

《嵌入式系统》

(第三次实验-1 电子钟、小键盘控制的电子钟实验)

厦门大学信息学院软件工程系 曾文华

2023年10月24日

目录

一、数码管应用实验1：电子钟

二、数码管应用实验2：小键盘控制的电子钟

一、数码管应用实验1： 电子钟

电子钟程序的功能（clock.c）

- 要求：在8个数码管上显示时间：
XX-XX-XX（分别对应：时-分-秒）
- 该程序的执行需要带参数（初始时间）。
- 在实验箱的“超级终端（Xshell 4）”上执行：
 - **cd /mnt/whzeng/clock**
 - **./clock 23:59:50**
- 则数码管上显示：**23-59-50**，并开始变化时间。
- 如果输入的参数不对（初始时间不对），则显示“input error!”，并退出程序：
 - **./clock 24:59:50**
 - **./clock 23:60:50**
 - **./clock 23:59:60**



电子钟程序编写的思路

- 使用多线程函数实现计时（**计时线程**）：hour、minute、second
 - second从0变到59；second变到60时，将其置为0，并将minute加1
 - minute从0变到59；minute变到60时，将其置为0，并将hour加1
 - hour从0变到23；hour变到24时，将其置为0
 - 通过usleep函数实现延时1秒的功能

- 主函数（main函数）主要完成（在实验二数码管实验**leddisplay.c**程序基础上修改）：

- 判断输入的参数（初始时间）是不是正确？（**用一个函数实现**）
- 在数码管上显示时间：

```
for(i=0; i<8; i++)
{
    *(cpld+(0xe6<<1)) = addr[i];           //数码管的位地址（即哪一个数码管）

    number = leddisplay[i];

    *(cpld+(0xe4<<1)) = tube[number];       //数码管的段值（即显示什么内容）

    usleep(1000);
}
```

```
addr[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
```

第1位 第2位 第3位 第4位 第5位 第6位 第7位 第8位

```
tube[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90, 0xbf,0xff};
```

0 1 2 3 4 5 6 7 8 9 - 全灭

电子钟程序（clock.c）

头文件和全局变量

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
```

```
#include "pthread.h"
```

数码管的七段码（25个：0至24）

```
unsigned char tube[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e,0x89,0xc7,0x8c,0xc8,0xc1,0xa3,0xbf,0x7f,0xff};
// 0 1 2 3 4 5 6 7 8 9 A b C d E F H L P n u o - .
```

```
// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
unsigned char addr[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
// 第1位 第2位 第3位 第4位 第5位 第6位 第7位 第8位
```

数码管的位置码：8个

```
unsigned int hour,minute,second;
```

时、分、秒



```
void * time_counter(void * data)
{
    for(;;)
    {
        second ++;

        if(second == 60)
        {
            second = 0;
            minute ++;

            if(minute == 60)
            {
                minute = 0;
                hour ++;

                if(hour == 24)
                    hour = 0;
            }
        }
        usleep(1015000);
    }

    return NULL;
}
```

计时线程

延时1秒



```
int check_input(char *argv[])
{
    if( argv[1][2] != ':' || argv[1][5] != ':' )
        return -1;

    else if(!((argv[1][0]-'0')>=0 && (argv[1][0]-'0')<=2))
        return -1;

    else if(!((argv[1][1]-'0')>=0 && (argv[1][1]-'0')<=9))
        return -1;

    else if(((argv[1][0]-'0')==2 && (argv[1][1]-'0')>=4))
        return -1;

    else if(!((argv[1][3]-'0')>=0 && (argv[1][3]-'0')<=5))
        return -1;

    else if(!((argv[1][4]-'0')>=0 && (argv[1][4]-'0')<=9))
        return -1;

    else if(!((argv[1][6]-'0')>=0 && (argv[1][6]-'0')<=5))
        return -1;

    else if(!((argv[1][7]-'0')>=0 && (argv[1][7]-'0')<=9))
        return -1;

    else
        return 0;
}
```

判断初始时间输入
是否正确的函数

主函数

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

```
    int i,number;
    int mem_fd;
    int leddisplay[8];
    unsigned char *cpld;
    void * retval;
    pthread_t th_time;
```

→ `mem_fd = open("/dev/mem", O_RDWR);`

打开数码管设备

→ `cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE | PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));`

```
if(cpld == MAP_FAILED)
    return;
```

```
for(i=0; i<8; i++)
{
```

数码管全灭

```
    *(cpld+(0xe6<<1)) = addr[i];
    *(cpld+(0xe4<<1)) = tube[24];
```

24: 数码管灭

//数码管地址 (0xe6<<1)为地址
//数码管个位 (0xe4<<1)为地址 数码管灭

```
if(check_input(argv) == -1)
```

```
{
    printf("input error!\n");
    return 0;
}
```

检测输入的时分秒初始值是否正确?

```
hour  = (argv[1][0] - '0')*10 + (argv[1][1] - '0');
minute = (argv[1][3] - '0')*10 + (argv[1][4] - '0');
second = (argv[1][6] - '0')*10 + (argv[1][7] - '0');
```

时分秒的初始值

```
leddisplay[2] = 22;
leddisplay[5] = 22;
```

// 显示 "-"
// 显示 "-"

→ `pthread_create(&th_time, NULL, time_counter, 0);`

```

while(1)
{
    leddisplay[0] = hour/10;
    leddisplay[1] = hour - leddisplay[0]*10;

    leddisplay[3] = minute/10;
    leddisplay[4] = minute - leddisplay[3]*10;

    leddisplay[6] = second/10;
    leddisplay[7] = second - leddisplay[6]*10;

    for(i=0; i<8; i++)
    {

```

5
9
—
5
9
—
5
0
5 9 — 5 9 — 5 0

//数码管地址 (0xe6<<1)为地址

在数码管上显示时间

```

        *(cpld+(0xe6<<1)) = addr[i];

```

```

        number = leddisplay[i];

```

```

        *(cpld+(0xe4<<1)) = tube[number];

```

//数码管个位 (0xe4<<1)为地址

```

        usleep(1000);

```

延时

number=0至9

```

    }
}

```

→ pthread_join(th_time, &retval);

```

for(i=0; i<8; i++)
{

```

数码管全灭

```

        *(cpld+(0xe6<<1)) = addr[i];

```

```

        *(cpld+(0xe4<<1)) = tube[24];

```

//数码管地址 (0xe6<<1)为地址

//数码管个位 (0xe4<<1)为地址

数码管灭

```

    }

```

→ munmap(cpld,0x10);

→ close(mem_fd);

```

    return 0;

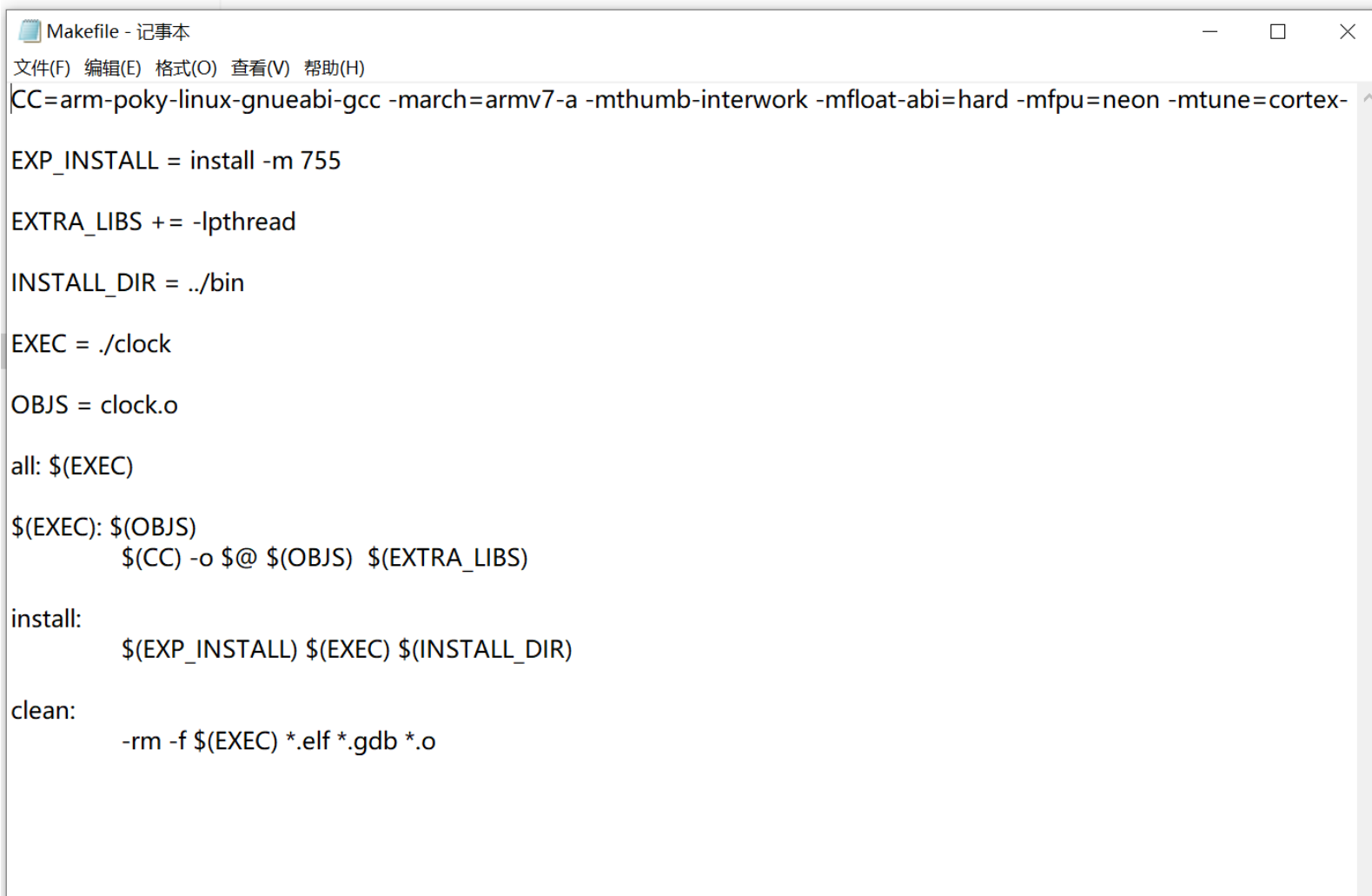
```

```

}

```

clock.c的Makefile文件



```
Makefile - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
CC=arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtune=cortex-
EXP_INSTALL = install -m 755
EXTRA_LIBS += -lpthread
INSTALL_DIR = ../bin
EXEC = ./clock
OBSJS = clock.o
all: $(EXEC)
$(EXEC): $(OBSJS)
    $(CC) -o $@ $(OBSJS) $(EXTRA_LIBS)
install:
    $(EXP_INSTALL) $(EXEC) $(INSTALL_DIR)
clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
```

在Ubuntu上执行的电子钟程序

- 在完成上述程序后，通过适当的修改，可以将其变为在Ubuntu上运行的电子钟程序。

- 修改如下：

- 1、隐掉与数码管相关的所有语句：

```
mem_fd = open("/dev/mem", O_RDWR);
cpId = (unsigned char*)mmap(NULL, (size_t)0x10, PROT_READ | PROT_WRITE | PROT_EXEC, MAP_SHARED, mem_fd, (off_t)(0x8000000));
if(cpId == MAP_FAILED)
    return;

*(cpId+(0xe6<<1)) = addr[i];                //数码管的位地址（即哪一个数码管）
*(cpId+(0xe4<<1)) = tube[num%10];           //数码管的段值（即显示什么内容）

munmap(cpId, 0x10);
close(mem_fd);
```

- 2、增加printf语句实现在电脑的显示器上（Ubuntu的终端上）显示时间，如“23-59-50”

- 3、修改Makefile文件：

- CC = gcc

Ubuntu环境下执行的电子钟程序（clock_pc.c）

只需要修改主函数，计时线程、判断初始时间输入是否正确的函数不需要修改

```
int main(int argc, char *argv[])
{
    int i,number;
    int mem_fd;
    int leddisplay[8];
    unsigned char *cpld;
    void * retval;
    pthread_t th_time;

    //      mem_fd = open("/dev/mem", O_RDWR);

    //      cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE | PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));

    //      if(cpld == MAP_FAILED)
    //          return;

    //      for(i=0; i<8; i++)
    //      {
    //          *(cpld+(0xe6<<1)) = addr[i];          //数码管地址 (0xe6<<1)为地址
    //          *(cpld+(0xe4<<1)) = tube[24];          //数码管个位 (0xe4<<1)为地址      数码管灭
    //      }

    if(check_input(argv) == -1)
    {
        printf("input error!\n");
        return 0;
    }

    hour  = (argv[1][0] - '0')*10 + (argv[1][1] - '0');
    minute = (argv[1][3] - '0')*10 + (argv[1][4] - '0');
    second = (argv[1][6] - '0')*10 + (argv[1][7] - '0');

    leddisplay[2] = 22;          // 显示 "-"
    leddisplay[5] = 22;

    pthread_create(&th_time, NULL, time_counter, 0);

    printf("\n\n");
```

隐掉



```

while(1)
{
    leddisplay[0] = hour/10;
    leddisplay[1] = hour - leddisplay[0]*10;

    leddisplay[3] = minute/10;
    leddisplay[4] = minute - leddisplay[3]*10;

    leddisplay[6] = second/10;
    leddisplay[7] = second - leddisplay[6]*10;

```

```

    for(i=0; i<8; i++)
    {

```

```

        //                *(cpld+(0xe6<<1)) = addr[i];                //数码管地址 (0xe6<<1)为地址

```

```

        //                number = leddisplay[i];

```

```

        //                *(cpld+(0xe4<<1)) = tube[number];                //数码管个位 (0xe4<<1)为地址

```

```

        if(leddisplay[i] == 22)
            printf("- ");

```

增加



```

        else
            printf("%d ",leddisplay[i]);

```

```

        usleep(1000);
    }

```



```

    printf("\r");                //回车

```

不换行

```

    pthread_join(th_time, &retval);

```

```

    //    for(i=0; i<8; i++)

```

```

    //    {
    //                *(cpld+(0xe6<<1)) = addr[i];                //数码管地址 (0xe6<<1)为地址
    //                *(cpld+(0xe4<<1)) = tube[24];                //数码管个位 (0xe4<<1)为地址
    //    }

```

```

    //    }

```

```

    //    munmap(cpld,0x10);

```

```

    //    close(mem_fd);

```

```

    printf("\n\n");

```

```

    return 0;

```

```


}

```

隐掉

隐掉

clock_pc.c的Makefile文件

 Makefile - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
CC = gcc
```

```
EXP_INSTALL = install -m 755
```

```
EXTRA_LIBS += -lpthread
```

```
INSTALL_DIR = ../bin
```

```
EXEC = ./clock_pc
```

```
OBJS = clock_pc.o
```

```
all: $(EXEC)
```

```
$(EXEC): $(OBJS)
```

```
$(CC) -o $@ $(OBJS) $(EXTRA_LIBS)
```

```
install:
```

```
$(EXP_INSTALL) $(EXEC) $(INSTALL_DIR)
```

```
clean:
```

```
-rm -f $(EXEC) *.elf *.gdb *.o
```


- 在Ubuntu的“终端”上，执行：

- `cd /imx6/whzeng/clock_pc`
- `make clean`
- `make`

请同学们在宿舍里完成！

- 如果编译正确，则在Ubuntu的“终端”上，执行（需要带参数）：

- `cd /imx6/whzeng/clock_pc`
- `./clock_pc 23:59:50`

- 此时，Ubuntu的“终端”上显示：

```
uptech@uptech:/imx6/whzeng/clock_pc$ ./clock_pc 23:59:50  
  
0 0 - 0 0 - 5 8
```

按Ctrl+C，退出程序

二、数码管应用实验2：小键盘控制的电子钟

- 使用小键盘控制数码管电子钟，要求：

- （1）程序执行后，数码管全灭。此时按小键盘的数字键（0-9）将从第一位（最左边的）数码管开始输入时间的初值。

- 按第一次数字键（如“2”），显示：2
- 按第二次数字键（如“3”），显示：23
- 按第三次数字键（如“5”），显示：23-5
- 按第四次数字键（如“9”），显示：23-59
- 按第五次数字键（如“5”），显示：23-59-5
- 按第六次数字键（如“0”），显示：23-59-50



- 在输入初始时间的过程中，可以随时按“**Stop/Clean**”键，此时8个数码管全灭，接下去再按数字键，则又从第一位开始输入初始时间。
- （2）初始时间输入完毕后，按“**Start**”键，则开始计时。如果输入的初始时间不正确（如输入了：24-59-50），则按下“**Start**”键后，8个数码管全灭。接下去再按数字键，则又从第一位开始输入初始时间。
- （3）在计时的过程中，按下“**Stop/Clean**”键，则停止计时。再按“**Start**”键，又开始计时（从停下的地方开始计时）。
- （4）计时停止时，再按“**Stop/Clean**”键，则8个数码管全灭（相当于Clean的功能）。接下去再按数字键，则又从第一位开始输入初始时间。



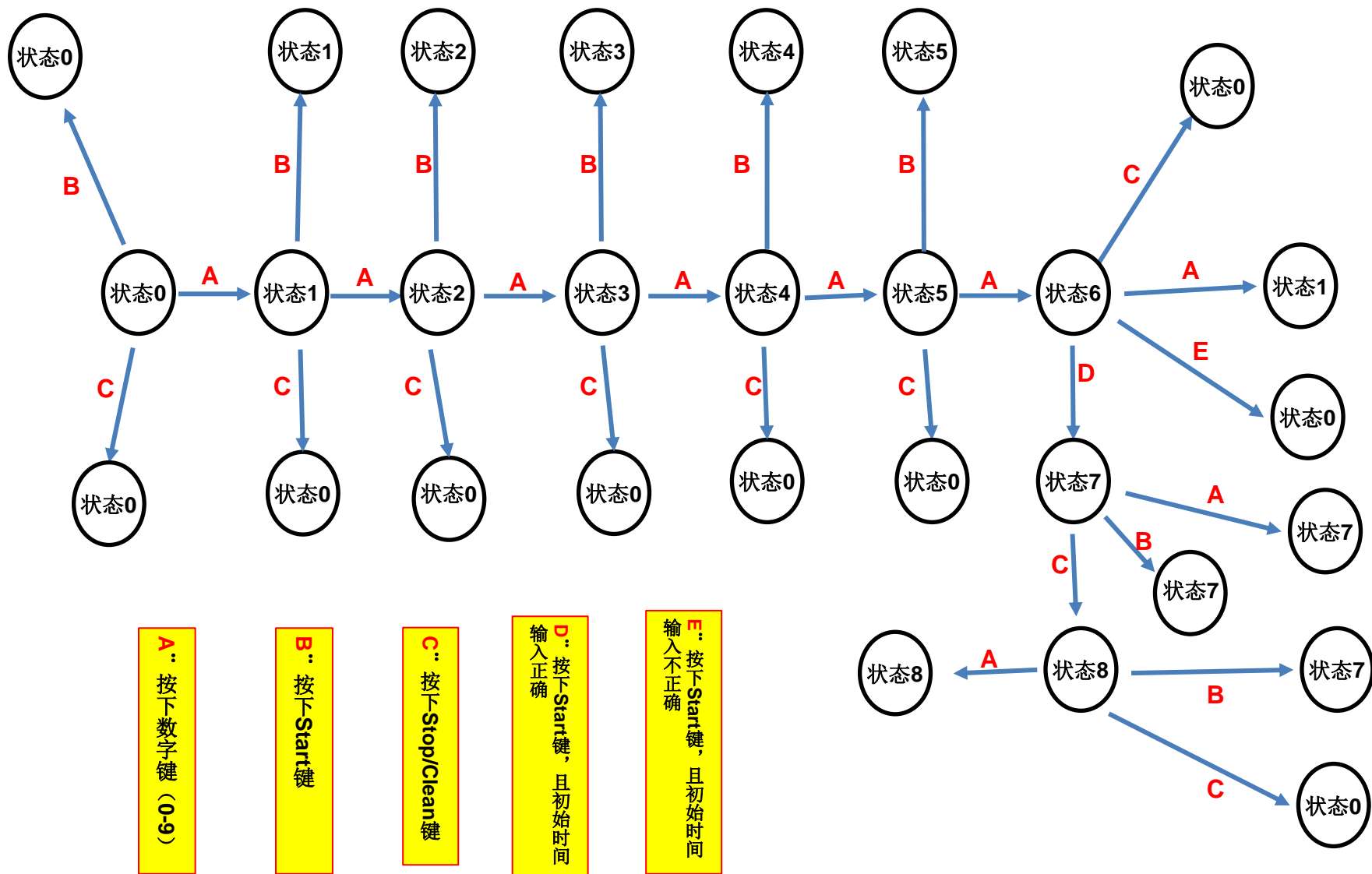
小键盘控制电子钟程序的编写思路

- 使用**2个多线程函数**:
 - 第1个多线程函数实现计时的功能（**计时线程**，与前面的电子钟程序相同）：**hour**、**minute**、**second**
 - **second**从0变动59
 - **minute**从0变动59
 - **hour**从0变到23
 - 通过**usleep**函数实现延时1秒的功能
 - 第2个多线程函数实现等待小键盘输入的功能（**小键盘输入线程**）；并且要根据输入的键值，**转入不同的状态**；然后根据不同的状态，**执行不同的任务**。

```
while(1)
{
    if(read(keys_fd,&t,sizeof(t)) == sizeof(t))
        if(t.type == EV_KEY)
            if(t.value == 0)
            {
                keyvalue = key_value(t.code); //将按键代码转换为键值
                keystate_current = check_key_state(keystate_before,keyvalue); //进行状态转换
                key_action(keystate_current,keystate_before,keyvalue); //根据状态做相应的动作
                keystate_before = keystate_current;
            }
}
```

- **状态转换函数**：根据前一次状态（`keystate_bedore`）和按键值（`keyvalue`），确定当前状态（`keystate_current`）。
 - `keystate_current = check_key_state(keystate_before,keyvalue);` //进行状态转换
- **执行动作函数**：根据前一次状态（`keystate_bedore`）、当前状态（`keystate_current`）和按键值（`keyvalue`），给出需要执行的动作。
 - `key_action(keystate_current,keystate_before,keyvalue);` //根据状态做相应的动作
- **主函数**（`main`函数）：主要完成在8个数码管上显示时间，与电子钟程序的主函数基本相同。

小键盘控制电子钟程序的9种状态（状态0至状态8）



- **状态0**: 初始状态, 数码管全灭
- **状态1**: 输入小时的十位, 数码管显示 “X”
- **状态2**: 输入小时的个位, 数码管显示 “X X”
- **状态3**: 输入分钟的十位, 数码管显示 “X X - X”
- **状态4**: 输入分钟的个位, 数码管显示 “X X - X X”
- **状态5**: 输入秒值的十位, 数码管显示 “X X - X X - X”
- **状态6**: 输入秒值的个位, 数码管显示 “X X - X X - X X”
- **状态7**: 开始计时, 数码管显示 “X X - X X - X X”, 时间在变化
- **状态8**: 停止计时, 数码管显示 “X X - X X - X X”, 时间停止变化

小键盘控制的电子钟程序（key_clock.c）

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <linux/input.h>
```

头文件和全局变量

```
#include "pthread.h"
```

```
#define KEYDevice "/dev/input/event4"
```

数码管的七段码

```
unsigned char tube[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e,0x89,0xc7,0x8c,0xc8,0xc1,0xa3,0xbf,0x7f,0xff};
// 0 1 2 3 4 5 6 7 8 9 A b C d E F H L P n u o - .
```

```
// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
unsigned char addr[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
// 第1位 第2位 第3位 第4位 第5位 第6位 第7位 第8位
```

数码管的位置码

```
unsigned int hour;
unsigned int minute;
unsigned int second;
```

```
unsigned int flag_timecounter;
unsigned int key_state;
```

```
unsigned int time_input[6] = {0,0,0,0,0,0};
unsigned int leddisplay[8] = {24,24,24,24,24,24,24,24};
```

```
unsigned int keys_fd;
unsigned char *cpid;
```

```
struct input_event t;
```


计时线程（与电子钟程序相同）

```
→ void * time_counter(void * data)
{
    for(;;)
    {
        if(flag_timecounter == 1)
            second ++;

        if(second == 60)
        {
            second = 0;
            minute ++;

            if(minute == 60)
            {
                minute = 0;
                hour ++;

                if(hour == 24)
                    hour = 0;
            }
        }
        usleep(1015000);
    }

    return NULL;
}
```

小键盘输入线程

```
→ void * key_input(void * data)  
{
```

```
    int keyvalue;  
    int keystate_before,keystate_current;
```

```
    keystate_before = key_state;
```

```
    // printf("keystate_before=%d\n",keystate_before);
```

```
    while(1)
```

```
    {  
        if(read(keys_fd,&t,sizeof(t)) == sizeof(t))
```

检测是否有按键?

```
            → if(t.type == EV_KEY)  
                if(t.value == 0)  
                {
```

```
                    printf("%c\n",key_code(t.code));
```

```
            → keyvalue = key_value(t.code);
```

得到按键值

```
            → keystate_current = check_key_state(keystate_before,keyvalue);
```

状态转换
函数

```
    // printf("keystate_current=%d\n",keystate_current);
```

```
            → key_action(keystate_current,keystate_before,keyvalue);
```

执行动作
函数

```
            → keystate_before = keystate_current;
```

```
        }
```

```
    }
```

```
}
```

状态转换函数（状态0至状态8）

```
int check_key_state(int keystate, int keyvalue)
```

```
{
```

```
    if(keystate == 0 && (keyvalue >= 0 && keyvalue <= 9))
        return 1;
```

```
    else if(keystate == 0 && keyvalue == 10)
        return 0;
```

```
    else if(keystate == 0 && keyvalue == 11)
        return 0;
```

```
    else if(keystate == 1 && (keyvalue >= 0 && keyvalue <= 9))
        return 2;
```

```
    else if(keystate == 1 && keyvalue == 10)
        return 1;
```

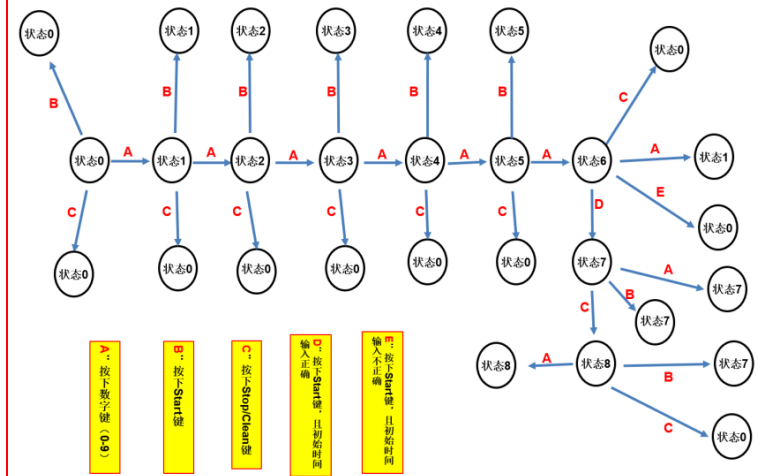
```
    else if(keystate == 1 && keyvalue == 11)
        return 0;
```

```
    else if(keystate == 2 && (keyvalue >= 0 && keyvalue <= 9))
        return 3;
```

```
    else if(keystate == 2 && keyvalue == 10)
        return 2;
```

```
    else if(keystate == 2 && keyvalue == 11)
        return 0;
```

小键盘控制电子钟程序的9种状态（状态0至状态8）



```
else if(keystate == 3 && (keyvalue >= 0 && keyvalue <= 9))
    return 4;
```

→ else if(keystate == 3 && keyvalue == 10)
return 3;

```
else if(keystate == 3 && keyvalue == 11)
    return 0;
```

```
else if(keystate == 4 && (keyvalue >= 0 && keyvalue <= 9))
    return 5;
```

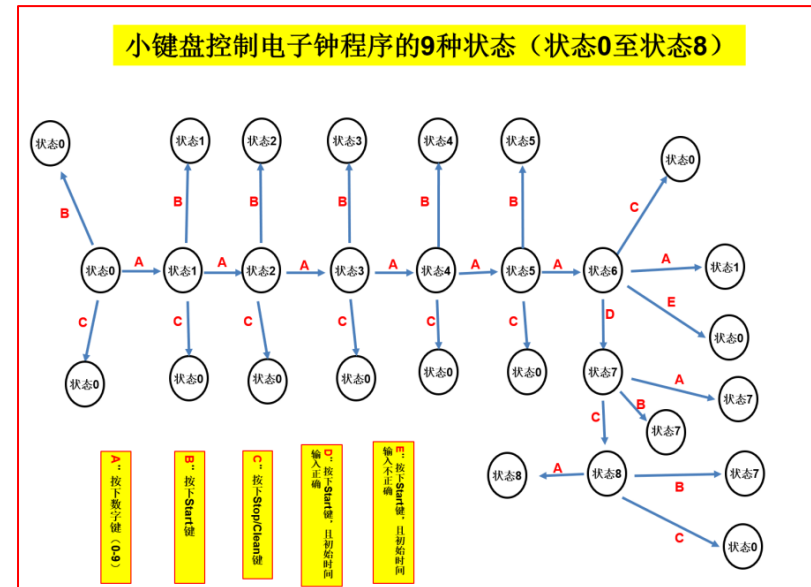
→ else if(keystate == 4 && keyvalue == 10)
return 4;

```
else if(keystate == 4 && keyvalue == 11)
    return 0;
```

```
else if(keystate == 5 && (keyvalue >= 0 && keyvalue <= 9))
    return 6;
```

→ else if(keystate == 5 && keyvalue == 10)
return 5;

```
else if(keystate == 5 && keyvalue == 11)
    return 0;
```



```

else if(keystate == 6 && (keyvalue >= 0 && keyvalue <= 9))
    return 1;

else if(keystate == 6 && keyvalue == 10 && check_key() == 0)
    return 7;

else if(keystate == 6 && keyvalue == 10 && check_key() == -1)
    return 0;

else if(keystate == 6 && keyvalue == 11)
    return 0;

```

```

else if(keystate == 7 && (keyvalue >= 0 && keyvalue <= 9))
    return 7;

else if(keystate == 7 && keyvalue == 10)
    return 7;

else if(keystate == 7 && keyvalue == 11)
    return 8;

```

```

else if(keystate == 8 && (keyvalue >= 0 && keyvalue <= 9))
    return 8;

else if(keystate == 8 && keyvalue == 10)
    return 7;

else if(keystate == 8 && keyvalue == 11)
    return 0;

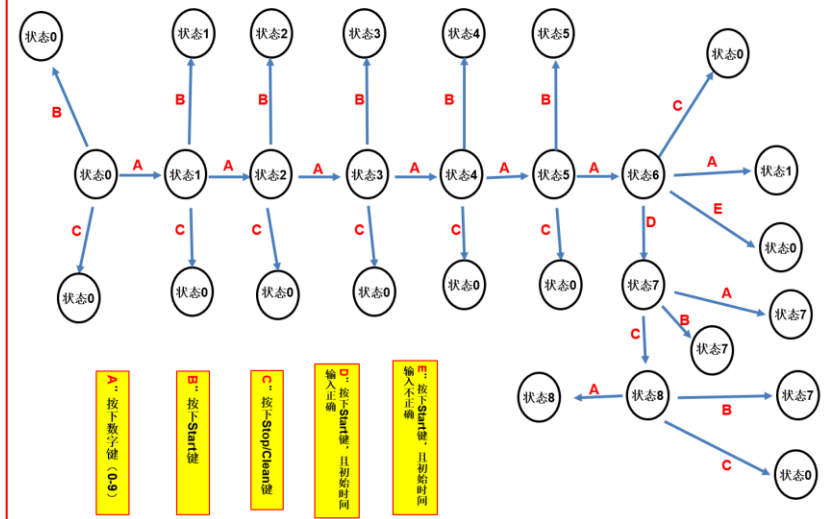
```

```

else
    return 0;

```

小键盘控制电子钟程序的9种状态（状态0至状态8）



```
void key_action(int keystate, int keystate_before, int keyvalue)
{
```

```
    int i;
```

```
    switch(keystate)
    {
```

```
         case 0:
```

```
            for(i=0; i<8; i++)
                leddisplay[i] = 24;
```

```
            break;
```

```
         case 1:
```

```
            if(keystate_before == 0 || keystate_before == 6)
            {
```

```
                leddisplay[0] = keyvalue;
                leddisplay[1] = 24;
                leddisplay[2] = 24;
                leddisplay[3] = 24;
                leddisplay[4] = 24;
                leddisplay[5] = 24;
                leddisplay[6] = 24;
                leddisplay[7] = 24;
```

```
                time_input[0] = keyvalue;
```

```
            }
```

```
            break;
```

```
         case 2:
```

```
            if(keystate_before == 1)
            {
```

```
                leddisplay[0] = time_input[0];
                leddisplay[1] = keyvalue;
                leddisplay[2] = 24;
                leddisplay[3] = 24;
                leddisplay[4] = 24;
                leddisplay[5] = 24;
                leddisplay[6] = 24;
                leddisplay[7] = 24;
```

```
                time_input[1] = keyvalue;
```

```
            }
```

```
            break;
```

执行动作函数（状态0至状态8）

- **状态0**: 初始状态，数码管全灭
- **状态1**: 输入小时的十位，数码管显示 “**X**”
- **状态2**: 输入小时的个位，数码管显示 “**X X**”
- **状态3**: 输入分钟的十位，数码管显示 “**X X - X**”
- **状态4**: 输入分钟的个位，数码管显示 “**X X - X X**”
- **状态5**: 输入秒值的十位，数码管显示 “**X X - X X - X**”
- **状态6**: 输入秒值的个位，数码管显示 “**X X - X X - X X**”
- **状态7**: 开始计时，数码管显示 “**X X - X X - X X**”，时间在变化
- **状态8**: 停止计时，数码管显示 “**X X - X X - X X**”，时间停止变化

→ case 3:

```
if(keystate_before == 2)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = keyvalue;
    leddisplay[4] = 24;
    leddisplay[5] = 24;
    leddisplay[6] = 24;
    leddisplay[7] = 24;

    time_input[2] = keyvalue;
}

break;
```

→ case 4:

```
if(keystate_before == 3)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = keyvalue;
    leddisplay[5] = 24;
    leddisplay[6] = 24;
    leddisplay[7] = 24;

    time_input[3] = keyvalue;
}

break;
```

- **状态0:** 初始状态，数码管全灭
- **状态1:** 输入小时的十位，数码管显示 “x”
- **状态2:** 输入小时的个位，数码管显示 “x x”
- **状态3:** 输入分钟的十位，数码管显示 “x x - x”
- **状态4:** 输入分钟的个位，数码管显示 “x x - x x”
- **状态5:** 输入秒值的十位，数码管显示 “x x - x x - x”
- **状态6:** 输入秒值的个位，数码管显示 “x x - x x - x x”
- **状态7:** 开始计时，数码管显示 “x x - x x - x x”，时间在变化
- **状态8:** 停止计时，数码管显示 “x x - x x - x x”，时间停止变化

→ case 5:

```
if(keystate_before == 4)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = time_input[3];
    leddisplay[5] = 22;
    leddisplay[6] = keyvalue;
    leddisplay[7] = 24;

    time_input[4] = keyvalue;
}

break;
```

→ case 6:

```
if(keystate_before == 5)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = time_input[3];
    leddisplay[5] = 22;
    leddisplay[6] = time_input[4];
    leddisplay[7] = keyvalue;

    time_input[5] = keyvalue;
}

break;
```

- **状态0:** 初始状态，数码管全灭
- **状态1:** 输入小时的十位，数码管显示 “x”
- **状态2:** 输入小时的个位，数码管显示 “x x”
- **状态3:** 输入分钟的十位，数码管显示 “x x - x”
- **状态4:** 输入分钟的个位，数码管显示 “x x - x x”
- **状态5:** 输入秒值的十位，数码管显示 “x x - x x - x”
- **状态6:** 输入秒值的个位，数码管显示 “x x - x x - x x”
- **状态7:** 开始计时，数码管显示 “x x - x x - x x”，时间在变化
- **状态8:** 停止计时，数码管显示 “x x - x x - x x”，时间停止变化



case 7:

```
if(keystate_before == 6)
{
    hour  = time_input[0]*10 + time_input[1];
    minute = time_input[2]*10 + time_input[3];
    second = time_input[4]*10 + time_input[5];
}
```

```
flag_timecounter = 1;
```

```
break;
```



case 8:

```
flag_timecounter = 0;
```

```
break;
```



default:

```
break;
```

```
}
```

- **状态0:** 初始状态, 数码管全灭
- **状态1:** 输入小时的十位, 数码管显示 “**x**”
- **状态2:** 输入小时的个位, 数码管显示 “**x x**”
- **状态3:** 输入分钟的十位, 数码管显示 “**x x - x**”
- **状态4:** 输入分钟的个位, 数码管显示 “**x x - x x**”
- **状态5:** 输入秒值的十位, 数码管显示 “**x x - x x - x**”
- **状态6:** 输入秒值的个位, 数码管显示 “**x x - x x - x x**”
- **状态7:** 开始计时, 数码管显示 “**x x - x x - x x**”, 时间在变化
- **状态8:** 停止计时, 数码管显示 “**x x - x x - x x**”, 时间停止变化

主函数

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

```
    int i,number;
    int mem_fd;
    void * retval;
    pthread_t th_time,th_key;
```

```
    flag_timecounter = 0;
    key_state = 0;
```

→ `keys_fd = open(KEYDevice, O_RDONLY);`

```
    if(keys_fd <= 0)
    {
        printf("open key device error!\n");
        return 0;
    }
```

→ `mem_fd = open("/dev/mem", O_RDWR);`

→ `cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE | PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));`

```
    if(cpld == MAP_FAILED)
        return;
```

```
    for(i=0; i<8; i++)
    {
```

```
        *(cpld+(0xe6<<1)) = addr[i];
        *(cpld+(0xe4<<1)) = tube[24];
```

```
    }
```

```
    for(i=0; i<8; i++)
        leddisplay[i] = 24;
```

→ `pthread_create(&th_time, NULL, time_counter, 0);`
`pthread_create(&th_key, NULL, key_input, 0);`

数码管全灭

//数码管地址 (0xe6<<1)为地址
//数码管个位 (0xe4<<1)为地址 数码管灭

```

while(1)
{
    if(flag_timecounter == 1)
    {
        leddisplay[0] = hour/10;
        leddisplay[1] = hour - leddisplay[0]*10;

        leddisplay[3] = minute/10;
        leddisplay[4] = minute - leddisplay[3]*10;

        leddisplay[6] = second/10;
        leddisplay[7] = second - leddisplay[6]*10;
    }

    for(i=0; i<8; i++)
    {

```

在数码管上显示时间

```

        *(cpld+(0xe6<<1)) = addr[i];

```

//数码管地址 (0xe6<<1)为地址

```

        number = leddisplay[i];

```

```

        *(cpld+(0xe4<<1)) = tube[number];

```

//数码管个位 (0xe4<<1)为地址

```

        usleep(1000);
    }
}

```

延时

```

pthread_join(th_time, &retval);
pthread_join(th_key, &retval);

```

```

for(i=0; i<8; i++)
{

```

数码管全灭

```

        *(cpld+(0xe6<<1)) = addr[i];

```

//数码管地址 (0xe6<<1)为地址

```

        *(cpld+(0xe4<<1)) = tube[24];

```

//数码管个位 (0xe4<<1)为地址 数码管灭

```

    }
}

```

```

munmap(cpld,0x10);

```

```

close(mem_fd);

```

```

close(keys_fd);

```

```

return 0;

```

```

}

```

判断初始时间输入是否正确的函数（与电子钟程序相同）



```
int check_key(void)
{
    if(!(time_input[0] >= 0 && time_input[0] <= 2))
        return -1;

    else if(!(time_input[1] >= 0 && time_input[1] <= 9))
        return -1;

    else if((time_input[0] == 2 && time_input[1] >= 4))
        return -1;

    else if(!(time_input[2] >= 0 && time_input[2] <= 5))
        return -1;

    else if(!(time_input[3] >= 0 && time_input[3] <= 9))
        return -1;

    else if(!(time_input[4] >= 0 && time_input[4] <= 5))
        return -1;

    else if(!(time_input[5] >= 0 && time_input[5] <= 9))
        return -1;

    else
        return 0;
}
```



```
char key_code(int code)
{
    switch(code)
    {
        case 2:
            return '1';

        case 3:
            return '2';

        case 4:
            return '3';

        case 5:
            return '4';

        case 6:
            return '5';

        case 7:
            return '6';

        case 8:
            return '7';

        case 9:
            return '8';

        case 10:
            return '9';

        case 1:
            return '*';

        case 115:
            return '0';

        case 114:
            return '#';

        default:
            return 'e';
    }
}
```

小键盘代码转换为电脑上的显示值函数



```
int key_value(int code)
{
    switch(code)
    {
        case 2:
            return 1;

        case 3:
            return 2;

        case 4:
            return 3;

        case 5:
            return 4;

        case 6:
            return 5;

        case 7:
            return 6;

        case 8:
            return 7;

        case 9:
            return 8;

        case 10:
            return 9;

        case 1:
            return 10;

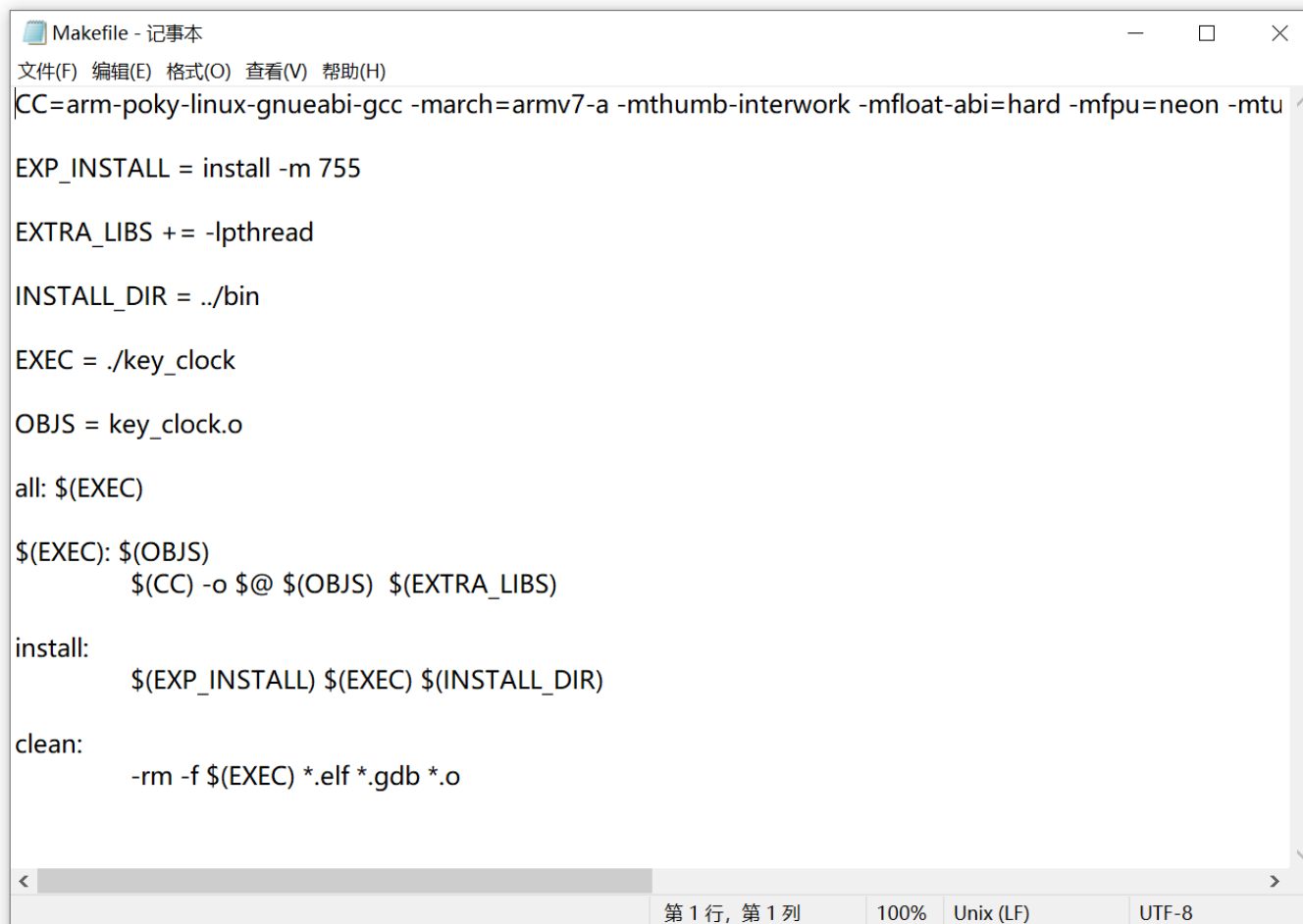
        case 115:
            return 0;

        case 114:
            return 11;

        default:
            return -1;
    }
}
```

小键盘代码转换为**keyvalue**值（0至11）函数

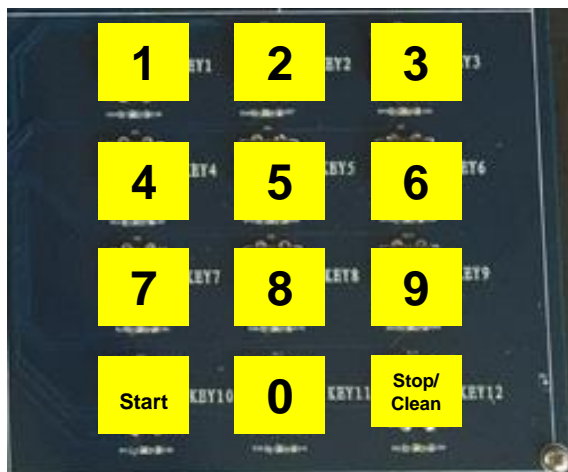
key_clock.c的Makefile文件



```
Makefile - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
CC=arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtu
EXP_INSTALL = install -m 755
EXTRA_LIBS += -lpthread
INSTALL_DIR = ../bin
EXEC = ./key_clock
OBSJ = key_clock.o
all: $(EXEC)
$(EXEC): $(OBSJ)
    $(CC) -o $@ $(OBSJ) $(EXTRA_LIBS)
install:
    $(EXP_INSTALL) $(EXEC) $(INSTALL_DIR)
clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
```

第 1 行, 第 1 列 100% Unix (LF) UTF-8

- 在实验箱的“超级终端（Xshell 4）”上执行（不需要带参数）：
 - `cd /mnt/whzeng/key_clock`
 - `./key_clock`
- 此时，通过小键盘的数字键设置时钟的初值，并启动时钟，然后停止时钟



在Ubuntu上执行的小键盘控制的电子钟程序

- 在完成上述程序后，通过适当的修改，可以将其变为在Ubuntu上运行的电脑键盘控制的电子钟程序，用电脑的0-9键代替小键盘的0-9键，电脑的s、t键代替小键盘的Start、Stop/Clean键。

- 修改如下：
 - 1、隐掉与数码管相关的所有语句：

```
mem_fd = open("/dev/mem", O_RDWR);
```

```
cpId = (unsigned char*)mmap(NULL, (size_t)0x10, PROT_READ | PROT_WRITE | PROT_EXEC, MAP_SHARED, mem_fd, (off_t)(0x8000000));
```

```
if(cpId == MAP_FAILED)
    return;
```

```
*(cpId+(0xe6<<1)) = addr[i];
```

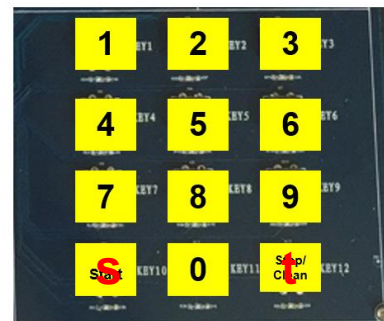
//数码管的位地址（即哪一个数码管）

```
*(cpId+(0xe4<<1)) = tube[num%10];
```

//数码管的段值（即显示什么内容）

```
munmap(cpId, 0x10);
```

```
close(mem_fd);
```



— 2、隐掉与小键盘相关的所有语句：

```
keys_fd = open(KEYDevice, O_RDONLY);

if(keys_fd <= 0)
{
    printf("open key device error!\n");
    return 0;
}

close(keys_fd);

if(read(keys_fd,&t,sizeof(t)) == sizeof(t))
    if(t.type == EV_KEY)
        if(t.value == 0)
```

```
int getch()
{
    struct termios tm, tm_old;
    int ch;

    if (tcgetattr(0, &tm) < 0)           //保存现在的终端设置
        return -1;

    tm_old = tm;
    cfmakeraw(&tm);                       //更改终端设置为原始模式，该模式下所有的输入数据以字节为单位被处理

    if (tcsetattr(0, TCSANOW, &tm) < 0)  //设置上更改之后的设置
        return -1;

    ch = getchar();

    if (tcsetattr(0, TCSANOW, &tm_old) < 0) //更改设置为最初的样子
        return -1;

    return ch;
}
```

getch()函数

— 3、隐掉两个小键盘键值转换函数。

— 4、增加电脑键值转换函数：0-9的ASCII为30H-39H，s、t的ASCII为73H、74H。

— 5、增加**printf语句**实现在电脑的显示器上（Ubuntu的终端上）显示时间，如“23-59-50”。

— 6、增加**getch()函数**实现电脑键盘的输入。

— 7、修改Makefile文件：

- CC = gcc

Ubuntu环境下执行的电脑键盘控制的电子钟程序（key_clock_pc.c）

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <linux/input.h>
```

头文件和全局变量

```
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <string.h>
#include <sys/signal.h>
#include <pthread.h>
```

```
#include "pthread.h"
```

```
#define KEYDevice "/dev/input/event4"
```

```
unsigned char tube[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e,0x89,0xc7,0x8c,0xc8,0xc1,0xa3,0xbf,0x7f,0xff};
// 0 1 2 3 4 5 6 7 8 9 A b C d E F H L P n u o - .
```

```
// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
unsigned char addr[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
// 第1位 第2位 第3位 第4位 第5位 第6位 第7位 第8位
```

```
unsigned int hour;
unsigned int minute;
unsigned int second;
```


```
unsigned int flag_timecounter;
unsigned int key_state;
```

```
unsigned int flag_exit;
```

```
unsigned int time_input[6] = {0,0,0,0,0,0};
unsigned int leddisplay[8] = {24,24,24,24,24,24,24,24};
```

```
unsigned int keys_fd;
unsigned char *cpld;
```

```
struct input_event t;
```



```
void * time_counter(void * data)
{
    for(;;)
    {
        if(flag_timecounter == 1)
            second ++;

        if(second == 60)
        {
            second = 0;
            minute ++;

            if(minute == 60)
            {
                minute = 0;
                hour ++;

                if(hour == 24)
                    hour = 0;
            }
        }
        usleep(1015000);
    }

    return NULL;
}
```

计时线程

不需要修改！

小键盘输入线程

需要修改!

```
void * key_input(void * data)
```

```
{
```

```
    int keyvalue;  
    int keystate_before,keystate_current;  
    char code;  
    int key;
```

```
    keystate_before = key_state;
```

```
    //printf("keystate_before=%d\n",keystate_before);
```

```
    while(1)
```

```
    {
```

```
        // if(read(keys_fd,&t,sizeof(t)) == sizeof(t))
```

```
            // if(t.type == EV_KEY)
```

```
            // if(t.value == 0)
```

```
            //{
```

```
                // printf("%c\n",key_code(t.code));
```

```
                //keyvalue = key_value(code);
```

```
                key = getch();
```

```
                if(key == 27)
```

```
                    flag_exit = 1;
```

```
                keyvalue = pc_key_value(key);
```

```
                //printf("keyvalue = %d\n",keyvalue);
```

```
                keystate_current = check_key_state(keystate_before,keyvalue);
```

```
                //printf("keystate_current=%d\n",keystate_current);
```

```
                key_action(keystate_current,keystate_before,keyvalue);
```

```
                keystate_before = keystate_current;
```

```
            //}
```

```
    }
```

```
}
```

隐掉



隐掉



按Esc键，退出



增加：电脑键盘输入函数

```
int getch()
{
    struct termios tm, tm_old;
    int ch;

    if (tcgetattr(0, &tm) < 0)           //保存现在的终端设置
        return -1;

    tm_old = tm;

    cfmakeraw(&tm);                       //更改终端设置为原始模式，该模式下所有的输入数据以字节为单位被处理

    if (tcsetattr(0, TCSANOW, &tm) < 0)  //设置上更改之后的设置
        return -1;

    ch = getchar();

    if (tcsetattr(0, TCSANOW, &tm_old) < 0) //更改设置为最初的样子
        return -1;

    return ch;
}
```

```

int check_key_state(int keystate, int keyvalue)
{
    if(keystate == 0 && (keyvalue >= 0 && keyvalue <= 9))
        return 1;

    else if(keystate == 0 && keyvalue == 10)
        return 0;

    else if(keystate == 0 && keyvalue == 11)
        return 0;

```



```

    else if(keystate == 0 && keyvalue == 10)
        return 0;

```

```

    else if(keystate == 0 && keyvalue == 11)
        return 0;

```

```

    else if(keystate == 1 && (keyvalue >= 0 && keyvalue <= 9))
        return 2;

```



```

    else if(keystate == 1 && keyvalue == 10)
        return 1;

```

```

    else if(keystate == 1 && keyvalue == 11)
        return 0;

```

```

    else if(keystate == 2 && (keyvalue >= 0 && keyvalue <= 9))
        return 3;

```



```

    else if(keystate == 2 && keyvalue == 10)
        return 2;

```

```

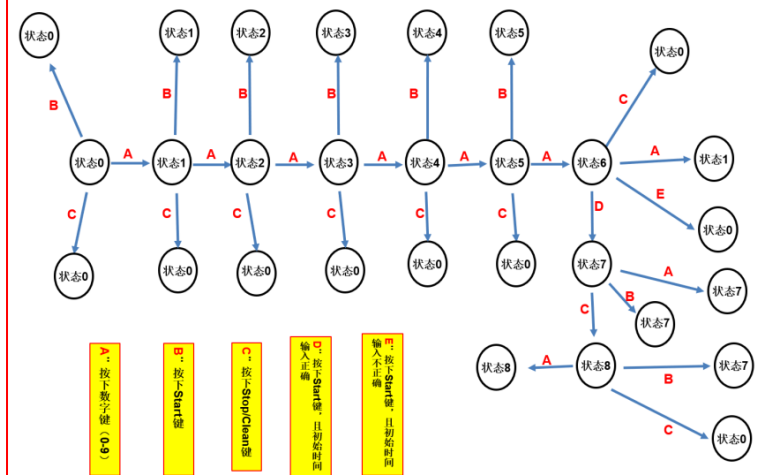
    else if(keystate == 2 && keyvalue == 11)
        return 0;

```

状态转换函数（状态0至状态8）

不需要修改！

小键盘控制电子钟程序的9种状态（状态0至状态8）



```
else if(keystate == 3 && (keyvalue >= 0 && keyvalue <= 9))
    return 4;
```

→ else if(keystate == 3 && keyvalue == 10)
return 3;

```
else if(keystate == 3 && keyvalue == 11)
    return 0;
```

```
else if(keystate == 4 && (keyvalue >= 0 && keyvalue <= 9))
    return 5;
```

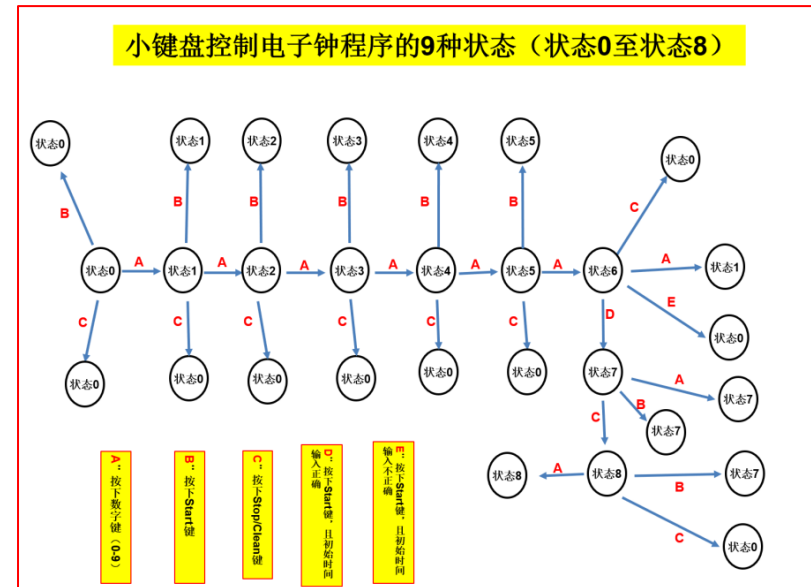
→ else if(keystate == 4 && keyvalue == 10)
return 4;

```
else if(keystate == 4 && keyvalue == 11)
    return 0;
```

```
else if(keystate == 5 && (keyvalue >= 0 && keyvalue <= 9))
    return 6;
```

→ else if(keystate == 5 && keyvalue == 10)
return 5;

```
else if(keystate == 5 && keyvalue == 11)
    return 0;
```




```
void key_action(int keystate, int keystate_before, int keyvalue)
{
```

```
    int i;
```

```
    switch(keystate)
    {
```

```
         case 0:
```

```
            for(i=0; i<8; i++)
                leddisplay[i] = 24;
```

```
            break;
```

```
         case 1:
```

```
            if(keystate_before == 0 || keystate_before == 6)
            {
```

```
                leddisplay[0] = keyvalue;
                leddisplay[1] = 24;
                leddisplay[2] = 24;
                leddisplay[3] = 24;
                leddisplay[4] = 24;
                leddisplay[5] = 24;
                leddisplay[6] = 24;
                leddisplay[7] = 24;
```

```
                time_input[0] = keyvalue;
```

```
            }
```

```
            break;
```

```
         case 2:
```

```
            if(keystate_before == 1)
            {
```

```
                leddisplay[0] = time_input[0];
                leddisplay[1] = keyvalue;
                leddisplay[2] = 24;
                leddisplay[3] = 24;
                leddisplay[4] = 24;
                leddisplay[5] = 24;
                leddisplay[6] = 24;
                leddisplay[7] = 24;
```

```
                time_input[1] = keyvalue;
```

```
            }
```

```
        break;
```

动作执行函数（状态0至状态8）

不需要修改！

- **状态0**: 初始状态，数码管全灭
- **状态1**: 输入小时的十位，数码管显示 “X”
- **状态2**: 输入小时的个位，数码管显示 “X X”
- **状态3**: 输入分钟的十位，数码管显示 “X X - X”
- **状态4**: 输入分钟的个位，数码管显示 “X X - X X”
- **状态5**: 输入秒值的十位，数码管显示 “X X - X X - X”
- **状态6**: 输入秒值的个位，数码管显示 “X X - X X - X X”
- **状态7**: 开始计时，数码管显示 “X X - X X - X X”，时间在变化
- **状态8**: 停止计时，数码管显示 “X X - X X - X X”，时间停止变化

→ case 3:

```
if(keystate_before == 2)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = keyvalue;
    leddisplay[4] = 24;
    leddisplay[5] = 24;
    leddisplay[6] = 24;
    leddisplay[7] = 24;

    time_input[2] = keyvalue;
}

break;
```

→ case 4:

```
if(keystate_before == 3)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = keyvalue;
    leddisplay[5] = 24;
    leddisplay[6] = 24;
    leddisplay[7] = 24;

    time_input[3] = keyvalue;
}

break;
```

- **状态0:** 初始状态，数码管全灭
- **状态1:** 输入小时的十位，数码管显示 “x”
- **状态2:** 输入小时的个位，数码管显示 “x x”
- **状态3:** 输入分钟的十位，数码管显示 “x x - x”
- **状态4:** 输入分钟的个位，数码管显示 “x x - x x”
- **状态5:** 输入秒值的十位，数码管显示 “x x - x x - x”
- **状态6:** 输入秒值的个位，数码管显示 “x x - x x - x x”
- **状态7:** 开始计时，数码管显示 “x x - x x - x x”，时间在变化
- **状态8:** 停止计时，数码管显示 “x x - x x - x x”，时间停止变化

→ case 5:

```
if(keystate_before == 4)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = time_input[3];
    leddisplay[5] = 22;
    leddisplay[6] = keyvalue;
    leddisplay[7] = 24;

    time_input[4] = keyvalue;
}

break;
```

→ case 6:

```
if(keystate_before == 5)
{
    leddisplay[0] = time_input[0];
    leddisplay[1] = time_input[1];
    leddisplay[2] = 22;
    leddisplay[3] = time_input[2];
    leddisplay[4] = time_input[3];
    leddisplay[5] = 22;
    leddisplay[6] = time_input[4];
    leddisplay[7] = keyvalue;

    time_input[5] = keyvalue;
}

break;
```

- **状态0:** 初始状态，数码管全灭
- **状态1:** 输入小时的十位，数码管显示 “x”
- **状态2:** 输入小时的个位，数码管显示 “x x”
- **状态3:** 输入分钟的十位，数码管显示 “x x - x”
- **状态4:** 输入分钟的个位，数码管显示 “x x - x x”
- **状态5:** 输入秒值的十位，数码管显示 “x x - x x - x”
- **状态6:** 输入秒值的个位，数码管显示 “x x - x x - x x”
- **状态7:** 开始计时，数码管显示 “x x - x x - x x”，时间在变化
- **状态8:** 停止计时，数码管显示 “x x - x x - x x”，时间停止变化



case 7:

```
if(keystate_before == 6)
{
    hour = time_input[0]*10 + time_input[1];
    minute = time_input[2]*10 + time_input[3];
    second = time_input[4]*10 + time_input[5];
}
```

```
flag_timecounter = 1;
```

```
break;
```



case 8:

```
flag_timecounter = 0;
```

```
break;
```



default:

```
break;
```

```
}
```

- **状态0:** 初始状态, 数码管全灭
- **状态1:** 输入小时的十位, 数码管显示 “**x**”
- **状态2:** 输入小时的个位, 数码管显示 “**x x**”
- **状态3:** 输入分钟的十位, 数码管显示 “**x x - x**”
- **状态4:** 输入分钟的个位, 数码管显示 “**x x - x x**”
- **状态5:** 输入秒值的十位, 数码管显示 “**x x - x x - x**”
- **状态6:** 输入秒值的个位, 数码管显示 “**x x - x x - x x**”
- **状态7:** 开始计时, 数码管显示 “**x x - x x - x x**”, 时间在变化
- **状态8:** 停止计时, 数码管显示 “**x x - x x - x x**”, 时间停止变化

主函数

需要修改!

```
int main(int argc, char *argv[])
{
    int i,number;
    int mem_fd;
    void * retval;
    pthread_t th_time,th_key;

    flag_timecounter = 0;
    key_state = 0;
    flag_exit = 0;

    // keys_fd = open(KEYDevice, O_RDONLY);

    // if(keys_fd <= 0)
    // {
    //     printf("open key device error!\n");
    //     return 0;
    // }

    // mem_fd = open("/dev/mem", O_RDWR);

    // cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE | PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));

    // if(cpld == MAP_FAILED)
    //     return;

    // for(i=0; i<8; i++)
    // {
    //     *(cpld+(0xe6<<1)) = addr[i];
    //     *(cpld+(0xe4<<1)) = tube[24];
    // }

    //数码管地址 (0xe6<<1)为地址
    //数码管个位 (0xe4<<1)为地址    数码管灭

    for(i=0; i<8; i++)
        leddisplay[i] = 24;

    pthread_create(&th_time, NULL, time_counter, 0);
    pthread_create(&th_key, NULL, key_input, 0);

    printf("\n\n");
```

隐掉

```

while(1)
{
    if(flag_timecounter == 1)
    {
        leddisplay[0] = hour/10;
        leddisplay[1] = hour - leddisplay[0]*10;

        leddisplay[3] = minute/10;
        leddisplay[4] = minute - leddisplay[3]*10;

        leddisplay[6] = second/10;
        leddisplay[7] = second - leddisplay[6]*10;
    }

    for(i=0; i<8; i++)
    {
        //          *(cpld+(0xe6<<1)) = addr[i];          //数码管地址 (0xe6<<1)为地址
        //          number = leddisplay[i];
        //          *(cpld+(0xe4<<1)) = tube[number];        //数码管个位 (0xe4<<1)为地址

        if(leddisplay[i] == 22)
            printf("- ");

        else if(leddisplay[i] == 24)
            printf(" ");

        //          else
        //              printf("%d ",leddisplay[i]);

        usleep(1000);
    }

    printf("\r");

    if(flag_exit == 1)
        return 0;
}

```

隐掉

增加

```

pthread_join(th_time, &retval);
pthread_join(th_key, &retval);

//      for(i=0; i<8; i++)
//      {
//                  *(cpld+(0xe6<<1)) = addr[i];
//                  *(cpld+(0xe4<<1)) = tube[24];
//      }
//      munmap(cpld,0x10);

//      close(mem_fd);

//      close(keys_fd);

return 0;
}

```

//数码管地址 (0xe6<<1)为地址
 //数码管个位 (0xe4<<1)为地址 数码管灭

判断初始时间输入是否正确的函数

不需要修改！



```
int check_key(void)
{
    if(!(time_input[0] >= 0 && time_input[0] <= 2))
        return -1;

    else if(!(time_input[1] >= 0 && time_input[1] <= 9))
        return -1;

    else if((time_input[0] == 2 && time_input[1] >= 4))
        return -1;


    else if(!(time_input[2] >= 0 && time_input[2] <= 5))
        return -1;

    else if(!(time_input[3] >= 0 && time_input[3] <= 9))
        return -1;

    else if(!(time_input[4] >= 0 && time_input[4] <= 5))
        return -1;

    else if(!(time_input[5] >= 0 && time_input[5] <= 9))
        return -1;

    else
        return 0;
}
```

 `int pc_key_value(int code)`
`{`
 `switch(code)`
 `{`
 `case 48:` `return 0;`
 `case 49:` `return 1;`
 `case 50:` `return 2;`
 `case 51:` `return 3;`
 `case 52:` `return 4;`
 `case 53:` `return 5;`
 `case 54:` `return 6;`
 `case 55:` `return 7;`
 `case 56:` `return 8;`
 `case 57:` `return 9;`
 `case 115:` `return 10;`
 `case 116:` `return 11;`
 `default:` `return -1;`
 `}`
`}`

0的ASCII码

9的ASCII码

s的ASCII码

t的ASCII码

增加：电脑键盘代码转换为
keyvalue值（0至11）的函数



```
char key_code(int code)
{
    switch(code)
    {
        case 2:
            return '1';

        case 3:
            return '2';

        case 4:
            return '3';

        case 5:
            return '4';

        case 6:
            return '5';

        case 7:
            return '6';

        case 8:
            return '7';

        case 9:
            return '8';

        case 10:
            return '9';

        case 1:
            return '*';


        case 115:
            return '0';

        case 114:
            return '#';

        default:
            return 'e';
    }
}
```

小键盘代码转换为电脑上的显示值

去掉这个函数！



```
int key_value(int code)
{
    switch(code)
    {
        case 2:
            return 1;

        case 3:
            return 2;

        case 4:
            return 3;

        case 5:
            return 4;

        case 6:
            return 5;

        case 7:
            return 6;

        case 8:
            return 7;

        case 9:
            return 8;

        case 10:
            return 9;

        case 1:
            return 10;

        case 115:
            return 0;

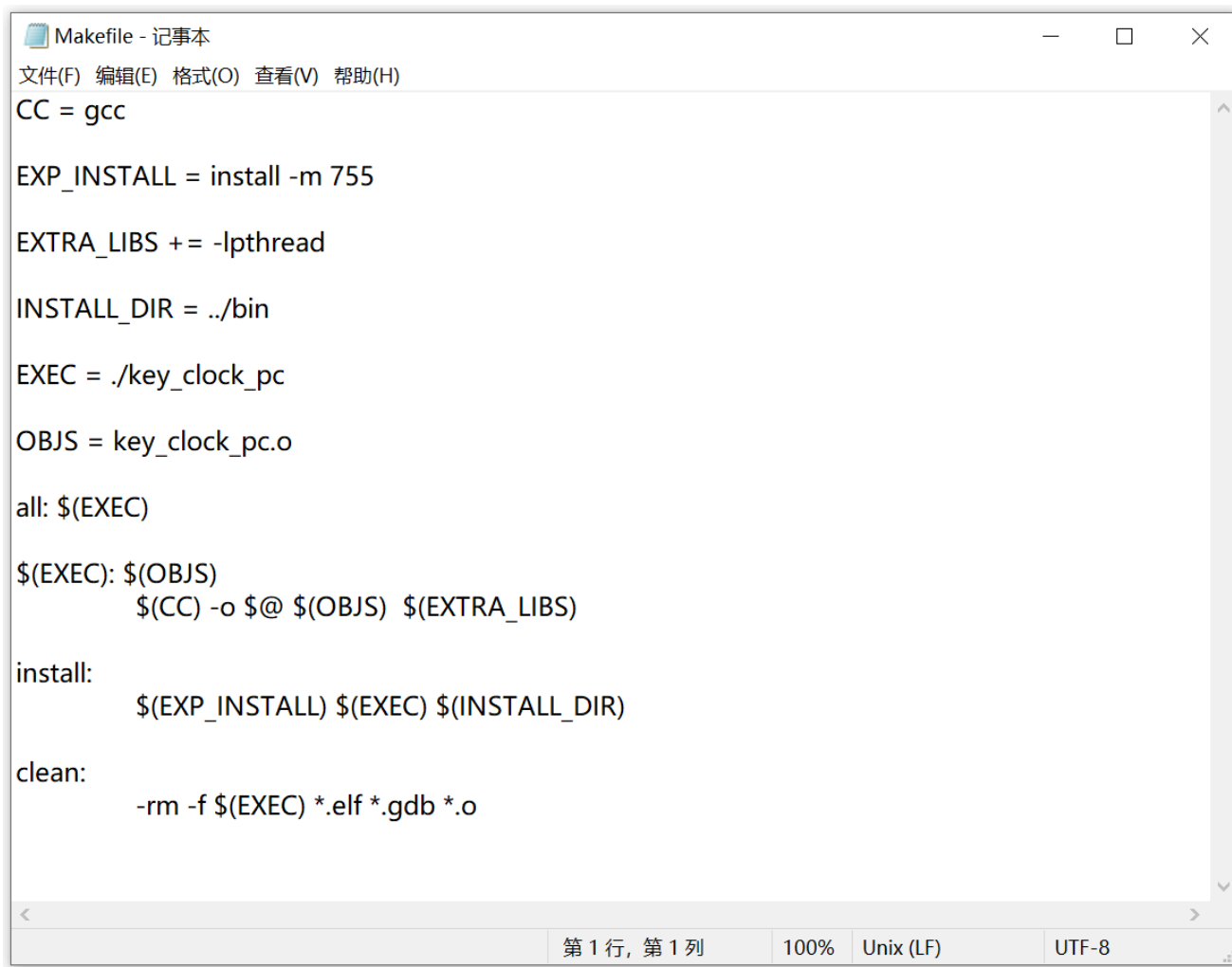
        case 114:
            return 11;

        default:
            return -1;
    }
}
```

小键盘代码转换为**keyvalue**值（0至11）

去掉这个函数！

key_clock_pc.c的Makefile文件



```
Makefile - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
CC = gcc

EXP_INSTALL = install -m 755

EXTRA_LIBS += -lpthread

INSTALL_DIR = ../bin

EXEC = ./key_clock_pc

OBJS = key_clock_pc.o

all: $(EXEC)

$(EXEC): $(OBJS)
    $(CC) -o $@ $(OBJS) $(EXTRA_LIBS)

install:
    $(EXP_INSTALL) $(EXEC) $(INSTALL_DIR)

clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
```

第 1 行, 第 1 列 100% Unix (LF) UTF-8

- 在Ubuntu的“终端”上，执行：

- `cd /imx6/whzeng/key_clock_pc`
 - `make clean`
 - `make`

请同学们在宿舍里完成！

- 如果编译正确，则在Ubuntu的“终端”上，执行（需要带参数）：

- `cd /imx6/whzeng/key_clock_pc`
 - `./key_clock_pc`

- 此时，Ubuntu的“终端”上显示：

```
uptech@uptech:/imx6/whzeng/key_clock_pc$ ./key_clock_pc
█
```

- 通过电脑键盘输入：2 3 5 9 5 0 （“-”会自动显示）

- 然后通过电脑键盘输入：s 则时钟开始计时

- 然后通过电脑键盘输入：t 则时钟停止计时

```
uptech@uptech:/imx6/whzeng/key_clock_pc$ ./key_clock_pc
2 3 - 5 9 - 5 2
```

Thanks