

**TI DSP, MCU 및 Xilinx Zynq
FPGA
프로그래밍 전문가 과정**

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 문한나

mhn97@naver.com

2 개월차 월말평가 오답노트

1. 파이프 통신을 구현하고 c type.c 라고 입력할 경우 현재 위치의 디렉토리에 type.c 파일을 생성하도록 프로 그래밍하시오.

```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

char *check_text(char *text){

    int i;
    static char filename[1024];
    int text_len = strlen(text);
    if(text[0] != 'c' && text[1] != ' ')
        return NULL;
    if(text[text_len -1] != 'c' && text[text_len -2] != '.')
        return NULL;
    for(i=2; i< text_len -2; i++){
        if(text[i] == ' ' || text[i] == '\t')
            return NULL;
        filename[i-2] = text[i];
    }
    strcat(filename, ".c");
    return filename;
}

int main(void){

    int fd, ret, fo;
    char *string = NULL;
    char buf[1024];
    fd = open("myfifo",O_RDWR);
    fcntl(0,F_SETFL,O_NONBLOCK);
    fcntl(fd,F_SETFL,O_NONBLOCK);

    for(;;){

        if((ret = read(0,buf,sizeof(buf)))>0){

            buf[ret-1]=0;
            printf("Keyboard input : [%s]\n",buf);
            string = check_text(buf);
            if(string != NULL){
                printf("String : %s\n", string);
            }
        }
        if((ret = read(fd,buf,sizeof(buf)))>0){
            buf[ret-1]=0;
            printf("pipe input : [%s]\n",buf);
            string = check_text(buf);
            if(string != NULL){
```

```

        printf("String : %s\n", string);
    }

    }
    fo = open(string, O_WRONLY | O_CREAT, 0644);
    close(fo);
}
close(fd);
return 0;
}

```

```

mhn@mhn-Z20NH-AS51B5U:~/linux/note/1$ gcc 1.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/1$ mkfifo myfifo
mhn@mhn-Z20NH-AS51B5U:~/linux/note/1$ ./a.out 1.c
pipe input : [hi~]
hello~~~
Keyboard input : [hello~~~]
c test1.c
Keyboard input : [c test1.c]
String : test1.c
pipe input : [c test2.c]
String : test2.c.c
^C
mhn@mhn-Z20NH-AS51B5U:~/linux/note/1$ ls
1.c a.out myfifo test1.c test2.c.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/1$

```

5. 리눅스에는 여러 장점이 있다. 아래의 장점들 각각에 대해 기술하라.

* 사용자 임의대로 재구성이 가능하다.

리눅스 커널이 F/OSS 에 속하여 있으므로 소스 코드를 원하는 대로 수정할 수 있다.

그러나 License 부분을 조심해야 한다.

* 열악한 환경에서도 HW 자원을 적절히 활용하여 동작한다.

리눅스 커널이 가볍고 좋을뿐만 아니라 소스가 공개되어 있어

다양한 분야의 사람들이 지속적으로 개발하여 어떠한 열악한 환경에서도 잘 동작한다.

* 커널의 크기가 작다.

최적화가 잘 되어 있다는 뜻

* 완벽한 멀티유저, 멀티태스킹 시스템

리눅스는 RT Scheduling 방식을 채택하여 Multi-Tasking 을 효율적으로 잘 해낸다.

* 뛰어난 안정성

네트워크 장비에 들어가며, 전 세계의 많은 개발자들이 지속적으로 유지보수하여 안정성이 뛰어나

* 빠른 업그레이드

전 세계의 많은 개발자들이 지속적으로 유지보수를 하고 있음

* 강력한 네트워크 지원

TCP/IP Stack 이 원채 잘 되어 있다보니 Router 및 Switch 등의 장비에서 사용함

* 풍부한 소프트웨어

GNU(GNU is Not Unix) 정신에 입각하여 많은 Tool 들이 개발되어 있다.

43 또한 Page Directory 를 가르키는 Intel 전용 Register 가 존재한다. 이 Register 의 이름을 적고 ARM 에서 이 역할을 하는 레지스터의 이름을 적으시오.

Intel 의 경우엔 CR3
ARM 의 경우엔 CP15

45. 메모리를 불연속적으로 할당하는 기법은 무엇인가 ?

vmalloc()

46. 메모리를 연속적으로 할당하는 기법은 무엇인가 ?

kmalloc()

51. Device Driver 는 Major Number 와 Minor Number 를 통해 Device 를 관리한다.
실제 Device 의 Major Number 와 Minor Number 를 저장하는 변수는 어떤 구조체의 어떤 변수인가 ?
(역시 Task 구조체에서부터 쪽 찾아오길 바람)

task_struct 내에 files_struct 내에 file 내에 path 내에 dentry 내에
inode 구조체에 존재하는 i_rdev 변수에 저장한다.

58. Compiler 의 Instruction Scheduling 은 Run-Time 이 아닌 Compile-Time 에 결정된다.
고로 이를 Static Instruction Scheduling 이라 할 수 있다. Intel 계열의 Machine 에서는 Compiler 의 힘을 빌리지 않고도 어느정도의 Instruction Scheduling 을 HW 의 힘만으로 수행할 수 있다.
이러한 것을 무엇이라 부르는가 ?

Dynamic Instruction Scheduling

60. CPU 들은 각각 저마다 이것을 가지고 있다.
Compiler 개발자들은 이것을 고려해서 Compiler 를 만들어야 한다. 또한 HW 입장에서 이것을 고려해서 설계를 해야 한다. 여기서 말하는 이것이란 무엇인가 ?

ISA(Instruction Set Architecture) : 명령어 집합

61. Intel 의 Hyper Threading 기술에 대해 상세히 기술하시오.

62. 그동안 많은 것을 배웠을 것이다.
최종적으로 Kernel Map 을 그려보도록 한다. (Networking 부분은 생략해도 좋다)
예로는 다음을 생각해보도록 한다. 여러분이 좋아하는 게임을 더블 클릭하여 실행한다고 할 때
그 과정 자체를 Linux Kernel 에 입각하여 기술하도록 하시오. (그림과 설명을 같이 넣어서 해석하도록 한다)
소스 코드도 함께 추가하여 설명해야 한다.

64. 서버와 클라이언트가 1 초 마다 Hi, Hello 를 주고 받게 만드시오.

<chat_serv.c>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/epoll.h>
```

```

#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 128
#define MAX_CLNT 256

typedef struct sockaddr_in si;
typedef struct sockaddr *sp;

int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
pthread_mutex_t mtx;
char rcv_msg[BUF_SIZE];

void err_handler(char *msg){

    fputs(msg,stderr);
    fputc('\n',stderr);
    exit(1);

}

void send_msg(char *msg, int len){

    int i;
    char msg_hi[] = "Hi~!";
    pthread_mutex_lock(&mtx);

    for(;;){

        write(1,msg,len);
        printf("\n");
        sleep(1);
    }

    pthread_mutex_unlock(&mtx);

}

void *rcv_msg(void *arg){

    int clnt_sock = *((int *)arg);
    int str_len = 0;
    char buf[1024];
    char msg_hi[] = "Hi~!";

    for(;;){

        write(clnt_sock, msg_hi, strlen(msg_hi));
        sleep(1);

    }

}

void *clnt_handler(void *arg){

    int clnt_sock = *((int *)arg);
    int str_len = 0, i;
    char msg[BUF_SIZE];

```

```

char msg_hi[] = "Hi~!";
char buf[1024];

        while((str_len = read(clnt_sock,msg,sizeof(msg))) != 0)
            send_msg(msg,str_len);

pthread_mutex_lock(&mtx);

for(i=0;i<clnt_cnt;i++){

        if(clnt_sock == clnt_socks[i]){
            while(i++ < clnt_cnt -1)
                clnt_socks[i] = clnt_socks[i+1];
            break;
        }
    }

    clnt_cnt--;
    pthread_mutex_unlock(&mtx);
    close(clnt_sock);

    return NULL;
}

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

    int serv_sock,clnt_sock;
    si serv_addr,clnt_addr;
    socklen_t addr_size;
    pthread_t t_id,rcv_thread;
    char msg_hi[] = "Hi~!";

    if(argc != 2){

        printf("Usage: %s <port>\n",argv[0]);
        exit(1);

    }

    pthread_mutex_init(&mtx,NULL);

    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]));

    if(bind(serv_sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("bind() error!");

    if(listen(serv_sock, 10) == -1)
        err_handler("listen error");

    for(;;){

```

```

        addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);

        pthread_mutex_lock(&mtx);
        clnt_socks[clnt_cnt++] = clnt_sock;
        pthread_mutex_unlock(&mtx);

        pthread_create(&t_id, NULL, clnt_handler, (void *)&clnt_sock);
        pthread_create(&rcv_thread, NULL, rcv_msg, (void *)&clnt_sock);
        pthread_detach(t_id);
        pthread_detach(rcv_thread);
        printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));

    }

close(serv_sock);

return 0;

}

```

<chat_clnt.c>

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/epoll.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 128
#define NAME_SLNT 32

typedef struct sockaddr_in si;
typedef struct sockaddr *sp;

char name[NAME_SLNT] = "[DEFAULY]";
char msg[BUF_SIZE];

void err_handler(char *msg){

    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);

}

void *send_msg(void *arg){

    int sock = *((int *)arg);
    char name_msg[NAME_SLNT + BUF_SIZE];
    char msg_hello[] = "Hello~!";
    char buf[1024];

    for(;;){
        write(sock, msg_hello, strlen(msg_hello));
        sleep(1);
    }
}

```

```

        return NULL;
    }

void *recv_msg(void *arg){

    int sock = *((int *)arg);
    char name_msg[NAME_SLNT+BUF_SIZE];
    char buf[1024];
    int str_len;

    for(;;){

        str_len = read(sock,buf,strlen(buf));
        if(str_len == -1)
            return (void *)-1;

        write(1, buf, str_len);
        printf("\n");
        sleep(1);
    }
    return NULL;
}

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

    int sock;
    si serv_addr;
    pthread_t snd_thread, rcv_thread;
    void *thread_ret;

    if(argc != 4){

        printf("Usage: %s <IP> <port> <name>\n",argv[0]);
        exit(1);

    }

    sprintf(name, "[%s]",argv[3]);
    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(atol(argv[2]));

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

    pthread_create(&snd_thread,NULL,send_msg,(void *)&sock);
    pthread_create(&rcv_thread,NULL,recv_msg,(void *)&sock);
    pthread_join(snd_thread, &thread_ret);
    pthread_join(rcv_thread, &thread_ret);

    close(sock);

    return 0;
}

```



```
mhn@mhn-900X3L:~/my_proj/linux/t/test$ gcc -o chat_serv chat_serv.c -lpthread  
mhn@mhn-900X3L:~/my_proj/linux/t/test$ ./chat_serv 7777  
Connected Client IP: 127.0.0.1  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
Hello~!  
^C  
mhn@mhn-900X3L:~/my_proj/linux/t/test$
```

```
mhn@mhn-900X3L:~/my_proj/linux/t/test$ gcc -o chat_clnt chat_clnt.c -lpthread  
mhn@mhn-900X3L:~/my_proj/linux/t/test$ ./chat_clnt 127.0.0.1 7777 test  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
Hi~!  
^C  
mhn@mhn-900X3L:~/my_proj/linux/t/test$
```

Process 는 자신의 고유한 공간으로 가상의 4GB 를 가지고 있다.
실제 이 공간을 모두 사용하지 않으며 Demand Paging 에 따라 필요한 경우에만
실제 물리 메모리 공간을 Paging Mechanism 을 통해 할당받아 사용한다.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

int area[1024];

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

    int i;
    int ret;
    int index;
    int fd;
    char buf[1024];

    fd = open(argv[1], O_RDONLY);
```

```

ret = read(fd, buf, sizeof(buf));

for(i = 0, index = 1; buf[i]; i++){

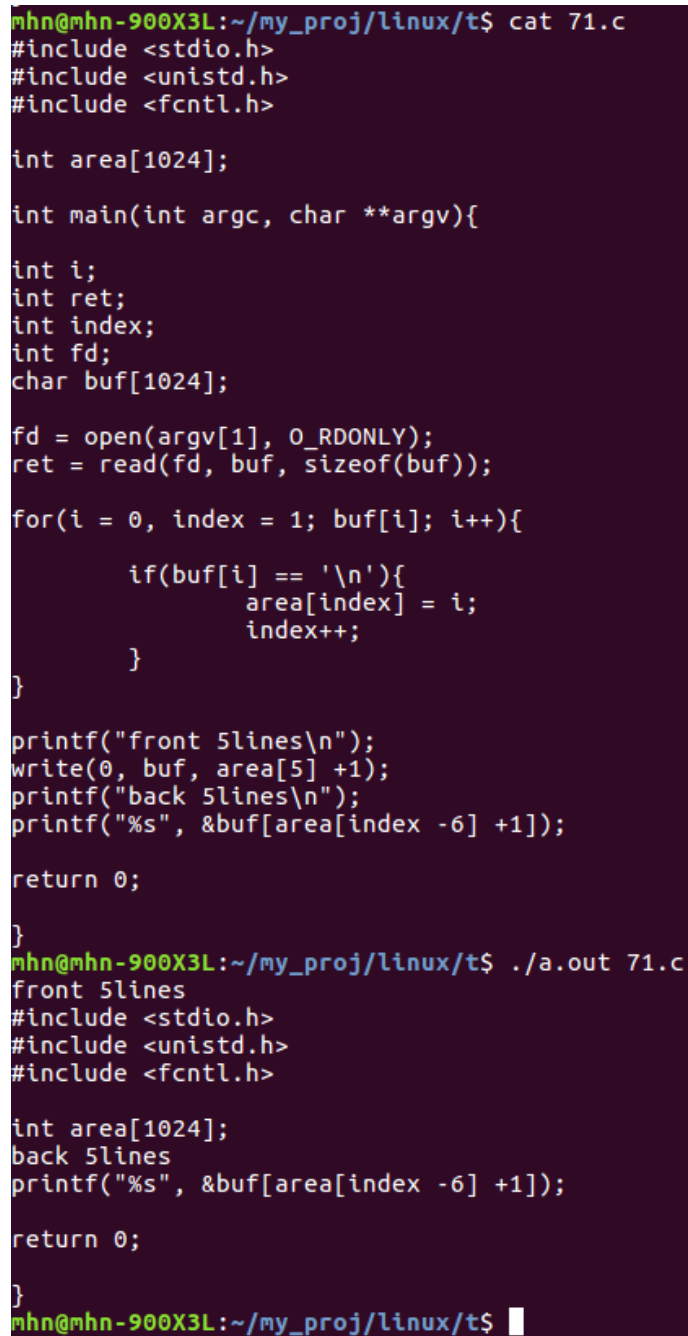
    if(buf[i] == '\n'){
        area[index] = i;
        index++;
    }
}

printf("front 5lines\n");
write(0, buf, area[5] +1);
printf("back 5lines\n");
printf("%s", &buf[area[index -6] +1]);

return 0;

}

```



```

mhn@mhn-900X3L:~/my_proj/linux/t$ cat 71.c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

int area[1024];

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

    int i;
    int ret;
    int index;
    int fd;
    char buf[1024];

    fd = open(argv[1], O_RDONLY);
    ret = read(fd, buf, sizeof(buf));

    for(i = 0, index = 1; buf[i]; i++){

        if(buf[i] == '\n'){
            area[index] = i;
            index++;
        }
    }

    printf("front 5lines\n");
    write(0, buf, area[5] +1);
    printf("back 5lines\n");
    printf("%s", &buf[area[index -6] +1]);

    return 0;

}
mhn@mhn-900X3L:~/my_proj/linux/t$ ./a.out 71.c
front 5lines
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

int area[1024];
back 5lines
printf("%s", &buf[area[index -6] +1]);

return 0;

}
mhn@mhn-900X3L:~/my_proj/linux/t$

```

81. stat(argv[2], &buf)일때 stat System Call 을 통해 채운 buf.st_mode 의 값에 대해 기술하시오.

buf.st_mode 에는 리눅스 커널 inode 의 i_mode 와 같은 값이 들어가 있다.

파일의 종류 4 비트와 setuid, setgid, sticky bit, 그리고 rwx 가 3 개씩 존재한다.

82. 프로세스들은 최소 메모리를 관리하기 위한 mm, 파일 디스크립터인 fd_array, 그리고 signal 을 포함하고 있는데 그 이유에 대해 기술하시오.

자신이 실제 메모리 어디에 위치하는지에 대한 정보가 필요하고

또 자신이 하드 디스크의 어떤 파일이 메모리에 로드되어

프로세스가 되었는지의 정보가 필요하며

마지막으로 프로세스들 간에

signal 을 주고 받을 필요가 있기 때문에 signal 이 필요하다.

83. 디렉토리를 만드는 명령어는 mkdir 명령어다.

man -s2 mkdir 을 활용하여 mkdir System Call 을 볼 수 있다.

이를 참고하여 디렉토리를 만드는 프로그램을 작성해보자!

```
#include <sys/stat.h>
```

```
#include <sys/types.h>
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

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

```
{
```

```
    if(argc != 2)
```

```
    {
```

```
        printf("Usage: exe dir_name\n");
```

```
        exit(-1);
```

```
    }
```

```
    mkdir(argv[1], 0755);
```

```
    return 0;
```

```
}
```

```
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test$ ls
83.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test$ gcc 83.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test$ ./a.out test
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test$ ls
83.c  a.out  test
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test$
```

84. 이번에는 랜덤한 이름(길어도 랜덤)을 가지도록 디렉토리를 3 개 만들어보자!
(너무 길면 힘들니까 적당한 크기로 잡도록함)

```
#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

int main(void){

    int i,j;
    int len;
    char buf[8] = "\0";
    char *dname[3] = {0};

    srand(time(NULL));

    for(i=0; i<3; i++){
        //랜덤 길이
        len = rand() % 5 +2;
        printf("len = %d\n",len);

        //랜덤 이름

        for(j=0; j<len; j++){

            buf[j] = rand() % 26 + 97;

        }
        printf("buf = %s\n",buf);

        dname[i] = (char *)malloc(sizeof(char) * 8);
        strcpy(dname[i], buf);
        mkdir(dname[i], 0755);

    }

    return 0;
}
```

```
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test2$ ls
84.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test2$ gcc -g 84.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test2$ ./a.out
len = 3
buf = sbi
len = 5
buf = zbcuw
len = 5
buf = ptxln
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test2$ ls
84.c  a.out  ptxln  sbi  zbcuw
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test2$
```

85. 랜덤한 이름을 가지도록 디렉토리 3 개를 만들고

각각의 디렉토리에 5 ~ 10 개 사이의 랜덤한 이름(길어도 랜덤)을 가지도록 파일을 만들어보자!

(너무 길면 힘드니까 적당한 크기로 잡도록함)

```
#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>

void make_file(char **dname){

    int i,j,z;
    int len,num,fd;
    char buf[9] = "\0";
    char *fname[] = {0};

    srand(time(NULL));

    for(i=0; i<3; i++){
        chdir(dname[i]);

        num = rand() % 6 +5;
        printf("폴더 안에 파일을 몇개만들지? %d\n",num);

        for(z=0; z<num; z++){

            //랜덤 길이
            len = rand() % 5 +2;

            //랜덤 이름
            for(j=0; j<len; j++){

                buf[j] = rand() % 26 + 97;

            }

            fd = open(buf, O_CREAT, 0644);
            close(fd);

        }
        chdir("..");
    }
}

int main(void){

    int i,j;
    int len;
    char buf[8] = "\0";
    char *dname[3] = {0};

    srand(time(NULL));

    for(i=0; i<3; i++){
```

```

//랜덤 길이
len = rand() % 5 + 2;

//랜덤 이름
for(j=0; j<len; j++){

    buf[j] = rand() % 26 + 97;

}

printf("파일 이름은? %s\n",buf);

dname[i] = (char *)malloc(sizeof(char) * 8);
strcpy(dname[i], buf);
mkdir(dname[i], 0755);

}

make_file(dname);

return 0;
}

```

```

mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ls
85.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ gcc 85.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ./a.out
파일 이름? afqzxi
파일 이름? nmalyc
파일 이름? vssbyc
폴더 안에 파일을 몇개 만들지? 5
폴더 안에 파일을 몇개 만들지? 6
폴더 안에 파일을 몇개 만들지? 6
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ls
85.c afqzxi a.out nmalyc vssbyc
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ls afqzxi/
busyiv cvsyiv fqzxiv isdyiv malyiv
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ls nmalyc/
izowyv phrxqv syowyv vmwiqv wtyayv zvbqv
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$ ls vssbyc/
ctmljw penojv qraojv rsdudw zidudw zzkkzi
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test3$

```

86. 85 번까지 진행된 상태에서 모든 디렉토리를 순회하며

3 개의 디렉토리와 그 안의 모든 파일들의 이름 중 a, b, c 가 1 개라도 들어있다면 이들을 출력하라!

출력할 때 디렉토리인지 파일인지 여부를 판별하도록 하시오.

```
#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <dirent.h>

void make_file(char **dname){

    int i,j,z;
    int len,num,fd;
    char buf[9] = "\0";
    char *fname[] = {0};

    srand(time(NULL));

    for(i=0; i<3; i++){
        chdir(dname[i]);

        num = rand() % 6 +5;
        printf("폴더 안에 파일을 몇개만들지? %d\n",num);

        for(z=0; z<num; z++){

            //랜덤 길이
            len = rand() % 5 +2;

            //랜덤 이름
            for(j=0; j<len; j++){

                buf[j] = rand() % 26 + 97;

            }

            fd = open(buf, O_CREAT, 0644);
            close(fd);

        }
        chdir("../");
    }
}

void find_file(char **dname){

    struct dirent *p;
    DIR *dp;
    int i,j,len;

    for(j=0; j<3; j++){
```

```

        chdir(dname[j]);
        dp = opendir(".");

        printf("오픈한 폴더 이름? %s\n", dname[j]);

        while(p = readdir(dp)){

            len = strlen(p->d_name);

            for(i=0; i<len; i++){

                if(!strcmp(&p->d_name[i], "a", 1) ||
                    !strcmp(&p->d_name[i], "b", 1) ||
                    !strcmp(&p->d_name[i], "c", 1))
                {
                    printf("a,b,c 가 한개라도 들어간 파일? %s\n", p->d_name);
                }
            }

            closedir(dp);
            chdir("..");
        }
    }

int main(void){

    int i,j;
    int len;
    char buf[8] = "\0";
    char *dname[3] = {0};

    srand(time(NULL));

    for(i=0; i<3; i++){
        //랜덤 길이
        len = rand() % 5 +2;

        //랜덤 이름
        for(j=0; j<len; j++){

            buf[j] = rand() % 26 + 97;

        }

        printf("폴더 이름은? %s\n",buf);

        dname[i] = (char *)malloc(sizeof(char) * 8);
        strcpy(dname[i], buf);
        mkdir(dname[i], 0755);

    }

    make_file(dname);
    find_file(dname);

    return 0;
}

```


}

```
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test4$ ls
86.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test4$ gcc 86.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test4$ ./a.out
폴더 이름? zefn
폴더 이름? eanoqu
폴더 이름? jishwp
폴더 안에 파일 몇개만들지? 10
폴더 안에 파일 몇개만들지? 10
폴더 안에 파일 몇개만들지? 7
открытый папка имя? zefn
a,b,c가 한개라도 어간 파일? bllzaf
a,b,c가 한개라도 어간 파일? bllzaf
a,b,c가 한개라도 어간 파일? eanoqu
a,b,c가 한개라도 어간 파일? hxbwwp
a,b,c가 한개라도 어간 파일? sqaqmf
a,b,c가 한개라도 어간 파일? bllaqmf
a,b,c가 한개라도 어간 파일? bllaqmf
a,b,c가 한개라도 어간 파일? ltbwwp
a,b,c가 한개라도 어간 파일? woaqmf
открытый папка имя? eanoqu
a,b,c가 한개라도 어간 파일? ablzld
a,b,c가 한개라도 어간 파일? ablzld
a,b,c가 한개라도 어간 파일? cuaghi
a,b,c가 한개라도 어간 파일? cuaghi
a,b,c가 한개라도 어간 파일? ojbmgf
a,b,c가 한개라도 어간 파일? papzld
a,b,c가 한개라도 어간 파일? chbuoh
a,b,c가 한개라도 어간 파일? chbuoh
a,b,c가 한개라도 어간 파일? vkhbth
открытый папка имя? jishwp
a,b,c가 한개라도 어간 파일? zkayzk
a,b,c가 한개라도 어간 파일? fbrizk
a,b,c가 한개라도 어간 파일? buvzzk
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test4$ ls
86.c a.out eanoqu jishwp zefn
mhn@mhn-Z20NH-AS51B5U:~/linux/note/test/test4$
```

87. 클라우드 기술의 핵심인 OS 가상화 기술에 대한 질문이다.
OS 가상화에서 핵심에 해당하는 3 가지를 기술하시오.

CPU 가상화, 메모리 가상화, I/O 가상화

90. 89 번조차도 공격할 수 있는 프로그램을 작성하시오.

<at_clnt.c>

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/epoll.h>
```

```
#define BUF_SIZE 128
#define NAME_SIZE 32
```

```
typedef struct sockaddr_in si;
typedef struct sockaddr *sp;
```

```

char name[NAME_SIZE] = "DEFAULT";
char msg[2048];

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

void *send_msg(void *arg)
{
    int sock = *((int *)arg);
    char name_msg[NAME_SIZE + BUF_SIZE];
    char msg2[] = "zzz 지금 당장 접속하세요!!!\n";
    char tmp1[32] = {0};

    for(;;)
    {
        sprintf(msg, "%s %s %s", name, tmp1, msg2);
        write(sock, msg2, strlen(msg2));
        sleep(1);

        /* if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n"))
        {
            close(sock);
            exit(0);
        }

        sprintf(name_msg, "%s %s", name, msg);
        write(sock, name_msg, strlen(name_msg));*/
    }
    return NULL;
}

void *recv_msg(void *arg)
{
    int sock = *((int *)arg);
    char name_msg[NAME_SIZE + 2048];
    int str_len;

    for(;;)
    {
        str_len = read(sock, name_msg, NAME_SIZE + 2047);

        name_msg[str_len]=0;
        fputs(name_msg, stdout);
    }
    return NULL;
}

int main(int argc, char **argv)
{
    int sock;
    si serv_addr;
    pthread_t snd_thread, rcv_thread;
    void *thread_ret;

```


91. 네트워크 상에서 구조체를 전달할 수 있게 프로그래밍 하시오.

<ch_serv.c>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/epoll.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);

}

struct __h{

int a;
float b;

};
typedef struct __h H;

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

    int serv_sock,len;
    int clnt_sock;

    si serv_addr;
    si clnt_addr;
    socklen_t clnt_addr_size;

    H HH;

    HH.a = 3;
    HH.b = 3.14;

    char msg[] = "Hello Network Programming";

    if(argc !=2){

        printf("use : %s <port>\n",argv[0]);
        exit(1);
    }

    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]));

if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
    err_handler("bind() error");

if(listen(serv_sock,5)==-1)
    err_handler("listen() error");

clnt_addr_size = sizeof(clnt_addr);
clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr,&clnt_addr_size);
if(clnt_sock == -1)
    err_handler("accept() error");

write(clnt_sock,&HH,sizeof(H));

close(clnt_sock);
close(serv_sock);

return 0;
}

```

<ch_clnt.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);

}

struct __h{

int a;
float b;

};

typedef struct __h H;

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

    int sock;
    int str_len;

```

```

si serv_addr;
char msg[32];
char buff[32];
H buf;
int len;

if(argc != 3){

    printf("use: %s <ip> <port>\n",argv[0]);
    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");

if(str_len == -1)
    err_handler("read() error!");

len = read(sock,&buf,sizeof(H));
printf("len = %d\n",len);

printf("buf.a->%d\n",buf.a);
printf("buf.b->%lf\n",buf.b);

close(sock);

return 0;

}

```

```

mhn@mhn-Z20NH-AS51B5U:~/linux/note$ gcc -o ch_serv ch_serv.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note$ ./ch_serv 7777
mhn@mhn-Z20NH-AS51B5U:~/linux/note$

```

```

mhn@mhn-Z20NH-AS51B5U:~/linux/note$ gcc -o ch_clnt ch_clnt.c
mhn@mhn-Z20NH-AS51B5U:~/linux/note$ ./ch_clnt 127.0.0.1 7777
len = 8
buf.a->3
buf.b->3.140000
mhn@mhn-Z20NH-AS51B5U:~/linux/note$

```