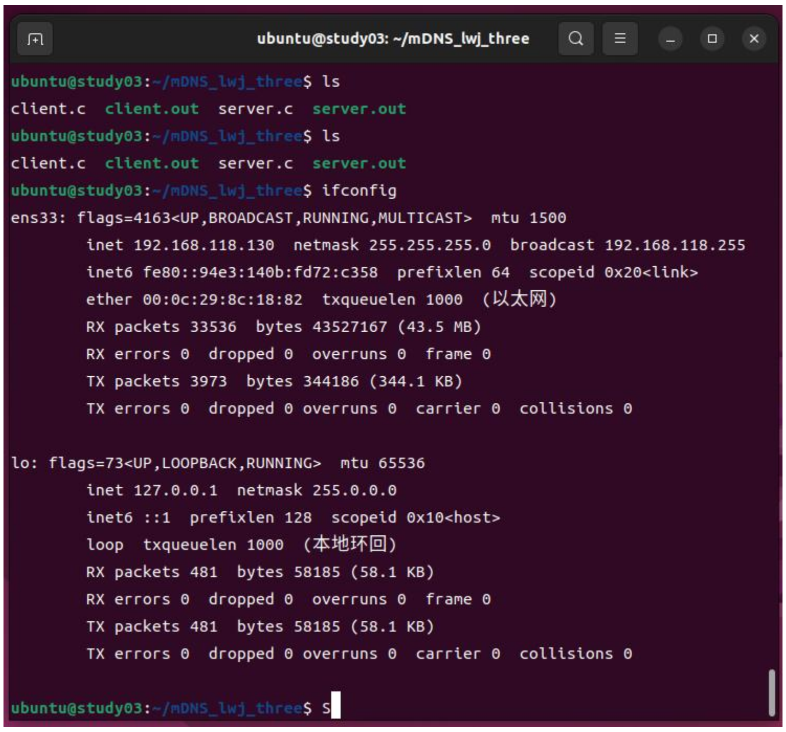
1.1 简介

利用两台虚拟机模拟两台局域网设备，第一台模拟器主机名 study03 运行服务端，第二台模拟器主机名 study02 运行客户端。客户端输出服务端主机名（study03.local）、IPv4（192.168.118.130）、IPv6 地址（fe80::94e3:140b:fd72:c358）、服务器名称（MQTTServer）、类型（_mqtt._tcp）、地址（mqtts://tb.com）。

1.2 配置环境

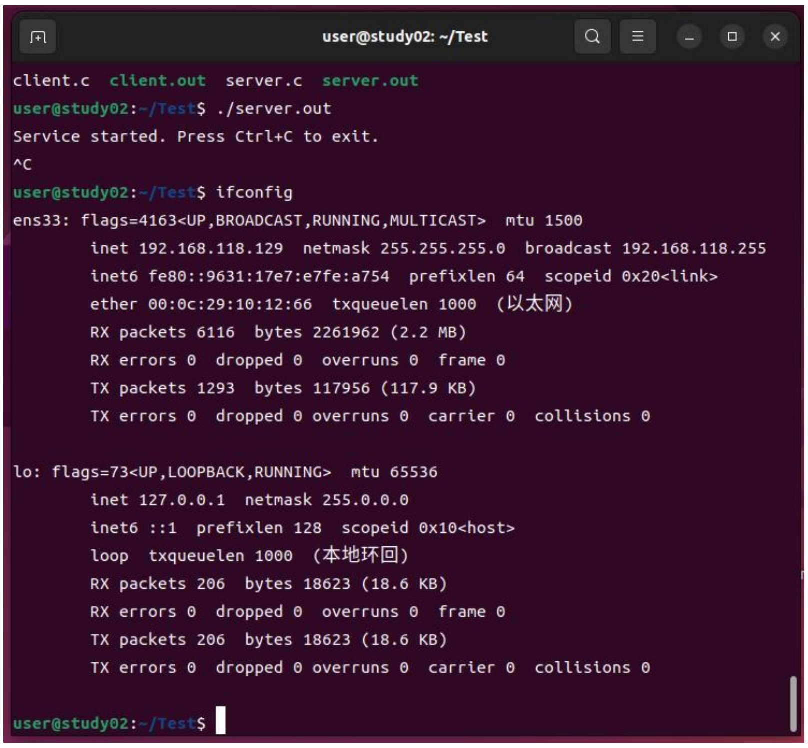
①确保虚拟机在同一局域网



```
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.118.130  netmask 255.255.255.0  broadcast 192.168.118.255
        inet6 fe80::94e3:140b:fd72:c358  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8c:18:82  txqueuelen 1000  (以太网)
        RX packets 33536  bytes 43527167 (43.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3973  bytes 344186 (344.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (本地环回)
        RX packets 481  bytes 58185 (58.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 481  bytes 58185 (58.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@study03:~/mDNS_lwj_three$ 
```



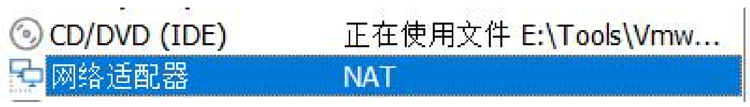
```
user@study02: ~/Test
client.c  client.out  server.c  server.out
user@study02:~/Test$ ./server.out
Service started. Press Ctrl+C to exit.
^C
user@study02:~/Test$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.118.129  netmask 255.255.255.0  broadcast 192.168.118.255
        inet6 fe80::9631:17e7:e7fe:a754  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:10:12:66  txqueuelen 1000  (以太网)
        RX packets 6116  bytes 2261962 (2.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1293  bytes 117956 (117.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (本地环回)
        RX packets 206  bytes 18623 (18.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 206  bytes 18623 (18.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

user@study02:~/Test$
```

根据 IP 地址可以确定两台虚拟机在同一个子网。

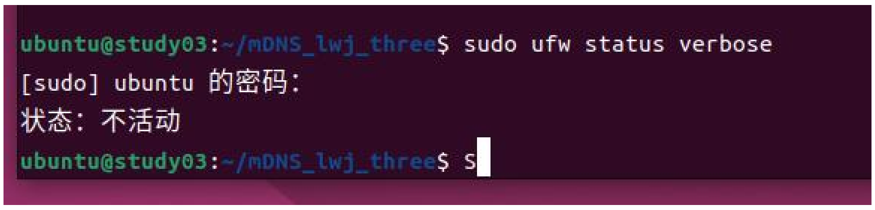
②确认虚拟机的网络模式



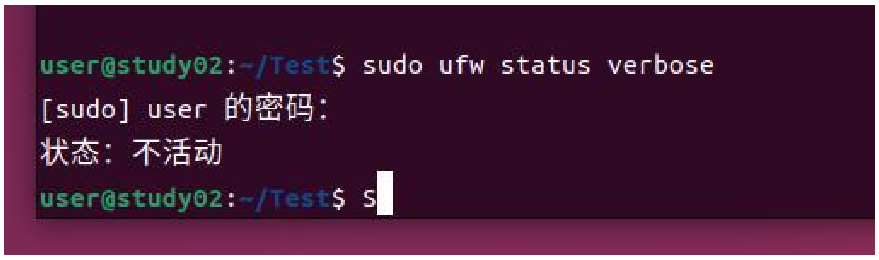
CD/DVD (IDE)	正在使用文件 E:\Tools\Vmw...
网络适配器	NAT

两台虚拟机均为 NAT 模式可以通信。

③确保防火墙没有阻止 UDP 5353 端口



```
ubuntu@study03:~/mDNS_lwj_three$ sudo ufw status verbose
[sudo] ubuntu 的密码:
状态: 不活动
ubuntu@study03:~/mDNS_lwj_three$ S
```

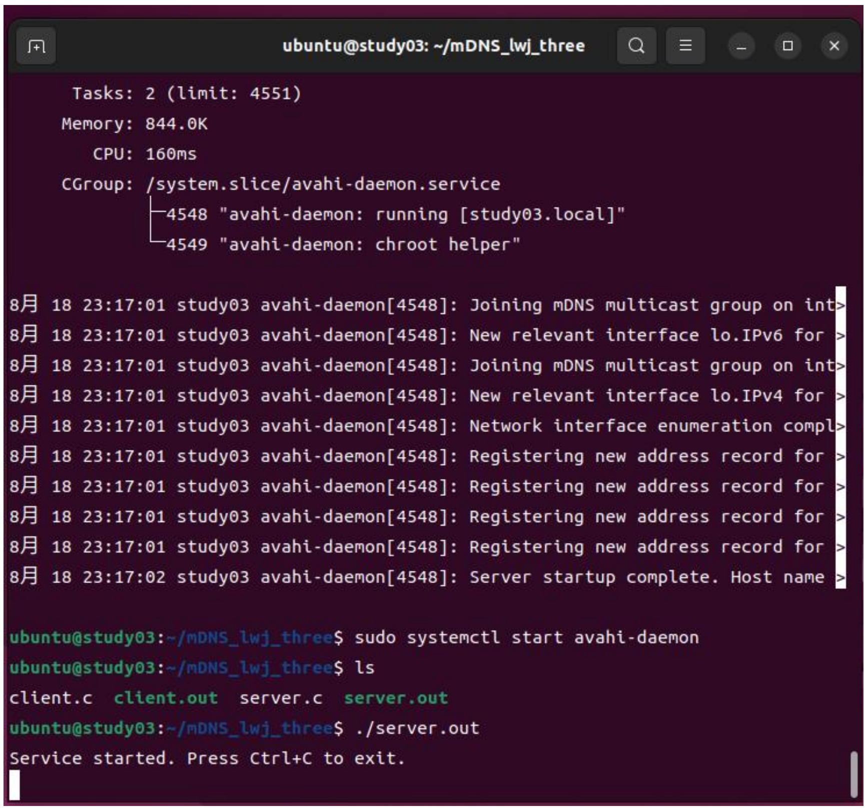


```
user@study02:~/Test$ sudo ufw status verbose
[sudo] user 的密码:
状态: 不活动
user@study02:~/Test$ S
```

两台虚拟机均关闭防火墙，如果开启可以用命令 sudo ufw allow 5353/udp 设置允许 5353 端口。

1.3 运行结果

开启 avahi 守护进程，检查状态（是否开启），第一台虚拟机运行服务端。



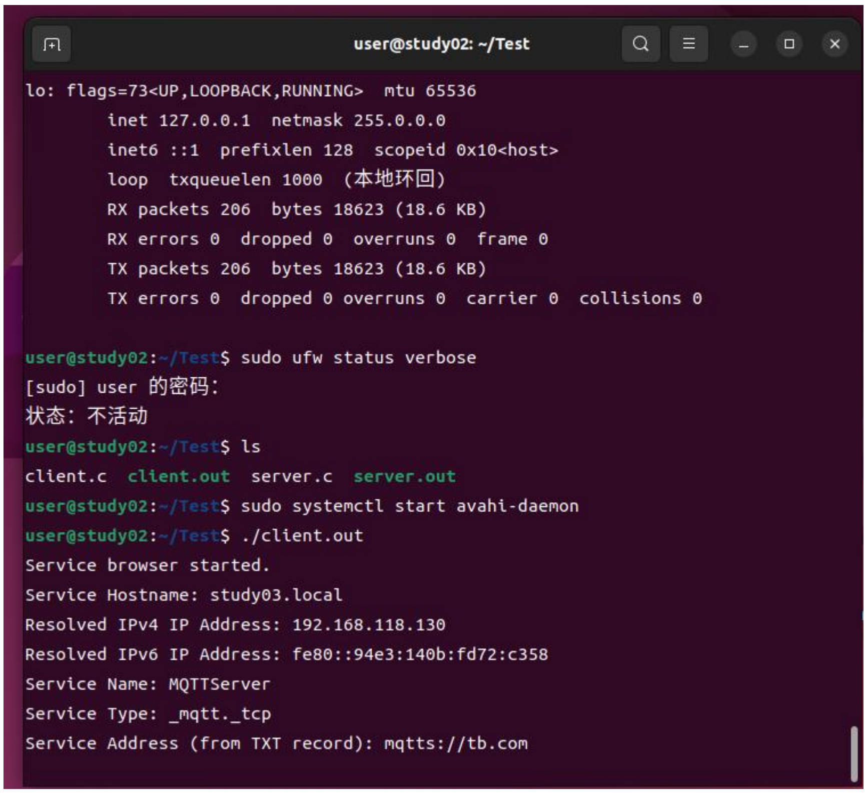
ubuntu@study03: ~/mDNS_lwj_three

```
      Tasks: 2 (limit: 4551)
     Memory: 844.0K
        CPU: 160ms
     CGroup: /system.slice/avahi-daemon.service
             ├─4548 "avahi-daemon: running [study03.local]"
             └─4549 "avahi-daemon: chroot helper"

8月 18 23:17:01 study03 avahi-daemon[4548]: Joining mDNS multicast group on int>
8月 18 23:17:01 study03 avahi-daemon[4548]: New relevant interface lo.IPv6 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Joining mDNS multicast group on int>
8月 18 23:17:01 study03 avahi-daemon[4548]: New relevant interface lo.IPv4 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Network interface enumeration compl>
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:02 study03 avahi-daemon[4548]: Server startup complete. Host name >

ubuntu@study03:~/mDNS_lwj_three$ sudo systemctl start avahi-daemon
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ./server.out
Service started. Press Ctrl+C to exit.
```

开启 avahi 守护进程，检查状态，第二台虚拟机运行客户端，输出服务端信息。



user@study02: ~/Test

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (本地环回)
        RX packets 206  bytes 18623 (18.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 206  bytes 18623 (18.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

user@study02:~/Test$ sudo ufw status verbose
[sudo] user 的密码:
状态: 不活动
user@study02:~/Test$ ls
client.c  client.out  server.c  server.out
user@study02:~/Test$ sudo systemctl start avahi-daemon
user@study02:~/Test$ ./client.out
Service browser started.
Service Hostname: study03.local
Resolved IPv4 IP Address: 192.168.118.130
Resolved IPv6 IP Address: fe80::94e3:140b:fd72:c358
Service Name: MQTTServer
Service Type: _mqtt._tcp
Service Address (from TXT record): mqtts://tb.com
```

1.4 程序

//服务端

```c
#include <avahi-client/client.h>
#include <avahi-client/publish.h>
#include <avahi-common/thread-watch.h>
#include <avahi-common/error.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <ifaddrs.h>
#include <arpa/inet.h>

int main() {
    // 创建 AvahiThreadedPoll 实例
    AvahiThreadedPoll *poll = avahi_threaded_poll_new();
    if(!poll) {
        fprintf(stderr, "Failed to create threaded poll.\n");
        return EXIT_FAILURE;
    }

    // 初始化 Avahi 客户端
    AvahiClient *client = avahi_client_new(avahi_threaded_poll_get(poll), AVAHI_CLIENT_NO_FAIL, NULL, NULL, NULL);

    if(!client) {
        fprintf(stderr, "Failed to create Avahi client.\n");
        avahi_threaded_poll_free(poll);  // 释放线程池
        return EXIT_FAILURE;
    }

    // 创建服务
    AvahiEntryGroup *group = avahi_entry_group_new(client, NULL, NULL);

    // 修改服务名称和类型为 mqtt
    const char *service_name = "MQTTServer";
    const char *service_type = "_mqtt._tcp";
    const char *mqtt_server_address = "mqtts://tb.com";

    // 获取本机的 IP 地址
    struct ifaddrs *ifaddr, *ifa;
    char ipv4_addr[INET_ADDRSTRLEN] = "";
    char ipv6_addr[INET6_ADDRSTRLEN] = "";

    if(getifaddrs(&ifaddr) == -1) {
        perror("getifaddrs");
        return EXIT_FAILURE;
    }

    for(ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
        if(ifa->ifa_addr == NULL) continue;

        int family = ifa->ifa_addr->sa_family;
        if(family == AF_INET) {
            inet_ntop(AF_INET, &((struct sockaddr_in *)ifa->ifa_addr)->sin_addr, ipv4_addr, INET_ADDRSTRLEN);
        }
        else if(family == AF_INET6) {
            inet_ntop(AF_INET6, &((struct sockaddr_in6 *)ifa->ifa_addr)->sin6_addr, ipv6_addr, INET6_ADDRSTRLEN);
        }
    }
    freeifaddrs(ifaddr);

    // 创建 TXT 记录
    AvahiStringList *txt = NULL;
    txt = avahi_string_list_add_pair(txt, "mqtt_server_address", mqtt_server_address);
    if(strlen(ipv4_addr) > 0) txt = avahi_string_list_add_pair(txt, "ipv4", ipv4_addr);
    if(strlen(ipv6_addr) > 0) txt = avahi_string_list_add_pair(txt, "ipv6", ipv6_addr);

    if(avahi_entry_group_add_service_strlst(group, AVAHI_IF_UNSPEC, AVAHI_PROTO_UNSPEC, 0, service_name, service_type, NULL, NULL, 0, txt) < 0) {
        fprintf(stderr, "Failed to add service TXT record: %s\n", avahi_strerror(avahi_client_errno(client)));
        avahi_client_free(client);
        avahi_threaded_poll_free(poll);
        return EXIT_FAILURE;
    }

       // 提交服务
    if(avahi_entry_group_commit(group) < 0) {
        fprintf(stderr, "Failed to commit entry group: %s\n", avahi_strerror(avahi_client_errno(client)));
        avahi_client_free(client);
        avahi_threaded_poll_free(poll);  // 释放线程池
        return EXIT_FAILURE;
    }

    // 启动事件循环
    avahi_threaded_poll_start(poll);

    // 让程序持续运行
    printf("Service started. Press Ctrl+C to exit.\n");
    while (1) {
        pause();  // 等待信号
    }

    // 释放资源
    avahi_entry_group_free(group);
    avahi_client_free(client);
    avahi_threaded_poll_free(poll);

    return EXIT_SUCCESS;
}
```

//客户端

```c
#include <avahi-client/client.h>
#include <avahi-client/lookup.h>
#include <avahi-common/thread-watch.h>
#include <avahi-common/error.h>
#include <avahi-common/malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>

// 标志位，用于确保只打印一次服务信息
int service_printed = 0;

int ipv4_printed = 0;   // IPv4 解析完成标志
int ipv6_printed = 0;   // IPv6 解析完成标志

void resolve_callback(AvahiServiceResolver *resolver, AvahiIfIndex interface, AvahiProtocol protocol, AvahiResolverEvent event,
    const char *name, const char *type, const char *domain, const char *hostname,
    const AvahiAddress *address, uint16_t port, AvahiStringList *txt, AvahiLookupResultFlags flags, void *userdata) {
    if(event == AVAHI_RESOLVER_FOUND) {
        char address_str[AVAHI_ADDRESS_STR_MAX];
        avahi_address_snprint(address_str, sizeof(address_str), address);

        if(protocol == AVAHI_PROTO_INET && !ipv4_printed) {
            printf("Service Hostname: %s\n", hostname);
            printf("Resolved IPv4 IP Address: %s\n", address_str);
            ipv4_printed = 1;
        }
        else if(protocol == AVAHI_PROTO_INET6 && !ipv6_printed) {
            printf("Resolved IPv6 IP Address: %s\n", address_str);
            ipv6_printed = 1;
        }

        // 只有在两种地址都打印后，才打印服务信息并设置标志位
        if(ipv4_printed && ipv6_printed && !service_printed) {
            printf("Service Name: %s\n", name);
            printf("Service Type: %s\n", type);

            while(txt) {
                char *key, *value;
                avahi_string_list_get_pair(txt, &key, &value, NULL);
                if(strcmp(key, "mqtt_server_address") == 0) {
                    printf("Service Address (from TXT record): %s\n", value);
                }
                avahi_free(key);
                avahi_free(value);
                txt = avahi_string_list_get_next(txt);
            }
            service_printed = 1;
        }
    }

    avahi_service_resolver_free(resolver);
}

void browse_callback(AvahiServiceBrowser *browser, AvahiIfIndex interface, AvahiProtocol protocol, AvahiBrowserEvent event,
    const char *name, const char *type, const char *domain, AvahiLookupResultFlags flags, void *userdata) {
    AvahiClient *client = userdata;
    if(event == AVAHI_BROWSER_NEW) {
        avahi_service_resolver_new(client, interface, AVAHI_PROTO_INET, name, type, domain, AVAHI_PROTO_UNSPEC, 0, resolve_callback, client);
        avahi_service_resolver_new(client, interface, AVAHI_PROTO_INET6, name, type, domain, AVAHI_PROTO_UNSPEC, 0, resolve_callback, client);
    }
}


int main() {
    AvahiThreadedPoll *poll = avahi_threaded_poll_new();
    if(!poll) {
        fprintf(stderr, "Failed to create threaded poll.\n");
        return EXIT_FAILURE;
    }

    AvahiClient *client = avahi_client_new(avahi_threaded_poll_get(poll), AVAHI_CLIENT_NO_FAIL, NULL, NULL, NULL);
    if(!client) {
        fprintf(stderr, "Failed to create Avahi client.\n");
        avahi_threaded_poll_free(poll);
        return EXIT_FAILURE;
    }

    AvahiServiceBrowser *browser = avahi_service_browser_new(client, AVAHI_IF_UNSPEC, AVAHI_PROTO_UNSPEC, "_mqtt._tcp", NULL, 0, browse_callback, client);
    if(!browser) {
        fprintf(stderr, "Failed to create service browser.\n");
        avahi_client_free(client);
        avahi_threaded_poll_free(poll);
        return EXIT_FAILURE;
    }

    printf("Service browser started.\n");

    avahi_threaded_poll_start(poll);

    while(1) {
        pause();
    }

    avahi_service_browser_free(browser);
    avahi_client_free(client);
    avahi_threaded_poll_free(poll);

    return EXIT_SUCCESS;
}
```