

# Sistem Programlama Final Ödevi

## “Character Device Uygulaması”

1306160048 Barış YARAR

1306150015 Onur ERMAN

1306151327 Mustafa TAYYAR

Bu projede simülasyon kullanılmıştır. Ubuntu kernel 5.9.2-generic sürümünde denenmiştir.

Arduino üzerindeki testlerimizde arduino uno cihazına veri yazdırdıktan sonra arduinonun kendini yeniden başlatması gibi birtakım sorunlar yaşayıp projenin simülasyon üzerinde gösterilmesi uygun görülmüştür.

## Kodlar

```
Activities Text Editor Oca 23 13:04 mychardev.c ~/driverlast
Open
1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include <linux/cdev.h>
4 #include <linux/device.h>
5 #include <linux/kernel.h>
6 #include <linux/uaccess.h>
7 #include <linux/fs.h>
8
9 //fonksiyon prototipleri
10 static int mychardev_open(struct inode *inode, struct file *file);
11 static int mychardev_release(struct inode *inode, struct file *file);
12 static long mychardev_ioctl(struct file *file, unsigned int cmd, unsigned long arg);
13 static ssize_t mychardev_read(struct file *file, char __user *buf, size_t count, loff_t *offset);
14 static ssize_t mychardev_write(struct file *file, const char __user *buf, size_t count, loff_t *offset);
15
16 //dosya işlemleri
17 static const struct file_operations mychardev_fops = {
18     .owner      = THIS_MODULE,
19     .open       = mychardev_open,
20     .release    = mychardev_release,
21     .unlocked_ioctl = mychardev_ioctl,
22     .read       = mychardev_read,
23     .write      = mychardev_write
24 };
25
26 //simülasyon cihazı için data tutan struct yapısı
27 struct mychar_device_data {
28     struct cdev cdev;
29 };
30
31 //Major tutan global değişken
32 static int dev_major = 0;
33 //sysfs class struct yapısı
34 static struct class *mychardev_class = NULL;
35 //simülasyon cihazı için dataları tutan dizi
36 static struct mychar_device_data mychardev_data[0];
37
38 //simülasyon cihazına yetki veren fonksiyon: 0666 -> rw-rw-rw-
39 static int mychardev_uevent(struct device *dev, struct kobj_uevent_env *env)
40 {
41     add_uevent_var(env, "DEVMODE=%#o", 0666);
42     return 0;
43 }
44
```

```
45 static int __init mychardev_init(void)
46 {
47     //simülasyon için mychardev0 adında sanal bir device oluşturuldu.
48     int err, i;
49     dev_t dev;
50     //simülasyon cihazı için alan ayrıldı
51     err = alloc_chrdev_region(&dev, 0, 0, "mychardev");
52     //major değeri atandı
53     dev_major = MAJOR(dev);
54     //sysfs class'ı oluşturuldu
55     mychardev_class = class_create(THIS_MODULE, "mychardev");
56     //okuma yazma yetkileri verildi
57     mychardev_class->dev_uevent = mychardev_uevent;
58
59     //yeni cihaz init edildi
60     cdev_init(&mychardev_data[0].cdev, &mychardev_fops);
61     mychardev_data[0].cdev.owner = THIS_MODULE;
62
63     //minor number'ı 0 olan cihaz sisteme eklendi
64     cdev_add(&mychardev_data[0].cdev, MKDEV(dev_major, 0), 1);
65     //cihaz düğümü oluşturuldu -> /dev/mychardev-0
66     device_create(mychardev_class, NULL, MKDEV(dev_major, 0), NULL, "mychardev-0");
67
68     return 0;
69 }
70
71 //simülasyonda oluşturulan cihaz, işlem bittikten sonra kaldırılmalıdır.
72 static void __exit mychardev_exit(void)
73 {
74     device_destroy(mychardev_class, MKDEV(dev_major, 0));
75     class_unregister(mychardev_class);
76     class_destroy(mychardev_class);
77     unregister_chrdev_region(MKDEV(dev_major, 0), MINORMASK);
78 }
79 // I/O işlemleri
80 static int mychardev_open(struct inode *inode, struct file *file)
81 {
82     printk("MYCHARDEV: Device open\n");
83     return 0;
84 }
85
86 static int mychardev_release(struct inode *inode, struct file *file)
87 {
88     printk("MYCHARDEV: Device close\n");
89     return 0;
90 }
91
92 static long mychardev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
93 {
94     printk("MYCHARDEV: Device ioctl\n");
95     return 0;
96 }
```

```
96 }
97 // copy_to_user kullanmak, buffer pointerları ile memory kopyalamaktan daha güvenli
98 static ssize_t mychardev_read(struct file *file, char __user *buf, size_t count, loff_t *offset)
99 {
100     uint8_t *data = "Hello from the kernel world!\n";
101     size_t datalen = strlen(data);
102
103     printk("Reading device: %d\n", MINOR(file->f_path.dentry->d_inode->i_rdev));
104
105     // kaç byte okunacağını kontrol etmek daha güvenli. Fazladan byte'lar okunursa kernel stack'ten veri okuyup sistemi tehlikeye atar.
106     if (count > datalen) {
107         count = datalen;
108     }
109     // copy_to_user okunamayan byte sayısını döndürür, 0 döndürmesi başarılı bir şekilde okunduğunu gösterir.
110     // if(0) -> false
111     if (copy_to_user(buf, data, count)) {
112         return -EFAULT;
113     }
114
115     return count;
116 }
117 // copy_from_user, buffer pointerları kullanmaktan daha güvenli
118 static ssize_t mychardev_write(struct file *file, const char __user *buf, size_t count, loff_t *offset)
119 {
120     size_t maxdatalen = 30, ncopied;
121     uint8_t databuf[maxdatalen];
122
123     printk("Writing device: %d\n", MINOR(file->f_path.dentry->d_inode->i_rdev));
124     //user space'ten kaç byte kopyalanacağını kontrol etmek gerekir.
125     if (count < maxdatalen) {
126         maxdatalen = count;
127     }
128
129     // copy_from_user kopyalanamayan byte sayısını döndürür. 0 döndürmesi başarılı bir şekilde kopyalandığını gösterir.
130     ncopied = copy_from_user(databuf, buf, maxdatalen);
131
132     if (ncopied == 0) {
133         printk("Copied %zd bytes from the user\n", maxdatalen);
134     } else {
135         printk("Could't copy %zd bytes from the user\n", ncopied);
136     }
137
138     databuf[maxdatalen] = 0;
139
140     printk("Data from the user: %s\n", databuf);
141
142     return count;
143 }
144 }
145
146 MODULE_LICENSE("GPL");
147 MODULE_AUTHOR("bydev");
148
149 module_init(mychardev_init);
150 module_exit(mychardev_exit);
```

## Makefile



```
Activities Text Editor ▾  
Open ▾ [icon] Makefile ~/driverlast Save [icon] [icon]  
1 BINARY := mychardev  
2 KERNEL := /lib/modules/$(shell uname -r)/build  
3 ARCH := x86  
4 C_FLAGS := -Wall  
5 KMOD_DIR := $(shell pwd)  
6 TARGET_PATH := /lib/modules/$(shell uname -r)/kernel/drivers/char  
7  
8 ccflags-y += $(C_FLAGS)  
9  
10 obj-m += mychardev.o  
11  
12 $(BINARY)-y := $(OBJECTS)  
13  
14 $(BINARY).ko:  
15     make -C $(KERNEL) M=$(KMOD_DIR) modules  
16  
17 install:  
18     cp $(BINARY).ko $(TARGET_PATH)  
19     depmod -a  
20  
21 uninstall:  
22     rm $(TARGET_PATH)/$(BINARY).ko  
23     depmod -a  
24  
25 clean:  
26     make -C $(KERNEL) M=$(KMOD_DIR) clean
```

Test Aşaması

echo "test test test" -> /dev/mychardev-0

echo "-----Selamlar-----" -> /dev/mychardev-0

Komutlarıyla mychardev-0 adlı cihaza "test test test" ve "-----Selamlar-----" yazdırılır.

```
jupic@ABRA: ~/driverlast  
jupic@ABRA:~/driverlast$ echo "test test test" > /dev/mychardev-0  
jupic@ABRA:~/driverlast$ echo "-----Selamlar-----" > /dev/mychardev-0  
jupic@ABRA:~/driverlast$  
  
jupic@ABRA: ~/driverlast  
Jan 19 12:30:18 ABRA kernel: [ 2658.776906] pcieport 0000:00:1c.4: device [8086:a114] error s  
Jan 19 12:30:18 ABRA kernel: [ 2658.776910] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:30:18 ABRA kernel: [ 2658.981760] pcieport 0000:00:1c.4: AER: Corrected error receive  
Jan 19 12:30:18 ABRA kernel: [ 2658.981769] pcieport 0000:00:1c.4: PCIe Bus Error: severity=Cor  
(Receiver ID)  
Jan 19 12:30:18 ABRA kernel: [ 2658.981772] pcieport 0000:00:1c.4: device [8086:a114] error s  
Jan 19 12:30:18 ABRA kernel: [ 2658.981776] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:30:18 ABRA kernel: [ 2659.084056] pcieport 0000:00:1c.4: AER: Corrected error receive  
Jan 19 12:30:18 ABRA kernel: [ 2659.084065] pcieport 0000:00:1c.4: PCIe Bus Error: severity=Cor  
(Receiver ID)  
Jan 19 12:30:18 ABRA kernel: [ 2659.084068] pcieport 0000:00:1c.4: device [8086:a114] error s  
Jan 19 12:30:18 ABRA kernel: [ 2659.084072] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:30:18 ABRA kernel: [ 2659.181998] MYCHARDEV: Device open  
Jan 19 12:30:18 ABRA kernel: [ 2659.182027] Writing device: 0  
Jan 19 12:30:18 ABRA kernel: [ 2659.182029] Copied 22 bytes from the user  
Jan 19 12:30:18 ABRA kernel: [ 2659.182031] Data from the user: -----Selamlar-----  
Jan 19 12:30:18 ABRA kernel: [ 2659.182031]  
Jan 19 12:30:18 ABRA kernel: [ 2659.182037] MYCHARDEV: Device close  
Jan 19 12:30:18 ABRA kernel: [ 2659.186441] pcieport 0000:00:1c.4: AER: Corrected error receive  
Jan 19 12:30:18 ABRA kernel: [ 2659.186450] pcieport 0000:00:1c.4: PCIe Bus Error: severity=Cor  
  
jupic@ABRA: ~/driverlast  
Jan 19 12:29:27 ABRA kernel: [ 2607.987240] pcieport 0000:00:1c.4: AER: Corrected error r  
Jan 19 12:29:27 ABRA kernel: [ 2607.987249] pcieport 0000:00:1c.4: PCIe Bus Error: severi  
(Receiver ID)  
Jan 19 12:29:27 ABRA kernel: [ 2607.987253] pcieport 0000:00:1c.4: device [8086:a114] e  
Jan 19 12:29:27 ABRA kernel: [ 2607.987257] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:29:27 ABRA kernel: [ 2608.089765] pcieport 0000:00:1c.4: AER: Corrected error r  
Jan 19 12:29:27 ABRA kernel: [ 2608.089775] pcieport 0000:00:1c.4: PCIe Bus Error: severi  
(Receiver ID)  
Jan 19 12:29:27 ABRA kernel: [ 2608.089779] pcieport 0000:00:1c.4: device [8086:a114] e  
Jan 19 12:29:27 ABRA kernel: [ 2608.089783] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:29:27 ABRA kernel: [ 2608.191995] pcieport 0000:00:1c.4: AER: Corrected error r  
Jan 19 12:29:27 ABRA kernel: [ 2608.192004] pcieport 0000:00:1c.4: PCIe Bus Error: severi  
(Receiver ID)  
Jan 19 12:29:27 ABRA kernel: [ 2608.192008] pcieport 0000:00:1c.4: device [8086:a114] e  
Jan 19 12:29:27 ABRA kernel: [ 2608.192013] pcieport 0000:00:1c.4: [ 0] RxErr  
Jan 19 12:29:27 ABRA kernel: [ 2608.470241] MYCHARDEV: Device open  
Jan 19 12:29:27 ABRA kernel: [ 2608.470272] Writing device: 0  
Jan 19 12:29:27 ABRA kernel: [ 2608.470274] Copied 15 bytes from the user  
Jan 19 12:29:27 ABRA kernel: [ 2608.470276] Data from the user: test test test  
Jan 19 12:29:27 ABRA kernel: [ 2608.470276]  
Jan 19 12:29:27 ABRA kernel: [ 2608.470283] MYCHARDEV: Device close  
Jan 19 12:29:28 ABRA kernel: [ 2608.539169] pcieport 0000:00:1c.4: AER: Corrected error r
```

