate_key   cert.key;
    #       ssl_session_timeout   5m;
    #       ssl_prefer_server_ciphers On;
    #       ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    #       ssl_ciphers
ECDH+AESGCM:ECDH+AES256:ECDH+AES128:ECDH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!eNULL:!MD5:!DSS:!EXP:!ADH:!LOW:!MEDIUM;

    #       location / {
    #           root    html;
    #           index   index.html index.htm;
```

```
        #        }
        #}

}
```

4. 启动服务器

进入windows的cmd;

```
> cd nginx-1.7.11.3-Gryphon
> nginx.exe -c conf\nginx-win-rtmp.conf
```

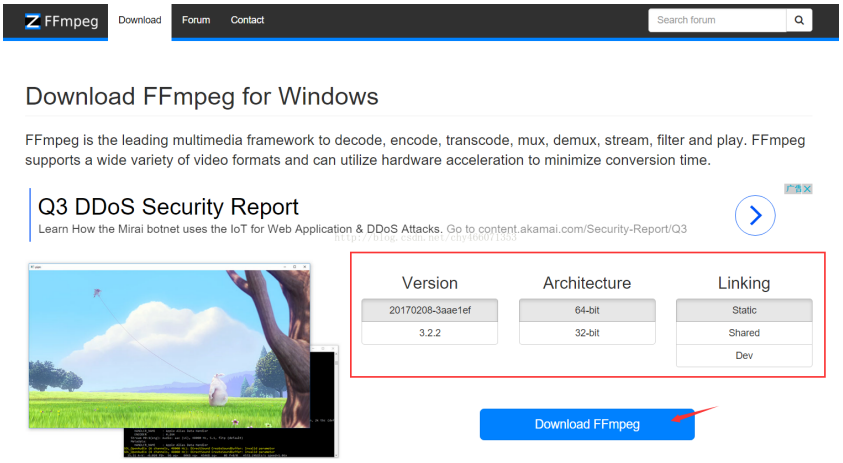
作者：北雨南萍

来源：CSDN

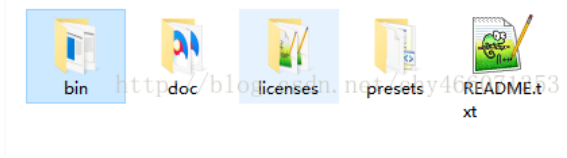
原文：https://blog.csdn.net/fireroll/article/details/51985688

第五步，安装ffmpeg,实现rtsp向rtmp的转换

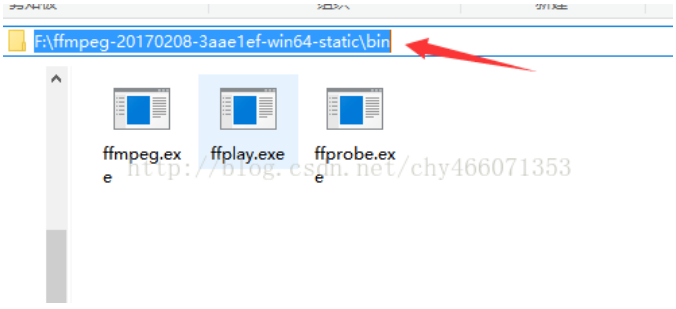
首先下载ffmpeg的windows版本https://ffmpeg.zeranoe.com/builds/



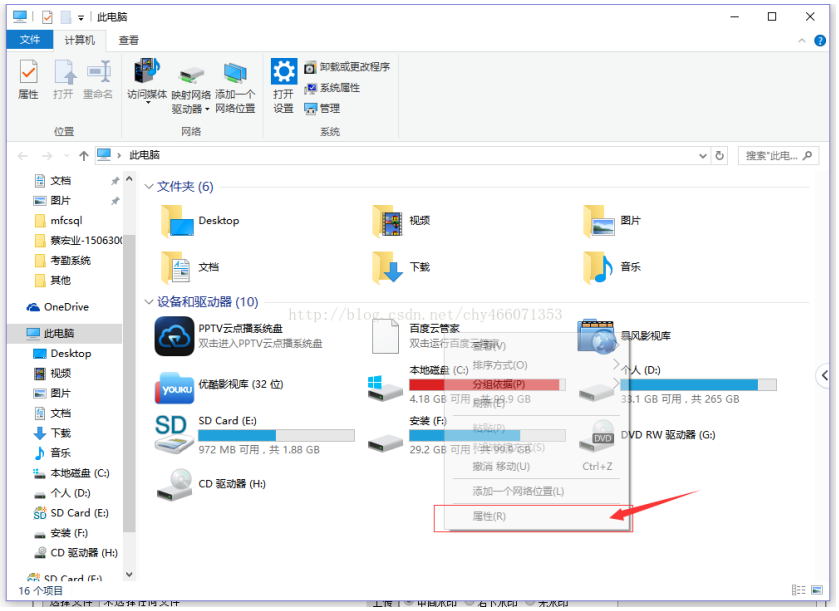
解压下载的压缩包得到



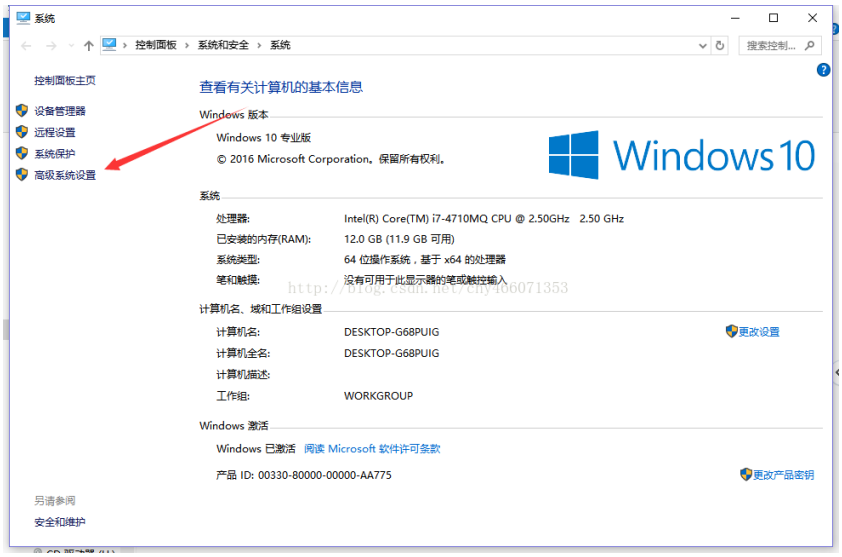
进入bin目录并获取路径



在此电脑界面下右击选择属性

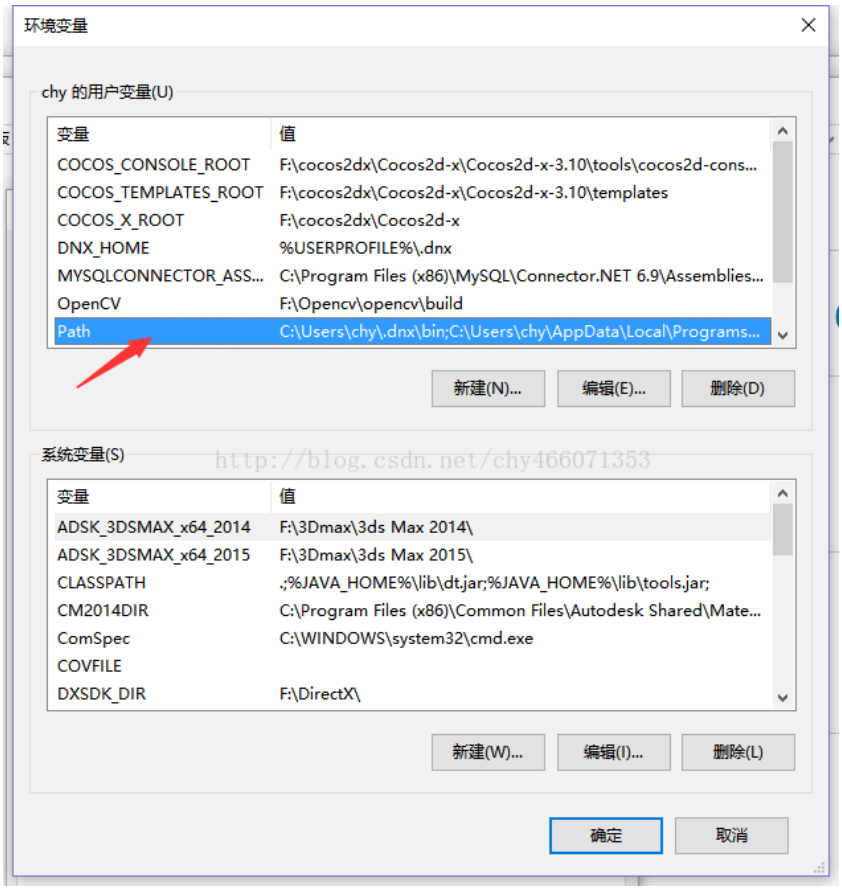


选择高级系统设置

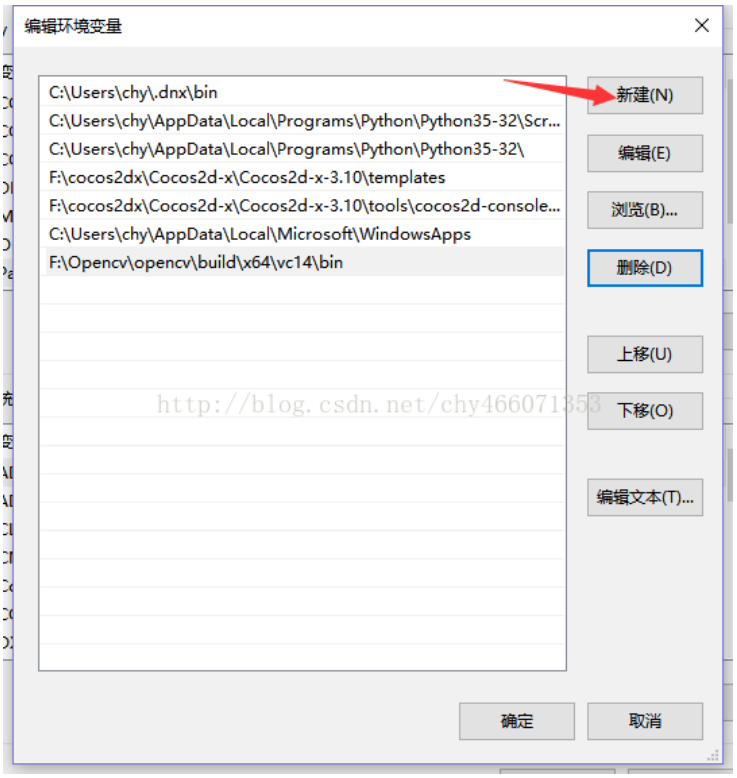


选择环境变量

在用户环境变量双击path

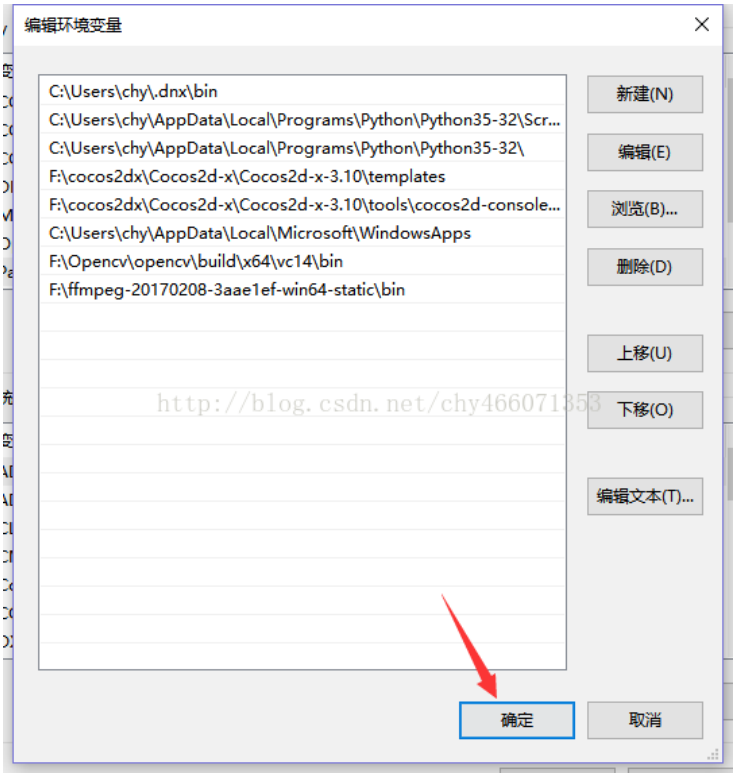


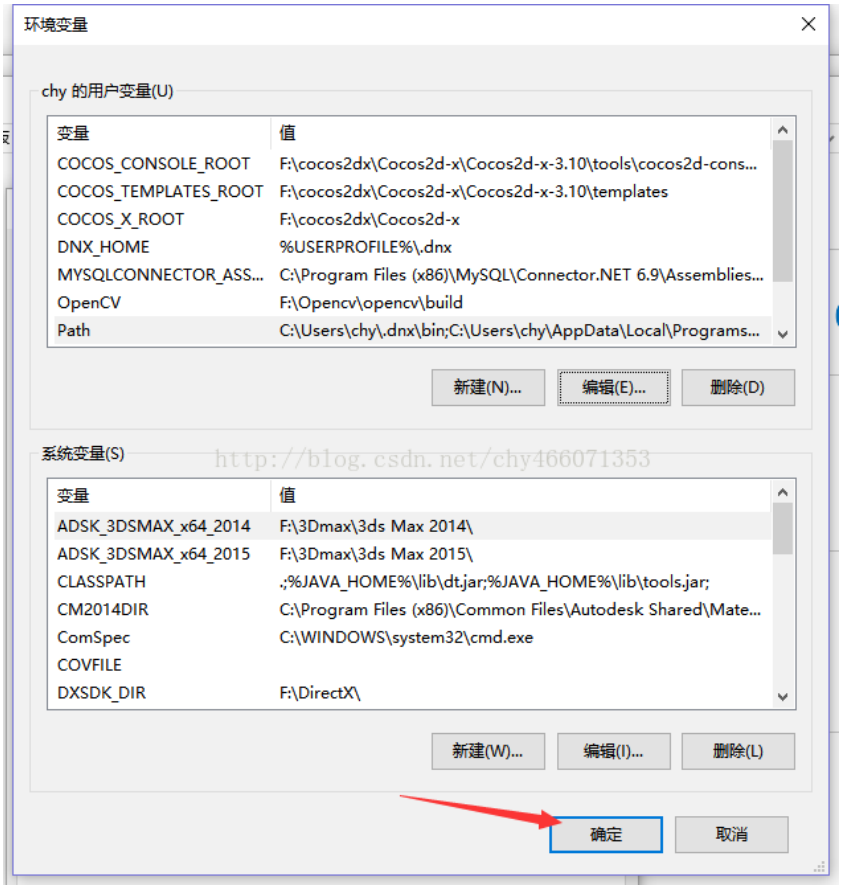
选择新建（注意不要更改其他环境变量）



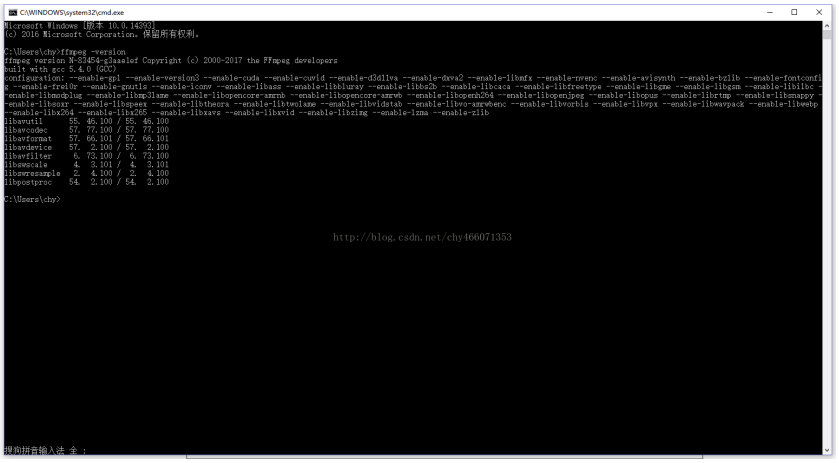
将刚才的bin路径粘贴进去

记得点下方的确定，再关闭当前窗口再点确定以保存





到这里，ffmpeg的配置就差不多了，调用命令行（windows+R输入cmd）输入“ffmpeg –version”，如果出现如下说明配置成功



最后，将rtsp转换为rtmp

ffmpeg -i "rtsp://admin:12345@192.168.0.188:554/h264/ch1/main/av_stream" -f flv -r 25 -s 640x480 -an "rtmp://192.168.0.14:1935/live/test2"