浙大城市学院实验报告

课程名称　物联网技术与应用　实验项目　实验六　HTTP Client 和 WebServer

专业班级　计算机 1803　学号　31801150　姓名

张帅

指导老师（签名 ）　蔡建平　日期　2020.11.9　实验成绩

注意:

- 务请保存好各自的源代码及实验报告文档，以备后用。
- 请把实验报告转为 PDF 文档上传到 BB 平台。
- 文件名格式：学号_姓名_日期_实验，如 30801001_姓名_20200305_实验 02

一、实验目的：

掌握 HTTP 协议的相关理论与实践；掌握 ESP8266 的 ESP8266WiFi.h、WiFiClient.h、ESP8266WebServer.h 等库的基本方法及使用；熟悉网络应用中 WebClient、Web Server 的角色。

二、实验内容：

1. 将 ESP8266 作为 HTTP Client；访问心知天气 API，获取天气信息；
2. 将 ESP8266 作为 Web Server，PC 浏览器可以基于 IP 地址访问 WebServer；
3. 实现 ESP8266 带登录认证的 Web Server；
4. 对以上实验抓包分析。

三、实验步骤：

1. 访问心知天气的 API，获取 JSON 并提取所需的字段，申请个人的心知天气的 ID（https://www.seniverse.com/），通过 TCP client 包装 HTTP 请求协议去调用天气接口获取天气信息。

心知天气的 host：

```
https://www.seniverse.com/products?iid=006c072b-1ca9-42f2-98f1-bb6b1946aef8
```

本人心知天气的 APIKEY：

```
SBD3OTFNJDBK1ibtX
```

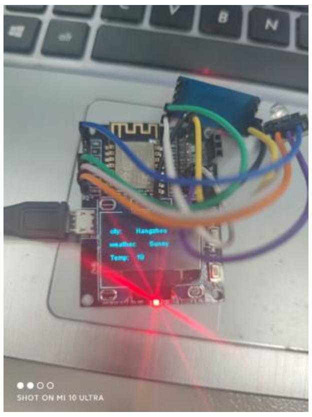
完整代码:

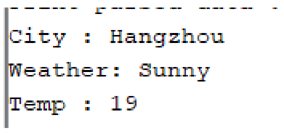
```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include"SSD1306Wire.h"
#include<dht11.h>
dht11 DHT11;
SSD1306Wire display(0x3c,2,14);
//以下三个定义为调试定义
#define DebugBegin(baud_rate)    Serial.begin(baud_rate)
#define DebugPrintln(message) Serial.println(message)
#define DebugPrint(message) Serial.print(message)

const char* ssid = "Mi 10000 Ultra";
const char* password = "88888888";
const char* host = "api.seniverse.com";
const char* APIKEY = "SBD3OTFNJDBK1ibtX";    //API KEY
const char* city = "hangzhou";
const char* language = "en";//zh-Hans   简体中文  会显示乱码

const unsigned long BAUD_RATE = 115200;
const unsigned long HTTP_TIMEOUT = 5000;
const size_t MAX_CONTENT_SIZE = 1000;


struct WeatherData {
  char city[16];//城市名称
char weather[32];//天气介绍（多云...）
char temp[16];//温度
char udate[32];//更新时间
};

WiFiClient client;
char response[MAX_CONTENT_SIZE];
char endOfHeaders[] = "\r\n\r\n";

void setup() {
    display.init();
// put your setup code here, to run once: WiFi.mode(WIFI_STA);    //设置 esp8266 工作模
式
DebugBegin(BAUD_RATE);
DebugPrint("Connecting to ");//提示
DebugPrintln(ssid);
WiFi.begin(ssid, password); //连接
  WiFi.setAutoConnect(true);
```

```
while (WiFi.status() != WL_CONNECTED) {
//这个函数是 wifi 连接状态，返回 wifi 链接状态
delay(500); DebugPrint(".");
}
DebugPrintln(""); DebugPrintln("WiFi connected"); delay(500);
DebugPrintln("IP address: "); DebugPrintln(WiFi.localIP());//WiFi.localIP()返回 8266 获得的
ip 地址
client.setTimeout(HTTP_TIMEOUT);
}

void loop() {
// put your main code here, to run repeatedly:
//判断 tcp client 是否处于连接状态，不是就建立连接
while (!client.connected()){
if (!client.connect(host, 80)){ DebugPrintln("connection...."); delay(500);
}
}
//发送 http 请求 并且跳过响应头 直接获取响应 body
if (sendRequest(host, city, APIKEY) && skipResponseHeaders()) {
//清除缓冲
clrEsp8266ResponseBuffer();
//读取响应数据
readReponseContent(response, sizeof(response)); WeatherData weatherData;
if (parseUserData(response, &weatherData)) { printUserData(&weatherData);
}

}
delay(5000);//每 5s 调用一次
}

/**
* @发送 http 请求指令
*/
bool sendRequest(const char* host, const char* cityid, const char* apiKey) {
// We now create a URI for the request
//心知天气  发送 http 请求
String GetUrl = "/v3/weather/now.json?key="; GetUrl += apiKey;
GetUrl += "&location="; GetUrl += city;
GetUrl += "&language="; GetUrl += language;
// This will send the request to the server
client.print(String("GET ") + GetUrl + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
DebugPrintln("create a request:"); DebugPrintln(String("GET ") + GetUrl + " HTTP/1.1\r\n"
+
"Host: " + host + "\r\n" + "Connection: close\r\n");
delay(1000); return true;
}

/**
* @Desc  跳过 HTTP  头，使我们在响应正文的开头
*/
bool skipResponseHeaders() {
// HTTP headers end with an empty line
bool ok = client.find(endOfHeaders);
if (!ok) {
DebugPrintln("No response or invalid response!");
}
return ok;
}

/**
* @Desc  从 HTTP 服务器响应中读取正文
*/
void readReponseContent(char* content, size_t maxSize) { size_t length =
client.readBytes(content, maxSize); delay(100);
DebugPrintln("Get the data from Internet!"); content[length] = 0;
DebugPrintln(content); DebugPrintln("Read data Over!"); client.flush();//清除一下缓冲
}

/**
  * @Desc  解析数据 Json 解析
  *  数据格式如下:
  * {
"results": [
*  {
* "location": {
* "id": "WX4FBXXFKE4F",
* "name": "北京",
* "country": "CN",
* "path": "北京,北京,中国",
* "timezone": "Asia/Shanghai",
* "timezone_offset": "+08:00"
* },
* "now": {
* "text": "多云",
* "code": "4",
* "temperature": "23"
* },
* "last_update":   "2017-09-13T09:51:00+08:00"
* }
  * ]
  *}
  */
bool parseUserData(char* content, struct WeatherData* weatherData) {
//   --   根据我们需要解析的数据来计算 JSON 缓冲区最佳大小
//   如果你使用 StaticJsonBuffer 时才需要
//   const size_t BUFFER_SIZE = 1024;
//   在堆栈上分配一个临时内存池
//   StaticJsonBuffer<BUFFER_SIZE>   jsonBuffer;
//   --   如果堆栈的内存池太大，使用 DynamicJsonBuffer jsonBuffer   代替
DynamicJsonBuffer jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(content);

if (!root.success()) { DebugPrintln("JSON parsing failed!"); return false;
}

//复制感兴趣的字符串
strcpy(weatherData->city, root["results"][0]["location"]["name"]);
strcpy(weatherData->weather,     root["results"][0]["now"]["text"]);
strcpy(weatherData->temp,     root["results"][0]["now"]["temperature"]);
strcpy(weatherData->udate, root["results"][0]["last_update"]);

//获取温度、最近更新时间

//        --   这不是强制复制，你可以使用指针，因为他们是指向“内容”缓冲区内，所以你需要确保
//   当你读取字符串时它仍在内存中
return true;
}

// 打印从 JSON 中提取的数据
void printUserData(const struct WeatherData* weatherData) {
 DebugPrintln("Print parsed data :");
DebugPrint("City : "); DebugPrint(weatherData->city);
DebugPrint("\nWeather: "); DebugPrint(weatherData->weather);
DebugPrint("\nTemp : "); DebugPrint(weatherData->temp);
DebugPrint("\nLast : "); DebugPrint(weatherData->udate);

  // put your main code here, to run repeatedly:
  display.flipScreenVertically();
  display.clear();
```

```
    display.drawString(0,10,"city:");
    display.drawString(40,10,weatherData->city);
    display.drawString(0,30,"weather:");
    display.drawString(60,30,weatherData->weather);
    display.drawString(0,50,"Temp:");
    display.drawString(40,50,weatherData->temp);
    display.display();
    delay(2000);

//打印天气、气温、最近更新时间

}
// 关闭与 HTTP 服务器连接
void stopConnect() { DebugPrintln("Disconnect"); client.stop();
}

void clrEsp8266ResponseBuffer(void){ memset(response, 0, MAX_CONTENT_SIZE);}
```

截图:





```
City : Hangzhou
Weather: Sunny
Temp : 19
```

抓包查看访问 API 的请求和响应

请求:

响应:

(选做：在 OLED 屏上显示天气信息)

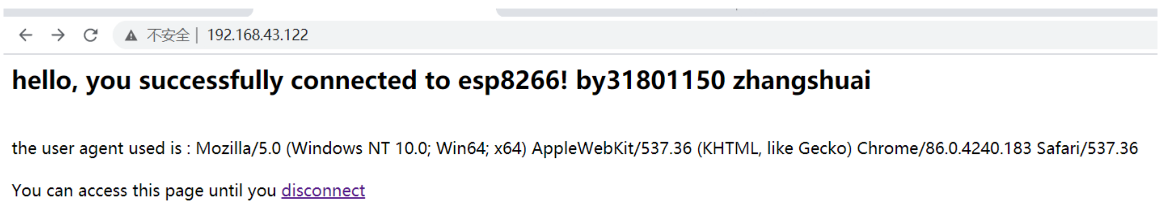
2.　实现 ESP8266 带高级登录认证（AdvancedAuth）的 Web Server；从 PC 端浏览器访问 ESP8266 的 WebServer，　登录后可以分别访问 GPIO 口的端口 0 和端口 1，获取各端口的状态 **(请在浏览器页面上显示本人学号信息)**。

完整代码:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include"SSD1306Wire.h"
#include<dht11.h>
dht11 DHT11;
SSD1306Wire display(0x3c,2,14);

#ifndef STASSID
#define STASSID "Mi 10000 Ultra"
#define STAPSK  "88888888"
#endif
const char* ssid = STASSID; const char* password = STAPSK; ESP8266WebServer
server(80);

//Check if header is present and correct
bool is_authenticated() {
Serial.println("Enter is_authenticated"); if (server.hasHeader("Cookie")) {
Serial.print("Found cookie: ");
String cookie = server.header("Cookie"); Serial.println(cookie);
if (cookie.indexOf("ESPSESSIONID=1") != -1) { Serial.println("Authentication Successful");
return true;
}
}
Serial.println("Authentication Failed"); return false;
}

//login page, also called for disconnect
void handleLogin() {
String msg;
if (server.hasHeader("Cookie")) { Serial.print("Found cookie: ");
String cookie = server.header("Cookie"); Serial.println(cookie);
}
if (server.hasArg("DISCONNECT")) { Serial.println("Disconnection");
server.sendHeader("Location", "/login"); server.sendHeader("Cache-Control", "no-cache");
server.sendHeader("Set-Cookie", "ESPSESSIONID=0"); server.send(301);
return;
}
if (server.hasArg("USERNAME") && server.hasArg("PASSWORD")) {
if (server.arg("USERNAME") == "admin" && server.arg("PASSWORD") == "admin")
{ server.sendHeader("Location", "/");
server.sendHeader("Cache-Control", "no-cache"); server.sendHeader("Set-Cookie", "ESPSESSIONID=1"); server.send(301);
Serial.println("Log in Successful"); return;
}
msg = "Wrong username/password! try again."; Serial.println("Log in Failed");
}
String content = "<html><body><form action='/login' method='POST'>To log in, please use : admin/admin<br>";
content += "User:<input type='text' name='USERNAME' placeholder='user name'><br>";
content += "Password:<input type='password' name='PASSWORD' placeholder='password'><br>";
content += "<input type='submit' name='SUBMIT' value='Submit'></form>" + msg + "<br>";
content += "You also can go <a href='/inline'>here</a></body></html>"; server.send(200, "text/html", content);
}

//root page can be accessed only if authentication is ok
void handleRoot() {
Serial.println("Enter handleRoot"); String header;
if (!is_authenticated()) { server.sendHeader("Location", "/login"); server.sendHeader("Cache-Control", "no-cache"); server.send(301);
return;
}
String content = "<html><body><H2>hello, you successfully connected to esp8266! by31801150 zhangshuai</H2><br>";
if (server.hasHeader("User-Agent")) {
content += "the user agent used is : " + server.header("User-Agent") + "<br><br>";
}
content += "You can access this page until you <a href=\"/login?DISCONNECT=YES\">disconnect</a></body></html>";
server.send(200, "text/html", content);
}



void handle0() {
Serial.println("Enter handleRoot"); String header;
if (!is_authenticated()) { server.sendHeader("Location", "/login"); server.sendHeader("Cache-Control", "no-cache"); server.send(301);
return;
}
```

```
String   content  =  "<html><body><H2>hello,  you  successfully   connected  to  gpio0! by31801150 zhangshuai</H2><br>";

server.send(200, "text/html", content);
}



void handle1() {
Serial.println("Enter handleRoot"); String header;
if         (!is_authenticated())         {         server.sendHeader("Location",         "/login"); server.sendHeader("Cache-Control", "no-cache"); server.send(301);
return;
}

String   content  =  "<html><body><H2>hello,  you  successfully   connected  to  gpio1! by31801150 zhangshuai</H2><br>";

server.send(200, "text/html", content);
}






//no need authentication
void handleNotFound() {
String message = "File Not Found\n\n"; message += "URI: ";
message += server.uri(); message += "\nMethod: ";
message   +=   (server.method()   ==   HTTP_GET)   ?   "GET"   :   "POST";   message   += "\nArguments: ";
message += server.args(); message += "\n";
for (uint8_t i = 0; i < server.args(); i++) {
message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
}
server.send(404, "text/plain", message);
}

void setup(void) { Serial.begin(115200); WiFi.mode(WIFI_STA); WiFi.begin(ssid, password); Serial.println("");
  display.init();
// Wait for connection
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}
Serial.println(""); Serial.print("Connected to "); Serial.println(ssid); Serial.print("IP address: ");
Serial.println(WiFi.localIP());
    display.flipScreenVertically();
    display.clear();
    display.drawString(0,10,(WiFi.localIP()).toString());
    display.display();
    delay(2000);



server.on("/", handleRoot);
server.on("/login", handleLogin);
server.on("/gpio/0", handle0);
server.on("/gpio/1", handle1);
server.on("/inline", []() {
server.send(200, "text/plain", "this works without need of authentication");
});

server.onNotFound(handleNotFound);
//here the list of headers to be recorded
const char * headerkeys[] = {"User-Agent", "Cookie"} ; size_t headerkeyssize =
sizeof(headerkeys) / sizeof(char*);
//ask server to track these headers
server.collectHeaders(headerkeys, headerkeyssize); server.begin();
Serial.println("HTTP server started");
}

void loop(void) { server.handleClient();

}
```

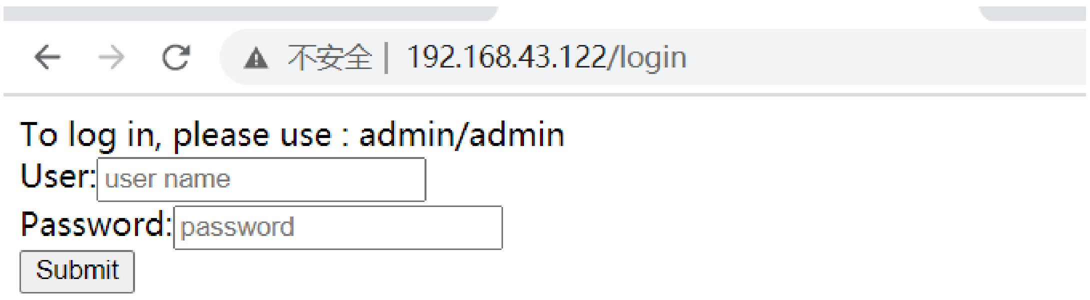
截图：



hello, you successfully connected to esp8266! by31801150 zhangshuai

the user agent used is : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36

You can access this page until you disconnect



不安全 | 192.168.43.122/login

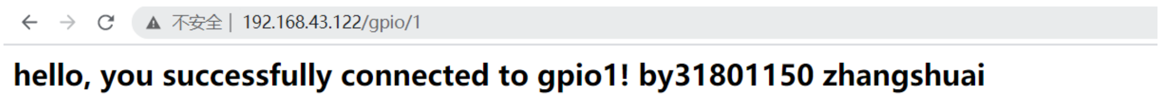
To log in, please use : admin/admin

User: user name

Password: password

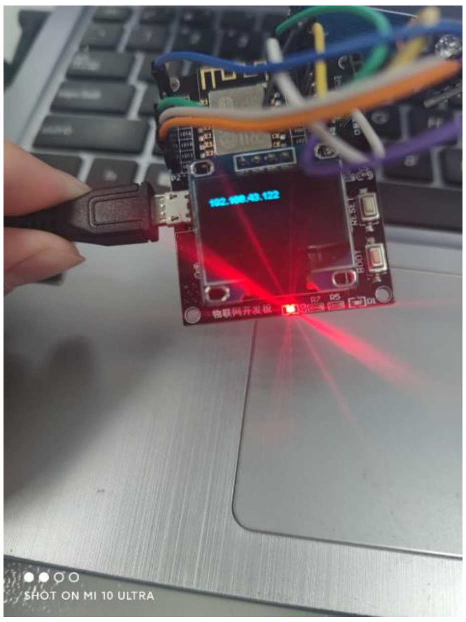
Submit

You also can go here



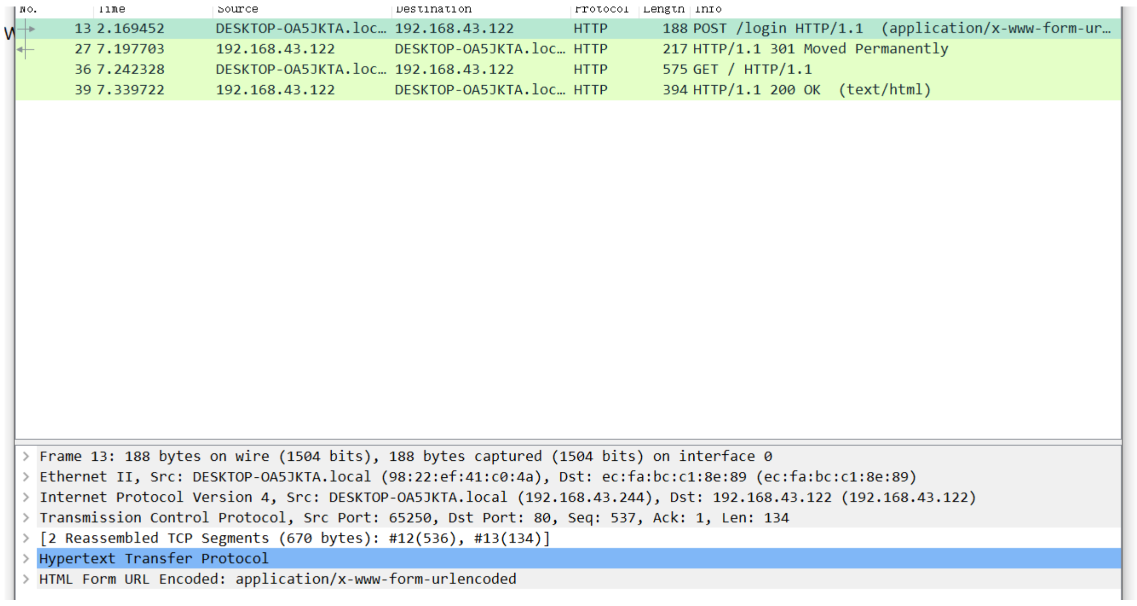
不安全 | 192.168.43.122/gpio/1

hello, you successfully connected to gpio1! by31801150 zhangshuai



抓包查看访问登录的请求和响应

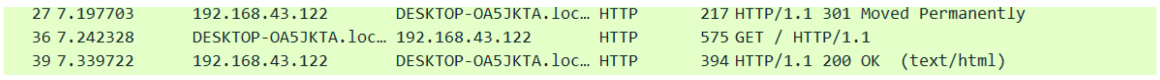
请求:



```
No.    Time       Source                   Destination              Protocol Length Info
    13 2.169452   DESKTOP-OA5JKTA.loc…     192.168.43.122           HTTP     188 POST /login HTTP/1.1  (application/x-www-form-ur…
    27 7.197703   192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     217 HTTP/1.1 301 Moved Permanently
    36 7.242328   DESKTOP-OA5JKTA.loc…     192.168.43.122           HTTP     575 GET / HTTP/1.1
    39 7.339722   192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     394 HTTP/1.1 200 OK  (text/html)

> Frame 13: 188 bytes on wire (1504 bits), 188 bytes captured (1504 bits) on interface 0
> Ethernet II, Src: DESKTOP-OA5JKTA.local (98:22:ef:41:c0:4a), Dst: ec:fa:bc:c1:8e:89 (ec:fa:bc:c1:8e:89)
> Internet Protocol Version 4, Src: DESKTOP-OA5JKTA.local (192.168.43.244), Dst: 192.168.43.122 (192.168.43.122)
> Transmission Control Protocol, Src Port: 65250, Dst Port: 80, Seq: 537, Ack: 1, Len: 134
> [2 Reassembled TCP Segments (670 bytes): #12(536), #13(134)]
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded
```

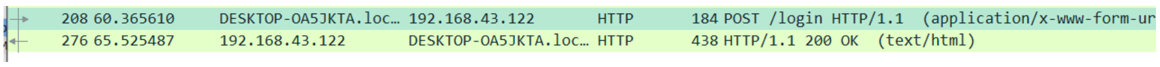
响应:

成功



```
    27 7.197703   192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     217 HTTP/1.1 301 Moved Permanently
    36 7.242328   DESKTOP-OA5JKTA.loc…     192.168.43.122           HTTP     575 GET / HTTP/1.1
    39 7.339722   192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     394 HTTP/1.1 200 OK  (text/html)
```

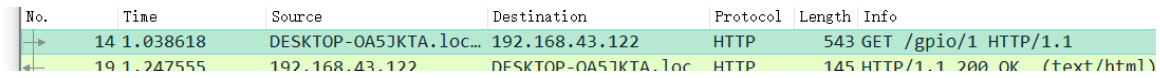
失败



```
   208 60.365610  DESKTOP-OA5JKTA.loc…     192.168.43.122           HTTP     184 POST /login HTTP/1.1  (application/x-www-form-ur
   276 65.525487  192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     438 HTTP/1.1 200 OK  (text/html)
```

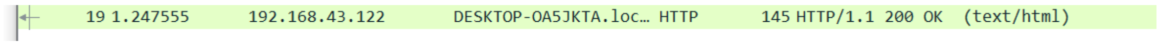
抓包查看访问访问端口 1 的请求和响应

请求:



```
No.    Time       Source                   Destination              Protocol Length Info
    14 1.038618   DESKTOP-OA5JKTA.loc…     192.168.43.122           HTTP     543 GET /gpio/1 HTTP/1.1
    19 1.247555   192.168.43.122           DESKTOP-OA5JKTA.loc      HTTP     145 HTTP/1.1 200 OK  (text/html)
```

响应:



```
    19 1.247555   192.168.43.122           DESKTOP-OA5JKTA.loc…     HTTP     145 HTTP/1.1 200 OK  (text/html)
```

（选做：在 OLED 屏上显示登录的主机的 IP 地址）