시스템 해킹 - C언어로 http 구현

<Header File>

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

▼ **stdio.h**

stdio.h은 Standard Input/Output library (표준입출력 라이브러리)의 약어로써, C 언어의 표준 라이브러리 함수의 매크로 정의, 상수, 여러 형의 입출력 함수가 포함된 헤더 파일이다.

▼ **stdlib.h**

<stdlib.h>은 C 표준 유틸리티 함수들을 모아놓은 헤더파일이다.

이 헤더파일에는 프로그래밍 시에 범용적으로 사용되는 여러가지 함수들을 모아 놓고 있는데,

예를 들면 동적 할당 관련 함수, 난수 생성 함수, 정수의 연산 함수, 검색 및 정렬 함수 등 이다.

▼ **unistd.h**

POSIX 운영체제 API에 대한 액세스를 제공하는 헤더파일이다. (read, write)

▼ **sys/types.h**

c언어의 버전이 업 되면서 변경된 자료형의 정의들과

시스템마다 int를 표현하는 바이트수가 다르므로 그런것들을 다른 자료형으로 정의하는

typedef문들이 들어있다.

(int가 어떤곳에선 2바이트 , 요즘에는 거의다 4바이트이기 때문에

이 두개의 자료형을 구분하는 어떤 타입정의가 필요함)

▼ **sys/socket.h**

socket (), socketpair (), bind (), getsockname (), connect (), getpeername ()

send (), recv (), sendto (), recvfrom (), sendmsg (), rcvmsg ()

getsockopt (), setsockopt (), listen (), accept (), shutdown () 등등 소켓 관련 함수들이 들어가있다.

▼ **netinet/in.h**

AF_INET 및 AF_INET6 주소 패밀리. 인터넷에서 널리 사용되는 IP 주소와 TCP / UDP 포트 번호가 포함된다.

▼ **netdb.h**

프로토콜 및 호스트 이름을 숫자 주소로 변환하는 기능을 정의합니다. 로컬 데이터와 DNS를 검색한다.

▼ **arpa/inet.h**

숫자로 IP 주소를 조작하는 기능의 정의

▼ **string.h**

이 헤더파일에는 C 형식 문자열 (널 종료 문자열) 을 다룰 수 있는 함수들을 포함하고 있다.

(memset, strcpy 등)

▼ **sys/stat.h**

파일의 크기, 파일의 권한, 파일의 생성일시, 파일의 최종 변경일 등, 파일의 상태나 파일의 정보를 얻는 함수를 포함하고 있습니다.

▼ **fcntl.h**

리눅스 시스템에서는 열려진 파일의 속성을 가져오거나 설정을 하기 위해 사용

<HTML FILE>

```
char webpage[] =
"HTTP/1.1 200 OK\r\n"
"Contect-Type: text/html; charset=UTF-8\r\n\r\n"
"<!DOCTYPE html>\r\n"
"<html><head><title>P4C 4th HW</title>\r\n"
"<style>body { background-color: #FFF00 }</style><head>\r\n"
"<body><center><h1>Hello! This is P4C HW! I'm recarrdo</h1><br>\r\n</center></body></html>";
```

<Main() File>

```
int main(int argc, char* argv[])
{
    struct sockaddr_in server_addr, client_addr;
    socklen_t sin_len = sizeof(client_addr);
    int fd_server, fd_client;
    char buf[2048];
    int fdimg;
    int on = 1;

    fd_server = socket(AF_INET, SOCK_STREAM, 0);
    if (fd_server < 0)
    {
        perror("socket");
        exit(1);
    }

    setsockopt(fd_server, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(int));

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(8080);

    if (bind(fd_server, (struct sockaddr*)&server_addr, sizeof(server_addr)) == -1)
    {
        perror("bind");
        close(fd_server);
        exit(1);
    }
```

```
    if (listen(fd_server, 10) == -1)
    {
        perror("listen");
        close(fd_server);
        exit(1);
    }

    while (1)
    {
        fd_client = accept(fd_server, (struct sockaddr*)&client_addr, &sin_len);

        if (fd_client == -1)
        {
            perror("Connection failed.....\n");
            continue;
        }

        printf("Got client connection...\n");

        if (!fork())
        {
            close(fd_server);
            memset(buf, 0, 2048);
            read(fd_client, buf, 2047);

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

            write(fd_client, webpage, sizeof(webpage) - 1);

            close(fd_client);
            printf("closing....\n");
            exit(0);
        }
        close(fd_client);
    }
    return 0;
}
```

▼ **socket()**

용도: 소켓을 생성하고 반환하는 함수입니다.

형태: int socket(int domain, int type, int protocol)

인수:

int domain: 인터넷을 통해 통신할 지, 같은 시스템 내에서 프로세스 끼리 통신할 지의 여부를 설정합니다.

int type: 데이터의 전송 형태 지정

int protocol: 통신에 있어 특정 프로토콜을 사용을 지정하기 위한 변수이며, 보통 0 값을 사용합니다.

▼ **setsockopt()**

용도: 소켓옵션값을 변경하기 위해서 사용한다.

형태: setsockopt(int s, int level, int optname, const void *optval, socklen_t optlen);

인수:

s : 소켓지정번호

level : 소켓의 레벨로 어떤 레벨의 소켓정보를 가져오거나 변경할 것인지를 명시한다. 보통 SOL_SOCKET와 IPPROTO_TCP 중 하나를 사용한다.

optname : 설정을 위한 소켓옵션의 번호

optval : 설정값을 저장하기 위한 버퍼의 포인터

optlen : optval 버퍼의 크기

▼ **AF_INET**

주소 체계(주소 패밀리)중 하나이다.

주소 체계를 설정하는 부분은 AF로 시작하는 상수를 사용하는 것이 좋다.

▼ **SOCK_STREAM**

데이터 전송 타입 중 하나로, 연결 지향형 소켓이다.
아래와 같은 3가지 특징이 있다.

-에러나 데이터의 손실 없이 무사히 전달.
-전송하는 순서대로 데이터가 전달
-전송되는 데이터의 경계가 존재하지 않음.

정리하자면, 신뢰성 있는 순차적인 바이트 기반의 연결 지향 전송 타입이다.

▼ **perror()**

에러메세지를 출력하는 함수이다.

▼ **bind()**

용도: bind()함수를 이용해 소켓에 포트 번호를 부여한다.
형태: int bind(int socket, struct sockaddr * localAddress, unsigned int addressLength**)**
인수:
socket은 이전의 socket()호출에서 반환된 식별자입니다.
한마디로 서버 소켓으로 사용할 소켓을 적어주면 된다.
localAddress는 주소 파라미터로 sockaddr의 포인터로 선언되는데
TCP/IP 어플리케이션에서는 자신의 해당 인터페이스 및 요구를 기다리는 포트 번호를 포함하는
sockaddr_in의 구조체를 가리키면 됩니다. 그럼 위에서 sockaddr_in구조체로 선언했던 변수를 적어주면 된다.
addressLength는 주소 구조체의 길이로 sizeof(struct sockaddr_in)로 써주면 된다.
bind()함수는 성공하면 0 실패하면 -1을 반환해준다.

▼ **listen()**

용도: listen()함수를 이용해 클라이언트의 접속을 기다린다.
형태: listen(int socket, int queueLimit)
인수:
socket은 사용할 소켓 즉, 식별자를 쓰면 된다.
queueLimit는 연결 요구 개수의 최대값을 지정하면 된다.
listen()함수는 주어진 소켓에 대해 내부 상태의 변경을 유발하여 들어오는 tcp연결 요구들이
프로그램에 의해 처리되고 받아들여질 수 있도록 큐에 저장되는 것을 가능케 한다.
성공시 0 실패시 -1을 반환 시켜준다.

주의 listen()함수는 대기만 하는것이지 연결을 해주는 함수는 아니다.

▼ **accept()**

용도: 클라이언트로부터 접속이 되면 accept()함수를 호출해 클라이언트 연결에 대한 새로운 소켓을 생성한다.
형태: accept(int socket, struct sockaddr * clientAddress, unsigned int * addressLength)
인수:
accept()함수는 socket을 위한 큐에서 다음 연결을 하나 꺼냅니다. 만약 큐가 비어있으면 accept()는 연결 요구가 도착할 때까지 블록된다.
성공을 하면 accept()는 clientAddress가 가리키는 sockaddr 구조체를 연결의 다른 종단점에 있는 클라이언트의 주소로 채웁니다.
그리고 그 소켓을 이용해 통신을 하는것이다.
addressLength는 clientaddress주소 구조체의 최대 크기를 지정하는데 반환시 실제로 주소를 위해 사용된 바이트 수를 가지게 된다.

성공하면 해당 클라이언트와 연결된 새로운 소켓의 식별자를 반환한다.

참고
accept()함수의 첫 번째 파라미터로 보내진 소켓은 변하지 않습니다.
다시 말해 클라이언트와 연결되지 않는다는 뜻!
그리고 계속해서 새로 들어오는 연결 요구를 듣습니다. 실패 시 -1을 반환합니다.

▼ **close()**

close()함수를 이용해 연결을 닫는다.

▼ **fork()**

프로세스를 생성하고자 할 때 fork 함수를 사용하면 된다.

▼ **memset()**

용도: memset 함수는 메모리의 내용(값)을 원하는 크기만큼 특정 값으로 세팅할 수 있는 함수 이다.
형태: void* memset(void* ptr, int value, size_t num);
인수:
void* ptr은 세팅하고자 하는 메모리의 시작 주소.
즉, 그 주소를 가리키고 있는 포인터가 위치하는 자리 입니다.
value는 메모리에 세팅하고자 하는 값을 집어 넣으면 됩니다.
int 타입으로 받지만 내부에서는 unsigned char 로 변환되어서 저장됩니다.
즉 'a' 이런것을 넣어도 무방하다는 뜻입니다.
size_t num은 길이를 뜻합니다. 이 길이는 바이트 단위로써 메모리의 크기 한조각 단위의 길이를 말합니다. 이는 보통 "길이 * sizeof(데이터타입)" 의 형태로 작성하면 됩니다.
반환값은 성공하면 첫번째 인자로 들어간 ptr을 반환하고, 실패한다면 NULL을 반환합니다

▼ **read()**

용도: 파일을 읽는데 사용한다.
형태: read (int fd, void *buf, size_t len)
인수:
open(): 시스템 콜로 열린 파일을 가리키는 파일 지정 **번호**
void* buf: 파일에서 읽은 데이터를 저장할 메모리 공간
len: 읽을 데이터의 크기(Byte 단위)

▼ **write()**

용도: 파일에 데이터를 작성하기 위해 사용한다.
형태: write (int fd, void *buf, size_t len)
인수:
int fd open() 시스템 콜로 열린 파일을 가리키는 파일 지정 번호

void* buf 파일에 쓸 데이터를 저장하고 있는 메모리 공간
len 파일에 쓸 데이터의 길이