process	body(stack data code) + process descriptor(pid state mm)
fork x = fork();	process를 복제 (child, parent) 1) body를 복사 2) p.d를 복사 3) 붙여넣기 된 child의 p.d 수정(PID, mm) 4) child 실행 후 return 0 5) parent 실행 후 return child ID ● hw7(6) : fork() 두 번 ● fork() n번으로 인해 생성되는 process 개수 : 2^n
exec char *x[10]; execve(x[0], x, 0);	process를 다른 process로 변경 x[0] : 명령어 (“/bin/cat”) x[1] : 필요한 아규먼트 (“f1”) x[2] : 마무리는 0 (0) ● hw8(6) : execve로 명령어처럼 만들기 + 반복 ● hw8(7) : (6)에서 getcwd를 통해 현재경로까지
exit exit(1);	프로그램이 종료되면 자동으로 exit() 호출 exit()를 만나면 body만 종료되고 p.d가 살아있는 zombie 상태가 됨
wait()	child process가 끝날 때까지 대기 --> child가 끝나면 exit()를 자동호출하여 body만 제거 --> wait()에서 p.d까지 제거 완료
shell	hw8과 비슷하게 명령어처럼 만드는 것 ● hw9(4) : /bin/ 포함 명령어처럼 ● hw9(5) : shell에서 & 핸들링 → 솔루션보기 ● hw9(6) : /bin/ 입력 X --> 진짜 명령어처럼 ● hw9(7) : 환경변수 활용해 /bin/이상 명령어도 OK → 솔루션보기
getpid() kill(getpid(). 15);	현재 process ID return
getppid() kill(getppid(). 15);	현재 process parent ID return

● hw10(3) → 솔루션 : 둘이서 채팅 + bye 만나면 탈출 + 무한루프 막기

● hw11(5) → cli.c // serv.c에 저장 : FTP + 여러 client 받아오기

● hw12(3) : ping pong pang pung 하고 complated 까지

● hw13(6) → cliping.c // servping.c

: client(name, age, partner) + ping pong pang pung + 5번 반복 + fork()

1) ping 받고 pong 보내기
2) pang 받고 pung. name? 보내기
3) 이름 받고 age? 보내기
4) 나이 받고 partner? 보내기
5) 파트너 받고 start 보내기
6) 누가 누구에게 어떤걸 보냈는지

● hw13(7) → cliping7.c // servping7.c (솔루션보기)

1) hello를 받고, name? 보내기
2) 내 ID 받고, ID를 저장 / ready? 보내기
3) yes를 받고, list 출력(sprintf)
4) 파트너 ID 받고, 파트너에 저장 / go 보내기
5) 채팅을 보내면 누가 누구한테 어떤 내용을 보내는지

select 문제는

- cliping7.c 가져오기

1) state 개수를 통해 입력의 반복이 몇 번인지 파악하기 (fork() 위의 반복문만 보기)

- servping7.c 가져오기

1) handle_protocol으로 state[x] 조건문을 통해 함수 호출
2) 만들어야할 실행문을 기존 실행문을 활용하여 제작 (hw13의 p.16~ 참고)

```
void handle_state_3(int x, fd_set * pset, char *buf, int state[]) {
   if(strcmp(buf, "yes") == 0) {
        printf("cli %s => serv: %s\n", cli[x].name, buf);
        int i; char sbuf[1000];
        strcpy(sbuf, "client list ( ");
        for(i=0; i<50; i++) {
           if(i != x && state[i] >= 3) {
                strcat(sbuf, cli[i].name);
                strcat(sbuf, " ");
           }
        }
        strcat(sbuf, ")");
        write(x, sbuf, strlen(sbuf));
        printf("serv => cli %s: client list (...)\n", cli[i].name);
        state[x] = 4;
   }
   else {
        printf("protocol error from socket %d, disconnecting\n", x);
        write(x, "protocol error", 14);
        close(x);
        FD_CLR(x, pset);
   }
}
```

if(i != x && state[i] >= 3) ready에 yes가 됐니? 라는 뜻

```
void handle_state_5(int x, fd_set * pset, char *buf, int state[]) {
   int i;
   char sbuf[70];
   printf("cli %s => serv: %s\n", cli[x].name, buf);
   printf("serv => other cli: %s\n", buf);
   for(i=0; i<50; i++) {
        if(i != x && state[i]==5) {
                sprintf(sbuf, "%s %s to %s %s: %s", cli[x].name, cli[x].age
, cli[i].name, cli[i].age, buf);
                write(i, sbuf, strlen(sbuf));
        }
   }
}
```

if(i != x && state[i] == 5) state[x]4까지 모두 완료한 친구들이니?

cli.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/stat.h>
#include <unistd.h>
#include <fcntl.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];
   char fname[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);
   }

   // send msg to the server
   // hello
   printf("now I am connected to the server.\n");
   write(x, "hello", 5);
   printf("client => server : hello\n");
   // read from server
   // which file?
   y=read(x, buf, 50);
   buf[y]=0;
   printf("server => client : %s\n", buf);

   // send file name to the server
   // file name
   printf("client => server : ");
   gets(fname);
   write(x, fname, 50);
   // content of file
   printf("server => client : file contents\n");
   for(;;){
      y = read(x, buf, 50);
      if(y==0) break;
      write(1, buf, y);
   }
   close(x);  // disconect the communication
}
```

serv.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/stat.h>
#include <unistd.h>
#include <fcntl.h>

#define SERV_TCP_PORT 62608
#define SERV_ADDR "165.246.38.151"

void main(){
   int s1,s2, x, y, i, ch;
   int file;
   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);

   // 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);
   //printf("we passed accept. new socket num is %d\n", s2);
```

```
   for(i=0; i<20; i++){
      // read msg from client
      // hello
      s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
      ch = fork();
      if(ch==0){
         printf("we passed accept. new socket num is %d\n", s2);
         y=read(s2, buf, 50);
         buf[y]=0;
         if(strcmp(buf, "hello")==0) {
            printf("client => server : %s\n", buf);

            //which file?
            write(s2, "which file?", 11);
            printf("server => client : which file?\n", buf);

            //file name
            y = read(s2, buf, 50);
            buf[y] = 0;
            printf("client => server : %s\n", buf);

            file = open(buf, O_RDONLY, 00777);
            if (file < 0){
               printf("file not found\n");
               close(s2);
               close(s1);
               exit(0);
            }

            //content of file
            printf("server => client : contents of file\n", buf);
            for(;;){
               y = read(file, buf, 50);
               if (y==0) break;
               write(s2, buf, y);
               write(1, buf, y);
            }
            printf("\n");
            close(file);
         }
         close(s2);   // disconnect the connection
         close(s1);   // close the original socket
         exit(0);
      }
      close(s2);
   }
   close(s1);
   exit(1);
                                                                  94,1          97%

[12180626@linuxer1 ~]$ serv
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
client => server : hello
server => client : which file?
client => server : f1
server => client : contents of file
excuseme

[12180626@linuxer1 ~]$ cli
Hi, I am the client
socket opened successfully. socket num is 3
now I am connected to the server.
client => server : hello
server => client : which file?
client => server : f1
server => client : file contents
excuseme
```