



# Linux 内核分析与驱动编程

Jing Qi

jingqi@pku.edu.cn

School of Software and  
Microelectronics Peking University



PEKING  
UNIVERSITY



# Agenda

1. Course Introduction
2. Linux Community and OSS Introduction
3. Linux Kernel Introduction
4. Driver Introduction



PEKING  
UNIVERSITY



# 小调查

- 我用过以下linux系统:

A.Ubuntu

B.Deepin

C.CentOS

D.Mint

E.麒麟

F.其它发行版

G.没有用过



PEKING  
UNIVERSITY



# 1. Course Introduction



PEKING  
UNIVERSITY



# 基本信息

- 课程名称（包括英文名称）：Linux 内核分析与驱动编程(**Linux Kernel Analysis and Driver Development**)
- 课程编号：1620
- 课程类型：选修
- 所属学科：计算机科学与技术
- 领域方向：软件工程
- 学时和学分：48学时； 3学分
- 主讲教员：荆琦



PEKING  
UNIVERSITY



# 基本知识要求

- 选课同学“应该”学习过计算机相关的如下课程：
  - 数据结构
  - 操作系统原理
  - 计算机体系结构
  - C 语言
  - 汇编语言
- 如果我没有学过以上课程怎么办？
  - 不选这门课
  - 想学习，稍微学习一下即可





# 课程主要目标

- 以Linux为实例，掌握操作系统的设计原理
- 掌握Linux操作系统的体系结构、设计原理
- 能在Linux上开发相关驱动程序
- 为今后进行相关实习与工作打下基础



PEKING  
UNIVERSITY



# 课程内容

日期	知识模块	知识点
2月20日	课程介绍	<b>Linux</b> 内核基本结构、 <b>Linux</b> 的历史、开源协议介绍、驱动程序介绍
2月27日	实验课一	内核编译，内核补丁
3月5日	内核编程基础知识概述	内核调试技术、模块编程、 <b>linux</b> 启动过程、 <b>git</b> 简介
3月12日	实验课2	内核调试
3月19日	进程管理与调度	<b>Linux</b> 进程基本概念、进程的生命周期、进程上下文切换、 <b>Linux</b> 进程调度策略、调度算法、调度相关的调用
3月26日	实验课3	提取进程信息
4月2日	系统调用、中断处理	系统调用内核支持机制、系统调用实现、 <b>Linux</b> 中断处理、下半部
4月9日	实验课4	添加系统调用、显示系统缺页次数
4月16日	内核同步	原子操作、自旋锁、 <b>RCU</b> 、内存屏障等 <b>linux</b> 内核同步机制
4月23日	内存管理	内存寻址、 <b>Linux</b> 页式管理、物理页分配伙伴系统、 <b>Slab</b> 管理、进程地址空间
4月30日	实验课5	观察内存映射、逻辑地址与物理地址的对应
5月7日	文件系统	<b>Linux</b> 虚拟文件系统、 <b>Ext2/Ext3/Ext4</b> 文件系统结构与特性
5月14日	<b>Linux</b> 设备驱动基础字符设备驱动程序设计	<b>Linux</b> 设备驱动基础、字符设备创建和加载、字符设备的操作、对字符设备进行 <b>poll</b> 和 <b>select</b> 的实现、字符设备访问控制、 <b>IOCTL</b> 、阻塞 <b>IO</b> 、异步事件等
5月21日	基于 <b>linux</b> 的容器平台技术概述	虚拟化技术与容器、 <b>Docker</b> 概述、 <b>Kubernetes</b> 概述
5月28日	实验课6	<b>Docker</b> 对容器的资源限制
6月4日	报告课	期末课程报告





# 实验课相关介绍

本课程根据课程进度，会进行适当的实验课，带领大家熟悉Linux操作系统的原理与实践

## 实验一：内核编译、添加内核补丁

- 动手编译Linux内核并安装新内核
- 在Ubuntu上熟悉并了解给内核添加补丁的过程

## 实验二：内核调试

- 在Ubuntu上实现使用对linux内核的调试，并完成对某个变量（configured）的watch，并显示出来





## 实验三：提取进程信息

- 实现一个内核模块，提取系统中所有进程的pid、state和名称进行显示

## 实验四：显示系统缺页次数

- 结合proc文件系统和模块编程，显示系统启动以来的缺页次数。

## 实验五：观察内存映射、获取物理地址

- 观察程序在内存中的映射
- 获取C程序中一个逻辑地址对应的物理地址





# 课件访问方式

- 北大教学网
- 讲义内容来源：
  - 教材资料
  - 网络资源
  - 实践总结



PEKING  
UNIVERSITY



# 教材及参考书

- 《**Understanding the Linux Kernel, 3rd Edition**》 作者: **Daniel P. Bovet, Marco Cesati** 出版社: **O'Reilly** 出版日期: **2005年12月**
- 《**Professional Linux Kernel Architecture**》 作者: **Wolfgang Maurer** 出版社: **Wiley** 出版日期: **2008年10月**
- 《**Linux技术内幕**》 作者: **罗秋明** 出版社: **清华大学出版社** 出版日期**2017年1月**
- 《**奔跑吧, Linux内核**》 作者: **张天飞** 出版社: **人民邮电出版社** 出版日期: **2017年9月**



PEKING  
UNIVERSITY



# 教材及参考书

- **《Linux 驱动程序设计》**，作者：**Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman**，魏永明,耿岳,钟书毅 译 语种：汉语 出版社：中国电力出版社 出版日期：**2005年12月**
- **《Essential Linux Device Drivers》** 作者：**Sreekrishnan Venkateswaran** 出版社：**Pearson Education Asia** 出版日期：**2008年5月**
- **《Linux设备驱动开发详解：基于最新的Linux4.0内核》** 作者：宋宝华 出版社：机械工业出版社 出版日期：**2015.8**





- 请您到学院ftp或者网络上下载本课需要的教材&参考书电子版！



PEKING  
UNIVERSITY





# 学期成绩的评定

考查课：无笔试

内容：

1) 平时成绩    2) 课程报告

成绩分布

• 平时成绩（**60%**）：独立完成；提出并解决问题

- 实验课：课堂表现+课后实验报告（一周之内及时提交）
- 课后作业：及时提交（一周之内），有自己的思考

• 期末课程报告（**40%**）：

- 相关材料（文档、视频等）提交（**20%**）：**5月28日之前**提交，有自己的见解
- 课堂报告（**20%**）：**5月28日之前**提交演示材料，要求思路清晰，课堂表达清楚



PEKING  
UNIVERSITY



# 学期成绩的评定

期末课程报告注意事项：

- 可以**参考**网络资料，但是不可大量原文、原码拷贝
- 注意内核版本**5.0**以上
- 课程报告为了更加深入，分组做
  - 视工作量分组，可一人或小组
  - 注意工作量，保证每个人都足够
  - 注意写清楚组内分工，以及组内互评分
  - 现场提问，注意基本知识的掌握
- 内容要求：
  - 主要工作与**Linux**内核相关，可以是源代码分析或者实验及结果分析
  - 代码分析：基本原理介绍、主要数据结构概述、代码概述（鼓励自己画图表述思路）、代码行注释、分析结果验证、问题及解决



PEKING  
UNIVERSITY



# 实验课要求

- 实验课一人一组独立完成实验
- 每次有一份参考代码，但是每个人都要自己编写代码并且做上注释，实验报告里要写明实验原理
- 实验课报告提交规格：第x次实验-学号-姓名，例如“第3次实验-12345698-张三”
- 如果实在调试不出来，可以将问题解决的过程写出来，**真实可靠的调试过程也代表了工作量！**



# 助教信息



姓名: 郭琪

专业: 软件工程


方向: 网络与系统安全

手机: 17611520935

邮箱: gq1378@pku.edu.cn

助教微信:



Amos   
Beijing Daxing



Scan the QR Code to add me on WeChat



PEKING  
UNIVERSITY



# 教学方法

- 理论与实践结合
- 自顶向下教学，即先理论基础，然后具体剖析内核源代码
- 课上只是就**Kernel**中的比较“大”的某点，给出所谓的“分析”，不是全面、详细剖析内核
- 课上学习时间+课下学习，主要关注源代码的阅读和实验项目的实现，所以本课需要花费一定的时间
- 本课不是讲授“**操作系统设计**”。
- 利用好网络资源







# 教学辅导

- 有问题的时候解决方法：
  - 北大教学网中课程讨论版
  - 看教材&参考书中是否论述
  - 社区（国内，国外），邮件列表
  - 上网搜索
  - 相关会议
  - 如果最后也无法找到答案，努力钻研源代码，争取给老师以及其他同学给出一个合理的解答







# FAQ

- 我在Linux下曾经有内核&驱动方面的开发经验，是否应该选修这门课？
  - 既然已经开发过相关的东西，就不要选修了！
  - 本课的对象是：初学者
- 请推荐几本书吧？
  - 本课的教材&参考书即可





# 学习建议

- 重视理论知识
- 重视英文文献的阅读能力培养
- 重视中文图书的阅读方法及毅力培养
- 重视动手能力培养
  - 一句话：说永远比做要容易，不要光说不做



PEKING  
UNIVERSITY



## 2. Linux Community and OSS Introduction



PEKING  
UNIVERSITY



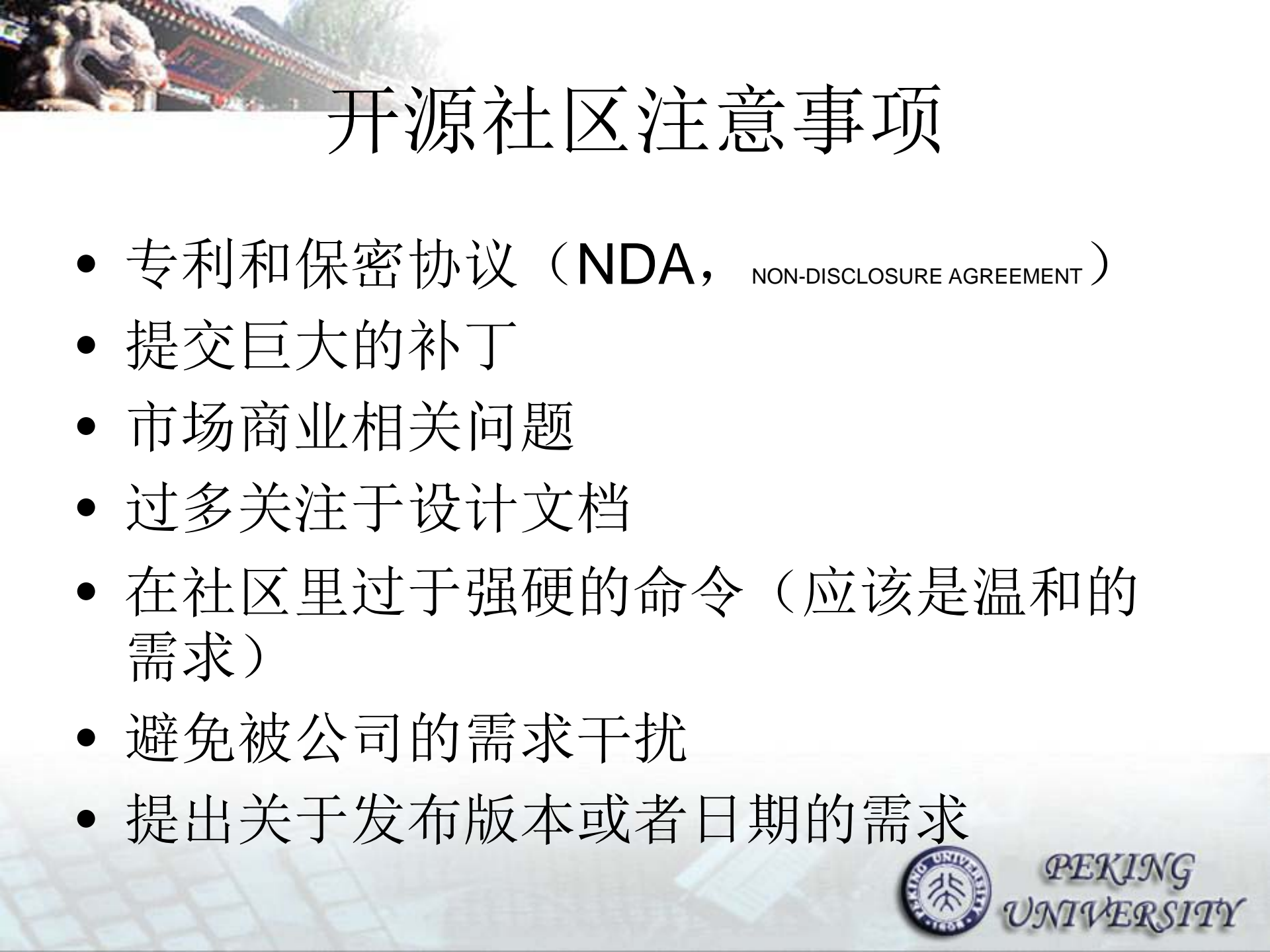
# Linux开源文化

- 开源开放
- 好的想法和好的代码会受到关注并有所回报
- 和市场需求关系不明显，技术驱动
- 不能只获取也要有贡献
  - **Linux**仍然是**GPL** (分发时)
  - 可以提高软件质量，让新的特性被更多人使用和理解
- 代码提交要遵守特定的规范和标准（如**POSIX**）
- 软件兼容性和跨平台特性
- 不规范的设计和开发（项目计划和设计文档都不规范也不充分）
- 新的想法用代码体现而不是规范或者需求
- **RERO: Release Early, Release Often**，对于所有的**comments, help, testing, community acceptance**
- 频繁持续的代码检查和开发

Have fun!!



PEKING  
UNIVERSITY



# 开源社区注意事项

- 专利和保密协议（**NDA**, NON-DISCLOSURE AGREEMENT）
- 提交巨大的补丁
- 市场商业相关问题
- 过多关注于设计文档
- 在社区里过于强硬的命令（应该是温和的需求）
- 避免被公司的需求干扰
- 提出关于发布版本或者日期的需求



PEKING  
UNIVERSITY



# 社区角色

- 等级松散，不明确
- 内核维护者（maintainer）：
  - Linus and Andrew Morton
- 稳定版维护者（stable）：
  - Greg Kroah-Hartman and Chris Wright
- 体系结构和子系统维护者（Arch and subsystem）
- 底层维护者

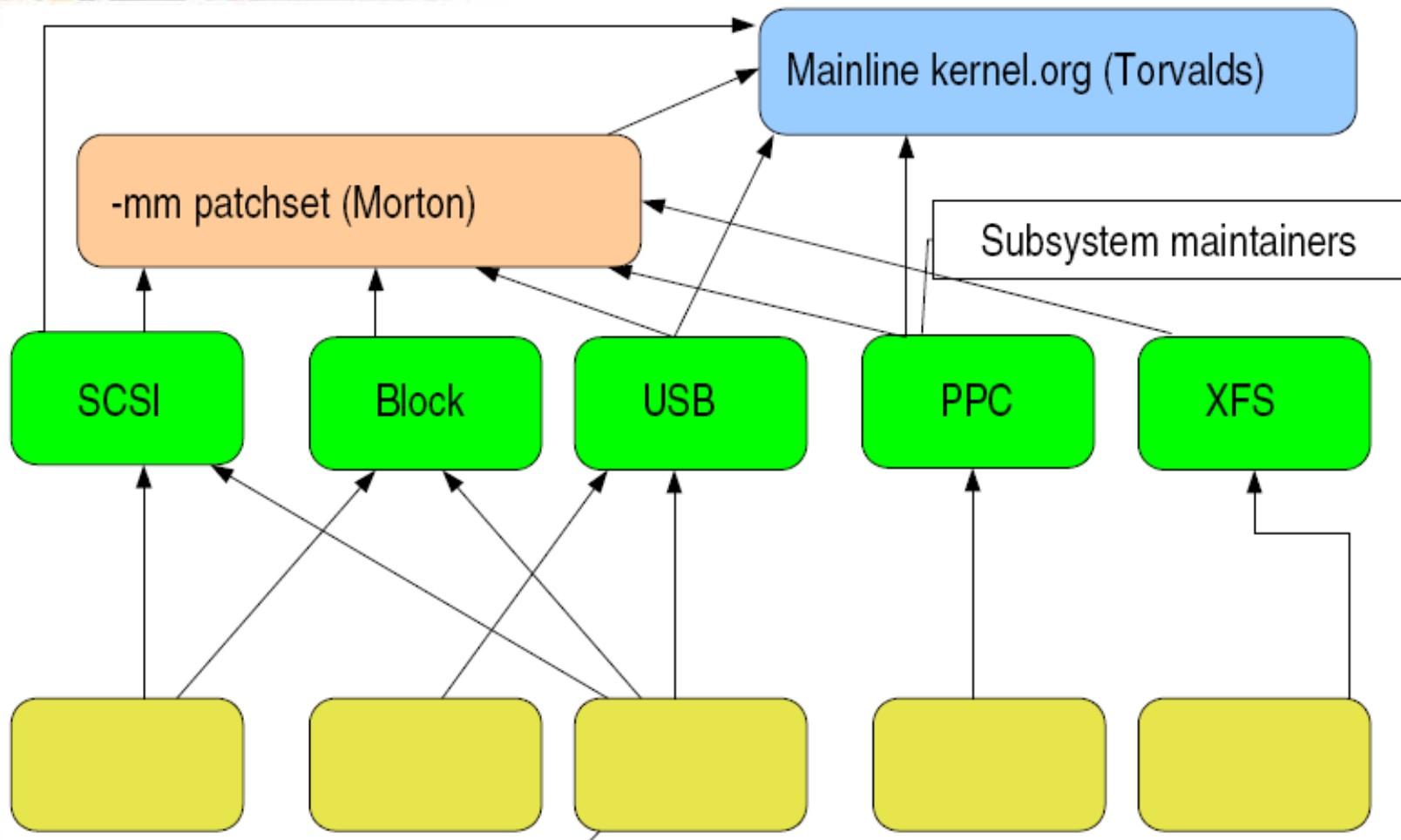
Toplevel maintainers are gatekeepers



PEKING  
UNIVERSITY



# Merges to mainline

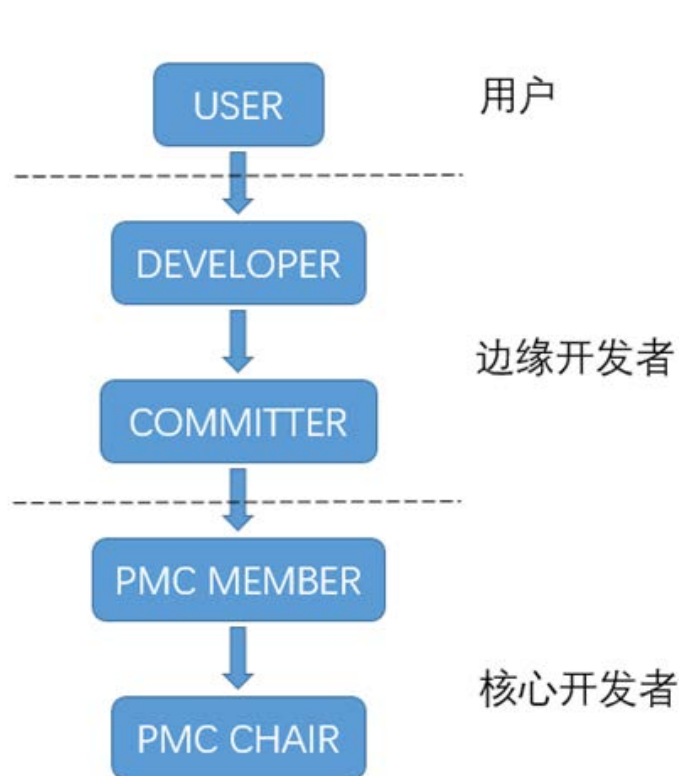


**-mm patchset: review/test here before merge into mainline**

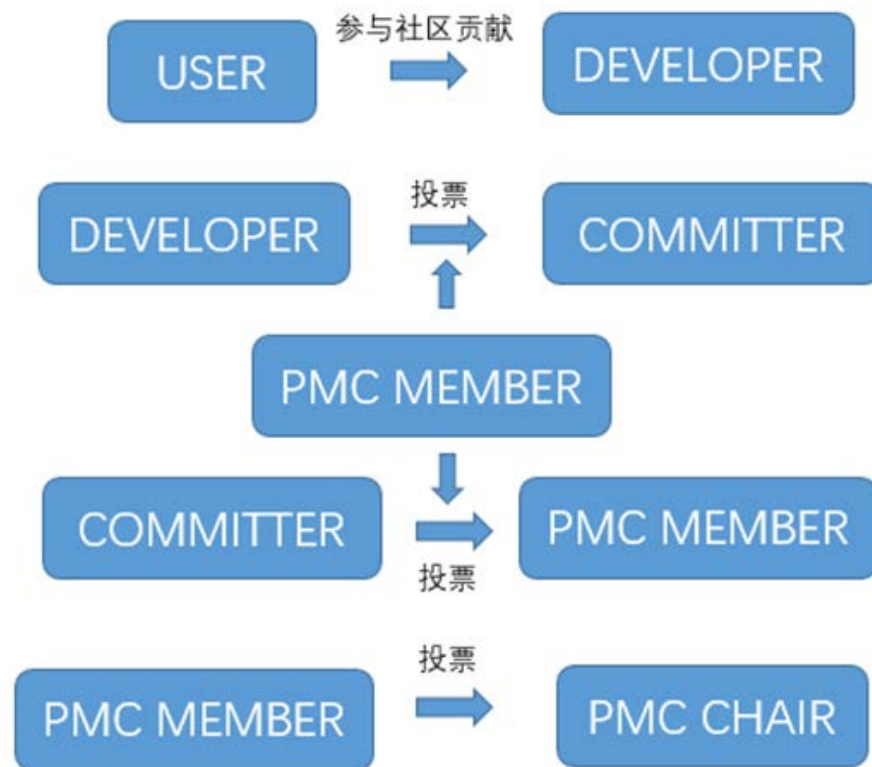


TSINGHUA UNIVERSITY

# 其他经典社区角色-Apache



Apache社区的角色层次



Apache社区的角色转换

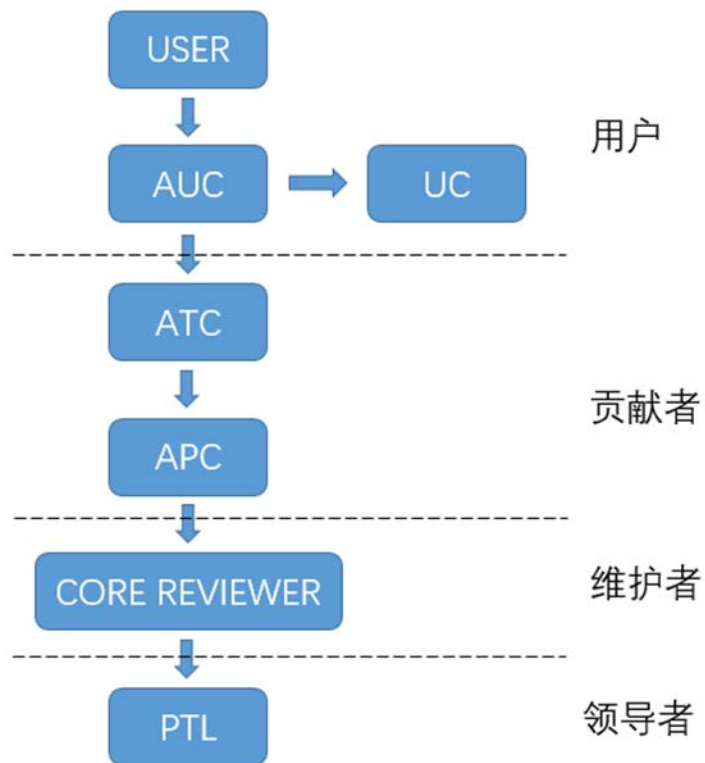


PEKING  
UNIVERSITY

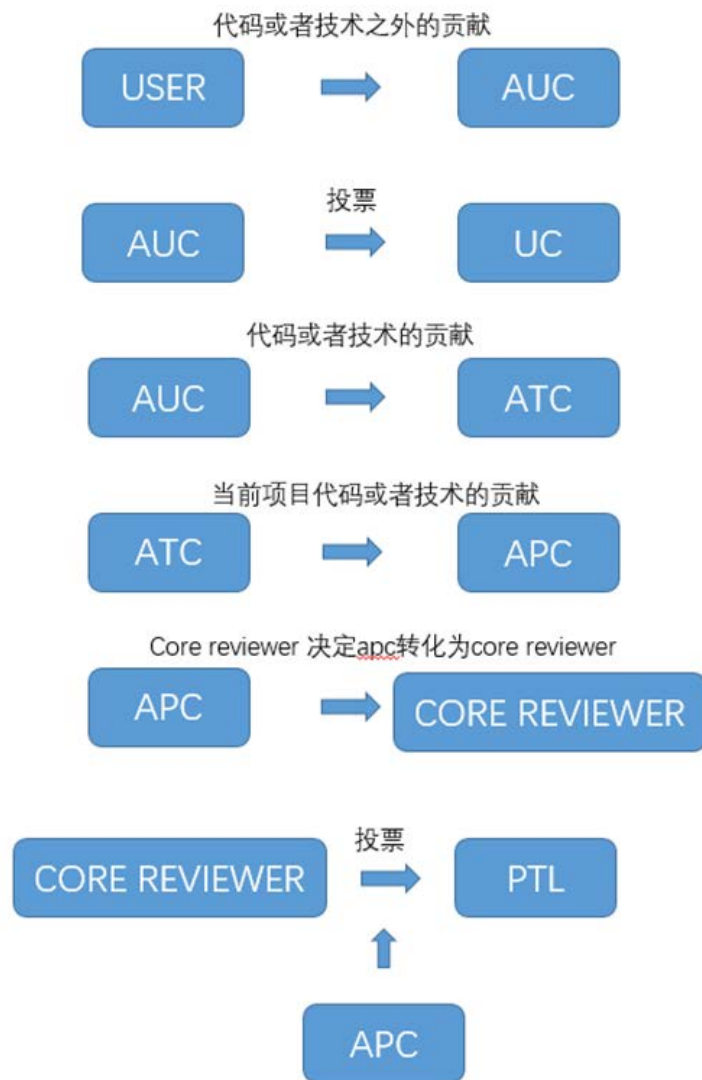




# Openstack



Openstack社区角色



Openstack社区的角色转换



PEKING  
UNIVERSITY



# 社区交流方式

- 邮件列表和文档
- 项目主页
- IRC频道
- 开发者会议
  - kernel summit, OLS, Linux\* Conf au, LinuxWorld, Linux\* Bangalore
  - 中国Linux内核开发者大会（CLK, China Linux Kernel Developer Conference）



PEKING  
UNIVERSITY



# 推荐的内核邮件列表

- Etiquette  
(<http://www.arm.linux.org.uk/armlinux/mletiquette.php>)
- LKML aka linuxkernel (@vger.kernel.org)
- LKML FAQ at <http://www.tux.org/lkml/>
- Index: <http://vger.kernel.org/vger-lists.html>
- Other subsystem mls
  - <http://kernelnewbies.org>
  - <http://janitor.kernelnewbies.org>
  - <http://www.kerneltraffic.org> summaries
  - <http://lwn.net/> summaries

**<http://vger.kernel.org/vger-lists.html>**



PEKING  
UNIVERSITY





# 如何做贡献

- 注意代码风格(参考Linux/Documentation/CodingStyle)
- 将**patches**发送给各级维护者（子系统维护者或者底层维护者）或者邮件列表
- 补丁（**patch**）中除了补丁代码之外，还应该包括特性的理由和解释
- 尽量使用**DCO**（Developer Certificate of Origin，非**CLA**）：  
**Signed-off-by: Your Name <[your.name@example.com](mailto:your.name@example.com)>**
- 每个邮件最好只包含一个补丁，不能是巨型补丁，也不要压缩
- 补丁里不要包括太多种内容
- 不要附件的形式发送补丁，最好将补丁直接内嵌入邮件（可以使用某些邮件客户端）
- 其他补丁相关事宜请参考：  
**Linux/Documentation/SubmittingPatches**

**Bug tracking (<http://bugme.kernel.org>)**



PEKING  
UNIVERSITY



# Linux内核开发的流程

- 每两三个月为一个周期，发布新的内核
- 周期的前两周接受新的**feature**，称为**merge window**
- **Merge window**关闭以后，生成版本-rc1
- 后面开始大量的测试，每周发布新版本-rcX
- 厂商维护自己的稳定版本





# 举例：阿里内核的发展历程

- 以RHEL6-2.6.32为基础，结合阿里业务特点进行优化和特性开发
- 2010年底内核组成立
- 2011年中发布第一个内核版本
- 每隔半年发布一个稳定版本
- 同时保持和redhat同步
- 14年版本base为2.6.32-358





# Open Source Licenses

<https://opensource.org/>



PEKING  
UNIVERSITY



# Monetizing Software: An Evolutionary History

- 1970's: software was not recognized as a “Property”
  - Value was mixed with hardware
  - The labor force used in software development was counted as a cost to the tangible product
  - Software Developer: “Labor Provider” , not “Property Creator”
  - Law: mostly a kind of “know-how”
- Late 1970's~Early 1980's: legislations changed the story
  - Many countries began to recognize it's copyrightability
  - e.g, CONTU report
  - Legally, it became a “Property”





# Monetizing Software: An Evolutionary History

- 1980's: software was recognized as a “Property”
  - Value separated from hardware
  - An independent business
  - Software company: a new sort of “Property Lease Company”
    - What: Software
    - How: Granting Licenses
    - Specialty: multiple leasees to same property
  - Open source movement kick-off
    - 1983, GNU programme established
    - 1985, Free Software Foundation Estbalished
    - 1989, GPL v1 Published

GPL: GNU GENERAL PUBLIC LICENSE



PEKING  
UNIVERSITY





# Monetizing Software: An Evolutionary History

- 1990's: software industry boomed
  - Proprietary software licensing became most profitable business in history
  - Open Source movement continues to grow
    - 1991, GPL v2 published; Linux kernel published;
    - 1995, Apache HTTP server launched
    - 1997, Eric Raymond, The Cathedral and the Bazaar
    - 1998, OSI established
  - Opensource software companies emerged
  - Proprietary software company: look opensource as an enemy
    - **“Linux is a cancer”**--Steve Ballmer, 2001

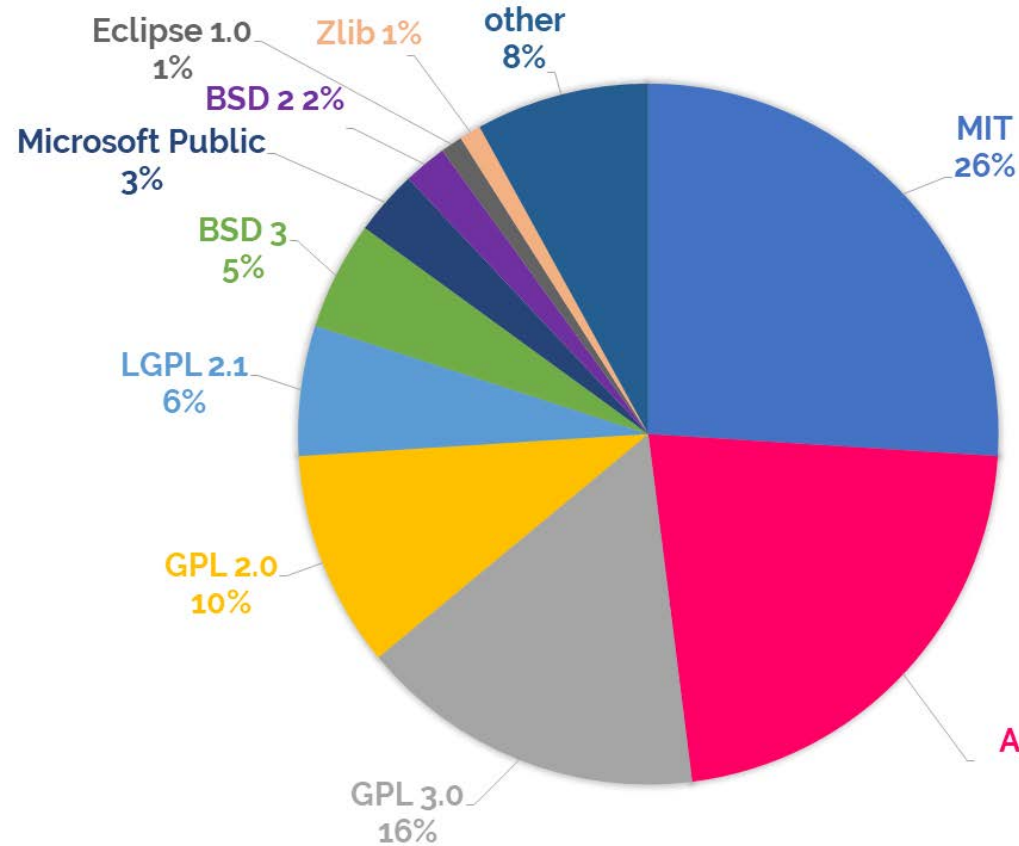




# Monetizing Software: An Evolutionary History

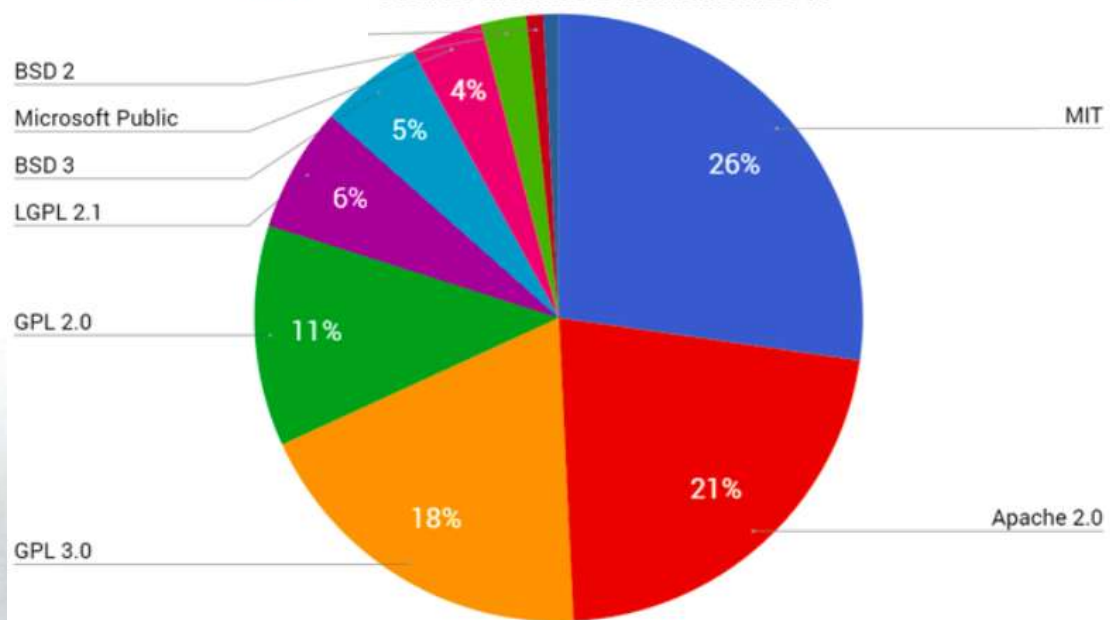
- After 2000's: “We love open source!”
  - Proprietary software companies embraced open source
    - 2006, Microsoft opened Microsoft Linux and open source labs
  - Internet giants use a lot of open source software
    - On the **servers**
    - Linux based **mobile** operating systems
- **“We love open source!”**--Steve Ballmer, 2010
  - Why not!
  - As long as you understand how to monetize open source software.
  - All in all, WHO DON'T LOVE MONEY?



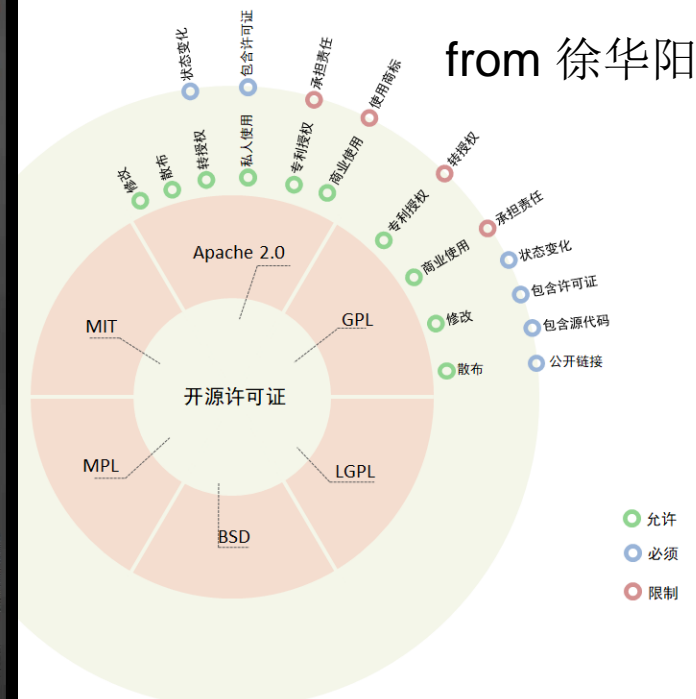
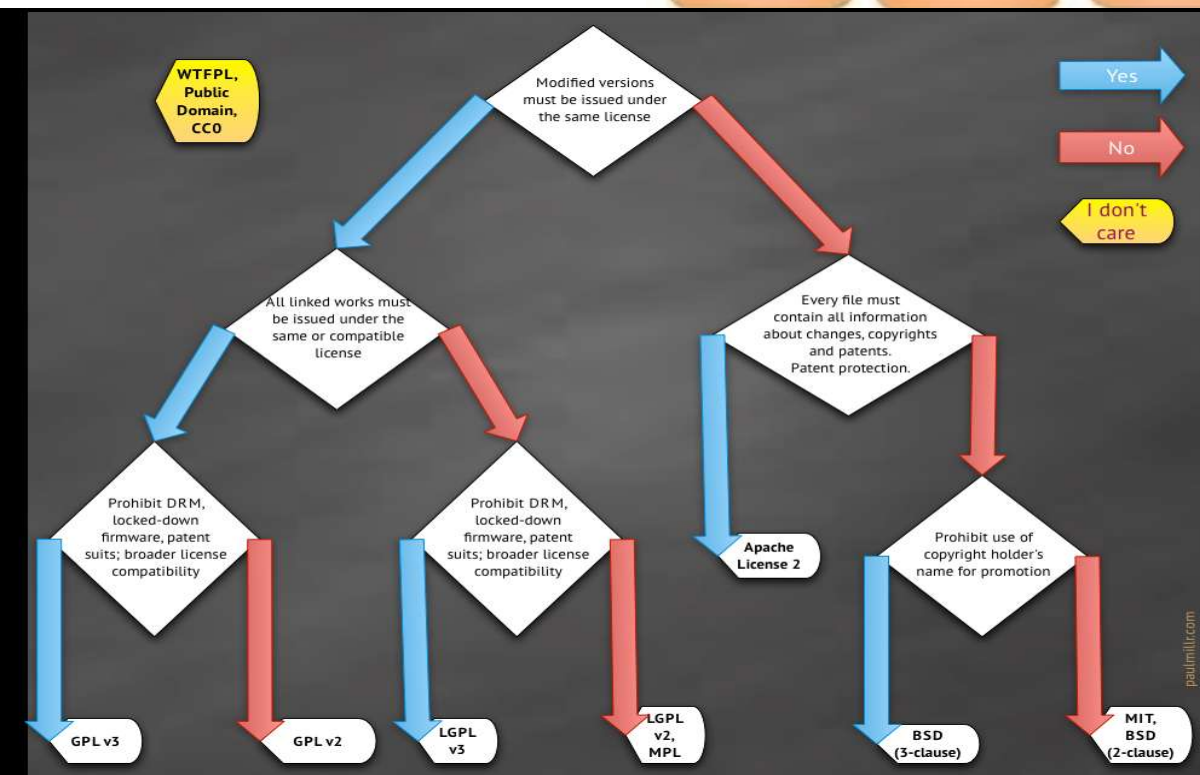
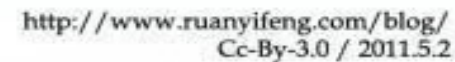


<https://opensource.org/>

Source License Distribution 2017



更详细的开源  
licenses介绍



# 原始程序



您要求他人再散布程序时必须提供原始码吗？

您允许他人对程序进行再授权吗？

您愿意将程序所包含的专利授权出来吗？

若是散布程序时，不包含原始码在内，而是另外提供原始码，此时您允许收取散布原始码的费用可以高于散布成本？

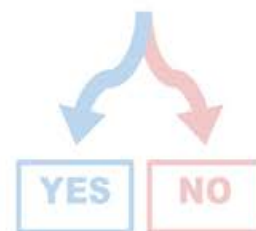


from 徐华阳





# 修改程序



您允許程序被修改后使用不同的授权条款吗

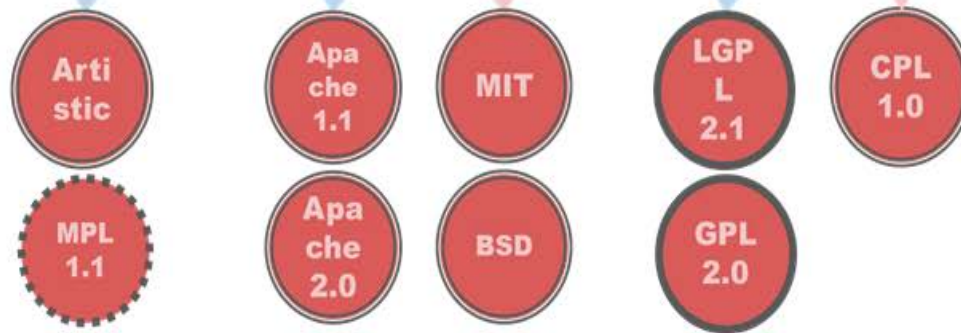
您要求程序被修改后必须公开原始码吗？

您要求后续修改者必须修改程序时必须附加修改文件吗

我希望使用者选择我所指定的授权条款

使用者只能采用原本的授权条款

只要不和原本采用的授权条款相违背。使用者可以自行决定授权内容







# 3. Linux Kernel Introduction



PEKING  
UNIVERSITY



## 3. Linux Kernel Introduction

3.1 Linux发展的五大支柱

3.2 内核发展趋势及特点

3.3 Linux发行版及软件包管理

3.4 内核源代码结构

3.5 内核编译介绍



PEKING  
UNIVERSITY



# 关于 Linux

- 严格来讲，Linux这个词本身只表示Linux内核
- 实际上Linux用来形容整个基于Linux内核，并且使用GNU 工程各种工具和数据库的操作系统(也被称为GNU/Linux)。





# Monolithic Kernel Versus Microkernel Designs

- Operating kernels can be divided into two main design camps: the monolithic kernel and the microkernel. （操作系统内核分为两大阵营：宏内核和微内核）





# Monolithic kernels (LKD)

- Monolithic kernels are implemented entirely as **single large processes running entirely in a single address space**. Consequently, such kernels typically exist on disk as single static binaries.
- **All kernel services** exist and execute in the large kernel address space.
- **Communication within the kernel** is trivial because everything runs in kernel mode in the same address space: The kernel can invoke functions directly, as a user-space application might.
- Proponents of this model cite the **simplicity** and **performance** of the monolithic approach. Most Unix systems are monolithic in design.





# Microkernels (LKD)

- The functionality of the kernel is broken down into **separate processes**, usually called servers.
- Idealistically, only the servers absolutely requiring such capabilities run in a privileged execution mode. **The rest of the servers run in user-space.** All the servers, though, are kept separate and **run in different address** spaces.
- Direct function invocation as in monolithic kernels is not possible. Instead, communication in microkernels is handled via message passing: An interprocess communication (**IPC**) mechanism is built into the system, and the various servers communicate and invoke "services" from each other by sending messages over the IPC mechanism.
- The separation of the various servers prevents a failure in one server from bringing down another.







# 3. Linux Kernel Introduction

## 3.1 Linux发展的五大支柱

- \* UNIX
- \* MINIX
- \* GNU计划及自由软件
- \* POSIX标准

## 3.2 内核发展趋势及特点

## 3.3 Linux发行版及软件包管理

## 3.4 内核源代码结构

## 3.5 内核编译介绍





# UNIX

- 1965 MIT Bell GE → Multics
- 1969 Ken.Thompson (Ken) → Unics
- 1973 Dennis Ritchie (DMR) → C 语言

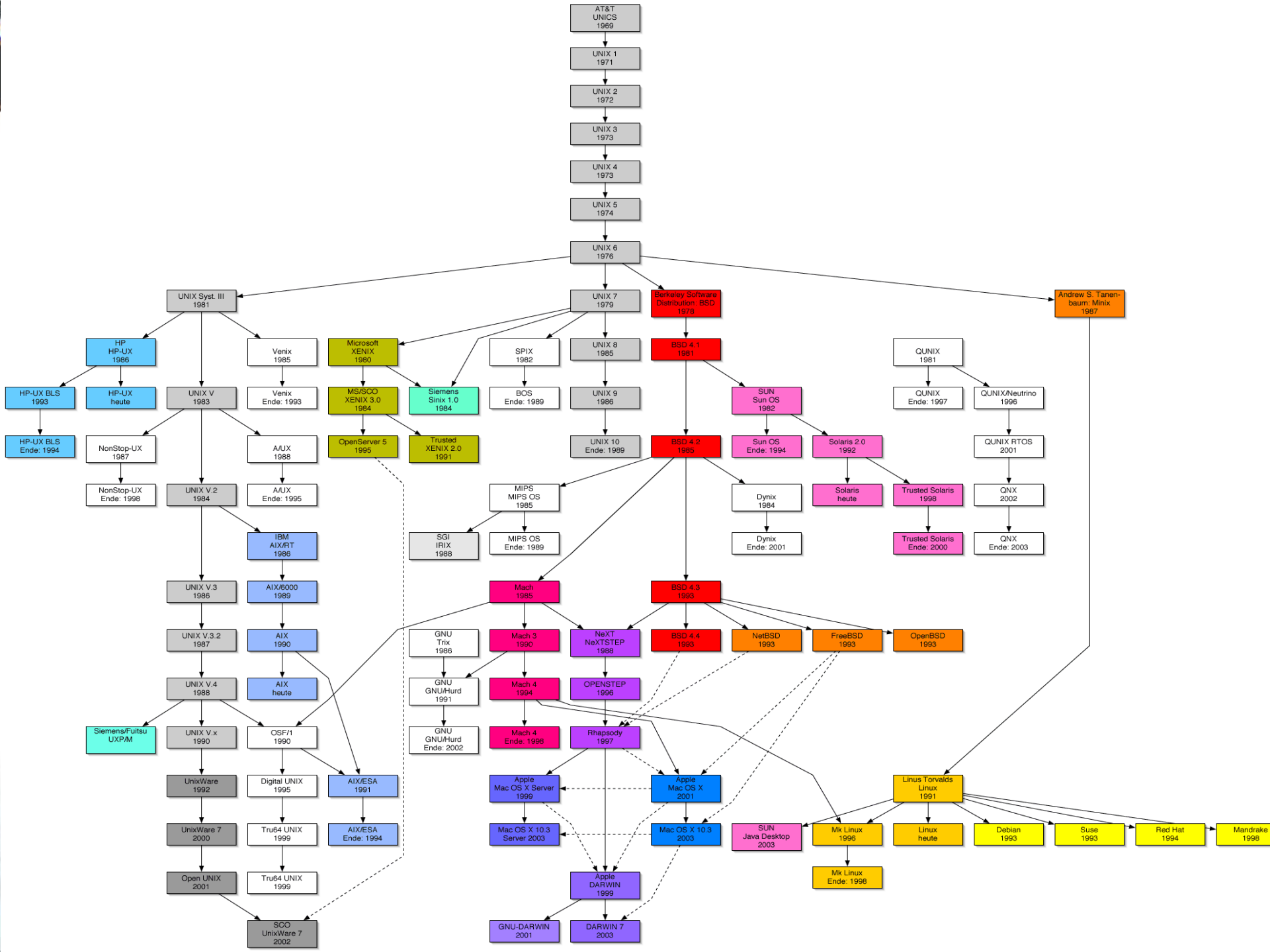


UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity.

[Unix家族演化](http://www.levenez.com/unix/): <http://www.levenez.com/unix/>



PEKING  
UNIVERSITY





# MINIX

- 1979 AT&T → Unix V7 版权声明
- 1984-1986 荷兰 Andrew S. Tanenbaum (AST) → Minix
  - AST 是在荷兰Amsterdam 的Vrije 大学数学与计算机科学系统工作，是ACM 和IEEE 的资深会员
  - Unix like
  - 微内核
  - 主要用于教学



PEKING  
UNIVERSITY



# Linux

- Creator: Linus Torvalds, Finland



1991 年8月25日 *comp.os.minix*

Hello everybody out there using minix-

I'm doing a (free) operation system (**just a hobby, won't be big and professional like gnu**) for 386(486) AT clones. ....

I've currently ported bash (1.08) and gcc (1.40), and things seem to work. This implies that i'll get something practical within a few months, and I'd like to know what features most people want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus ([torvalds@kruuna.helsinki.fi](mailto:torvalds@kruuna.helsinki.fi))

Ps. Yes - it's free of any minix code, and it has a multi-threaded fs. **It is NOT portable (uses 386 task switching etc) . and it probably never will support anything other than AT-hard-disks**, as that's all I have: -(



PEKING  
UNIVERSITY





# 3. Linux Kernel Introduction

## 3.1 Linux发展的五大支柱

- \* UNIX
- \* MINIX
- \* **GNU**计划及自由软件
- \* POSIX标准

## 3.2 内核发展趋势及特点

## 3.3 Linux发行版及软件包管理

## 3.4 内核源代码结构

## 3.5 内核编译介绍



# GNU 计划

- 1984 Richard M. Stallman (RMS) → GNU 计划和自由软件基金会(the Free Software Foundation - FSF)
- 开发一个类似 Unix、并且是自由软件的完整操作系统：GNU 系统
- GNU 是 “GNU’s Not Unix”的递归缩写
- 到上世纪90 年代初，GNU 项目已经开发出许多高质量的软件
  - Emacs, bash shell, gcc, gdb, glibc.....
- GPL(General Public License) & copyleft
- GNU 的操作系统HURD一直在开发中





自由软件基金会认为，自由软件应保护用户的如下四大自由：

- **运行**任何程序实现任何目的的自由；
- 研究程序如何工作并按个人需要**修改**的自由。能够获取源代码是其先决条件
- **分发拷贝**以便帮助身边其他人的自由，以及
- **改进程序并向公众发布的自由**，以便让整个社群受益。能够获取源代码是其先决条件

<http://www.gnu.org/licenses/copyleft.html>

所谓**Copyleft**是指任何人都可以重新分发软件，不管有没有进行修改，但必须同时保留软件所具有的自由特性。**Copyleft**是为了保证所有用户都拥有自由。



PEKING  
UNIVERSITY



# 3. Linux Kernel Introduction

## 3.1 Linux发展的五大支柱

- \* UNIX
- \* MINIX
- \* GNU计划及自由软件
- \* **POSIX标准**

## 3.2 内核发展趋势及特点

## 3.3 Linux发行版及软件包管理

## 3.4 内核源代码结构

## 3.5 内核编译介绍





# POSIX 标准

- POSIX(Portable Operating System Interface for Computing Systems)是由IEEE 和ISO/IEC 开发的一簇标准。
- 该标准是基于现有的UNIX 实践和经验，描述了**操作系统的调用服务接口**，用于保证编写的应用程序可以在源代码一级上在多种操作系统上移植运行
- 90 年代初，POSIX 标准的制定正处在最后投票敲定的时候。此时正是Linux刚刚起步的时候，这个UNIX 标准为Linux 提供了极为重要的信息，使得Linux 的能够在标准的指导下进行开发，**能够与绝大多数UNIX 系统兼容**







# 3. Linux Kernel Introduction

## 3.1 Linux发展的五大支柱

## 3.2 内核发展趋势及特点

- \* 内核源文件
- \* Linux内核发展趋势及特点
- \* Android内核特点

## 3.3 Linux发行版及软件包管理

## 3.4 内核源代码结构

## 3.5 内核编译介绍





# 内核源文件

## The Linux Kernel Archives



[About](#) [Contact us](#) [FAQ](#) [Releases](#) [Signatures](#) [Site news](#)

Protocol	Location
<a href="#">HTTP</a>	<a href="https://www.kernel.org/pub/">https://www.kernel.org/pub/</a>
<a href="#">GIT</a>	<a href="https://git.kernel.org/">https://git.kernel.org/</a>
<a href="#">RSYNC</a>	<a href="rsync://rsync.kernel.org/pub/">rsync://rsync.kernel.org/pub/</a>

Latest Stable Kernel:



5.5.4

mainline:	5.6-rc1	2020-02-10	<a href="#">[tarball]</a>	<a href="#">[patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a>
stable:	5.5.4	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	5.4.20	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	4.19.104	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	4.14.171	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	4.9.214	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	4.4.214	2020-02-14	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	3.16.82	2020-02-11	<a href="#">[tarball]</a>	<a href="#">[pgp]</a> <a href="#">[patch]</a> <a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
linux-next:	next-20200214	2020-02-14				<a href="#">[browse]</a>

### Other resources

### Social

Linux 内核源代码  
的官方网站:  
**www.kernel.org**

# uname -a

```
root@ubuntu:~# uname -a
Linux ubuntu 3.4.106 #1 SMP Mon Mar 23 21:32:30 CST 2015 x86_64 x86_64 x86_64 GN
U/Linux
root@ubuntu:~#
```

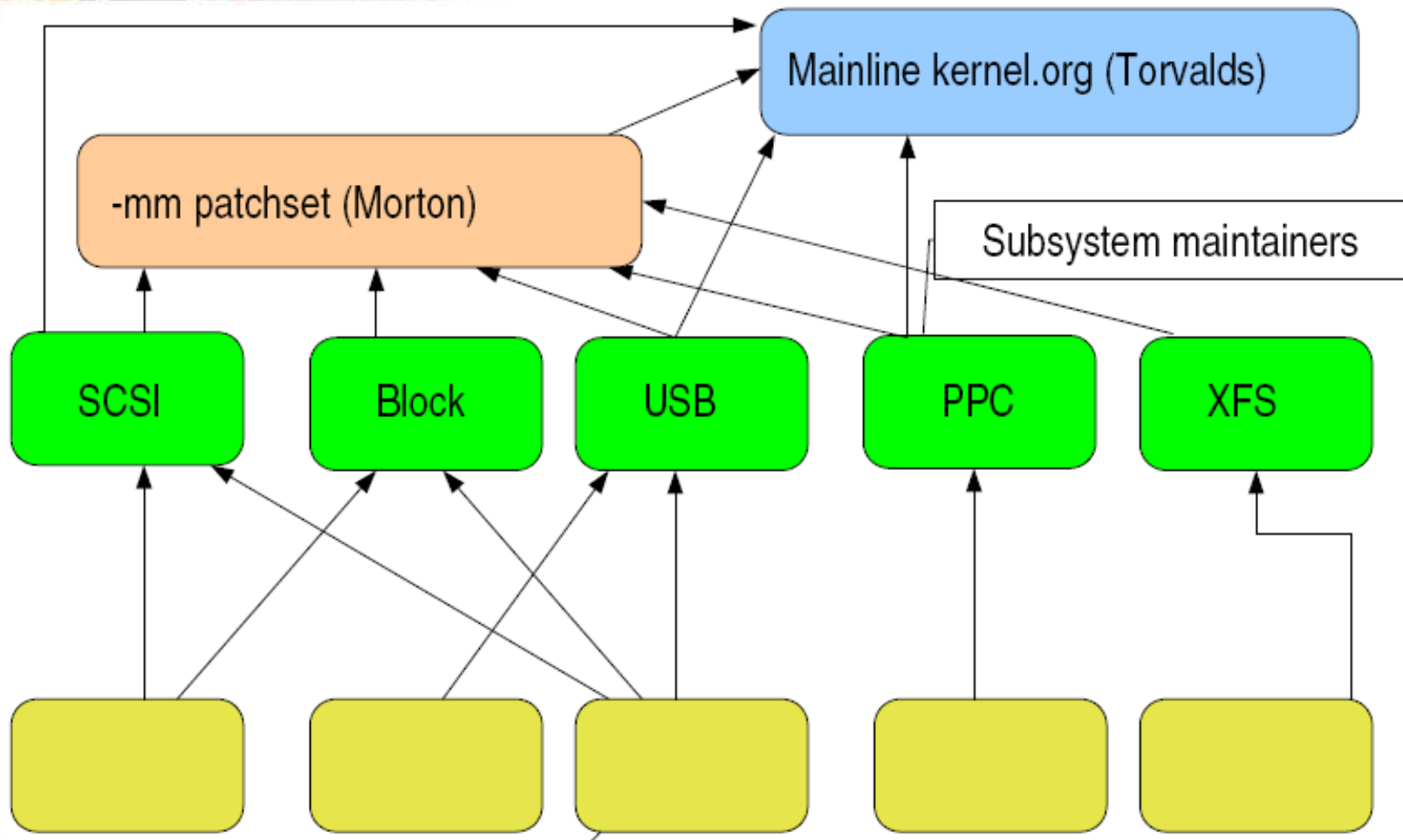


# Kernel releases

- **Prepatch:** Prepatch or "RC" kernels are mainline kernel pre-releases that are mostly aimed at other kernel developers and Linux enthusiasts. They must be compiled from source and usually contain new features that must be tested before they can be put into a stable release. Prepatch kernels are maintained and released by **Linus Torvalds**.
- **Mainline:** Mainline tree is maintained by **Linus Torvalds**. It's the tree where all new features are introduced and where all the exciting new development happens. New mainline kernels are released **every 2-3 months**.
- **Stable:** After each mainline kernel is released, it is considered "stable." Any **bug fixes** for a stable kernel are backported from the mainline tree and applied by a designated stable kernel maintainer. There are usually only a few bugfix kernel releases until next mainline kernel becomes available -- unless it is designated a "longterm maintenance kernel." Stable kernel updates are released on as-needed basis, usually once a week.
- **Longterm:** There are usually several "longterm maintenance" kernel releases provided for the purposes of backporting bugfixes for **older kernel trees**. Only important bugfixes are applied to such kernels and they don't usually see very frequent releases, especially for older trees.



# Merges to mainline



**-mm patchset: review/test here before merge into mainline**

# Linux内核发展趋势及特点

- 0.00 (1991.2-4?) 两个进程分别显示AAA BBB
- 0.01 (1991.9.17) 10,239 lines of code
- 0.02 (1991.10.5)
- 0.10 (1991.10) 由Ted Ts'o 发布的Linux 内核版本
- 0.11 (1991.12.8) 基本可以正常运行的内核版本。
- 0.12 (1992.1.15) 主要加入对数学协处理器的软件模拟程序，
- 0.95 (0.13) (1992.3.8) 开始加入虚拟文件系统思想的内核版本。
- 0.96 (1992.5.12) 开始加入网络支持和虚拟文件系统VFS。
- 0.97 (1992.8.1)
- 0.98 (1992.9.29)
- 0.99 (1992.12.13)
- 1.0 (1994.3.14) 176,250 lines of code
- 1.2.0 (1995.3.7) 310,950 lines of code
- 2.0 (1996.2.9)
- 2.2.0 (1999.1.26) 1,800,847 lines of code
- 2.4.0 (2001.1.4) 3,377,902 lines of code, 进一步提升了SMP的扩展性，集成了很多用于支持桌面系统的特性（USB、PCMCIA的支持，内置的即插即用等）
- 2.6.0 (2003.12.17) 5,929,913 lines of code
- 3.0 (2011.7.22) 近1500万行



Linux					
Version	Files <sup>1</sup>	Source lines <sup>2</sup>	Days	Commits <sup>3</sup>	Changes <sup>4</sup>
<a href="#">2.6.31</a>	29111	12046317 (10778469)	92	10883	8938 files changed; 914135 insertions(+); 504980 deletions(-)
<a href="#">2.6.32</a>	30485	12606910 (11242136)	84	10998	10315 files changed; 1092987 insertions(+); 530428 deletions(-)
<a href="#">2.6.33</a>	31565	12990041 (11564768)	83	10871	9673 files changed; 859458 insertions(+); 479452 deletions(-)
<a href="#">2.6.34</a>	32297	13320934 (11861616)	82	9443	11154 files changed; 609584 insertions(+); 278958 deletions(-)
<a href="#">2.6.35</a>	33316	13545604 (12250679)	77	9801	8889 files changed; 691927 insertions(+); 467252 deletions(-)
<a href="#">2.6.36</a>	34301	13499457 (12539782)	80	9501	9202 files changed; 582139 insertions(+); 628362 deletions(-)
<a href="#">2.6.37</a>	35191	13996612 (13006967)	76	11446	11104 files changed, 1093202 insertions(+), 598350 deletions(-)
<a href="#">2.6.38</a>	35877	14294439 (13294464)	69	9542	9133 files changed, 747809 insertions(+), 455603 deletions(-)
<a href="#">2.6.39</a>	36719	14619185 (13605251)	65	10268	10985 files changed, 847537 insertions(+), 523387 deletions(-)

<sup>1</sup> `find . -type f -not -regex '\.\/.git\/.*' | wc -l`

<sup>2</sup> (Without documentation): `find . -type f -not -regex '\.\/.git\/.*' | xargs cat | wc -l` (`find . -name *. [hcS] -not -regex '\.\/.git\/.*' | xargs cat | wc -l`)

<sup>3</sup> `git-log --no-merges --pretty=oneline v2.6.(x-1)..v2.6.(x) | wc -l`

<sup>4</sup> `git diff --shortstat v2.6.(x-1)..v2.6.(x)`

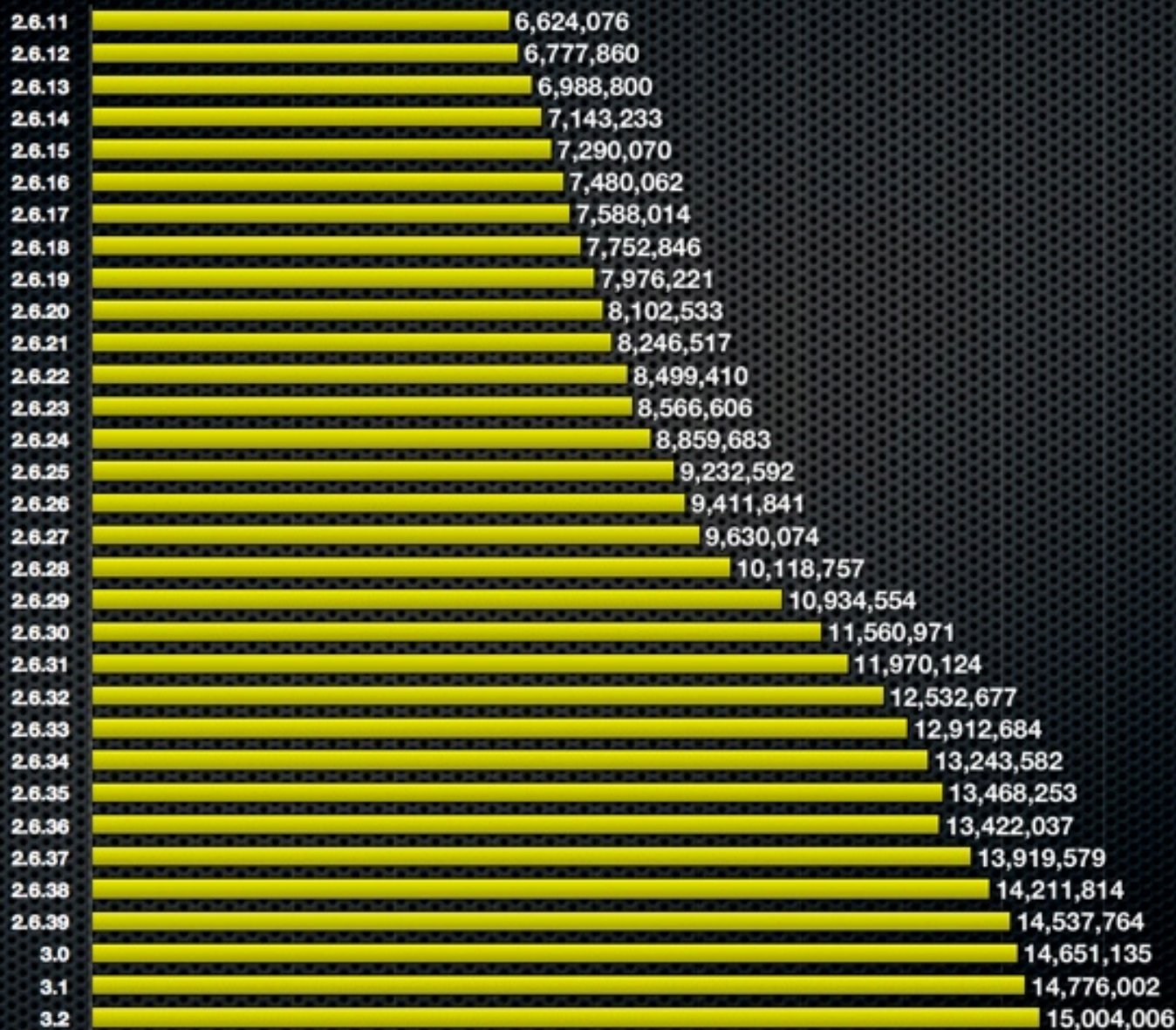


PEKING  
UNIVERSITY



# Number of lines of code in the Linux kernel

Linux kernel version



Data source: Linux Foundation

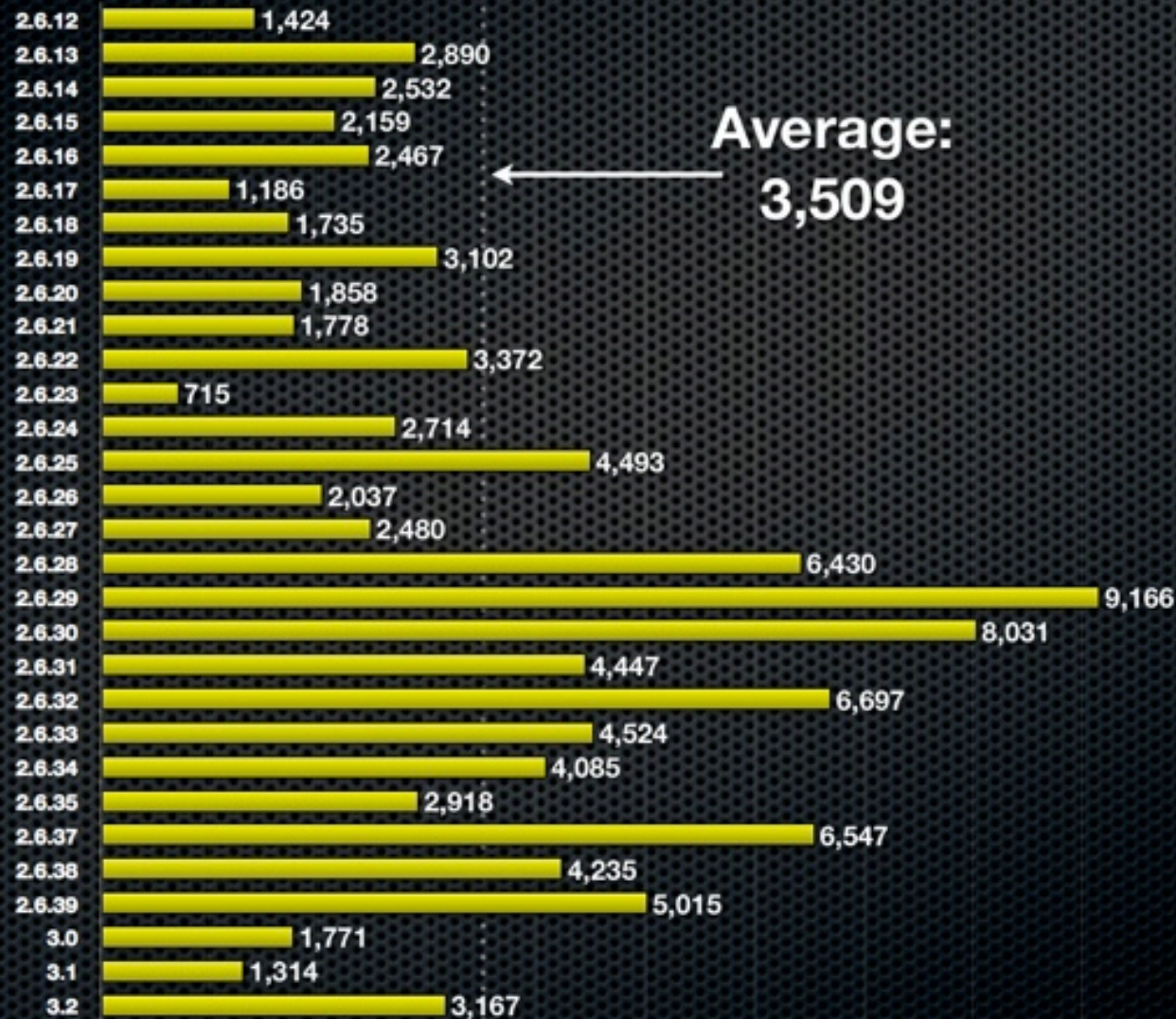
[www.pingdom.com](http://www.pingdom.com)

KING  
UNIVERSITY



# Number of lines of code added to the Linux kernel per each day of development

Linux kernel version



Data sources: Linux Foundation and Pingdom

[www.pingdom.com](http://www.pingdom.com)

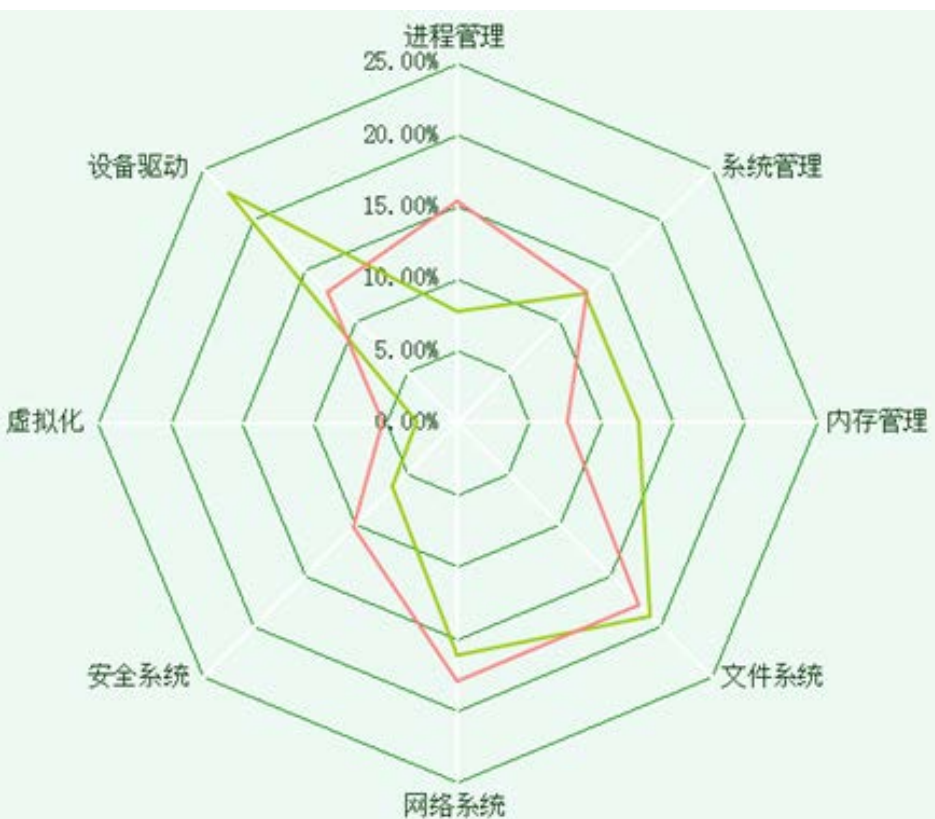
LEARNING  
UNIVERSITY



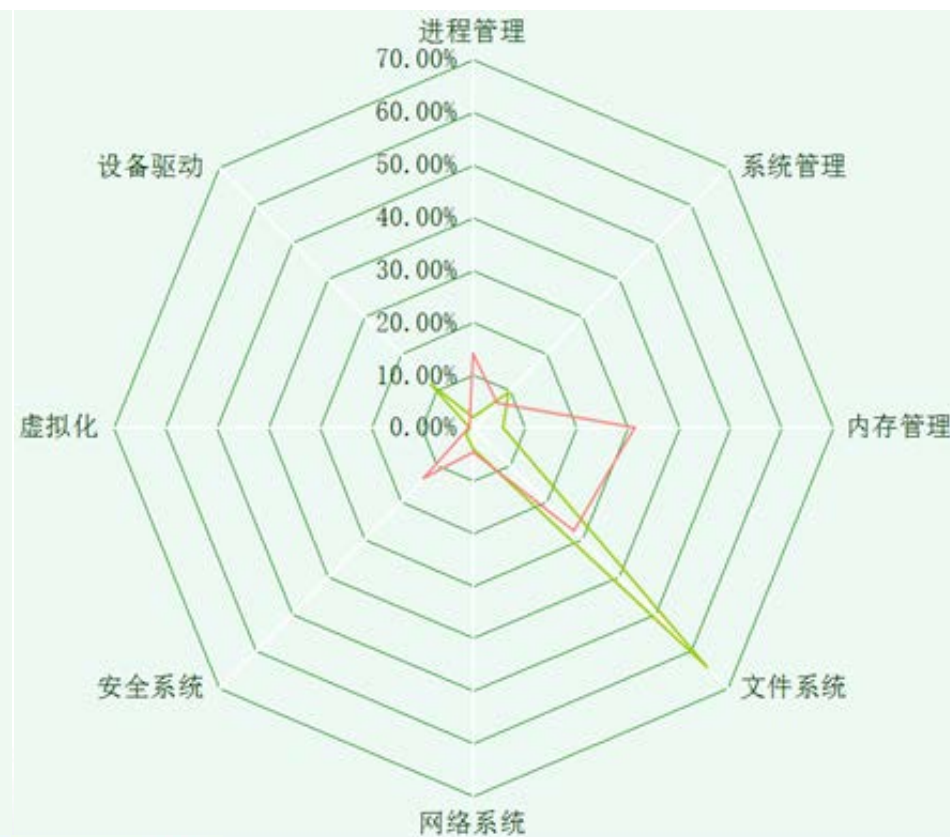


# Linux 3.5到3.19版本与此间longterm版本各子系统特性数据对比

Linux 3.5到3.19和longterm版本  
变更**特性数**占总特性数比例对比图



Linux 3.5到3.19和longterm版本变  
更**代码行数**占总变更代码行数比例对比图



— Linux 3.5-3.19  
— Linux longterm



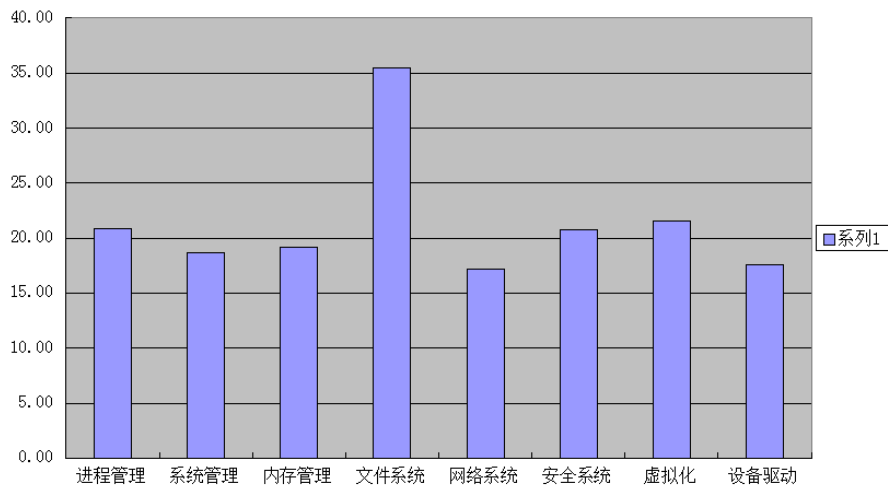
PEKING  
UNIVERSITY

# 重要特性分析

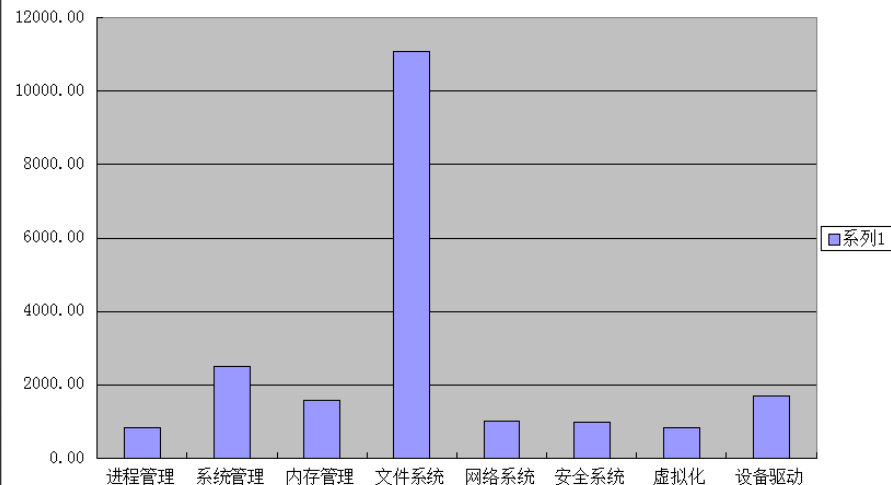
从下图是**每特性**变更代码量和文件数的统计，从该角度来看各子系统特性分析难度：

- 文件系统分析难度最大，因为平均每特性涉及的文件数最多、修改的代码量也最大。
- 进程管理、安全系统和虚拟化变更涉及文件较多，但代码分析工作量相对较小。
- 系统管理、内存管理、网络和设备驱动子系统变更涉及的文件数较少，但代码工作量相对较大。

平均每特性变更文件数



平均每特性新增代码行数





## TFO特性的更新过程中涉及的主要commit如下图所示：

### Diffstat

```
-rw-r--r-- include/linux/tcp.h      10
-rw-r--r-- include/net/tcp.h        9
-rw-r--r-- net/ipv4/Makefile         2
-rw-r--r-- net/ipv4/syncookies.c     2
-rw-r--r-- net/ipv4/sysctl_net_ipv4.c 7
-rw-r--r-- net/ipv4/tcp_fastopen.c  11
-rw-r--r-- net/ipv4/tcp_input.c     26
-rw-r--r-- net/ipv4/tcp_ipv4.c       2
-rw-r--r-- net/ipv4/tcp_minisocks.c  4
-rw-r--r-- net/ipv4/tcp_output.c    25
-rw-r--r-- net/ipv6/syncookies.c     2
-rw-r--r-- net/ipv6/tcp_ipv6.c       2
```

12 files changed, 86 insertions, 16 deletions

### Diffstat

```
-rw-r--r-- Documentation/networking/ip-sysctl.txt 11
-rw-r--r-- include/linux/socket.h                1
-rw-r--r-- include/net/inet_common.h             6
-rw-r--r-- include/net/tcp.h                     3
-rw-r--r-- net/ipv4/af_inet.c                     19
-rw-r--r-- net/ipv4/tcp.c                         61
-rw-r--r-- net/ipv4/tcp_ipv4.c                   3
```

7 files changed, 92 insertions, 12 deletions

### Diffstat

```
-rw-r--r-- Documentation/networking/ip-sysctl.txt 29
-rw-r--r-- include/linux/rmp.h                   4
-rw-r--r-- include/linux/tcp.h                   45
-rw-r--r-- include/net/request_sock.h            36
-rw-r--r-- include/net/tcp.h                     46
-rw-r--r-- net/ipv4/proc.c                        4
-rw-r--r-- net/ipv4/sysctl_net_ipv4.c            45
-rw-r--r-- net/ipv4/tcp_fastopen.c              83
-rw-r--r-- net/ipv4/tcp_input.c                  4
```

9 files changed, 276 insertions, 20 deletions

### Diffstat

```
-rw-r--r-- Documentation/networking/ip-sysctl.txt 4
-rw-r--r-- net/ipv4/tcp_fastopen.c                2
```

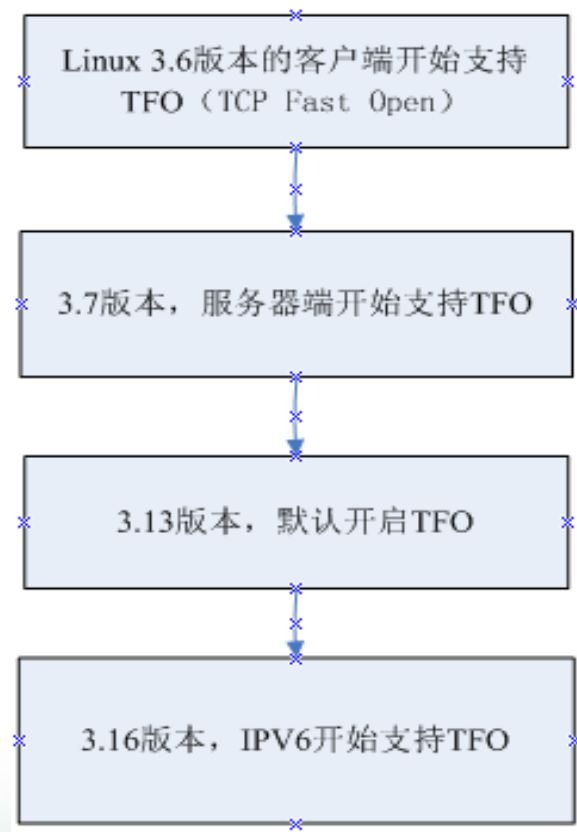
2 files changed, 3 insertions, 3 deletions

### Diffstat

```
-rw-r--r-- net/ipv4/tcp_fastopen.c 57
-rw-r--r-- net/ipv6/tcp_ipv6.c     40
```

2 files changed, 71 insertions, 26 deletions

<http://kernelnewbies.org/>



PEKING  
UNIVERSITY



# 3. Linux Kernel Introduction

## 3.1 Linux发展的五大支柱

## 3.2 内核发展趋势及特点

- \* 内核源文件
- \* Linux内核发展趋势及特点
- \* **Android**内核特点

## 3.3 Linux发行版及软件包管理

## 3.4 内核源代码结构

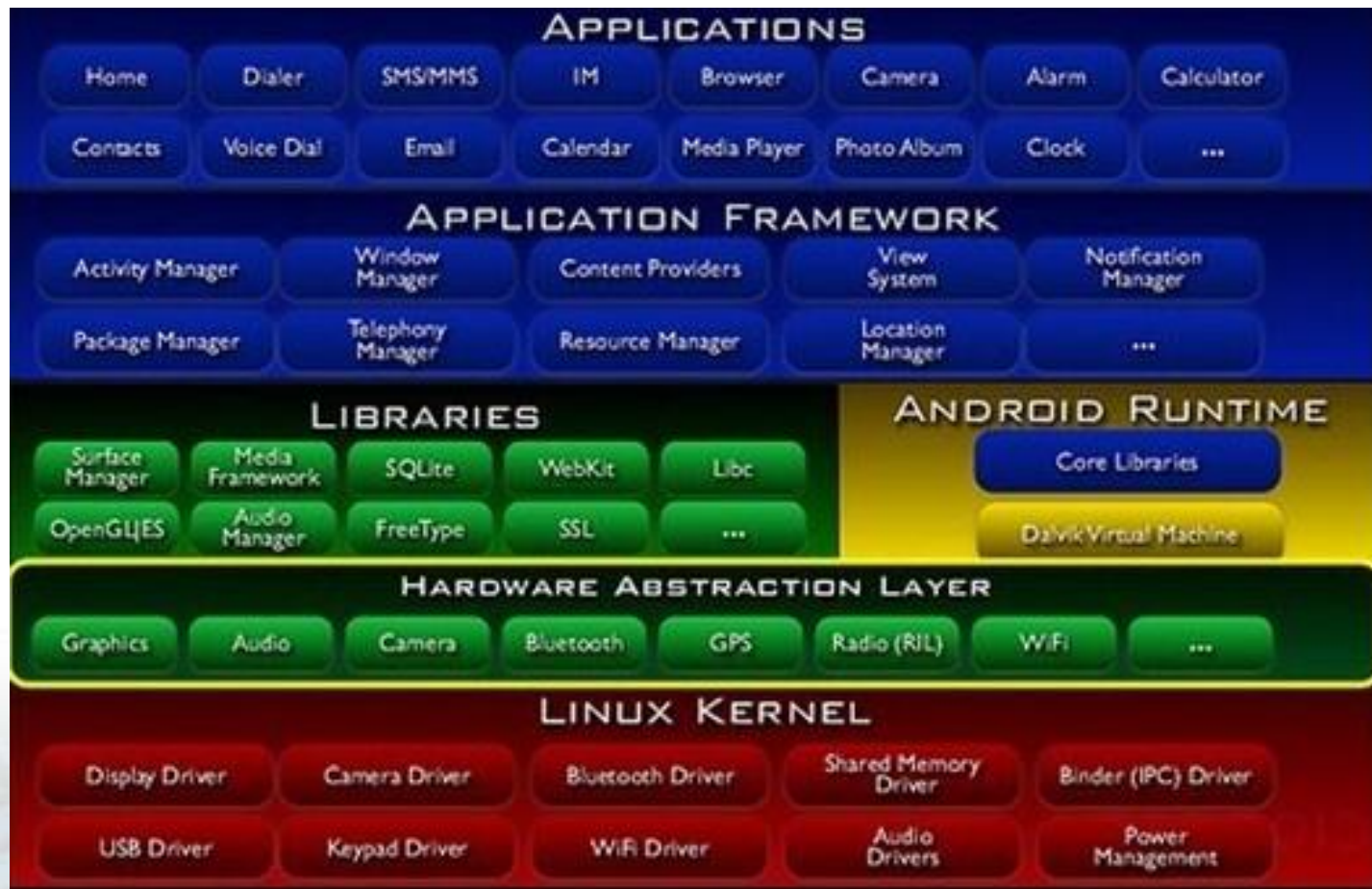
## 3.5 内核编译介绍





# android系统

- Android 系统采用了分层的架构，可分为五层；从高层到低层分别为应用程序层、应用程序框架层、系统运行库层、硬件抽象层和Linux内核层：





# Android 系统与一般的 Linux 发行版的差异

- Android 不使用GNU/Linux 的 **X 窗口** 系统
- Android 不使用 **glibc**，而是使用Google自己开发的一套 **Bionic Libc** 来代替glibc。Bionic Libc采用BSD License,大小只有约200K，并且比glibc更快，其目标是达到**轻量化**以及**高运行速度**
- Android 不包括一整套标准的 Linux 实用**程序**。即不包含 Linux 发行版中/bin 目录下的大部分实用程序





# android内核特点

Android 系统内核源自于 Linux 内核，但做了较多的改动

Android 内核与 Linux 主线内核的区别包括：

- Android 内核使用**专有的驱动**，如 Android Binder, Android 电源管理, USB Gadget 驱动等
- Android 对 Linux 内核中的某些功能进行了**增强**，如低内存管理器（Low Memory Killer）和匿名共享内存（Ashmem）等
- **新增**了一些功能特性，如 yaffs2 FLASH 文件系统等
- Android 使用自己的内核配置，一般参考 **Goldfish** 内核的配置文件，其中提供了 **ANDROID\_PARANOID\_NETWORK**, **ASHMEM** 等“一般需要支持的内核功能”以及 **CONFIG\_YAFFS\_DISABLE\_LAZY\_LOAD** 等“推荐不要支持的内核功能”





# Android内核特点

- 主线内核中的某些特性对 **Android** 手机意义不大
  - 比如在**3.12**主线内核中引入的**AMD**硬件图像性能提升，改进了**cpufreq ondemand**调控器的算法使得性能提升。因为**Android**设备一般不使用**AMD**图像硬件，这样的提升对于**Android**应用意义不大
- **android**内核更新与**Linux**的更新相比非常慢
  - 一直都是**google**的开发者负责内核的升级
  - 与**Linux**庞大的社区相比，**google**开发者不可能面面俱到的将**Linux**每一个对**android**有意义的**linux**更新都应用到**android**中
  - **adroid**内核开发者以及手机厂商经常将**Linux**内核重要的新特性打到**Android**内核上
    - 如在**3.14**内核中加入的**zram**特性，在**Android 5.0** 的**3.10**内核中已经加入，源码中**/drivers/staging/zram**已经存在





- 商用手机内核

- 商用手机采用的内核又在 **Android** 内核的基础上进行了修改
- 如华为手机加入了华为自己的驱动控制程序，**/drivers/hisi/**和**/drivers/hisi\_pilot**目录，其中是对华为海思处理器提供支持的相关代码。**/drivers/Huawei\_platform**和**/drivers/Huawei\_platform\_legacy**目录包括对华为平台的支持代码







## 3. Linux Kernel Introduction

3.1 Linux发展的五大支柱

3.2 内核发展趋势及特点

**3.3 Linux发行版及软件包管理**

3.4 内核源代码结构

3.5 内核编译介绍



PEKING  
UNIVERSITY



# Linux Distributions

- **CentOS**
- **Debian**
- **Fedora Core**
- Gentoo Linux
- Knoppix Linux
- **Linux deepin**
- **Linux Mint**
- Mageia
- Mandriva Linux
- **Red Hat Linux**
- **Red flag**
- Slackware Linux
- Stanix Live CD
- **SUSE Linux**
- Turbo Linux
- **Ubuntu**
- Ubuntu Kylin

<http://distrowatch.com/>

<http://www.linuxfromscratch.org/>



PEKING  
UNIVERSITY



Linux 基金会官网 **Linux.com** :

## Top 5 Linux Distributions for New Users

[Elementary OS](#)

[Deepin](#)

[Ubuntu](#)

[Linux Mint](#)

[Ubuntu Budgie](#)

## Top 5 Linux Distributions for Development in 2019

[Ubuntu](#)

[openSUSE](#)

[Fedora](#)

[Pop! OS](#)

[Manjaro](#)

## Top 5 Linux Server Distributions

RHEL

SUSE

Ubuntu Server

Debian

CentOS



PEKING  
UNIVERSITY



# 软件包管理器

- **DPKG: Debian Packager**

- 适用于发行版**Debian Mint Ubuntu Deepin.....**
- 软件包格式**deb**，文件名后缀为**.deb**
- 传统方式：使用命令**dpkg**进行管理，可离线安装，需自行解决依赖
- 为了解决依赖、自动安全配置更新升级：**APT (Advanced Packaging Tools)**
- 从软件源安装：例如**Ubuntu**官方仓库，可添加**PPA**源
  - **PPA (Personal Package Archive)**：较新

- **RPM: Redhat Package Manager**

- 适用于发行版**RHEL CentOS SUSE Fedora.....**
- 文件后缀名为**.rpm**
- **Yum (Yellow dog Updater, Modified)**

## Yum源

- 自带的：除了版权争议的软件、版本较老
- **EPEL、RPMForge**



PEKING  
UNIVERSITY



## 3. Linux Kernel Introduction

3.1 Linux发展的五大支柱

3.2 内核发展趋势及特点

3.3 Linux发行版及软件包管理

**3.4 内核源代码结构**

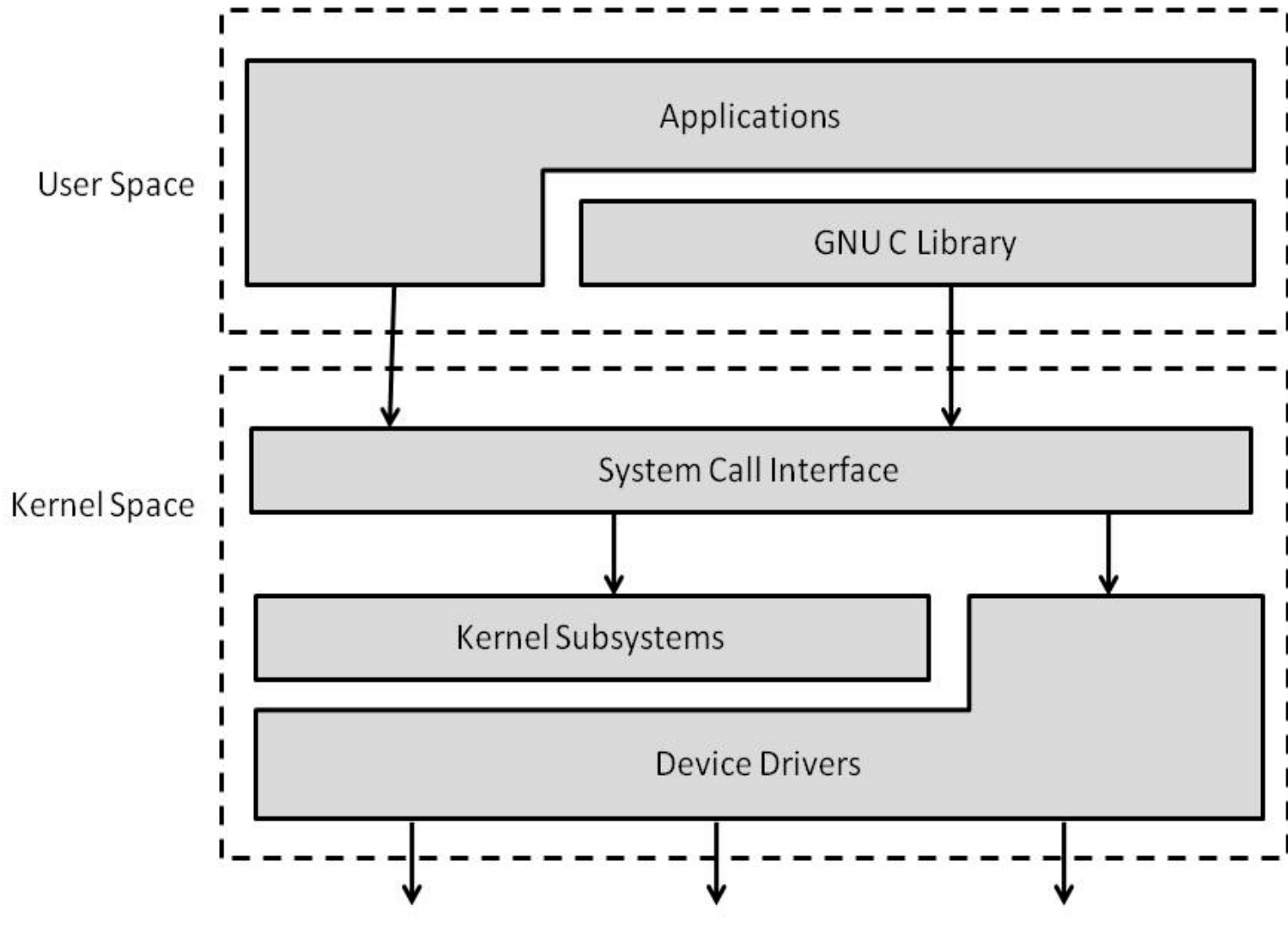
3.5 内核编译介绍



PEKING  
UNIVERSITY



# Overview of Operating Systems and Kernels





# 内核源代码结构

- 初始化系统部分
- 内存管理部分
- 进程管理&进程通信与同步&进程调度部分
- 网络部分（很多代码）
- 文件系统部分（很多选择）
- 安全相关
- 设备驱动程序
- 平台相关代码 (**x86, ARM, Alpha, Sparc, ...**)





# Cont.

- ... ./include Header files (.h)
- ... ./lib Common functions
- ... ./init Kernel initialization code
- ... ./kernel Kernel core
- ... ./mm Memory management
- ... ./ipc Inter-process Communication
- ... ./net Networking code
- ... ./scripts Helping tools for building kernel
- ... ./**Documentation**





# Cont.

- ... ./fs Filesystems
  - ... ./fs/ext4 Most popular Linux filesystem
  - ... ./fs/msdos MSDOS file system (C> drive)
  - ... ./fs/vfat MS Windows (VFAT)
  - ... ./fs/proc virtual file system (process info)
- ... ./drivers Device drivers
  - ... ./drivers/block hard drives
  - ... ./drivers/scsi SCSI device
  - ... ./drivers/char character-stream device
  - ... ./drivers/net Network cards





# Cont.

- .... /arch      Platform-dependent code
  - .... /arch/i386 Intel 386 (IBM PC architecture)
  - ... ./arch/alpha Compaq's Alpha architecture
  - ... ./arch/sparc Sun's SPARC architecture
  - ... ./arch/um UML's virtual machine architecture
- .... /security Linux Security Module
- .... /crypto      Crypto API







# Supported architectures

- **x86**
  - Intel、AMD、Cyrix、Winchip、Rise
  - Itanium、AMD
- **ppc**
  - AIM(Apple、IBM、Motorola)
- **sparc**
  - Sun
- **mips**
  - MIPS
- **alpha**
  - Compaq
- ...



## electronics

**user peripherals**

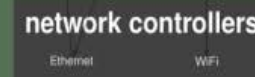
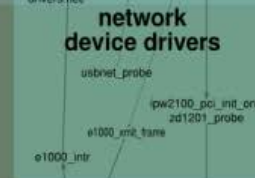
## I/O mem. I/O

## CPU

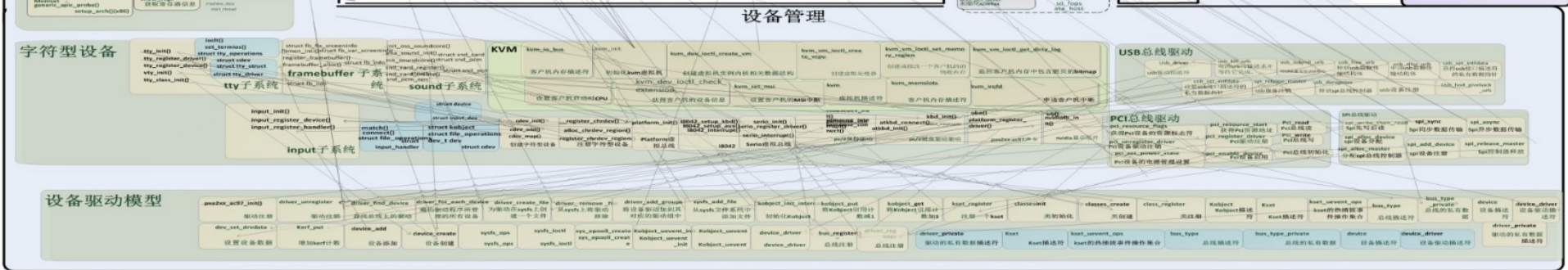
memory

**disk controllers**

**network controllers**









## 3. Linux Kernel Introduction

3.1 Linux发展的五大支柱

3.2 内核发展趋势及特点

3.3 Linux发行版及软件包管理

3.4 内核源代码结构

3.5 内核编译实践



PEKING  
UNIVERSITY





# Linux发行版及内核版本推荐

- **Linux发行版**

- 考虑到发行版的易用性以及Linux稳定支持问题，Linux发行版可以使用Ubuntu、CentOS等

- **Linux内核版本**

- 内核版本推荐用Long term，作业要求统一，下载地址<https://www.kernel.org/>

- **Linux系统安装**

- 虚拟机安装
- 双系统安装 可以用u盘制作一个linux启动盘
- 开始时推荐用虚拟机安装，内核编译时相对比较简单，不会影响原系统的使用







# 安装Linux系统

- 请在自己机器安装
- 虚拟机
- 新版内核
- 完全安装



PEKING  
UNIVERSITY



# 编译步骤

- A. 下载内核源代码
- B. 解压Linux源代码目录
- C. 配置内核
- D. 编译内核和模块
- E. 安装
- F. 配置启动文件
- G. 重启并进入新内核

实验环境：虚拟机软件不限



PEKING  
UNIVERSITY

# A. 下载内核源代码

本学期发行版推荐**ubuntu 18.04.3 LTS**，其内核是**5.0.0**

第一次课编译内核可以选择**5.3.0**，

下载地址：

1.<https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.3.tar.xz>

2.(清华镜像)

<https://mirrors.tuna.tsinghua.edu.cn/kernel/v5.x/linux-5.3.tar.xz>

3.(北交大镜像)

<https://mirror.bjtu.edu.cn/kernel/linux/kernel/v5.x/linux-5.3.tar.xz>

具体内核选择哪个版本可以在以下网站了解（不用选择太新的，可能不稳定，对以后的实验也有影响）：

1.[www.kernel.org](http://www.kernel.org)

2.<https://cdn.kernel.org/pub/linux/kernel/>

3.<https://mirrors.tuna.tsinghua.edu.cn/kernel/>

4.<https://mirror.bjtu.edu.cn/kernel/linux/kernel/>



PEKING  
UNIVERSITY



## A. 下载内核源代码

- 将下载的压缩包移到Linux工作目录下；或者直接使用wget命令直接下载到你的工作目录下

```
# cd ~
```

```
# wget
```

```
https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.3.0.tar.xz
```



PEKING  
UNIVERSITY




## B. 解压Linux源代码目录

- 解压源码压缩包 `# tar -xvf linux-5.3.0.tar.xz`
- 解压出来的是一个Linux目录，里面就是5.3.0的内核源代码。








## C. 配置内核

- 查看内核源代码目录下的**README**文件。
- 在编译内核前，需要对内核进行相应的配置。
- 配置是精确控制新内核功能的一个机会。
- 配置过程控制哪些需编译到内核的二进制映象中(在启动时被载入)，哪些是需要时才装入的内核模块 (**module**)。





## C. 配置内核


- 命令 **make mrproper**。它将清除目录下所有配置文件和先前生成核心时产生的中间文件：

```
#cd linux-5.3.0
```

```
#make mrproper
```

- make clean: remove most generated files but **keep the config**
- make mrproper: remove all generated files + config+ various backup files





# C. 配置内核

- 以下几种方式可以进行配置：
  - **make config**是基于文本的传统配置界面
  - **make menuconfig**是基于文本的选单式配置界面，是最为灵活的内核配置工具
    - 如果报错，并提示缺少ncurses，执行  
`#sudo apt-get install libncurses5-dev`
    - 其他类似报错，均下载相关库或者命令即可解决
  - 图形窗口模式：基于图形窗口模式的配置界面
    - **make xconfig** 需要**xWindow**图形环境的支持（QT）
    - **make gconfig** 需要**GTK+**支持
  - **make oldconfig**用于在原来内核配置的基础上作修改





## C. 配置内核 -cont.

### – make x86\_64\_defconfig

- 默认配置的编译模块较多，花费时间太长。对于x86的64位cpu，可大量减少编译的模块（32位可将命令中的64改为32）

### – make localmodconfig

- 生成仅包含正在使用的内核模块的.config文件，从而缩短内核的编译时间





```
amos@ubuntu:~$ cd linux-5.3
amos@ubuntu:~/linux-5.3$ ls -a
.          CREDITS          include      .mailmap    security
..         crypto         init         MAINTAINERS sound
arch       Documentation  ipc         Makefile    tools
block      drivers        Kbuild      mm          usr
certs      fs             Kconfig     net         virt
.clang-format .get_maintainer.ignore kernel       README
.cocciconfig .gitattributes lib          samples
COPYING     .gitignore    LICENSES    scripts
amos@ubuntu:~/linux-5.3$
```



PEKING  
UNIVERSITY



## .config - Linux/x86 5.3.0 Kernel Configuration

### Linux/x86 5.3.0 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [\*] built-in [ ]

\*\*\* Compiler: gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0 \*\*\*

General setup --->

[\*] 64-bit kernel

Processor type and features --->

Power management and ACPI options --->

Bus options (PCI etc.) --->

Binary Emulations --->

Firmware Drivers --->

[\*] Virtualization --->

General architecture-dependent options --->

⌄(+)

<Select>

< Exit >


< Help >

< Save >

< Load >



PEKING  
UNIVERSITY



## C. 配置内核

- 对每一个配置选项，用户有三种选择，它们分别代表的含义如下：

**Y** - 将该功能编译进内核；

**N** - 不将该功能编译进内核；

**M** - 将该功能编译成可以在需要时动态插入到内核中的模块。





- 配置完毕之后，选择**Exit**退出

```
Do you wish to save your new configuration?  
(Press <ESC><ESC> to continue kernel configuration.)
```

```
< Yes >      < No >
```





## D. 编译内核和模块

- **#make**

生成vmlinux内核文件

- **#make zImage**

编译产生压缩形式的内核文件

- **#make bzImage**

需要内核支持较多的外设和功能时，内核可能变得很大，此时可以用此命令编译本内核产生压缩率更高的内核文件

- 编译内核需要较长的时间(通常要几十分钟)，具体与机器的硬件条件及内核的配置等因素有关





## D. 编译内核和模块 --cont.

- 如果是双核**PC**，可以使用**make -j4**命令来编译，提高编译速度。
- 编译后如果某些硬件没有驱动起来，可以通过配置内核或寻找相应驱动来解决。







# 注释

make(1)程序能把编译过程拆分成多个作业。其中的每个作业独立并发地运行，这能极大地提高多处理器系统上的编译过程，并有利于改善处理器的利用率，因为编译大型源代码树还包括I/O等待所花费的时间（也就是处理器空下来等待I/O请求完成所花费的时间）。

默认情况下，make(1)只衍生一个作业。Makefiles时常把文件之间的依赖弄乱。对于不正确的依赖，多个作业可能会互相踩踏，导致编译过程出错。当然，内核的Makefiles没有这样的编码错误。为了以多个作业编译内核，使用以下命令：

```
$ make -jn
```

这里，n是要衍生的作业数，在实际中，每个处理器上一般衍生一个或者两个作业。例如，在一个双处理器上，可以输入如下命令：

```
$ make -j4
```

利用出色的distcc(1)或者ccache(1)工具，也可以动态地改善内核的编译时间。





## D. 编译内核和模块

- 如果选择了可加载模块，编译完内核后，要对选择的模块进行编译：  
**#make modules**
- 用下面的命令将模块安装到标准的模块目录中：  
**# sudo make modules\_install**
  - 模块在系统中的标准目录位于 **/lib/modules/x.y.z**，后面的 **x.y.z** 是版本号



# E. 安装内核

- # sudo make install

```
amos@ubuntu:~/linux-5.3$ sudo make install
sh ./arch/x86/boot/install.sh 5.3.0 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.3.0
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.3.0
update-initramfs: Generating /boot/initrd.img-5.3.0
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.3.0
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.3.0 /
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.3.0-28-generic
Found initrd image: /boot/initrd.img-5.3.0-28-generic
Found linux image: /boot/vmlinuz-5.3.0
Found initrd image: /boot/initrd.img-5.3.0
Found linux image: /boot/vmlinuz-4.19.23
Found initrd image: /boot/initrd.img-4.19.23
Found linux image: /boot/vmlinuz-4.18.0-21-generic
Found initrd image: /boot/initrd.img-4.18.0-21-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

```
amos@ubuntu:~/linux-5.3$ ls /boot
config-4.18.0-21-generic  grub  initrd.img-5.3.0-28-generic  System.map-4.18.0-21-generic
vmlinuz-4.18.0-21-generic  initrd.img-4.18.0-21-generic  memtest86+.bin  System.map-4.19.23
config-4.19.23  initrd.img-4.19.23  memtest86+.elf  System.map-5.3.0
vmlinuz-4.19.23  initrd.img-5.3.0  memtest86+_multiboot.bin  System.map-5.3.0-28-generic
config-5.3.0-28-generic  vmlinuz-5.3.0-28-generic
```

## F. 配置启动文件

- Ubuntu 系统为 **GRUB**(GRand Unified Bootloader)引导，需要更新**grub**
- 用**update-grub**命令可以直接更新**grub**启动菜单（**grub.cfg**），生成相应的启动项，此时新版本的内核在第一个启动位置；

```
amos@ubuntu:~/linux-5.3$ sudo update-grub
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.3.0-28-generic
Found initrd image: /boot/initrd.img-5.3.0-28-generic
Found linux image: /boot/vmlinuz-5.3.0
Found initrd image: /boot/initrd.img-5.3.0
Found linux image: /boot/vmlinuz-4.19.23
Found initrd image: /boot/initrd.img-4.19.23
Found linux image: /boot/vmlinuz-4.18.0-21-generic
Found initrd image: /boot/initrd.img-4.18.0-21-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```



- 在开机时进入**Grub** 界面后，按 **shift** 键可以进行选择内核
- 默认情况下**grub**界面并不显示，修改**grub**界面显示时长需要修改相应的配置文件 **/etc/default/grub**，在**ubuntu18.04**中只需修改**GRUB\_TIMEOUT** 参数

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

- 修改完毕该文件之后需重新执行**update-grub**命令使设置生效







## G. 重启并进入新内核

- 重启： 执行**reboot** 命令进行重启并选择刚刚安装的内核
  - 上一步执行**update-grub** 命令后默认以5.3.0内核运行
- 进入系统之后执行**uname -a**命令查看系统版本， 如果为5.3.0则内核安装启动成功

```
amos@ubuntu:~$ uname -a
Linux ubuntu 5.3.0 #1 SMP Sat Feb 15 19:31:58 +08 2020
x86_64 x86_64 x86_64 GNU/Linux
```





# 注意事项

- 在ubuntu18.04中，执行**make modules\_install**和安装内核**make install**时需要提供**root**权限，需使用**sudo**命令执行安装
- 如果是多核**PC**，在编译内核过程中，可以使用  
**#make -jn**  
命令来编译，提高编译速度，其中**n**是要衍生的作业数，  
例如双核可以配置为**4**。
- 重启后进入新内核时如果出现黑屏，登录无响应等情况，可能与虚拟机设置有关，可以通过关闭**3D**加速，启用**PAE/NX**，启用**I/O APIC**等设置来解决
- 重启后，如果某些硬件没有驱动起来，可以通过配置内核或寻找相应驱动来解决





- 4. Driver Introduction

mainly from

《Linux Device Drivers》 3rd Edition



PEKING  
UNIVERSITY



# 驱动程序介绍

- 驱动程序是了解内核对硬件控制的最直接的方式
- 内核提供“标准”的内核接口(一些操作设备的系统调用)，内核把这些“标准”的接口，映射到设备实际的操作之上
- 驱动程序可以使用“模块”加载的方式动态加入到内核运行空间





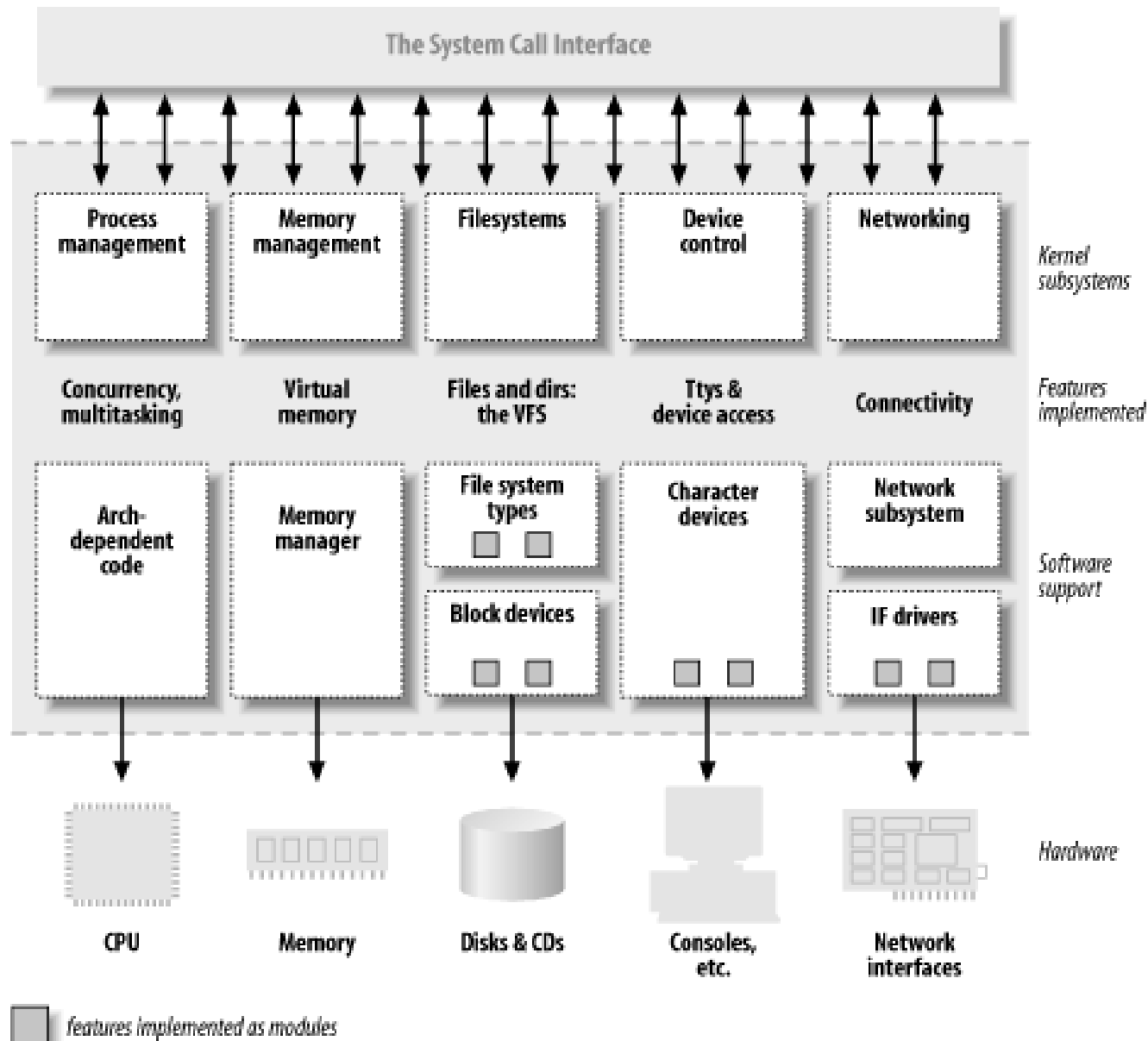
# 驱动程序的角色

- 只是提供机制，不是策略
  - 机制: "what capabilities are to be provided" (the mechanism)
  - 策略: "how those capabilities can be used" (the policy).
  - The driver should deal with **making the hardware available**, leaving all the issues about how to use the hardware to the applications.





# Figure 1-1. A split view of the kernel





# 设备分类(1)

- 字符设备（Character devices）
  - 这类设备可以像一个文件一样被打开、读写、关闭
  - 例如：触摸屏、鼠标等
  - 和普通文件的区别：普通文件可以使用lseek等操作“来回”读数据，而“大部分”字符设备只是一个内核和用户空间的数据通道，这样的设备一般情况下只能**顺序读取数据**。当然也有某些字符设备看起来是一个数据区域，在这个区域内， mmap 和lseek可以来回移动读数据，这样的设备一般是对“数据帧”或成块的内存进行处理，比如视频数据
  - 在/dev目录下与之对应的是类型为c的设备文件





# 设备分类(2)

- 块设备(Block devices)

- 块设备是能“容纳”文件系统的设备，比如磁盘设备
- 通常块设备只能处理以块为单位的数据操作，通常块的大小是**512**字节或者其整数倍。Linux块设备有点特殊之处在于可以对其以字节单位读取，这样的话，块设备&字符设备在驱动程序的角度最大的区别是其在内核中的**内部数据管理方式和接口的不同**
- 块设备在操作系统中在/dev目录下与之对应的是类型为b的文件





# 设备分类(3)

- 网络接口(network interface)
  - 网络接口可以是硬件接口，也可以是纯软件接口
  - 通过内核网络部分相关代码驱动网络接口发送和接收数据包
  - 网络接口不在/dev目录下出现，而是以某个系统中唯一的名字出现，比如eth0





# 作业1（实验报告）

- 编译新版内核，并用该内核启动系统
  - 拷屏（名字带有特征）+解释
  - 遇到的问题及解决
- 或者 **Linux From Scratch**
  - Linux From Scratch (LFS) is a project that provides you with step-by-step instructions for building your own custom Linux system, entirely from source code.
  - <http://www.linuxfromscratch.org/>

