微算機系統實習

期末專題

日期:2021/06/18

1. 組員姓名：

第一組

資工二　108590044　何柏憲

資工二　108590452　林峻霆

1. 實驗步驟截圖與說明：

實驗目標:這次分兩個部分

* QT UI 介面設計和控制：使用pthread並指定Mutex方式控制LED閃爍
* Nodejs設計和控制：使用pthread並指定Mutex方式控制LED閃爍

實驗過程:

1.首先我們先參考範例了解此次實驗要做什麼，發現是延伸lab2 到lab6 中的所有的實驗結合，說真的講到這裡我覺得很棒ㄟ，可以複習先前的。

2.因為這次也是遠距教學，第一週我們組員有見面一起討論此實驗有先完成nodejs的部分了，ＱＴ的程式碼已完成但編譯上出了點問題，第二週則是線上meet ，也解決的ＱＴ上不能編譯的問題ㄌ。

3.第二週我們重寫了ＱＴ的程式碼，也重新安裝了後來才成功了。

4. 讀取參數並執行

5.錄影並執行課程之要求

以下是我們的程式碼：

ＱＴ：

Gpio.h:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <iostream>

using namespace std;

int gpio\_export(unsigned int gpio){

int fd,len;

char buff[64];

fd = open("/sys/class/gpio/export",O\_WRONLY);

if (fd<0){

perror("gpio/export");

return fd;

}

len = snprintf(buff,sizeof(buff),"%d",gpio);

write(fd,buff,len);

close(fd);

return 0;

}

int gpio\_unexport(unsigned int gpio){

int fd,len;

char buff[64];

fd = open("/sys/class/gpio/unexport",O\_WRONLY);

if (fd<0){

perror("gpio/unexport");

return fd;

}

len = snprintf(buff,sizeof(buff),"%d",gpio);

write(fd,buff,len);

close(fd);

return 0;

}

int gpio\_set\_dir(unsigned int gpio,string dirStatus){

int fd;

char buff[64];

snprintf(buff,sizeof(buff),"/sys/class/gpio/gpio%d/direction",gpio);

fd = open(buff,O\_WRONLY);

if(fd<0){

perror("gpio/direction");

return fd;

}

if(dirStatus == "out"){

write(fd,"out",4);

}else{

write(fd,"in",3);

}

close(fd);

return 0;

}

int gpio\_set\_value(unsigned int gpio,int value){

int fd;

char buff[64];

snprintf(buff,sizeof(buff),"/sys/class/gpio/gpio%d/value",gpio);

fd=open(buff,O\_WRONLY);

if(fd<0){

perror("gpio/set-value");

return fd;

}

if(value == 0 ){

write(fd,"0",2);

}else{

write(fd,"1",2);

}

close(fd);

return 0;

}

Main.cpp

#include "mainwindow.h"

#include <QApplication>

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

#include<errno.h>

#include<unistd.h>

#include<fcntl.h>

#include<iostream>

#include <pthread.h>

#include <semaphore.h>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QTimer>

#include <QDebug>

#include <string>

#include <pthread.h>

#include <semaphore.h>

#include "gpio.h"

#define LED1\_PIN 396

#define LED2\_PIN 397

#define LED3\_PIN 255

#define LED4\_PIN 481

using namespace std;

string status = "";

int times;

int second;

sem\_t semaphore;

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->counter->setText("00:00");

second =0;

}

MainWindow::~MainWindow()

{

delete ui;

}

void led(int pin,int status){

gpio\_export(pin);

gpio\_set\_dir(pin,"out");

if(status==1){

gpio\_set\_value(pin,1);

}else{

gpio\_set\_value(pin,0);

gpio\_unexport(pin);

}

}

//----------------------------------------------

void\* led1\_thread(void\* arg){

sem\_wait(&semaphore);

for(int i=0;i<times;i++){

led(LED1\_PIN,status[0]-'0');

sleep(1);

led(LED1\_PIN,1-(status[0]-'0'));

sleep(1);

}

pthread\_exit(NULL);

}

void\* led2\_thread(void\* arg){

sem\_wait(&semaphore);

for(int i=0;i<times;i++){

led(LED2\_PIN,status[1]-'0');

sleep(1);

led(LED2\_PIN,1-(status[1]-'0'));

sleep(1);

}

pthread\_exit(NULL);

}

void\* led3\_thread(void\* arg){

//string = (char \*)input;

sem\_wait(&semaphore);

for(int i=0;i<times;i++){

led(LED3\_PIN,status[2]-'0');

sleep(1);

led(LED3\_PIN,1-(status[2]-'0'));

sleep(1);

}

pthread\_exit(NULL);

}

void\* led4\_thread(void\* arg){

sem\_wait(&semaphore);

for(int i=0;i<times;i++){

led(LED4\_PIN,status[3]-'0');

sleep(1);

led(LED4\_PIN,1-(status[3]-'0'));

sleep(1);

}

pthread\_exit(NULL);

}

void\* countTime(void\* arg){

sem\_wait(&semaphore);

QLabel\* count = (QLabel\*)arg;

for(int i=0;i<times\*2;i++){

int min , sec;

sleep(1);

second++;

sec = second % 60;

min = (second - sec)/60;

QString str\_s,str\_m;

if(sec<10){

str\_s = QString::number(0)+QString::number(sec);

}

else{

str\_s = QString::number(sec);

}

if(min<10){

str\_m = QString::number(0)+QString::number(min);

}

else{

str\_m = QString::number(min);

}

count->setText(str\_m+":"+str\_s);

}

pthread\_exit(NULL);

}

void MainWindow::on\_LED\_Shining\_clicked()

{

if(ui->LED1->isChecked())

status[0] = '1';

else

status[0] = '0';

if(ui->LED2->isChecked())

status[1] = '1';

else

status[1] = '0';

if(ui->LED3->isChecked())

status[2] = '1';

else

status[2] = '0';

if(ui->LED4->isChecked())

status[3] = '1';

else

status[3] = '0';

times = ui->times->text().toInt();

sem\_init(&semaphore,0,0);

QLabel\* count = ui->counter;

pthread\_t led1,led2,led3,led4,count\_time;

pthread\_create(&led1,NULL,led1\_thread,NULL);

pthread\_create(&led2,NULL,led2\_thread,NULL);

pthread\_create(&led3,NULL,led3\_thread,NULL);

pthread\_create(&led4,NULL,led4\_thread,NULL);

pthread\_create(&count\_time,NULL,countTime,(void\*)count);

for(int i=0;i<times\*5;i++)

{

sem\_post(&semaphore);

}

}

Mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QLabel>

#include <QCheckBox>

#include <QKeyEvent>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private slots:

void on\_LED\_Shining\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

Mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>496</width>

<height>450</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralWidget">

<widget class="QCheckBox" name="LED1">

<property name="geometry">

<rect>

<x>50</x>

<y>80</y>

<width>92</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>LED1</string>

</property>

<property name="shortcut">

<string>1</string>

</property>

</widget>

<widget class="QCheckBox" name="LED2">

<property name="geometry">

<rect>

<x>50</x>

<y>110</y>

<width>92</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>LED2</string>

</property>

<property name="shortcut">

<string>2</string>

</property>

</widget>

<widget class="QCheckBox" name="LED3">

<property name="geometry">

<rect>

<x>50</x>

<y>140</y>

<width>92</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>LED3</string>

</property>

<property name="shortcut">

<string>3</string>

</property>

</widget>

<widget class="QCheckBox" name="LED4">

<property name="geometry">

<rect>

<x>50</x>

<y>170</y>

<width>92</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>LED4</string>

</property>

<property name="shortcut">

<string>4</string>

</property>

</widget>

<widget class="QLabel" name="Times">

<property name="geometry">

<rect>

<x>180</x>

<y>180</y>

<width>67</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Count:</string>

</property>

</widget>

<widget class="QPushButton" name="LED\_Shining">

<property name="geometry">

<rect>

<x>260</x>

<y>230</y>

<width>89</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Semaphore</string>

</property>

<property name="shortcut">

<string>Return</string>

</property>

</widget>

<widget class="QLineEdit" name="times">

<property name="geometry">

<rect>

<x>250</x>

<y>180</y>

<width>113</width>

<height>25</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>70</x>

<y>40</y>

<width>91</width>

<height>16</height>

</rect>

</property>

<property name="text">

<string>LED</string>

</property>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>260</x>

<y>90</y>

<width>141</width>

<height>16</height>

</rect>

</property>

<property name="text">

<string>Process Time</string>

</property>

</widget>

<widget class="QLabel" name="counter">

<property name="geometry">

<rect>

<x>270</x>

<y>120</y>

<width>67</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>TextLabel</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>496</width>

<height>22</height>

</rect>

</property>

</widget>

<widget class="QToolBar" name="mainToolBar">

<attribute name="toolBarArea">

<enum>TopToolBarArea</enum>

</attribute>

<attribute name="toolBarBreak">

<bool>false</bool>

</attribute>

</widget>

<widget class="QStatusBar" name="statusBar"/>

</widget>

<layoutdefault spacing="6" margin="11"/>

<resources/>

<connections/>

</ui>

Nodejs:

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name ="viewport" content="width=device-width, initial-scale=1.0">

<title>Nodejs</title>

</head>

<body>

<p>

<h1>Nodejs</h1>

</p>

<hr>

<form action="/index">

<h1>LED Control Panel</h1>

<input type="checkbox" id="LED1" name="LED1" value="true" >

<label for="LED1"> LED1</label><br>

<input type="checkbox" id="LED2" name="LED2" value="true">

<label for="LED2"> LED2</label><br>

<input type="checkbox" id="LED3" name="LED3" value="true">

<label for="LED3"> LED3</label><br>

<input type="checkbox" id="LED4" name="LED4" value="true">

<label for="LED4"> LED4</label><br>

<!-- <input type="submit" value="submit"> -->

<!-- <input type="radio" id="on" name="Switch" value="on">

<label for="on">On</label>

<input type="radio" id="off" name="Switch" value="off">

<label for="off">Off</label> -->

</form>

<hr>

<form action="/index1" method="get">

<h1>Count</h1>

<label><input type="text" name="frequency"></label><br>

<input type="submit" value="Mutex">

</form>

</body>

</html>

Main

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <semaphore.h>

#include "main.h"

pthread\_mutex\_t mutex;

int counter=0;

int pin[] = {396,397,255,481};

int LEDStatus[4]={0};

void turnOn(unsigned int target){

char s[] = "out";

gpio\_export(target);

gpio\_set\_dir(target,"out");

gpio\_set\_value(target,1);

}

void turnOff(unsigned int target){

gpio\_set\_value(target,0);

gpio\_unexport(target);

}

void \*child(void \*arg){

pthread\_mutex\_lock(&mutex);

for(int j = 0 ; j < 4 ; j++){

if(LEDStatus[j] == 1){

turnOn(pin[j]);

}else{

turnOff(pin[j]);

}

}

sleep(1);

for(int j = 0 ; j < 4 ; j++){

if(LEDStatus[j] == 0){

turnOn(pin[j]);

}else{

turnOff(pin[j]);

}

}

sleep(1);

pthread\_mutex\_unlock(&mutex);;

pthread\_exit(NULL);

}

int main(int argc,char \*argv[]){

int times = atoi(argv[2]);

printf("times: %d\n",times);

for(int i = 0 ; i < 4;i++){

LEDStatus[i]= argv[1][i]-'0';

turnOn(pin[i]);

turnOff(pin[i]);

}

pthread\_mutex\_init(&mutex,0);

pthread\_t t[times];

for(int i = 0 ; i < times ;i++){

printf("thread[%d] create\n",i);

pthread\_create(&t[i],NULL,child,(void\*)times);

}

for(int i = 0 ; i < times;i++){

pthread\_join(t[i],NULL);

}

return 0;

}

Makefile

semaphore:

gcc main.c -lpthread -o main

sudo ./main 0111 5

mutex:

gcc main2.c -lpthread -o main2

sudo ./main2 1011 3

mutex.h

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <semaphore.h>

#include "main.h"

pthread\_mutex\_t mutex;

int counter=0;

int pin[] = {396,397,255,481};

int LEDStatus[4]={0};

void turnOn(unsigned int target){

char s[] = "out";

gpio\_export(target);

gpio\_set\_dir(target,"out");

gpio\_set\_value(target,1);

}

void turnOff(unsigned int target){

gpio\_set\_value(target,0);

gpio\_unexport(target);

}

void \*child(void \*arg){

pthread\_mutex\_lock(&mutex);

for(int j = 0 ; j < 4 ; j++){

if(LEDStatus[j] == 1){

turnOn(pin[j]);

}else{

turnOff(pin[j]);

}

}

sleep(1);

for(int j = 0 ; j < 4 ; j++){

if(LEDStatus[j] == 0){

turnOn(pin[j]);

}else{

turnOff(pin[j]);

}

}

sleep(1);

pthread\_mutex\_unlock(&mutex);;

pthread\_exit(NULL);

}

Server.js

const express = require("express")

const app = express()

app.use(express.static("../finalNode"))

// app.get("/index",(req,res)=>{

// let q =req.query;

// let response = {

// "LED1" :[q.LED1,q.LED2,q.LED3,q.LED4],

// "on&off" : q.Switch,

// }

// response.LED1.forEach((e,i) => {

// if(e == "true")

// controlLED(i+1,q.Switch);

// });

// res.send(response);

// });

app.get("/index1",(req,res)=>{

let q =req.query;

shine(parseInt(q.frequency));

res.status(200).send("good");

});

// let controlLED=(LED,Switch)=>{

// let child\_process =require("child\_process");

// let process = child\_process.execFile('sudo',["../lab02/gpio",LED,Switch]);

// console.log('sudo',["../lab02/gpio",LED,Switch]);

// process.stdout.on('data',(data)=>{

// console.log(`stdout: ${data}`);

// });

// process.stderr.on('data',(data)=>{

// console.error(`stdout: ${data}`);

// });

// }

let shine=(times)=>{

let child\_process =require("child\_process");

let process = child\_process.execFile('make',["mutex"]);

// exec(times);

//let p = child\_process.execFile(times);

process.stdout.on('data',(data)=>{

console.log(`stdout: ${data}`);

});

process.stderr.on('data',(data)=>{

console.error(`stdout: ${data}`);

});

}

// set port,listen for requests

const PORT = process.env.PORT || 8080;

app.listen(PORT,()=>{

console.log(`Server is running on port ${PORT}.`);

});

以下是實驗結果

QT: <https://www.youtube.com/watch?v=l9aPSUuUTxk>

Nodejs: <https://www.youtube.com/watch?v=k6JPfCWEuvE>

1. 組員貢獻比例：

何柏憲：50%

林峻霆：50%

1. 心得：

何柏憲：

　　 這次實驗跟之前的內容差不多，把所有程式碼整合起來就和撰寫新的QT，和js檔，不過這次是連續第三次要在家裡自己做，少了助教的當場協助，做起來也花了更多的時間，第一週去組員來到我家一起做實驗，我們先完成了第二部分，其實這部份相當簡單，第二週再延續前一週的進度完成第一部分，最後很順利的完成最後的目的，這學期也就這樣結束了，非常感謝助教和教授。

林峻霆：

這次實驗延續了這學期( lab2 到 lab6 )的實驗內容，我覺得遠距的微算機實習真的頗難的，第一週去組員家的和撰寫新的QT、和js檔，js檔案書寫上我們很快就完成了，後來一直卡在QT的部分，我們編譯一直過不了，第二週我們重寫了ＱＴ的程式碼，也重新安裝了後來才成功了。這學期也就這樣結束了，非常感謝助教和教授。