



Lab1: Linux Introduction

杨润东 10205102454[at]stu.ecnu.edu.cn

- 🖥️ 环境搭建——老生常谈之Linux安装
- 📄 经典“黑框框”
- \$ Shell入门
- ⚠️ 注意事项



🐳 环境搭建

I use arch, btw.

Q是一位OSLab的新生，他想要搭建实验所需要的环境。

Q：Linux这么多版本，该装哪一个？

- 虚假的程序员：你好，Ubuntu。
- 有经验的程序员：你好，ArchLinux。
- 老道的程序员：你好，Debian。
- GNU教徒：触发连招：

你说得对，但是你所说的Linux，实际上是GNU/Linux，或者叫GNU+Linux。Linux本身并不是一个操作系统，而是完全正常工作的GNU系统所需的另一个自由的部件，整个GNU系统是由GNU corelibs, shell工具组和主要系统部件构成的、符合POSIX定义的完整操作系统。无论你是否使用 GNU/Linux，请不要含糊不清地使用“Linux”这一名称来混淆大众。Linux 是一个内核，是系统的主要基本部件之一。整个系统基本上是 GNU 系统，加上 Linux。如果你说的是这个组合，请称之为“GNU/Linux”。

——GNU/Linux 问答 - GNU 工程 - 自由软件基金会




环境搭建


I use debian, btw.

事实上，把大象放进冰箱只需要三步：

1. 选择你想用的GNU/Linux发行版；
2. 选择安装的方式；
3. 按照教程安装或者Google it。

GNU/Linux发展至今，已经有了非常非常多不同的发行版。事实上，每种发行版都有其独特的风格和特点，只是其针对的人群不同。大家都是开源软件，生而平等。

目前较为常用的GNU/Linux发行版也有很多，例如  Ubuntu、 ArchLinux、 Debian、 Fedora、 CentOS，以及国产的Deepin等等。理论上讲，这些发行版都能满足我们实验所需要的环境。但实际上由于各种发行版的支持与更新不同，**可能会**存在某些工具链缺失或者需要自行编译的情况。

大家可以自行选择想用的发行版进行安装。当然如果你不想花太多时间在配置上，最常用也不需要复杂配置的  Ubuntu就能胜任工作。

🐧 环境搭建

I use fedora, btw.

接下来需要选择安装的方式，也就是：

- 安装在什么上？
- 用什么方式安装？

通常来说，你可以选择把系统安装在虚拟机或者物理机上。

如果你选择虚拟机，你需要一个软件来创建、管理和使用你的虚拟机。有很多好用的虚拟机软件可以用，例如我们已经接触过很多次的VMware、VirtualBox以及Windows自带的Hyper-V，这些都可以帮助你快速创建一个虚拟机。并且只需要占用你当前系统中的一些存储空间和内存，就可以和你的现有系统一起运行。

使用虚拟机安装GNU/Linux的优点是方便快捷，只需要下载镜像，然后按照虚拟机软件的向导创建新虚拟机就可以了，并且管理方便。但缺点是性能发挥非常受限，尤其是在一些显示效果上（由于显卡驱动的问题），虚拟机会显得比较卡顿。

或者，如果你现在在使用Windows 10/11，建议可以尝试使用Windows Subsystem for Linux 2(也即WSL 2)。它不是虚拟机，但可以在Windows上提供完整的Linux开发环境。并且可以和Windows协同工作，使用体验堪称完美。

🍷 环境搭建

I use hackintosh, btw.

当然，如果你打算尝试一下一个“完全体”的GNU/Linux，你也可以选择直接装在一台物理机上。

同样，也有很多方式来安装在一台机器上。如果你的电脑恰好有多个盘位，你又恰好有一个空的硬盘可以使用，或者你恰好懂得如何在一个盘上装上两个系统，那么你就可以在你的电脑上装上另一个系统，不过会有亿点点麻烦，并且在两个系统之间互相切换也非常的麻烦，比如数据的同步就是一个大问题。

或者你恰好有一台闲置的电脑可以用来单独安装环境，那么无疑是更好的选择。通常来说，你也许需要一个U盘来烧录镜像，然后在BIOS中选择从U盘启动以进入安装向导。

当然，如果你打算从此抛弃Windows，此生只用GNU/Linux，那么直接将你当前使用的系统抹掉也是可以的，只是有亿点点危险。不建议这样做，因为不要把鸡蛋放在一个篮子里。

又或者，如果你是钞能力者，那么也可以租/买一个云服务器来搭建环境，不过这样颇有些杀鸡用牛刀的意味。

如果在之前的某些课程需要下已经安装过一些GNU/Linux发行版，那么大概率你可以直接使用他们，不需要再重新安装。

理论上MacOS也是可以直接使用的，因为我们只是需要用到一些工具。

☞ 经典“黑框框”

■ Shell和Terminal

In computing, a shell is a computer program that exposes an operating system's services to a human user or other programs. In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer's role and particular operation. It is named a shell because it is the outermost layer around the operating system.[1][2]

——*wikipedia*


那么终端是什么？简而言之就是人类用户与计算机交互的设备。（Shell的Shell）



终端现在通常是指虚拟终端机(Terminal emulator)，是实现了CLI的一种软件。

各种系统上都可以见到终端的身影，比如Windows下的Windows Terminal、Mac OS X下的iTerm2，以及GNU/Linux发行版各种自带的终端。你可以在网上下载到各种各样的终端来使用。

☞ 经典“黑框框”

- 我们常用的Shell有 \$ sh(Bourne shell)、 Bash(Bourne-Again SHell)、zsh(Z-shell)等。
- sh较旧，功能比较少；而Bash有许多功能，并且比较清爽；zsh功能最多，拓展性极强，但是有时加载会比较慢，并且不够清爽。
- 大多数情况下，这些shell的区别在于一些方便使用的特性（例如是否支持自动补全等），而shell的语法基本都是通用的，只在个别情况下会有语法上的小差别。
- 建议使用Bash或者zsh。

Q：为什么都说CLI更强大？GUI不是更好吗？我讨厌这个黑框框，什么都看不见。

A：事实上，CLI和GUI互有优劣。GUI的优势在于把复杂的功能经过巧妙的设计，可以使得一些功能的使用更加方便快捷，例如一些按钮等。这实际上也是抽象层次提高的一种表现，所以同样地，GUI也损失了一些软件功能上的灵活性。比如，你很难在两个毫无关联的软件之间通过鼠标点击或拖动完成交互（除了文字）。而在CLI下，由于大家的输入和输出都是文字（字符串），所以程序之间的交互会变得更加灵活。

- 有很多**真正小而美**的软件供开发者使用，将他们组合使用就可以简单地实现很有效的功能。

\$ Shell入门

1. Hello World!

```
echo Hello World!  
echo "Hello World!"  
echo 'Hello World!'  
echo '"Hello World!"'
```

2. 变量

```
foo = bar           #error  
foo=bar  
echo foo  
echo $foo  
echo ${foo}  
echo "$foo"  
echo "$foofoo"      #undefined (empty)  
echo "${foo}ooo"  
echo '$foo'  
pwd  
echo pwd  
echo $pwd           #undefined (empty)  
echo $(pwd)
```


\$ Shell入门

3. 来在Shell里活动活动身子罢

```
ls      # list directory contents
cd      # change directory
mkdir   # make directory
touch   # change file timestamps, but usually is used to create an empty file
mv      # move
cp      # copy
rm      # remove(non-recursive) BE CAREFUL! VERY DANGEROUS!
```

4. 给程序们“加点料”

```
ls -a
ls -l
cd -
cd ~
rm -r
rm -rf
sudo rm -rf /* # !!!FBI WARNING! 俗称删库跑路
```

\$ Shell入门

5. 小而美实用工具

```
man ls      # manual
cat a.txt   # concatenate files and print on stdout
head a.txt  # output first part of files
tail a.txt  # output last part of files
wc a.txt    # word counter
find        # search for files in a directory hierarchy
tr          # translate or delete characters
tee         # read and write
awk
grep
sed         #Linux三剑客
tldr        # too long don't read
apt         # super power package manager
```

\$ Shell入门

6. 重定向(rewire)

■ 输出重定向(stdout->file)

```
echo 'Hello World!' > a.txt    # 覆盖
echo 'Hello World!' >> a.txt   # 追加
ls >> a.txt
echo $PATH >> a.txt
gcc -E test.c > test.i
```

你甚至可以用`echo`来写代码

■ 输入重定向(file->stdin)

```
cat file1.txt file2.txt file3.txt > output.txt
cat file1.txt - file3.txt < file2.txt > output.txt
```

事实上重定向不仅可以重定向stdin&stdout，还有很多玩法。这里有一个简易教程[Linux Shell重定向（输入输出重定向）精讲](http://biancheng.net)
(biancheng.net)

\$ Shell入门

这位更是重量级

7. 管道(pipe)

```
ls | wc    # Is it correct?
ls | xargs wc
find . | xargs wc
python3 mylogger.py | tee mylog.log
watch -t -n1 "date +%T|figlet" # where is the pipe?
```

■ 管道可以套娃

```
find . | grep ".dart" | xargs wc --lines
# download the textbooks
curl "https://pages.cs.wisc.edu/~remzi/OSTEP/" | grep -Po "(?<=href=).+\.pdf" | awk '{print "https://pages.cs.wisc.e \
du/~remzi/OSTEP/" $0}'
```

■ 通过管道、重定向，可以实现程序间的交互

在一些Shell中，管道甚至被实现了传输流数据（图片或流视频）的功能，非常强大。

\$ Shell入门

9. 常用操作:

- `Ctrl-C` "abort the current task and regain user control"
- `Ctrl-D` "maybe eof or logout"
- `Ctrl-Z` "suspend current job"
- `Ctrl-L` "(in bash and zsh) clear your console. (But not deleting history)"
- `bg` "Move jobs to the background"

 `python3 logger.py > /dev/null &`
- `fg` "Move job to the foreground"
- `jobs` "Display status of jobs"
- `wait` "wait for job completion and return exit status"

\$ Shell入门

10. Shell脚本

- 顺序执行一堆shell命令
- 增加了一些控制语句，并且支持变量和函数

Special Parameters

\$0	filename of the script
\$1 ~ \$9	Arguments to the scripts
\$#	The number of the arguments
\$?	Return error code of the previous command
\$@	All command line arguments
\$\$	PID for the current script

\$ Shell入门

■ 控制语句

```
if [expr1]; then
    # ...
elif [expr2]; then
    # ...
else
    # ...
fi

for i in bob frank lily; do
    echo $i
done

for i in {1..5}; do
    echo $i
done

while [ expr ]; do
    # ...
done
```

\$ Shell入门

- 常用的条件表达式(expr, or Conditional Expressions)

[-f FILE] [-d FILE]	FILE exists and is file / directory
[-r FILE] [-w FILE] [-x FILE]	FILE readable / writable / executable
[-z STRING]	if STRING is empty
[-n STRING] [STRING]	if STRING is NOT empty
[STRING1 == STRING2]	STRING equal
[INTEGER1 -eq INTEGER2]	INTEGER equal
[INTEGER1 -ge INTEGER2]	INTEGER >=

- 某个条件/某个工具/某个变量不知道怎么办?

RTFM(``man test``) || STFW(Google "bash string equals")

\$ Shell入门

■ 函数

```
#my_cd.sh  
mycd () {  
    mkdir -p $1  
    cd $1  
}
```

如何运行脚本？

- ``sh script.sh`` 或者 ``bash script.sh``：用sh或者bash这个程序来读取解释运行脚本文件
- ``.`./script.sh``：根据文件内容让Shell自行决定如何运行该文件
 - 通常来说，为了确保文件被正常运行，如果希望通过这种方式直接运行脚本，最好在第一行添加一行声明解释器，例如 ``#!/bin/bash`` 或者 ``#!/usr/bin/python3``。这一行也叫Shebang。
 - 文件需要可以被执行
- ``source script.sh``：在当前shell环境下运行脚本
- ``.`. script.sh``：等同于 ``source script.sh``

\$ Shell入门

- 环境变量

```
env  
export foo=bar
```

- ``export``: shell在创建子进程时设置环境变量
- ``name=value command``: 在执行这句命令时单独设置环境变量

- 权限

```
$ ls -l  
-rw-r--r-- 1 subway subway 111 Sep 10 22:35 1  
drwxr-xr-x 2 subway subway 4096 Apr 12 04:36 bin  
lrwxrwxrwx 1 root root 4 Aug 29 2022 /usr/bin/sh -> dash
```

- ``chmod +777`` super power!
- ``sudo command``: 以root权限运行command
- ``sudo !!``: 以root权限运行上一条命令

⚠ 注意事项

让我看看作业什么时候交——昨天。

- 作业周期可能会比较长，考虑到工作量和难度，把作业拖到DDL前一天是不太明智的 (除了Train Fun)
- 作业提交时间一般**不是**晚上十二点，而是晚上九点，所以请妥善安排时间。
- 早开始，早行动，不要抄袭，以及不要试图直接完全依赖ChatGPT。
- 虽然老师大概率不会点名，但是建议哪怕迟到一小时也不要缺课，否则有可能一步跟不上，步步打急慌。
- 课程主页中提到的Operating Systems: Three Easy Pieces, OSTEP是一部非常棒的课本，其线上完全免费，并且PDF分章节提供。虽然是全英但是简单易懂，叙述简洁有力。建议做作业没思路或者对细节有问题时仔细阅读。
- 欢迎随时提问。
- 由于特殊原因，部分情况下Github可能会抽风（比如push不了）。如果需要协助可以私信TA。

☰ 补充材料

- The Missing Semester Train Fun 倾情翻译
 - jyy's Linux Intro 蒋炎岩yyds
 - GNU 官网
- 我用的什么做的PPT?
- Slidev