System Programming Final Exam, Summer, 2022, Class 1

12180626 성시열

아래 문제에 대해 **1) 전체 소스코드 캡쳐(일부 코드만 제시하면 안됨) 2) 실행 결과 3) 코드 및 실행결과**에 대한 설명 을 제시해야 합니다. word 나 pdf 로 제출하기 바랍니다. 모든 것이 오픈이지만 다른 사람에게서 도움을 받아서는 안됩니다.

FTP 서버와 클라이언트 프로그램을 작성하되 나이를 물어보고 나이가 18 이상이면 모든 파일을 허용하고 18 미만이면 아동용 파일만을 허용하도록 하시오. 파일 이름이 A로 시작하는 파일은 아동용 파일이 아니고 성인용 파일로 가정합니다. 18 미만의 사용자가 성인용 파일을 요청하면 2번까지 재시도를 허용하고 3번째는 연결을 끊습니다. 스트링을 숫자로 변환하기 위해 atoi 함수를 사용할 수 있습니다. 서버는 아래 조건을 만족해야 합니다.

- select를 사용해서 여러 클라이언트를 동시에 핸들할 수 있어야 하며
- 파일 사이즈에는 제한이 없어야 합니다.
- 그리고 중첩루프를 사용하면 안됩니다.

client1 -> server: hello
client2 -> server: hello
client3 -> server: hello

server -> client1: age?
client1 -> server: 17
server -> client2: age
client2 -> server: 17
server -> client3: age?
client3 -> server: 19

server -> client1: file name?
client1 -> server: Af1
server -> client1: not allowed. 1 fail. try again.

server -> client2: file name?
client 2-> server: f1
server -> client2: (the content of f1)

server -> client1: file name?
client1 -> server: Af2
server -> client1: not allowed. 2 fail. try again.

server -> client3: file name?
client3 -> server: Af
server -> client3: (the content of Af1)

server -> client1: file name?
client1 -> server: Af3
server -> client1: not allowed. 3 fail. disconnecting...

cliping8.c

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/select.h>

#define SERV_TCP_PORT 62608
#define SERV_ADDR "165.246.38.151"

void main(){
   int x,y;
   struct sockaddr_in serv_addr;
   char buf[50];
   printf("Hi, I am the client\n");

   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family=PF_INET;
   serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
   serv_addr.sin_port=htons(SERV_TCP_PORT);

   //open a tcp socket
   if ((x=socket(PF_INET, SOCK_STREAM, 0))<0){
      printf("socket creation error\n");
      exit(1);
   }
   printf("socket opened successfully. socket num is %d\n", x);

   //connect to the server
   if (connect(x, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0){
      printf("can't connect to the server\n");
      exit(1);
   }
```

연결된 클라이언트의 socket 번호가 4번부터 시작이므로 해당 값을 조절하여 cli_num을 생성한다.

```
  int f, i;
  for(i=0; i<3; i++){
     f = fork();
     if(f==0){ //child
        printf("client %d -> server : ", cli_num);
        gets(buf);
        write(x, buf, strlen(buf));
     }
     else{
        y = read(x, buf, 50);
        write(1, buf, y);
        printf("\n");
     }
  }
  close(x);
}
```

여러 개의 클라이언트를 핸들링해야하기 때문에 fork 를 통해 복제를 진행하고

Child 일 때 값을 입력하고 parent 일 때 값을 출력하는 코드를 작성하였다.

servping8.c

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/select.h>
#include <fcntl.h>

#define SERV_TCP_PORT 62608
#define SERV_ADDR "165.246.38.151"

struct client{
   char age[5];
   char adu_chi[10];
};

struct client cli[50];

int state[50];
void handle_protocol(int x, fd_set * pset, int state[]);
void handle_state_1(int x, fd_set * pset, char * buf, int state[]);
void handle_state_2(int x, fd_set * pset, char * buf, int state[]);
void handle_state_3(int x, fd_set * pset, char * buf, int state[]);

void main(){
   int s1,s2, i, x, y;
   struct sockaddr_in serv_addr, cli_addr;
   char buf[50];
   socklen_t  xx;

   printf("Hi, I am the server\n");

   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family=PF_INET;
   serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
   serv_addr.sin_port=htons(SERV_TCP_PORT);

   //open a tcp socket
   if ((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
      printf("socket creation error\n");
      exit(1);
   }
   printf("socket opened successfully. socket num is %d\n", s1);
```

먼저 클라이언트의 나이와, 성인인지 아동인지 구분할 수 있는 변수를 struct client 에 추가하였다.

이후 handle_protocol 함수와 handle_state_1,2,3 함수 사용을 위해 함수 프로토타입을 선언하였다.

상태를 저장할 수 있는 state 정수형 배열을 선언하였다.

```
   // bind ip
   x =bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
   if (x < 0){
      printf("binding failed\n");
      exit(1);
   }
   printf("binding passed\n");
   listen(s1, 5);
   xx = sizeof(cli_addr);

   fd_set rset, pset;
   int maxfd=50; // just monitor max 50 sockets
   int state[50] = { 0 };
   FD_ZERO(&rset); // init rset
   FD_ZERO(&pset); // init pset

   // step 1. monitor conn req packet
   FD_SET(s1, &pset);
   // and loop on select
   for(i=0;i<1000;i++){ // should be infinite loop in real life
      rset=pset;  // step 2
      select(maxfd, &rset, NULL, NULL, NULL); // step 3
      // now we have some packets
      for(x=0;x<maxfd;x++){ // check which socket has a packet
         if (FD_ISSET(x, &rset)){ // socket x has a packet
            // s1 is a special socket for which we have to do "accept"
            // otherwise do ping-pong-pang-pung
            if (x==s1){ // new client has arrived
               // create a socket for this client
               s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
               printf("new cli at socket %d\n",s2);
               state[s2] = 1;
               FD_SET(s2, &pset); // and include this socket in pset
            }else{ // data packet. do ping-pong-pang-pung protocol
               handle_protocol(x, &pset, state);
            }
         }
      }
   }
}
```

Main 함수 내에서 state[50]을 0으로 초기화해주고

Select를 활용하여 여러 클라이언트 관리가 용이하도록 코드를 구성한다.

조건문을 통해

만약 새로운 클라이언트가 연결된다면 (if(x==s1)) accept를 진행하고 state[s2]를 1로 지정해준다.

이후 기존 클라이언트와 연결되었을 때 handle_protocol 함수를 호출한다.

```
void handle_protocol(int x, fd_set * pset, int state[]){
   int y; char buf[50];
   y=read(x, buf, 50); // read data
   buf[y]=0; // make it a string
   if (state[x] == 1){ // if it is a ping
      handle_state_1(x, pset, buf, state);
   }
   else if (state[x] == 2){
      handle_state_2(x, pset, buf, state);
   }
   else if (state[x] == 3){
      handle_state_3(x, pset, buf, state);
   }
}

void handle_state_1(int x, fd_set * pset, char * buf, int state[]){
   if(strcmp(buf, "hello")==0){
      write(x, "age?", 4);
      state[x] = 2;
   }
}

void handle_state_2(int x, fd_set * pset, char * buf, int state[]){
   strcpy(buf, cli[x].age);
   int ages;
   ages = atoi(cli[x].age);
   if(ages >= 18){
      strcpy(cli[x].adu_chi, "adult");
   }
   else {
      strcpy(cli[x].adu_chi, "child");
   }
   write(x, "file name?", 10);
   state[x] = 3;
}
```

함수 handle_protocol는 클라이언트로부터 받은 값을 buf에 받아오고
state[x]의 값에 맞게 handle_state_1,2,3,를 순차적으로 호출하는 역할을 한다.

handle_state_1에서는 입력받은 값 buf가 hello일 때 서버에서는 클라이언트에게 age?라는 문자열을 넘겨주고 state[x] = 2로 변경한다.
hadle_state_2에서는 클라이언트로부터 나이를 입력받게 되는데
입력받은 값 buf를 cli[x].age에 저장하고
정수형 변수 ages를 선언하여 문자열로 저장된 나이를 숫자로 바꿔준다.
이후 조건문을 활용하여 입력받은 나이값이 18이상일 때 clil[x].adu_chi에 adult라는 문자열을 저장하고, 그렇지 않을 때(18미만) clil[x].adu_chi에 child라는 문자열을 저장한다.
서버에서는 클라이언트에게 file name?라는 문자열을 넘겨주고 state[x] = 3으로 변경한다.

```
void handle_state_3(int x, fd_set * pset, char * buf, int state[]){
   int y;
   char fname[50];
   strcpy(fname, buf);

   if(strcmp(cli[x].adu_chi, "child")==0 && fname[0] == "A"){
      write(x, "not allowed. 1 fail. try again.", 31);
      write(x, "file name?", 10);
      y = read(x, buf, 50);
      buf[y] = 0;
      if(strcmp(cli[x].adu_chi, "child")==0 && fname[0] == "A"){
         write(x, "not allowed. 2 fail. try again.", 31);
         write(x, "file name?", 10);
         y = read(x, buf, 50);
         buf[y] = 0;
      }
         if(strcmp(cli[x].adu_chi, "child")==0 && fname[0] == "A"){
            write(x, "not allowed. 3 fail. disconnecting", 34);
            close(x);
         }
   }
   int fd = open(fname, O_RDONLY, 00777);
   if(fd<0){
      printf("file not found");
      close(x);
      exit(0);
   }
   for(;;){
      y = read(fd, buf, 50);
      if(y==0) break;
      write(x, buf, y);
   }
   close(x);
   exit(0);
}
```

handle_state_3 에서는 사용자로부터 받아온 파일이름이 buf에 저장되었을 것이다. 이를 루믄에 옮긴다. 이전 함수에서 마킹한 것을 활용하여, cli[x].adu_chi가 child라는 문자열이며, 파일이름 첫 글자가 A라는 문자일 때 다음과 같은 오류처리를 진행한다. 중첩 조건문을 활용하여 2번까지 재입력이 가능하고 3번째는 연결을 끊는다.

만약 조건에 성립하지 않는다면 파일 내용을 전달하는 철차에 맞게 파일내용을 클라이언트에게 보낸다.

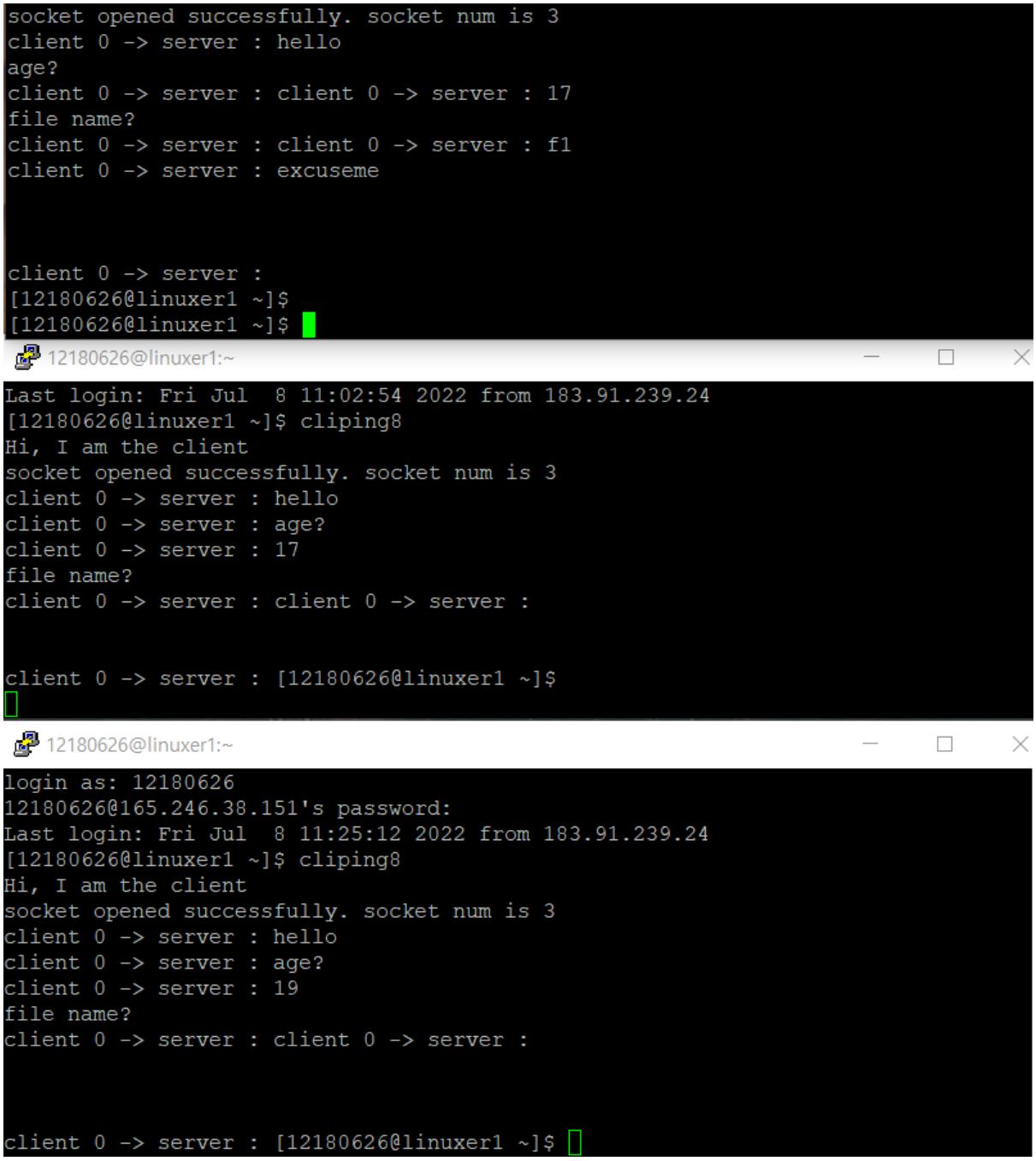
무한루프와 break를 사용하여 파일크기에 상관없이 파일을 보낼 수 있다.

```
[12180626@linuxer1 ~]$ servping8
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
new cli at socket 4
```

```
[12180626@linuxer1 ~]$ vi cliping8.c
[12180626@linuxer1 ~]$ cliping8
Hi, I am the client
socket opened successfully. socket num is 3
client 1 -> server : hello
age? : server -> client 1
client 1 -> server : 17
file name? : server -> client 1
client 1 -> server : f1
excuseme
```

컴파일 후 실행한 뒤 하나의 파일을 실행할 때는 f1 파일의 내용을 적절하게 출력한 것을 알 수 있다.

```
[12180626@linuxer1 ~]$ servping8
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
new cli at socket 4
new cli at socket 5
new cli at socket 6
```



여러 개의 클라이언트를 실행한 결과 연결이 잘 되었고, 나이 수집까지는 잘 되었으나

첫번째 클라이언트의 파일만 출력한 뒤 모든 클라이언트의 연결이 끊기는 것을 알 수 있었다.