厦门大学《嵌入式系统》课程期末试卷

信息学院　　软件工程系　　2018 级　　软件工程专业

主考教师：曾文华　试卷类型：（A 卷答案）　考试时间：2020. 12. 29

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. ARM 指令有两种状态，分别是：（3）ARM 状态和（4）Thumb 状态。
4. RT-Linux 中的 RT 是指（5）硬实时。
5. μCLinux 是专门针对没有（6）MMU（存储管理单元）的处理器设计的。
6. make 编译工具读取的文件名称为（7）Makefile 文件。
7. Bootloader 是（8）引导加载程序。
8. 使用 mmap 系统调用，可以将（9）内核空间的地址映射到（10）用户空间。
9. 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（11）汇编语言来实现；Boot Loader 的阶段 2，通常用（12）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。
10. U-boot 2014 的目标结构中的 arch 子目录，存放的是与（13）体系结构相关的代码。
11. Flash Memory（闪存）有两种技术，分别是（14）NOR Flash 和（15）NAND Flash。
12. IMX6 实验箱 Linux 系统挂载的第一个文件系统是（16）根文件系统。
13. 设备的控制操作是通过调用 file_operations 结构体中的（17）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（18）请求函数完成的。
15. 网络设备驱动程序在/dev 目录下没有对应的设备文件，对网络设备的访问必须使用（19）套接字（Socket），而非读写设备文件。
16. 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（20）网络文件系统，59.77.5.122 是指（21）虚拟机的 IP 地址。
17. 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（22）编译的意思，“-o”中的 o 是（23）输出的意思。
18. 如果需要将实验箱的 IP 地址设置为 192.168.1.34，其命令是：（24）ifconfig eth0 192.168.1.34。
19. 假设虚拟机的/tftpboot 目录下有文件 inittab，可以在 IMX6 实验箱的超级终端上执行有关命令，通过 TFTP 方式，将 inittab 文件下载到实验箱中，这条命令为（假设虚拟机的 IP 地址为 192.168.1.56）（25）tftp -gr inittab 192.168.1.56。

20. 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）：（26）mknod /dev/lp0 c 6 0 。
21. Atlas 200 DK 是华为公司生产的 AI 开发套件，其核心是海思公司生产的（27）Ascend（昇腾）310 AI 处理器。
22. ModelArts 是面向（28）AI 开发者的一站式开发平台。
23. NDK 集成了交叉编译器，可以把你用 C、C++书写的代码，编译为（29）.so 格式的文件，在 Android 程序当中可以用 Java 语言（JNI）调用这些代码。
24. Android 的 NDK 开发工具集，是 Android 为了更方便 Android 程序开发者，通过（30）JNI 的机制，达到 Java 和本地 C/C++代码相互沟通的强有力武器。

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. CAN：Controller Area Network，控制器局域网络
2. CPSR：Current Program Status Register，当前程序状态寄存器
3. DVPP：Digital Vision Pre-Processing，数字视觉预处理模块
4. GPIO：General Purpose Input/Output，通用输入输出，通用可编程接口
5. JNI：Java Native Interface，Java 本地接口
6. JFFS：Journalling Flash File System，闪存设备日志型文件系统
7. OBS：Object Storage Service，对象存储服务
8. Ramfs：RAM File System，基于 RAM 的文件系统
9. TBE：Tensor Boost Engine，张量加速引擎
10. TFTP：Trivial File Transfer Protocol，简单文件传输协议

三、简答题（11 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. ARM Cortex-A、ARM Cortex-R、ARM Cortex-M 系列处理器分为针对什么应用场合？（3 分）

答：

（1）ARM Cortex-A 系列又称“高性能处理器”（Highest Performance），它是面向移动计算如智能手机、

平板电脑和服务器市场定制的高端处理器内核，支持了包括 Linux、Android、Windows 和 iOS 等系统必须的内存管理单元（MMU），而且也是与我们平时接触最为密切的存在。

（2）ARM Cortex-R 系列实时处理器（Real-Time Processing）为要求可靠性、高可用性、容错功能、可维护性和实时响应的嵌入式系统提供高性能计算解决方案。Cortex-R 系列处理器通过已经在数以亿计的产品中得到验证的成熟技术提供极快的上市速度，并利用广泛的 ARM 生态系统、全球和本地语言以及全天候的支持服务，保证快速、低风险的产品开发。

（3）Cortex-M 系列针对成本和功耗敏感（Lowest Power, Lower Cost）的 MCU 和终端应用（如智能测量、人机接口设备、汽车和工业控制系统、大型家用电器、消费性产品和医疗器械）的混合信号设备进行过优化。

3. IMX6 嵌入式教学科研平台（实验箱）的主 CPU 和从 CPU 分别采用哪个 ARM 处理器的产品？（2 分）

答：

主 CPU 采用：Cortex-A9：Freescale i.MX6

从 CPU 采用：Cortex-M3：STM32F103

4. 请简述设备驱动程序与应用程序的区别。（2 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>。

5. Linux 设备驱动程序开发调试有两种方法？（2 分）

答：第一种是直接编译到内核，再运行新的内核来测试；第二种是编译为模块的形式，单独加载运行调试。

6. IMX6 嵌入式教学科研平台（实验箱）开机后（打开电源或者按 Reset 键后），将完成哪些任务？（2 分）

答：开机后，先执行 Bootloader，进行硬件和内存的初始化工作，然后加载 Linux 内核和根文件系统映像，完成 Linux 系统的启动。

7. IMX6 嵌入式教学科研平台（实验箱）程序的执行方式有二种，一种是下载方式，另一种是挂载方式。请举例说明这两种方式的区别。（2 分）

答：下载方式：使用 FTP、TFTP 等软件，利用宿主机（虚拟机，电脑）与目标机（实验箱，平台）的网络硬件进行，此种方法通常是将宿主机端编译好的目标机可执行的二进制文件通过网线或串口线下载固化到目标机的存储器(FLASH)中。在目标机嵌入式设备存储资源有限的情况下受到存储容量的限制，因此，在调试阶段通常的嵌入式开发经常使用 NFS 挂载的方式进行，而在发布产品阶段才使用下载方式。

挂载方式：利用宿主机端（虚拟机，电脑）NFS 服务，在宿主机端创建一定权限的 NFS 共享目录，在目标机端（实验箱，平台）使用 NFS 文件系统挂载该目录，从而达到网络共享服务的目的。这样做的好处是不占用目标机存储资源，可以对大容量文件进行访问。缺点是由于实际并没有将宿主机端文件存储到目标机存储设备上，因此掉电不保存共享文件内容。通常在嵌入式开发调试阶段，采用 NFS 挂载方式进行。

8. 宿主机（PC 机）与目标板（IMX6 实验箱）的连接方式有哪些？（2 分）

答：

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

9. 什么是 Android NDK？简述 Android NDK 程序的开发过程，包括开发环境的搭建、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

10. IMX6 嵌入式教学科研平台（实验箱）从 CPU（第二个 CPU）的软件开发工具是什么？请简述该软件开发工具的安装过程。从 CPU 的 LED 灯实验程序是怎么执行的？（4 分）

答：从 CPU 采用 MDK 嵌入式软件开发工具进行开发。

MDK 的安装过程包括：（1）安装 MDK5 主体；（2）安装对应 MCU 的 Pack 支持包；（3）破解；（4）安装调试工具 J-Link。

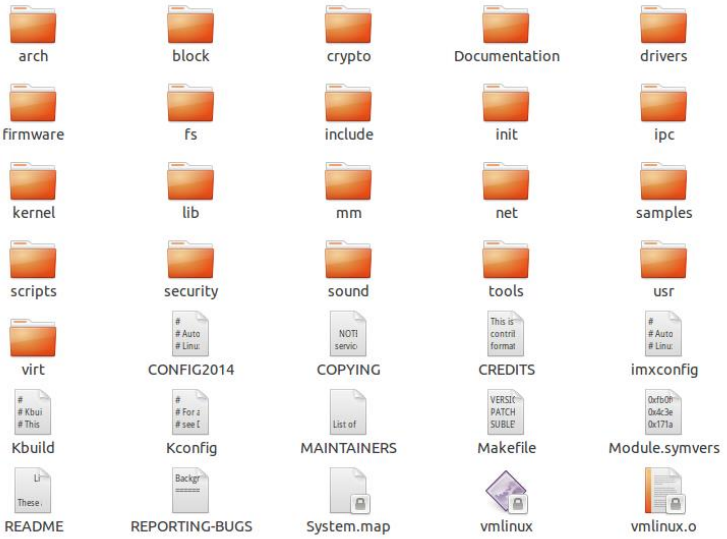
LED 灯实验程序的执行：首先打开 MDK5；然后打开 LED 灯工程，编译生成目标文件；下载目标文件到实验箱；按实验箱从 CPU 的 Reset 键，即可执行程序。

11. “云+端协同实验 —— 猫狗图像识别”，云部分完成什么任务？端部分完成什么任务？（4 分）

答：云部分在 ModelArts 平台上，使用 Notebook 构建模型的方法，在 Jupyter 开发环境中编写并执行代码，生成模型 vgg16trained.pb；端部分在 Mind Studio 平台上，首先将 vgg16trained.pb 模型转换为.om 模型，然后使用 sample-classification 工程，完成猫和狗的图像识别。

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. 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）：

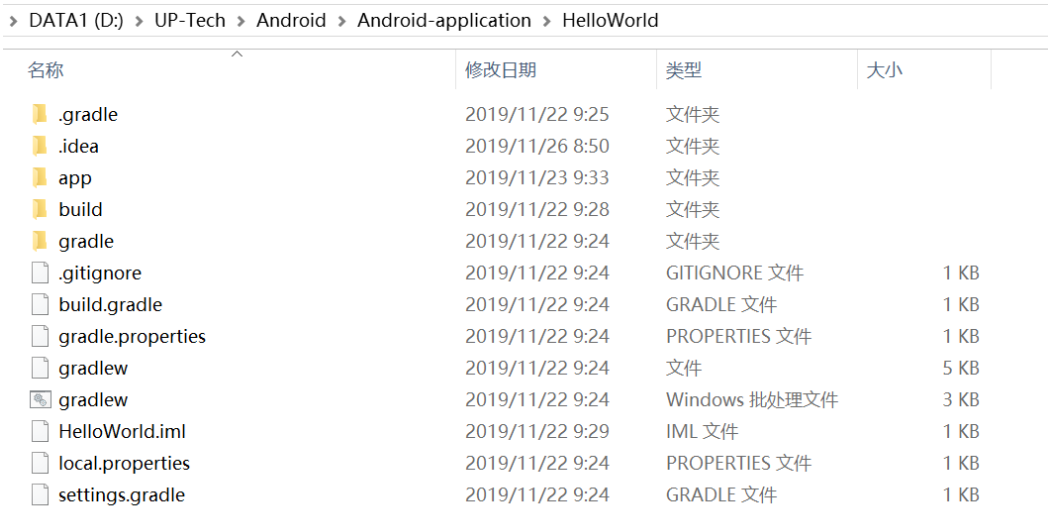


请问，该目录结构中的 arch、block、drivers、fs、mm、net 子目录中分别存放什么内容？（3 分）

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

2. Android Hello World 程序的工程文件夹如下：



请问该文件夹的 5 个子文件夹中存放什么内容？（3 分）

答：

（1）.gradle 和.idea：这两个目录下放置的都是 Android Studio 自动生成的一些文件，我们无须关心，

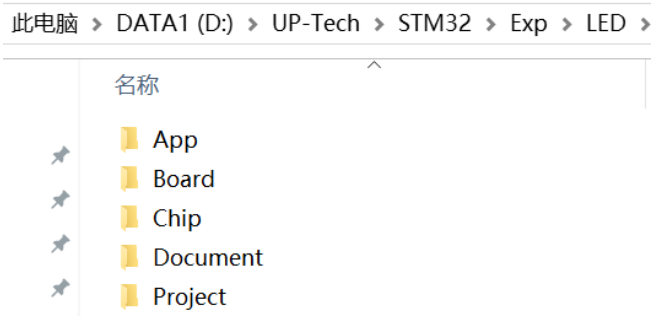
也不要去手动编辑。

（2）app：项目中的代码、资源等内容几乎都是放置在这个目录下的，我们后面的开发工作也基本都是在这个目录下进行的，待会儿还会对这个目录单独展开进行讲解。

（3）build：这个目录你也不需要过多关心，它主要包含了一些在编译时自动生成的文件。

（4）gradle：这个目录下包含了 gradle wrapper 的配置文件，使用 gradle wrapper 的方式不需要提前将 gradle 下载好，而是会自动根据本地的缓存情况决定是否需要联网下载 gradle。Android Studio 默认没有启动 gradle wrapper 的方式，如果需要打开，可以点击 Android Studio 导航栏 --> File --> Settings --> Build，Execution，Deployment --> Gradle，进行配置更改。

3. 实验箱从 CPU 的 LED 灯程序的工程文件夹如下：



请问该文件夹的 5 个子文件夹中存放什么内容？（3 分）

答：（1）App 存放用户程序

（2）Board 存放开发板（实验箱）的驱动程序

（3）Chip 存放 STM32 芯片驱动程序

（4）Document 存放工程说明文档

（5）Project 存放工程文件

4. 以下为 CAN 总线双机通信中接收程序的主函数，请什么程序中第 10）、12）、15）、18）、21）行的含义。（5 分）

```
1)      int main(int argc, char *argv[])
2)      {
3)        int s, nbytes, nbytes_send;
4)        struct sockaddr_can addr;
5)        struct ifreq ifr;
6)        struct can_frame frame_rev;
7)        struct can_frame frame_send;
8)        struct can_filter rfilter[1];
9)        int len = sizeof(addr);
10)       s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
```

```
11)     strcpy(ifr.ifr_name, "can0" );
12)     ioctl(s, SIOCGIFINDEX, &ifr);
13)     addr.can_family = AF_CAN;
14)     addr.can_ifindex = ifr.ifr_ifindex;
15)     bind(s, (struct sockaddr *)&addr, sizeof(addr));
16)     rfilter[0].can_id = 0x00;
17)     rfilter[0].can_mask = CAN_SFF_MASK;
18)     setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));
19)   while(1)
20)     {
21)         nbytes = read(s, &frame_rev, sizeof(frame_rev));
22)         if(nbytes > 0)
23)         {
24)             printf("ID=0x%X                                                    DLC=%d
      data[0]=%X\n",frame_rev.can_id,frame_rev.can_dlc,frame_rev.data[0]);
25)         }
26)     }
27)     close(s);                                                    /
28)     return 0;
29)   }
```

答：第 10）创建套接字
第 12）行指定 can0 设备
第 15）行将套接字与 can0 绑定
第 18）行设置过滤规则，只接收表示符等于 0x00 的报文
第 21）行接收报文

5. 以下是 LED 灯驱动程序的头文件和全局变量，请问该程序的第 15)、16)、17）行分别是做什么事情？（3 分）

```
1)    #include <linux/module.h>
2)    #include <linux/kernel.h>
3)    #include <linux/fs.h>
4)    #include <linux/init.h>
5)    #include <linux/miscdevice.h>
6)    #include <linux/delay.h>
7)    #include <linux/device.h>
8)    #include <linux/cdev.h>
9)    #include <linux/platform_device.h>
10)   #include <asm/irq.h>
11)   #include <linux/of.h>
12)   #include <linux/of_device.h>
13)   #include <linux/of_gpio.h>
14)   MODULE_LICENSE("GPL");
15)   #define DEVICE_NAME       “ledtest”
16)   #define DEVICE_MAJOR 231
```

```
17)     #define DEVICE_MINOR    0
18)     struct cdev *mycdev;
19)     struct class *myclass;
20)     dev_t devno;
21)     static unsigned int led_table [4] = {};
```

答：

第 15）行：定义设备名

第 16）行：定义主设备号

第 17）行：定义次设备号

6. 以下是 RS-485 驱动程序的模块初始化和模块退出函数，请填写该程序中 2 个空格（第 11）、12）行）中的内容：（2 分）

```
1)      static int __init gpio_uart485_init(void)
2)      {
3)      printk("\n\n\nkzkuan___%s\n\n\n",__func__);
4)      return platform_driver_register(&gpio_uart485_device_driver);
5)      }

6)      static void __exit gpio_uart485_exit(void)
7)      {
8)      printk("\n\n\nkzkuan___%s\n\n\n",__func__);
9)      platform_driver_unregister(&gpio_uart485_device_driver);
10)     }

11)     _____（1）_____(gpio_uart485_init);
12)     _____（2）_____(gpio_uart485_exit);
```

答：

（1）module_init

（2）module_exit

7. 以下为按键（小键盘）程序的主函数，请说明程序中的第 14）、16）、17）的具体功能是什么？（3 分）

```
1)  int main(int argc,char *argv[])
2)  {
3)      int keys_fd;
4)      char ret[2];
5)      struct input_event t;
6)      keys_fd = open(argv[1], O_RDONLY);
7)      if(keys_fd<=0)
8)      {
9)          printf("open %s device error!\n",argv[1]);
10)          return 0;
11)     }
12)     while(1)
13)     {
14)         if(read(keys_fd,  &t,  sizeof(t)) == sizeof(t))
15)         {
16)             if(t.type == EV_KEY)
17)                 if(t.value == 0 || t.value == 1)
18)                     printf("key %d %s\n",t.code,(t.value)?"Pressed":"Released");
19)         }
20)     }
21)     close(keys_fd);
22)     return 0;
23) }
```

答：

第 14）行：读取输入设备的值，并判断是否读取成功

第 16）行：判断输入设备是不是键盘？

第 17）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）？

8. 以下为步进电机程序的主函数，请说明程序中的第 6）、7）、14）、16）行的具体功能是什么？（4 分）

```
1)  int main(int argc, char *argv[])
2)  {
3)      unsigned char data;
4)      int mem_fd;
5)      unsigned char *cpld;
6)      mem_fd = open("/dev/mem", O_RDWR);
```

```
7)      cpld = (unsigned char*)mmap(NULL,(size_t)0x04,PROT_READ | PROT_WRITE |
                PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));
8)      if(cpld == MAP_FAILED)
9)          return;
10)     while(1)
11)     {
12)         printf("请输入步进电机状态：\n");
13)         scanf("%d",&data);
14)         *(cpld+(0xe2<<1)) = data;
15)     }
16)     munmap(cpld,0x04);
17)     close(mem_fd);
18)     return 0;
19) }
```

答：第 6）打开内存设备

第 7）行内存映射

第 14）行控制步进电机转的状态（正转、反转、停止）

第 16）行解除内存映射

9. 以下为 Atlas 200 DK“创建首个 AI 应用”中 graph.config 文件的内容，请什么该文件中第 8）、12）、25）、46）行的作用。（4 分）

```
1)      graphs {
2)        priority: 0
3)        engines {
4)          id: 611
5)          engine_name: "DataInput"
6)          side: HOST
7)          thread_num: 1
8)          so_name: "./libHost.so"
9)          ai_config {
10)           items {
11)             name: "path"
12)             value: "/home/ascend/AscendProjects/MyApp/resource/data/"
```

```
            }
            items {
                name: "target"
                value: "RC"
            }
        }
    }
    engines {
        id: 814
        engine_name: "ImagePreProcess"
        side: DEVICE
        thread_num: 1
        so_name: "./libDevice.so"
        ai_config {
            items {
                name: "resize_width"
                value: "224"
            }
            items {
                name: "resize_height"
                value: "224"
            }
        }
    }
    engines {
        id: 226
        engine_name: "MindInferenceEngine"
        side: DEVICE
        thread_num: 1
        so_name: "./libDevice.so"
        ai_config {
            items {
                name: "model_path"
                value: "/home/ascend/MindStudio-ubuntu/samples/modelfile/resnet18.om"
            }
```

```
48)             }
49)         }
50)         engines {
51)             id: 368
52)             engine_name: "SaveFilePostProcess"
53)             side: HOST
54)             thread_num: 1
55)             so_name: "./libHost.so"
56)             ai_config {
57)             }
58)         }
59)         connects {
60)             src_engine_id: 611
61)             src_port_id: 0
62)             target_engine_id: 814
63)             target_port_id: 0
64)         }
65)         connects {
66)             src_engine_id: 814
67)             src_port_id: 0
68)             target_engine_id: 226
69)             target_port_id: 0
70)         }
71)         connects {
72)             src_engine_id: 226
73)             src_port_id: 0
74)             target_engine_id: 368
75)             target_port_id: 0
76)         }
77)     }
```

答：第 8）行定义 Host 库文件（libHost.so）

第 12）行定义输入数据的路径

第 25）行定义 Device 库文件（libDevice.so）

第 46）行定义离线模型文件