Part 03. 리눅스 소켓 프로그래밍

# Chapter 04. 주소 변환

## 진행 순서

Chapter 04_01 주소 변환
Chapter 04_02 주소 변환 실습

# Chapter 04_01 주소 변환

주소 변환 관련 주요 함수들

| | |
|---|---|
| gethostbyname | 문자열 호스트 이름(도메인 이름)으로 호스트 정보(IP주소 정보 등)를 얻는 함수 |
| inet_addr | IPv4 문자열 주소 형태를 바이너리 주소 형태(network byte order)로 변환하는 함수 |
| inet_aton | IPv4 문자열 주소 형태를 바이너리 주소 형태(network byte order)로 변환하는 함수 |
| inet_pton | IPv4/IPv6 문자열 주소 형태를 바이너리 주소 형태로 변환하는 함수 |
| => getaddrinfo | |
| | |
| gethostbyaddr | 바이너리 IP 주소로 호스트 정보를 얻는 함수 |
| inet_ntoa | IPv4 바이너리 주소(network) 형태를 문자열 주소 형태(address)로 변환하는 함수 |
| inet_ntop | IPv4/IPv6 바이너리 주소(network) 형태를 문자열 주소 형태로 변환하는 함수 |
| => getnameinfo | |

# Chapter 04_01 주소 변환

```
#include <netdb.h>
extern int h_errno;
struct hostent *gethostbyname(const char *name);

#include <sys/socket.h>      /* for AF_INET */
struct hostent *gethostbyaddr(const void *addr, socklen_t len, int type);

The hostent structure is defined in <netdb.h> as follows:
struct hostent {
        char  *h_name;          /* official name of host */
        char **h_aliases;       /* alias list */
        int    h_addrtype;      /* host address type */
        int    h_length;        /* length of address */
        char **h_addr_list;     /* list of addresses */
}
```

gethostbyname() 및 gethostbyaddr() 함수는 더 이상 사용되지 않습니다.
애플리케이션은 대신 getaddrinfo() 및 getnameinfo() 를 사용해야합니다.

# Chapter 04_01 주소 변환

```
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char **argv)
{
    struct hostent *hp = NULL;
    int i;

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

    hp = gethostbyname(argv[1]);
    if (!hp) {
        printf("gethostbyname error: %s\n", hstrerror(h_errno));
        return 0;
    }
```

```
    printf("h_name: %s\n", hp->h_name);
    printf("h_length: %d\n", hp->h_length);

    for (i = 0; i < hp->h_length && hp->h_addr_list[i]; i++) {
        char *addr = (char *)hp->h_addr_list[i];
        struct in_addr *in = (struct in_addr *)addr;
        printf("h_addr_list[%d]: %s\n", i, inet_ntoa(*in));
    }

    return 0;
}
```

```
$ gcc -g gethostbyname_example.c -o gethostbyname_example
$ ./gethostbyname_example
Usage: ./gethostbyname_example <hostname>
[parallels@localhost ch04]$ ./gethostbyname_example daum.net
h_name: daum.net
h_length: 4
h_addr_list[0]: 211.231.99.17
h_addr_list[1]: 203.133.167.81
h_addr_list[2]: 203.133.167.16
h_addr_list[3]: 211.231.99.80
```

# Chapter 04_01 주소 변환

getaddrinfo() 및 getnameinfo() 함수는 운영 체제의 네트워킹 API를 위해 사람이 읽을 수 있는 텍스트 표현과 구조화 된 이진 형식 간에 도메인 이름, 호스트 이름 및 IP 주소를 변환합니다.
두 기능 모두 POSIX 표준 API (Application Programming Interface)에 포함되어 있습니다.

getaddrinfo와 getnameinfo는 서로 역함수입니다. 네트워크 프로토콜에 구애 받지 않으며 IPv4와 IPv6을 모두 지원합니다. 프로토콜 독립 응용 프로그램을 구축 할 때 이름을 확인하고 레거시 IPv4 코드를 IPv6 인터넷으로 전환하는 데 권장되는 인터페이스입니다.

내부적으로 이 함수는 gethostbyname()과 같은 다른 하위 수준 함수를 호출하여 DNS (Domain Name System)를 사용하여 확인을 수행합니다.

네트워킹 API 내에서 주소와 호스트 이름을 나타내는 데 사용되는 C 데이터 구조는 다음과 같습니다.

```
struct addrinfo {
   int      ai_flags;
   int      ai_family;
   int      ai_socktype;
   int      ai_protocol;
   socklen_t ai_addrlen;
   struct   sockaddr* ai_addr;
   char*    ai_canonname;     /* canonical name */
   struct   addrinfo* ai_next; /* this struct can form a linked list */
};
```

# Chapter 04_01 주소 변환

getaddrinfo()는 호스트 이름 또는 IP 주소를 나타내는 사람이 읽을 수 있는 텍스트 문자열을 구조체 addrinfo 구조의 동적으로 할당된 링크된 목록으로 변환합니다. 이러한 기능의 기능 프로토타입은 다음과 같이 지정됩니다.

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

int getaddrinfo(const char* hostname,
                const char* service,
                const struct addrinfo* hints,
                struct addrinfo** res);
```

hostname
"example.com"과 같은 도메인 이름, "127.0.0.1"과 같은 주소 문자열 또는 NULL 일 수 있습니다.이 경우 힌트 플래그에 따라 주소 0.0.0.0 (hints-> ai_flags가 AI_PASSIVE로 설정) 또는 127.0.0.1이 할당됩니다.

service
"80"과 같은 문자열로 전달 된 포트 번호 또는 서비스 이름 (예 : ”echo". 후자의 경우 일반적인 구현에서는 getservbyname()을 사용하여 /etc/services 파일을 쿼리하여 서비스를 포트 번호로 확인합니다.

hints
요청 된 서비스 유형이 있는 NULL 또는 addrinfo 구조 일 수 있습니다.

res
함수가 성공적으로 완료된 후 요청 된 정보를 사용하여 새로운 addrinfo 구조를 가리키는 포인터입니다.

성공 시 0을 반환하고 실패하면 0이 아닌 오류 값을 반환합니다.

# Chapter 04_01 주소 변환

getnameinfo()는 구조체 sockaddr 포인터 형태의 IP 주소의 내부 이진 표현을 호스트 이름으로 구성된 텍스트 문자열로 변환하거나, 주소를 이름으로 확인할 수 없는 경우 텍스트 IP 주소 표현과 서비스 포트 이름 또는 번호로 변환합니다. 기능 프로토타입은 다음과 같이 지정됩니다.

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr* sa, socklen_t salen,
        char* host, size_t hostlen,
        char* serv, size_t servlen,
        int flags);
```

| flags | |
|---|---|
| NI_NAMEREQD | 설정된 경우 호스트 이름을 확인할 수 없으면 오류가 반환됩니다. |
| NI_DGRAM | 설정된 경우 서비스는 스트림 (TCP)이 아닌 데이터 그램 (UDP) 기반입니다. |
| NI_NOFQDN | 설정된 경우 로컬 호스트에 대한 완전한 도메인 이름의 호스트 이름 부분만 리턴하십시오. |
| NI_NUMERICHOST | 설정된 경우 호스트 이름의 숫자 형식이 반환됩니다. |
| NI_NUMERICSERV | 설정된 경우 서비스 주소의 숫자 형식이 반환됩니다. |

# Chapter 04_01 주소 변환

이 함수는 getaddrinfo() 함수에 의해 할당 된 메모리를 해제합니다.
getaddrinfo()의 결과는 addrinfo 구조체의 링크 된 목록이므로, freeaddrinfo()는 목록을 반복하고 각 목록을 차례로 해제합니다.

```
#include <sys/socket.h>
#include <netdb.h>

void freeaddrinfo(struct addrinfo *ai);
```

ai는 addrinfo 목록의 헤드입니다.

```
const char *gai_strerror(int errcode);
```

getaddrinfo() 에러 발생 시 오류코드를 리턴합니다.
gai_strerror() 함수는 이러한 오류 코드를 사람이 읽을 수 있는 문자열로 변환하여 오류보고에 적합합니다.

# Chapter 04_02 주소 변환 실습

다음 예제는 getaddrinfo()를 사용하여 입력 받은 도메인 이름을 주소 목록으로 해석한 다음 각 결과에서 getnameinfo()를 호출하여 주소의 표준 이름을 출력합니다. <출처: https://en.wikipedia.org/wiki/Getaddrinfo>

```
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>

#ifndef   NI_MAXHOST
#define   NI_MAXHOST 1025
#endif

int main(int argc, char *argv[])
{
   struct addrinfo* result;
   struct addrinfo* res;
   int error;

   if (argc < 2) {
                fprintf(stderr, "Usage: %s <domain name, ex. www.example.com>\n", argv[0]);
                exit(EXIT_FAILURE);
   }
```

# Chapter 04_02 주소 변환 실습

```
  /* resolve the domain name into a list of addresses */
  error = getaddrinfo(argv[1], NULL, NULL, &result);
  if (error != 0) {
     if (error == EAI_SYSTEM) {
        perror("getaddrinfo");
     } else {
        fprintf(stderr, "error in getaddrinfo: %s\n", gai_strerror(error));
     }
     exit(EXIT_FAILURE);
  }

  /* loop over all returned results and do inverse lookup */
  for (res = result; res != NULL; res = res->ai_next) {
     char hostname[NI_MAXHOST];
     error = getnameinfo(res->ai_addr, res->ai_addrlen, hostname, NI_MAXHOST, NULL, 0, 0);
     if (error != 0) {
        fprintf(stderr, "error in getnameinfo: %s\n", gai_strerror(error));
        continue;
     }
     if (*hostname != '\0')
        printf("hostname: %s\n", hostname);
  }

  freeaddrinfo(result);
  return 0;
}
```

# Chapter 04_02 주소 변환 실습

```
[parallels@localhost ch04]$ gcc -g traslate_addr.c -o traslate_addr
[parallels@localhost ch04]$ ./traslate_addr
Usage: ./traslate_addr <domain name, ex. www.example.com>
[parallels@localhost ch04]$ ./traslate_addr daum.net
hostname: 203.133.167.16
hostname: 203.133.167.16
hostname: 203.133.167.16
hostname: 211.231.99.80
hostname: 211.231.99.80
hostname: 211.231.99.80
hostname: 203.133.167.81
hostname: 203.133.167.81
hostname: 203.133.167.81
hostname: 211.231.99.17
hostname: 211.231.99.17
hostname: 211.231.99.17
[parallels@localhost ch04]$ ./traslate_addr naver.com
hostname: 125.209.222.141
hostname: 125.209.222.141
hostname: 125.209.222.141
hostname: 210.89.164.90
hostname: 210.89.164.90
hostname: 210.89.164.90
hostname: 210.89.160.88
hostname: 210.89.160.88
hostname: 210.89.160.88
hostname: 125.209.222.142
hostname: 125.209.222.142
hostname: 125.209.222.142
[parallels@localhost ch04]$
```