Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інтегрованих інформаційних систем

Лабораторна робота №2

Мережеве програмування у середовищі UNIX

Тема: «Мережеві адреси та імена»

Виконав:
Студент групи ІА-12
Оверчук Дмитро Максимович

Перевірив:
Сімоненко А.В.

Київ 2025

Завдання на роботу

Розробити програму, яка виконує наступне:

1. Отримує текстове представлення мережевої адреси або мережеве ім’я та/або текстове представлення номера порту або ім’я сервісу в аргументах командного рядка. Програма повинна підтримувати аргумент, який дозволяє вказувати тільки текстові представлення мережевих адрес та номерів портів. Також має бути аргумент для вказування версії IP.
2. Конвертує отриманні дані в структури адрес сокетів.
3. Виводить текстове представлення отриманих структур адрес сокетів. Інформацію про протокол треба виводити в числовому значення та іменем. Має бути аргумент командного рядка, який вказує виводити відповідне мережеве ім’я для мережевої адреси, та аргумент, який вказує виводити ім’я сервісу для номера порту.

Код прогарми

lab.2

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <ctype.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <sys/socket.h>

#include <sys/types.h>


void print_usage(const char *prog_name)

{

    printf("Usage: %s [-t] [-v <4|6>] [-d] [-s] -a <ip/hostname> -p
<port/service>\n", prog_name);

    printf("  -a: IP address or hostname to be converted\n");

    printf("  -p: Port number or service name to be converted\n");

    printf("  -t: Prohibit domain names and service names (use only numeric
values)\n");
```

```
    printf("  -v <4|6>: Use specific IP version (IPv4 or IPv6)\n");
    printf("  -d: Output domain name for specified IP\n");
    printf("  -s: Output service name for specified port\n");
}

// Function to check if a string represents a valid numeric IP address
int is_valid_ip(const char *ip)
{
    struct in_addr ipv4_addr;
    struct in6_addr ipv6_addr;

    if (inet_pton(AF_INET, ip, &ipv4_addr) == 1)
    {
        return 1; // Valid IPv4
    }
    if (inet_pton(AF_INET6, ip, &ipv6_addr) == 1)
    {
        return 1; // Valid IPv6
    }
    return 0; // Not a valid IP address
}

// Function to check if a string contains only digits (valid port number)
int is_valid_port(const char *port)
{
    for (int i = 0; port[i] != '\0'; i++)
    {
        if (!isdigit((unsigned char)port[i]))
        {
            return 0; // Contains non-numeric character
        }
    }
```

```
    int port_num = atoi(port);
    return (port_num >= 0 && port_num <= 65535); // Valid port range
}

// Function to print protocol info
void print_protocol_info(int protocol)
{
    struct protoent *proto = getprotobynumber(protocol);
    printf("Protocol: %d", protocol);
    if (proto != NULL) {
        printf(" (%s)", proto->p_name);
    }
    printf("\n");
}


int main(int argc, char *argv[])
{
    int opt;
    char *hostname = NULL;
    char *port = NULL;
    int only_numeric = 0;
    int ip_version = 0; // 4 for IPv4, 6 for IPv6, 0 for any
    int show_domain = 0;
    int show_service = 0;

    // Обробка аргументів командного рядка
    while ((opt = getopt(argc, argv, "a:p:tv:ds")) != -1)
    {
        switch (opt)
        {
```

```
        case 'a':
            hostname = optarg;
            break;
        case 'p':
            port = optarg;
            break;
        case 't':
            only_numeric = 1;
            break;
        case 'v':
            if (strcmp(optarg, "4") == 0)
                ip_version = 4;
            else if (strcmp(optarg, "6") == 0)
                ip_version = 6;
            else
            {
                fprintf(stderr, "Invalid IP version: %s\n", optarg);
                return 1;
            }
            break;
        case 'd':
            show_domain = 1;
            break;
        case 's':
            show_service = 1;
            break;
        default:
            print_usage(argv[0]);
            return 1;
        }
    }
```

```
    if (!hostname || !port)
    {
        fprintf(stderr, "Missing required arguments: -a <ip/hostname> and -p <port/service>\n");
        print_usage(argv[0]);
        return 1;
    }

    // If -t is specified, ensure -a contains only a numeric IP and -p contains only a numeric port
    if (only_numeric)
    {
        if (!is_valid_ip(hostname))
        {
            fprintf(stderr, "Error: With -t, the argument for -a must be a numeric IP address.\n");
            return 1;
        }
        if (!is_valid_port(port))
        {
            fprintf(stderr, "Error: With -t, the argument for -p must be a numeric port number.\n");
            return 1;
        }
    }

    // Prepare hints for getaddrinfo
    struct addrinfo hints, *result, *rp;
    memset(&hints, 0, sizeof(struct addrinfo));

    if (ip_version == 4)
        hints.ai_family = AF_INET;    // IPv4
    else if (ip_version == 6)
        hints.ai_family = AF_INET6;   // IPv6
```

```
    else
        hints.ai_family = AF_UNSPEC;  // Allow any address family

    hints.ai_socktype = SOCK_STREAM;

    if (only_numeric) {
        hints.ai_flags = AI_NUMERICHOST | AI_NUMERICSERV; // Numeric host and
service
    }

    // Get address info
    int status = getaddrinfo(hostname, port, &hints, &result);
    if (status != 0) {
        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
        return 1;
    }

    // Print the results
    printf("Results for %s:%s\n", hostname, port);
    printf("------------------------\n");

    for (rp = result; rp != NULL; rp = rp->ai_next) {
        char addrstr[INET6_ADDRSTRLEN];
        void *addr;
        int port_num;

        // Get the pointer to the address itself
        if (rp->ai_family == AF_INET) { // IPv4
            struct sockaddr_in *ipv4 = (struct sockaddr_in *)rp->ai_addr;
            addr = &(ipv4->sin_addr);
            port_num = ntohs(ipv4->sin_port);
            printf("IPv4 Address: ");
```

```
        } else { // IPv6
            struct sockaddr_in6 *ipv6 = (struct sockaddr_in6 *)rp->ai_addr;
            addr = &(ipv6->sin6_addr);
            port_num = ntohs(ipv6->sin6_port);
            printf("IPv6 Address: ");
        }

        // Convert IP to string
        inet_ntop(rp->ai_family, addr, addrstr, sizeof(addrstr));
        printf("%s\n", addrstr);

        // Print port
        printf("Port: %d\n", port_num);

        // Print protocol info
        print_protocol_info(rp->ai_protocol);

        // If -d is specified, try to resolve hostname
        if (show_domain) {
            struct sockaddr_storage temp_addr;
            memcpy(&temp_addr, rp->ai_addr, rp->ai_addrlen);
            char host[NI_MAXHOST];

            if (getnameinfo((struct sockaddr*)&temp_addr, rp->ai_addrlen,
                            host, NI_MAXHOST, NULL, 0, 0) == 0) {
                printf("Hostname: %s\n", host);
            } else {
                printf("Hostname: Unable to resolve\n");
            }
        }

        // If -s is specified, try to resolve service name
```

```
        if (show_service) {
            struct servent *service = getservbyport(htons(port_num),
                                                    (rp->ai_socktype == SOCK_DGRAM)
? "udp" : "tcp");
            if (service != NULL) {
                printf("Service: %s\n", service->s_name);
            } else {
                printf("Service: Unknown\n");
            }
        }

        printf("Socket type: %s\n",
               (rp->ai_socktype == SOCK_STREAM) ? "SOCK_STREAM" :
               (rp->ai_socktype == SOCK_DGRAM) ? "SOCK_DGRAM" :
               (rp->ai_socktype == SOCK_RAW) ? "SOCK_RAW" : "Unknown");

        printf("------------------------\n");
    }

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

Опис програми

Програма представляє собою консольну утиліту для роботи з IP-адресами, доменними іменами, портами та сервісами. Вона дозволяє конвертувати між різними форматами мережевих адрес та отримувати додаткову інформацію про них. Основні можливості програми:

- Конвертація між IP-адресами та доменними іменами
- Конвертація між номерами портів та іменами сервісів
- Підтримка як IPv4, так і IPv6 адрес
- Отримання додаткової інформації про протоколи та типи сокетів

Аргументи командного рядка: -a - IP-адреса або ім’я хоста для обробки (обов’язковий параметр) -p - номер порту або ім’я сервісу для обробки

(обов’язковий параметр) -t - заборона використання доменних імен та імен сервісів (лише числові значення) -v - вказівка конкретної версії IP (4 для IPv4 або 6 для IPv6) -d - виведення доменного імені для вказаної IP-адреси -s - виведення імені сервісу для вказаного порту

Алгоритм роботи програми:

1. Обробка аргументів командного рядка за допомогою getopt()
2. Перевірка наявності обов’язкових аргументів та їх валідність
3. Налаштування структури addrinfo відповідно до заданих параметрів
4. Виклик функції getaddrinfo() для отримання інформації про адресу та порт
5. Ітерація по результатах і виведення детальної інформації для кожної знайденої адреси:
   - IP-адреса (IPv4 або IPv6)
   - Номер порту
   - Інформація про протокол
   - Доменне ім’я (якщо запитано через -d)
   - Ім’я сервісу (якщо запитано через -s)
   - Тип сокету

Приклади використання програми

Приклад 1: Базове використання

```
$ ./lab2.out -a google.com -p 80
Results for google.com:80
-----------------------
IPv4 Address: 142.250.186.206
Port: 80
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
-----------------------
IPv6 Address: 2a00:1450:401b:80e::200e
Port: 80
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
-----------------------
```

Ця команда перетворить доменне ім’я “google.com” на відповідну IP-адресу та відобразить інформацію про порт 80. Програма покаже всі знайдені IP-адреси (як IPv4, так і IPv6), їхні номери портів, протоколи та типи сокетів.

Приклад 2: Використання з отриманням доменного імені та сервісу

```
$ ./lab2.out -a 8.8.8.8 -p 53 -d -s
Results for 8.8.8.8:53
-----------------------
IPv4 Address: 8.8.8.8
Port: 53
Protocol: 6 (tcp)
Hostname: dns.google
Service: domain
Socket type: SOCK_STREAM
-----------------------
```

У цьому випадку програма працює з IP-адресою 8.8.8.8 (публічний DNS-сервер Google) та портом 53 (стандартний порт DNS). Прапорець -d запускає зворотний DNS-пошук, намагаючись знайти доменне ім'я, пов'язане з цією IP-адресою. Це корисно для ідентифікації серверів за їхніми IP-адресами. Прапорець -s змушує програму шукати назву сервісу, пов'язаного з портом 53. У результаті ви побачите, що цей порт використовується для сервісу DNS (Domain Name System).

Приклад 3: Використання лише числових значень

```
./lab2.out -a 192.168.1.1 -p 22 -t
Results for 192.168.1.1:22
-----------------------
IPv4 Address: 192.168.1.1
Port: 22
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
-----------------------

$ ./lab2.out -a 192.168.1.1 -p ssh -t
Error: With -t, the argument for -p must be a numeric port number.
```

Прапорець -t обмежує програму використанням лише числових значень. Це означає, що програма не спробує розв'язати доменні імена або назви сервісів. Вона працюватиме безпосередньо з IP-адресою 192.168.1.1 та портом 22 (SSH).

Приклад 4: Використання лише IPv4

```
$ ./lab2.out -a example.com -p 443 -v 4
Results for example.com:443
-----------------------
IPv4 Address: 23.192.228.80
Port: 443
```

```
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
IPv4 Address: 96.7.128.198
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
IPv4 Address: 23.192.228.84
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
IPv4 Address: 96.7.128.175
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
IPv4 Address: 23.215.0.136
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
IPv4 Address: 23.215.0.138
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
```

Прапорець -v 4 вказує програмі шукати лише IPv4-адреси для доменного імені “example.com”. Програма ігноруватиме будь-які IPv6-адреси, навіть якщо вони доступні для цього домену.

Приклад 5: Використання з ім’ям сервісу замість номера порту

```
$ ./lab2.out -a github.com -p https
Results for github.com:https
----------------------
IPv4 Address: 140.82.121.4
Port: 443
Protocol: 6 (tcp)
Socket type: SOCK_STREAM
----------------------
```

У цьому прикладі замість конкретного номера порту використовується ім’я сервісу “https”. Програма автоматично перетворить його на відповідний номер порту (443) завдяки використанню системних файлів, таких як /etc/services, які зіставляють імена сервісів із номерами портів.

Приклад 6: Повний приклад з усіма доступними опціями

```
$ ./lab2.out -a stackoverflow.com -p 443 -v 4 -d -s
Results for stackoverflow.com:443
-----------------------
IPv4 Address: 104.18.32.7
Port: 443
Protocol: 6 (tcp)
Hostname: 104.18.32.7
Service: https
Socket type: SOCK_STREAM
-----------------------
IPv4 Address: 172.64.155.249
Port: 443
Protocol: 6 (tcp)
Hostname: 172.64.155.249
Service: https
Socket type: SOCK_STREAM
-----------------------
```

Цей повний приклад демонструє використання багатьох прапорців одночасно: -a stackoverflow.com: перетворення доменного імені на IP -p 443: використання порту 443 (HTTPS) -v 4: обмеження результатів лише IPv4-адресами -d: виконання зворотного DNS-пошуку -s: пошук назви сервісу для порту 443