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

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

sigaction

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
struct sigaction act_new;
struct sigaction act_old;

void sigint_handler(int signo){
    printf("Ctrl + C\n");
    printf("If you push it one more time than exit\n");
    sigaction(SIGINT, &act_old, NULL);
}

int main(void){
    //sa : signal action
    //signal handller set
    act_new.sa_handler = sigint_handler;
    //empty the mask
    sigemptyset(&act_new.sa_mask);

    //act_old : save previous handler.
    //& 들어간 애들은, 값이 변경될수도 있음.(다른사람프로그램 보는법)
    //output이 여러개 가능.
    sigaction(SIGINT, &act_new, &act_old);
    while(1){
        printf("sigaction test\n");
        sleep(1);
    }

    return 0;
}

```


int sigaction(int signum, const struct sigaction *act, struct sigaction* oldact);

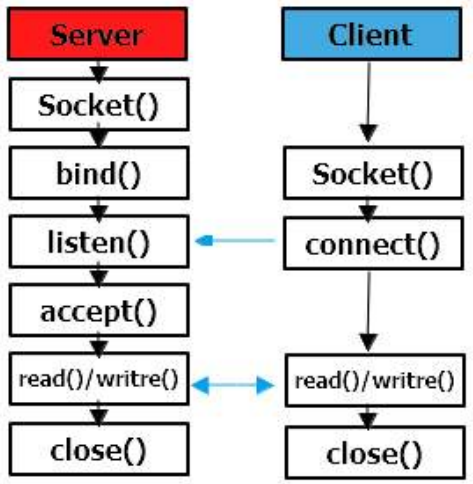
The sigaction() system call is used to change the action taken by a process on receipt of a specific signal, but SIGKILL and SIGSTOP.

Implement kill

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
int main(int argc,char* argv[]){
    if(argc < 2)
        printf("Usage : ./exe pid\n");
    else
        kill(atoi(argv[1]),SIGINT);

    return 0;

}

```

Socket network

client - server architecture




```c
#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;

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

int main(int argc,char* argv[]){

	int sock;
	int str_len;
	char msg[32];
	si serv_addr;
	if(argc != 3){
		printf("err\n");
		exit(1);
	}

	sock = socket(PF_INET, SOCK_STREAM, 0);

	if(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 = inet_addr(argv[1]);
```

```
        serv_addr.sin_port = htons(atoi(argv[2]));

        if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
            err_handler("connect() error");

        str_len = read(sock, msg, sizeof(msg) -1 );
        if(str_len == -1)
            err_handler("read() error");
        printf("msg from serv : %s\n",msg);

        close(sock);
        return 0;

    }
```



```
    #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;

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

    }

    int main(int argc,char** argv){
        int serv_sock;
        int clnt_sock;

        si serv_addr;
        si clnt_addr;
        socklen_t clnt_addr_size;
        char msg[] = "Hello Network Programming";

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

        //bind : server ip address setting
        if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
            err_handler("bind() error");
```

```
44          //listen : max people , wait client.
45          if(listen(serv_sock, 5) == -1)
46                err_handler("listen() error");
47
48          clnt_addr_size = sizeof(clnt_addr);
49          //client permit.
50          clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
51          if(clnt_sock == -1)
52                err_handler("accept() error");
53
54          write(clnt_sock, msg, sizeof(msg));
55
56          close(clnt_sock);
57          close(serv_sock);
58
59    }
60
```