

# 实 验 指 南

|  |  |
| --- | --- |
| 课 程： | 嵌入式系统协同设计 |
| 实验序号： | 第5次实验 |
| 实验名称： | LiteOS-M向STM32平台的移植实验 |
| 院 系： | 软件学院 |
| 专业班级： | 软件工程2102班 |
| 学 号： | U202117281 |
| 姓 名： | 张骁凯 |

2023年07月07日

**一、实验目的**

（1）掌握LiteOS-M对硬件的依赖。

（2）掌握LiteOS-M移植到STM32F103开发板的过程。

（3）掌握LiteOS-M的启动过程。

**二、实验内容**

（1）把LiteOS-M移植到STM32F103开发板，并完成开发板上LED和KEY的控制，实现用开关KEY控制LED灯的亮/灭程序。

**三、实验环境**

宿主机操作系统：Windows X版本/Ubuntu X版本（内核版本：Linux 5.10.184）

虚拟机环境：QEMU和版本号

开发板：开发板或智能小车的品牌或名字，CPU型号和主要外设资源

编译工具：gcc/g++

编辑工具：vi/vim/vscode/notepad++

主要依赖包：make, gcc, gdb, bison, flex, libncurses5-dev, libssl-dev, libidn11

**四、实验设计思路**

**4.1 华为LiteOS-M内核目录和源文件结构**

此节介绍 LiteOS-M内核的目录和源文件结构，每个目录的作用，每个源文件的主要作用，并把主要源文件的首页前面部分内容截图放在文档中。

新起一段，稍微详细地描述LiteOS-M与特定CPU和硬件相关文件和主要参数。

**4.2 LiteOS-M对硬件的依赖**

本节主要介绍LiteOS-M对特定CPU、内存布局，时钟，中断，串口等硬件参数的依赖。介绍每个设置的取值或参数做详细解释。

新起一段：用户若要移植到新的开发板上，一般应该修改或增删哪些数据结构，源文件。

**4.3 程序结构（或程序流程、或程序原理，任选一个或多个组合）**

**4.3.1 硬件结构**

具体描STM32开发的CPU和硬件电路的原理，特别要突出介绍与移植有关的硬件机制和参数。

**4.3.2 内核的裁剪和应用程序**

具体描述内核针对STM32开发板的裁剪过程，需要增加哪些源文件或修改哪些源代码。读者可以按你描述的过程重现本实验。

新起一段，具体描述增加哪些STM32相关的支持库，这些支持库是如何获得的，做了哪些修改。

新起一段，具体描述应用程序的开发，用开关KEY控制LED灯的亮/灭的程序，描述应用程序的基本结构，主要数据结构。需要增加哪些源文件或修改哪些源代码。读者可以按你描述的过程重现本实验。

**4.3.3 关键函数或参数或某种机制**

本节如下几个示例列举应用编程中用到的或参考到的2-6个关键函数或某个参数或某种机制的原型和说明，或者说明内核中于设备管理的相关的函数或某个参数或某种机制。

（1）SYSCALL\_DEFINEn(name, ...)

SYSCALL\_DEFINEn()用于系统调用实现函数的定义，其中n为参数个数（0~5）。系统调用实现函数不能像常规函数一样直接定义，需要借助SYSCALL\_DEFINEn()的宏展开。宏参数中name为函数名，可变参数为函数原型的参数列表。

例如：SYSCALL\_DEFINE2(add, int, arg1, int, arg2)

（2）long int syscall(long int sysno, ...)

syscall()函数定义在unistd.h头文件中，用于直接根据系统调用号来使用系统调用。参数sysno是系统调用号，可变参数为需要向系统调用传递的0~5个参数。

例如：

ret = syscall(442, 3, 6, 9);

**五、关键代码分析**

本节如下几个示例列举驱动程序中或应用编程中的关键函数的片段。

**5.1 程序关键片段一：添加系统调用号**

在/usr/src/linux-5.10.184/arch/x86/entry/syscalls目录，使用gedit syscall\_64.tbl命令在547后面添加系统调用号548和549，分别为add函数和max函数的调用

548 common add sys\_add

549 common max sys\_max

**5.2 程序关键片段二：添加函数声明**

在/usr/src/linux-5.10.184/arch/x86/include/asm目录，使用gedit syscalls.h命令添加两个函数声明。

long \_sys\_add(int x, int y);

long \_sys\_max(int x, int y, int z);

**5.3 程序关键片段三：添加函数定义**

在/use/src/linux-5.10.184/kernel目录，使用gedit sys.c命令添加两个函数的定义：

SYSCALL\_DEFINE2(add, int, arg1, int, arg2)

{

return (arg1 + arg2);

}

SYSCALL\_DEFINE3(max, int, arg1, int, arg2, int, arg3)

{

if (arg1 > arg2)

return (arg1 > arg3 ? arg1 : arg3);

else

return (arg2 > arg3 ? arg2 : arg3);

}

**5.4 程序关键片段四：编写测试程序**

在测试目录编写测试程序：

#include <stdio.h>

#include <unistd.h>

#include <linux/kernel.h>

#include <sys/syscall.h>

int main()

{

int nRet;

nRet = syscall(548, 20, 18);

printf("%d\n", nRet);

nRet = syscall(549, 20, 18, 4);

printf("%d\n", nRet);

return 0;

}

**六、程序运行结果和分析**

本节图文并茂的方式介绍内核或应用程序的运行和调试截图。写出运行的命令或操作，截图展示运行效果，强调需要用户观察的现象或结果。

**七、实验错误排查和解决方法**

如下几个示例，写出实验过程出现的错误或bug，并写出排除的过程和结果。

**7.1 环境配置问题/无法进行配置和安装**

确认依赖程序是否安装

$ sudo apt install make gcc gdb bison flex libncurses5-dev libssl-dev libidn11 libelf-dev build-essential

**7.2：编译安装时提示“No space left on device”**

空间分配不足：在VMWare中重新为Kylin OS分配存储，并在disks中将新分配的存储设置为可用。（No space left on device）

**7.3：编译时提示“没有规则可制作目标“debian/canonical-certs.pem”**

修改"/usr/src/linux-5.11.11/.config"文件内容

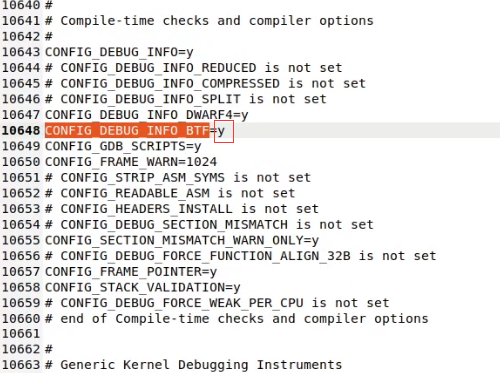
CONFIG\_SYSTEM\_TRUSTED\_KEYS="debian/canonical-certs.pem"修改为:

CONFIG\_SYSTEM\_TRUSTED\_KEYS=""

保存后继续编译即可。

**7.4：编译时报错“tmp\_vmlinux.btf: pahole (pahole) is not available. Failed to generate BTF for vmlinux“**

修改 .config 文件中的 CONFIGDEBUGINFO\_BTF：找到CONFIGDEBUGINFO\_BTF=y，改为CONFIGDEBUGINFO\_BTF=n，保存退出。



**7.5：重启后无法进入高级设置**

修改配置文件：打开/etc/default/grub，将 GRUB\_TIMEOUT设置为-1。运行命令update-grub完成修改。

1. **实验参考资料和网址**

（1）教学课件

（2）《深入理解Linux内核》第五版，第三章

（3）网上链接1：https://www.xgboke.com/16586.html

（4）网上链接2：