

ネットワークプログラミング 2

自マシンのIPアドレスを知る

- プログラムでは、IPアドレスとポートをしている
 - プログラムからマシンのIPアドレスを知る方法はないだろうか？

【ip_list_out1.c】

- `gethostname()` と `gethostbyname()` を使用

```
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    char hostname[256];
    if (gethostname(hostname, sizeof(hostname)) == 0) {
        struct hostent *host = gethostbyname(hostname);
        if (host != NULL) {
            printf("ホスト名:%s\n", host->h_name);
            struct in_addr **addr_list = (struct in_addr **)host->h_addr_list;
            for (int i = 0; addr_list[i] != NULL; i++) {
                printf("IPアドレス: %s\n", inet_ntoa(*addr_list[i]));
            }
        }
    }
    return 0;
}
```

- `gethostbyname()` には、セキュリティ上の脆弱性(GHOST)が存在する
 - どのような脆弱性か？以下の観点を踏まえて調査・理解をしてください
 - バッファオーバーフロー
 - キャッシュポイズニング
 - どのように改善すればよいだろうか？

課題

- 上記、レポートにて提出のこと
 - ファイル名：12_report.*

補足

- Linuxは、glibc（現在はlibc6）を元にコンパイルされている
- ライブラリのバージョン確認方法（ubuntuの場合）

```
$ dpkg -l | grep libc6
ii  libc6:amd64                2.39-0ubuntu8.5          amd64
    GNU C Library: Shared libraries
ii  libc6-dbgsym:amd64        2.39-0ubuntu8.5          amd64
    GNU C Library: detached debugging symbols
ii  libc6-dev:amd64           2.39-0ubuntu8.5          amd64
    GNU C Library: Development Libraries and Header Files
```

- 吉村の環境（ubuntu24.04）では、バージョン2.39を用いていることが分かる
- Ubuntu以外の場合

```
$ ldd --version
ldd (Ubuntu GLIBC 2.39-0ubuntu8.5) 2.39
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
作者 Roland McGrath および Ulrich Drepper。
```

改良版

【ip_list_out1_fix.c】

```
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h> /* memset() */
#include <unistd.h>

int main() {
    char hostname[256];
    if (gethostname(hostname, sizeof(hostname)) == 0) {
        struct addrinfo hints;
```

```

memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = 0;

struct addrinfo *result, *rp;
int s = getaddrinfo(hostname, NULL, &hints, &result);
if (s != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
    exit(EXIT_FAILURE);
}

for (rp = result; rp != NULL; rp = rp->ai_next) {
    struct sockaddr_in *addr = (struct sockaddr_in *)rp->ai_addr;
    char ip[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &addr->sin_addr, ip, sizeof(ip));
    printf("IPアドレス: %s\n", ip);
}

freeaddrinfo(result);
}
return 0;
}

```

別の方法

【ip_list_out2.c】

- `getifaddrs()` を使用

```

#include <arpa/inet.h>
#include <ifaddrs.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    struct ifaddrs *ifaddr, *ifa;
    if (getifaddrs(&ifaddr) == -1) {
        perror("getifaddrs");
        return -1;
    }
    for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
        if (ifa->ifa_addr != NULL && ifa->ifa_addr->sa_family == AF_INET) {
            struct sockaddr_in *addr = (struct sockaddr_in *)ifa->ifa_addr;
            char ip[INET_ADDRSTRLEN];
            inet_ntop(AF_INET, &(addr->sin_addr), ip, INET_ADDRSTRLEN);

```

```
        printf("インターフェイス名:%10s\tIPアドレス: %s\n", ifa->ifa_name, ip);
    }
}
freeifaddrs(ifaddr);
return 0;
}
```

プログラムの修正

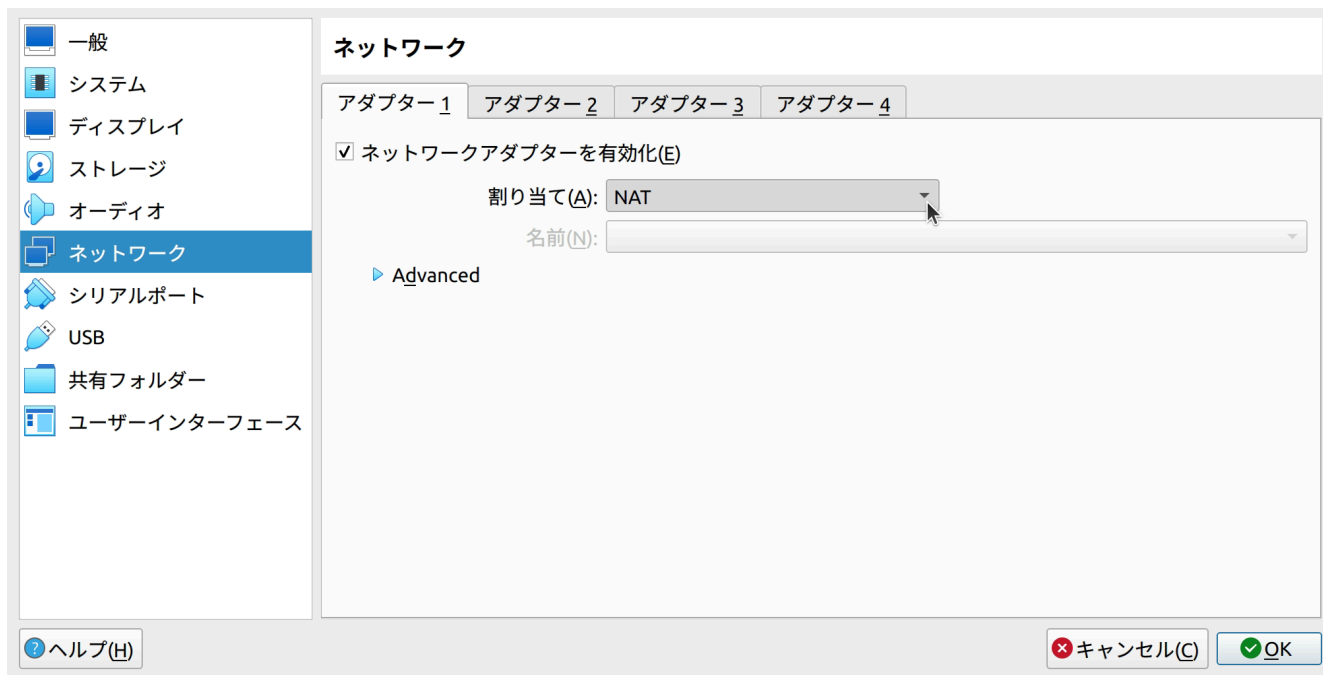
- 昨今のPCでは、1台のマシンで複数のIPアドレスを保持することが可能になるため、明示的に指定するしか方法がない
- そこで以下のプログラムをコマンドライン引数で、IPアドレス・ポート番号を設定できるようにしてください
 - socket_tcp_server.c
 - socket_tcp_client.c
- 正しく動作しているのを確認してください

課題

- 上記プログラム提出
 - コンパイルし、正常に動作するのをテストした上で提出

外部マシンからの接続

- 現在、仮想マシンのネットワーク設定では「NAT」になっている



- これを「ブリッジアダプター」に変更することで、Host側（Windows）と同じネットワークへ参加することになる
 - 他のWindowsマシンから接続することが可能になる
 - 変更するためには仮想マシンを電源OFF & 起動（再起動）する必要がある
- ブリッジで、起動したらマシンのIPアドレスを確認してください
 - Linuxでは、`ip a` で自マシンのネットワーク状況を確認することができる
 - Windowsでは `ipconfig /all`
- 実際に友人のマシンと通信を行ってください

wiresharkの利用

- パケットがどのように届いているか確認してみよう

wiresharkのインストール(Linux版)

```
$ sudo apt install wireshark
```

- これで、インストール完了



- 実行したときに「Couldn't run /usr/bin/dumpcap in child process:許可がありません」とエラーが出力される
 - デフォルトでは管理者しか使えない→実行時に `sudo` を利用する
 - dumpcapコマンドを一般ユーザが利用できるようにする

```
$ sudo chmod +x /usr/bin/dumpcap
```

これで、普通に起動しても利用可能になる

標準コマンドtcpdump

標準コマンドでパケットはキャプチャする

- インターフェイス local-loopback (127.0.0.1) を監視

```
$ sudo tcpdump -i lo
```

- 表示結果をテキストで保存する場合

```
$ sudo tcpdump -i lo > result.txt
```

- バイナリのまま `dump.txt` に保存する場合

```
$ sudo tcpdump -i lo -w dump.txt
```

- 保存されたデータはバイナリデータなので、Editor等では確認できない
- そこで `xxd` コマンドで表示することができる

```
$ xxd dump.txt
```

- wiresharkで開くことでもOK

vscodeに拡張機能をインストール




Hex Editor pre-release

Microsoft  microsoft.com |  5,918,557 |  (59)

Allows viewing and editing files in a hex editor

無効にする ▼

アンインストール ▼

☒ 自動更新 

詳細 機能 変更ログ

A custom editor extension for Visual Studio Code which provides a hex editor for viewing and manipulating files in their raw hexadecimal representation.

Features

- Opening files as hex

別の方法2（ヒント）

- OSの機能を利用する
- `popen()` を使用すると、OSの持つコマンドを実行し、結果を受け取ることが可能。
 - ただし、文字列処理になるので、なれないと面倒

【system.c】

```
#include <stdio.h>
#include <stdlib.h>

#define MAXBUF 256

int main(int argc, char *argv[]) {
    FILE *fp;
```

```

// lsコマンド
const char *command = "ls";

// popenでコマンド実行（読み取りモード）
if ((fp = popen(command, "r")) == NULL) {
    perror("can not exec commad");
    exit(EXIT_FAILURE);
}

char buffer[MAXBUF];
// 出力を読み込んで表示
while (!feof(fp)) {
    fgets(buffer, sizeof(buffer), fp);
    printf("=> %s", buffer);
}

// 終了処理
if (pclose(fp) == -1) {
    perror("pclose failed");
    exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}

```

【system2.c】

```

#include <stdio.h>
#include <stdlib.h>

#define MAXBUF 1024

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

    // tcpdumpコマンド（sudo付き。パスワード不要設定されている前提）
    const char *command = "sudo tcpdump -i lo -nn -c 10"; // -nn: 名前解決無効, -c 10: 10パケットで終了

    // popenでコマンド実行（読み取りモード）
    if ((fp = popen(command, "r")) == NULL) {
        perror("can not exec commad");
        exit(EXIT_FAILURE);
    }

    char buffer[MAXBUF];
    // 出力を読み込んで表示
    while (!feof(fp)) {
        fgets(buffer, sizeof(buffer), fp);
    }
}

```



```
        printf("=> %s", buffer);
    }

    // 終了処理
    if (pclose(fp) == -1) {
        perror("pclose failed");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```