

# git入门与实战

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

如果下面问题你都会，不用学习本课程了

- 问题1：如何在本地建一个git repo？
- 问题2：如何用git分支来管理我们日常的开发？
- 问题3：假设现在客户要求要在Linux 4.0这个内核版本上开发一个新的驱动，新建一个git repo，但是必须包含Linux 4.0里面所有的社区的git log。
- 问题4：过了几个月，客户说，我们现在要把已经开发的东西，全部rebase到Linux 4.15上，而且要包含Linux 4.15上所有的git log信息

## • 问题5:

- 如何给Linux内核社区制作和发送多个patch的补丁集?
- 当你发送v1版本的补丁集之后, 社区给了feedback, 然后你要制作v2版本的补丁集, 那怎么玩?

## 问题6:

- git rebase和git merge有啥区别?
- 做分支合并的时候遇到冲突, 你知道怎么处理吗?
- 同事给你一个patch, 你git am发现遇到冲突了, 怎么处理呢?

《奔跑吧Linux内核》高清视频已经上线  
请关注<https://c9pp45683645.taobao.com/>

上述6个问题，您都会了，没必要浪费一个汉堡。

还若有所思，快来观看吧



# Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git入门（1）——基本用法

笨叔叔

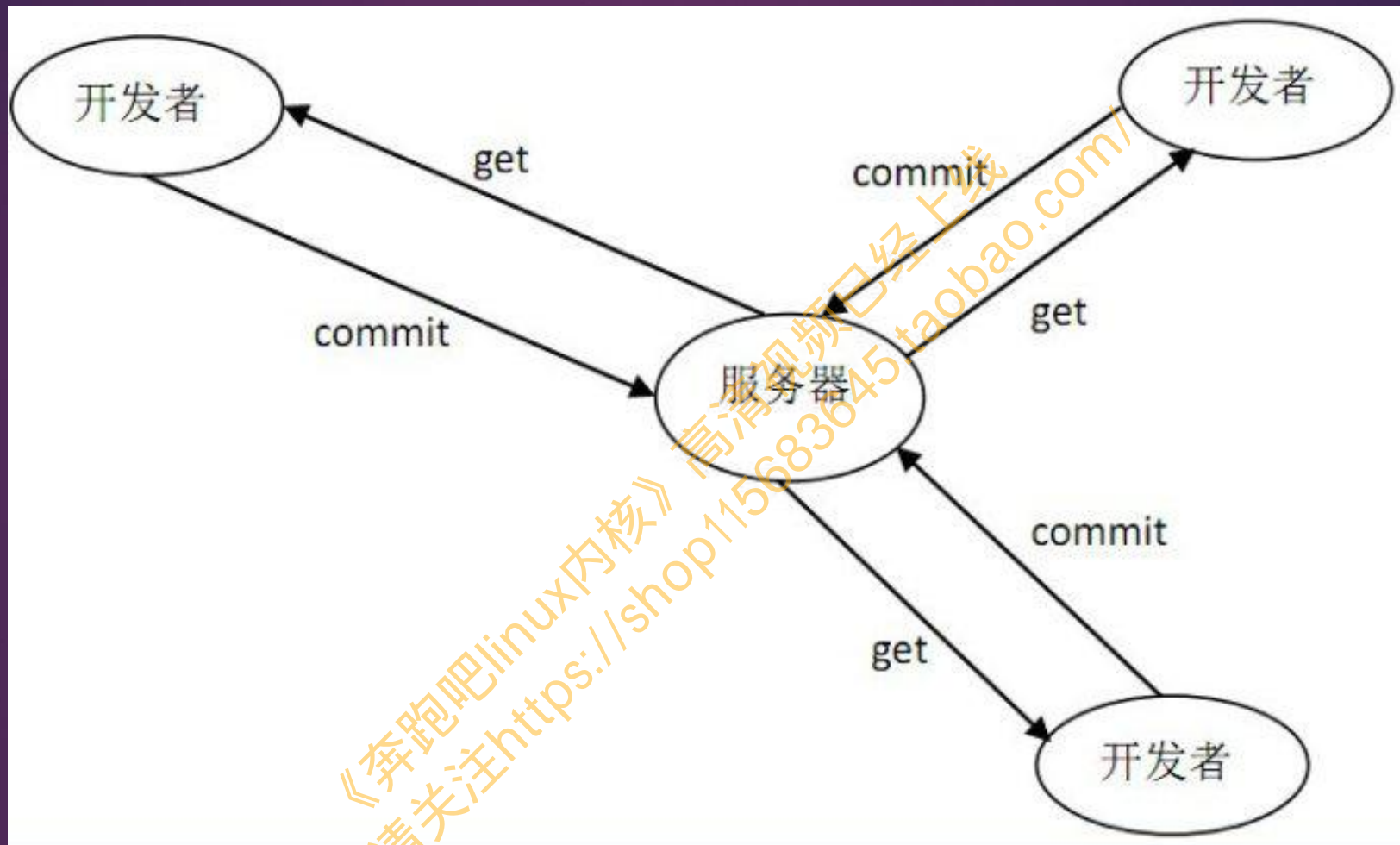
《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git 历史

- 早期Linux内核采用BitKeeper来管理
- 社区里有人试图破解BitKeeper被发现
- Linus花了2周设计了git的原型

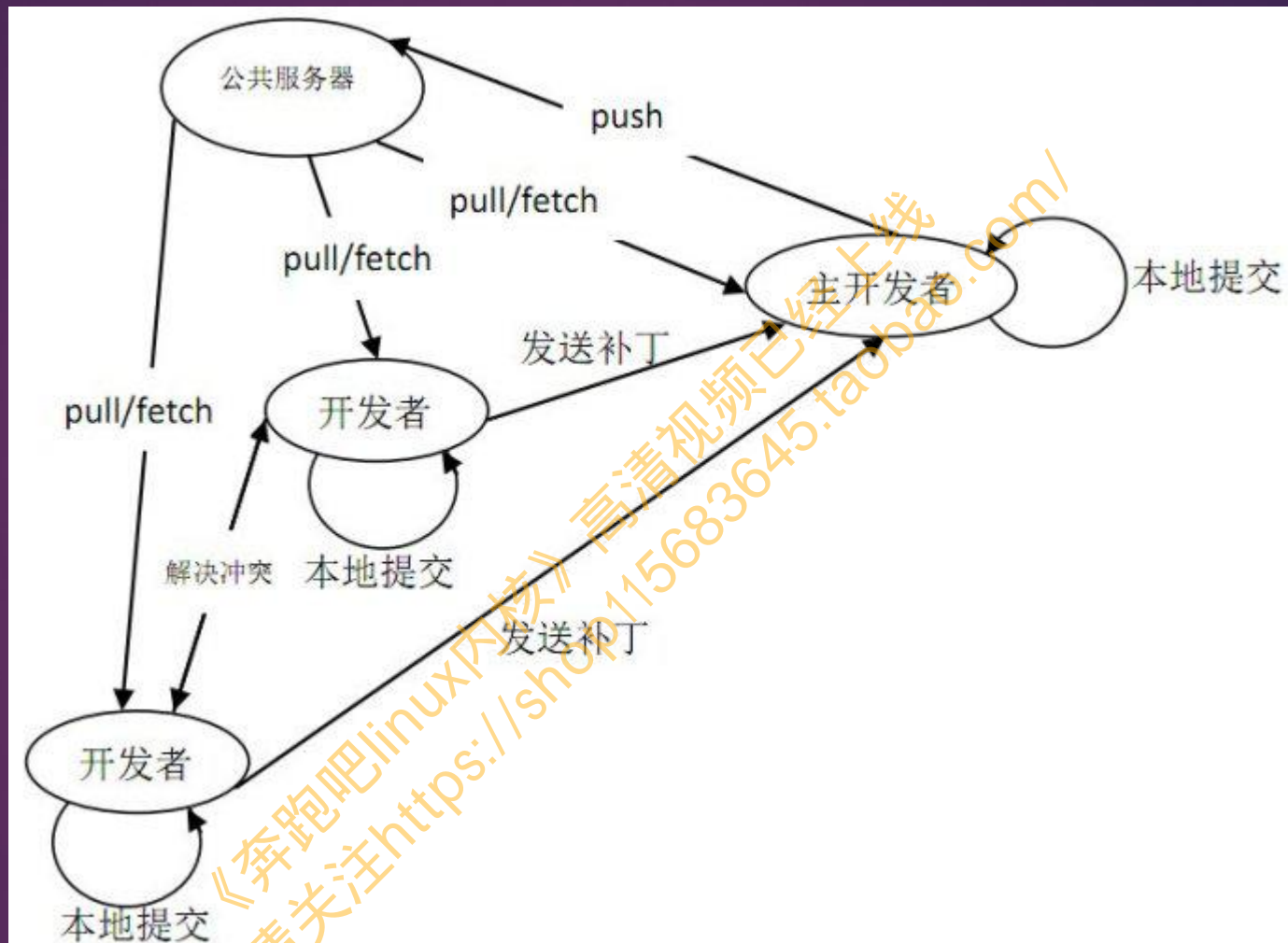


git有啥绝活？



集中式代码管理





git工作模式

# git的绝活？

- 分布式开发。
- 本地有完整的版本库。
- 支持和鼓励基于分支来开发。
- 性能优异，完整性和可靠性。

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop11683645.taobao.com/>

# git使用之前的配置

- 安装git。

```
sudo apt install git
```

- repository仓库。

```
git config --global user.name "Ben Shushu"  
git config --global user.email runninglinuxkernel@126.com
```

- git配置，配置作者和邮箱

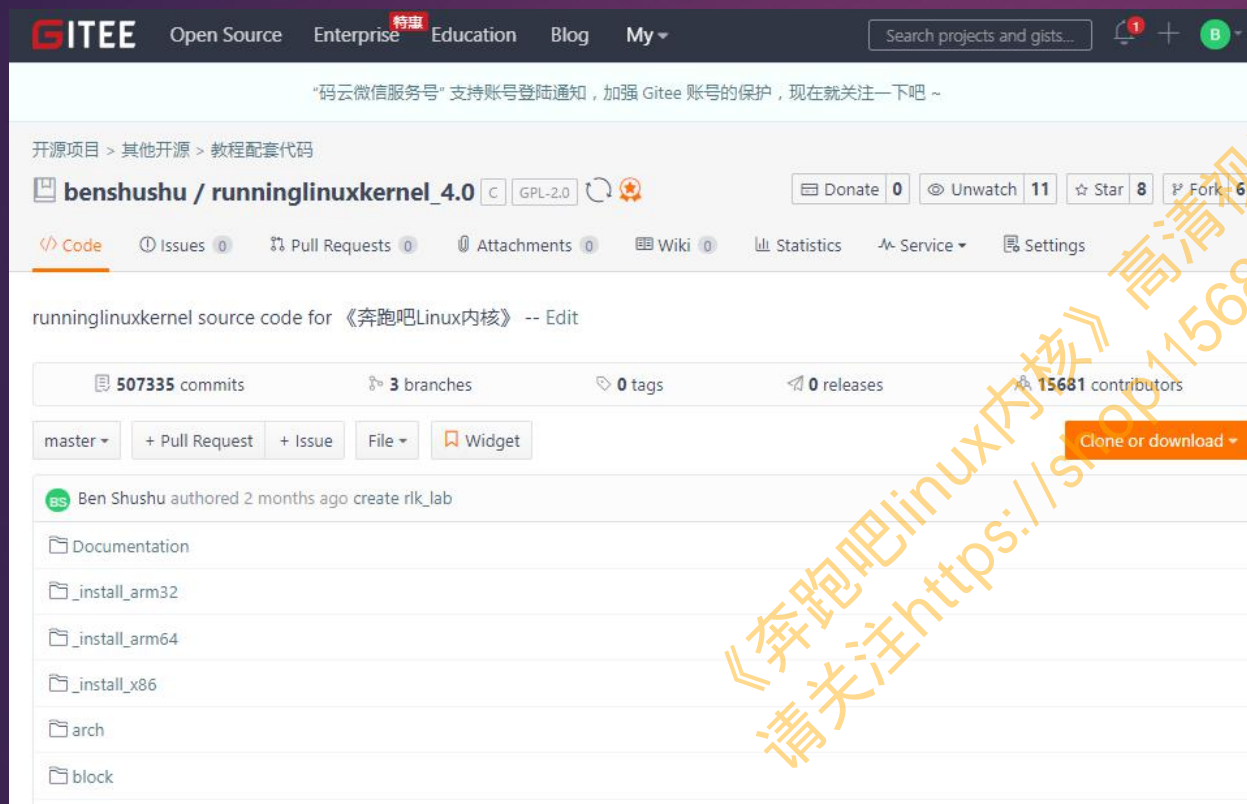
- 显示配置信息： `git config --list`

```
ben@ubuntu:~/work/runninglinuxkernel_4.0$ git config --list  
user.email=runninglinuxkernel@126.com  
user.name=Ben Shushu  
sendemail.smtpencryption=tls  
sendemail.smtpserver=smtp.126.com  
sendemail.smtpuser=figo1802@126.com  
sendemail.smtpserverport=25  
core.repositoryformatversion=0  
core.filemode=true  
core.bare=false  
core.logallrefupdates=true
```

# git clone仓库

- git clone笨叔的奔跑吧Linux内核的git 仓库

➤ # git clone https://gitee.com/benshushu/runninglinuxkernel\_4.0.git



➤ 《奔跑吧Linux内核》配套实验代码

➤ 支持“O0”编译的内核

✓ 光标不会乱跳

✓ 不会出现<optimized out>变量不能显示问题

## git log命令

- git log
- git log --oneline
- git log --author=Linus --oneline
- git log --patch-with-stat

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 提交一个commit的步骤

1. 修改源代码
2. `git diff`查看修改的文件的差异
3. `git status`命令查看哪些文件被修改了
4. `git add`命令添加修改文件
5. `git commit`命令，来把刚才的修改 生成一个`git commit`提交到本地的仓库里

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# 实验1：本地建一个git repo

## 1. git 服务器端

- git --bare init命令创建了一个空的远程仓库

## 2. client端

- git init 建本地的仓库
- git add 添加文件
- git commit来生成一个新的提交
- git remote add命令来添加刚才远程仓库的地址
- git push origin master 推送到远程仓库

《奔跑吧！Linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>



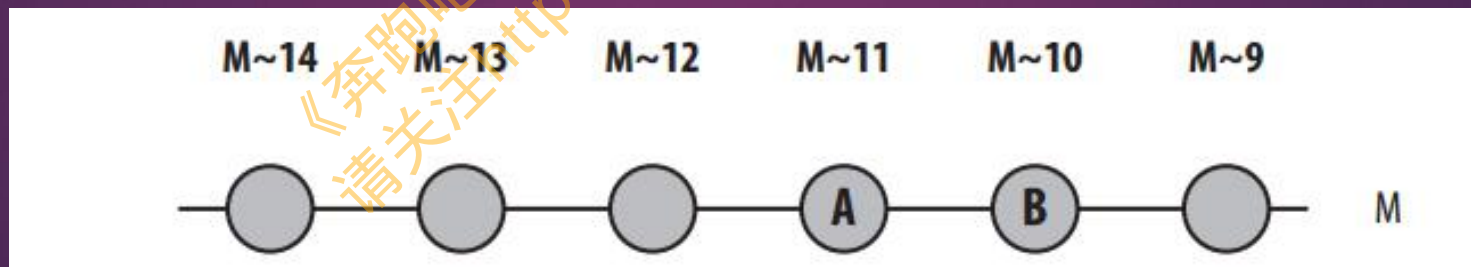
# git入门二

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 查看git log信息

- 绝对提交名：40位十六进制的SHA1的哈希ID值，也可以使用前缀来表示。
- 符号引用：HEAD => 指向当前分支最近的提交。
- 相对提交名：master表示master分支的头，那master^ 表示master分支的倒数第二个commit
- 范围：符号（..）可以表示一个范围  
master~12..master~10：表示master分支上倒数第11和第10个提交。



## 表示范围的几个例子

- `git log master`: 从master分支HEAD开始显示所有的commit
- `git log --pretty=short master~12..master~10`
  - 这里显示master~12 到master~10之间的所有的提交。`--pretty=short`只是显示 commit 的标题，这里可以选择online, short和full。
- `git log --pretty=short master~12..master~10 --stat`
  - `--stat`来显示commit里修改了那些文件。
- `git log -1 -p xxxx`
  - 这里可以使用-p选项来指定某一个commit的 ID, -1表示输出commit的数量为一个。

## git diff的几个小技巧

- git diff命令可以显示在缓存中或者未在缓存中改动，常用的选项有：
  - 显示尚未缓存的改动： `git diff`
  - 查看已经缓存的改动： `git diff --cached`
  - 查看所有的改动： `git diff HEAD`
  - 显示摘要： `git diff --stat`

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## git status命令

- git把所有文件分成3类:

- ✓已跟踪的（Tracked）：已追踪的文件是指已经在版本库里的文件，或者已经缓存到索引的文件，反正就是被git识别和跟踪的文件。通常通过git add命令添加的文件，都是在已跟踪的文件。
- ✓被忽略的（Ignored）：这里说的被忽略的文件，通常是指临时文件，编译输出的文件等等这些。通常是没啥用的文件
- ✓未跟踪的（Untracked）：通常是除了上述两类，剩下的文件就是没有被跟踪的文件，通常新建的文件属于这类。

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop145683645.taobao.com/>

## git add、rm、mv命令

- **git add 命令**：把一个文件添加到git系统中，或者说把文件放到git缓存里或者索引里。
- **git rm命令**：其实是和git add 相反的命令，把版本库的文件和目录删除。
- **git mv命令**：移动或者重命名文件。

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## git commit命令

- 修改提交 `-amend`
- `$ git commit --amend`
- 此命令将使用当前的暂存区域快照提交。如果刚才提交完没有作任何改动，直接运行此命令的话，相当于有机会重新编辑提交说明，但将要提交的文件快照和之前的一样。

《奔跑吧linux内核》高伟视频已经上线  
请关注<https://shop115683645.taobao.com/>

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>



# git入门三：分支管理

笨叔叔

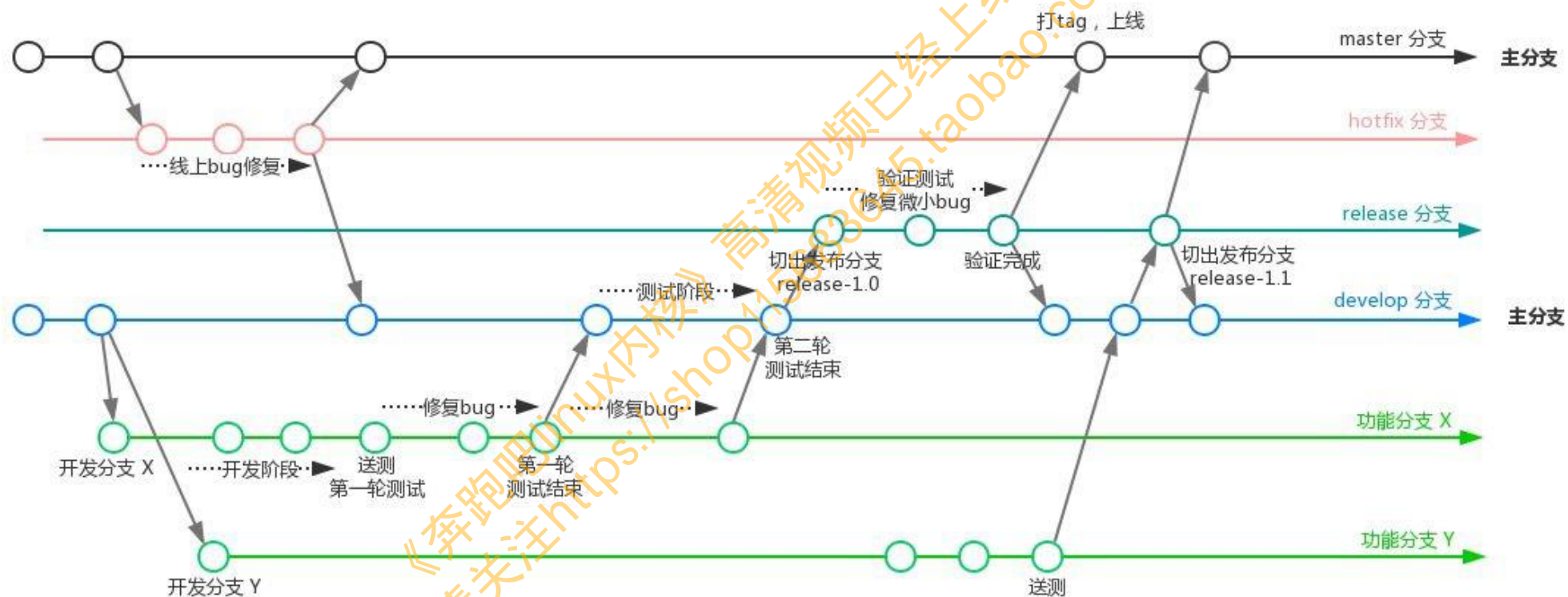
《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# 啥是分支？

- 啥是分支？
- git这个工具天生就对分支情有独钟



# 实际项目中分支管理





# 分支的使用

- 创建分支

\$ git branch 分支名

\$ git branch ben/dev (斜杠创建一个分层的命名)

- 查看分支

\$ git branch -a

\$ git show-branch -a

```
ben@benhushu:~/work/runninglinuxkernel_4.0$ git show-branch -a
! [figo/dev] create rlk_lab
* [master] create rlk_lab
! [origin/HEAD] create rlk_lab
! [origin/debug] add debug code for RLK Training vedio
! [origin/master] create rlk_lab
! [origin/rlk_basic] rlk_basic: add lab9 for chapter_5
-----
+ [origin/rlk_basic] rlk_basic: add lab9 for chapter_5
+ [origin/rlk_basic^] lab: add some example drivers
+ [origin/rlk_basic~2] add some experiment lab
+*+*+* [figo/dev] create rlk_lab
+*+*+* [figo/dev^] remove enable-kvm from run.sh (#8)
+*+*+* [figo/dev~2] update script for x86 and arm platforms (#7)
+*+*+* [figo/dev~3] update busybox to v1.28.3 for x86
+*+*+* [figo/dev~4] update busybox to v1.28.3 for arm64
+*+*+* [figo/dev~5] update busybox to v1.28.3 for arm32
+ [origin/debug] add debug code for RLK Training vedio
- - - - - [figo/dev~6] Merge pull request #3 from Figo-zhang/patch-1
```

- 切换分支

\$git checkout 分支名

\$ git checkout -b 分支名 （创建新分支并切换到新分支上）

- 删除分支

\$ git branch -D 分支名 （删除新分支）

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 合并分支

- `git merge xxx` : 将xxx分支合并到当前分支
- `git rebase xxx`: 将xxx分支合并到当前分支
- 它们两个之间有啥区别呢?

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 实验2: git合并分支实验

- 假设一个git repo里有master分支和dev分支，它们在合并之前的状态是这样的：那当它们分别执行如下语句之后，会变成什么样子？

```
$ git merge master
```

```
$ git rebase master
```

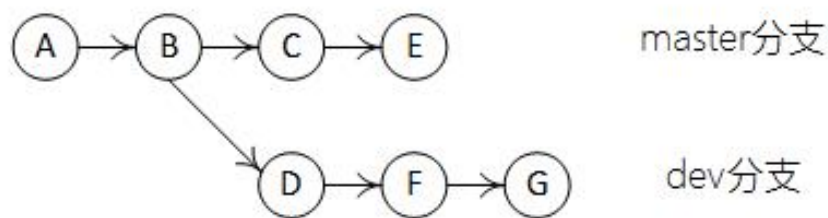


图2.12 执行合并分支之前

Graph	Short Log	Author	Author Date
	Nothing to commit	-	
●	dev origin/dev Node G	Ben Shushu<runninglinuxker...	2018/7/8 下午12:49
●	Node F	Ben Shushu<runninglinuxker...	2018/7/8 下午12:48
●	Node D	Ben Shushu<runninglinuxker...	2018/7/8 下午12:45
●	master origin/master Node E	Ben Shushu<runninglinuxker...	2018/7/8 下午12:47
●	Node C	Ben Shushu<runninglinuxker...	2018/7/8 下午12:43
●	Node B	Ben Shushu<runninglinuxker...	2018/7/8 下午12:41
●	Node A	Ben Shushu<runninglinuxker...	2018/7/8 下午12:39
●	init git repo	Ben Shushu<runninglinuxker...	2018/7/8 下午12:37

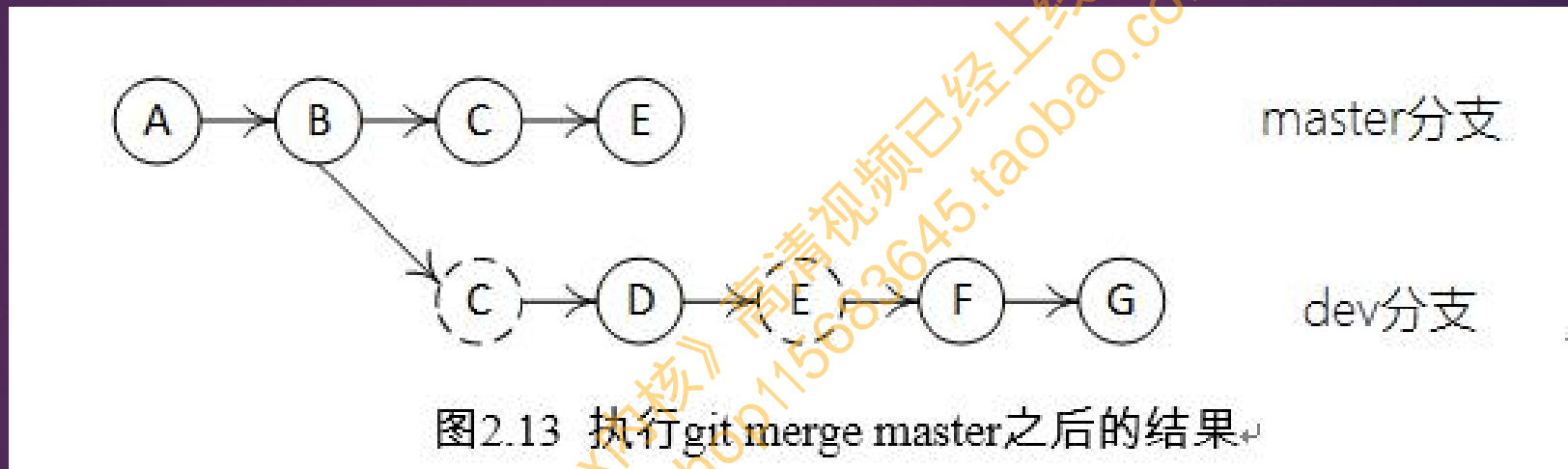
其中ABCDEFG这些commit节点的提交是按照时间顺序的

表 2.8 节点提交时间表

节点	提交时间
A	1号
B	2号
C	3号
D	4号
E	5号
F	6号
G	7号

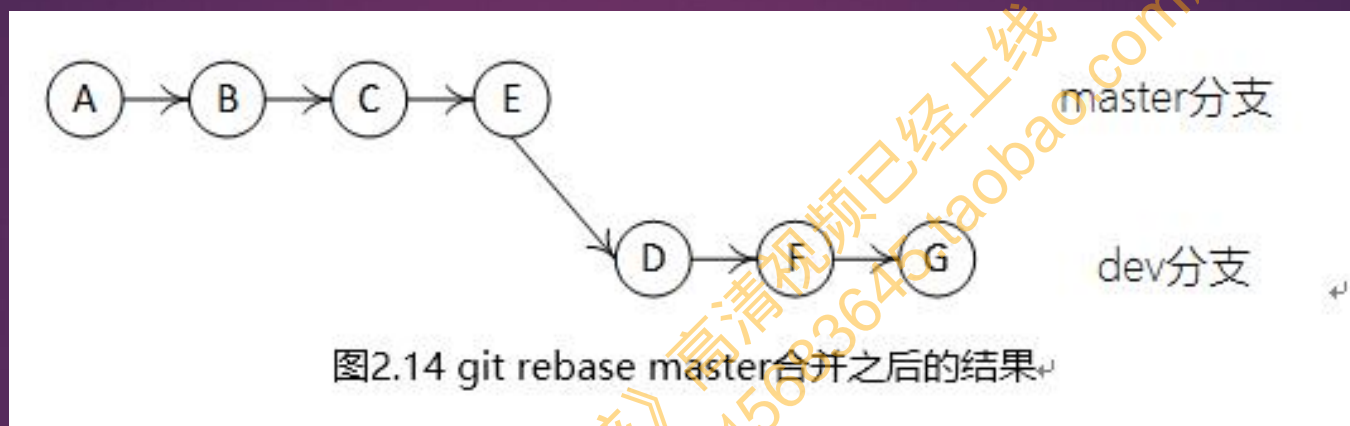


\$ git merge master之后 dev分支的情况



我们可以看到git merge master命令执行之后，dev分支上的提交都是基于时间轴来合并的。

\$ git rebase master之后， dev分支的情况



`git rebase`命令是用来改变一串提交基于那个分支为基础，如`git rebase master`就是把`dev`分支的D、F和G这三个提交基于最新的`master`分支上，也就是基于E这个提交之上。

# merge和rebase的区别

- 区别：
  - merge: 两个分支按照提交时间顺序揉在一起
  - rebase: 站在另外一个分支的肩膀上
- 用途：
  - 当你需要合并别人的修改，可以考虑使用merge命令，如项目管理上需要合并其他开发者的分支。
  - 当你的开发工作或者提交的补丁需要基于某个分支之上，那用rebase命令，如给Linux内核社区提交补丁。

《奔跑吧Linux内核》高清视频已经上线  
请关注<https://shimo115603645.taobao.com/>

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git入门实战4


## 冲突解决

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 实验3：合并分支冲突

- 实验3：在本地创建一个git repo，创建master分支和dev分支。人为在这两个分支中创造冲突。
  - 使用git merge命令进行合并分支，把dev分支合并到master上
  - 使用git rebase命令进行合并分支，把dev分支合并到master上
  - 从dev分支最新commit生成一个patch发给master分支，并在master分支上打上这个patch

Rev list			
Graph	Short Log	Author	Author Date
	Nothing to commit	-	
	<b>master</b> add int j	Ben Shushu<runninglinuxk...	7/8/18 5:20 PM
	<b>dev</b> malloc 100 bytes	Ben Shushu<runninglinuxk...	7/8/18 5:19 PM
	hello world	Ben Shushu<runninglinuxk...	7/8/18 5:17 PM
	Node A	Ben Shushu<runninglinuxk...	7/8/18 4:02 PM

# git merge 分支冲突解决

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115687945.taobao.com/>

## 1. git merge dev

```
ben@benshushu:~/work/test$ git merge dev
Auto-merging test.c
CONFLICT (content): Merge conflict in test.c
Automatic merge failed; fix conflicts and then commit the result.
ben@benshushu:~/work/test$
```

## 2. 查看冲突, vim test.c, 然后手工修改冲突

```
#include <stdio.h>

int main()
{
<<<<<< HEAD
    int i;
    int j = 5;
=====
    int i = 10;
    char *buf;

    buf = malloc(100);
>>>>>> dev

    printf("hello word\n");

    return 0;
}
```

手工修改冲突

```
#include <stdio.h>

int main()
{
    int i = 10;
    int j = 5;
    char *buf;

    buf = malloc(100);

    printf("hello word\n");

    return 0;
}
```



### 3. 添加修改文件

```
ben@benshushu:~/work/test$ git add test.c
```

### 4. 使用git merge --continue 继续下一个冲突的地方

```
ben@benshushu:~/work/test$ git merge --continue  
[master 9ad3b85] Merge branch 'dev'
```

# git rebase 分支冲突解决

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115663645.taobao.com/>

## 1. git rebase dev

```
ben@benshushu:~/work/test_rebase$ git rebase dev
First, rewinding head to replay your work on top of it...
Applying: add int j
Using index info to reconstruct a base tree...
M    test.c
Falling back to patching base and 3-way merge...
Auto-merging test.c
CONFLICT (content): Merge conflict in test.c
error: Failed to merge in the changes.
Patch failed at 0001 add int j
Use 'git am --show-current-patch' to see the failed patch

Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
```

## 2. 查看冲突, vim test.c, 然后手工修改冲突

```
test.c
1  #include <stdio.h>
2
3  int main()
4  {
5  <<<<<< HEAD
6      int i = 10;
7  W      char *buf;
8
9  W      buf = malloc(100);
10 <=====
11      int i;
12  W      int j = 5;
13 <>>>>>> add int j
14
15      printf("hello word\n");
16
17      return 0;
18 }
```

手工修改冲突

```
#include <stdio.h>

int main()
{
    int i = 10;
    int j = 5;
    char *buf;

    buf = malloc(100);

    printf("hello word\n");

    return 0;
}
```

### 3. 添加修改文件

```
ben@benshushu:~/work/test$ git add test.c
```

### 4. 使用git rebase --continue 继续下一个冲突的地方

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git am 冲突解决

《奔跑吧linux内核》书籍视频已经上线  
请关注<https://shop115645645.taobao.com/>

## 1. 当git am xxx.patch发生冲突的时候

```
ben@benshushu:~/work/test_am$ git am 0001-malloc-100-bytes.patch
Applying: malloc 100 bytes
error: patch failed: test.c:2
error: test.c: patch does not apply
Patch failed at 0001 malloc 100 bytes
Use 'git am --show-current-patch' to see the failed patch
When you have resolved this problem, run "git am --continue".
If you prefer to skip this patch, run "git am --skip" instead.
To restore the original branch and stop patching, run "git am --abort".
```

2.

\$ git am --show-current-patch 查看当前的patch

\$ git apply PATCH --reject

\$ edit edit edit

（在对应的文件目录下面，根据rej文件手动解决所有冲突）

\$ git add FIXED\_FILES

\$ git am --resolved

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>



# git入门与实战 6

## 远程仓库

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# 什么是远程仓库？

- 远程仓库和本地仓库区别

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git clone命令

- git clone把远程仓库下载到本地
- .git/config文件

```
1 [core]
2     repositoryformatversion = 0
3     filemode = true
4     bare = false
5     logallrefupdates = true
6 [remote "origin"]
7     url = https://gitee.com/benshushu/runninglinuxkernel_4.0.git
8     fetch = +refs/heads/*:refs/remotes/origin/*
9 [branch "master"]
10     remote = origin
11     merge = refs/heads/master
```

```
ben@benshushu:~/work/runninglinuxkernel_4.0/.git/refs$ tree
```

```
.
├── heads
│   ├── ben
│   ├── figo
│   └── dev
├── master
├── remotes
│   └── origin
│       └── HEAD
└── tags
```

```
5 directories, 4 files
```

```
ben@benshushu:~/work/runninglinuxkernel_4.0/.git/refs$
```

## git remote命令

- git remote命令用来管理远程仓库

➤git remote命令是可以让我们添加一个远程仓库本地中。

# git remote add ben 远程仓库地址

➤git remote show xxx， 可以查看这个xx远程版本库的所有信息。

➤git remote update 抓取远程版本库中所有可用更新到本地版本库里

➤git fetch xxx 抓取远程xx版本库到本地

《奔跑吧！Linux内核》高清视频已经上线  
请关注<https://shop115683649.taobao.com/>

## 添加和删除远程分支

- 添加远程分支:

# git push <远程主机名> <本地分支名>: <远程分支名>

例子:

```
$ git push origin master
```

```
$ git push origin master:dev
```

- 删除远程分支名

```
# git push <远程主机名> :<远程分支名>
```

```
# git push <远程主机名> -d <远程分支名>
```

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git入门与实战 7

## 实战git和内核开发与管理

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>



## 实验4：基于Linux 4.0进行内核开发和管理

- 本实验通过模拟一个项目的实际操作来演示，如何利用git来进行Linux内核开发和管理。
- 某公司的linux设备驱动开发和维护项目。
- 该项目的需求如下：
  - 该项目需要基于Linux-4.0内核为基础进行二次开发，主要是开发设备驱动，以及项目后期维护。
  - 创建一个ben-linux-dev的git仓库，需要包含所有Linux 4.0的git log信息，然后基于这个git 仓库进行项目开发。

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 本实验用到的git命令和技巧

- git –bare init
- git clone
- git commit
- git remote add
- git remote –v
- git fetch
- git reset
- git merge
- git push

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git入门与实战 8

实战rebase到最新代码

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683649.taobao.com/>

## 实验5：实战git开发之rebase到最新代码

- 项目需求：基于实验4.

- 首先需要在Linux 4.0上做开发。为了简化开发，我们假设只需要修改Linux 4.0根目录下面的Makefile，修改如下：

- ✓ VERSION = 4

- ✓ PATCHLEVEL = 0

- ✓ SUBLEVEL = 0

- ✓ EXTRAVERSION =

- ✓ NAME = Hurr durr I'ma sheep //修改这里，改成 benshushu

- 把修改推送到远程仓库上。

- 过了几个月，这个项目需要rebase到Linux 4.15的内核，并且把之前做的工作也需要rebase到Linux 4.15内核，并且更新到远程仓库上。rebase时遇到冲突，需要修复冲突。

# 本实验用到的git命令和技巧

- 这个实验主要是 考察大家对分支合并和冲突解决的能力
  - 在这个实验里，会学习到如何合并一个分支以及如何rebase到最新的master分支上。
  - 在合并分支和rebase分支的过程中，可能会遇到冲突，在本实验中学会如何修复冲突。
- 在实际项目开发和管理中，非常常见

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>



# git入门与实战 9

如何给Linux内核社区发补丁

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 订阅邮件列表

内核邮件列表（Linux Kernel Mailing List，简称LKML），是内核开发者进行发布、讨论、技术辩论的主战场

- 有兴趣的读者可以订阅这个邮件列表，订阅办法如下：

- ✓ 发送如下信息到[majordomo@vger.kernel.org](mailto:majordomo@vger.kernel.org)中

`subscribe linux-kernel <your@email.com>`

《奔跑吧linux内核》  
请关注<https://shop115603645.taobao.com/>

# 订阅子模块的邮件列表

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

## Majordomo lists at VGER.KERNEL.ORG

**REMEMBER:** Subscription to these lists go via <majordomo@vger.kernel.org> !

Note about archives: Listed archives are those that have been reported to vger's maintainers, or that we have found out otherwise. As things are, *list of archives is not complete*.

[alsa-devel](#), [autofs](#), [backports](#), [ceph-devel](#), [cgroups](#), [cpufreq](#), [dash](#), [dcp](#), [devicetree-compiler](#), [devicetree-spec](#), [devicetree](#), [dmaengine](#), [dwarves](#), [ecryptfs](#), [flo](#), [fstests](#), [git-commits-24](#), [git-commits-head](#), [git](#), [hail-devel](#), [initramfs](#), [ltda-users](#), [kernel-janitors](#), [kernel-packagers](#), [kernel-testers](#), [keyrings](#), [kvm-commits](#), [kvm-ia64](#), [kvm-ppc](#), [kvm](#), [lartc](#), [libzbc](#), [linux-8086](#), [linux-acpi](#), [linux-admin](#), [linux-alpha](#), [linux-aoi](#), [linux-apps](#), [linux-arch](#), [linux-arm-msm](#), [linux-assembly](#), [linux-bbs](#), [linux-bcache](#), [linux-block](#), [linux-bluetooth](#), [linux-bi-ace](#), [linux-btrfs](#), [linux-c-programming](#), [linux-can](#), [linux-cifs](#), [linux-clk](#), [linux-config](#), [linux-console](#), [linux-coverity](#), [linux-crypto](#), [linux-diald](#), [linux-doc](#), [linux-edac](#), [linux-efi](#), [linux-embedded](#), [linux-ext4](#), [linux-fbdev](#), [linux-fido](#), [linux-fpga](#), [linux-fscrypt](#), [linux-fsdevel](#), [linux-fsf](#), [linux-ftp](#), [linux-gcc](#), [linux-gpio](#), [linux-hams](#), [linux-hexagon](#), [linux-hotplug](#), [linux-hwmon](#), [linux-i2c](#), [linux-ia64](#), [linux-ibcs2](#), [linux-ide](#), [linux-ilo](#), [linux-input](#), [linux-integrity](#), [linux-ipc](#), [linux-isdn](#), [linux-japanese](#), [linux-kbuild](#), [linux-kernel-announce](#), [linux-kernel-announce.posters](#), [linux-kernel](#), [linux-kseltest](#), [linux-laptop](#), [linux-leds](#), [linux-linuxss](#), [linux-lugnuts](#), [linux-m68k-cvscommit](#), [linux-m68k](#), [linux-man](#), [linux-mca](#), [linux-media](#), [linux-metag](#), [linux-mmc](#), [linux-modules](#), [linux-msdos-devel](#), [linux-msdos](#), [linux-new-lists](#), [linux-newbie](#), [linux-next](#), [linux-nfs](#), [linux-nlfs](#), [linux-numa](#), [linux-omap](#), [linux-opengl](#), [linux-parisc](#), [linux-pci](#), [linux-perf-users](#), [linux-pm](#), [linux-ppp](#), [linux-pwm](#), [linux-raid](#), [linux-rdma](#), [linux-remoteproc](#), [linux-renesas-soc](#), [linux-rt-users](#), [linux-rtc](#), [linux-s390](#), [linux-samsung-soc](#), [linux-scsi](#), [linux-sctp](#), [linux-security-module](#), [linux-serial](#), [linux-sh](#), [linux-smp](#), [linux-soc](#), [linux-sound](#), [linux-sparse](#), [linux-spi](#), [linux-standards](#), [linux-syscall](#), [linux-tape](#), [linux-tegra](#), [linux-tip-commits](#), [linux-trace-devel](#), [linux-trace-users](#), [linux-unionfs](#), [linux-usb](#), [linux-userfs](#), [linux-watchdog](#), [linux-wireless](#), [linux-word](#), [linux-wpan](#), [linux-x11](#), [linux-x25](#), [linux-x86\\_64](#), [linux-xfs](#), [live-patching](#), [ipc-netdev](#), [lvs-devel](#), [mm-commits](#), [netdev](#), [netfilter-devel](#), [netfilter](#), [netbook](#), [platform-driver-x86](#), [reiserfs-devel](#), [smatch](#), [sparclinux](#), [stable-commits](#), [stable-rt](#), [stable](#), [stgt](#), [target-devel](#), [trinity](#), [ultralinux](#), [util-linux](#), [xdp-newbies](#).

# 给社区发补丁三部曲 1

- 如何发现内核缺陷

- 查找编译警告。
- 编码规范。
- 其他人新提交的补丁集或者staging源代码。

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 给社区发补丁三部曲 2

- 如何制作补丁

- 基于Linux内核主仓库最新的主分支创建一个新的分支。

```
$git checkout -b "my-fix"
```

- 修改文件。这一步很重要重要的是进行测试，包括编译测试、单元测试和功能测试等。

- 生成新的提交（commit）。

- 生成补丁。

```
$git format-patch -1 #生成一个补丁
```

- 对补丁进行代码格式检查

```
$/scripts/checkpatch.pl your_fix.patch
```

《奔跑吧Linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## 给社区发补丁三部曲 3

- 如何发送补丁

➤我们推荐使用git send-email工具来发送补丁到内核社区。安装git send-email工具。

```
$ sudo apt-get install git-email
```

➤配置send-email。修改~/.gitconfig文件，增加如下配置。

```
<~/.gitconfig>
```

```
[sendemail]
```

```
smtpencryption = tls
```

```
smtpserver = smtp.126.com
```

```
smtpuser = figo1802@126.com
```

```
smtpserverport = 25
```

➤使用get\_maintainer.pl来获取这些维护者的名字和邮箱地址。

```
$ ./scripts/get_maintainer.pl your_fix.patch
```

➤最后就可以发送你的补丁，可以使用如下命令来发送：

```
$ git send-email --to "tglx@linutronix.de" 0001-your-fix-patch.patch
```

**发送补丁之前请仔细检查！！**

《奔跑吧linux内核》高清视频已上线  
请关注<https://shop116683645.taobao.com/>



Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# git入门与实战 5

## 更改提交

笨叔叔

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## git commit --amend

- 修改最近一次提交，流程如下：

- 修改你需要修改的文件

- git diff 查看

- git add

- git commit --amend

- 如何修改最近一个提交的作者？

- `$ git commit -amend -author "xxx <xxx@126.com>"`

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

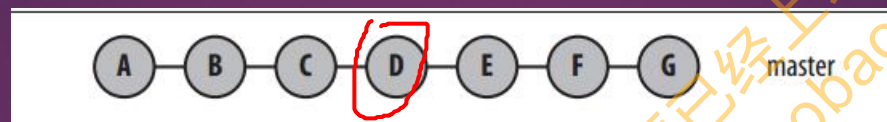
## git reset命令

- git reset命令会跳转HEAD引用执行某个给定的提交
  - --soft: 将HEAD引用指向给定的提交, 索引和工作目录的内容保持不变。
  - --mixed: 将HEAD引用指向给定的提交, 索引内容跟着改变, 工作目录的内容保持不变。
  - --hard: 将HEAD引用指向给定的提交, 索引和工作目录的内容保持都发生改变。
- git reset命令具有破坏性, 使用时要小心

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

## git revert命令

- git revert命令用来把某个提交给撤销了。
- git revert master~3



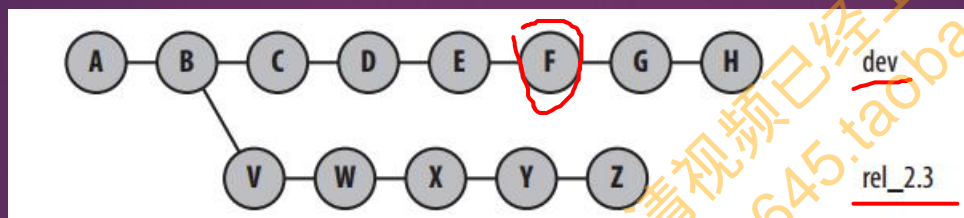
git revert之前



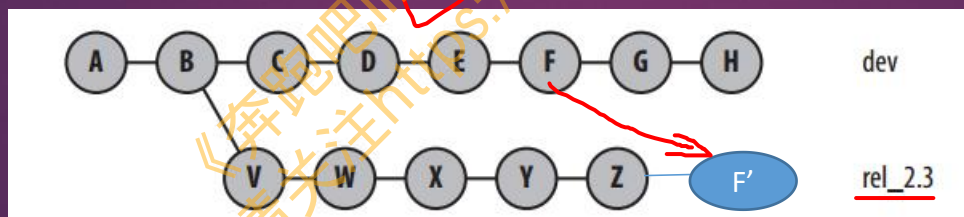
git revert之后

## git cherry-pick

- 从某个git tree或者分支上摘取一个commit到当前分支
- 把dev分支上的F节点commit摘取到rel-2.3分支上



```
$ git checkout rel_2.3  
$ git cherry-pick dev~2
```



Thanks

《奔跑吧linux内核》高清视频已经上线  
请关注<https://shop115883645.taobao.com/>



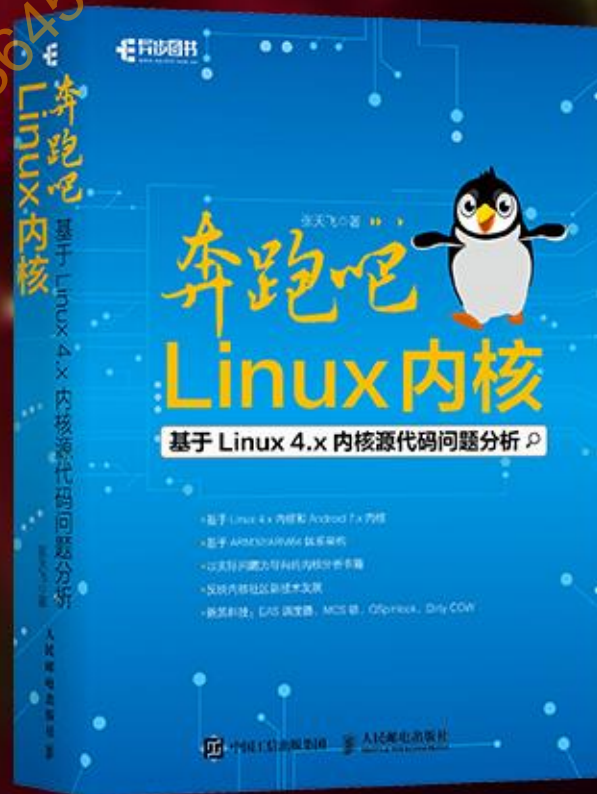
# 奔跑吧Linux社区介绍





# 配套视频 旗舰篇

## 第 1 季 内存管理



# 课程简介

《奔跑吧linux内核》一书是笨叔叔经过长达3年时间创造而成，于2017年8月底在由人民邮电出版社异步社区正式出版。出版后得到国内以及海外linux从业人员和爱好者的喜欢。有很多阅读了本书的读者朋友希望能有一个配套的视频教学节目，笨叔叔也觉得图书+视频的方式可以让读者更好的理解Linux内核的运行机制和原理，把快速地运用到实际工作中。

本季视频节目是《奔跑吧linux内核》配套视频的第一季，主题和内存管理相关。其中前20期节目完全和《奔跑吧linux内核》第二章内容吻合。后面的内容是根据最新的技术发展进行动态的扩充，包括异构内存管理、Meltdown和Spectre漏洞分析、hugepage内存管理、内存参数调优，内存泄漏实战等等内容。《奔跑吧》的小伙伴只需要一次订阅，终生免费更新，后续笨叔会源源不断的不定期更新。

# 课程特色

 一次订阅，免费更新

 全新模式“图书+视频”

 跟踪Linux最新技术发展，永不落空

 笨叔笨色出演，绝非电脑录屏

 学员可按需点播新知识点



# 为何选择笨叔的视频？

- 笨叔亲自出演，绝非简单电脑录屏。
  - 电脑录屏制作的节目，只能看到字幕，看不到人，这种节目一定枯燥乏味，除非是老罗来讲。
- 专业录音棚录制，1080P高清+广播级音质。
  - 目前大部分计算机视频节目都是采用电脑录屏方法制作，其音质差和噪声大，看这种视频其实是一种折磨。
- 全新模式，图书+视频。
- 全球首创，一键订阅，免费更新。知识永不落后！

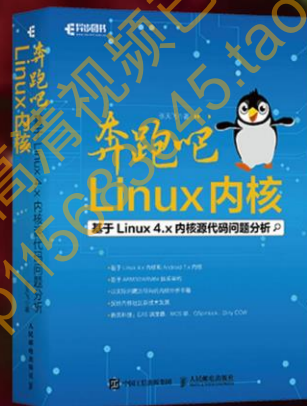
《笨叔Linux内核》高清视频已经上线  
请关注<https://shop115683645.taobao.com/>

# 奔跑吧linux内核教学视频

旗舰篇一次订阅，持续更新

配套视频 **旗舰篇**

第**1**季  
内存管理



官方淘宝店



微信公众号

课程优势

1. 最有深度和广度的 Linux 内核视频
2. 手把手解读 Linux 内核代码
3. 紧跟 Linux 内核社区技术热点
4. 一键订阅，持续更新
5. 图书 + 视频，全新学习模式
6. 笨叔叔的 VIP 私密微信群答疑

第一季内存管理旗舰篇课程目录

课程名称	时长
序言一：Linux内核学习方法论	0:09:13
序言二：学习前准备	
序言2.1 Linux发行版和开发板的选择	0:13:56
序言2.2 搭建Qemu+gdb单步调试内核	0:13:51
序言2.3 搭建Eclipse图形化调试内核	0:10:59
实战运维1：查看系统内存信息的工具（一）	0:20:19
实战运维2：查看系统内存信息的工具（二）	0:16:32
实战运维3：读懂内核log中的内存管理信息	0:25:35
实战运维4：读懂 proc meminfo	0:27:59
实战运维5：Linux运维能力进阶线路图	0:09:40
实战运维6：Linux内存管理参数调优（一）	0:19:46
实战运维7：Linux内存管理参数调优（二）	0:31:20
实战运维8：Linux内存管理参数调优（三）	0:22:58
运维高级如何单步调试RHEL—CENTOS7的内核一	0:15:45
运维高级如何单步调试RHEL—CENTOS7的内核二	0:41:28
vim:打造比source insight更强更好用的IDE（一）	0:24:58
vim:打造比source insight更强更好用的IDE（二）	0:20:28
vim:打造比source insight更强更好用的IDE（三）	0:23:25
实战git项目和社区patch管理	
2.0 Linux内存管理背景知识介绍	
奔跑2.0.0 内存管理硬件知识	0:15:25
奔跑2.0.1 内存管理总览一	0:23:27
奔跑2.0.2 内存管理总览二	0:07:35
奔跑2.0.3 内存管理常用术语	0:09:49
奔跑2.0.4 内存管理究竟管些什么东西	0:28:02
奔跑2.0.5 内存管理代码框架导读	0:38:09
2.1 Linux内存初始化	
奔跑2.1.0 DDR简介	0:06:47
奔跑2.1.1 物理内存三大数据结构	0:19:39
奔跑2.1.2 物理内存初始化	0:11:13
奔跑2.1 内存初始化之代码导读一	0:43:54
奔跑2.1 内存初始化之代码导读二	0:23:31
奔跑2.1 代码导读C语言部分（一）	0:27:34
奔跑2.1 代码导读C语言部分（二）	0:21:28
2.2 页表的映射过程	
奔跑2.2.0 ARM32页表的映射	0:08:54
奔跑2.2.1 ARM64页表的映射	0:10:58
奔跑2.2.2 页表映射例子分析	0:11:59
奔跑2.2.3 ARM32页表映射那些奇葩的事	0:09:42
2.3 内存布局图	
奔跑2.3.1 内存布局一	0:10:35
奔跑2.3.2 内存布局二	0:13:30
2.4 分配物理页面	
奔跑2.4.1 伙伴系统原理	0:10:10

课程名称	时长
奔跑2.4.2 Linux内核中的伙伴系统和碎片化	0:11:14
奔跑2.4.3 Linux的页面分配器	0:21:37
2.5 slab分配器	
奔跑2.5.1 slab原理和核心数据结构	0:18:36
奔跑2.5.2 Linux内核中slab机制的实现	0:16:56
2.6 vmalloc分配	
奔跑2.6 vmalloc分配	0:15:48
2.7 VMA操作	
奔跑2.7 VMA操作	0:16:42
2.8 malloc分配器	
奔跑2.8.1 malloc的三个迷惑	0:17:41
奔跑2.8.2 内存管理的三个重要的函数	0:17:38
2.9 mmap分析	
奔跑2.9 mmap分析	0:23:14
2.10 缺页中断处理	
奔跑2.10.1 缺页中断一	0:31:07
奔跑2.10.2 缺页中断二	0:16:58
2.11 page数据结构	
奔跑2.11 page数据结构	0:29:41
2.12 反向映射机制	
奔跑2.12.1 反向映射机制的背景介绍	0:19:01
奔跑2.12.2 RMAP四部曲	0:07:31
奔跑2.12.3 手撕Linux2.6.11上的反向映射机制	0:07:35
奔跑2.12.4 手撕Linux4.x上的反向映射机制	0:10:08
2.13 回收页面	
奔跑2.13 页面回收一	0:16:07
奔跑2.13 页面回收二	0:11:41
2.14 匿名页面的生命周期	制作中会更新
2.15 页面迁移	制作中会更新
2.16 内存规整	制作中会更新
2.17 KSM	制作中会更新
2.18 Dirty COW内存漏洞	制作中会更新
2.19 内存数据结构和API总结	制作中会更新
2.20 Melttdown漏洞分析	
奔跑2.20.1 Melttdown背景知识	0:10:13
奔跑2.20.2 CPU体系结构之指令执行	0:11:25
奔跑2.20.3 CPU体系结构之乱序执行	0:11:03
奔跑2.20.4 CPU体系结构之异常处理	0:03:48
奔跑2.20.5 CPU体系结构之cache	0:10:56
奔跑2.20.6 进程地址空间和页表及TLB	0:17:39
奔跑2.20.7 Melttdown漏洞分析	0:06:04
奔跑2.20.8 Melttdown漏洞分析之x86篇	0:12:07
奔跑2.20.9 ARM64上的KPTI解决方案	0:25:39
2.21 spectre漏洞分析	制作中会更新
2.22 异构内存管理	制作中会更新
2.23 Hugepage巨页	制作中会更新
2.24 运维人员必会的内存调优	制作中会更新
2.25 实战内存泄漏	制作中会更新
2.26 从内存管理代码中学会的优化技巧	制作中会更新

后期课程不定期更新中，等您来提交topic

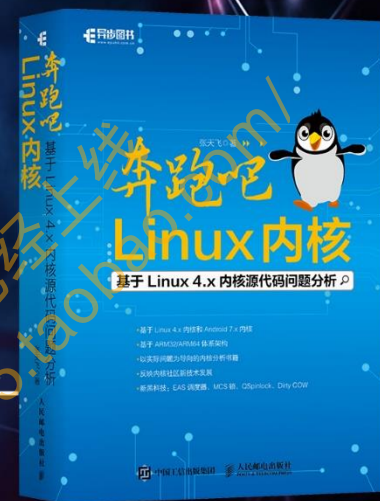
截至 18 年 5 月已录制完成约 20 小时， 后续精彩视频不断



震撼上线

# 全书配套视频

第1季 内存管理篇



更多高清和精彩节目尽在：淘宝：[shop115683645.taobao.com](https://shop115683645.taobao.com)