



厦门大学《嵌入式系统》课程期末试卷

信息学院 软件工程系 2020 级 软件工程专业

主考教师：曾文华 试卷类型：(A 卷) 考试时间：2023. 2. 15

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空格的编号）

1. Ubuntu 是 Linux 系统最受欢迎的 (1)。
2. 嵌入式系统的前身通常称为 (2)。
3. ARM Cortex-A 系列处理器又称为 (3) 处理器，ARM Cortex-R 系列处理器是针对 (4) 要求高的嵌入式系统提供的解决方案，ARM Cortex-M 系列处理器是针对 (5) 敏感的嵌入式系统提供的解决方案。
4. ARM 指令系统的每条指令字长都是 32 位，只有 Load 和 Store 指令才能访问内存，采用 ARM 指令系统的计算机是典型的 (6) 体系结构计算机。
5. μ CLinux 是专门针对没有 (7) 的处理器设计的。
6. RT-Linux 中的 RT 是指 (8)。
7. 嵌入式系统很少使用 IDE 硬盘，而是选用 (9) 代替 IDE 硬盘。
8. “mount -t nfs 192.168.33.129:/imx6 /mnt” 命令中的 nfs 的中文全称是 (10)，192.168.33.129 是 (11) 的 IP 地址。
9. 嵌入式 Linux 系统启动后，先执行 (12)，进行硬件和内存的初始化工作，然后加载 (13) 和 (14)，完成 Linux 系统的启动。
10. “arm-poky-linux-gnueabi-gcc” 是 ARM 平台的 (15) 工具。
11. Atlas 200 DK 是华为公司生产的面向 (16) 应用的开发者套件，其核心是 (17) 处理器。
12. 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 (18)，另一种是编译为 (19) 的形式；第一种方法效率较低，第二种方式效率较高。
13. 设备驱动程序是 Linux 系统内核和机器硬件之间的接口，设备驱动程序为应用程序屏蔽了硬件的细节，在应用程序看来，硬件设备只是一个 (20)。
14. Linux 抽象了对硬件的处理，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 (21)，驱动程序的主要任务是实现这些系统调用函数。
15. 使用 mmap 系统调用，可以将 (22) 空间的地址映射到 (23) 空间。
16. Android 的软件架构采用了分层结构，由上至下分别为：Application 应用层、Application Framework 应用框架层、Android Runtime & Libraries 运行时库和本地库层、(24) 内核层。
17. 字符设备最关键的数据结构是 (25)。
18. 块设备没有 read 和 write 操作函数，对块设备的读写是通过 (26) 函数完成的。
19. 对网络设备的访问必须使用 (27)，而非读写设备文件。
20. U-boot 2017.03 的目标结构中的 arch 子目录，存放的是与 (28) 相关的代码。
21. 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 (29) 的意思，“-o”中的 o 是 (30) 的意思。

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
2. CPSR
3. EDA
4. GPIO
5. I2C
6. IP 核
7. JFFS3
8. NFC
9. SoC
10. SPI

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. （3 分）常见的嵌入式操作系统有哪些？
2. （2 分）在 Ubuntu 上执行 make 命令前，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，请问该命令的作用是什么？
3. （2 分）什么是本地开发（本地编译）？什么是交叉开发（交叉编译）？
4. （3 分）嵌入式系统开发中通常有两种方式运行应用程序，请说明这两种方式的名称和具体过程。
5. （3 分）请比较 NOR Flash 存储器和 NAND Flash 存储器。
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）请简述设备驱动程序与应用程序的区别。
8. （4 分）Android HelloWorld 工程可以在 4 个地方运行，请说出这 4 个地方的具体名称。
9. （3 分）简述在 IMX6 实验箱上开发（运行）Android NDK 程序的具体步骤。
10. （3 分）简述云（ModelArts）+ 端（Atlas 200 DK）协同猫狗识别实验的具体步骤。

四、综合题（8 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. （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，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。假设实验箱的 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. （2 分）以下程序为汇编语言调用 C 语言的程序，请补充程序中 2 个划线处的内容。

```
int add(int x, int y)
{
    return(x+y);
}
```

_____(1)____ add @声明要调用的 C 函数

MOV r0, 1
 MOV r1, 2 @通过 r0、r1 传递参数（参数传递规则）
 (2) add @调用 C 函数 add；返回结果由 r0 带回（子程序返回结果规则）

3. （2 分）以下程序是 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 个空格部分的内容。

```
static int __init gpio_uart485_init(void){
    printk("\n\nnkzkuan__%s\n\n", __func__);
    return platform_driver_register(&gpio_uart485_device_driver);
}
static void __exit gpio_uart485_exit(void){
    printk("\n\nnkzkuan__%s\n\n", __func__);
    platform_driver_unregister(&gpio_uart485_device_driver);
}
(1) (gpio_uart485_init);
(2) (gpio_uart485_exit);
```

4. （3 分）以下程序为 RS-485 双机通讯程序的一部分，请问该程序中的第 5)、6)、12) 行分别是做什么事情？

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

5. （5 分）以下程序为 CAN 总线双机通信中接收程序的主函数，请问该程序中第 9)、11)、14)、17)、19) 行的含义。

```
1)   int main(int argc, char *argv[]) {
2)       int s, nbytes, nbytes_send;
3)       struct sockaddr_can addr;
4)       struct ifreq ifr;
5)       struct can_frame frame_rev;
6)       struct can_frame frame_send;
7)       struct can_filter rfilter[1];
```

```

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

```

6. （4 分）以下程序为小键盘程序的主函数，请说明该程序中第 11)、12)、13)、14) 行的具体功能是什么？

```

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

```

7. （6分）以下程序为小键盘控制的电子钟程序的主函数，请说明该程序中第 7)、12)、13)、32)、34)、44) 行的具体功能是什么？

```

1)  int main(int argc, char *argv[]){
2)      int i,number,mem_fd;
3)      void * retval;
4)      pthread_t th_time,th_key;
5)      flag_timecounter = 0;
6)      key_state = 0;
7)      keys_fd = open(KEYDevice, O_RDONLY);
8)      if(keys_fd <= 0){
9)          printf("open key device error!\n");
10)         return -1;
11)     }
12)     mem_fd = open("/dev/mem", O_RDWR);
13)     cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE |
        PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));
14)     if(cpld == MAP_FAILED) return -1;
15)     for(i=0; i<8; i++){
16)         *(cpld+(0xe6<<1)) = addr[i];
17)         *(cpld+(0xe4<<1)) = tube[24];
18)     }
19)     for(i=0; i<8; i++) leddisplay[i] = 24;
20)     pthread_create(&th_time, NULL, time_counter, 0);
21)     pthread_create(&th_key, NULL, key_input, 0);
22)     while(1){
23)         if(flag_timecounter == 1){
24)             leddisplay[0] = hour/10;
25)             leddisplay[1] = hour - leddisplay[0]*10;
26)             leddisplay[3] = minute/10;
27)             leddisplay[4] = minute - leddisplay[3]*10;
28)             leddisplay[6] = second/10;
29)             leddisplay[7] = second - leddisplay[6]*10;
30)         }
31)         for(i=0; i<8; i++){
32)             *(cpld+(0xe6<<1)) = addr[i];
33)             number = leddisplay[i];
34)             *(cpld+(0xe4<<1)) = tube[number];
35)             usleep(1000);
36)         }
37)     }
38)     pthread_join(th_time, &retval);
39)     pthread_join(th_key, &retval);
40)     for(i=0; i<8; i++){
41)         *(cpld+(0xe6<<1)) = addr[i];
42)         *(cpld+(0xe4<<1)) = tube[24];
43)     }
44)     munmap(cpld,0x10);

```

```

45)    close(mem_fd);
46)    close(keys_fd);
47)    return 0;
48) }

```

8. (5分) 以下程序为 NFC 实验的主程序，请说明该程序中第 6)、12)、19)、21)、23) 行的具体功能是什么？

```

1)  int main(int argc, char *argv[]) {
2)      unsigned char uartdata[25],c;
3)      unsigned long uid;
4)      int re,i,COMDevice;
5)      COMDevice = com_init(B115200);
6)      re = write(COMDevice,wakeup,sizeof(wakeup));
7)      if(re>0)
8)          printf("write ok\n");
9)      else
10)         printf("write error\n");
11)     while(1) {
12)         re = read(COMDevice,&c,1);
13)         if(re==1) {
14)             for(i=0;i<24;i++){
15)                 uartdata[i]=uartdata[i+1];
16)             }
17)             uartdata[24] = c;
18)             printf("0x%x ",c);
19)             if(uartdata[24-3] == 0xd5 && uartdata[24-2] == 0x15) {
20)                 printf("\n");
21)                 write(COMDevice,getUID,sizeof(getUID));
22)             }
23)             else if(uartdata[0]==0x00 && uartdata[1]==0x00 && uartdata[2]==0xFF &&
uartdata[3]==0x00 && uartdata[4]==0xff && uartdata[5]==0x00 &&
uartdata[6]==0x00 && uartdata[7] == 0x00 && uartdata[8] == 0xFF && uartdata[12] == 0x4b &&
uartdata[18] == 0x04 && uartdata[24] == 0x00 ) {
24)                 printf("\n\n");
25)                 uid = uartdata[19]<<24 | uartdata[20]<<16 | uartdata[21]<<8 | uartdata[22];
26)                 printf("uid=0x%x\n",uid);
27)                 write(COMDevice,getUID,sizeof(getUID));
28)             }
29)         }
30)     }
31)     close(COMDevice);
32)     return 0;
33) }

```