**TCP/IP 연습문제 < 학년: 3 반: C 학번: 202244099 이름: 이선용>**

1. 다음은 인터넷 주소를 초기화하는 template이다. 빈칸을 채우시오.

struct sockaddr\_in addr;

char \*serv\_ip="211.217.168.13";

char \*serv\_port="9190";

memset( &serv\_adr, 0, sizeof(serv\_adr) );

addr.sin\_family = ( AF\_INET );

addr.sin\_addr.s\_addr = ( inet\_addr )(serv\_ip);

addr.sin\_port = ( hthons )(atoi(serv\_port));

2. ICMP의 용도를 간략히 쓰시오.

* 호스트 또는 라우터 사이에 오류 정보나 제어 정보를 전

달하는데 사용

* Ping 같은 응용 프로그램이 직접 사용하기도 함

3. ARP는 ( IP ) 주소를 알고 있을 때 ( MAC ) 주소를 알기 위해 사용하며, RARP는 ( MAC ) 주소를 알고 있을 때 ( IP ) 주소를 알기 위해 사용한다.

<-- 괄호 안에 적절한 단어를 쓰시오

4. DNS의 순환 (recursive) 쿼리 방식을 그림을 이용하여 설명하시오.

5. DHCP는 동적으로 IP를 할당한다. 이 때 보통 ( 사설 ) IP 주소를 할당한다. 외부와 통신 하기 위해서는 ( 공인 ) IP 주소가 필요한데 이 두 주소간의 mapping 관계를 저장하고 변환하는 것을 ( NAT )라고 한다.

<-- 괄호 안에 적절한 단어와 숫자를 쓰시오

6. 서버에서, 현재 시스템의 IP주소를 자동적으로 찾아서 할당해주려 한다. 다음의 괄호 안을 채우시오. (네트워크 바이트 순서까지 고려해야 함)

struct sockaddr\_in addr;

addr.sin\_addr.s\_addr = ( inet\_addr(serv\_ip) );

7. 다음은 bind 함수를 사용하려 한다. 괄호 안을 채우시오.

struct sockaddr\_in serv\_addr;

bind(serv\_sock, ( struct sockaddr\* ) &serv\_addr, sizeof(serv\_addr));

8. 다음의 괄호 안을 채우시오.

연결 요청 대기 큐를 생성하고 연결 요청을 기다리는 함수는 ( listen ) 이며 연결 요청을 수락하는 함수는 ( accept ) 이다. 클라이언트 쪽에서 연결을 요청하는 함수는 ( connect )이다.

9. 서버 소켓, 연결 요청 대기 큐와 새로운 클라이언트 소켓이 생성되는 관계를 그림을 이용하여 보이시오.

10. 다음은 TCP 에코 서버의 일부분이다. 괄호 안을 채우시오.

while( (str\_len=read(clnt\_sock, message, BUFSIZE)) != ( 0 )) {

write( ( clnt\_sock ), message, str\_len); /\* client로 전송 \*/

}

11. TCP는 한번에 write()를 사용하여 “ABCD” 문자를 전송할지라도, 그 데이타들이 하나의 패킷을 형성하여 전송 되는 것은 아니다. 이를 ( 데이터에 경계 )가 없다라고 표현한다. 이러한 현상은 TCP의 ( 흐름 ) 제어 때문에 발생한다.

<-- 괄호 안을 채우시오.

12. 데이터의 경계 문제로 인하여 에코 클라이언트에서 전송한 내용이 한 번에 수신되지 않을 수 있는 문제가 발생할 수 있다. 이를 해결하기 위해 다음의 코드를 수정하시오. (새로운 변수가 필요할 경우 미리 선언되어 있다고 가정해도 됨)

write(sock, message, strlen(message));

str\_len=read(sock, message, BUFSIZE-1);

while(1){

fputs("Insert message(q to quit): ",stdout);

fgets(message,sizeof(message),stdin);

if(!strcmp(message,"q\n")||!strcmp(message,"Q\n"))

break;

write(sock, message, strlen(message));

str\_len=read(sock,message,BUF\_SIZE);

message[str\_len]=0;

printf("Message from server: %s",message);

}

13. 도메인 네임을 알고 있고 IP 주소를 알고 싶은 경우 ( gethostbyname ) 함수를 사용하고, 반대로 IP 주소를 알고 있고 도메인 네임을 알고 싶은 경우는 ( gethostbyaddr ) 함수를 사용한다. <-- 괄호안에 적절한 함수명을 적으시오.

14. TCP기반의 에코 서버 / 클라이언트 함수 호출 관계를 그림을 활용하여 보이시오. (소켓 생성에서 연결 종료의 전 과정을 보이시오)

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

15. UDP기반의 에코 서버 / 클라이언트 함수 호출 관계를 그림을 활용하여 보이시오. (소켓 생성에서 연결 종료의 전 과정을 보이시오)

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

16. 다음은 UDP 에코 클라이언트의 일부분이다. 괄호 안을 채우시오.

sendto(sock, message, strlen(message),

0, ( struct sockaddr\* )&serv\_addr , ( sizeof(serv\_addr) ));

adr\_sz=sizeof(from\_addr);

str\_len = recvfrom(sock, message, BUFSIZE,

0, ( struct sockaddr\* )&from\_addr, ( &adr\_sz ) );

17. TCP 에코 서버 소스를 UDP 에코 서버 소스로 바꾸려 한다. 다음의 필요한 변경 과정을 쓰시오.

a. 변경: SOCK\_STREAM --> SOCK\_DGRAM

b. 삭제해야 될 function: listen, accept

c. 변경해야 될 function: read --> recvfrom

write --> sendto

18. UDP 방식에서 connect()를 사용할 때 얻는 이점을 구체적으로 쓰시오.

- 커널에서 소켓의 연결을 유지

- connect() 호출시 연결생성

19. Half Close에 대해 간략히 설명하시오. (또한, 이때 사용되는 함수명도 쓰시오)

Int shutdown(int s, int how)

- 소켓 스트림의 일부만을 종료(half close)

=> 전송은 가능, 수신은 불가한 상황, 또는 수신은 가능, 전송은 불가능

=> Half close : 입력 및 출력 스트림 중 하나의 스트림만 종료하는 행위

=> 스트림의 반만 닫음

20. 다음은 파일 전송 서버의 일부분이다. 데이터 전송 후, 소켓의 출력 스트림만을 닫으려 한다. 괄호 안을 채우시오.

/\* 데이터 전송 후, 소켓의 출력 스트림만을 닫음 \*/

if( shutdown( clnt\_sd, ( SHUT\_WR ) == -1 )

error\_handling("error");

21. 다음은 TCP 파일 전송 클라이언트의 일부분이다. 괄호 안을 채우시오.

#define BUFSIZE 30

int sd;

FILE \*fp;

char buf[BUF\_SIZE];

int read\_cnt;

fp = fopen("receive.dat", "wb" );

sd = socket(PF\_INET, SOCK\_STREAM, 0 );

/\* 데이터를 전송 받아서 파일에 저장한다 \*/

while( (read\_cnt=read(sd, buf, BUF\_SIZE ))!=0 )

{

( fwrite((void\*)buf, 1, read\_cnt, fp); )

}

22. 리눅스 기반의 다중 접속 서버 구현을 할 수 있는 세가지 방법을 쓰시오.

멀티태스킹

멀리플렉싱

멀티쓰레딩

23. 다음은 fork 함수를 이용한 프로그램의 수행 결과이다. 결과의 괄호 안을 채우시오.

pid=fork();

if(pid==-1)

printf("fork 실패, 프로세스 id : %d \n", pid);

printf("fork 성공, 프로세스 id : %d \n", pid);

수행결과

fork 성공, 프로세스 ID: 1403

fork 성공, 프로세스 ID: ( 0 )

24. fork() 함수를 통해 child process를 생성하였다. parent process와 child process를 구분할 수 있는 방법을 설명하시오.

fork() 함수는 호출한 프로세스를 복사하여 새로운 프로세스를 만든다. 이때, fork() 함수는 두 번 반환되는데, 한 번은 부모 프로세스에서, 한 번은 자식 프로세스에서 반환된다. 이 반환값을 이용해 두 프로세스를 구분할 수 있다.

25. 좀비 프로세스의 정의와 생성 이유를 쓰시오.

1. 좀비 프로세스란?

- 프로세스 종료 후 메모리상에서 사라지지 않는 프로세스

- 시스템의 리소스를 점유하여 성능을 저하시킴

2. 좀비 프로세스의 생성 이유.

- 커널은 비록 자식 프로세스가 종료되었더라도 리턴값을 부모 프로세스에

넘겨줄 때까지 자식 프로세스를 소멸시키지 않음

- 자식 프로세스는 부모 프로세스에게 실행 결과에 대한 값을 반환해야 한다.