第一章 Linux是什么，如何学习

一、Linux是什么?

1.1 Linux是什么？操作系统/应用程序?

1. 明确核心问题：Linux 是什么?

目标：理解 Linux 是操作系统（核心），而非应用程序。
关键点：

- **操作系统的作用**：管理硬件资源（CPU、内存、硬盘）、提供基础功能（如网络、文件系统）和开发接口。
- 类比解释：
  - 操作系统 = 公路（协调硬件资源，让程序有序运行）。
  - 应用程序 = 车辆（在交警管理下完成具体任务，如浏览器、游戏）。
- “手机上的 Android 和电脑上的 Windows 是操作系统还是应用程序？”
- “如果电脑没有操作系统，你能打开微信吗？为什么？”

---

2. Linux 的组成：核心 + 工具 + 应用

目标：区分 Linux 核心与完整操作系统的关系。
关键点：

- **Linux 核心**：仅控制硬件和提供基础接口（如系统调用）。
- **完整操作系统** = Linux 核心 + GNU 工具（如 Bash、GCC） + 用户软件（如 Firefox）。
  示例：
- 原始 Linux 核心（1991 年）只能“驱动硬件”，需自行安装工具（如文本编辑器、编译器）。
- 现代发行版（如 Ubuntu）预装工具和软件，开箱即用。

---

3. 硬件兼容性：为何 Linux 能跨平台运行?

目标：理解“可移植性”与硬件架构差异。
关键点：

- **硬件架构差异**：不同 CPU 的指令集不同（如 x86、ARM、PowerPC）。
- **开源优势：Linux 代码可被修改适配任何硬件（对比 Windows 闭源）。**

  被动开源：员工离职公开了122行代码

  案例对比：
- **Windows**：仅支持 x86，无法在 ARM 手机或 PowerPC 服务器运行。
- **Linux**：通过修改内核代码，支持树莓派（ARM）、IBM 大型机（PowerPC）等。

---

4. 为什么 Linux 免费且合法？开源协议解析

目标：理解自由软件理念与 GPL 协议。
关键点：

- **GPL 协议四大自由**：自由使用、研究、修改、分发。
- **商业模式**：企业通过技术支持、定制开发盈利，而非软件授权费。
  互动讨论：
- “为什么 Red Hat 公司可以靠免费 Linux 赚钱？”
- “如果你修改了 Linux 代码，能否闭源并收费？”（答案：违反 GPL，必须开源修改版）
  现实案例：
- 安卓系统：基于 Linux 核心，但部分组件闭源（如 Google 服务）。

---

5. 稳定性的根源：Unix 基因与开源协作

目标：理解 Linux 为何适合服务器和关键任务。
关键点：

- **Unix 设计哲学**：模块化、权限控制、日志系统。
- **社区驱动开发**：全球开发者共同修复漏洞、优化性能。
  数据支撑：
- 全球 90% 的云计算服务器运行 Linux（如 AWS、阿里云）。
- 超级计算机 100% 使用 Linux（2023 年 Top500 数据）。
  实验活动：
- 在 Linux 中运行 `uptime` 命令，查看服务器连续运行时间。
- 对比 Windows 和 Linux 在相同负载下的崩溃频率。

---

6. 教学总结与知识巩固

总结框架：

问题	答案要点
Linux 是操作系统吗?	是，负责管理硬件和提供基础功能。
为什么 Linux 能免费使用?	遵循 GPL 开源协议，允许自由使用和修改。
为何 Linux 支持多种硬件?	开源代码可移植适配不同架构（如 x86、ARM）。
Linux 为何稳定?	继承 Unix 设计（如权限隔离）、开源社区快速修复漏洞。

1.2 Linux之前，Unix的历史

从 Unix 到 Linux 的演进

关键人物介绍

1. Ken Thompson（肯·汤普森）

- **身份**：Unix 之父、计算机科学家。
- 贡献：
  - **1969年**：用汇编语言写出 Unix 原型（Unics），提出“一切皆文件”的设计哲学。

    - **1971年**：与 Dennis Ritchie 合作开发 B 语言（C 语言前身），并移植 Unix 到 PDP-11 主机。
    - **正则表达式**：发明了正则表达式语法，用于文本处理（如 `grep` 工具）。
- **趣闻**：为了在 PDP-7 上玩《星际旅行》游戏而开发 Unix。
- **后续影响**：2007 年加入 Google，参与 Go 语言开发。

---

2. Dennis Ritchie（丹尼斯·里奇）

- **身份**：C 语言之父、Unix 联合创始人。
- 贡献：
    - **1973年**：将 B 语言改进为 C 语言，并用 C 重写 Unix 内核，实现跨平台移植。

      windows：C语言

      微软在用RUST语言重新开发windows

      kerberos协议原理

      在电脑上使用打印服务

      验证身份>>>身份票据（黄金票据）

      拿着身份票据去换>>>服务票据（白银票据）

      拿着服务票据去找对应服务器

    - **《C 程序设计语言》**：与 Brian Kernighan 合著，成为 C 语言圣经（K&R C）。
- **荣誉**：1983 年与 Thompson 共获图灵奖。
- **名言**：**“Unix is simple. It just takes a genius to understand its simplicity.”**
  （Unix 很简单，但需要天才才能理解它的简洁。）

---

3. Richard Stallman（理查德·斯托曼）

- **身份**：自由软件运动发起人、GNU 计划创始人。
- 贡献：
    - **1984年**：发起 GNU 计划，目标是创建自由 Unix 替代品。
    - **GPL 协议**：制定“传染性”开源协议，要求衍生作品必须开源。
    - **开发工具**：GCC 编译器、Emacs 编辑器、GNU Coreutils 工具包。
- **争议**：坚持“自由软件”（Free Software）一词，反对“开源”（Open Source）的模糊化。
- **现状**：2023 年仍在倡导自由软件，反对非自由硬件（如 DRM 技术）。

---

4. Linus Torvalds（林纳斯·托瓦兹）

- **身份**：Linux 内核创始人、Git 版本控制系统发明者。
- 贡献：
    - **1991年**：在赫尔辛基大学开发 Linux 内核，通过 Usenet 新闻组开源协作。
    - **领导风格**：以“仁慈独裁者”模式管理内核开发，确保技术决策效率。
    - **Git**：2005 年为管理 Linux 代码开发 Git，现为全球主流版本控制工具。
- **性格**：以直言不讳著称，曾因邮件列表中的激烈言辞引发争议。

- **名言**：**“Talk is cheap. Show me the code.”**
  （空谈无用，用代码证明。）

---

5. Andrew Tanenbaum（安德鲁·塔嫩鲍姆）

- **身份**：计算机科学家、Minix 操作系统开发者。
- 贡献：
  - **1987年**：为教学开发 Minix（微型 Unix），提供可读的内核源码。
  - **操作系统教材**：《现代操作系统》《操作系统设计与实现》全球广泛使用。
  - **学术影响**：Minix 启发了 Torvalds 开发 Linux，但两者因内核设计（微内核 vs 宏内核）公开争论。
- **现状**：2020 年发布 Minix 3，专注于嵌入式系统和高可靠性场景。

---

6. Bill Joy（比尔·乔伊）

- **身份**：BSD Unix 主要开发者、Sun Microsystems 联合创始人。
- 贡献：
  - **1977年**：在伯克利主导 BSD Unix 开发，引入 vi 编辑器、TCP/IP 协议栈。
  - **Sun 公司**：1982 年联合创立 Sun，推动工作站和 Solaris 操作系统发展。
  - **SPARC 处理器**：设计 RISC 架构处理器，影响现代 CPU 设计。
- **趣闻**：因开发 vi 被戏称“让无数程序员手指抽筋的人”。

总结

这些人物共同塑造了现代计算的基础：

- **Thompson 和 Ritchie** 奠定了操作系统和编程语言的基础。
- **Stallman** 捍卫软件自由，为 Linux 铺平道路。
- **Torvalds** 用开源协作颠覆传统开发模式。
- **Tanenbaum 和 Joy** 通过教育和企业推动技术普及。

理解他们的贡献，就能明白为何 **“站在巨人肩膀上”** 是技术演进的核心逻辑。

unix历史

1960s：操作系统的黎明

- **1965年**：MIT、贝尔实验室、GE 合作开发 **Multics**（多用户分时系统），但进度滞后。
- 1969年：贝尔实验室退出 Multics，Ken Thompson 用汇编语言写出 Unix 原型（Unics），核心思想：
  - **一切皆文件**（程序、设备都以文件形式管理）。
  - **单一目的原则**（每个工具只做一件事，但做到极致）。

1970s：Unix 的崛起与分裂

- **1973年**：Dennis Ritchie 用 **C语言** 重写 Unix，实现跨平台移植性。
- **1977年**：加州伯克利大学开发 **BSD**（Unix 分支），成为 Sun 公司（Solaris）和 FreeBSD 的基石。
- **1979年**：AT&T 收回 Unix 版权，禁止学术使用源码，引发商业闭源浪潮。

关键影响：

- **C语言** 成为操作系统开发的标准语言。
- **Unix 闭源** 促使学术界寻求替代方案（如 BSD 和后续 Minix）。

1980s：自由软件运动与开源萌芽

- 1984年：
  - **Minix**：Andrew Tanenbaum 为教学开发的小型 Unix-like 系统（闭源但提供源码）。
  - **GNU 计划**：Richard Stallman 发起，目标创建 自由 Unix 操作系统，开发了：
    - **GCC 编译器**、**Bash Shell**、**GNU Coreutils**（基础命令工具）。
  - **GPL 协议**：要求衍生作品必须开源，奠定自由软件法律基础。
- **1988年**：**XFree86** 项目启动，为 Linux 提供开源图形界面（GUI）。

核心矛盾：

- **GNU 工具链** 已成熟，但缺少自由的操作系统内核。

1991年：Linux 的诞生

- **8月25日**：Linus Torvalds 在新闻组发布消息，宣布开发 **Linux 内核**（0.01版）。
- 关键特性：
  - 基于 **Minix 设计思想**，但彻底开源。
  - 依赖 **GNU 工具**（Bash、GCC）构建完整操作系统。
- **社区协作**：全球开发者通过邮件列表贡献代码，迅速完善功能。

划时代意义：

- **GNU + Linux** = 首个完整的自由操作系统（GNU/Linux）。
- **开源模式** 颠覆软件产业，推动互联网、云计算和移动设备革命。

核心概念解析

1. Unix vs Linux：父子还是兄弟?

- **Unix**：商业闭源的“贵族系统”，运行于大型机和服务器。
- **Linux**：开源平民版，继承 Unix 设计哲学（如文件系统结构、权限管理），但内核代码全新编写。

类比：

- Unix 是传统燃油车，Linux 是开源设计的电动车，共享驾驶理念但技术实现不同。

---

2. GNU 工具链：Linux 的“瑞士军刀”

- 作用：提供用户与内核交互的桥梁，例如：
  - **GCC**：编译 Linux 内核和应用程序。
  - **Bash**：命令行操作的核心工具。
- **开源协作**：GNU 工具 + Linux 内核 = 完整操作系统（如 Ubuntu、Debian）。

比喻：

- Linux 内核是发动机，GNU 工具是方向盘、油门和刹车，共同组成可驾驶的汽车。

---

3. 开源文化的胜利

- **技术民主化**：任何人可查看、修改代码，推动快速创新（如 Android 基于 Linux）。
- **商业模式**：红帽（RHEL）通过订阅服务盈利，而非软件授权费。

案例：

- **1998年**：微软内部备忘录称 Linux 是“最大威胁”，但开源已不可阻挡。
- **2023年**：全球 100% 的超级计算机、90% 的云服务器运行 Linux。

1.3 关于GNU计划、自由软件与开放源代码

1. GNU 计划：自由软件的基石

- 起源：
  - **1984年**由 Richard Stallman（RMS）发起，目标是创建完全自由的类 Unix 操作系统（GNU）。
  - **核心理念**：软件应尊重用户的自由，而非成为厂商控制的工具。
- 关键成就：
  - 开发了 **GCC 编译器**、**GNU Coreutils**（基础命令工具）和 **GPL 许可证**。
  - 为 Linux 内核提供了用户空间工具链，形成完整的操作系统（GNU/Linux）。

---

2. 自由软件（Free Software）的四大自由

自由软件的核心是 **“自由”而非“免费”**，强调用户拥有以下权利：

1. **自由使用**：无限制运行软件。
2. **自由学习与修改**：可查看和修改源代码。
3. **自由分发**：分享软件副本帮助他人。
4. **自由分发修改版**：改进后回馈社区，推动共同进步。

经典比喻：

- **自由如同言论自由（Free Speech）**：你可以自由传播思想（代码）。
- **免费如同免费啤酒（Free Beer）**：仅指价格为零，不涉及权利。

---

3. GPL 许可证：自由软件的法律保障

- 核心规则：
  - **传染性**：任何基于 GPL 代码的衍生作品必须**继续开源**（如 Linux 内核）。
  - **禁止额外限制**：不得添加专利或闭源条款。
- 商业应用：
  - **允许销售**：可对软件收费，但需提供源代码（如 Red Hat 销售 RHEL 订阅服务）。
  - **服务盈利**：技术支持、定制开发和培训是主要商业模式。

案例：微软曾因 Windows 内核包含 GPL 代码（如 exFAT 驱动）被迫开源部分组件。

---

4. 开放源代码（Open Source）的实用性导向

- **目标**：通过开源协作提升软件质量和普及性，弱化意识形态。
- 与自由软件的区别：

维度	自由软件（GPL）	开源软件（如 BSD/MIT）
核心诉求	强调用户自由，反对闭源	强调代码开放，允许商业闭源
许可证限制	严格（衍生作品必须开源）	宽松（允许闭源和专利集成）
典型代表	Linux、GNU 工具链	FreeBSD、Apache、Node.js

---

5. 常见开源许可证对比

许可证	核心特点	适用场景
GPL	传染性强，衍生作品必须开源	Linux、GIMP
LGPL	允许动态链接闭源库	Glibc（C 语言标准库）
Apache 2.0	允许专利集成，明确贡献者版权	Apache HTTP Server、Android
MIT	极简规则，仅保留版权声明	jQuery、Rails
BSD	允许闭源和商业使用，无传染性	FreeBSD、macOS（部分组件）

---

6. 闭源软件（专利软件）的类型

- Freeware（免费软件）：
  - 免费使用，但**不开放源代码**（如 WinRAR 个人版）。
  - 可能存在隐私风险（如广告插件）。
- Shareware（共享软件）：
  - 免费试用，到期需付费（如 Photoshop 试用版）。
  - 功能受限，依赖用户付费解锁。

---

7. 开源与闭源的商业选择

- 开源的优势：
  - **快速迭代**：社区协作修复漏洞（如 Log4j 漏洞 24 小时内修复）。
  - **降低成本**：无需重复造轮子，企业可专注核心业务。
- 闭源的优势：
  - **控制权**：厂商完全掌控代码和功能更新（如 Windows 更新策略）。
  - **高利润**：通过许可证销售（如 Oracle 数据库）。

二、托马斯和Linux发展

1. Linus Torvalds的早期背景与动机

- 成长环境：
  - 外祖父是统计学家，引导他接触计算机编程（汇编语言）。
  - 1988年进入赫尔辛基大学计算机科学系，首次使用Unix但资源受限（仅16个终端机）。
- 技术启发：
  - **Minix系统**：Andrew Tanenbaum开发的教学用Unix-like系统，提供源码但功能有限。
  - **硬件测试**：贷款购买Intel 386计算机，验证其多任务能力（编写程序输出交替的A/B）。

关键点：

- Minix的源码开放为Torvalds提供了学习内核设计的跳板。
- 386硬件的性能验证促使他开发更灵活的操作系统。

---

2. Linux 0.02的诞生与开源发布

- 开发工具：
  - 使用GNU的**Bash**和**GCC编译器**编写内核，结合POSIX标准确保Unix兼容性。
- 历史性公告：
  - 1991年在Usenet发布消息："只是个人爱好，不会像GNU那样庞大"，吸引全球开发者关注。
  - 首版内核命名为**Linux**（源自存放目录名），支持Minix文件系统。

开源意义：

- **协作模式**：通过邮件列表和FTP共享代码，形成全球开发者参与的虚拟团队。
- **GPL协议**：确保代码自由修改与共享，避免闭源垄断。

---

3. 虚拟团队与分层管理模式

- 核心贡献者：
  - **Alan Cox**、**Stephen Tweedie**等协助测试和整合代码，形成"仁慈独裁者"模型。
  - **kernel.org**：统一代码托管平台，加速协作与版本管理。
- 开发流程：
  - 用户反馈问题 → 开发者提交补丁 → 核心团队审核 → 合并至主线版本。
  - **模块化设计**：驱动和功能模块独立开发，降低维护复杂度。

案例：

- 1994年发布Linux 1.0，支持X Window系统，标志其从个人项目转向成熟操作系统。
- 2015年Linux 4.0引入热补丁技术（Live Patching），无需重启更新内核。

---

4. Linux核心版本管理策略

- **版本号结构**：`主版本.次版本.发布版本-修改版本`（如`3.10.0-123.el7.x86_64`）。
- 历史演变：
  - **奇偶版本（2.6前）**：奇数为开发版（如2.5），偶数为稳定版（如2.6）。
  - **主线版本（3.0后）**：每2-3个月更新主线，长期维护版本（LTS）提供5年以上支持。
  - centos -8 stream

- 长期维护版本的意义：
  - 企业依赖稳定内核（如CentOS 7基于3.10），避免频繁升级风险。

数据：

- 截至2023年，Linux内核代码量超**3000万行**，年均新增代码约**100万行**。
- 贡献者来自**2000+公司**（微软、谷歌、华为等）。

---

5. Linux发行版（Distributions）的分类与选择

- **两大派系**：

软件管理	代表发行版	特点
RPM	Red Hat、Fedora、CentOS	企业级稳定，支持商业服务（如RHEL）。
DPKG	Debian、Ubuntu、Linux Mint	社区驱动，桌面友好，软件更新快。

- **选择建议**：
  - **企业服务器**：RHEL（付费支持）、CentOS（免费替代）。
  - **个人开发**：Ubuntu（易用性）、Fedora（前沿技术）。
  - **嵌入式/轻量**：Debian、Arch Linux（高度定制）。

案例对比：

- **CentOS**：完全兼容RHEL，适合需要稳定性的生产环境。
- **Ubuntu**：每6个月发布新版本，集成最新桌面工具（如Snap包管理）。

---

6. Linux成功的关键因素

- 技术优势：
  - **POSIX兼容**：无缝运行Unix软件（如Apache、MySQL）。
  - **硬件支持广泛**：从x86到ARM架构（树莓派、Android手机）。
- 开源生态：
  - **协作文化**：全球开发者贡献代码，快速修复漏洞（如Heartbleed漏洞48小时内修复）。
  - **商业支持**：红帽（IBM）、Canonical（Ubuntu）提供付费服务，推动企业采用。

行业影响：

- **云计算**：AWS、Azure 90%的实例运行Linux。
- **超级计算**：Top500超算100%使用Linux内核。

总结：Linux的启示

1. **开放协作的力量**：从个人项目到全球基础设施，证明开源模式的技术与社会价值。
2. **技术务实主义**：Linus的“先求有再求好”哲学，推动快速迭代与功能扩展。
3. **持续学习与社区参与**：开发者通过贡献代码、文档或测试，共同塑造技术未来。

思考题：

- 如果Linux闭源，今天的互联网生态会如何不同？
- 为何企业愿意付费支持开源软件（如购买RHEL订阅）？

三、Linux当前的角色

3.1 Linux 在企业环境中的应用

Linux 凭借其高稳定性、灵活性和开源生态，成为企业级应用的首选。

1. 网络服务器

- 核心优势：
  - 继承 Unix 的高稳定性，适合长期运行关键服务。
  - 支持主流网络协议（HTTP/HTTPS、FTP、SMTP 等）。
- 典型应用：
  - **Web 服务器**：Apache、Nginx（全球 35% 的网站使用）。
  - **邮件服务器**：Postfix、Sendmail。
  - **文件服务器**：Samba（兼容 Windows 文件共享）、NFS。
  - **数据库服务器**：MySQL、PostgreSQL（如阿里云 RDS 底层基于 Linux）。

2. 关键任务应用

- 金融行业：
  - 银行交易系统（如纽约证券交易所使用 Linux 处理每日数亿笔交易）。
  - 数据库管理（Oracle、IBM DB2 在 Linux 上优化运行）。
- 安全强化：
  - 防火墙（iptables、Firewalld）。
  - 入侵检测系统（Snort、Suricata）。

3. 高性能计算（HPC）

- 超级计算机：
  - 2023 年 Top500 超算 100% 运行 Linux（如美国 Summit 超算）。
  - 应用场景：气候模拟、基因测序、流体力学计算。
- 容器化与编排：
  - Kubernetes 管理大规模集群（如 Google Borg 系统的开源实现）。

3.2 Linux 在个人环境中的使用

从桌面到移动设备，Linux 无处不在。

1. 桌面电脑

- 图形环境：
  - **GNOME**：简洁现代（Ubuntu 默认界面）。
  - **KDE Plasma**：功能丰富（OpenSUSE 首选）。
- 生产力工具：
  - **办公套件**：LibreOffice（兼容 Microsoft Office 格式）。
  - **浏览器**：Firefox、Chromium。
  - **开发工具**：VS Code、Eclipse。

2. 手持设备与嵌入式系统

- 智能手机：
  - **Android**：基于 Linux 内核，全球 70% 手机使用（2023 年数据）。
  - **Ubuntu Touch**：开源移动操作系统（如 PinePhone）。
- 嵌入式设备：
  - **智能家居**：路由器（OpenWRT）、智能电视（三星 Tizen）。
  - **单板计算机**：树莓派（Raspberry Pi OS）、NVIDIA Jetson。

3.3、Linux 在云端运算中的核心地位

云计算与虚拟化依赖 Linux 提供基础设施。

1. 公有云与私有云

- 主流云平台
  ：AWS EC2 90% 实例运行 Linux。
  - Azure 和 Google Cloud 默认提供多种 Linux 发行版。
- 虚拟化技术：
  - **KVM**：Linux 原生虚拟化方案（Red Hat 主导）。
  - **Docker**：容器化技术（依赖 Linux 命名空间和 cgroups）。

2. 轻量级端点设备

- 边缘计算：
  - 工业物联网（IIoT）设备运行定制化 Linux（如 Ubuntu Core）。
  - 5G 基站使用 Linux 实现低延迟数据处理。

四、 如何高效学习

1、养成思考的好习惯

排错提升能力要远超于复现

尽量去看官方的文档，形成自己的东西

2、记得复习

出错的地方

博客上记录的，一定是自己需要的

3、放平心态

以为自己知道

以为自己不知道

以为自己不知道，实际自己知道

知道自己知道