# 시스템프로그래밍실습

#Assignment3-3

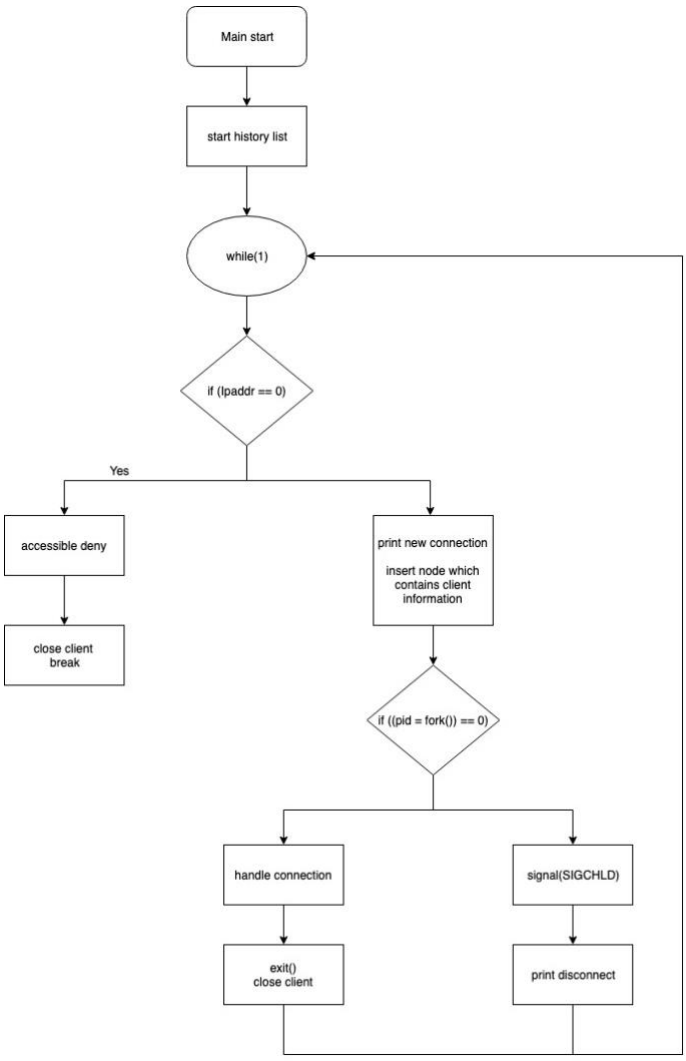
이름: 이진수
학번 : 2015722013
학과: 컴퓨터공학과
신영주 교수님

1. Introduction

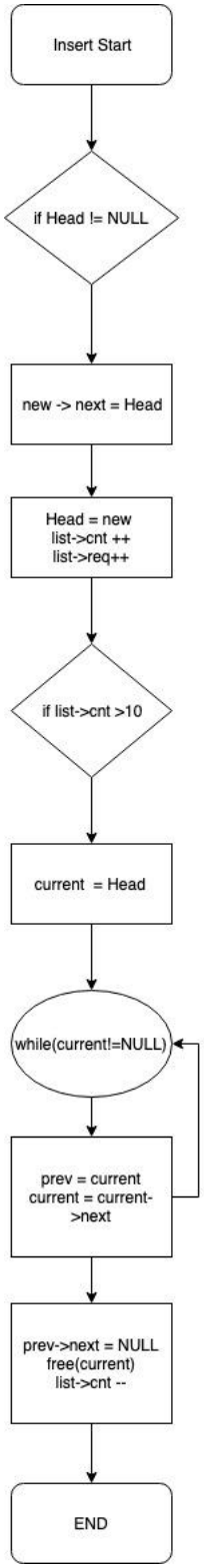
   이번 과제는 이전에 구현했던 socket program을 다중 client 가 접속 할 수 있게 끔 구현하는 것이다. Child processor를 통해서 connection을 처리 해주고 바로 접속을 끊어준다. Parent processor 에서는 child processor 가 disconnect가 될시에 정보를 받아 준다. Connection이 연결되고 끊길 때마다 ip 와 port number를 출력해 주고 10초에 한번씩 connection history를 최근 최대 10개 까지만 출력해준다. 접근 불가한 ip에 대해선 accessible deny 를 띄운다.

2. Flow chart

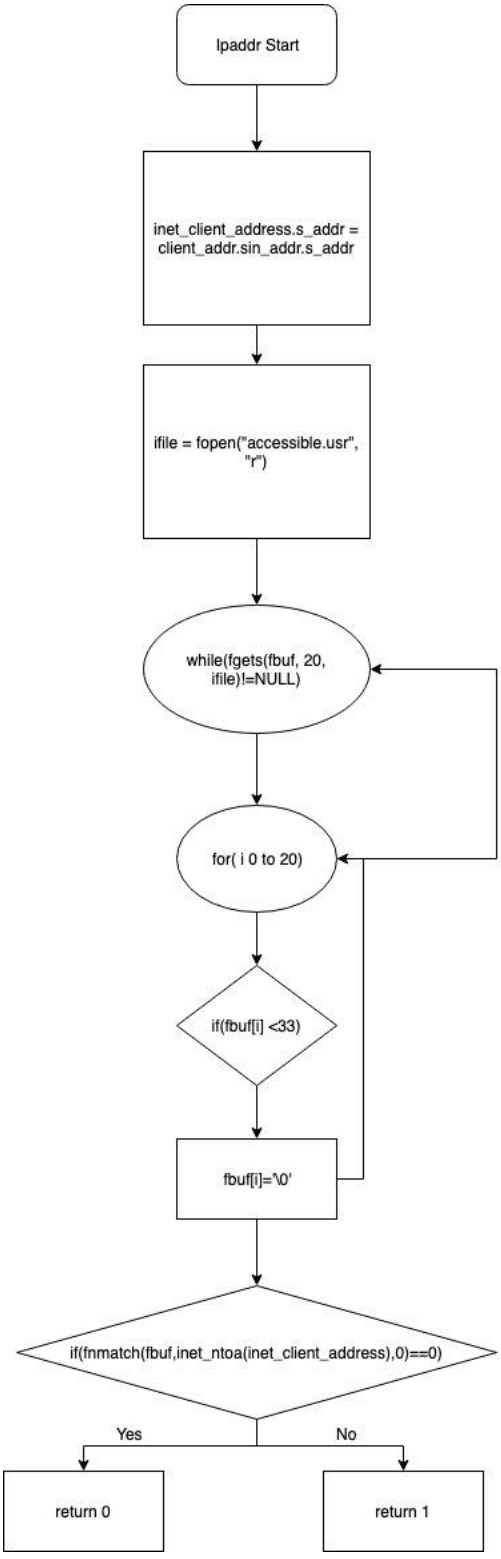
   - Main function flow chart




   - Insert flow chart




- Ipaddr flow chart




3. Pseudo code

- Pseudo code for Insert function

RInsert(){

```
        new -> ip = client ip number
        new -> pi = client pid
        new -> port = client port number
        new -> time = client connection time

        if(list->head != NULL){
                new -> next = list->head
        }

        list->head = new ;
        list -> cnt ++;
        list->req++;

        if(list->cnt>10){
                current = list->head
                while(until current -> next is null){
                        prev = current
                        current=current->next;
                }

                before the tail of list -> next = NULL
                free(tail of the list);
                list->cnt--;
        }

    }
```

- Pseduo code for Ipaddr function

```
Ipaddr(){
        inet_client_address.s_addr = client_addr.sin_addr.s_addr;

        ifile = open file "accessible.usr";

        char fbuf[20] = {0,};

        while(fgets(fbuf, 20, ifile)!=NULL) // get line by line
        {
                for(int i=0; i<20; i++){
```

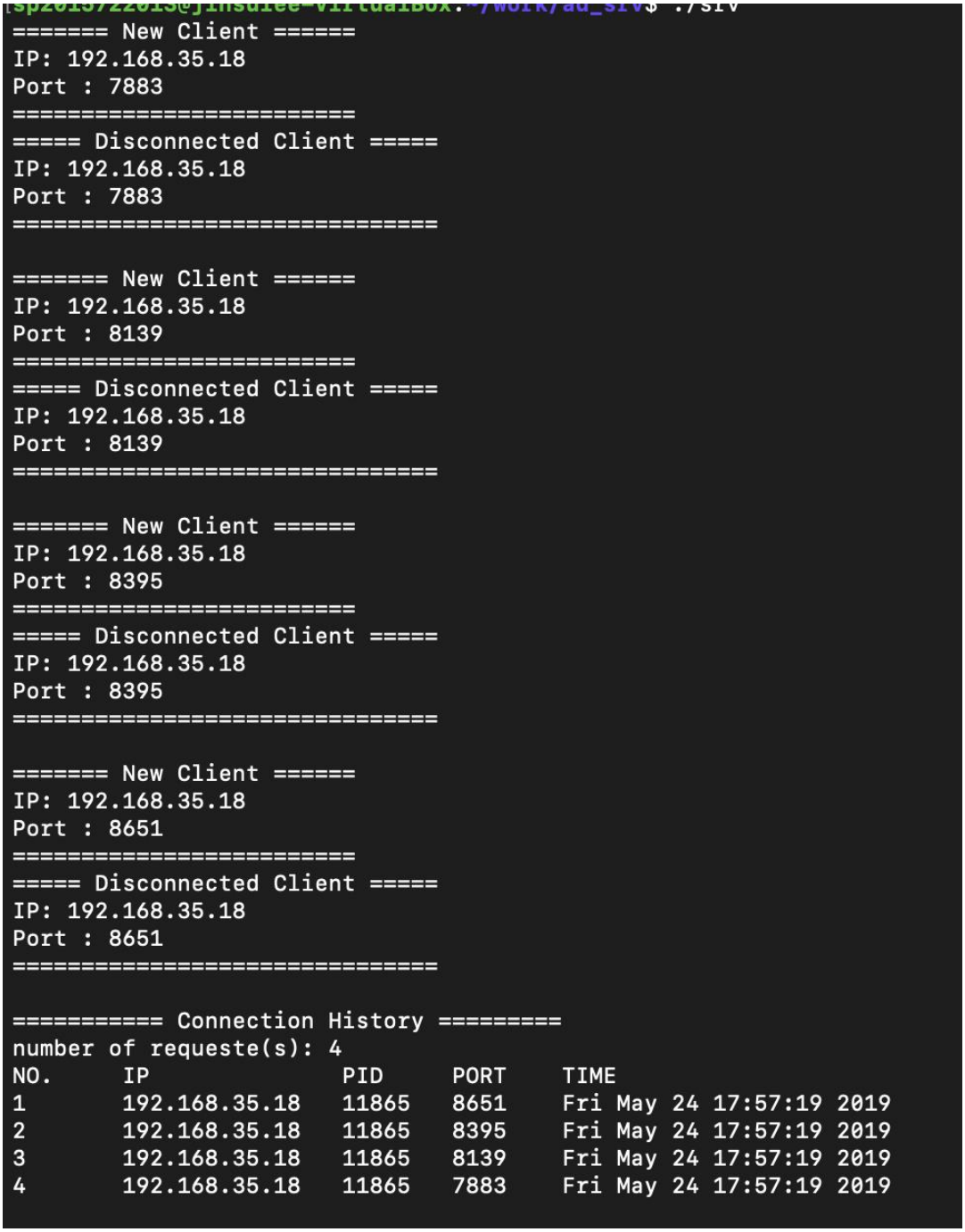
```
                        if(fbuf[i] != string){
                                fbuf[i] = '₩0';
                        }
                }

                if(fnmatch(fbuf, ip address of client)==0){
                        return 1;
                }
        }
        close(file);
        return 0;
    }
```

- Pseudo code for main function

```
main(){
        list = (Rlist *)malloc(sizeof(RList));

        connect server;
        bind server and socket_fd;
        listen(socket_fd);
        alarm(10) // alarm for 10 second

        while(1){
                client_fd = accept(socket_fd);

                inet_client_address.s_addr = get client's address;

                if(inet_client_address == '0.0.0.0'){
                        close(client_fd);
                        continue;
                }

                if(url == '/favicon.ico'){
                        close(client_fd);
                        continue;
                }

                print(" new client ")
```

```
                print(client's ip and port number);

                tim = get current time;
                RInsert(list, client's ip address, getpid(), client's port number, tim);

                if((pid = fork()== 0){
                        exit(0);
                }
                else{
                        signal(SIGCHLD, signalHandler);
                        print("disconnected client")
                        print(disconnected client's port number and ip address);
                }
        }
        else {
                continue;
        }

        close(socket_fd);
}
```

4. Result

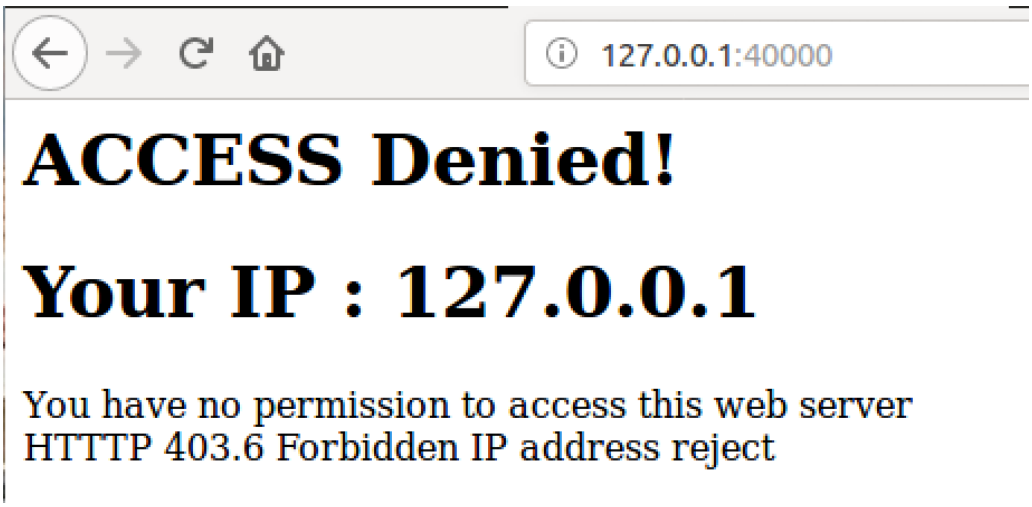



<최근 client가 가장 작은 number로 되어 있는 모습>

```
======= New Client ======
IP: 192.168.35.18
Port : 4043
=========================
===== Disconnected Client =====
IP: 192.168.35.18
Port : 4043
==============================

======= New Client ======
IP: 192.168.35.18
Port : 4299
=========================
===== Disconnected Client =====
IP: 192.168.35.18
Port : 4299
==============================

======= New Client ======
IP: 192.168.35.18
Port : 4555
=========================
===== Disconnected Client =====
IP: 192.168.35.18
Port : 4555
==============================

======= New Client ======
IP: 192.168.35.18
Port : 4811
=========================
===== Disconnected Client =====
IP: 192.168.35.18
Port : 4811
==============================

======= New Client ======
IP: 192.168.35.18
Port : 5067
=========================
===== Disconnected Client =====
IP: 192.168.35.18
Port : 5067
==============================

========== Connection History =========
number of requeste(s): 28
NO.     IP              PID     PORT    TIME
1       192.168.35.18   11834   5067    Fri May 24 17:56:02 2019
2       192.168.35.18   11834   4811    Fri May 24 17:56:02 2019
3       192.168.35.18   11834   4555    Fri May 24 17:56:02 2019
4       192.168.35.18   11834   4299    Fri May 24 17:56:02 2019
5       192.168.35.18   11834   4043    Fri May 24 17:56:02 2019
6       192.168.35.18   11834   3787    Fri May 24 17:56:02 2019
7       192.168.35.18   11834   3531    Fri May 24 17:56:02 2019
8       192.168.35.18   11834   3275    Fri May 24 17:56:02 2019
9       192.168.35.18   11834   3019    Fri May 24 17:56:02 2019
10      192.168.35.18   11834   2763    Fri May 24 17:56:02 2019
```

<Request가 10개 이상 들어올 시의 결과 화면>


<accessible.usr 안에 들어 있는 ip 주소들>




<접근 권한이 없는 ip로 접속을 시도 할 시 보이는 결과 화면>

5. Conclusion

이번 과제를 통해서 parent processor와 child processor가 어떻게 동작하는지 확실히 알게 되었다. 이번 과제는 child에서 모든 client connection을 처리 해주어서 저번과제에서 조금 더 발전만 시키면 구현이 가능했다. 하지만 connection history를 출력하기 위해 history list를 만들었는데 처음에는 list에 예상치 못한 client의 정보들이 들어 갔다. 이 이유는 저번 과제에서 favicon.ico가 들어왔을 시 혹은 다른 예외들을 while문 안에서 모두 continue 시켰는데 이번 과제에서 그냥 continue 시켜 버리면 parent 가 child processor의 정보를 받지 못한체로 child가 종료가 되지 않고 이상하게 진행되어 버린다. 그래서 모든 예외문에서 exit(0)을 해주었더니 list에 옳바르게 들어가는것을 확인 할 수 있었다. 수업 시간에 따라한 예시 코드만 봤을 때는 signalHandler를 왜 사용하는지 몰랐는데 이번 과제를 통해서 signal의 default action을 바꾸며 진행 해주어야 했기 때문에 사용하는 이유에 대해 정확하게 알게 되었다.