厦门大学《嵌入式系统》课程期末试卷

信息学院　　软件工程系　　2017 级　　软件工程专业

主考教师：曾文华 试卷类型：（B 卷） 考试时间：2020.1.7

一、填空题（30 个空，每 1 空 1 分，共 30 分；在答题纸填写答案时请写上每个空格的对应编号）

1、嵌入式系统的前身通常称为（1）单片机。

2、ARM 处理器的特权模式是指除（2）用户模式外的其他六种模式。

3、Thumb 指令只有（3）B 指令是可以条件执行的指令，其他都不能条件执行；而大多数 ARM 指令是可以条件执行的。

4、RT-Linux 通过在 Linux 内核与硬件中断之间增加一个精巧的可抢先的（4）实时内核，把标准的 Linux 内核作为（4）实时内核的一个进程与用户进程一起调度。

5、μCLinux 的 μ 是指（5）微，C 是指（6）控制，μCLinux 是专门针对没有 MMU（存储管理单元）的处理器设计的。

6、Linux 的版本号包括主版本号（序号的第 1 位）、次版本号（序号的第 2 位）和修订号（序号的第 3 位）。如果序号的第 2 位为偶数，则表示该版本是（7）稳定版；如果序号的第 2 位为奇数，则表示该版本为（8））测试版。

7、用的 Boot Loader 有：（9）Blob、（10）U-boot 和（11）vivi。

8、Cramfs 是专门针对闪存设计的（12）只读压缩的文件系统。

9、Flash Memory（闪存）有两种技术，分别是（13）NOR Flash 和（14）NAND Flash。

10、Linux 系统挂载的第一个文件系统就是（15）根文件系统。

11、Linux 的设备驱动程序开发调试有两种方法，第一种是直接编译到（16）内核；第二种是编译为（17）模块的形式，单独加载运行调试。

12、创建字符设备文件的命令是（假设设备名为/dev/lp0，主设备号为 6，次设备号为 0）：（18）mknod /dev/lp0 c 6 0。

13、设备的控制操作是通过调用 file_operations 结构体中的（19）ioctl() 函数完成的。

14、块设备驱动程序没有 read 和 write 操作函数，对块设备的读写是通过（20）请求函数完成的。

15、IMX6 实验箱打开电源（或按 Reset 键）后，通常需要重新设置 IP 地址，并执行挂载命令“mount -t nfs 59.77.5.122:/imx6 /mnt”。该挂载命令中的 nfs 是指（21）网络文件系统，59.77.5.122 是指（22）虚拟机的 IP 地址。

16、假设某个 make 命令的执行结果为“gcc -O2 -pipe -g -feliminate-unused-debug-types -c -o hello.o hello.c”，该结果里“-c”中的 c 是（23）编译的意思，“-o”中的 o 是（24）输出的意思。

17、Boot Loader 的阶段 1 主要包含依赖于 CPU 的体系结构硬件初始化的代码，通常都用（25）汇编语言来实现；Boot Loader 的阶段 2，通常用（26）C 语言完成，以便实现更复杂的功能，也使程序有更好的可读性和可移植性。

18、YAFFS（Yet Another Flash File System）是专为嵌入式系统使用（27）NAND 型闪存而设计的一种日志型文件系统。

19、U-boot 2014 的目标结构中的 arch 子目录，存放的是与（28）体系结构相关的代码。

20、使用 mmap 系统调用，可以将（29）内核空间的地址映射到（30）用户空间。

二、名词解释（请写出下列英文缩写的中文全称，10 小题，每 1 小题 1 分，共 10 分；在答题纸填写答案时请写上每小题的对应编号）

1、CPSR：Current Program Status Register，程序状态寄存器

2、Cramfs：Compressed ROM File System，只读压缩的文件系统

3、CAN：Controller Area Network, 控制器局域网络

4、DSP：Digital Signal Processor，数字信号处理器

5、GPIO：General Purpose Input/Output，通用输入输出，通用可编程接口

6、JFFS：Journalling Flash File System，闪存设备日志型文件系统

7、Ramfs：RAM File System，基于 RAM 的文件系统

8、SoC：System on Chip，片上系统

9、SMBus：System Management Bus，系统管理总线

10、TFTP：Trivial File Transfer Protocol，简单文件传输协议

三、简答题（10 小题，共 30 分；在答题纸填写答案时请写上每小题的对应编号）

1、常见的嵌入式操作系统有哪些？（3 分）

答：

- 嵌入式 Linux
- VxWorks
- μC/OS-II
- Windows CE
- Sysbian
- Android
- iOS
- QNX
- Palm OS
- LynxOS
- Nucleus
- ThreadX
- eCos

2、MMU（Memory Management Unit）的主要作用是什么？（2 分）

答：

（1）地址映射

（2）对地址访问的保护和限制

3、请写出 ARM 指令的格式。（3 分）

答：

<opcode> {<cond>} {S} <Rd>, <Rn> {, <shift_op2>}

< >内的项是必须的，{ }内的项是可选的

opcode：指令助记符（操作码），如 LDR，STR 等

cond：执行条件（条件码），如 EQ，NE 等

S：可选后缀，加 S 时影响 CPSR 中的条件码标志位，不加 S 时则不影响

Rd：目标寄存器

Rn：第 1 个源操作数的寄存器

op2：第 2 个源操作数

shift：位移操作

4、GNU 汇编语言语句格式为：

[<label>:][<instruction or directive or pseudo-instruction>] @comment

请指出，该汇编语言语句格式中，各个字段的含义。（2 分）

答：

① <label>: 标号

② instruction 指令

③ directive 伪操作

④ pseudo-instruction 伪指令

⑤ @comment 语句的注释

5、ARM 指令格式中的条件码（<cond>）中有 4 个状态位，分别是什么？（2 分）

答：N（负数标志位）、Z（零标志）、C（进位标志）、V（溢出标志）。

6、宿主机（PC 机）与目标板（IMX6 实验箱）的连接方式有哪些？（3 分）

答：

1) 串口
2) 以太网接口
3) USB 接口
4) JTAG 接口（Joint Test Action Group）

7、甲乙两台嵌入式设备都有 RS-232 串口，现要通过 RS-232 串口实现两台设备的通讯（双向通讯），请问怎么连接两台设备的 RS-232 串口（即两台设备的 RS-232 串口信号怎么连接）？（3 分）

答：甲设备 RS-232 串口的 TxD 连接乙设备 RS-232 串口的 RxD，甲设备 RS-232 串口的 RxD 连接乙设备 RS-232 串口的 TxD，两台设备 RS-232 串口的 GND 连在一起。

8、什么是 Boot Loader？其作用是什么？常见的 Boot Loader 有那几个？（4 分）

答：

（1）Bootloader：引导加载程序

（2）嵌入式系统（实验箱）启动后（打开电源，或者按 Reset 键），先执行 Bootloader，进行硬件和内存的初始化工作，然后加载 Linux 内核和根文件系统，完成 Linux 系统的启动。

（3）常见的 Boot Loader 有：U-Boot、vivi、Blob

9、请简述设备驱动程序与应用程序的区别。（4 分）

答：

① 应用程序一般有一个 main 函数，从头到尾执行一个任务。

② 设备驱动程序却不同，它没有 main 函数，通过使用宏 module_init()，将初始化函数加入内核全局初始化函数列表中，在内核初始化时执行驱动的初始化函数，从而完成驱动的初始化和注册，之后驱动便停止等待被应用软件调用；驱动程序中有一个宏 module_exit()注册退出处理函数，它在驱动退出时被调用。

③ 应用程序可以和 GLIBC 库连接，因此可以包含标准的头文件，比如<stdio.h>、<stdlib.h>。

④ 在设备驱动程序中是不能使用标准 C 库的，因此不能调用所有的 C 库函数，比如输出打印函数只能使用内核的 printk 函数，包含的头文件只能是内核的头文件，比如<linux/module.h>。

10、简述 Android NDK 开发过程，包括 Android NDK 开发环境的搭建、HelloJni 程序的编译和运行过程。（4 分）

答：

第一步搭建 NDK 开发环境：

（1）在 Vmware 虚拟机中，打开 Android NDK 系统用的 Ubuntu

（2）下载 Android NDK，得到源码包：android-ndk-r9-linux-x86.tar.bz2，并将其拷贝到 Ubuntu 中

（3）解压 NDK 源码，在虚拟机 Ubuntu 环境下执行：# tar xjvf android-ndk-r9-linux-x86.tar.bz2 -C /Android/
（4）配置环境变量

第二步：NDK 开发与编译。执行$NDK，编译 C/C++代码，生成 libhello-jni.so

第三步：将 libhello-jni.so 文件拷贝到电脑硬盘的 HelloJni 的..\app\libs\armeabi\目录中

第四步：打开 Android Studio，打开 HelloJni 工程，编译 HelloJni 工程

第五步：在 Android Studio 中，选择在实验箱上运行 HelloJni 工程

四、综合题（10 小题，共 30 分；在答题纸填写答案时请写上每小题的对应编号）

1、以下程序为 C 语言调用汇编语言的例子：

```
void enable_IRQ(void)
{
    int tmp;
    _______                                    //声明内联汇编代码
    {
        MRS tmp, CPSR
        BIC tmp, tmp, #0x80
        MSR CPSR_c, tmp
    }
}
```

请填写程序中空白的那一行（划线的部分）。（2 分）

答：

__asm

2、以下程序为 C 语言调用汇编语言的例子：

```
extern int add(int x, int y);                  //声明 add 为外部函数
void main()
{
int a=1, b=2, c;
c = add(a, b);
}

_____________          @声明 add 子程序将被外部函数调用
add:
ADD r0,r0,r1
MOV pc,lr
```

请填写程序中空白的那一行（划线的部分）。（2 分）

答：

EXPORT add

3、以下是 RS-485 驱动程序的头文件和全局变量，请问该程序的第 17）、第 18）行分别是做什么事情？（2 分）

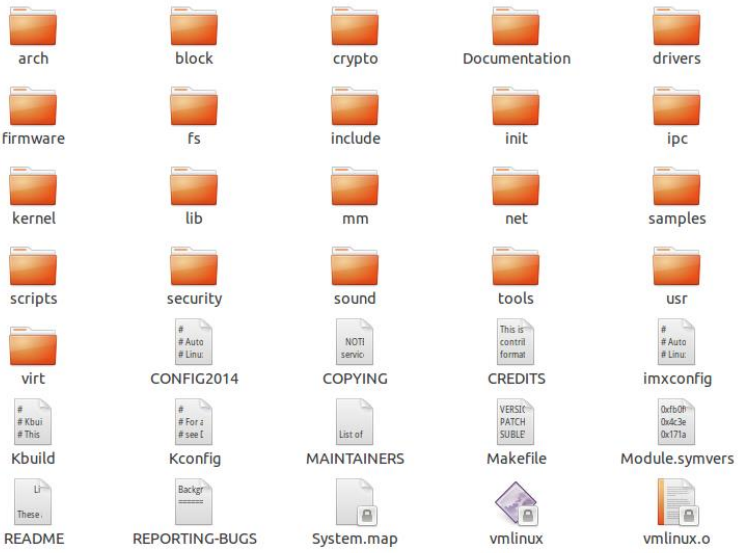
```
1)      #include <linux/kernel.h>
2)      #include <linux/module.h>
3)      #include <linux/init.h>
4)      #include <linux/errno.h>
5)      #include <linux/fs.h>
6)      #include <linux/cdev.h>
7)      #include <linux/types.h>
8)      #include <linux/device.h>
9)      #include <asm/system.h>
10)     #include <asm/uaccess.h>
11)     #include <linux/platform_device.h>
12)     #include <asm/irq.h>
13)     #include <linux/of.h>
14)     #include <linux/of_device.h>
15)     #include <linux/of_gpio.h>
16)     #define DRVNAME "UART485"
17)     #define UART485_MAJOR 30
18)     #define UART485_MINOR 0
19)     #define UART485_TX 1
20)     #define UART485_RX 0
```

答：

第 17）行：定义主设备号

第 18）行：定义次设备号

4、ARM-Linux 内核的代码目录结构如下（位于/home/uptech/fsl-6dl-source/kernel-3.14.28/目录下）：



请问，该目录结构中的 arch、block、drivers、fs、mm、net 子目录中分别存放什么内容？（3 分）

答：

1) arch ：包含和硬件体系结构相关的代码
2) block：块设备驱动程序
3) drivers ：设备驱动程序
4) fs：Linux 所支持的各种文件系统
5) mm：内存管理代码
6) net：网络相关代码

5、以下是 RS-485 驱动程序的模块初始化和模块退出函数，请填写程序中的 2 个空格部分的内容。（2 分）

```
static int __init gpio_uart485_init(void)
{
    printk("\n\n\nkzkuan___%s\n\n\n",__func__);
    return platform_driver_register(&gpio_uart485_device_driver);
}

static void __exit gpio_uart485_exit(void)
{
    printk("\n\n\nkzkuan___%s\n\n\n",__func__);
    platform_driver_unregister(&gpio_uart485_device_driver);
}
```

_____（1）_____(gpio_uart485_init);
_____（2）_____(gpio_uart485_exit);

答：
（1）module_init
（2）module_exit

6、我们在做实验时，通常采用挂载的方式，在实验箱的“超级终端（Xshell 2.0）”下，执行存放在 Ubuntu 中的可执行文件。此时运行实验箱的“超级终端（Xshell 2.0）”后，我们首先需要设置实验箱的 IP 地址，执行挂载命令，然后再运行可执行文件。设实验箱的 IP 地址为 59.77.5.120，Ubuntu 的 IP 地址为 59.77.5.122，需要将 Ubuntu 的“/imx6”目录挂载到实验箱的“/mnt”目录下，可执行文件（hello）存放在 Ubuntu 的 /imx6/whzeng/hello 目录下。请写出设置实验箱的 IP 地址的命令，实现挂载功能的命令，以及运行 hello 可执行文件的命令。（2 分）

答：

```
ifconfig eth0 59.77.5.120

mount    -t    nfs 59.77.5.122:/imx6          /mnt
```

7、如果我们不采用挂载的方式，而是采用下载的方式运行程序，即将 Ubuntu 中的可执行文件下载到实验箱中，再运行程序，请写出操作步骤（包括下载程序、运行程序）。设可执行文件（hello）存放在 Ubuntu 的/imx6/whzeng/hello/目录下，tftpd32.exe 文件（TFTP 服务）在 Windows 的 D:\UP-Tech\Linux 目录下，需要将可执行文件（hello）下载到实验箱的/home/root 目录中，Windows 系统的 IP 地址为 59.77.5.121。（4 分）

答：
第一步：将 Ubuntu 下的/imx6/whzeng/hello/hello 文件，复制到 Windows 的 D:\UP-Tech\Linux 目录下（使用 Samba 服务）

第二步：在 Windows 下运行“tftpd32.exe”（TFTP 服务），将 tftpd32 的 Current Directory 设为 D:\UP-Tech\Linux，Server interface 设为 59.77.5.121。

第三步：在实验箱的“超级终端（Xshell 2.0）”下，执行：

```
cd /home/root
tftp -gr hello 59.77.5.121
```

第四步：在实验箱的“超级终端（Xshell 2.0）”下，执行：

```
chmod 777 hello
./hello
```

8、以下为 RS-485 双机通讯程序的一部分，请问该程序中的第 7）、8）、9）、14）行分别是做什么事情？（4 分）

```
1)  void* receive(void * data)
2)  {
3)      int c;
4)      printf("RS-485 Receive Begin!\n");
5)      for(;;)
6)      {
7)          ioctl(fd485, UART485_RX);
8)          read(fdCOMS1,&c,1);
9)          write(1,&c,1);
10)         if(c == 0x0d)
11)             printf("\n");
12)         if(c == ENDMINITERM)
13)             break;
14)         ioctl(fd485, UART485_TX);
15)     }
16)     printf("RS-485 Receive End!\n");
17)     return NULL;
18) }
```

答：
第 7）行：设置 RS-485 为接收模式
第 8）行：从 RS-485 中读 1 个字符
第 9）行：将读到的字符在标准输出设备（显示器）上输出（显示）
第 14）行：设置 RS-485 为发送模式

9、以下为按键（小键盘）的主程序，请说明程序中的第 20）、22）、23）的具体功能是什么？ （3 分）

```
1)  #include <stdio.h>
2)  #include <sys/types.h>
3)  #include <sys/stat.h>
4)  #include <fcntl.h>
5)  #include <linux/input.h>
6)  #define NOKEY 0
7)  int main(int argc,char *argv[])
8)  {
9)      int keys_fd;
10)     char ret[2];
11)     struct input_event t;
12)     keys_fd = open(argv[1], O_RDONLY);
13)     if(keys_fd<=0)
14)     {
15)         printf("open %s device error!\n",argv[1]);
16)         return 0;
17)     }
18)     while(1)
19)     {
20)         if(read(keys_fd,  &t,  sizeof(t)) == sizeof(t))
21)         {
22)             if(t.type == EV_KEY)
23)                 if(t.value == 0 || t.value == 1)
24)                     printf("key %d %s\n",t.code,(t.value)?"Pressed":"Released");
25)         }
26)     }
27)     close(keys_fd);
28)     return 0;
29) }
```

答：
第 22）行：判断输入设备是不是键盘？
第 23）行：判断有没有键按下（0 表示按下），或者有没有键释放（1 表示释放）？
第 24）行：显示按下或释放的键的代码

10、以下为小键盘控制电子钟的主程序中的关键代码，请说明程序中的第 3）、4）、5）、12）、13）、19）行的具体功能是什么？ （6 分）

```
1)  int main(int argc, char *argv[])
2)  {
3)      keys_fd = open(KEYDevice, O_RDONLY);
4)      mem_fd = open("/dev/mem", O_RDWR);
5)      cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE |
    PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));
6)      pthread_create(&th_time, NULL, time_counter, 0);
7)      pthread_create(&th_key, NULL, key_input, 0);
8)      while(1)
9)      {
10)         for(i=0; i<8; i++)
11)         {
12)             *(cpld+(0xe6<<1)) = addr[i];
13)             *(cpld+(0xe4<<1)) = tube[number];
14)             usleep(1000);
15)         }
16)     }
17)     pthread_join(th_time, &retval);
18)     pthread_join(th_key, &retval);
19)     munmap(cpld,0x10);
20)     close(mem_fd);
21)     close(keys_fd);
22)     return 0;
}
```

答：
第 3）行：打开小键盘输入设备
第 4）行：打开数码管设备
第 5）行：内存映射操作，将数码管的内容映射到用户空间
第 12）行：设置数码管的位地址（即哪一个数码管）
第 13）行：设置数码管的段值（即显示什么内容，七段码或八段码）
第 19）行：解除内存映射