嵌入式工作室2022暑期招新——操作系统方向

任务概述

以 STM32 为平台，初步认识操作系统实现多线程技术的底层原理。

任务划分

任务一、第一盏灯

如果你接触过 STM32 编程，请跳过此任务。

开始任务前，请确保你拥有以下硬件设备：

1. 一台用于编程的个人电脑，建议使用Windows操作系统。
2. STM32 开发板一个（参考价格：30元（C8T6），15元（C6T6））
   建议使用 STM32F103C8T6 或者 STM32F103C6T6（内存较小）完成任务，建议购买已焊接排针版本。

3. ST-Link（参考价格：20元）
   ST-Link可以将PC上编译好的程序烧录到开发板中，并提供调试功能。

4. TTL转USB转接器一个（参考价格：10~20元）

   为了让开发板能够通过串口将字符串输出到PC端口。转接器样式较多，可能与参考图片差异较大。

5. 杜邦线若干（参考价格：白嫖）

   至少会用到8根母对母杜邦线，用来连接前面提到的各个器件。可以到工作室白嫖，工作室杜邦线已经多到泛滥成灾了。如果购买ST-Link和TTL转USB转接器时赠送了相应的杜邦线，则不用额外购买。

详细步骤：

1. 下载并安装 keil（写代码的）和 STM32CubeMX（初始化项目的）。
2. 学会使用 STM32CubeMX 创建 keil 能够打开的项目。
3. 在 keil 中打开刚才新建的项目。
4. 向主函数中添加 GPIO 初始化和设置电平的代码。
5. 连接好 PC、ST-Link 和开发板，烧录程序并运行。
6. 完成后对开发板拍照并提交。

注意：

1. 使用 STM32CubeMX创建项目时勾选 "copy only the necessary library files" 以避免项目过大。
2. 记得把 "System Core -> SYS -> Debug" 选为 "Serial wire"，以防 ST-Link 失效。

任务二、Hello World！

如果你会使用 USART 协议向上位机发送消息，并使用串口助手接受消息，请跳过此任务。

详细步骤：

1. 使用 STM32CubeMX 初始化一个 USART 外设，设定波特率为 115200。
2. 向主函数中添加发送字符串 "Hello World!" （不包含引号）的代码。
3. 连接好 PC、ST-Link、开发板和 TTL 转 USB 转接器，烧录程序并运行。
4. 完成后在串口助手中截图并提交。

注意：

1. 你的 printf 函数可能不会正常工作！因此暂时不要使用“stdio.h”中的内容！
2. 如果你想使 printf 正常工作，请重写 fputc 函数（armcc）或者 _write 函数（gcc）。

任务三、ARM汇编基础

详细步骤：

1. 了解 ARM 架构的通用寄存器 R0~R12、SP、LR、PC。
2. 了解 ARM 汇编指令 STR、LDR、STM、LDM、MRS、MSR、BX、BL等。
3. 了解如何在 keil 中编写汇编文件，内嵌汇编。
4. 使用 C 语言定义一个变量 flag，然后调用汇编编写的一个函数，将其修改。
5. 使用 C 语言定义一个函数，其包含两个 int 参数，功能是向上位机发送二者之和。在汇编中正确调用该函数。
6. 完成后提交相关代码文件。

任务四、线程创建与切换

详细步骤：

1. 了解什么是线程栈、上下文、栈帧。
2. 思考暂停一个线程需要保存哪些内容，恢复一个线程需要加载哪些内容？
3. 思考创建线程时要在栈中指定哪些寄存器的值，如何指定线程的入口函数？
4. 以如下框架完善程序：

```
//创建一个运行 func 函数的线程，初始栈指针为 stack，创建完成后返回新的线程栈指针。
void* create_thread(void (*func)(void), void* stack){
    ...
}

//暂停当前线程，将当前线程栈保存到 *cur，并将新的栈指针赋值给 *cur。切换到线程栈为 dst 的线程运行。
void switch_thread(void* dst, void** cur);

//发送字符串到上位机。
void send(const char *str){
    ...
}


int stack[2][256];
void* thread[2];

void func_0(void){
    send("thread 0 start.\n");
    switch_thread(thread[1], &thread[0]);
    send("thread 0 restart\n");
    switch_thread(thread[1], &thread[0]);
}

void func_1(void){
    send("thread 1 start.\n");
    switch_thread(thread[0], &thread[1]);
    send("thread 1 restart.\n");
    while(1);
}

int main(){
    ...
    void* main_thread;
    thread[0] = create_thread(func_0, stack + 1);
    thread[1] = create_thread(func_1, stack + 2);
    switch_to(thread[0], &main_thread);
}
```

附加任务、时间片轮转

如果你有编写过多任务嵌入式操作系统内核，请以该内核代码代替此任务提交。

1. 了解 STM32（cortex-m3） 中断发生时，硬件替我们做了哪些工作？
2. 如何组织线程控制块，使得我们可以方便地决定下一个运行的线程？
3. 何时进行上下文切换？如果在中断中进行上下文切换，需要满足什么条件？上下文切换是否能中断其他中断？是否能被其他中断中断？
4. 以如下框架完善程序：

```c
//系统时钟计数。操作系统启动后每 1ms 增加一次。
volatile int os_tick_count = 0;

//初始化你的操作系统。
void os_init(){
    ...
}

//创建一个线程，线程将在操作系统启动后运行 func(arg)。
void os_task_create(void (*func)(void*), void* arg){
    ...
}

//启动操作系统，加载运行第一个线程，并使时间片开始流转。
void os_start(){
    ...
}

//发送字符串到上位机。
void send(const char *str){
    ...
}

//延时 t ms。
void delay(int t){
    int target = os_tick_count + t;
    while(os_tick_count < target);
}

//线程所执行的函数。
void task_func(void* arg){
    while(1){
        //输出 arg 所代表的字符串。
        send((const char*)arg);
        //延时 1s。
        delay(1000);
    }
}

int main(){

    //初始化硬件。
    ...
    //初始化操作系统。
    os_init();
    //创建两个线程。
    os_task_create(task_func, "This is task 1.\n");
    os_task_create(task_func, "This is task 2.\n");
    //启动操作系统。
    os_start();
    //程序应当永远不会运行到此处。
while(1){
    }
}
```

5. 完成后提交相关代码文件。

通过标准

达到以下条件之一

- 完成任务三；积极思考任务四中提出的问题，提交学习总结，视学习态度以及进度决定是否通过。
  即使无法完成任务，也对今后的课程学习有帮助。
- 完成附加任务。
  难度高，适合对嵌入式开发有一定经验、熟练掌握 C 语言且对操作系统有浓厚兴趣的同学。

遇到任何问题（不知道从何下手、遇到问题不会解决、题目有误等等）请联系 QQ 527517418，加 QQ 时备注问题。

可以参考网络上的实现，但不能照抄。若在任务三、任务四和附加任务中发现原封不动的抄袭（包括只修改变量名），则直接判定为不通过。若多份提交雷同，抄袭者与被抄袭者同样判定为不通过。

截止时间：约为2022年秋期开学第一周周末、具体时间请关注群内通知。

如何提交

提交一个名为“姓名-学号-嵌入式工作室暑期招新.zip”的压缩包，发送到 mikiwang@std.uestc.edu.cn。

压缩包内应包含多个文件夹，命名为“任务*”，按任务要求提交相应的内容。