

System Programming

(Assignment3-3)

과 목	시스템프로그래밍실습
담당교수	이기훈 교수님
학 과	컴퓨터공학과
학 번	2010720149
성 명	이동현
날 짜	2016. 05. 20 (금)

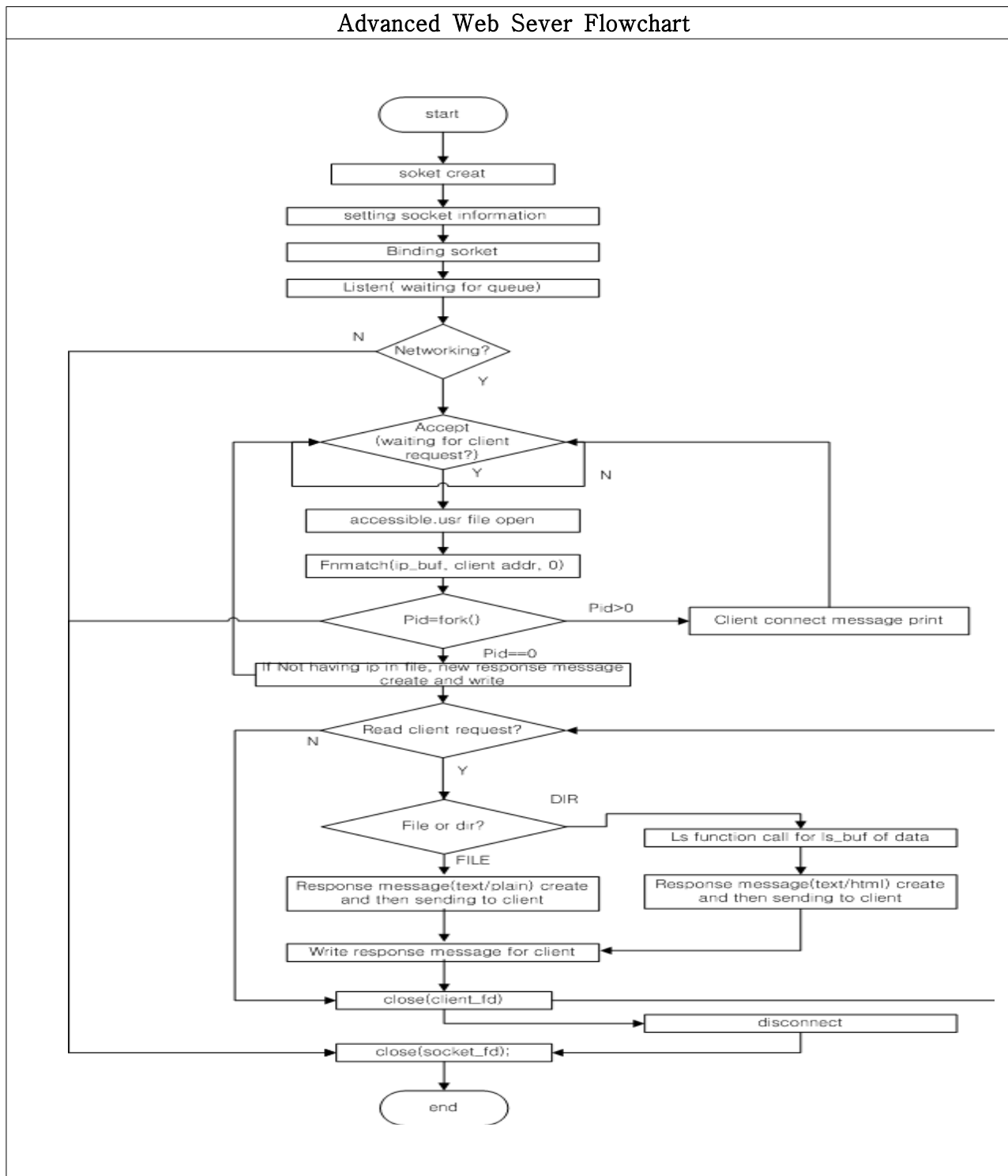


A. Introduction

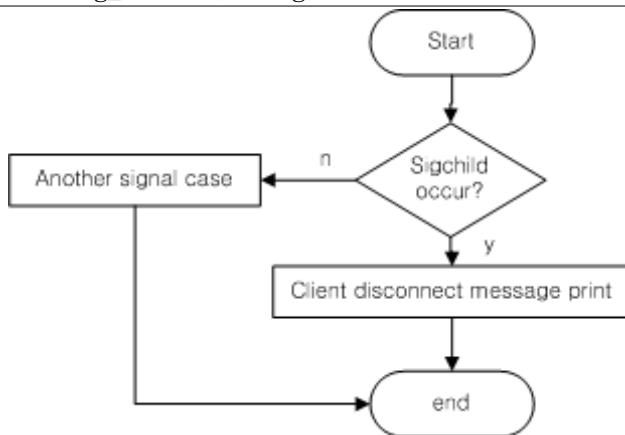
♣ Advanced Web Sever ♣

이번 과제는 기존의 소켓프로그래밍이 가능한 html_ls의 ls 명령어 구현 프로그램에서 추가적으로 다중 접속과 접근 제어를 지원하는 웹 서버 프로그램을 구현하는 것이다. 다중 접속 지원은 동시에 여러 클라이언트가 같은 서버에 접속이 가능하고, 접근 제어 지원은 접근 가능한 ip를 제어한다.

B. Flowchart



```
void sig_handler(int sig)
```



C. Pseudo code

Basic Web Sever Pseudo code

```

int main(){
    initializing buf, arv
    socket_fd <- socket(PF_INET, SOCK_STREAM, 0)
    opt<-1;
    bzero((char*)&server_addr, sizeof(server_addr));
    server_addr.sin_family <- AF_INET;
    server_addr.sin_addr.s_addr <- htonl(INADDR_ANY);
    server_addr.sin_port <- htons(PORTNO);
    setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
    bind(socket_fd, (struct sockaddr*)&server_addr, sizeof(server_addr));
    listen(socket_fd, 10);
    while(1){
        initializing buf, response, argv
        len <- sizeof(client_addr)
        client_fd <- accept(socket_fd, (struct sockaddr*)&client_addr, &len);
        cur_time_time[idx]<-current time set
        client_num++
        fp<- fopen("accessible usr", "r")
        while(!feof(fp))
            fscanf(fp, "%s", ip_buf)
            if(fnmatch(ip_buf, inet_ntoa(client_addr.sin_addr), 0)==0)
                flag<-1;
        pid<-fork();
        if pid==0
            if flag==0
                No access Response Message create and write to client
                continue;
        while(len_out <- read(client_fd, buf, BUFSIZE)>0){
            argc <-2;
            Dflag, Fflag <-0;
  
```

```

        if(Dflag==1)
            ls(argc, argv, ls_buf);
            strcpy(response, "HTTP/1.1 200 OK\r\nAccept-Ranges: bytes\r\nConnection:
close\r\n");

            strcat(response, "Content-Length: 100000\r\n");
            strcat(response, "Content-Type: text/html\r\n");
            strcat(response, "\r\n");
            strcat(response, ls_buf);
        else if(Fflag==1)
            strcpy(response, "HTTP/1.1 200 OK\r\nAccept-Ranges: bytes\r\nConnection:
close\r\n");

            strcat(response, "Content-Length: 100000\r\n");
            strcat(response, "Content-Type: text/plain\r\n");
            strcat(response, "\r\n");
            strcat(response, ls_buf);
            write(client_fd, response, 100000);
            initializing buf, response
    }
    close(client_fd);
    exit(0);
}
else
    connect message print

allocation delete for argv
close(socket_fd);
}

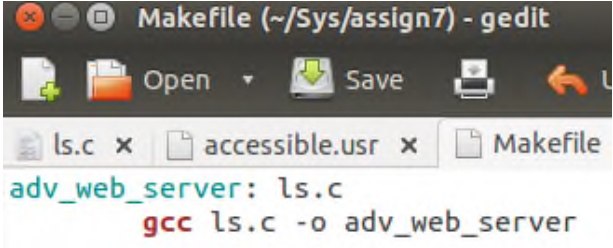
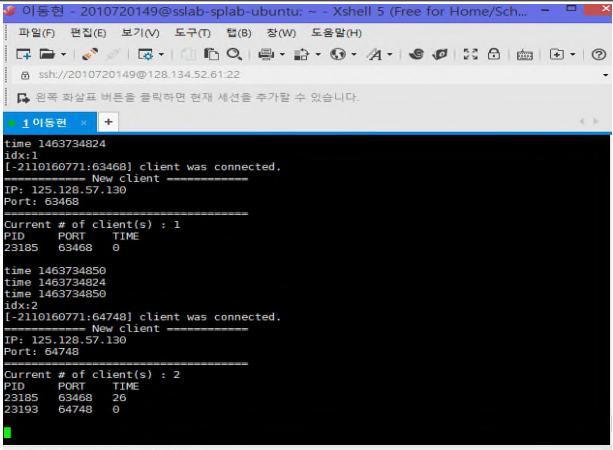
void sig_handler(int sig)
{
    if sig == SIGCHLD
        pid <- wait(NULL)
        disconnect message print
        return ;
    else
        another signal exe
        return;
}

```

D. Reference

- 강의자료 ' 2016-1_SPLab_11_Basic+Server_v2

E. Conclusion

♣ 조건 ♣	Makefile
<p>다중 접속 지원</p> <ul style="list-style-type: none"> - 클라이언트와 소켓 연결된 이후의 작업을 새로운 프로세스에서 수행하여 동시에 다수의 클라이언트가 접속하도록 지원. - 연결 및 해제 시 주어진 정보대로 출력. <p>접근 제어 지원</p> <ul style="list-style-type: none"> - 해당하는 문자열을 전부 출력해야 함. 	 <pre>adv_web_server: ls.c gcc ls.c -o adv_web_server</pre>
My port number	My shell
40051	 <pre>time 1463734824 idx:1 [2110160771:63468] client was connected. New client IP: 125.128.57.130 Port: 63468 Current # of client(s) : 1 PID PORT TIME 23185 63468 0 time 1463734850 time 1463734824 time 1463734850 idx:2 [2110160771:64748] client was connected. New client IP: 125.128.57.130 Port: 64748 Current # of client(s) : 2 PID PORT TIME 23185 63468 25 23193 64748 0</pre>
my ifconfig ip and port	
http://192.168.111.131:40051/	

♣ 단일 접속 결과 화면 1 ♣

♣ 단일 접속 결과 화면 1 ♣

```
ls -al /home/dong/Sys/assign7
Directory path: /home/dong/Sys/assign7
total: 184
```

Name	Permission	Link	Owner	Group	Size	Last M
.	-rwxrwxr-x	2	dong	dong	4096	5.20
..	-rwxrwxr-x	6	dong	dong	4096	5.18
accessible_usr	-rw-rw-r--	1	dong	dong	32	5.20
accessible_usr_	-rw-rw-r--	1	dong	dong	0	5.20
adv_web_server	-rwxrwxr-x	1	dong	dong	44992	5.20
clientc	-rw-rw-r--	1	dong	dong	1152	5.12
clientc_	-rw-rw-r--	1	dong	dong	392	5.8.4
html_is.html	-rw-rw-r--	1	dong	dong	0	5.20
ls.c	-rw-rw-r--	1	dong	dong	50832	5.20
ls.c_	-rw-rw-r--	1	dong	dong	50832	5.20
Makefile	-rw-rw-r--	1	dong	dong	50	5.20
Makefile_	-rw-rw-r--	1	dong	dong	34	5.17
server.c	-rw-rw-r--	1	dong	dong	2268	5.16
server.c_	-rw-rw-r--	1	dong	dong	1852	5.16

```
[24094912:31214] client was connected.
```

```
===== New client =====
```

```
IP: 192.168.111.1
```

```
Port: 31214
```

```
=====
```

```
Current # of client(s) : 1
```

```
PID    PORT    TIME
```

```
10234  31214  0
```

```
10234 child process is terminated!
```

```
[24094912:31214] client was disconnected.
```

```
===== Disconnected client =====
```

```
IP: 192.168.111.1
```

```
Port: 29596
```

```
=====
```

```
Current # of client(s) : 0
```

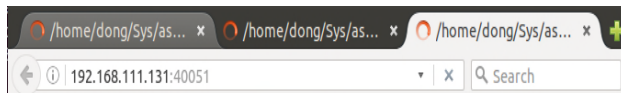
```
PID    PORT    TIME
```

하나의 클라이언트가 서버에게 요청을 보낼 경우 다음 과 같이 하나의 클라이언트가 연결 되었음을 std 터미널 창을 확인 할 수 있다. client의 ip와 port를 출력해주며 연결 중인 client의 pid와 port와 time 그리고 개수를 확인 할 수 있다

다음은 연결 된 클라이언트의 창을 끝 경우, 연결이 종료된 정보를 출력해주며 현재 클라이언트의 정보엔 아무런 클라이언트가 없음을 확인 할 수 있다

단일 접속 상태 확인!!!

♣ 다중 접속 결과 화면 2 ♣



```
ls -al /home/dong/Sys/assign7
```

ls -al /home/dong/Sys/assign7

Directory path: /home/dong/Sys/assign7

total: 184

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	dong	dong	4096	5 20 03:08
..	drwxrwxr-x	6	dong	dong	4096	5 18 20:36
accessible_usr	-rw-rw-r--	1	dong	dong	37	5 20 01:58
accessible_usr_	-rw-rw-r--	1	dong	dong	0	5 20 01:57
adv_web_server	-rwxrwxr-x	1	dong	dong	44995	5 20 03:08
client.c	-rw-rw-r--	1	dong	dong	1157	5 12 05:28
client.c	-rw-rw-r--	1	dong	dong	1157	5 12 05:28

```
dong@ubuntu:~/Sys/assign7$ ./adv_web_server
[-2089834304:35761] client was connected.
===== New client =====
IP: 192.168.111.131
Port: 35761
=====
Current # of client(s) : 1
PID      PORT      TIME
10272    35761     0

[-2089834304:36273] client was connected.
===== New client =====
IP: 192.168.111.131
Port: 36273
=====
Current # of client(s) : 2
PID      PORT      TIME
10272    35761     6
10277    36273     0

[-2089834304:36529] client was connected.
===== New client =====
IP: 192.168.111.131
Port: 36529
=====
Current # of client(s) : 3
PID      PORT      TIME
10272    35761     10
10277    36273     4
10278    36529     0

10277 child process is terminated!
[-2089834304:36529] client was disconnected.
===== Disconnected client =====
IP: 192.168.111.131
Port: 29596
=====
Current # of client(s) : 2
PID      PORT      TIME
10272    35761     18
10278    36529     8

10278 child process is terminated!
[-2089834304:36529] client was disconnected.
===== Disconnected client =====
IP: 192.168.111.131
Port: 29596
=====
Current # of client(s) : 1
PID      PORT      TIME
10272    35761     22

10272 child process is terminated!
[-2089834304:36529] client was disconnected.
===== Disconnected client =====
IP: 192.168.111.131
Port: 29596
=====
Current # of client(s) : 0
PID      PORT      TIME
```

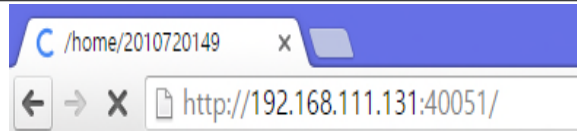
이번은 다중 접속 클라이언트를 확인 할 수 있다

우선 서버 접속을 위한 세 개의 클라이언트가 접속 중임을 확인 할 수 있다 그리고 중간 클라이언트를 종료시키고 마지막 클라이언트 그리고 첫 번째 클라이언트를 종료시키도록 하였다 std 타이틀을 확인해보자

우선 첫 클라이언트가 접속이 되었는데 클라이언트 수는 1이며 0초가 걸린 클라이언트를 확인 할 수 있다 그리고 이어서 두 번째 클라이언트를 추가시키면 서로 다른 pid를 가진 프로세스가 클라이언트 접속을 할당을 하을 하는데 그전에 접속한 첫 번째 클라이언트의 연결 지속시간이 6초가 흐름을 알 수 있다 이어서 3번째 클라이언트 접속 시 마찬가지로 각 pid와 port time값을 통해 서로 독립적인 프로세스들이 수행이 된다 그리고 종료는 두 번째 세 번째 첫 번째 순으로 종료를 시켰으며 각 클라이언트를 종료시 할당된 프로세스들이 종료되었음을 확인 할 수 있다

다중 클라이언트 접속 확인!!!!!!

♣ 접근 제어 결과 화면 3♣



< 접속할 IP >

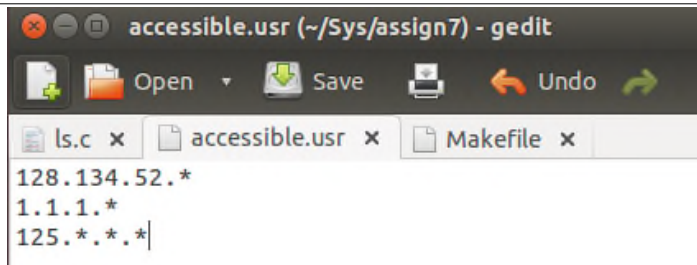
1) 접근제어 허용 IP

```
ls.c x accessible.usr x Makefile x
128.134.52.*
192.168.111.*
125.*.*.*|
```

접근을 허용하는 ip를 accessible.usr에 작성하여 접근을 제어하도록 한다
내가 접속할 ip가 accessible.usr 파일에 존재하기 때문에 다음 과 같이 클라이언트가 서버와 접속이 허용이 되어 결과창이 제대로 나옴을 확인 할 수 있다



2) 접근제어 제한 IP



내가 접근할 ip주소가 허용되었는지 않기 때문에 해당 ip로 클라이언트가 서버에 접속 시 다음과 같은 접근 제한 에러 메시지를 확인을 할 수 있다



♣ 고찰 ♣

이번 과제는 advanced 한 web_server를 구현하는데 목적이 있다. 저번 과제에서 클라이언트가 다중 접속이 가능하고 접근 제어가 가능하도록 하는 것이다. fork를 이용하여 클라이언트의 접속을 수행하는 프로세스를 생성하여 처리하는 것이 가장 일반적인 생각인데, 여기서 부모 프로세스와 자식 프로세스 중에서 web_server를 통신을 가능하도록 해야할지가 가장 고민이 되었다. 이번 과제의 가장 큰 핵심은 부모 프로세스는 자식 프로세스를 생성을 하고 생성된 자식 프로세스는 클라이언트 요청을 처리하도록 소켓 통신을 이루어지며, 부모 프로세스는 다음 클라이언트 요청을 accept에서 대기하는 큰 틀을 이해해야 한다. 그렇지 않으면 처음 클라이언트의 다중 접속을 가능하게 하는데 어려움이 느꼈다. 추가적으로 그 접속 정보를 terminal에 출력을 하는데, 변수들의 독립성에 관해서 어려움을 느꼈다. 쉽게 말해서, 부모프로세스의 data, stack 등의 정보를 복사하는 자식 프로세스는 복사가 된 후에는 서로 독립적인 메모리에서 data 등을 사용하기 때문에, 변수들을 독립적으로 처리하여 std 터미널에 출력하는데 생각이 필요했다. 자식프로세스에서 count등을 증가 시켜도 부모프로세스는 영향을 받지 않기 때문에, 다음 클라이언트 요청에 따른 부모 프로세스의 count 등 처리가 자식 프로세스와 독립적이기 때문이다. 그래서, terminal에 클라이언트 정보를 출력하는 것은 모두 부모 프로세스에서 수행하지 않는다면, 매우 변수의 변하는 값을 감당하기 어려웠다. 그래서 부모프로세스 공간에 connect 메시지를 출력하도록 구현하고, disconnect는 자식프로세스가 종료될 때 발생하는 SIGCHLD가 발생하여 수행하는 함수에 disconnect 메시지를 출력하도록 하였다. 함수를 이용하기 때문에 출력에 필요한 정보의 변수는 전역변수를 이용하였다. 그 이외에 방법을 생각을 해보았지만, 가능한 방법이 이뿐이라고 생각이 들었다. 그리고 중요한 것이 wait인데, 이것은 원래 부모의 프로세스 위치에서 자식프로세스를 기다리고 자식프로세스를 정리하는 것인데, 이번 과제는 서버가 다중 클라이언트 접속을 기다리고 있어야하기 때문에 자식프로세스가 종료 위치에 wait를 추가하였다. 처음에는 부모프로세스의 역할 때문에 wait가 필요없다고 생각을 했지만, wait의 기능이 자식프로세스를 정리시키기 때문에, 추가를 하지 않으면 남아있는 것 때문에 좀비 프로세스가 발생하기 때문에 유의를 해야 한다.

추가적으로 ip에 관해서 매우 헷갈렸다. 내가 접속한 ip와 클라이언트 ip는 서로 달랐는데 조교님의 말을 들으니 공유기 등 이외에 관련이 될 수 있기 때문에 큰 문제는 없지만, 큰 문제는 없었다. 그리고 내가 구현 시 애를 먹었던 부분은, 익스플로어나 구글에서 클라이언트 요청을 보낼 경우 이상하게 가끔 처음부터 두 번의 connect가 요청이 되었다. 리눅스 가상머신상의 firefox에서는 이런 경우가 없기 때문에 오히려 더 혼란이 되었다. 이 부분이 무엇이 잘 못이 되었는지 찾아 보았지만, 부모프로세스가 accept를 기다리지 않고 한번더 connect가 되었다. 이 것은 처음 클라이언트 요청 시에만 발생하여 이유를 찾기 힘들었다. 그 이후 다중접속이나 소켓 통신에 대한 오류나 에러 발생이 없었고, 파이어폭스에 가능하기 때문에 큰 문제가 아니라고 생각 했다.

이번 과제도 추가적인 기능을 추가함으로써 마치 프로젝트처럼 완성도 높아지는 프로그램을 작성을 하는 것 같다. 난이도가 점점 생소해서 어려워지는 만큼 개념의 복습과 연습이 철저히 필요하다고 생각이 드는 과제였다.