

Chapter 08

도메인 이름과 인터넷 주소

ex) ✓ google.com
✓ www.com

도메인 이름(Domain Name)

kn0.ac.kr

■ 도메인 이름이란

org

- 영문으로 표현되는 계층적 주소 체계 방식
- 각 나라마다 존재하는 Network Information Center에서 관리
- 한국은 KRNIC(Korea Network Information Center: <http://www.nic.or.kr/>) 담당
- IP 주소를 대신해서 사용하는 서버의 주소
 - ex) www.naver.com
- 실제 접속에 사용되는 주소는 아니며, 도메인 정보는 IP 주소로 변환이 되어야 접속이 가능함

■ Domain Name System (DNS)

- IP주소와 도메인 이름 사이에서 변환을 수행하는 시스템
- 경북대학교 DNS 서버: 155.230.10.2

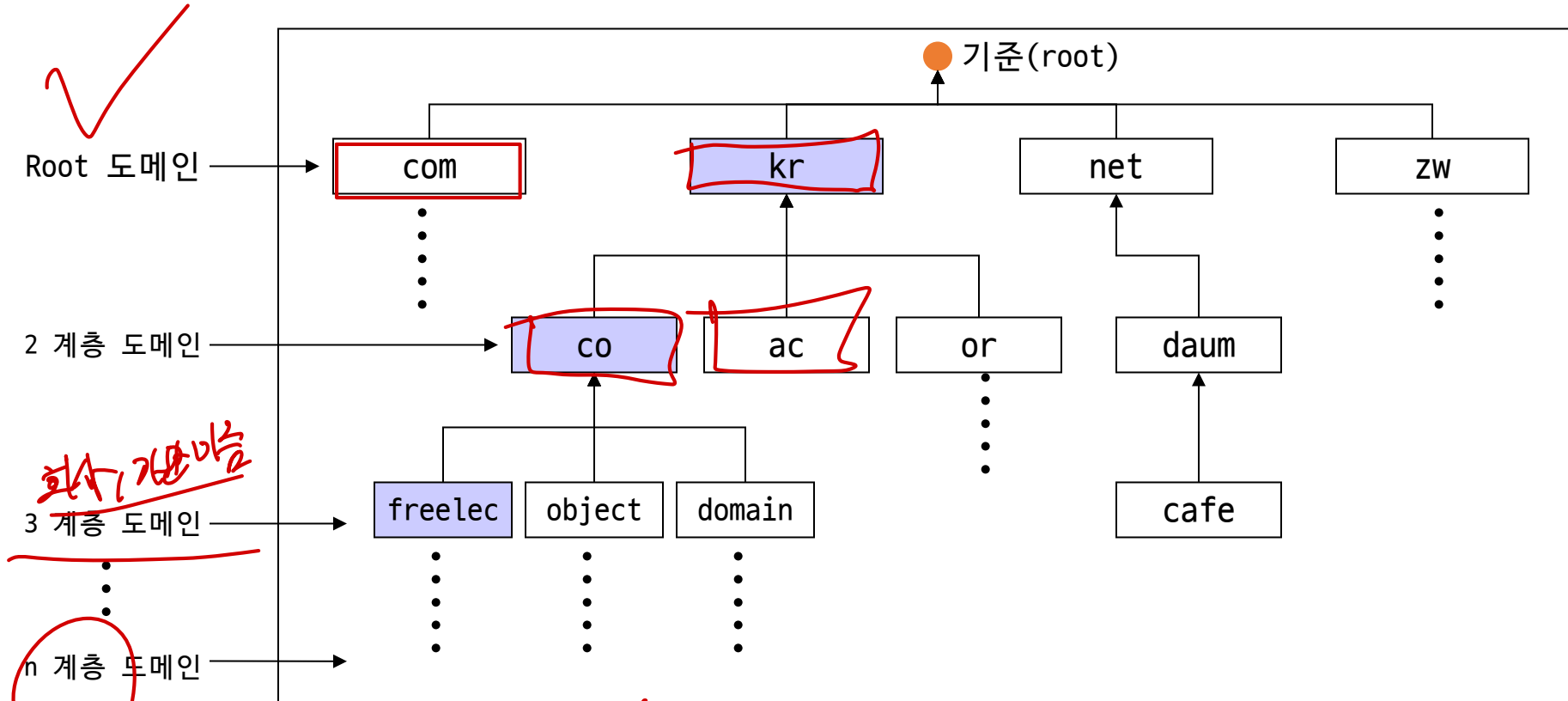
- DNS(Domain Name System) 서버
 - 도메인 이름을 IP 주소로 변환해주는 서버
 - www.naver.com -> 해당 IP 주소로 변환
 - DNS는 일종의 분산 데이터베이스 시스템 ✓

- 호스트는 Default DNS Server에게 도메인 이름의 IP 주소를 요청
- Default DNS Server는 상위 계층의 DNS Server에 요청
- 다시 Root DNS Server에게 요청

-
- The diagram illustrates the DNS hierarchy and its interaction with local networks. At the top is the **Root DNS Server**. Below it are two **DNS Server** units. These servers are connected to two separate **Local Area Network** boxes. The left Local Area Network contains a **Default DNS Server** and a laptop (circled in red). The right Local Area Network contains a **Default DNS Server** and a **Destination Server**. Red arrows indicate the flow of DNS queries from the Root Server to the DNS Servers, and from the Default DNS Servers to the Root Server. A blue arrow points from the laptop to the Default DNS Server in the left network. Handwritten red text "know DNS stuff" is present near the bottom center.

인터넷상에서의 주소 체계

knv.ac.kr

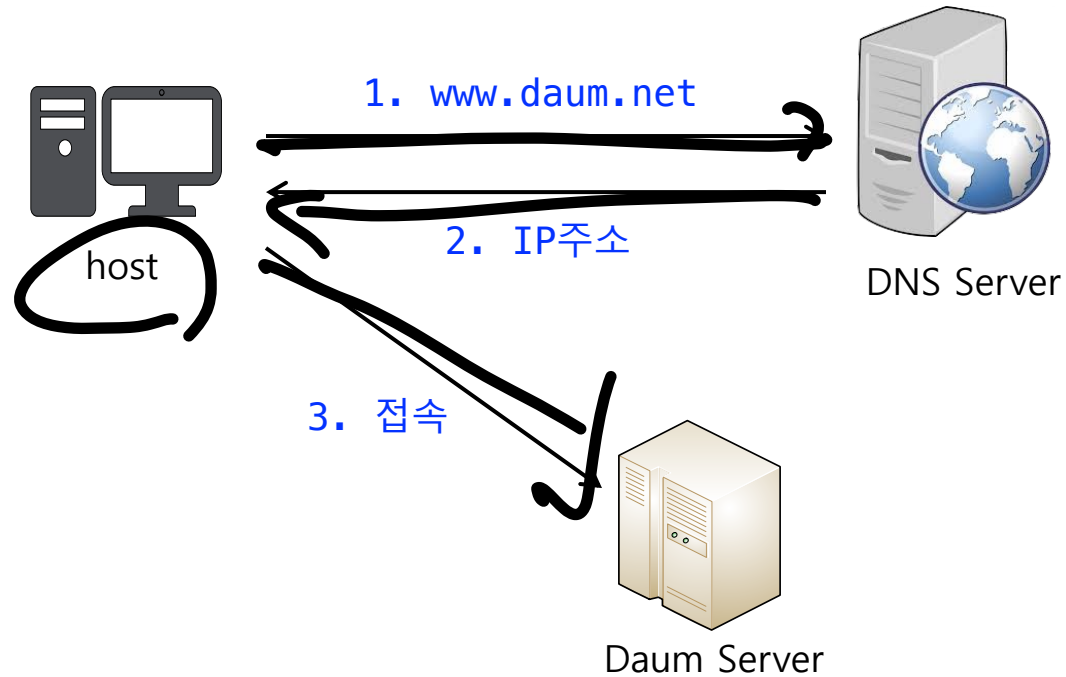


Computer.knv.ac.kr

package com.knv 프로젝트명

DNS (Domain Name System) 서버

- 모든 Domain 이름은 해당 IP와 함께 DNS 서버에 등록되어야 함
- Domain 이름을 IP 주소로 변환해 주는 작업을 수행함



ping & nslookup

■ ping 명령어

- 도메인 이름에 해당하는 IP주소를 확인하고 싶을 때 사용
- ex) ping www.naver.com

```
$ ping www.knu.ac.kr
PING www.knu.ac.kr (155.230.11.1) 56(84) bytes of data.
64 bytes from www.knu.ac.kr (155.230.11.1): icmp_seq=1 ttl=251 time=1.00 ms
64 bytes from www.knu.ac.kr (155.230.11.1): icmp_seq=2 ttl=251 time=0.542 ms
64 bytes from www.knu.ac.kr (155.230.11.1): icmp_seq=3 ttl=251 time=0.567 ms
```

1.5초 다 왔다 왔다고 표시

응답 시간 줄이기

→ 응답시간.

■ nslookup 명령어

- 디폴트 DNS 서버의 주소를 알고 싶을 때 사용: nslookup server
- /etc/systemd/resolved.conf 에 설정된 내용
 - 예전 Linux 버전: /etc/resolv.conf

```
$ nslookup server
Server:      155.230.10.2
Address:     155.230.10.2#53

** server can't find server: NXDOMAIN
```

```
$ nslookup google.com
Server: 155.230.10.2
Address: 155.230.10.2#53

Non-authoritative answer:
Name: google.com
Address: 172.217.161.238
```

→ 설정 관련된 이슈가 많이 생김.

리눅스 연습 할려면
두알블링 V
응답 시간 줄이기!!

도메인 이름을 이용한 IP주소 얻어오기

- `gethostbyname()` 함수
 - 문자열 형태의 도메인 이름으로 IP주소 정보를 얻음

```
#include <netdb.h>
```

```
struct hostent* gethostbyname(const char *hostname);
```

-> 성공 시 hostent 구조체 변수의 주소 값, 실패 시 NULL 포인터 반환

"www.nate.com"



➤ hostname에 문자열 도메인 이름 전달

- 도메인 이름을 사용하는 이유
 - IP 주소는 도메인 이름에 비해 변동이 심함

이 때문에 바꿀때

- 프로그램 코드에서 서버의 IP 주소를 직접 입력하는 경우, *이런 소스*

- 서버의 IP 주소가 변경될 때마다 컴파일을 다시 해야 됨

웹브라우저 업데이트하면

- 상대적으로 변동이 덜한 도메인 이름 사용

- 서버에 대한 IP 주소를 얻어오게 구현하면, 코드 재 컴파일이 필요 없음

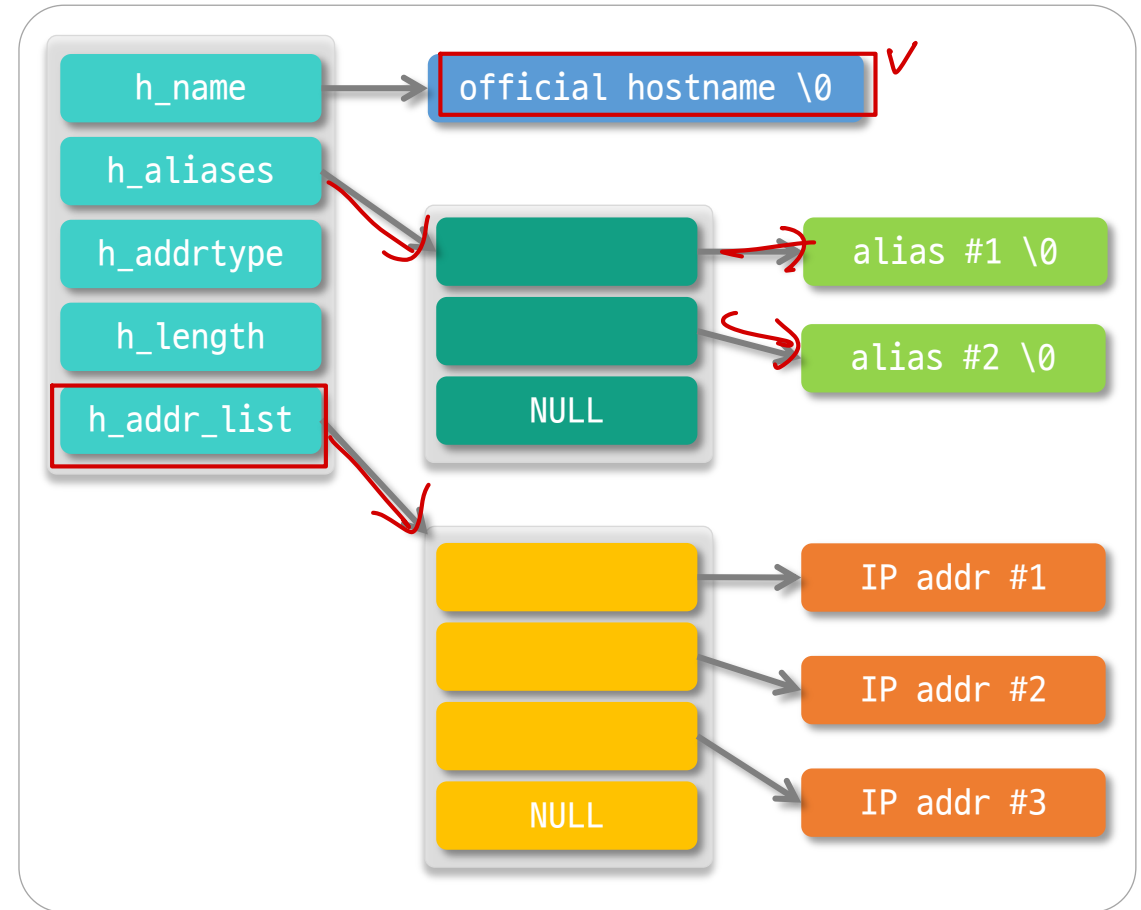
```
struct hostent
{
    char *h_name;           // official name
    char **h_aliases;       // alias list
    int h_addrtype;         // Host address type
    int h_length;           // address length
    char **h_addr_list;     // address list
}
```

24시간 이상 걸림...!!

구조체 hostent에 채워지는 정보의 형태

- h_name
 - 공식 도메인 이름
- h_aliases *** ***
 - 별칭의 도메인 이름
- h_addrtype
 - 반환된 IP 주소의 정보가 IPv4인 경우, AF_INET이 반환
- h_length
 - 반환된 IP 정보의 크기 저장
 - IPv4: 4
 - IPv6: 16
- h_addr_list
 - IP의 주소 정보, 둘 이상인 경우 모두 반환
 - 접속자 수가 많은 서버는 하나의 도메인에 여러 IP 주소 사용

hostent 구조체



gethostbyname 함수 호출 예

예제 gethostbyname.c의 일부

← host ent를 검색

```
host = gethostbyname(argv[1]);
if(!host)
    error_handling("gethost ... error");

printf("Official name: %s\n", host->h_name);

for(i=0; host->h_aliases[i]; i++)
    printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);

printf("Address type: %s\n",
       (host->h_addrtype == AF_INET)? "AF_INET" : "AF_INET6");

for(i=0; host->h_addr_list[i]; i++)
    printf("IP addr %d %s\n", i+1,
           inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));
```

실행 결과

```
$ gcc gethostbyname.c -o hostname
$ ./hostname www.naver.com ✓
Official name: www.naver.com.nheos.com
→ Aliases 1: www.naver.com
Address type: AF_INET
IP addr 1 223.130.200.104
IP addr 2 223.130.195.95
```

- 반복문을 통해서 반환된 모든 정보를 출력하고 있음
- 정보의 끝은 NULL로 표시가 됨

gethostbyname.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int i;
    struct hostent *host;
    if(argc != 2) {
        printf("Usage: %s <addr>\n", argv[0]);
        exit(1);
    }
    host = gethostbyname(argv[1]);

    if(!host)
        error_handling("gethost ... error");

    printf("Official name: %s\n", host->h_name);

    for(i=0; host->h_aliases[i]; i++)
        printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);
```

```
        printf("Address type: %s\n",
               (host->h_addrtype == AF_INET)? "AF_INET" : "AF_INET6");

        for(i=0; host->h_addr_list[i]; i++)
            printf("IP addr %d %s\n", i+1,
                  inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));

    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

h_addr_list를
char* inet_ntoa() 함수를 이용해서
변환함

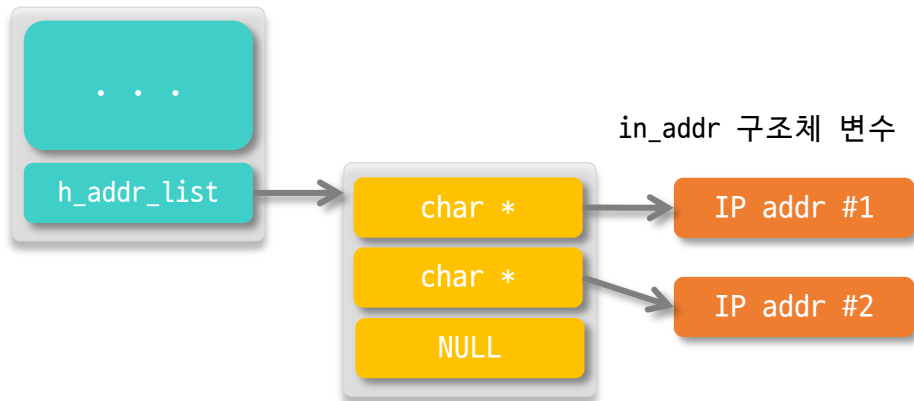
gethostbyname.c 실행

```
$ gcc gethostbyname.c -o hostname

$ ./hostname www.naver.com
Official name: www.naver.com.nheos.com
Aliases 1: www.naver.com
Address type: AF_INET
IP addr 1 223.130.200.107
IP addr 2 223.130.195.200

$ ./hostname www.google.com
Official name: www.google.com
Address type: AF_INET
IP addr 1 142.250.196.100
```

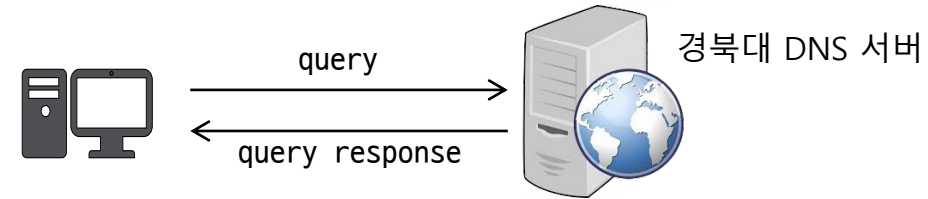
hostent 구조체 변수



- h_addr_list는 둘 이상의 문자열 주소값으로 구성된 배열을 가리킴
- 실제 h_addr_list는 in_addr 구조체 변수의 주소값을 가리킴
 - IPv4 뿐만 아니라 IPv6 주소를 저장하기 위함

✓gethostbyname 쿼리 과정

- wireshark 캡처 내용
 - protocol: dns



Wireshark Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
82	2.992701	155.230.120.233	155.230.120.2	DNS	83	Standard query 0x297f A www.naver.com.nheos.com
84	2.995069	155.230.120.2	155.230.120.233	DNS	220	Standard query response 0x297f A www.naver.com.nheos.com A 223.130.195.200 A 223.130.200.107

Wireshark Packet Details (Frame 84):

- Frame 84: 220 bytes on wire (1760 bits), 220 bytes captured (1760 bits) on interface en0, id 0
- Ethernet II, Src: Cisco_05:ff:41 (00:6c:bc:05:ff:41), Dst: Apple_47:60:e1 (9c:76:0e:47:60:e1)
- Internet Protocol Version 4, Src: 155.230.120.2, Dst: 155.230.120.233
- User Datagram Protocol, Src Port: 53, Dst Port: 50641
- Domain Name System (response)
 - Transaction ID: 0x297f
 - Flags: 0x8180 Standard query response, No error
 - Questions: 1
 - Answer RRs: 2
 - Authority RRs: 3
 - Additional RRs: 3
 - Queries
 - www.naver.com.nheos.com: type A, class IN
 - Answers
 - www.naver.com.nheos.com: type A, class IN, addr 223.130.195.200
 - www.naver.com.nheos.com: type A, class IN, addr 223.130.200.107
 - Authoritative nameservers
 - Additional records
 - [Request In: 82]
 - [Time: 0.002368000 seconds]

캡처 내용

↑ KNU DNS 서버에 요청

이런걸 해보고 싶다?
(캡처) ↓
pcaplib
가 있다.

IP주소를 이용해서 도메인 정보 얻어오기

- `gethostbyaddr()` 함수
 - IP 주소를 이용하여 도메인 정보를 얻어옴
 - `gethostbyname()` 함수의 역 기능

```
#include <netdb.h>

struct hostent* gethostbyaddr(const char *addr, socklen_t len, int family);
```

-> 성공 시 hostent 구조체 변수의 주소 값, 실패 시 NULL 포인터 반환

- `addr`
 - IP주소를 가지는 `in_addr` 구조체 변수의 포인터 전달
 - IPv4 이외의 다양한 정보를 전달받을 수 있도록 char형 포인터로 선언 (범용)
- `len`
 - 주소 정보의 길이 전달 (IPv4: 4, IPv6: 16)
- `family`
 - `AF_INET`: IPv4
 - `AF_INET6`: IPv6

gethostbyaddr 함수의 호출 예

kyungpook.ac.kr

예제 gethostbyaddr.c의 일부

```
memset(&addr, 0, sizeof(addr));
addr.sin_addr.s_addr=inet_addr(argv[1]);

host = gethostbyaddr((char*)&addr.sin_addr, 4, AF_INET);

if(!host)
    error_handling("gethost ... error");

printf("Official name: %s\n", host->h_name);

for(i=0; host->h_aliases[i]; i++)
    printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);

printf("Address type: %s\n", (host->h_addrtype == AF_INET) ?
        "AF_INET":"AF_INET6");

for(i=0; host->h_addr_list[i]; i++)
    printf("IP addr %d: %s\n", i+1,
        inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));
```

실행 결과

```
$ ./hostaddr 155.230.10.2
Official name: ns.kyungpook.kr
Address type: AF_INET
IP addr 1: 155.230.10.2
```

gethostbyaddr.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int i;
    struct hostent *host;
    struct sockaddr_in addr;

    if(argc != 2) {
        printf("Usage: %s <IP>\n", argv[0]);
        exit(1);
    }

    memset(&addr, 0, sizeof(addr));
    addr.sin_addr.s_addr=inet_addr(argv[1]);

    host = gethostbyaddr((char*)&addr.sin_addr, 4, AF_INET);

    if(!host)
        error_handling("gethost ... error");
```

```
    printf("Official name: %s\n", host->h_name);

    for(i=0; host->h_aliases[i]; i++)
        printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);

    printf("Address type: %s\n",
        (host->h_addrtype == AF_INET)?"AF_INET":"AF_INET6");

    for(i=0; host->h_addr_list[i]; i++)
        printf("IP addr %d: %s\n", i+1,
            inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));

    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

gethostbyaddr.c 실행

```
$ gcc gethostbyaddr.c -o hostaddr
```

```
$ ./hostaddr 155.230.10.2
```

```
Official name: ns.knu.ac.kr
```

```
Address type: AF_INET
```

```
IP addr 1: 155.230.10.2
```

```
$ ./hostaddr 168.126.63.1
```

```
Official name: kns.kornet.net
```

```
Address type: AF_INET
```

```
IP addr 1: 168.126.63.1
```


Questions?

LMS Q&A 게시판에 올려주세요.