实验环境搭建（Windows）

Linux是一个免费可用的操作系统，非常适用于操作系统开发。在操作系统实验课程中，我们使用Ubuntu 24.04 LTS 作为标准的实验环境。

本部分内容会向大家介绍如何在windows系统中搭建本课程需要使用的实验环境。

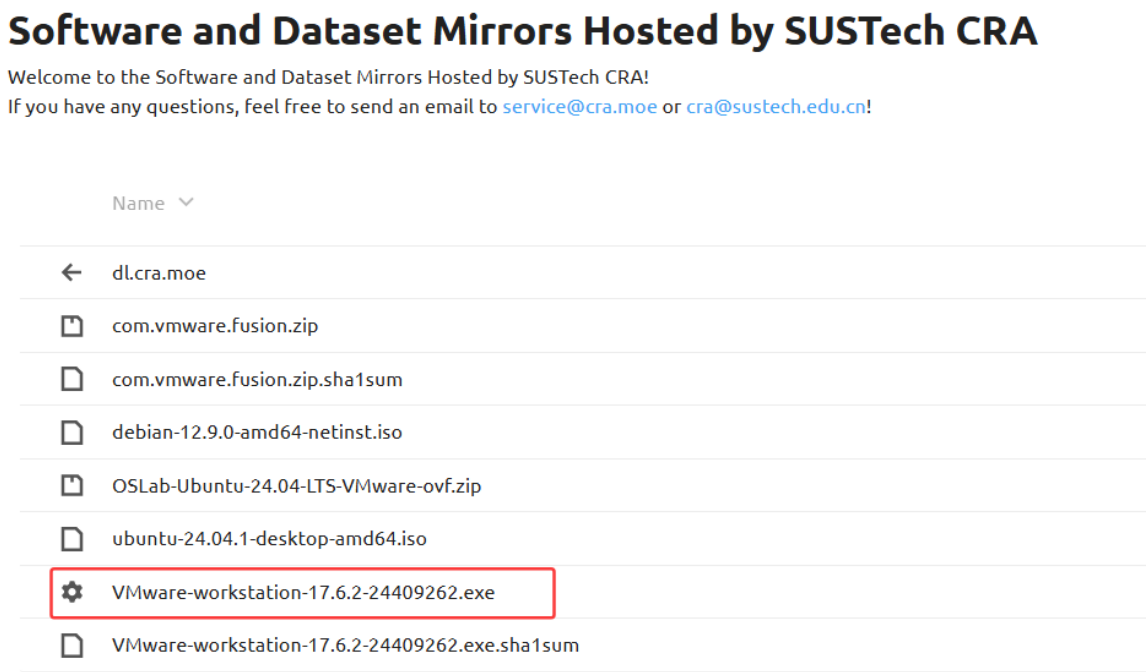
配置Ubuntu虚拟机

在这部分内容中，我们将引导同学们在windows系统中安装VMware Workstation Pro, 并在VMware中配置Ubuntu虚拟机

安装VMware Workstation Pro 16

同学们可以通过以下链接下载VMware Workstation Pro 17，并将其安装在自己的计算机中。

Download VMware Workstation Pro



安装Ubuntu24.04

接下来，关于安装Ubuntu24.04，我们提供两种选择，大家可以选择下载已经搭载实验环境的Ubuntu系统镜像导入进虚拟机，或者Ubuntu官方原版镜像。

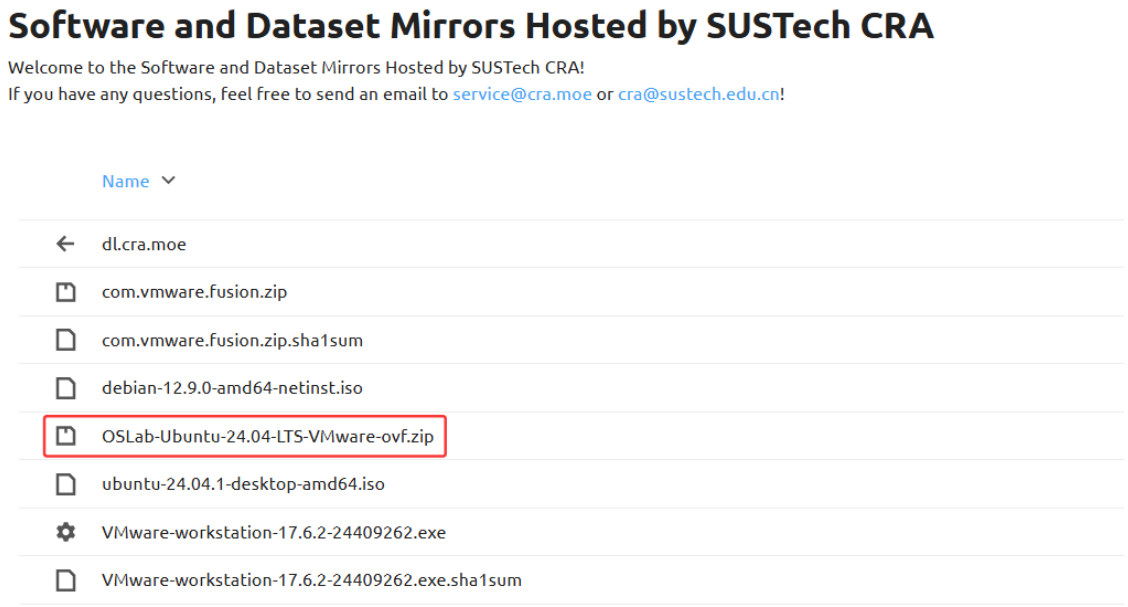
> **Warning**
>
> 以下1和2只用选择其一即可

1. 导入已搭载实验环境的系统镜像

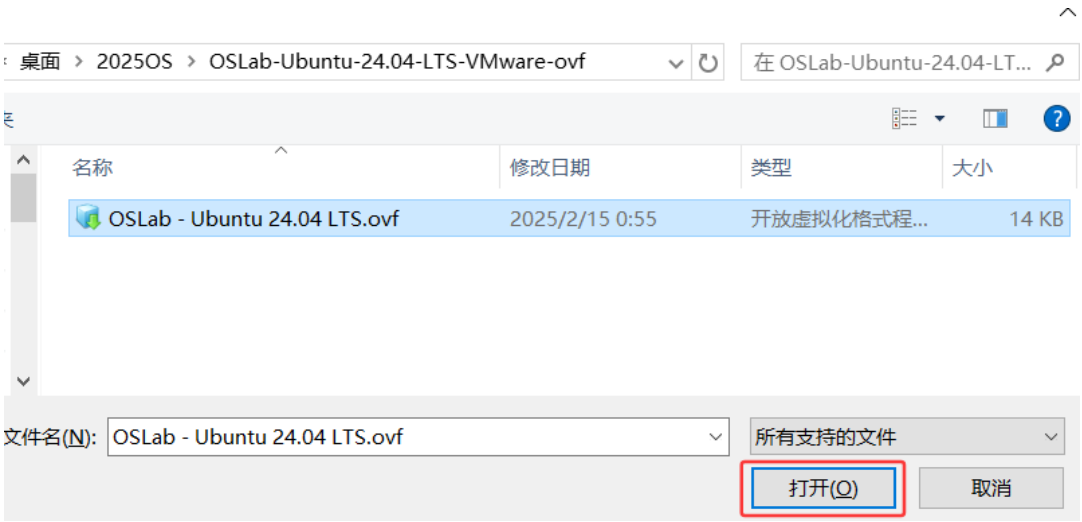
此步骤为在VMware中导入已经搭载实验环境的系统镜像。如你已经拥有一个现有环境，可参考`手动配置开发环境`的步骤在系统中配置实验环境，不过我们更加推荐使用我们提供的环境以防止系统版本导致的实验差异。

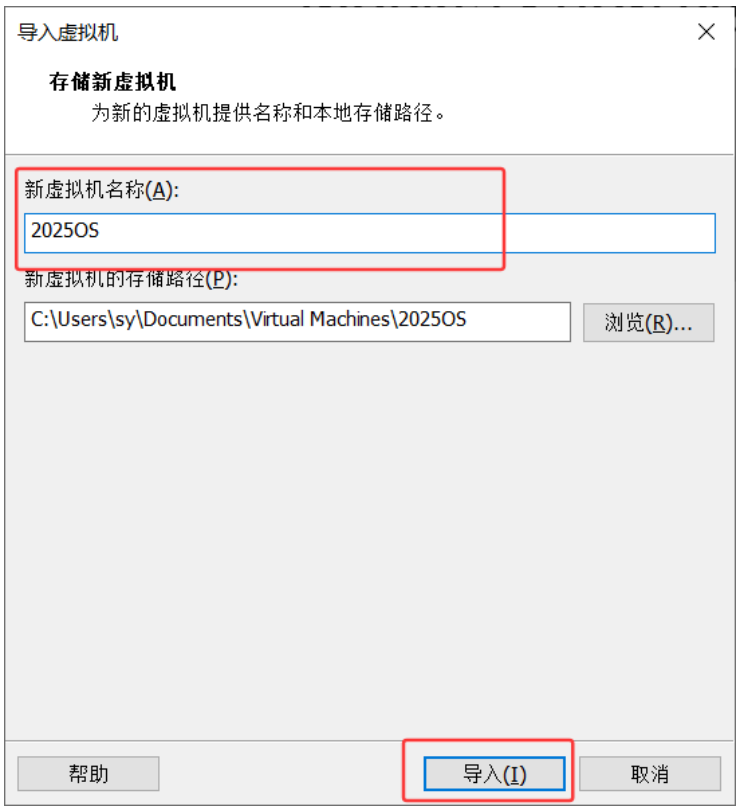
首先通过以下链接下载Ubuntu24.04的ovf格式（Open Virtualization Format）文件并进行解压。

[Download_Ubuntu_VMware_ovf]



之后，打开VMware选择`文件`->`打开`，找到ovf文件选择打开，输入虚拟机名称后等待导入完成即可。





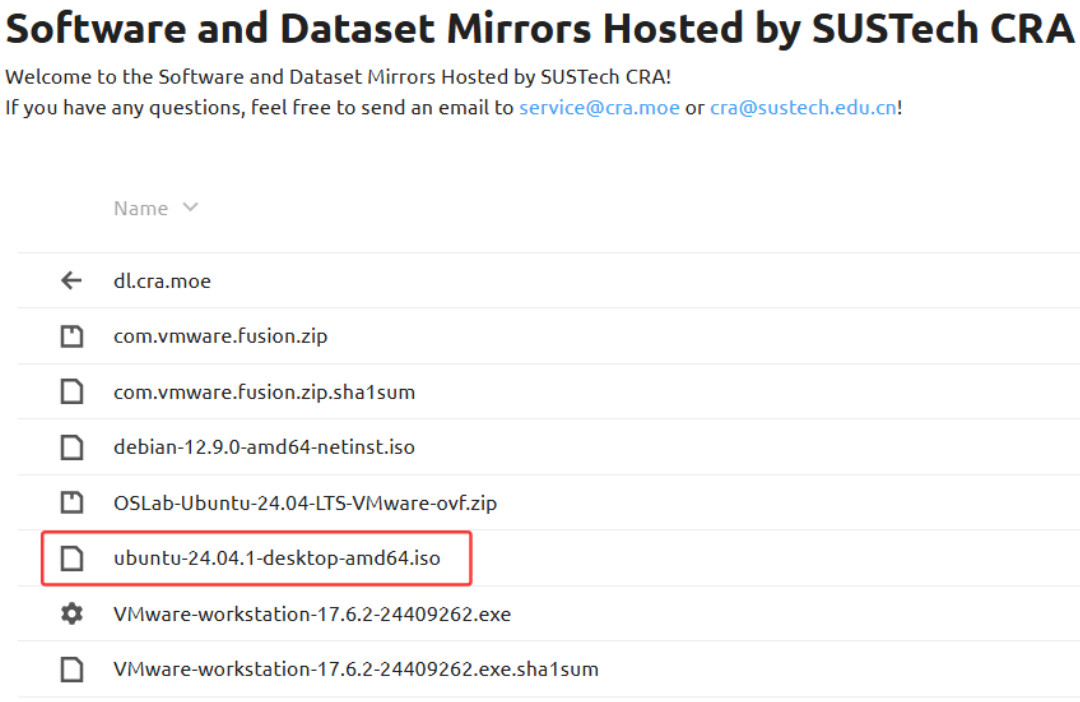
> **Note**
>
> ovf镜像系统密码为123456

2. 安装官方ubuntu 24.04

> **Warning**
>
> 如果已导入ovf文件则此步骤可以跳过

同学们可以通过以下链接下载原版Ubuntu24.04镜像，并将其安装在自己的虚拟机中。

Download Ubuntu 24.04

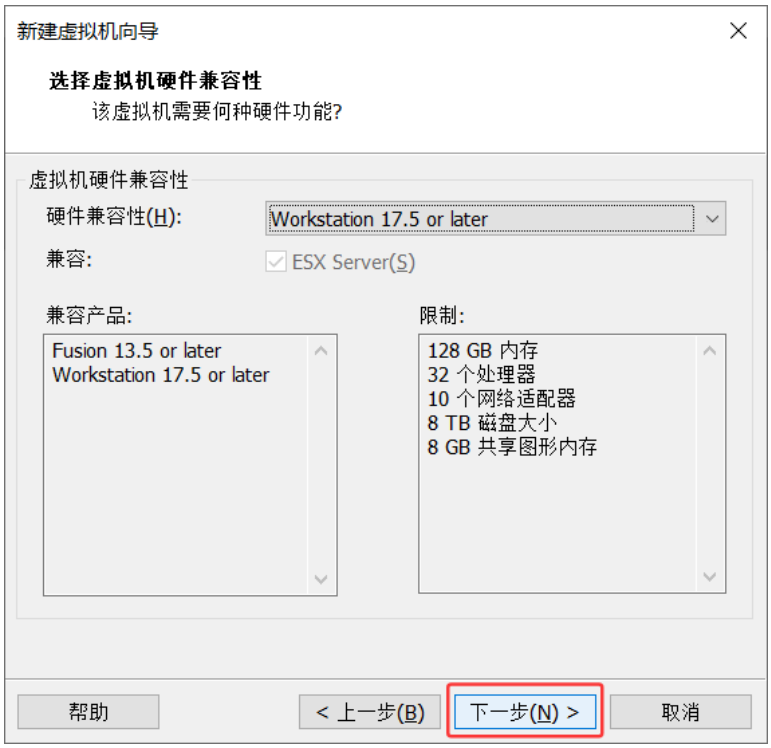


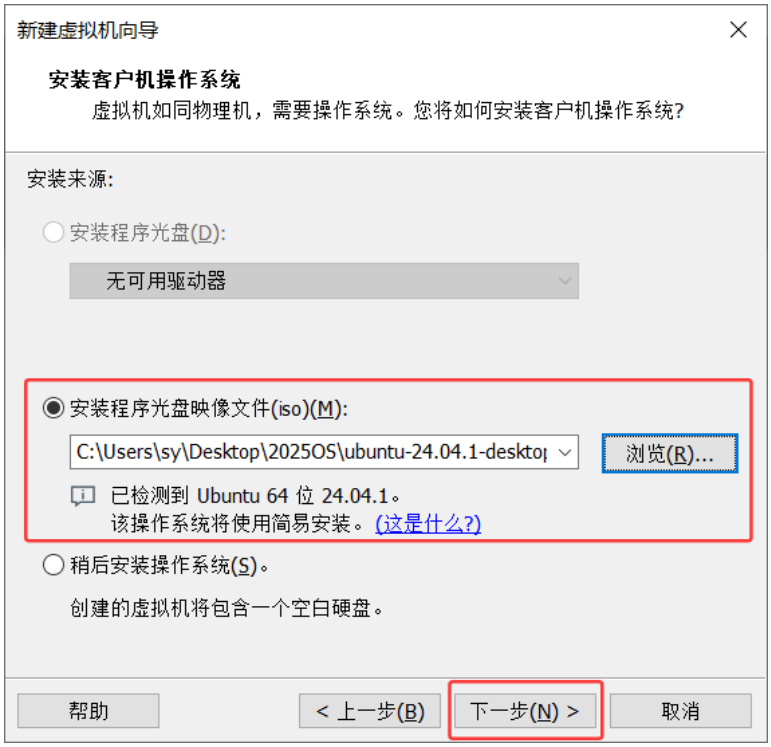
下载好镜像后，打开VMware workstation，选择 `创建新的虚拟机`：

根据以下指引完成虚拟机配置：

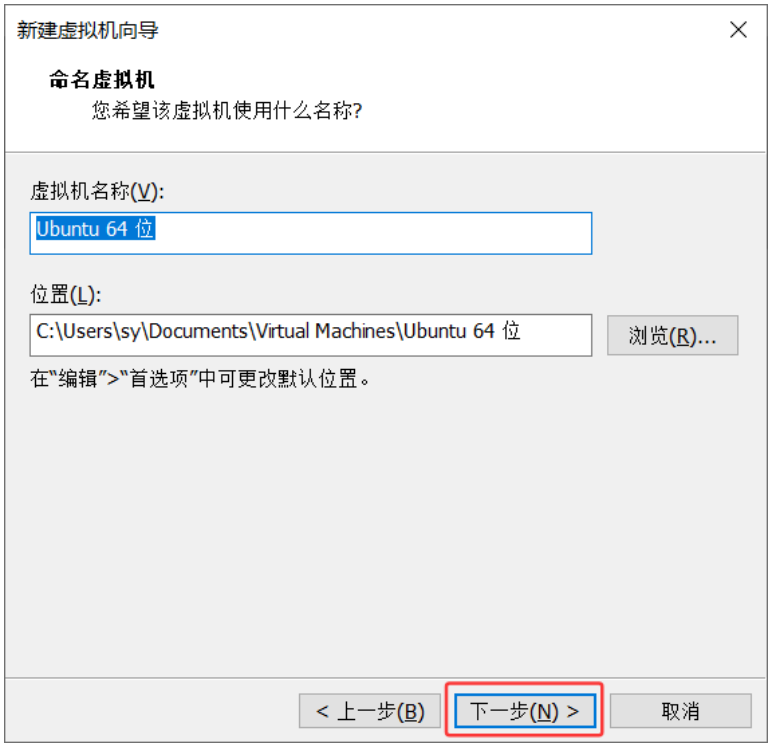
> **Note**
>
> 请同学们在安装系统时尽量满足以下要求：
> 1. 语言尽量选择英文，中文路径可能导致实验内容无法顺利完成
> 2. CPU尽量选择4核或以上
> 3. 虚拟机硬盘空间建议选择30G以上

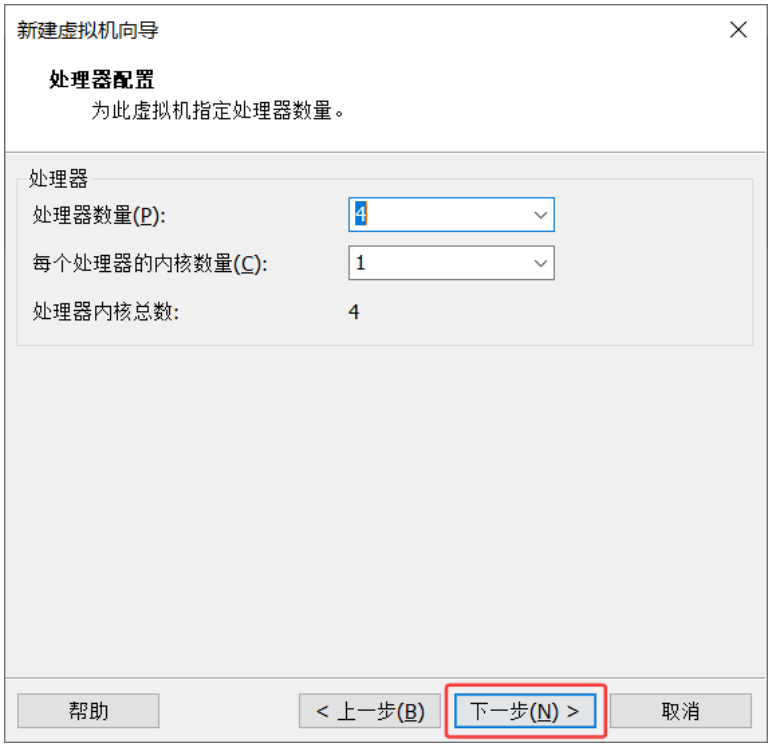

新建虚拟机向导
vmware
WORKSTATION
PRO™
17
欢迎使用新建虚拟机向导
您希望使用什么类型的配置?
典型(推荐)(T)
通过几个简单的步骤创建 Workstation 17.5 or later 虚拟机。
自定义(高级)(C)
创建带有 SCSI 控制器类型、虚拟磁盘类型以及与旧版 VMware 产品兼容性等高级选项的虚拟机。
帮助
< 上一步(B)
下一步(N) >
取消

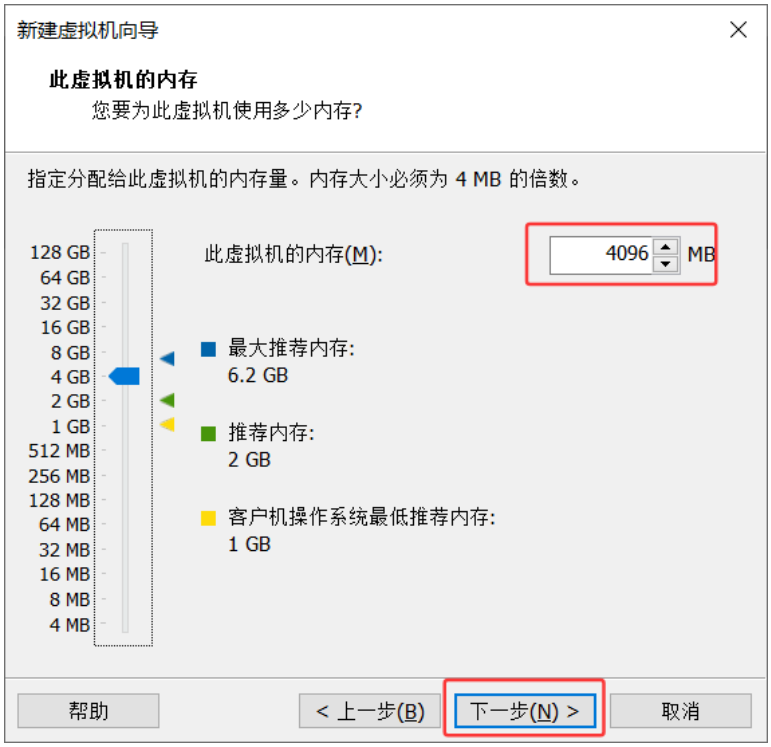

新建虚拟机向导
选择虚拟机硬件兼容性
该虚拟机需要何种硬件功能?
虚拟机硬件兼容性
硬件兼容性(H):
Workstation 17.5 or later
兼容:
ESX Server(S)
兼容产品:
Fusion 13.5 or later
Workstation 17.5 or later
限制:
128 GB 内存
32 个处理器
10 个网络适配器
8 TB 磁盘大小
8 GB 共享图形内存
帮助
< 上一步(B)
下一步(N) >
取消

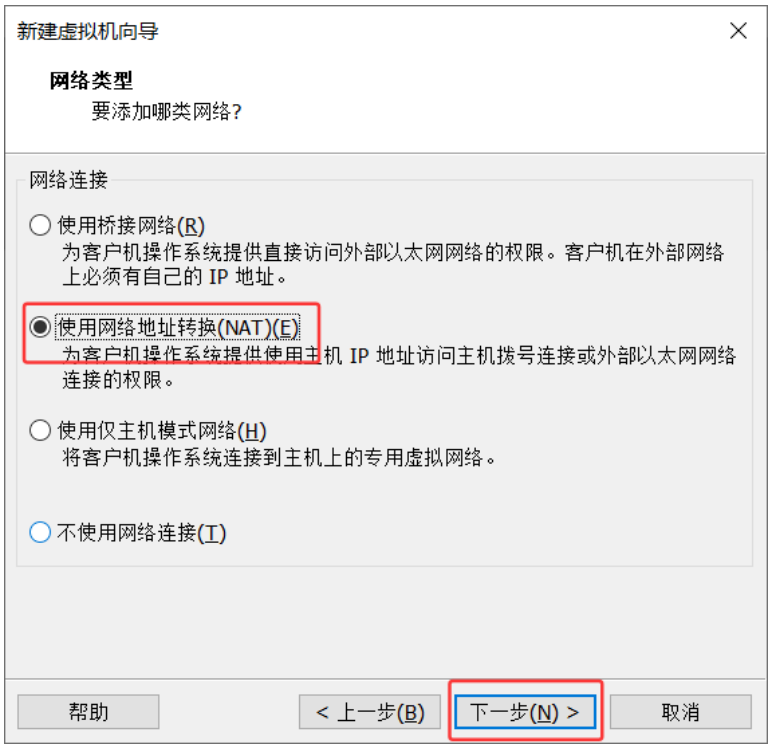

新建虚拟机向导
安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统?
安装来源:
安装程序光盘(D):
无可用驱动器
安装程序光盘映像文件(iso)(M):
C:\Users\sy\Desktop\2025OS\ubuntu-24.04.1-desktop
浏览(R)...
已检测到 Ubuntu 64 位 24.04.1。
该操作系统将使用简易安装。(这是什么?)
稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。
帮助
< 上一步(B)
下一步(N) >
取消

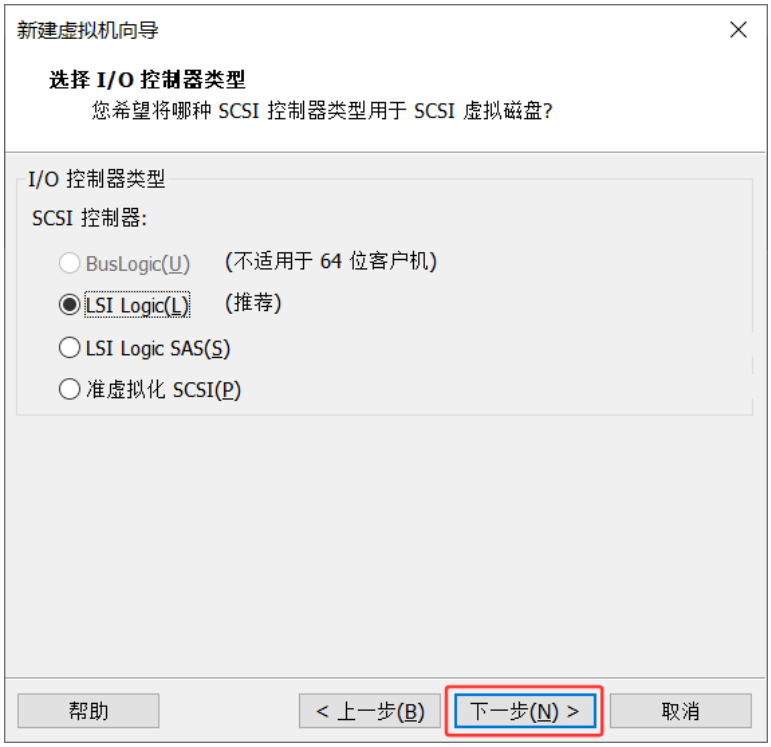

新建虚拟机向导
简易安装信息
这用于安装 Ubuntu 64 位。
个性化 Linux
全名(F):
OS
用户名(U):
sy22221111
密码(P):
确认(C):
帮助
< 上一步(B)
下一步(N) >
取消

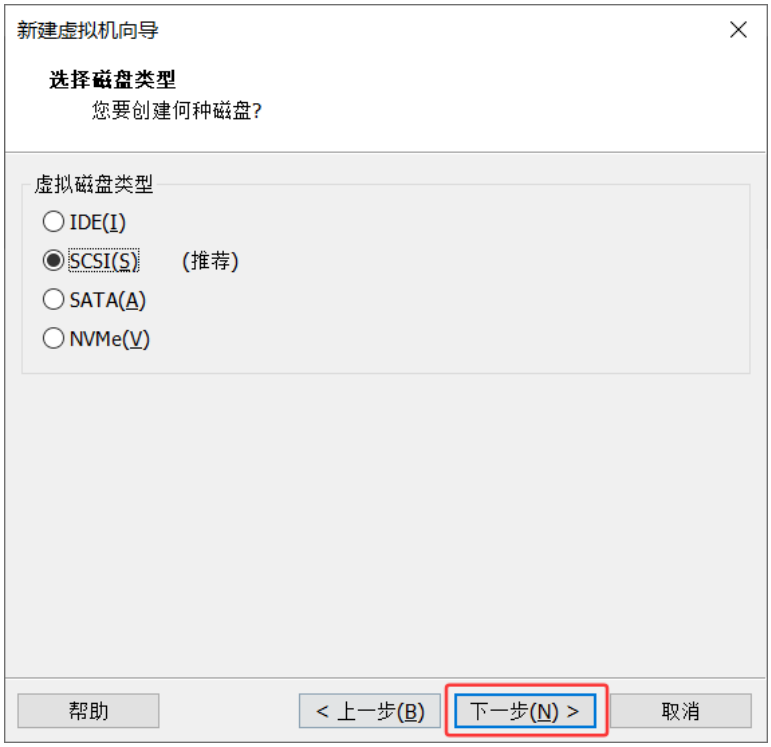

新建虚拟机向导
命名虚拟机
您希望该虚拟机使用什么名称?
虚拟机名称(V):
Ubuntu 64 位
位置(L):
C:\Users\sy\Documents\Virtual Machines\Ubuntu 64 位
浏览(R)...
在"编辑">"首选项"中可更改默认位置。
< 上一步(B)
下一步(N) >
取消

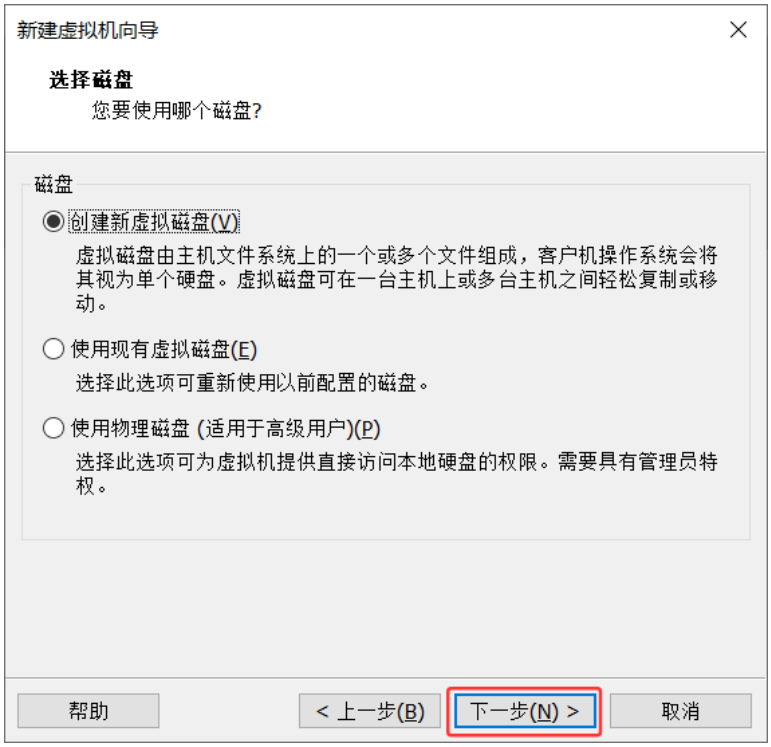

新建虚拟机向导
处理器配置
为此虚拟机指定处理器数量。
处理器
处理器数量(P):
4
每个处理器的内核数量(C):
1
处理器内核总数:
4
帮助
< 上一步(B)
下一步(N) >
取消

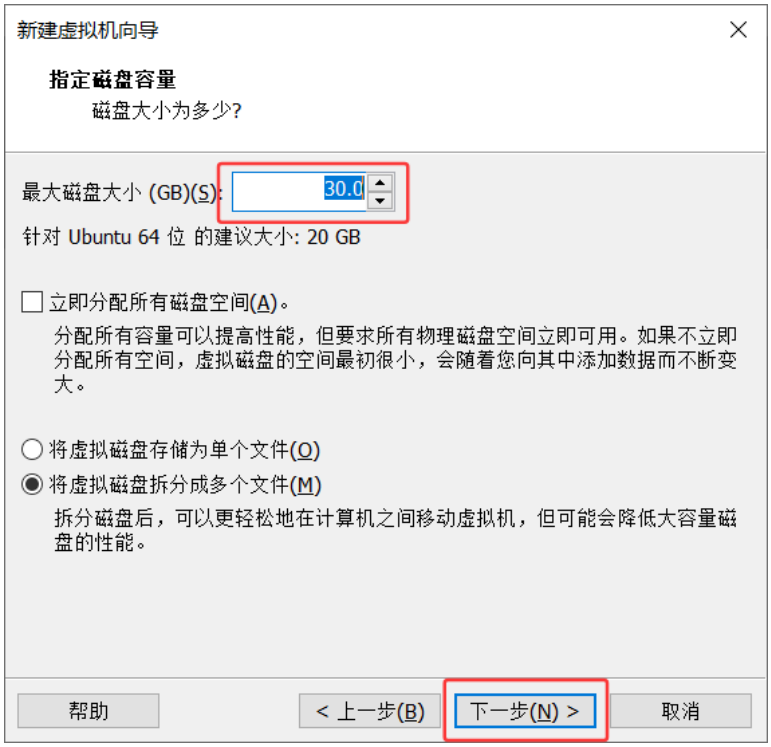

新建虚拟机向导
此虚拟机的内存
您要为此虚拟机使用多少内存?
指定分配给此虚拟机的内存量。内存大小必须为 4 MB 的倍数。
128 GB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
1 GB
512 MB
256 MB
128 MB
64 MB
32 MB
16 MB
8 MB
4 MB
此虚拟机的内存(M):
4096
MB
最大推荐内存:
6.2 GB
推荐内存:
2 GB
客户机操作系统最低推荐内存:
1 GB
帮助
< 上一步(B)
下一步(N) >
取消

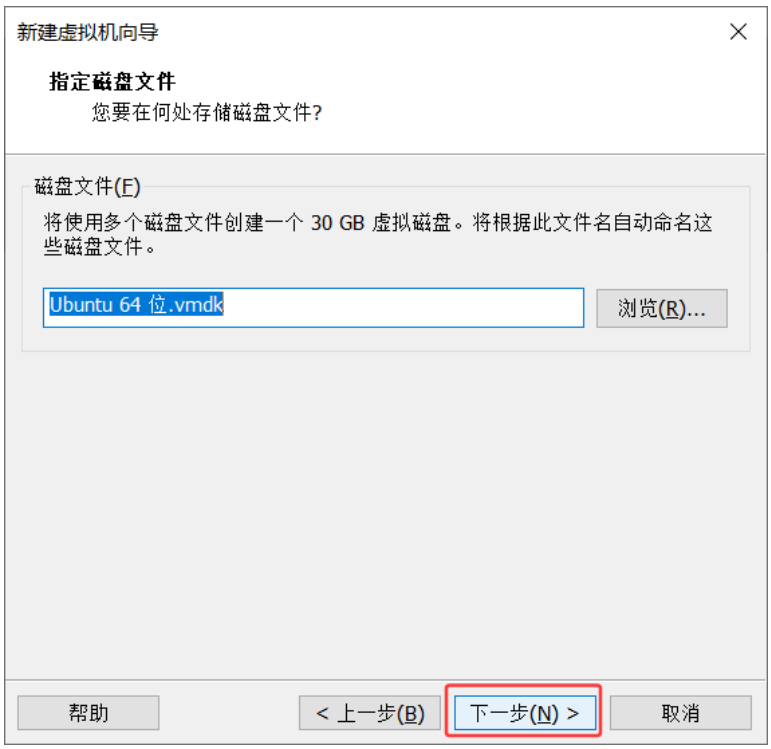

新建虚拟机向导
网络类型
要添加哪类网络?
网络连接
使用桥接网络(R)
为客户机操作系统提供直接访问外部以太网网络的权限。客户机在外部网络上必须有自己的 IP 地址。
使用网络地址转换(NAT)(E)
为客户机操作系统提供使用主机 IP 地址访问主机拨号连接或外部以太网网络连接的权限。
使用仅主机模式网络(H)
将客户机操作系统连接到主机上的专用虚拟网络。
不使用网络连接(T)
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
选择 I/O 控制器类型
您希望将哪种 SCSI 控制器类型用于 SCSI 虚拟磁盘?
I/O 控制器类型
SCSI 控制器:
BusLogic(U)
(不适用于 64 位客户机)
LSI Logic(L)
(推荐)
LSI Logic SAS(S)
准虚拟化 SCSI(P)
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
选择磁盘类型
您要创建何种磁盘?
虚拟磁盘类型
IDE(I)
SCSI(S)
(推荐)
SATA(A)
NVMe(V)
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
选择磁盘
您要使用哪个磁盘?
磁盘
创建新虚拟磁盘(V)
虚拟磁盘由主机文件系统上的一个或多个文件组成，客户机操作系统会将其视为单个硬盘。虚拟磁盘可在一台主机上或多台主机之间轻松复制或移动。
使用现有虚拟磁盘(E)
选择此选项可重新使用以前配置的磁盘。
使用物理磁盘 (适用于高级用户)(P)
选择此选项可为虚拟机提供直接访问本地硬盘的权限。需要具有管理员特权。
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
指定磁盘容量
磁盘大小为多少?
最大磁盘大小 (GB)(S):
30.0
针对 Ubuntu 64 位 的建议大小: 20 GB
立即分配所有磁盘空间(A)。
分配所有容量可以提高性能，但要求所有物理磁盘空间立即可用。如果不立即分配所有空间，虚拟磁盘的空间最初很小，会随着您向其中添加数据而不断变大。
将虚拟磁盘存储为单个文件(O)
将虚拟磁盘拆分成多个文件(M)
拆分磁盘后，可以更轻松地在计算机之间移动虚拟机，但可能会降低大容量磁盘的性能。
帮助
< 上一步(B)
下一步(N) >
取消



之后运行虚拟机完成虚拟机安装即可。

> **Note**
>
> 由于原版Ubuntu系统不包含本课程实验的开发环境，还需要参考`手动配置开发环境`完成实验开发环境配置。除此之外，我们还推荐安装vscode用于代码的阅读与编写。而这些配置在ovf中是已经包含的。

手动配置开发环境

如果你使用我们打包的 ovf 格式的镜像，**你可以跳过这一步。**

到这一步时，请确保你已经配置好 Linux 环境，能打开一个 Terminal 执行命令。

> **Warning**
>
> 请使用尽可能新的发行版本。以下代码均已 Ubuntu 24.04 LTS 为基准。

安装 gcc 工具链以及 QEMU

使用包管理器 apt 安装依赖，在terminal中执行：

```
sudo apt update && sudo apt install gcc-riscv64-unknown-elf qemu-system-misc git
make cmake python3-pip elfutils gdb-multiarch
```

安装完成后，运行 `riscv64-unknown-elf-gcc --version`，`qemu-system-riscv64 --version` 和 `gdb-multiarch --version` 检查版本：

```
$ riscv64-unknown-elf-gcc --version
riscv64-unknown-elf-gcc (13.2.0-11ubuntu1+12) 13.2.0
Copyright (C) 2023 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.

$ qemu-system-riscv64 --version
QEMU emulator version 8.2.2 (Debian 1:8.2.2+ds-0ubuntu1)
Copyright (c) 2003-2023 Fabrice Bellard and the QEMU Project developers

$ gdb-multiarch --version
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

请确保 gcc 版本 >= 13.0.0，qemu-system-riscv64 版本 >= 8.0.0，gdb-multiarch 版本 >= 15

编译与运行内核

克隆内核代码和用户程序代码仓库：

```
git clone https://github.com/yuk1i/SUSTechOS
cd SUSTechOS
git clone https://github.com/yuk1i/SUSTechOS-2025S-user user
```

编译用户程序：

```
make user
```

编译内核:

```
make
```

运行内核：

```
make run
```

如果看到以下运行结果，代表配置成功：

```
OpenSBI v1.5
   ____                   _____ ____ _____
  / __ \                 / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : medeleg
Platform HART Count       : 1
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 10000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : syscon-reboot
Platform Shutdown Device  : syscon-poweroff
Platform Suspend Device   : ---
Platform CPPC Device      : ---
Firmware Base             : 0x80000000
Firmware Size             : 327 KB
Firmware RW Offset        : 0x40000
Firmware RW Size          : 71 KB
Firmware Heap Offset      : 0x49000
Firmware Heap Size        : 35 KB (total), 2 KB (reserved), 11 KB (used), 21 KB
(free)
Firmware Scratch Size     : 4096 B (total), 416 B (used), 3680 B (free)
Runtime SBI Version       : 2.0

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*
Domain0 Region00          : 0x0000000000100000-0x0000000000100fff M: (I,R,W)
S/U: (R,W)
Domain0 Region01          : 0x0000000010000000-0x0000000010000fff M: (I,R,W)
S/U: (R,W)
Domain0 Region02          : 0x0000000002000000-0x000000000200ffff M: (I,R,W)
S/U: ()
Domain0 Region03          : 0x0000000080040000-0x000000008005ffff M: (R,W) S/U:
()
Domain0 Region04          : 0x0000000080000000-0x000000008003ffff M: (R,X) S/U:
()
Domain0 Region05          : 0x000000000c400000-0x000000000c5fffff M: (I,R,W)
S/U: (R,W)
Domain0 Region06          : 0x000000000c000000-0x000000000c3fffff M: (I,R,W)
S/U: (R,W)
Domain0 Region07          : 0x0000000000000000-0xffffffffffffffff M: () S/U:
(R,W,X)
Domain0 Next Address      : 0x0000000080200000
Domain0 Next Arg1         : 0x000000009fe00000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes
```

```
Domain0 SysSuspend        : yes

Boot HART ID              : 0
Boot HART Domain          : root
Boot HART Priv Version    : v1.12
Boot HART Base ISA        : rv64imafdch
Boot HART ISA Extensions  : sstc,zicntr,zihpm,zicboz,zicbom,sdtrig,svadu
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 2 bits
Boot HART PMP Address Bits: 54
Boot HART MHPM Info       : 16 (0x0007fff8)
Boot HART Debug Triggers  : 2 triggers
Boot HART MIDELEG         : 0x0000000000001666
Boot HART MEDELEG         : 0x0000000000f0b509


=====
Hello World!
=====

Boot stack: 0x0000000080210000
clean bss: 0x0000000080210e00 - 0x0000000080228000
Boot m_hartid 0
[INFO  0,-1] bootcpu_entry: basic smp inited, thread_id available now, we are
cpu 0: 0x00000000802270d8
Kernel Starts Relocating...
Kernel size: 0x0000000000028000, Rounded to 2MiB: 0x0000000000200000
[INFO  0,-1] bootcpu_start_relocation: Kernel phy_base: 0x0000000080200000,
phy_end_4k:0x0000000080228000, phy_end_2M 0x0000000080400000
Mapping Identity: 0x0000000080200000 to 0x0000000080200000
Mapping kernel image: 0xffffffff80200000 to 0x0000000080200000
Mapping Direct Mapping: 0xffffffc080400000 to 0x0000000080400000
Enable SATP on temporary pagetable.
Boot HART Relocated. We are at high address now! PC: 0xffffffff80203cc4
[INFO  0,-1] kvm_init: boot-stage page allocator: base 0xffffffc080400000, end
0xffffffc080600000
[INFO  0,-1] kvmmake: Memory after kernel image (phys) size = 0x0000000003c00000
[INFO  0,-1] kvm_init: enable pageing at 0x8000000000080400
[INFO  0,-1] kvm_init: boot-stage page allocator ends up: base
0xffffffc080400000, used: 0xffffffc080411000
Relocated. Boot halt sp at 0xfffffffffff001fb0
Boot another cpus.
- booting hart 1: hsm_hart_start(hartid=1, pc=_entry_sec, opaque=1) = -3.
waiting for hart online
skipped for hart 1
- booting hart 2: hsm_hart_start(hartid=2, pc=_entry_sec, opaque=1) = -3.
waiting for hart online
skipped for hart 2
- booting hart 3: hsm_hart_start(hartid=3, pc=_entry_sec, opaque=1) = -3.
waiting for hart online
skipped for hart 3
System has 1 cpus online

UART inited.
[INFO  0,-1] kpgmgrinit: page allocator init: base: 0xffffffc080411000, stop:
0xffffffc084000000
[INFO  0,-1] allocator_init: allocator mm inited base 0xffffffff00000000
[INFO  0,-1] allocator_init: allocator vma inited base 0xffffffff01000000
```

```
[INFO  0,-1] allocator_init: allocator proc inited base 0xffffffd02000000
[INFO  0,-1] allocator_init: allocator kstrbuf inited base 0xffffffd03000000
applist:
        init
        sh
        test_arg
        test_malloc
[INFO  0,-1] load_init_app: load init proc init
[INFO  0,-1] bootcpu_init: start scheduler!
init: starting sh
```