

OpenWrt实验：环境部署与程序开发

一、实验要求

1. 自主完成 OpenWrt 操作系统虚拟机的安装部署，并在虚拟机上开展本次实验。
2. 运用 C 语言进行编程开发，实现流量监控功能。需要获取的数据包括但不限于：
源IP地址/目的IP地址
累计接收(发送)流量
流量峰值
过去某个时间段的平均流量，例：过去2s、过去10s、过去40s
3. 加分项：
 - a. 开发前端可视化界面与后端服务器接口。通过发送 HTTP 请求获取流量监控的实时数据 信息，并将其直观呈现于前端界面，以实现数据的实时监测与可视化展示。
 - b. 购买路由器固件，为其烧制openwrt操作系统，在真实的路由器环境中部署流量监控程序

二、实验任务

1. 在虚拟机上安装OpenWrt，并成功运行；
2. 在Ubuntu系统上配置OpenWrt开发环境，编译HelloWorld软件包，并在OpenWrt上运行，用固件编译和SDK编译两种方式实现，在报告中说明他们的区别；
3. 编程实现流量监控软件包，并在虚拟机中运行；
4. 在路由器硬件上安装OpenWrt，并成功运行系统和开发的流量监控软件包。

三、实验软件

虚拟机：VMWare Workstation

镜像转换工具：StarWind V2V Converter

远程控制工具：Xftp(文件传输)、Xshell(命令终端)、WinSCP(文件传输)

四、VMWare Workstation 中安装 OpenWrt

镜像下载

从 OpenWrt 的镜像网站上下载24.10的openwrt操作系统镜像。以64位的x86架构系统为例，下载地址为：<https://archive.openwrt.org/releases/24.10.0/targets/x86/64/>

Image Files

These are the image files for the **x86/64** target. Check that the sha256sum of the file you downloaded matches the sha256sum below. Shortened image file names below have the same prefix: `openwrt-24.10.0-x86-64-`...

Image for your Device	sha256sum	File Size	Date
generic-ext4-combined-efi.img.gz	b0f3ba38bd9d0274fcf7868c02704eaa2c2caee62629b22828f6543dc27d6092	13486.0 KB	Tue Feb 4 13:22:56 2025
generic-ext4-combined.img.gz	a873d9f2e0e667d03dd06f75ed1f1b580beb7ce4337d5a44b60911a266965fb5	13295.7 KB	Tue Feb 4 13:22:58 2025
generic-ext4-rootfs.img.gz	1253a1695c609ce42b8d0a2f34d36bcf51338e8042c4c865344130d5c71b5ae2	7297.5 KB	Tue Feb 4 13:22:56 2025
generic-kernel.bin	2a0deaeab7dd3edf23c68597e1c79e0bd0f1ad2381cc90b3abd0187e96f28fe	5605.0 KB	Tue Feb 4 13:22:18 2025
generic-squashfs-combined-efi.img.gz	0c0e48fd2739093e0ebc8459876bed11b355cf132de2dd2a5cacd3dea890801	12080.4 KB	Tue Feb 4 13:22:57 2025
generic-squashfs-combined.img.gz	099e62e393dee1cc19123808dea3eaf2f0ffa0e8efaa12ba54bbfa1193812186	11889.7 KB	Tue Feb 4 13:22:56 2025
generic-squashfs-rootfs.img.gz	363a441352bc69540bf34e862d291bda3d574214621c440ba0fb9a478ab3459e	5891.1 KB	Tue Feb 4 13:22:56 2025
rootfs.tar.gz	25c1e8682bd3c4950be6480d859257bb5c86d5b3658982443141986fe64d93	4465.4 KB	Tue Feb 4 13:22:21 2025

Supplementary Files

These are supplementary resources for the **x86/64** target. They include build tools, the imagebuilder, sha256sum, GPG signature file, and other useful files.

Filename	sha256sum	File Size	Date
kmods/	-	-	Tue Feb 4 13:30:02 2025
packages/	-	-	Tue Feb 4 13:29:54 2025
config.buildinfo	dc432aeb9ca85bae0df6e27a1cce65a52975762eb2c53dde0ffcd73aaf5a151	2.2 KB	Tue Feb 4 13:27:45 2025
feeds.buildinfo	c7d97cd938d1e7407bb29d96ee29e6355552951d6ad23ba0ebff257cf59cd57	0.4 KB	Tue Feb 4 13:27:45 2025
kernel-debug.tar.zst	17a689d954c638ccb0a00a8827b9a24c8dc0868e71aae43b13a619f445d4a380	123815.5 KB	Tue Feb 4 13:22:18 2025
lvm-bpf-18.1.7.Linux-x86_64.tar.zst	1ef9d5f408283908b93b0551445894a046253083d305be363de23beef367a5c	45836.9 KB	Tue Feb 4 13:24:11 2025
openwrt-24.10.0-x86-64.bom.cdx.json	ea38f0b0cd9c870cd3e65b5a8bc71d9a3cdab79d6f6313e16229257136068713	15.6 KB	Tue Feb 4 13:22:59 2025
openwrt-24.10.0-x86-64.manifest	2d473f5565378ba396776c3e8f9a3b908cd1fec10f5d567636402f5709b3ecf1	3.8 KB	Tue Feb 4 13:22:58 2025
openwrt-imagebuilder-24.10.0-x86-64.Linux-x86_64.tar.zst	4e47c644057f9e7b62d14c89cbb505549644819f307bd950d60cb773a7e2098a	42968.2 KB	Tue Feb 4 13:26:07 2025
openwrt-sdk-24.10.0-x86-64_gcc-13.3.0_musl.Linux-x86_64.tar.zst	94f55f15599b57e3b341681d6deffa8e60ce38fc829e0ed14ad5c4f4a4d85a4dc	252789.0 KB	Tue Feb 4 13:27:11 2025
openwrt-toolchain-24.10.0-x86-64_gcc-13.3.0_musl.Linux-x86_64.tar.zst	2bcac43e3420630d16dbd9a393192509af5a295f5c5db32ff521aa5447b33670	68572.9 KB	Tue Feb 4 13:23:56 2025
profiles.json	36799a7465fe63e5f8612be48d26c24f3e062650800be46f79d7304f8cbea20b	2.5 KB	Tue Feb 4 13:28:09 2025
sha256sums	-	181.4 KB	Tue Feb 4 13:29:46 2025
sha256sums.asc	-	0.3 KB	Tue Feb 4 13:29:54 2025
sha256sums.sig	-	0.1 KB	Tue Feb 4 13:29:54 2025
version.buildinfo	47914cae8bab8a1c928fce61317425390cd9e5bd4320559990f70002752b228a	0.0 KB	Tue Feb 4 13:27:45 2025

镜像文件

1. generic-ext4-combined-efi.img.gz

- 用途：**适用于需要 UEFI 启动的设备。它是一个EXT4文件系统的组合镜像，包含了根文件系统和内核。
- 适用设备：**适合较新型的支持 UEFI 启动的设备。

2. generic-image-ext4-combined.img.gz

- 用途：**这是一个标准的EXT4 文件系统镜像，适用于没有 UEFI 启动需求的设备。它是一个完整的镜像，包含根文件系统和内核。
- 适用设备：**适合不需要 UEFI 启动的设备。

3. generic-ext4-rootfs.img.gz

- 用途：**这个镜像仅包含EXT4 文件系统的根文件系统，适合您已经拥有内核并希望单独部署根文件系统的情况。
- 适用设备：**适用于需要自行配置内核的设备，或者在已经有内核的情况下仅需要根文件系统。

4. generic-kernel.bin

- 用途：**仅包含内核文件，不包括根文件系统。您可以将其与其他根文件系统镜像一起使用。
- 适用设备：**适用于只需要内核文件而不需要其他文件系统的设备。

5. generic-squashfs-combined-efi.img.gz

- 用途：**适用于UEFI 启动的设备，使用SquashFS文件系统。这是一个包含根文件系统和内核的组合镜像。
- 适用设备：**适合使用 SquashFS 文件系统并且支持 UEFI 启动的设备。
- 包含一个只读分区，可用于后续文件恢复

6. generic-squashfs-combined.img.gz

- **用途：**适用于没有 UEFI 启动需求的设备，使用 SquashFS 文件系统。它是一个包含根文件系统和内核的组合镜像。
- **适用设备：**适合没有 UEFI 启动需求的设备，但希望使用 SquashFS 文件系统的设备。

7. generic-squashfs-rootfs.img.gz

- **用途：**这个镜像仅包含 SquashFS 文件系统的根文件系统，适合您已经拥有内核并希望单独部署根文件系统的情况。
- **适用设备：**适用于需要自行配置内核的设备，或者在已经有内核的情况下仅需要根文件系统。

8. rootfs.tar.gz

- **用途：**这是
- **适用设备：**适用于需要自行部署根文件系统到目标设备的情况。

补充文件

这些文件包含了与 x86/64 架构的 OpenWrt 配置、构建和管理相关的附加资源。

1. **kmods/：**包含内核模块的目录。
2. **config.buildinfo：**包含 OpenWrt 镜像构建信息的文件。
3. **feeds.buildinfo：**包含 OpenWrt 使用的软件包源（feeds）的信息。
4. **kernel-debug.tar.zst：**一个压缩的内核调试符号 tar 包。
5. **llvm-bpf-18.1.7.Linux-x86-64.tar.zst：**包含用于 x86/64 架构的 BPF 相关工具的压缩包。
6. **openwrt-24.10.1-x86-64.bom.cdx.json：**一个与构建清单或内容描述相关的文件。
7. **openwrt-manifest：**列出 OpenWrt 构建组件的清单文件。
8. **openwrt-sdk-24.10.0-x86-64-linux-x86_64.tar.zst：**用于 OpenWrt 开发的 SDK（软件开发工具包）。
9. **openwrt-toolchain-24.10.1-x86-64-gcc-13.0.1-musl.Linux-x86_64.tar.zst：**用于在 x86-64 系统上编译 OpenWrt 的工具链。
10. **profiles.json：**包含 OpenWrt 构建配置文件信息的 JSON 文件。
11. **sha256sums：**列出所有文件的 SHA256 校验和文件。
12. **sha256sums.asc：**一个包含签名 SHA256 校验和的文件，用于验证文件的完整性。
13. **version.buildinfo：**包含 OpenWrt 版本信息的文件。

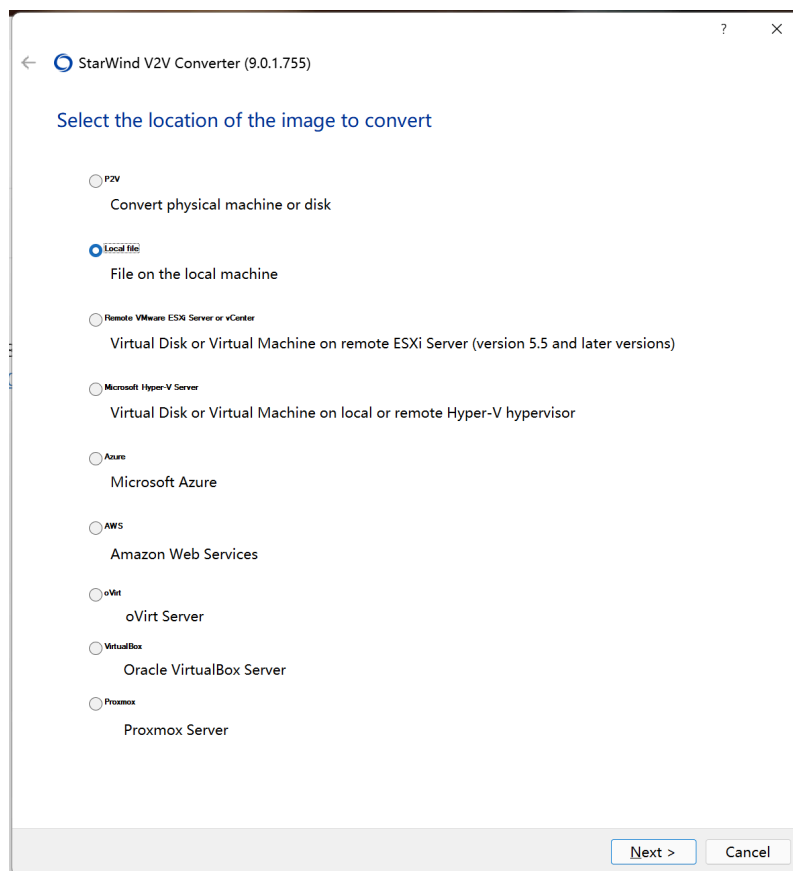
格式转换

下载得到的 img 文件无法直接用于虚拟机安装，需要将格式转换为 vmdk。

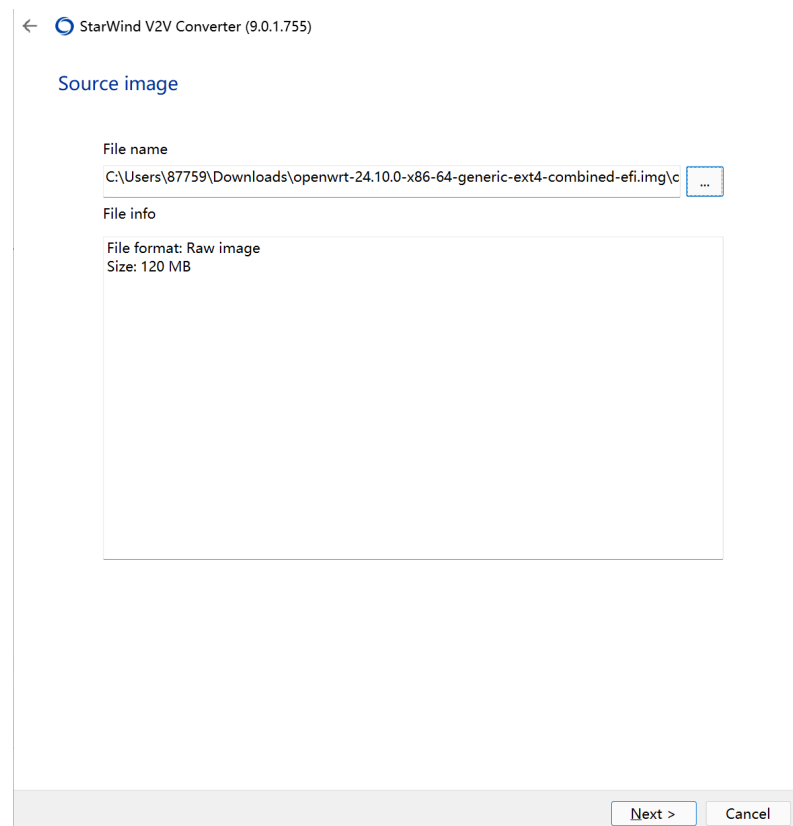
下载格式转换软件 [StarWind V2V Converter](#)

安装后打开软件

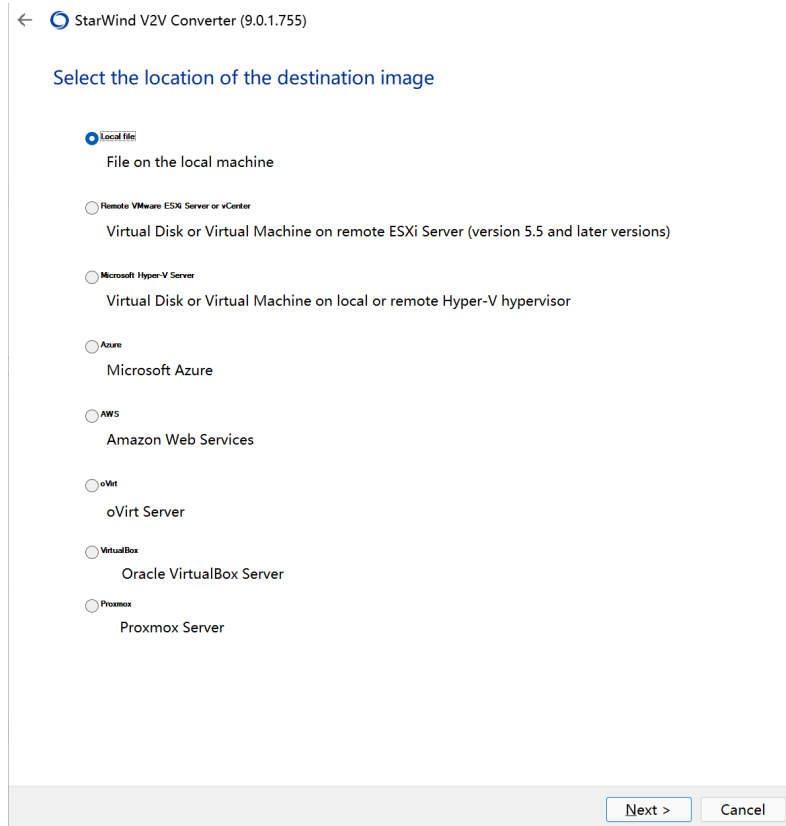
1. 选择本地文件



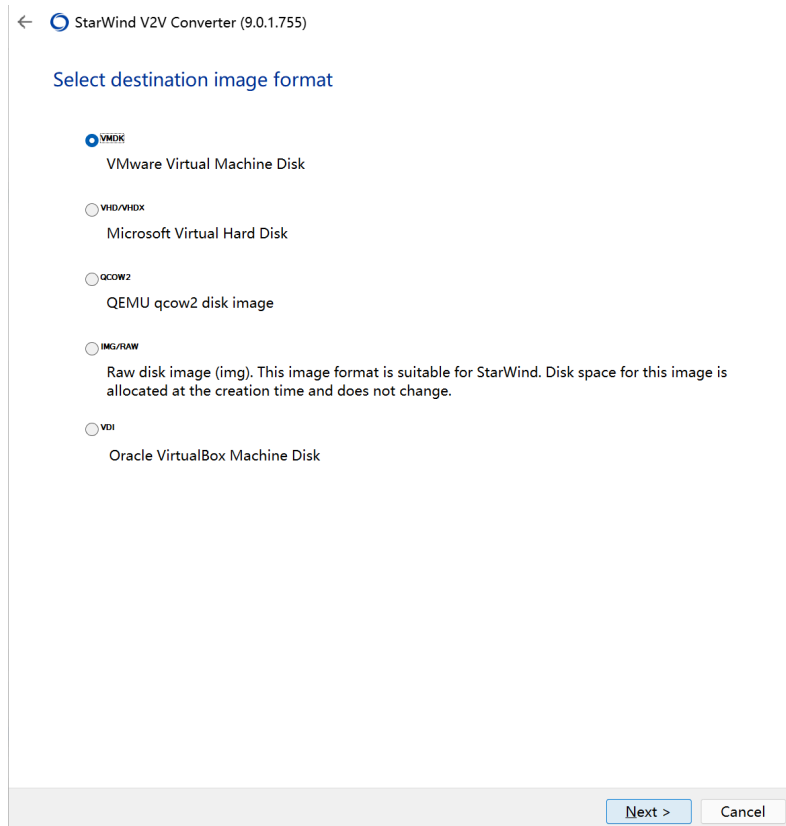
2. 选择前文中下载的镜像文件（将原文件解压后得到 img 镜像文件）



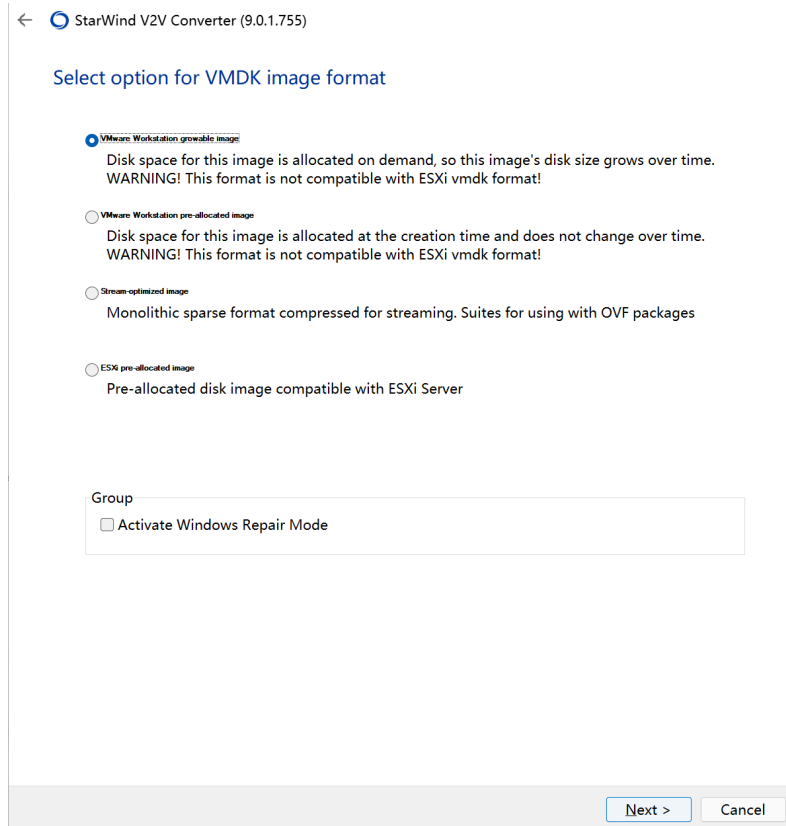
3. 选择转换为本地文件



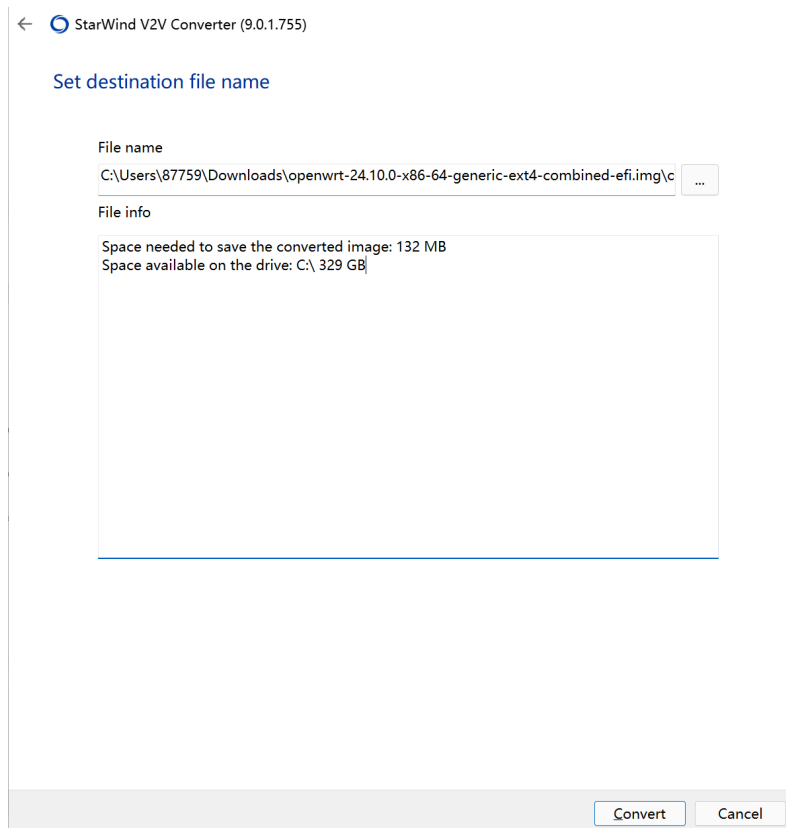
4. 选择目标格式为 VMDK




5. 选择 VMDK 格式为可变镜像




6. 确认最终路径与文件名后点击 **Convert** 即可



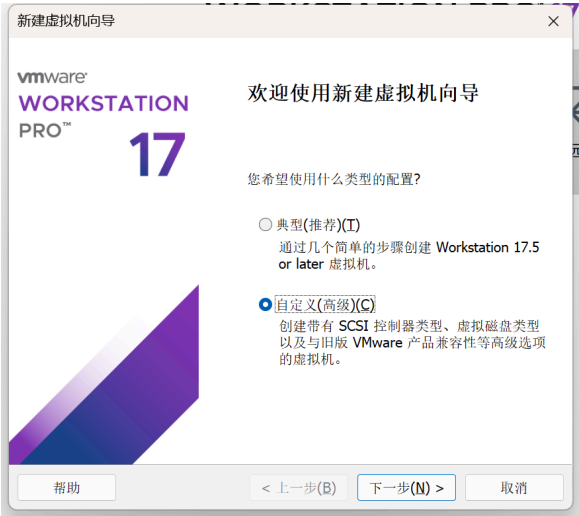
7. 最后得到对应的 VMDK 磁盘

 openwrt-24.10.0-x86-64-generic-ext4-combined-efi.vmdk

 openwrt-24.10.0-x86-64-generic-ext4-combined-efi.img

虚拟机创建

1. 新建虚拟机中选择自定义选项



2. 如图操作



3. 这里选择稍后安装操作系统



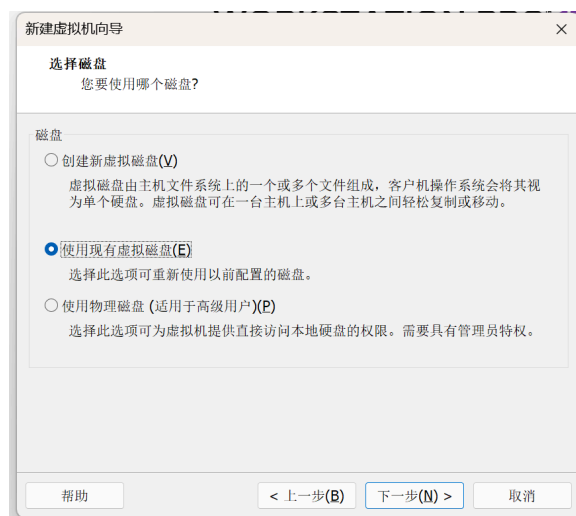
4. 选择 Linux 版本，注意对应 OpenWrt 版本选择对应的 Linux 版本，这里使用的 24.10.0 版本对应 Linux 6.x 内核版本



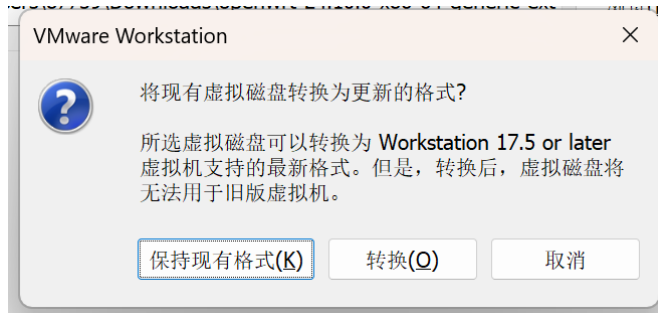
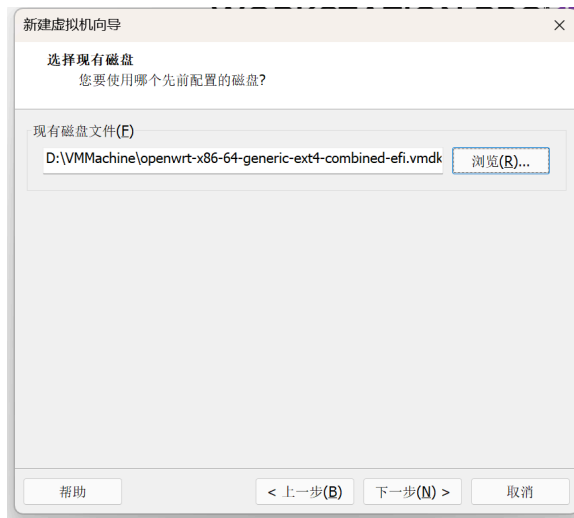
5. 虚拟机命名



6. 后续配置默认即可，在如图选择磁盘时，选择使用现有磁盘

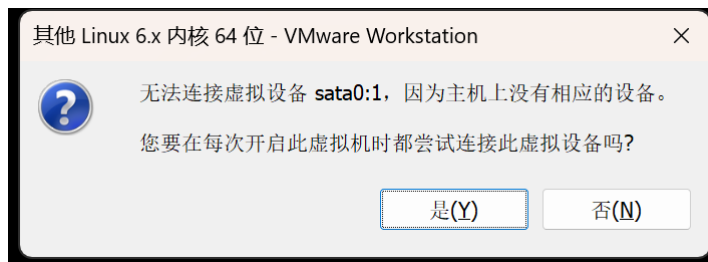


7. 选择前文中转换得到的 VMDK 磁盘，点击下一步时，选择保持现有格式即可。后续点击完成，虚拟机创建完成。



虚拟机配置

1. 启动虚拟机，可能出现如图提示。

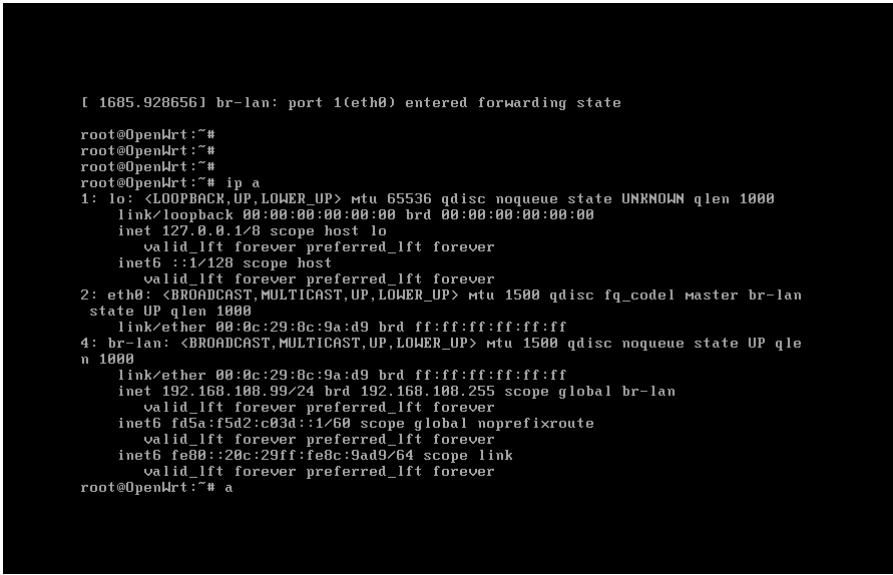


2. 如图即系统启动完成，进入虚拟机回车即出现终端界面，使用 `passwd` 指令设置用户密码

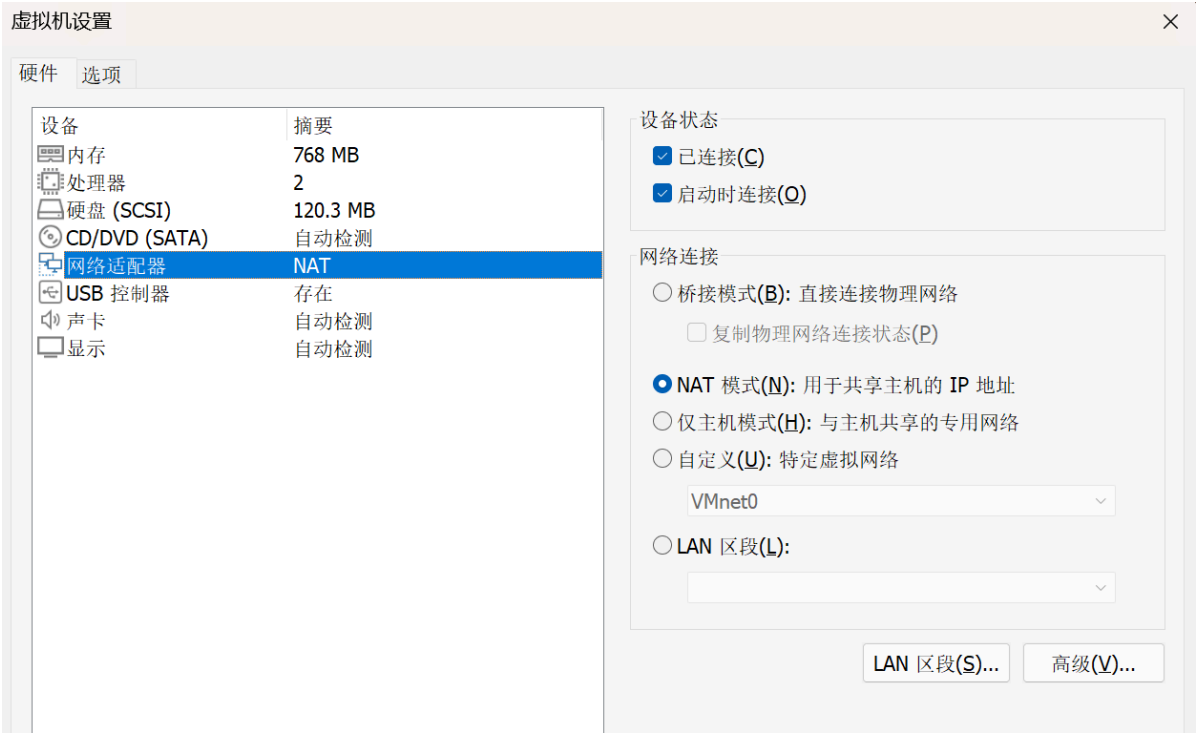
其中 `ipaddr` 和 `gateway` 需要根据虚机的配置为准，注意 `ipaddr` 不要与现有虚拟机重复。

重启网络即可生效

```
/etc/init.d/network restart
```



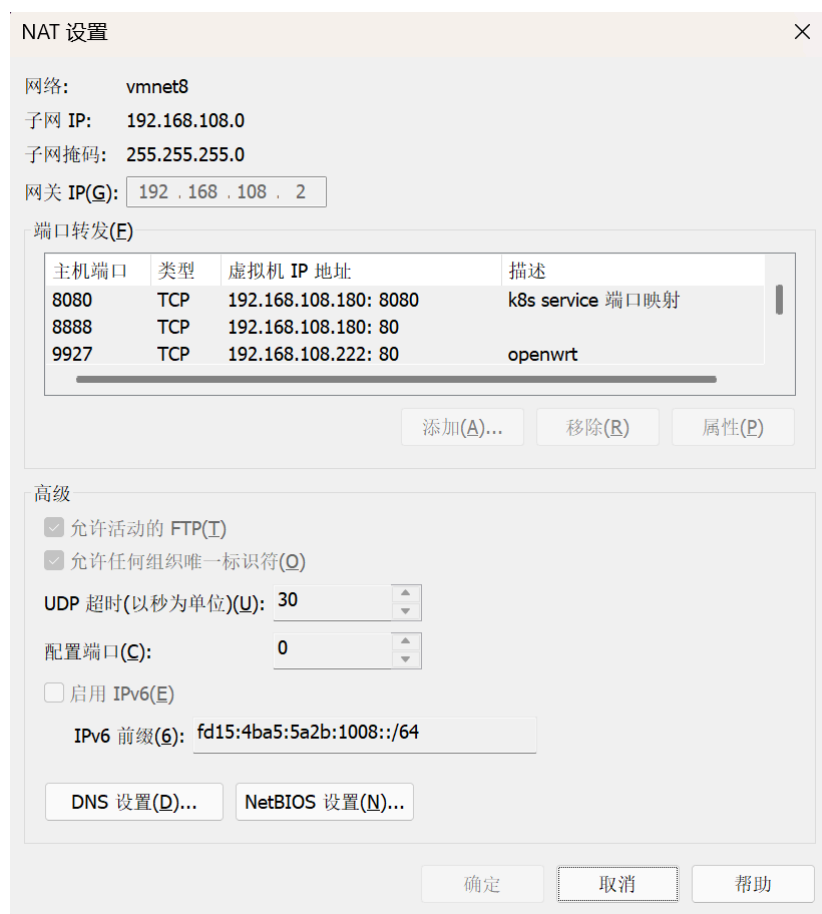
关于 `ipaddr` 和 `gateway` 具体值的设置：作为演示，点击打开 `虚拟机 > 设置` 可以看到该虚机的网络适配器采用了 NAT 模式。



点击 `编辑 > 虚拟网络编辑器` 可以看到 VMWare 当前的网络配置。点击查看使用 NAT 模式的虚拟网卡。



可以看到对应的子网 IP、子网掩码和网关 IP。



也可直接选择桥接模式

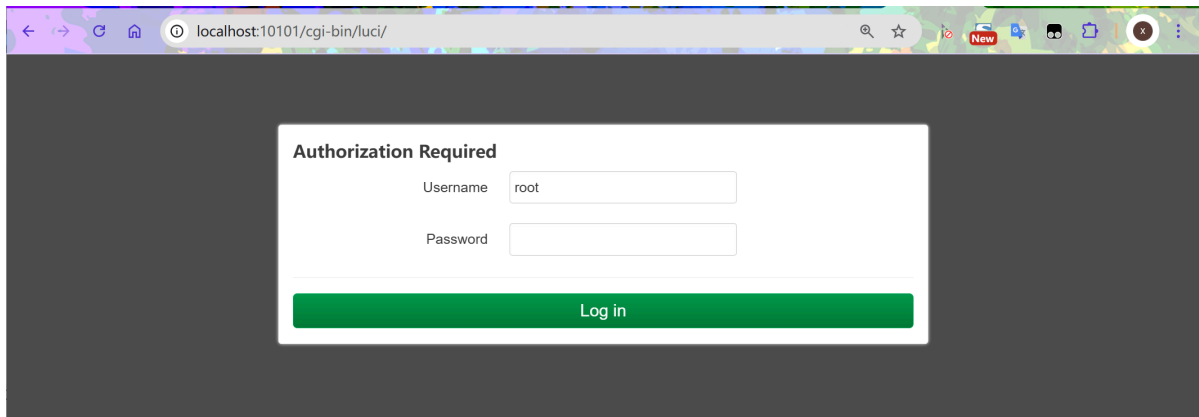
3. 设置端口转发（可选）

因为无法直接从宿主机访问虚机内部端口，所以若需要访问 OpenWrt 的管理页面，需要设置端口转发。

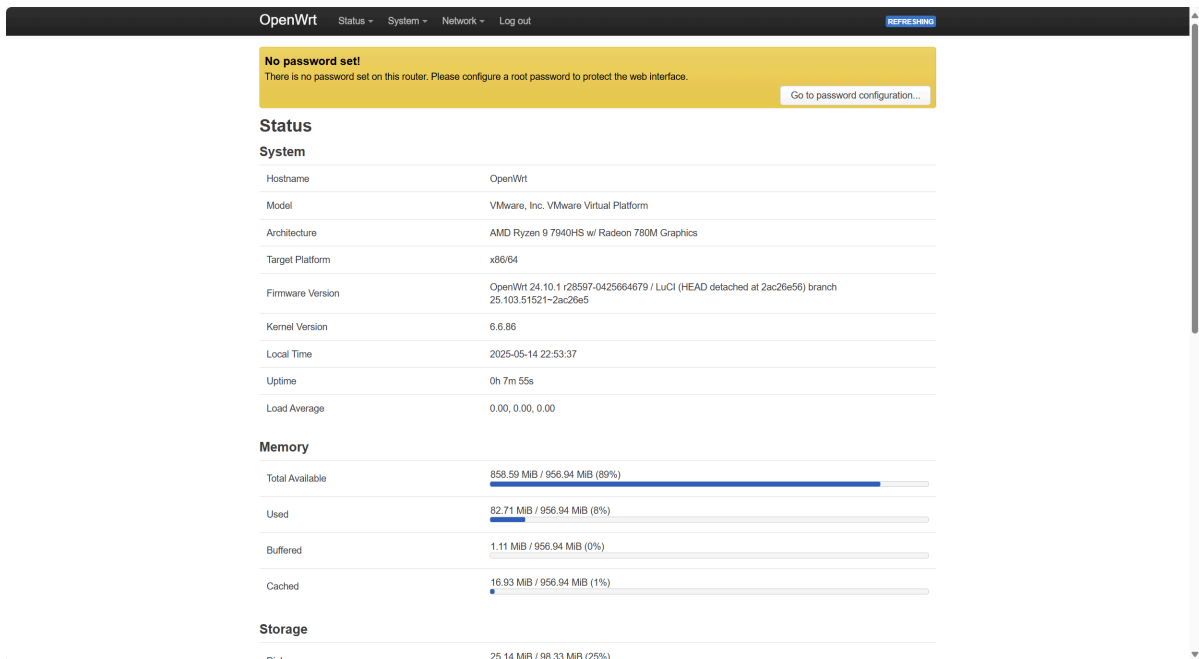
在前文中的虚拟网络编辑器中点击 更改设置，随后进入 NAT 设置。选择 添加。如图设置后确定即可，其中主机端口为后续的访问端口，虚拟机 IP 地址为刚刚在配置文件中设置的 IP 地址。



设置完成后可以在本地浏览器通过 localhost:10101 访问到管理界面



未设置密码可以直接进入Openwrt管理界面



4. 确认 SSH 服务（可选）

OpenWrt 中使用 Dropbear 来提供简单的 SSH 服务。一般来说通过 root 用户与对应的密码即可登录。可通过如下命令来检查与开启服务

```
# 检查服务状态
/etc/init.d/dropbear status

# 启动服务
/etc/init.d/dropbear start

# 设置开机自启
/etc/init.d/dropbear enable
```

五、固件编译与 SDK 使用

安装开发环境

实验采用Ubuntu，可通过WSL安装，VMWare安装

固件编译

除了直接使用官方提供的镜像外，还可以通过源码自行编译镜像。此处使用 WSL 作为演示的编译环境。

```
# 安装编译所需的工具与依赖
sudo apt update
sudo apt install build-essential clang flex bison g++ gawk \
gcc-multilib g++-multilib gettext git libncurses5-dev libssl-dev \
python3-setuptools rsync swig unzip zlib1g-dev file wget

# 拉取OpenWrt源码
git clone https://git.openwrt.org/openwrt/openwrt.git

# 进入目录并切换至所需的版本
cd openwrt
git pull
git checkout v24.10.0

# 更新下载相关依赖
./scripts/feeds update -a
./scripts/feeds install -a

# 进行相关编译设置
make menuconfig
```

在 `make menuconfig` 中主要关注 `Target System`，`Subtarget` 和 `Target Profile` 这三项，确认可以用于期望的设备。其余选项根据自身需求选择。设置完成后，保存退出。

```
# 开始编译
# 因为使用的WSL，这里专门声明环境变量，其余环境下不需要专门声明
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin make
```

等待编译完成后在 `./bin/target` 下可以找到对应镜像。

SDK 使用

OpenWrt 的 SDK (Software Development Kit, 软件开发工具包) 是专门为开发和编译 OpenWrt 额外软件包而准备的一套集成工具和环境。SDK 包含交叉编译工具链、头文件、Makefile 框架等，让你可以在本地直接编译 OpenWrt 支持的软件包。SDK 版本和你的 OpenWrt 固件版本是配套的，编译出来的包能确保兼容你现在的系统，避免因 ABI 不一致导致的各种“无法安装”问题。

对于前文中虚拟机中的 OpenWrt 而言，可以选择该文件：

https://archive.openwrt.org/releases/24.10.0/targets/x86/64/openwrt-sdk-24.10.0-x86-64_gcc-13.3.0_musl.Linux-x86_64.tar.zst

对于输入路由器中的 OpenWrt 而言，可以选择该文件：

https://archive.openwrt.org/releases/24.10.0/targets/mediatek/filogic/openwrt-sdk-24.10.0-mediatek-filogic_gcc-13.3.0_musl.Linux-x86_64.tar.zst

在选择 SDK 时，注意确认你的目标 OpenWrt 固件版本。

示例：helloworld

以一个简单的 helloworld 程序，来演示 SDK 的使用方法，演示环境为 WSL。

首先解压 SDK 文件，选择的 SDK 用于构建用于路由器的软件包。

```
tar -I zstd -xvf openwrt-sdk-24.10.1-x86-64_gcc-13.3.0_musl.Linux-x86_64.tar.zst
```

随后进入目录，并创建 helloworld 项目

```
cd openwrt-sdk-24.10.0-mediatek-filogic_gcc-13.3.0_musl.Linux-x86_64/package/  
mkdir -p helloworld  
cd helloworld  
touch helloworld.c  
touch Makefile
```

如此得到如下的目录结构

```
package/  
├─ helloworld/  
│   └─ src/  
│       ├── helloworld.c  
│       └─ Makefile
```

接下来在文件中写入内容

```
// helloworld.c  
#include <stdio.h>  
  
int main(int argc, char **argv) {  
    printf("Hello, OpenWrt SDK!\n");  
    return 0;  
}
```

```
# makefile  
  
# 引入 OpenWrt 的全局构建规则和变量  
include $(TOPDIR)/rules.mk  
  
# 指明包名，版本与发布号  
PKG_NAME:=helloworld  
PKG_VERSION:=1.0  
PKG_RELEASE:=1
```

```

# 引入 Openwrt 的软件包构建框架，包括一系列函数和变量。
include $(INCLUDE_DIR)/package.mk

# 定义软件包元数据，用来描述包在 menuconfig 里的分类和名字。
define Package/helloworld
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=Hello World Program
endef

# 包的详细描述，显示在 make menuconfig 和 opkg info 里，用于说明包的用途。
define Package/helloworld/description
    A simple Hello world program for Openwrt SDK demo.
endef

# 编译前准备，将源码复制到编译专用的路径下
define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

# Compile阶段，指定如何编译源码。
define Build/Compile
    $(TARGET_CC) $(TARGET_CFLAGS) $(TARGET_LDFLAGS) \
        -o $(PKG_BUILD_DIR)/helloworld $(PKG_BUILD_DIR)/helloworld.c
endef

# 安装阶段，定义如何把生成的文件打包进最终的ipk包。
define Package/helloworld/install
    $(INSTALL_DIR) $(1)/usr/bin
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/helloworld $(1)/usr/bin/
endef

# Openwrt标准调用，告诉编译系统“把上面定义的 helloworld 包的所有步骤整合成一套完整的编译/打包流程”。
$(eval $(call BuildPackage,helloworld))

```

接下来回到目录 `openwrt-sdk-24.10.0-mediatek-filogic-gcc-13.3.0_musl.Linux-x86_64/` 下，执行命令

```

# 准备依赖，helloworld 可能用不上
./script/feeds update -a
./script/feeds install -a

# 编译打包
make package/helloworld/compile V=s

```

编译完成后，得到对应 ipk 包，在目录 `./bin/packages/aarch64_cortex-a53/base/` 下存在 ipk 包 `helloworld_1.0-r1_aarch64_cortex-a53.ipk`

```

# 确认在SDK根目录下
cd bin/packages/aarch64_cortex-a53/base/

```

接下来将 ipk 文件上传到 OpenWrt 中，使用 opkg 安装后执行命令即可


```
opkg install helloworld_1.0-r1_aarch64_cortex-a53.ipk
helloworld
```

示例：流量监控

myflowtraffic

SRC	DST/HOST	2s(Avg/Peak)	10s(Avg/Peak)	40s(Avg/Peak)
192.168.108.222	=> 192.168.108.1	4179.0b/4179.0 b	2871.0b/2871.0 b	0.0b/0.0 b
192.168.108.1	<= 192.168.108.222	60.0b/90.0 b	60.0b/60.0 b	0.0b/0.0 b
fe80::20c:29ff:fe49:f23c	=> ff02::16	0.0b/55.0 b	0.0b/11.0 b	0.0b/0.0 b
fe80::20c:29ff:fe49:f23c	=> ff02::1	0.0b/47.0 b	0.0b/17.4 b	0.0b/0.0 b
fe80::dcdd:fb24:b0ad:dd92	<= fe80::20c:29ff:fe49:f23c	0.0b/43.0 b	17.2b/17.2 b	0.0b/0.0 b
fe80::20c:29ff:fe49:f23c	=> fe80::dcdd:fb24:b0ad:dd92	0.0b/43.0 b	15.4b/16.4 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> www.msftconnecttest.com	103.0b/103.0 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> www.msftconnecttest.com	0.0b/41.5 b	0.0b/0.0 b	0.0b/0.0 b
8.8.8.8	=> www.msftconnecttest.com	0.0b/233.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> www.msftconnecttest.com	335.5b/335.5 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> 8.8.8.8	0.0b/276.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> ipvs.msftconnecttest.com	0.0b/42.0 b	0.0b/104.0 b	0.0b/0.0 b
8.8.8.8	=> ipvs.msftconnecttest.com	0.0b/269.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> tracking.miui.com	0.0b/289.5 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> tracking.miui.com	0.0b/38.5 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> tracking.miui.com	0.0b/38.5 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> tracking.miui.com	0.0b/48.5 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> tracking.miui.com	0.0b/38.5 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.223	=> tracking.miui.com	0.0b/38.5 b	0.0b/0.0 b	0.0b/0.0 b
8.8.8.8	=> tracking.miui.com	0.0b/194.0 b	0.0b/0.0 b	0.0b/0.0 b
114.114.114.114	=> tracking.miui.com	0.0b/157.5 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> tracking.miui.com	0.0b/107.0 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.222	=> 114.114.114.114	0.0b/74.5 b	0.0b/0.0 b	0.0b/0.0 b
114.114.114.114	=> tracking.miui.com	0.0b/157.5 b	0.0b/0.0 b	0.0b/0.0 b
8.8.8.8	=> tracking.miui.com	0.0b/194.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> tracking.miui.com	0.0b/107.0 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.223	=> 114.114.114.114	0.0b/74.5 b	0.0b/0.0 b	0.0b/0.0 b
fe80::dcdd:fb24:b0ad:dd92	=> ff02::c	0.0b/111.0 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.1	=> 255.255.255.255	1436.0b/1436.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	1047.0b/1047.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fe80::20c:29ff:fe32:3728	=> fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> fe80::20c:29ff:fe32:3728	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fe80::20c:29ff:fe49:f23c	=> fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> fe80::20c:29ff:fe49:f23c	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> ipvs.msftconnecttest.com	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
192.168.108.223	=> ipvs.msftconnecttest.com	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
114.114.114.114	=> ipvs.msftconnecttest.com	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b
fd9c:8ae3:8381:0:8d30:4fe1:6a93:6ee0	=> ipvs.msftconnecttest.com	0.0b/0.0 b	0.0b/0.0 b	0.0b/0.0 b

功能未完善

六、路由器（H3C Magic NX30 Pro）中使用 OpenWrt

固件下载

在官网（<https://firmware-selector.openwrt.org/>）下搜索设备对应的固件

 简体中文

下载适用于您设备的 OpenWrt 固件

输入设备的名称或型号，然后选择一个稳定或快照版本。

H3C Magic NX30 Pro

24.10.1

关于此构建

型号: **H3C Magic NX30 Pro**
平台: mediatek/filogic
版本: 24.10.1 (r28597-0425664679)
链接: [📄](#) [📦](#) [🔗](#)
▶ 自定义预安装软件包和/或首次启动脚本

下载映像

 BL31-UBOOT.FIP

引导程序映像。用于启动时加载操作系统的底层软件。
sha256sum: 9976bf494ea502c635fb7013fe9a80a27bbf3f008df92828751cdb86ae33804e

 KERNEL

集成最小文件系统的 Linux 内核。适用于首次安装或故障恢复。
sha256sum: 6060d495195b77e64e628742b14c3ee57f1f9886ba111c41a973d33baf21c298d

 PRELOADER.BIN




其他映像类型。
sha256sum: f99ef696a0311c172dd435c23b37e0a21482a7816913bd86e1ebbb62ccd30713

 SYSUPGRADE

使用 Sysupgrade 映像以更新现有运行 OpenWrt 的设备。该映像可以在 LuCI 界面或终端中使用。
sha256sum: b5aceae57a1b3339c6a48c644aa8a942f2881b6a3a35ec9f836c333762993369

所有文件 | 反馈 | OFS v5.0.0-2-g4336d760

下载得到如下文件，即图中1，2，4项

-  openwrt-24.10.0-mediatek-filogic-h3c_magic-nx30-pro-initramfs-recovery.itb
-  openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-bl31-uboot.fip
-  openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-squashfs-sysupgrade.itb

将文件 `openwrt-24.10.0-mediatek-filogic-h3c_magic-nx30-pro-initramfs-recovery.itb` 重命名为 `openwrt-mediatek-filogic-h3c_magic-nx30-pro-initramfs-recovery.itb`

连接路由器

该款路由器（H3C Magic NX30 Pro）默认开启了telnet服务。

首先进入路由器管理网页设置登录密码。随后通过该密码与用户名 `H3C` 使用telnet服务进入路由器后台（telnet端口为99）

开启 SSH 服务（可选）

按照以下步骤开启 SSH

```
curl -o /tmp/dropbear.ipk https://downloads.openwrt.org/releases/packages-19.07/aarch64_cortex-a53/base/dropbear_2019.78-2_aarch64_cortex-a53.ipk
opkg install /tmp/dropbear.ipk
/etc/init.d/dropbear enable
/etc/init.d/dropbear start
```

固件备份

查看分区信息

```
root@Openwrt:/dev# cat /proc/mtd
dev:   size  erasesize  name
mtd0: 00100000 00020000 "BL2"
mtd1: 00080000 00020000 "u-boot-env"
mtd2: 00200000 00020000 "Factory"
mtd3: 00200000 00020000 "FIP"
mtd4: 04000000 00020000 "ubi"
mtd5: 00600000 00020000 "pdt_data"
mtd6: 00600000 00020000 "pdt_data_1"
mtd7: 00100000 00020000 "exp"
mtd8: 02580000 00020000 "plugin"
```

分区名	作用	是否需要备份	原因
BL2	二级引导程序（Bootloader Stage 2），负责初始化硬件和加载主引导程序。	否	通常不需要修改。如果损坏，可能需要通过特定工具重新刷入。
u-boot-env	存储 U-Boot 的环境变量（如启动参数）。	否	环境变量通常可以重置或重新配置。

分区名	作用	是否需要备份	原因
Factory	存储设备出厂信息（如 MAC 地址、序列号）。	选择性	如果需要保留设备唯一信息，可以备份此分区，但通常不涉及运行时数据。
FIP	固件包，可能包含启动文件（如 ATF 或 OP-TEE）。	否	固件一般是固定的，重新烧录即可恢复。
ubi	操作系统核心组件、文件系统和驱动程序	是	该分区丢失或损坏可能导致设备变砖
pdt_data	产品数据，可能是用户数据或配置备份。	选择性	如果其中的数据对设备运行至关重要，可以备份它，但这通常不是必要的。
pdt_data_1	产品数据的备份分区（冗余分区）。	选择性	通常作为冗余备份存在，实际使用频率较低。
exp	可能是调试、测试或实验用分区。	否	通常用途有限，不需要备份。
plugin	插件数据分区，可能包含用户安装的插件或扩展功能。	选择性	如果插件可以重新安装，备份就不是必须的。

```
dd if=/dev/mtd5 of=/tmp/backup.img
```

备份后使用 scp 命令传出并删除文件，避免存储空间不足

OpenWrt 刷入

将前文下载的文件 `openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-b131-uboot.fip` 传入路由器中。推荐使用 winscp 工具。

写入 uboot

```
mtd write ./openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-b131-uboot.fip FIP
```

随后需要刷入 kernel。先将路由器断电，按住背后 Reset 按钮不放，再插电，等待 10s 左右进入 uboot。接着使用 PC 机用网线连接到路由器的 LAN 口。在网络设置中将本机的 IP 设置为 `192.168.1.x`，网关设置为 `192.168.1.1`

编辑网络 IP 设置

手动

IPv4

开

IP 地址

子网掩码

网关

首选 DNS

DNS over HTTPS

保存 取消

随后使用TFTP服务将内核上传到uboot中，可以使用 tftpd64工具，这里使用 [tftp-now](#) 工具，需要断网前提前下好。

在前文下载文件的目录下执行命令

```
.\tftp-now-windows-amd64.exe serve
```

可以看到路由器的主动请求，请求 `openwrt-mediatek-filogic-h3c_magic-nx30-pro-initramfs-recovery.itb` 文件，这是前文重命名的原因。

若出现上传失败的情况，可以关闭防火墙并再次尝试。

上传成功后，通过在浏览器中访问 `192.168.1.1` 进入 Web 界面。

最后在 `System > Backup / Flash Firmware` 中上传之前下载的 `openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-squashfs-sysupgrade.itb` 文件即可升级到对应版本的OpenWrt。
