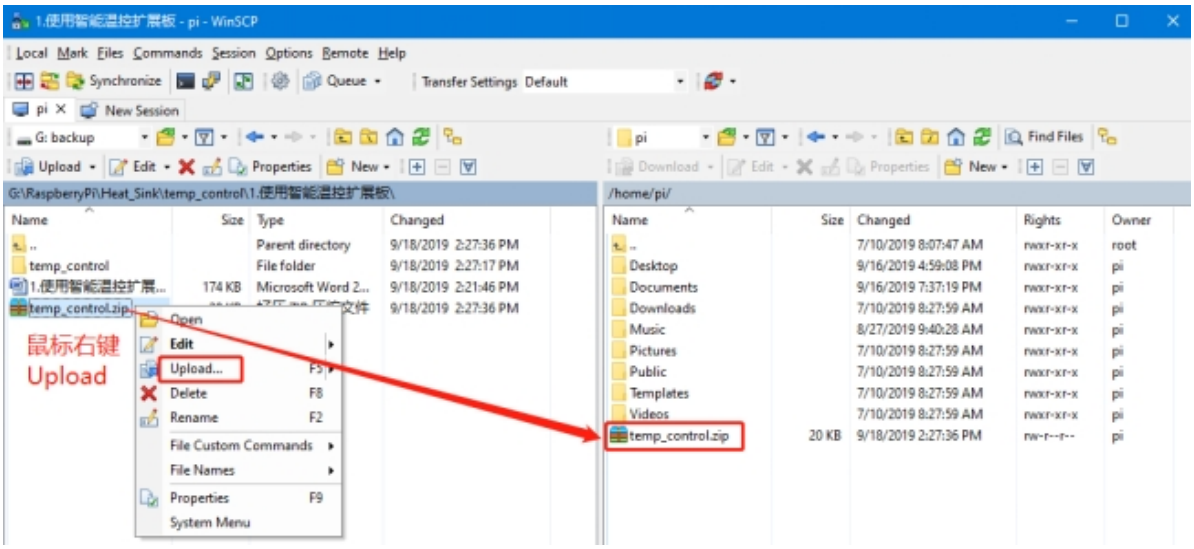
4.Oled display

RGB cooling HAT needs to be correctly inserted into the GPIO port of the RDK X3, and the I2C function of the RDK X3 needs to be turned on.

The phenomenon of this experiment is that the OLED displays the RDK X3 CPU usage, CPU temperature, running memory usage, disk usage, IP address and other information.

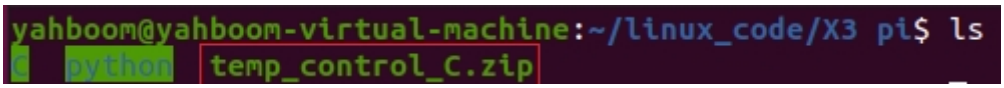
1.File transfer

1.Install WinSCP on your computer, connect to RDK X3, and transfer the **temp_control_C.zip** package downloaded from the documentation to the pi directory of the RDK X3.



2.Extract file

Open the terminal in RDK X3 system and find **temp_control_C.zip** file.



Enter the following command to extract the file.

```
unzip temp_control.zip
```

```
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi$ unzip temp_control_C.zip
Archive:  temp_control_C.zip
   creating: temp_control_C/i2c_fan/
  inflating: temp_control_C/i2c_fan/fan
  inflating: temp_control_C/i2c_fan/fan_i2c_dan(┤e├╢└┴┘).c
  inflating: temp_control_C/i2c_fan/i2c_fan.c
  inflating: temp_control_C/i2c_fan/myi2c.c
  inflating: temp_control_C/i2c_fan/myi2c.h
   creating: temp_control_C/i2c_OLED/
  inflating: temp_control_C/i2c_OLED/myi2c.c
  inflating: temp_control_C/i2c_OLED/myi2c.h
  inflating: temp_control_C/i2c_OLED/oled
  inflating: temp_control_C/i2c_OLED/oled.c
  inflating: temp_control_C/i2c_OLED/oled_fonts.h
  inflating: temp_control_C/i2c_OLED/ssd1306_i2c.c
  inflating: temp_control_C/i2c_OLED/ssd1306_i2c.h
   creating: temp_control_C/i2c_RGB/
  inflating: temp_control_C/i2c_RGB/I2C_RGB.c
  inflating: temp_control_C/i2c_RGB/myi2c.c
  inflating: temp_control_C/i2c_RGB/myi2c.h
  inflating: temp_control_C/i2c_RGB/RGB
   creating: temp_control_C/temp_control/
  inflating: temp_control_C/temp_control/install.sh
  inflating: temp_control_C/temp_control/myi2c.c
  inflating: temp_control_C/temp_control/myi2c.h
  inflating: temp_control_C/temp_control/oled_fonts.h
 extracting: temp_control_C/temp_control/readme.txt
  inflating: temp_control_C/temp_control/rgb_temp.c
  inflating: temp_control_C/temp_control/rgb_temp.h
  inflating: temp_control_C/temp_control/ssd1306_i2c.c
  inflating: temp_control_C/temp_control/ssd1306_i2c.h
  inflating: temp_control_C/temp_control/start.sh
  inflating: temp_control_C/temp_control/temp_control
  inflating: temp_control_C/temp_control/temp_control.c
  inflating: temp_control_C/temp_control/temp_control.h
```

2. Compiling and running the program

1.Enter the folder and view the files in the current folder

```
cd temp_control_c/i2c_OLED
ls
```

```
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi$ cd temp_control_c/i2c_OLED
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi/temp_control_c/i2c_OLED$ ls
myi2c.c  myi2c.h  oled.c  oled_fonts.h  ssd1306_i2c.c  ssd1306_i2c.h
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi/temp_control_c/i2c_OLED$
```

2.Compile program files

```
gcc -o oled oled.c ssd1306_i2c.c myi2c.c
```

```
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi/temp_control_c/i2c_OLED$ ls
myi2c.c  myi2c.h  oled  oled.c  oled_fonts.h  ssd1306_i2c.c  ssd1306_i2c.h
```

Among them, the gcc compiler is called, -o means to generate files, followed by the generated file name, oled.c and ssd1306_i2c.c are source programs, and myi2c.c is the source code for the device on the X3 pie driver I2C bus.

3.Run program

```
sudo ./oled
```

```
yahboom@yahboom-virtual-machine:~/linux_code/X3 pi/temp_control_c/i2c_OLED$ sudo ./oled
```

At this point, we can see the RDK X3 CPU usage, CPU temperature, running memory usage, disk usage, IP address and other information displayed on the OLED screen.

3.About code

1.Import I2C library, oled display library, file control library, read IP library and read disk library.

```c
// 导入oled显示屏库
#include "ssd1306_i2c.h"
#include "myi2c.h"
#include "oled_fonts.h"

// 导入文件控制函数库
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/sysinfo.h>
// 读取IP库
#include <ifaddrs.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
// 读取磁盘库
#include <sys/vfs.h>
#include <unistd.h>

#define CPU_USAGE_PATH "/proc/stat"
#define TEMP_PATH "/sys/class/thermal/thermal_zone0/temp"
#define MAX_SIZE 32
```

2.Define temperature, CPU usage, system information, disk information, IP and other related parameters.

```
int fd_temp;
double temp = 0;
char buf[MAX_SIZE];

char buf_cpu[128];
FILE *fp_CPUusage;
char cpu[5];
float usage;
long int user, nice, sys, idle, iowait, irq, softirq;
long int total_1, total_2, idle_1, idle_2;

// get system usage / info
struct sysinfo sys_info;
struct statfs disk_info;

struct ifaddrs *ifAddrStruct = NULL;
void *tmpAddrPtr = NULL;
getifaddrs(&ifAddrStruct);
```

3.Initialize the oled display and output the initialization success information from the terminal.

```
ssd1306_begin(SSD1306_SWITCHCAPVCC, SSD1306_I2C_ADDRESS);
// ssd1306_display();      //show logo
// ssd1306_clearDisplay();
// delay(500);
printf("init ok!\n");
```

4.Read system information. If it fails, sysinfo-Error will be displayed on the oled, and wait for 0.5 seconds to read again.

```
// 读取系统信息
if (sysinfo(&sys_info) != 0) // sysinfo(&sys_info) != 0
{
    printf("sysinfo-Error\n");
    ssd1306_clearDisplay();
    char *text = "sysinfo-Error";
    ssd1306_drawString(text);
    ssd1306_display();
    delay(500);
    continue;
}
```

5.Read CPU usage.

The first file you need to open is /proc/stat, which stores CPU activity information. All values in this file are accumulated from system startup to the current moment.

Enter cat /proc/stat in the terminal to check CPU activity data.

To calculate the CPU usage, we only need to use the top row of data. Below I will only explain the meaning of this row of data.

(jiffies is a global variable in the kernel, which is used to record the number of beats generated since the system started. In Linux, a beat can be roughly understood as the minimum time slice for operating system process scheduling. Different Linux kernels may have different values. We can assume that: 1 jiffies = 10ms)

Parameter	**Analysis (unit: jiffies)**
user(969)	The running time in user mode accumulated from system startup to the current time, excluding processes with negative nice values.
nice(0)	The CPU time occupied by processes with negative nice values since system startup until the current time
system(1557)	The running time in the kernel state from the system startup to the current time
idle(20390)	Waiting time other than IO waiting time accumulated from system startup to the current time
iowait(623)	IO waiting time accumulated from system startup to the current time
irq(0)	Hard interrupt time accumulated from system startup to the current time
softirq(10)	Soft interrupt time accumulated from system startup to the current time
stealstolen(0)	When running in a virtual environment, how much time is spent in other operating systems
guest(0)	How much time does the guest OS's virtual CPU spend running under Linux kernel control?

The calculation formula for total CPU time (accumulated value):

totalTime = user+nice+system+idle+iowait+irq+softirq+stealstolen+guest

We need to calculate the current CPU usage by reading the parameters above. Since it is accumulated from the system startup to the current moment, the difference between the two parameters collected at short intervals (1 second) can be used to calculate the total CPU time, and then the idle time is calculated in the same way. Finally, the CPU usage = 100*(total-idle)/total.

In addition: In fact, you can also directly enter the command in the X3 terminal to view the current CPU usage. The main input of the following code can display the CPU usage:

cat <(grep 'cpu ' /proc/stat) <(sleep 1 && grep 'cpu ' /proc/stat) | awk -v RS="" '{print ($13-$2+$15-$4)*100/($13-$2+$15-$4+$16-$5) "%"}'

```
pi@raspberrypi:~/Documents $ cat <(grep 'cpu ' /proc/stat) <(sleep 1 && grep 'cpu ' /proc/stat) | awk -v RS
="" '{print ($13-$2+$15-$4)*100/($13-$2+$15-$4+$16-$5) "%"}'
0.740741%
pi@raspberrypi:~/Documents $
```

About code:

```
// CPU占用率
char CPUInfo[MAX_SIZE];
//unsigned long avgCpuLoad = sys_info.loads[0] / 1000;
//sprintf(CPUInfo, "CPU:%ld%%", avgCpuLoad);
fp_CPUusage = fopen(CPU_USAGE_PATH, "r");
if (fp_CPUusage == NULL)
{
    printf("failed to open /proc/stat\n");
}
else
{
    /* CPU使用率计算方式: usage = 100*(total-idle/total)
     * 通过两次获取/proc/stat的数据差得到total和idle
    */
    // 第一次读取数据
    fgets(buf_cpu, sizeof(buf_cpu), fp_CPUusage);
    sscanf(buf_cpu,"%s%d%d%d%d%d%d%d",cpu,&user,&nice,&sys,&idle,&iowait,&irq,&softirq);
    total_1 = user + nice + sys + idle + iowait + irq + softirq;
    idle_1 = idle;
    rewind(fp_CPUusage);

    // 延时并且清空数据
    sleep(1);
    memset(buf_cpu, 0, sizeof(buf_cpu));
    cpu[0] = '\0';
    user = nice = sys = idle = iowait = softirq = 0;

    // 第二次读取数据
    fgets(buf_cpu, sizeof(buf_cpu), fp_CPUusage);
    sscanf(buf_cpu,"%s%d%d%d%d%d%d%d",cpu,&user,&nice,&sys,&idle,&iowait,&irq,&softirq);
    total_2 = user + nice + sys + idle + iowait + irq + softirq;
    idle_2 = idle;

    usage = (float)(total_2-total_1-(idle_2-idle_1)) / (total_2-total_1)*100;
    sprintf(CPUInfo, "CPU:%.0f%%", usage);
    //printf("cpu:%.0f%%\n", usage);
    fclose(fp_CPUusage);
}
```

First, open the /proc/stat file and read the data with the fgets function. The sscanf function saves the read parameters into the corresponding variables, and then calculates the total_1 and idle_1 read for the first time; then delays for 1 second and clears the variable data. It has been tested that the read data will be invalid if the time is less than 1 second; the second read data is saved in total_2 and idle_2; finally, the CPU usage value is calculated and stored in CPUInfo.

6.Read the running memory usage. The data read out is in bytes. For the convenience of display, it needs to be converted into Mb. The value can be completed by shifting right 20 bits.

It can also be written as follows: unsigned long totalRam = sys_info.totalram/1024 /1024;

```
// 运行内存占用率，剩余/总内存
char RamInfo[MAX_SIZE];
unsigned long totalRam = sys_info.totalram >> 20;
unsigned long freeRam = sys_info.freeram >> 20;
sprintf(RamInfo, "RAM:%ld/%ld MB", freeRam, totalRam);
```

7.Read the IP address, you can display the IP addresses of the network cable and WiFi network, and give priority to displaying the IP address of the network cable.

```
// 获取IP地址
char IPInfo[MAX_SIZE];
while (ifAddrStruct != NULL)
{
    if (ifAddrStruct->ifa_addr->sa_family == AF_INET)
    { // check it is IP4 is a valid IP4 Address
        tmpAddrPtr = &((struct sockaddr_in *)ifAddrStruct->ifa_addr)->sin_addr;
        char addressBuffer[INET_ADDRSTRLEN];
        inet_ntop(AF_INET, tmpAddrPtr, addressBuffer, INET_ADDRSTRLEN);

        if (strcmp(ifAddrStruct->ifa_name, "eth0") == 0)
        {
            sprintf(IPInfo, "eth0:IP:%s", addressBuffer);
            break;
        }
        else if (strcmp(ifAddrStruct->ifa_name, "wlan0") == 0)
        {
            sprintf(IPInfo, "wlan0:%s", addressBuffer);
            break;
        }
    }
    ifAddrStruct = ifAddrStruct->ifa_next;
}
```

8.Read temperature.

```
// 读取CPU温度
char CPUTemp[MAX_SIZE];
fd_temp = open(TEMP_PATH, O_RDONLY);
if (fd_temp < 0)
{
    temp = 0;
    printf("failed to open thermal_zone0/temp\n");
}
else
{
    // 读取温度并判断
    if (read(fd_temp, buf, MAX_SIZE) < 0)
    {
        temp = 0;
        printf("fail to read temp\n");
    }
    else
    {
        // 转换为浮点数打印
        temp = atoi(buf) / 1000.0;
        // printf("temp: %.1f\n", temp);
        sprintf(CPUTemp, "Temp:%.1fC", temp);
    }
}
close(fd_temp);
```

9.Read disk space

```
// 读取磁盘空间，剩余/总空间
char DiskInfo[MAX_SIZE];
statfs("/", &disk_info);
unsigned long long totalBlocks = disk_info.f_bsize;
unsigned long long totalSize = totalBlocks * disk_info.f_blocks;
size_t mbTotalsize = totalSize >> 20;
unsigned long long freeDisk = disk_info.f_bfree * totalBlocks;
size_t mbFreedisk = freeDisk >> 20;
sprintf(DiskInfo, "Disk:%ld/%ldMB", mbFreedisk, mbTotalsize);
```

10.Set the content to be displayed on the oled

The ssd1306_drawText(int x, int y, char *str) function is used to set the content to be displayed on the oled. The first parameter is x, which controls the left and right offsets, the second parameter is y, which controls the up and down offsets, and the third is the string pointer, which is the content to be displayed.

Finally, the ssd1306_display() function must be run to refresh the display.

```
// 在显示屏上要显示的内容
ssd1306_drawText(0, 0, CPUInfo);
ssd1306_drawText(56, 0, CPUTemp);
ssd1306_drawText(0, 8, RamInfo);
ssd1306_drawText(0, 16, DiskInfo);
ssd1306_drawText(0, 24, IPInfo);

// 刷新显示
ssd1306_display();
```