

任课教师: _____
学号: _____
姓名: _____
班级: _____

订线
订线
订线
订线

西安电子科技大学

考试时间 120 分钟

试 题

| 题号 | 一 | 二 | 三 | 四 | 总分 |
|----|---|---|---|---|----|
| 分数 | | | | | |

1. 考试形式: 闭卷 ☒ 开卷 ☐ ; 2. 本试卷共四大题, 满分 100 分;

3. 考试日期: 年 月 日; (答题内容请写在装订线外)

第 11 页图一是嵌入式系统常用的开发环境模型。试题中部分试题会用到该图中的信息

一、简答题 (共 40 分)

1. 嵌入式开发环境搭建相关问题 (20 分)

(1) 嵌入式开发过程中为什么要搭建交叉编译环境? (3 分)

(2) 宿主机与目标机之间常用的调试接口有 JTAG、RS-232、以太网接口, 在 Bootloader 未下载之前, 可以用哪些接口调试目标机? (3 分)

(3) 如果要用串口在宿主机与目标机之间进行通信, 在宿主机上通常要使用什么软件? 在使用该软件进行通信时要设置哪些参数, 在图一系统中应分别设置成什么? (3 分)

(4) 给出将宿主机 eth1 的 IP 地址设为 192.168.0.100、子网掩码设为 255.255.255.0 的完整命令。如果执行该命令时总是提示权限问题, 该如何应对? (3 分)

(5) 为了能通过 tftp 方式将编译好的文件下载到目标机上, 在宿主机上要搭建什么服务器? 如果目标机子网掩码为 255.255.255.0, 需要把目标机的 IP 地址设置为什么? 在目标机上如何判断是否与主机之间的网络是否已经连通? (8 分)

2. 阅读 makefile 文件并回答问题（8 分）

```
objects = main.o print.o
CFLAGS = -O3
CC=gcc
helloworld : $(objects)
    CC -o helloworld $(objects)
main.o : main.c print.h
    CC $(CFLAGS) -c main.c
print.o : print.c print.h
    CC $(CFLAGS) -c print.c
clean :
    rm helloworld $(objects)
```

（1）假定执行 make 命令没有出现错误，且对 main.c、 print.c 均进行过修改，则可以产生哪些新的文件？（2 分）

（2）如果想用 GDB 工具对最终生成的可执行程序进行调试，应如何修改该 makefile 文件。（2 分）

（3）利用什么命令删除所有新生成的文件？（2 分）

（4）如果要用生成在目标机上运行的程序，应如何修改 makefile 文件？（2 分）

3. 解释 tfkm 与 vm 的文件类型和访问权限（2 分）

```
[root@localhost os]# ls -l
-rwxr-xr-x 1 root root 80 06-13 11:01 tfkm
drwxr-xr-x 2 root root 4096 05-25 11:57 vm
```

4. 在 Linux 的 API 函数 signal 的声明为:

```
void (*signal( int sig, void (* handler)( int )))( int );
```

那么 signal 函数的参数和返回值的类型分别是什么? (4 分)

5. 编写一个 C 语言程序, 判断所处的嵌入式处理器是大端机还是小端机。(可以忽略必要的头文件。)(6 分)

二、阅读下面的 C 语言程序并回答问题 (共 16 分)

```
#include <stdio.h>
#include <unistd.h>
int main(void)
{
    int i, k=0
    for(i=0; i<3; i++) {
        pid_t fpid = fork();
        if(fpid == 0)
            printf("son, %d,*\n", ++k);
        else
            printf("fatger, %d,*\n", ++k);
    }
    return 0;
}
```

(1) 假定每次调用 fork 都可以成功生成一个新的进程, 则 test 程序在运行过程中共可产生多少个进程? (包括第一个进程在内。)(4 分)

(2) 假定每次调用 `fork` 都可以成功生成一个新的进程, 则 `test` 程序在运行过程中用户在标准输出设备上共可以看到多少个*号、多少个+号? (4 分)

(3) 在利用 `fork` 创建新进程时, 写时拷贝机制发挥什么作用? (4 分)

(4) 该程序输出的 `k` 的最大值是多少? (4 分)

三、阅读下面的 C 语言程序并回答问题 (共 24 分)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/wait.h>
#include <netinet/in.h>
#define BUFSIZE 1024
FILE *fplog; /* 日志文件指针 */
int start_server();
int process_request(char *rq, int fd);
void sigchld_handler(int sig);
int main()
{
    int sock, fd;
    pid_t pid;
    FILE *fpin;
    char request[BUFSIZE];
    fplog = fopen("logfile", "w+");
    sock = start_server();
    signal(SIGCHLD, sigchld_handler);
    while(1) {
```

```

        fd = accept(sock, NULL, NULL);
        while((pid=fork())!=-1);
        if(pid==0) {
            fpin = fdopen(fd, "r+");
            memset(request,0,sizeof(request));
            fgets(request, BUFSIZE, fpin);
            process_request(request, fd);
            fclose(fpin);
            close(fd);
            exit(0);
        }
        fclose(fpin);
        close(fd);          /* 位置 A */
    }
    close(sock);
    fclose(fplog);
    exit(0);
}

int start_server()
{
    struct sockaddr_in server_sockaddr;
    int sockfd;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server_sockaddr.sin_family = AF_INET;
    server_sockaddr.sin_port = htons(8006);
    server_sockaddr.sin_addr.s_addr = INADDR_ANY;
    bzero(server_sockaddr.sin_zero, 8);
    bind(sockfd, (struct sockaddr *)&server_sockaddr, sizeof(struct sockaddr));
    while(listen(sockfd, 5)!=-1);
    return sockfd;
}

void sigchld_handler(int sig)
{
    while(waitpid(-1,0,WNOHANG)>0);
    return;
}

int process_request(char *rq, int fd)
{
    char cmd[BUFSIZE], arg1[BUFSIZE], arg2[BUFSIZE];
    int a,b;

```

```

FILE *fp;
if(sscanf(rq, "%s %s %s", cmd, arg1, arg2) != 3) {
    fprintf(fplog, "! error: %s", rq); /* 记录日志 */
    fclose(fplog); /* 关闭日志文件 */
    return 0;
}
a = atoi(arg1);
b = atoi(arg2);
if (strcmp(cmd, "A") == 0)
    a = a+b;
else if(strcmp(cmd,"S") == 0)
    a = a-b;
else if(strcmp(cmd,"M") == 0)
    a = a*b;
else if(strcmp(cmd,"D") == 0)
    a = a/b;
else{
    fprintf(fplog, "! error: %s", rq); /* 记录日志 */
    fclose(fplog); /* 关闭日志文件 */
    return 0;
}
fp = fdopen(fd, "w");
fprintf(fp, "%d\r\n", a);
fflush(fp);
fprintf(fplog, "Request: %s", rq); /* 记录日志 */
fprintf(fplog, "Response: %d\r\n", a); /* 记录日志 */
fclose(fp);
fclose(fplog); /* 关闭日志文件 */
return 1;
}

```

(1) 简述基于 TCP 套接字客户端程序的实现流程。(4 分)

(2) 客户端和服务端建立连接后，如果向该服务器发送请求包 “A 16 4\r\n”，则得到的响应数据包是什么？(4 分)

(3) 该服务端程序在哪个端口监听？(2 分)

(4) 解释“僵尸进程”的含义及 sigchld_handler 函数的作用。(4 分)

(5) main 函数位置 A 处的语句“close(fd)”关闭了套接字 fd，为什么不影响子进程和客户端进行通信？(4 分)

(6) 该程序利用文件 logfile 记录运行日志的方法存在什么问题？应如何改正？(3 分)

(7) start_server、process_request、sigchld_handler 三个函数，哪个(或哪些)是在子进程的进程上下文中执行的？(3 分)

四、阅读下面的 C 语言程序并回答问题（共 20 分）

```
#include<linux/init.h>
.....
/*hello driver structure*/
#define HELLO_DEVICE "hello_test"
#define HELLO_NODE "hello"
static dev_t num_dev; /*declare device number variable*/
static struct cdev *cdev_p; /*declare character driver variable*/
static struct class *hello_class; /*declare a class*/
static unsigned char hello_value = 0; /*declare and initialize a variable*/

static int hello_open(struct inode* inode,struct file* filp){.....}
static int hello_release(struct inode* inode,struct file* filp){.....}
static ssize_t hello_read(struct file* filp,char user *buf,size_t count,loff_t* f_pos){.....}
static ssize_t hello_write(struct file* filp,char user *buf,size_t count,loff_t* f_pos){.....}
/*declare file operations*/
static struct file_operations hello_fops={
    .owner      = THIS_MODULE,
    .open       = _____, /* 位置 A */
    .release    = _____, /* 位置 B */
```

```

        .read      = _____,          /* 位置 C */
        .write     = _____,          /* 位置 D */
    };

/*Initialize the LED lights and load the LED device driver*/
static int hello_ctrl_init(void)
{
    int err;
    struct device* temp=NULL;
    /* dynamically delete hello_test device, num_dev is the device id */
    err=alloc_chrdev_region(&num_dev,0,1,HELLO_DEVICE);
    if (err < 0) {
        printk(KERN_ERR "HELLO: unable to get device name %d/n", err); /* 位置 1 */
        return err;
    }
    /*dynamically allocate cdev RAM space*/
    cdev_p = cdev_alloc();
    cdev_p->ops = &hello_fops;
    /*load the device driver*/
    err=cdev_add(cdev_p,num_dev,1);
    if(err){
        printk(KERN_ERR "HELLO: unable to add the device %d/n", err); /* 位置 2 */
        return err;
    }
    /* create hello_test folder at the /sys/class directory*/
    hello_class=class_create(THIS_MODULE,HELLO_DEVICE);
    if(IS_ERR(hello_class))
    {
        err=PTR_ERR(hello_class);
        goto unregister_cdev;
    }
    /* create hello device file based on /sys/class/hello_test and /dev */
    temp=device_create(hello_class, NULL,num_dev, NULL, HELLO_NODE);
    if(IS_ERR(temp))
    {
        err=PTR_ERR(temp);

```



```

        goto unregister_class;
    }
    return 0;
unregister_class:
    class_destroy(hello_class);
unregister_cdev:
    cdev_del(cdev_p);
    return err;
}
/*initialization*/
static int __init hello_init(void)
{
    int ret;
    printk("The driver is insmoded successfully.\n"); /* 位置 3 */
    ret = hello_ctrl_init();
    if(ret)
    {
        printk(KERN_ERR "Apply: Hello_driver_init--Fail !!!\n"); /* 位置 4 */
        return ret;
    }
    return 0;
}
/*exit */
static void __exit hello_exit(void)
{
    printk("The driver is rmmoded successfully.\n"); /* 位置 5 */
    device_destroy(hello_class,HELLO_DEVICE);
    class_destroy(hello_class);
    cdev_del(cdev_p);
    unregister_chrdev_region(num_dev,1);
}
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("First Linux Driver");
module_init(hello_init);
module_exit(hello_exit);

```

(1) 该程序是一个简单的 Linux 驱动程序，简述 Linux 驱动程序和 Linux 应用程序的主要区别有哪些？（4 分）

(2) Linux 驱动程序可以静态编译进入内核，也可以动态加载，请分别写出动态加载、卸载本驱动模块的命令和列出内核已加载模块的命令；如何查看位置 1~5 处的输出信息是否输出？（4 分）

(3) 请写出位置 A、B、C、D 处对应的函数名（4 分）

A

B

C

D

(4) 某同学在实验过程中发现一下现象：

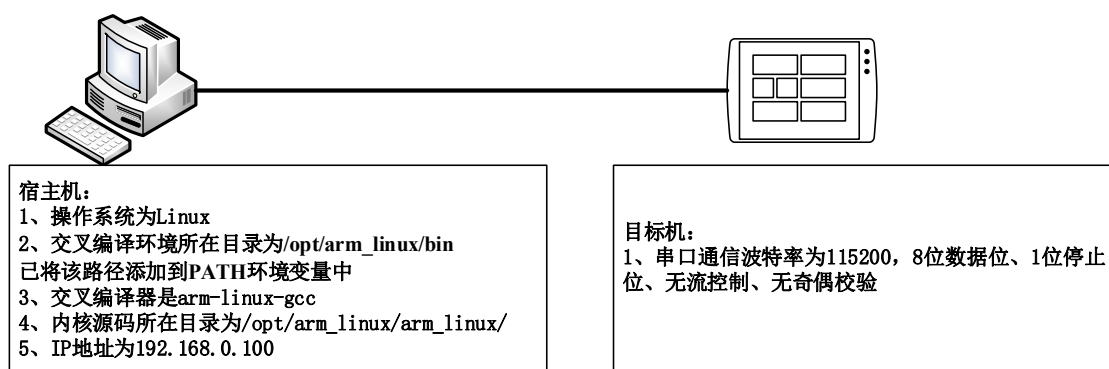
a、目标机开机后，第一次动态加载驱动程序模块正常，也可以通过应用程序对相应的设备进行正常操作；

b、驱动模块卸载操作无任何异常提示；

c、不重启目标机，再次加载驱动模块，系统提示：设备已存在不能不能再创建设备；

d、重新启目标机后 a~c 现象重复。

该同学通过以上现象判断是 `hello_exit(void)` 函数中有错误，且已确认是该函数问题，请修改该函数，以解决上述问题。（8 分）



图一