TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

Instructor – Innova Lee(Sanghoon Lee)
gcccompil3r@gmail.com
Student – Hyungju Kim
mihaelkel@naver.com

File transportation

We can trasform some files, using C/S architecture. It's simple. We've already leanned how to make a file, and send a message. Only You have to do is send a message after reading a file.

file_server.c

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE      32

void err_handler(char* msg){
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc,char** argv){
    int serv_sock, clnt_sock, fd;
    char buf[BUF_SIZE] = {0};
    int read_cnt;

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;

    if(argc != 2){
        printf("use : %s <port>\n",argv[0]);
        exit(1);
    }

    fd = open("file_server.c",O_RDONLY);
    serv_sock = socket(PF_INET, SOCK_STREAM, 0);

    if(serv_sock == -1)
        err_handler("socket() error");

    memset(&serv_addr, 0 , sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("bind() error");
    if(listen(serv_sock, 5)== -1)
        err_handler("listen() error");

    clnt_addr_size = sizeof(clnt_addr);

    clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);

    for(;;){
        read_cnt = read(fd, buf, BUF_SIZE);
        if(read_cnt < BUF_SIZE){
```

```
56                  write(clnt_sock, buf, read_cnt);
57                  break;
58              }
59              write(clnt_sock, buf, BUF_SIZE);
60          }
61
62          shutdown(clnt_sock, SHUT_WR);
63          read(clnt_sock, buf, BUF_SIZE);
64          printf("msg from client : %s\n",buf);
65
66          close(fd);
67          close(clnt_sock);
68          close(serv_sock);
69
70          return 0;
71      }
72
```

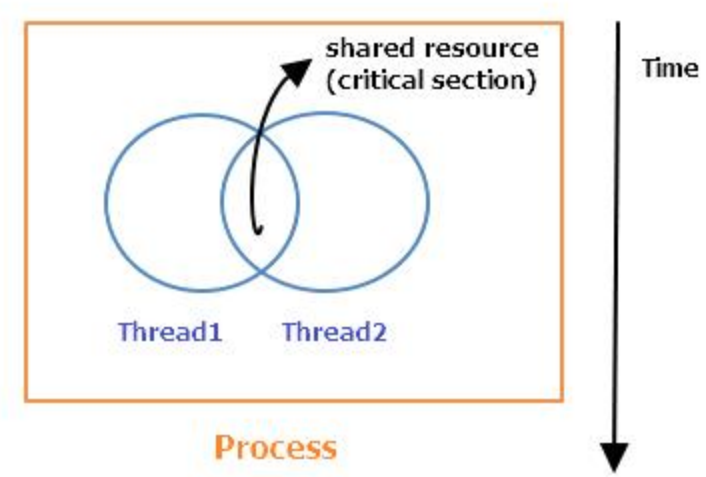

file_client.c

```
1   #include <fcntl.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <string.h>
5   #include <unistd.h>
6   #include <arpa/inet.h>
7   #include <sys/socket.h>
8
9   typedef struct sockaddr_in si;
10  typedef struct sockaddr * sap;
11
12  #define BUF_SIZE     32
13
14  void err_handler(char* msg){
15      fputs(msg, stderr);
16      fputc('\n', stderr);
17      exit(1);
18  }
19
20  int main(int argc,char** argv){
21      char buf[BUF_SIZE] = {0,};
22      int fd, sock, read_cnt;
23      si serv_addr;
24
25      if(argc != 3){
26          printf("use: %s <IP> <PORT>\n",argv[0]);
27          exit(1);
28      }
29
30      fd = open("receive.txt", O_CREAT | O_WRONLY);
31      sock = socket(PF_INET, SOCK_STREAM, 0);
32
33      if(sock == -1)
34          err_handler("socket() error");
35
36      memset(&serv_addr, 0, sizeof(serv_addr));
37      serv_addr.sin_family = AF_INET;
38      serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
39      serv_addr.sin_port = htons(atoi(argv[2]));
40
```

```
41        if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
42            err_handler("connect() error");
43        else
44            puts("Connected .....");
45
46        while((read_cnt = read(sock, buf, BUF_SIZE)) != 0)
47            write(fd, buf, read_cnt);
48        puts("Received File Data");
49        write(sock, "Thank you", 10);
50        close(fd);
51        close(sock);
52
53    }
54
```


Multi-Thread

Process and Thread both can run in parallel by context switching. Using Process is hard to share their memory with other process. otherwise, Thread is easier because thread are in the same process. that is, We don't have to use IPC.



Implement a Thread

```
int pthread_create(pthread_t* th_id, const pthread_attr_t* attr, void (*func)(void*), void* arg);
```

return value : 0 (success)

```
int pthread_join(pthread_t th_id, void** thread_return);
```

wait until a thread, which th_id is pointing, be exited. It acts like "wait(&status)" function.