인터넷 프로토콜 03장

도메인 네임 시스템과 주소 패밀리 (IPv4-IPv6 서비스)

목포해양대 해양컴퓨터공학과

목차

기존 IPv4전용, IPv6전용 코드의 문제

- 전용주소 코드(IPv4 only, IPv6 only)의 의미
  - IPv4 전용 코드는 IPv4 형식의 IPv4만 취하고 IPv6 전용 코드는 IPv6 주소 형식만 취한다.
  - 상대방의 IP 주소 버전을 모를 경우, 두 가지 버전을 모두 준비해야 함
- IPv4, IPv6 범용 코드
  - 실행 시간에 주소 버전을 확인하여 IPv4, IPv6 주소 타입에 관계없이 동작하게 하는 코드
  - 내부적으로는 이름-주소 변환 함수인 getaddrinfo() 함수를 사용하여 동작
    - getaddrinfo()는 /etc/hosts, DNS 시스템에 질의하여 가능한 모든 IPv4(A record), IPv6(AAAA record) 주소를 리스트화 하여 반환
    - 이름 주소-> IP주소로 변환하는 네임 시스템 API
  - 하나의 코드로 IPv4, IPv6에 모두 대비

도메인 네임 시스템

- 도메인 네임 시스템(Domain Name System)이란?
  - 인터넷에서 호스트를 구분하기 위하여 IP 대신 네임(이름 주소)을 사용할 수 있도록 하는 서비스
  - 네임주소 - IP 주소를 매핑하는 DB를 활용하여 서비스
    - 로컬 DB 활용 : /etc/hosts(linux) or windows/system32/dirvers/etc/hosts(windows)
    - 분산 DB 활용 : DNS(domain Name System)

도메인 네임 시스템

- 장점
  - 읽기, 쓰기, 기억의 편의성
    - 인터넷에서 호스트는 IP 주소로 구분이 가능
    - 숫자 형태보다, 계층화된 이름 주소가 더 좋은 사용 편의성을 제공
  - 고정된 주소 값 제공
    - IP 주소는 특성상 위치 이동 시 변경되나 네임 주소는 이를 클라이언트에 숨겨주어 다른 사람에게 항상 같은 주소를 제공한다
  - 부하 분배(load balancing)
    - 하나의 네임주소에 여러 개의 IP 주소 매핑이 가능하며 결과적으로 서로 다른 물리적인 서버가 클라이언트의 요청에 대응하게 할 수 있다
- 특징
  - DNS가 TCP/IP 프로그래밍의 필수요소는 아님

IPv4, IPv6 통합 네임 서비스 API

```
int getaddrinfo (const char *hostStr, const char *serviceStr,
        const struct addrinfo *hints, struct addrinfo **results);
```

- 기능 : 프로토콜 버전에 상관없이 네임 주소 -> IP 주소 해석을 해주는 함수
  - 호스트 주소(IP 혹은 도메인 네임)와 서비스(서비스 이름 혹은 port 번호)을 전달하면 위 정보에 연결 가능한 주소 정보(addrinfo) 리스트를 반환한다
  - 호스트 연결 시 도메인 네임, IPv4 주소, IPv6 주소를 모두 사용가능
    - hostStr: 네임 주소 혹은 IP 주소
    - serviceStr: 서비스 이름 혹은 port 번호
    - hints: 반환을 원하는 주소 정보의 형태
      - IPv4 및 IPv6 선택, 프로토콜 종류 등의 선택이 가능
    - results: 반환되는 주소들의 결과 리스트

addrinfo 구조체

```
struct addrinfo{
    int ai_flags;// 제어 정보 해설을 위한 flag
    int ai_family;//Family:AF_INET,AF_INET6,AF_UNSPEC
    int ai_socktype;//Socket type:SOCK_STREAM,SOCK_DGRAM
    int ai_protocol;//Protocol:0(default)or IPPROTO_XXX
    socklen_t ai_addrlen;// 소켓 주소인 ai_addr의 길이
    struct sockaddr *ai_addr;//소켓 주소
    char *ai_canonname;//Canonical 네임
    struct addrinfo *ai_next;//연결리스트에서 다음 addrinfo의 위치
};
```

GetAddrInfo.c (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include "Practical.h"

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

        if (argc != 3) // Test for correct number of arguments
                DieWithUserMessage("Parameter(s)", "<Address/Name> <Port/Service>");

        char *addrString = argv[1];   // Server address/name
        char *portString = argv[2];   // Server port/service

        // Tell the system what kind(s) of address info we want
        struct addrinfo addrCriteria;                   // Criteria for address match
        memset(&addrCriteria, 0, sizeof(addrCriteria)); // Zero out structure
        addrCriteria.ai_family = AF_UNSPEC;             // Any address family
        addrCriteria.ai_socktype = SOCK_STREAM;         // Only stream sockets
        addrCriteria.ai_protocol = IPPROTO_TCP;         // Only TCP protocol

```

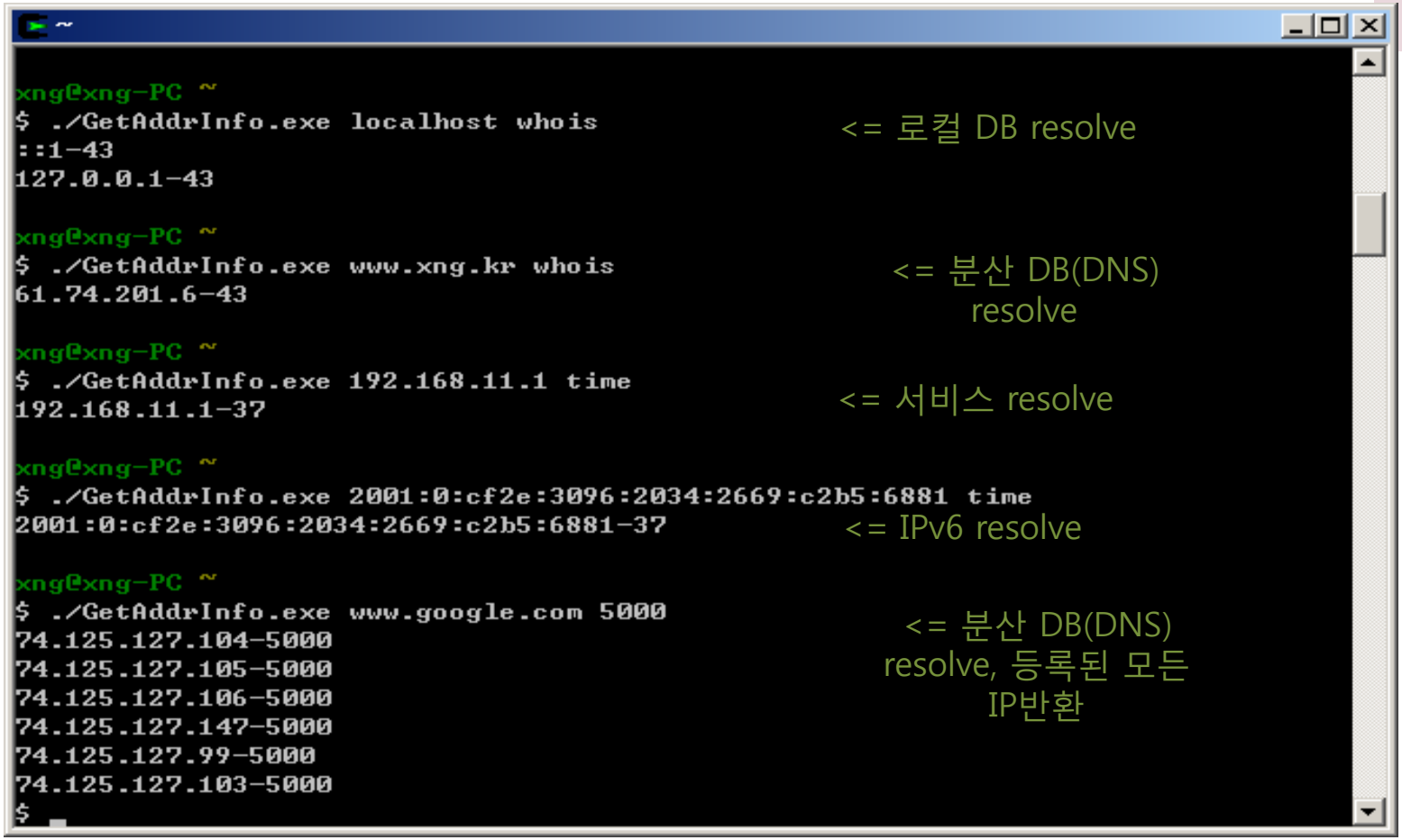
GetAddrInfo.c (2)

```
22          // Get address(es) associated with the specified name/service
23          struct addrinfo *addrList; // Holder for list of addresses returned
24          // Modify servAddr contents to reference linked list of addresses
25          int rtnVal = getaddrinfo(addrString, portString, &addrCriteria, &addrList);
26          if (rtnVal != 0)
27                  DieWithUserMessage("getaddrinfo() failed", gai_strerror(rtnVal));
28
29          // Display returned addresses
30          for (struct addrinfo *addr = addrList; addr != NULL; addr = addr->ai_next) {
31                  PrintSocketAddress(addr->ai_addr, stdout);
32                  fputc('\n', stdout);
33          }
34
35          freeaddrinfo(addrList); // Free addrinfo allocated in getaddrinfo()
36
37          exit(0);
38  }
```

Makefile

```
CFLAGS = -g -std=gnu99
COM_OBJS = DieWithMessage.o AddressUtility.o
CLI_OBJS = TCPEchoClient4.o
SRV_OBJS = TCPServerUtility.o TCPEchoServer4.o
ADD_OBJS = GetAddrInfo.o

TARGETS = echo_cli echo_srv

all:    $(TARGETS)

echo_cli : $(COM_OBJS) $(CLI_OBJS)
        gcc -o echo_cli $(CLI_OBJS) $(COM_OBJS)

echo_srv : $(SRV_OBJS) $(COM_OBJS)
        gcc -o echo_srv $(SRV_OBJS) $(COM_OBJS)

get_addr : $(COM_OBJS) $(ADD_OBJS)
        gcc -o get_addr $(COM_OBJS) $(ADD_OBJS)

clean:
        rm $(COM_OBJS) $(CLI_OBJS) $(SRV_OBJS)
        rm $(TARGETS)
```

get_addr 실행 예시



과제

- getAddrInfo.c 작성 후 실행 파일 만들기 (100점)
  - 관련 API 정리
  - 프로그램에 주석 달기
  - 실행 파일로 주요 서버 ip 주소와 포트 확인하기

    [kgu@lily ch03]$ ./a.out iris.mmu.ac.kr ssh
    203.232.252.109-22
    [kgu@lily ch03]$ ./a.out www.ibm.com www
    129.42.56.216-80
- getAddrInfo.c에서 AF_UNSPEC과 AF_INET 차이 확인 (200점)
  - 함수 호출 전 시간을 기록하여 실제로 getAddrInfo 함수 실행 시간 출력
  - 각각의 결과 최대한 출력
    - 결과 수, 각 주소/포트 등