

```

1  /*
2  * test.c
3  *
4  * This file provides a set of user space code to exorcise the
5  * sample driver.
6  */
7
8  #include <stdio.h>
9  #include <sys/poll.h>
10 #include <sys/types.h>
11 #include <sys/stat.h>
12 #include <unistd.h>
13 #include <fcntl.h>
14 #include <errno.h>
15 #include <signal.h>
16
17 #define IOMYDEV_ALLOC_MEM      0x6b01
18 #define IOMYDEV_MEM_INFO      0x6b02
19
20 static int write_command(int);
21 static int read_command(int);
22 static int rewind_command(int);
23 static int async_toggle_command(int);
24 static int mem_set_command(int);
25 static int mem_get_command(int);
26 static int quit(int);
27
28 typedef int(*command_ptr)(int);
29
30 static command_ptr command_handlers[] = {
31     write_command,
32     read_command,
33     rewind_command,
34     async_toggle_command,
35     mem_set_command,
36     mem_get_command,
37     quit,
38     };
39
40 #define MAX_COMMAND (sizeof(command_handlers) / sizeof(command_ptr))
41
42 int
43 main(int argc, char *argv[])
44 {
45     int fd = open("/dev/mynod0", O_RDWR);
46     struct pollfd poll_data;
47
48     if(fd < 0) {
49         perror("Could not open file: ");
50         return errno;
51     }
52
53     do {
54         int command;
55
56         printf("Driver Testing\n\n");
57

```

```

58         printf("1. Write data\n");
59         printf("2. Read data\n");
60         printf("3. Rewind file\n");
61         printf("4. Toggle async notification\n");
62         printf("5. Set memory size\n");
63         printf("6. Get memory size\n");
64         printf("7. Quit\n");
65         printf("\n\n");
66
67         scanf("%d", &command);
68
69         if(command <= MAX_COMMAND && command > 0);
70             command_handlers[command-1](fd);
71     } while(1);
72
73     close(fd);
74     return 0;
75 }
76
77 static int
78 write_command(int fd)
79 {
80     static char buffer[80];
81
82     printf("Enter input below, EOF when done\n");
83
84     while(scanf("%s", buffer) != EOF) {
85         buffer[79] = '\0';
86         write(fd, buffer, strlen(buffer));
87         write(fd, " ", strlen(" "));
88     }
89
90     printf("\n\n");
91
92     return 0;
93 }
94
95 static int
96 read_command(int fd)
97 {
98     int len;
99     static char buffer[4096];
100
101     printf("Number of bytes to read: ");
102     scanf("%d", &len);
103     if(len > 4096)
104         return 0;
105
106     len = read(fd, buffer, len);
107
108     if(len >= 0) {
109         printf("Read returned %d bytes:\n", len);
110         printf("%s", buffer);
111         printf("\n\n");
112     } else
113         perror("Read error: ");
114

```

```

115     return 0;
116 }
117
118 static int
119 rewind_command(int fd)
120 {
121     lseek(fd, 0, SEEK_SET);
122 }
123
124 static void
125 async_handler(int signal)
126 {
127     if(signal == SIGIO) {
128         printf("\n\n*** ASYNC DATA NOTIFICATION ***\n\n");
129     }
130     else
131         printf("\n\n*** Unknown signal ***\n\n");
132 }
133
134
135 static int
136 async_toggle_command(int fd)
137 {
138     static int async = 0;
139
140     static struct sigaction action;
141     static struct sigaction old_action;
142
143     action.sa_handler = async_handler;
144     action.sa_flags = SA_ONESHOT;
145
146     unsigned long oflags;
147
148     async = !async;
149
150     if(async) {
151         fcntl(fd, F_SETOWN, getpid());
152         oflags = fcntl(STDIN_FILENO, F_GETFL);
153         fcntl(fd, F_SETFL, oflags | FASYNC);
154         if(sigaction(SIGIO, &action, &old_action))
155             perror("Could not set signal handler: ");
156         else
157             printf("\nAsync notification now ON\n\n");
158     } else {
159         oflags = fcntl(STDIN_FILENO, F_GETFL);
160         fcntl(fd, F_SETFL, oflags & ~FASYNC);
161         if(sigaction(SIGIO, &old_action, &action))
162             perror("Could not set signal handler: ");
163         else
164             printf("\nAsync notification now OFF\n\n");
165     }
166
167     return 0;
168 }
169
170 static int
171 mem_set_command(int fd)

```

```
172 {
173     unsigned long size;
174
175     printf("\n\nNew memory size(hex): ");
176     scanf("%x", &size);
177
178     ioctl(fd, IOMYDEV_ALLOC_MEM, size);
179
180     printf("\n");
181
182     return 0;
183 }
184
185 static int mem_get_command(int fd)
186 {
187     unsigned long size;
188
189     ioctl(fd, IOMYDEV_MEM_INFO, &size);
190
191     printf("\n\nDevice memory size = %#010x\n\n", size);
192
193     return 0;
194 }
195
196
197 static int
198 quit(int fd)
199 {
200     close(fd);
201
202     exit(0);
203 }
204
205
206
```