厦门大学《嵌入式系统》课程期末试卷

信息学院　　软件工程系　　2021 级　　软件工程专业

主考教师：曾文华　试卷类型：（A 卷）　考试时间：2024.1.10

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. IMX6 嵌入式教学科研平台（实验箱）有二个 CPU，一个是主 CPU，采用飞思卡尔公司生产的基于 ARM（1）的最新单核的 IMX6DL 嵌入式微处理器（Freescale IMX6DL）；另一个是从 CPU，采用 ARM（2）内核架构的意法半导体公司生产的 STM32F103 芯片。
2. 嵌入式系统主要的存储设备为（3）和（4）。
3. ARM 处理器有两种状态，分别是（5）状态和（6）状态。
4. Ubuntu 是 Linux 系统最受欢迎的（7）。
5. μCLinux 中的 μ 表示（8），C 表示（9），μCLinux 是专门针对没有（10）的处理器设计的。
6. RT-Linux 是具有（11）特性的多任务操作系统；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（12），把标准的 Linux 内核作为（12）的一个进程与用户进程一起调度。
7. 嵌入式 Linux 系统启动后，先执行（13），进行硬件和内存的初始化工作，然后加载（14）和（15），完成 Linux 系统的启动。
8. Linux 的设备驱动程序开发调试有两种方法，一种是直接编译到（16），另一种是编译为（17）的形式；第一种方法效率较（18），第二种方式效率较（19）。
9. Linux 抽象了对硬件的处理，所有的硬件设备都可以作为普通文件一样对待，可以使用标准的系统调用接口来完成对设备的打开（open）、关闭（close）、读写（read、write）和（20），驱动程序的主要任务是实现这些系统调用函数。
10. Qt 是一个由 Qt Company 开发的跨平台 C++（21）应用程序开发框架。
11. Boot Loader 的操作模式有两种，分别是（22）和（23）。
12. 在 Ubuntu 上执行 make 命令（交叉编译生成可执行文件）前，需要先执行“source /opt/fsl-imx-wayland/4.9.88-2.0.0/environment-setup-cortexa9hf-neon-poky-linux-gnueabi”命令，该命令的作用是（24）。
13. 使用 mmap 系统调用（mmap()函数），可以将（25）空间的地址映射到（26）空间。
14. 块设备没有 read 和 write 操作函数，对块设备的读写是通过（27）函数完成的。
15. 对网络设备的访问必须使用（28），而非读写设备文件。
16. Android 系统的底层是由（29）操作系统作为内核。
17. Atlas 200 DK 是华为公司生产的面向 AI 应用的开发者套件，其核心是（30）处理器。

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. Android NDK
2. CAN
3. CPSR
4. GPIO
5. I^2C

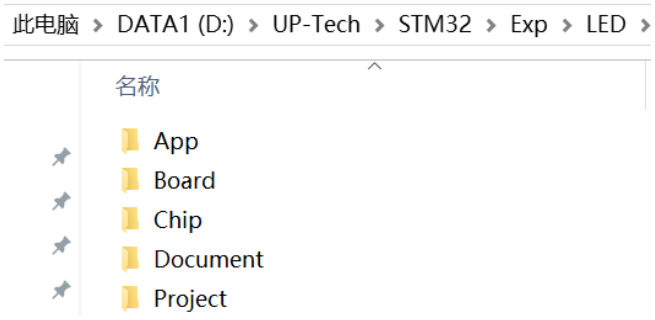
6. JTAG
7. JFFS3
8. NFC
9. Ramfs
10. SPI

三、简答题（7 小题，共 20 分；在答题纸填写答案时请写上每小题的对应编号）

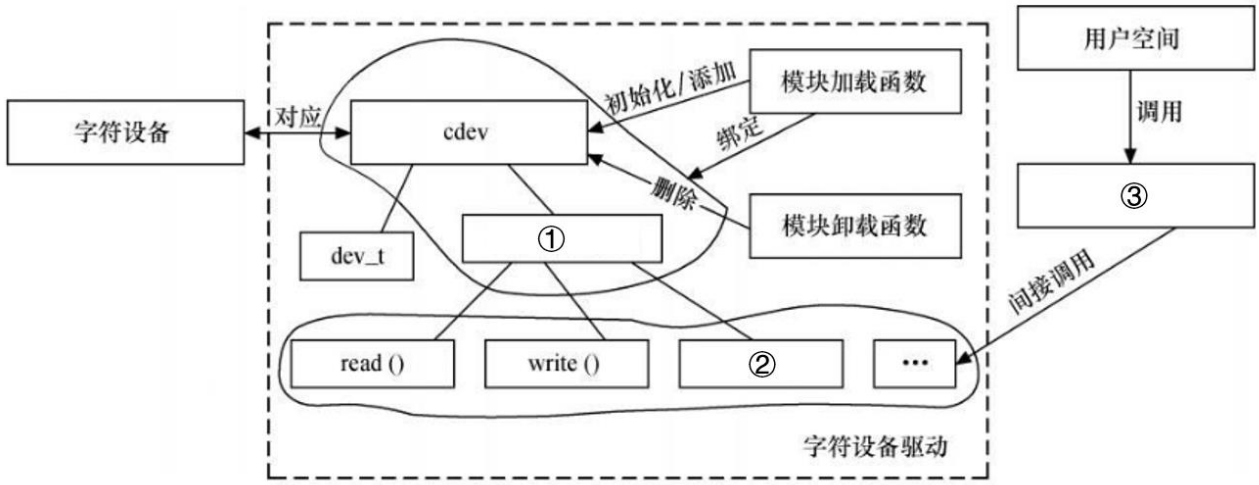
1. （2 分）常见的嵌入式操作系统有哪些？
2. （3 分）ARM Cortex-A、ARM Cortex-R、ARM Cortex-M 系列处理器分别针对什么应用场合？
3. （2 分）什么是交叉开发（交叉编译）？
4. （3 分）什么是 Boot Loader？其作用是什么？常见的 Boot Loader 有哪些？
5. （3 分）请写出 ARM 指令的格式。
6. （4 分）宿主机与目标机通常有 4 种连接方式，请结合 IMX6 实验箱分别说明每一种连接方式的具体内容和应用场景。
7. （3 分）简述在 IMX6 实验箱上开发 Android NDK 程序的步骤。

四、综合题（10 小题，共 40 分；在答题纸填写答案时请写上每小题的对应编号）

1. （5 分）STM32 LED 灯实验程序的工程项目文件夹如下，请问该工程项目文件夹的 5 个子文件夹中分别存放什么内容？



2. （3 分）下图为字符设备驱动框架，请填写图中 3 个空白方框（①②③）的内容。



3. （3 分）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，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。

4. （5 分）已知当前目录下有 pthread.c 和 Makefile 两个文件，其中 Makefile 文件的内容如下：

```
CC=arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtune=cortex-a9 --sysroot=/opt/fsl-imx-wayland/4.9.88-2.0.0/sysroots/cortexa9hf-neon-poky-linux-gnueabi
EXTRA_LIBS += -lpthread
EXP_INSTALL = install -m 755
INSTALL_DIR = ../bin
EXEC = ./pthread
OBJS = pthread.o
all: $(EXEC)
$(EXEC): $(OBJS)
    $(CC) -o $@ $(OBJS)   $(EXTRA_LIBS)
install:
    $(EXP_INSTALL) $(EXEC) $(INSTALL_DIR)
clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
```

请问在当前目录下分别执行 make、make install、make clean 命令，分别会显示什么结果？如果要将编译后的可执行文件能够在 Ubuntu 环境（x86 环境）下运行，请问如何修改 Makefile 文件？

假设：pthread.c 文件是正确的（不会出现编译错误）。在写显示结果时请用 CC1 代替 arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtune=cortex-a9 --sysroot=/opt/fsl-imx-wayland/4.9.88-2.0.0/sysroots/cortexa9hf-neon-poky-linux-gnueabi -O2 -pipe -g -feliminate-unused-debug-types；用 CC2 代替 arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtune=cortex-a9 --sysroot=/opt/fsl-imx-wayland/4.9.88-2.0.0/sysroots/cortexa9hf-neon-poky-linux-gnueabi。

5. （5 分）以下 2 个程序为 C 语言调用汇编语言的程序，请问程序 1 和程序 2 分别属于什么调用形式（什么汇编）？并补充 2 个程序中 3 个划线处（①②③）的内容。

程序 1：

```
____①____ add                          @声明 add 子程序将被外部函数调用
add:
    ADD r0,r0,r1
    MOV pc,lr
____②____ 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);
}
```

程序 2：

```
void enable_IRQ(void)
{
    int tmp;
    ____③_____                                    //声明内联汇编代码
    {
        MRS tmp, CPSR
        BIC tmp, tmp, #0x80
        MSR CPSR_c, tmp
    }
}
```

6. （2 分）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 2 个划线处（①②）的内容。

```
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);
```

7. （3 分）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）、14）行分别是做什么事情？

```
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) }
```

8. （5 分）以下程序为读取小键盘按键值的主程序，请问该程序中的第 5）、13）、15）、16）、18）行分别是完成什么任务？

```
1)  int main(int argc,char *argv[])
2)  {
3)      int keys_fd;
4)      struct input_event t;
5)      keys_fd = open(KEYDevice, O_RDONLY);
6)      if(keys_fd <= 0)
7)      {
8)          printf("open key device error!\n");
9)          return 0;
10)     }
11)     while(1)
12)     {
13)         if(read(keys_fd,&t,sizeof(t)) == sizeof(t))
14)         {
15)             if(t.type == EV_KEY)
16)                 if(t.value == 0)
17)                 {
18)                     printf("%c\n",key_value(t.code));
19)                 }
20)         }
21)     }
22)     close(keys_fd);
23)     return 0;
24) }
```

9. （4 分）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 5）、12）、13）、19）行的具体功能是什么？

```
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;
}
```

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

```
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%XDLC=%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)     }
```