目录

[基本操作： 2](#_Toc472103857)

[概念介绍： 2](#_Toc472103858)

[设置邮箱地址和姓名： 3](#_Toc472103859)

[提高git操作输出颜色（增加可读性）： 4](#_Toc472103860)

[Git—仓库初始化： 4](#_Toc472103861)

[git status——查看仓库的状态 4](#_Toc472103862)

[git add——向暂存区中添加文件 4](#_Toc472103863)

[gitcommit——保存仓库的历史记录 5](#_Toc472103864)

[gitlog——查看提交日志 5](#_Toc472103865)

[git diff——查看更改前后的差别 5](#_Toc472103866)

[查看工作树和最新提交的差别 6](#_Toc472103867)

[分支操作： 6](#_Toc472103868)

[git branch——显示分支一览表 7](#_Toc472103869)

[git checkout -b——创建、切换分支 7](#_Toc472103870)

[git merge——合并分支 8](#_Toc472103871)

[git log --graph——以图表形式查看分支 9](#_Toc472103872)

[git reset——回溯历史版本 9](#_Toc472103873)

[创建fix-B，并修改holdmc-git.md， 11](#_Toc472103874)

[下面操作的目标： 11](#_Toc472103875)

[恢复到 feature-A 分支合并后的状态 12](#_Toc472103876)

[git commit --amend——修改提交信息 13](#_Toc472103877)

[Example: 15](#_Toc472103878)

[建立uboot git库 15](#_Toc472103879)

[初始化uboot git库： 15](#_Toc472103880)

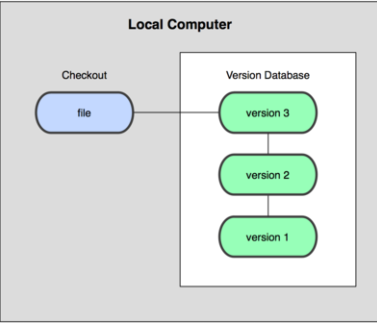
## 基本操作：

### 概念介绍：

版本控制类型：

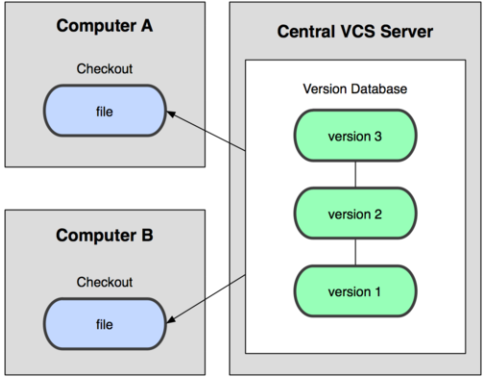
a.本地版本控制系统

复制整个项目目录的方式来保存不同的版本，这么做相对比较简单，但是弄错了文件或者丢了数据则没有后退的路。



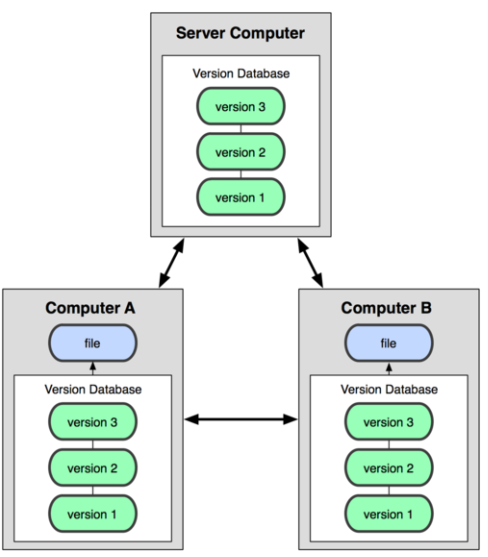
b.集中化的版本控制系统(CVCS: Centralized Version Control Systems)

解决在不同系统上的开发人员协同工作。即将所有的文件的修订版使用一个单一的集中管理服务器，让协同开发人员通过客户端连接到这台服务器，取出最新的文件或者提交更新。



c.分布式版本控制系统(DVCS :Distributed Version Control System)

在这类系统中，诸如 Git，Mercurial，Bazaar 还有 Darcs 等，客户端并不只提取最新版本的文件快照，而是把原始的代码仓库完整地镜像下来。这么一来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复。因为每一次的提取操作，实际上都是一次对代码仓库的完整备份



### 设置邮箱地址和姓名：

$ git config --global user.name "chen qigan"

$ git config --global user.email [chenqiganmail@foxmail.com](mailto:chenqiganmail@foxmail.com)

### 提高git操作输出颜色（增加可读性）：

$ git config --global color.ui auto

以上配置即为生成：

$ cat ~/.gitconfig

这些命令，会在“ ~/.gitconfig”中以如下形式输出设置文件

$ cat ~/.gitconfig

[ganye]

king = chen qigan

chenqiganmail = chenqiganmail@foxmail.com

[user]

email = chenqiganmail@foxmail.com

name = chen qigan

[color]

ui = auto

### Git—仓库初始化：

mkdir git-guide

cd git-guide/

git init

Initialized empty Git repository in /home/holdmc/work/git-test/git-guide/.git/

$ ls -a

. .. .git

Note：生成.git文件夹即表示git仓库初始化成功

### git status——查看仓库的状态

$ touch holdmc-git.md

$ git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

holdmc-git.md

nothing added to commit but untracked files present (use "git add" to track)

NOTE：

在仓库下单单建立文件git是不跟踪的，可以看提示：Untracked files:

### git add——向暂存区中添加文件

$ git add holdmc-git.md

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: holdmc-git.md

NOTE:

使用git add之后holdmc-git.md的任何变化都将被git跟踪，输入git status holdmc-git.md可查看文件变化信息。

### git commit——保存仓库的历史记录

将当前暂存区中的文件实际保存到仓库的历史记录中，也就是将之前git add的文件真正提交。

$ git commit -m "First commit from HOLDMC"

[master (root-commit) a6bde1e] First commit from HOLDMC

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 holdmc-git.md

NOTE:

-m <msg>, --message=<msg>

Use the given <msg> as the commit message.

-m即为对提交的历史记录设置一个简介。

### git log——查看提交日志

$ git log

commit a6bde1e1d78fdbea59821ff0622774356a1c7cf7

Author: chen qigan <chenqiganmail@foxmail.com>

Date: Fri Jan 13 15:01:41 2017 +0800

First commit from HOLDMC

指定显示目录或文件的log：git file\_name or dir\_name log

Note：查看以往仓库中提交的日志，包括什么时候什么人进行了什么操作以及操作前后的差别。

黑色部分为这个提交的嘻哈值，最喜爱git中所有的索引就使用这种嘻哈值进行的而非名字。

在后面数据回溯哦，历史推进都有用处。

### git diff——查看更改前后的差别

为了体现效果：

向holdmc-git.md添加 # HOLDMC GIT

$ vi holdmc-git.md

$ git diff

diff --git a/holdmc-git.md b/holdmc-git.md

index e69de29..3e6f6ae 100644

--- a/holdmc-git.md

+++ b/holdmc-git.md

@@ -0,0 +1 @@

+# HOLDMC GIT

NOTE：

这里解释一下显示的内容。“ +”号标出的是新添加的行，被删除的行则用“ -”号标出。我们可以看到，这次只添加了一行

### 查看工作树和最新提交的差别

$ git diff HEAD

diff --git a/holdmc-git.md b/holdmc-git.md

index e69de29..3e6f6ae 100644

--- a/holdmc-git.md

+++ b/holdmc-git.md

@@ -0,0 +1 @@

+# HOLDMC GIT

NOTE:可以看到和第一次刚刚建立文件的时候的差别，这里跟使用git diff效果一样只是巧合。

在执行 git commit命令之前先执行git diff HEAD命令，查看本次提交与上次提交之间有什么差别，等确认完毕后再进行提交。

HEAD：指向当前分支中最新一次提交的指针。

#### 提交更改：

$ git add holdmc-git.md

$ git commit -m "Add index from HOLDMC"

[master 79bf69c] Add index from HOLDMC

1 file changed, 1 insertion(+)

#### 确认下提交成功：

$ git log

commit 79bf69cc883d262986adb7b46be937fbb2bfbd43

Author: chen qigan <chenqiganmail@foxmail.com>

Date: Fri Jan 13 15:18:05 2017 +0800

Add index from HOLDMC

commit a6bde1e1d78fdbea59821ff0622774356a1c7cf7

Author: chen qigan <chenqiganmail@foxmail.com>

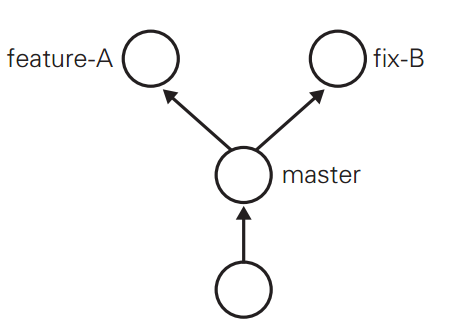
Date: Fri Jan 13 15:01:41 2017 +0800

First commit from HOLDMC

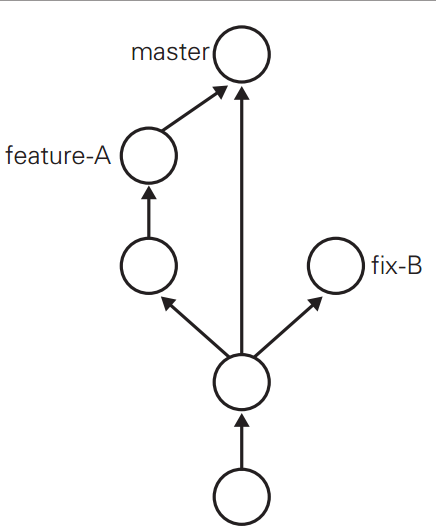
看到第二次提交。

## 分支操作：

在进行多个并行作业时，我们会用到分支。在这类并行开发的过程中，往往同时存在多个最新代码状态。master 分支是 Git 默认创建的分支，因此基本上所有开发都是以这个分支为中心进行的。



不同分支中，可以同时进行完全不同的作业。等该分支的作业完成之后再与 master 分支合并。比如 feature-A 分支的作业结束后与 master合并：



通过灵活运用分支，可以让多人同时高效地进行并行开发

### git branch——显示分支一览表

git branch命令可以将分支名列表显示，同时可以确认当前所在分支

$ git branch

\* master

NOTE”:

“ \*”（星号），表示这是我们当前所在的分支

结果中没有显示其他分支名，表示本地仓库中只存在 master 一个分支

### git checkout -b——创建、切换分支

$ git checkout -b feature-A

Switched to a new branch 'feature-A'

NOTE：

-b:将创建一个分支，并转到feature-A分支

#### 切换分支：

$ git checkout master

Switched to branch 'master'

#### 查看现在的分支：

$ git branch

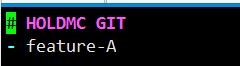
feature-A

\* master

#### 分支和分支之间的操作并不影响：

切换到feature-A分支

修改holdmc-git.md ：



$ git add holdmc-git.md

$ git commit -m "Add from feature-A branch by HOLDMC"

[feature-A aaaa7f6] Add from feature-A branch by HOLDMC

1 file changed, 1 insertion(+)

$ cat holdmc-git.md

# HOLDMC GIT

- feature-A

切换到master 分支：

$ git checkout master

Switched to branch 'master'

$ cat holdmc-git.md

# HOLDMC GIT

NOTE:这里master和feature-A分支中同一个文件holdmc-git.md内容却不同，证明分支和分支之间是独立的，这也是git的最大用处之一。利用这种特性可以在不互相影响的情况下同时进行多个功能的开发。

#### 切换回上一个分支 git checkout -

$ git status

On branch feature-A

nothing to commit, working directory clean

$ git checkout -

Switched to branch 'master'

$ git status

On branch master

nothing to commit, working directory clean

### git merge——合并分支

我们假设 feature-A 已经实现完毕，想要将它合并到主干分支 master 中。首先切换到 master 分支:

git merge --no-ff feature-A

随后编辑器会启动，用于录入合并提交的信息。

Merge branch 'feature-A'

# Please enter a commit message to explain why this merge is necessary,

# especially if it merges an updated upstream into a topic branch.

#

# Lines starting with '#' will be ignored, and an empty message aborts

# the commit.

直接退出即可。

Merge made by the 'recursive' strategy.

holdmc-git.md | 1 +

1 file changed, 1 insertion(+)

如此 feature-A分支内容则被合并到master中去了。

### git log --graph——以图表形式查看分支

$ git log --graph

\* commit 4a7e0972c6f06da819529726df63b7800eca7d98

|\ Merge: 79bf69c aaaa7f6

| | Author: chen qigan <chenqiganmail@foxmail.com>

| | Date: Fri Jan 13 15:43:42 2017 +0800

| |

| | Merge branch 'feature-A'

| |

| \* commit aaaa7f61e6466fe151ed8a4de0dcfe43b191ad6a

|/ Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:36:33 2017 +0800

|

| Add from feature-A branch by HOLDMC

|

\* commit 79bf69cc883d262986adb7b46be937fbb2bfbd43

| Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:18:05 2017 +0800

|

| Add index from HOLDMC

|

\* commit a6bde1e1d78fdbea59821ff0622774356a1c7cf7

Author: chen qigan <chenqiganmail@foxmail.com>

Date: Fri Jan 13 15:01:41 2017 +0800

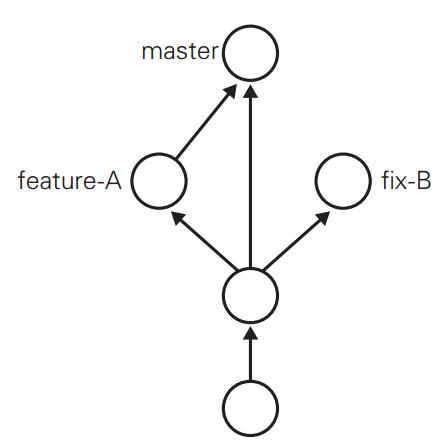
First commit from HOLDMC

NOTE:

左边的线条可以很清晰的看出从主分支产生feature-A到其中间的一些修改操作。

### git reset——回溯历史版本

可以回到任何状态：



在master1分支已经合并feature-A产生master2分支后，依然可以创建基于master1的fix-B分支。

#### 回溯到创建 feature-A 分支前

##### 查看目标时间点的哈希值：

$git log --graph

\* commit 4a7e0972c6f06da819529726df63b7800eca7d98

|\ Merge: 79bf69c aaaa7f6

| | Author: chen qigan <chenqiganmail@foxmail.com>

| | Date: Fri Jan 13 15:43:42 2017 +0800

| |

| | Merge branch 'feature-A'

| |

| \* commit aaaa7f61e6466fe151ed8a4de0dcfe43b191ad6a

|/ Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:36:33 2017 +0800

|

| Add from feature-A branch by HOLDMC

|

\* commit 79bf69cc883d262986adb7b46be937fbb2bfbd43

| Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:18:05 2017 +0800

|

| Add index from HOLDMC

|

\* commit a6bde1e1d78fdbea59821ff0622774356a1c7cf7

Author: chen qigan <chenqiganmail@foxmail.com>

Date: Fri Jan 13 15:01:41 2017 +0800

First commit from HOLDMC

这里可以清晰的看到合并feature-A前master的嘻哈值为：79bf69cc883d262986adb7b46be937fbb2bfbd43

#### git rest –hard <目标时间点的哈希值>

git reset --hard 79bf69cc883d262986adb7b46be937fbb2bfbd43

HEAD is now at 79bf69c Add index from HOLDMC

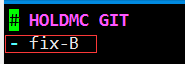
至此已经回溯到合并feature-A时的目标点。

### 创建fix-B，并修改holdmc-git.md，

$ git checkout -b fix-B

Switched to a new branch 'fix-B'

$ vi holdmc-git.md



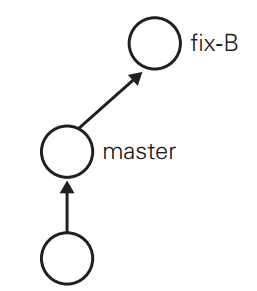
$ git add holdmc-git.md

$ git commit -m "Fix B from HOLDMC"

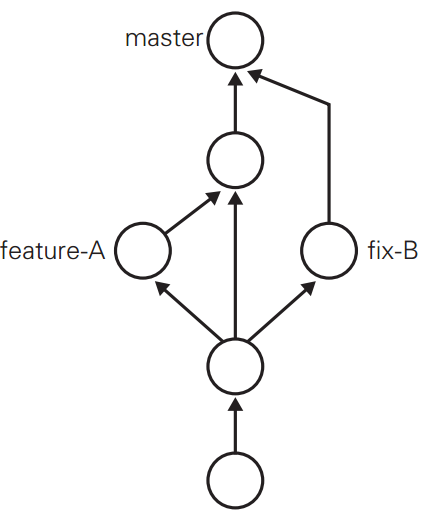
[fix-B d2ca2c7] Fix B from HOLDMC

1 file changed, 1 insertion(+)

#### 当前分支状态：



### 下面操作的目标：



### 恢复到 feature-A 分支合并后的状态

#### git reflog

git log命令只能查看以当前状态为终点的历史日志。所以这里要使用git reflog在日志中找出回溯历史之前的哈希值，通过 git reset --hard命令恢复到回溯历史前的状态。

$ git reflog

d2ca2c7 HEAD@{0}: commit: Fix B from HOLDMC

79bf69c HEAD@{1}: checkout: moving from master to fix-B

79bf69c HEAD@{2}: reset: moving to 79bf69cc883d262986adb7b46be937fbb2bfbd43

4a7e097 HEAD@{3}: merge feature-A: Merge made by the 'recursive' strategy.

79bf69c HEAD@{4}: checkout: moving from feature-A to master

aaaa7f6 HEAD@{5}: checkout: moving from master to feature-A

79bf69c HEAD@{6}: checkout: moving from feature-A to master

aaaa7f6 HEAD@{7}: commit: Add from feature-A branch by HOLDMC

79bf69c HEAD@{8}: checkout: moving from master to feature-A

79bf69c HEAD@{9}: checkout: moving from feature-A to master

79bf69c HEAD@{10}: checkout: moving from feature-A to feature-A

79bf69c HEAD@{11}: checkout: moving from feature-A to feature-A

79bf69c HEAD@{12}: checkout: moving from master to feature-A

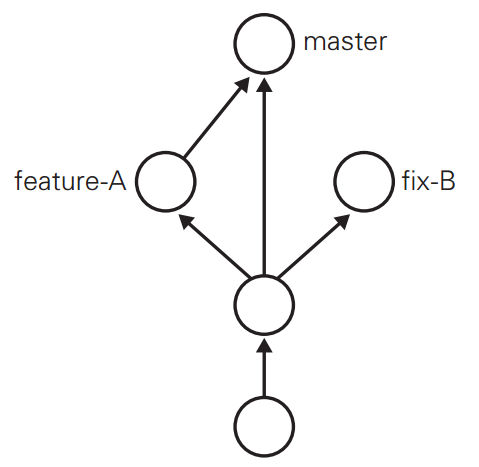
79bf69c HEAD@{13}: commit: Add index from HOLDMC

a6bde1e HEAD@{14}: commit (initial): First commit from HOLDMC

$ git reset --hard 4a7e097

HEAD is now at 4a7e097 Merge branch 'feature-A'

#### 恢复后的状态：



NOTE:

哈希值只要输入 4 位以上就可以执行。

#### 合并fix-B：

合并前一定要先确定下当前分支是否在主分支下：

$ git merge --no-ff fix-B

Auto-merging holdmc-git.md

CONFLICT (content): Merge conflict in holdmc-git.md

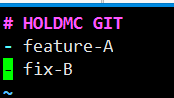
Automatic merge failed; fix conflicts and then commit the result.

出现冲突无法正常合并：

原因是fix-B和feature-A分支对同一个文件修改的位置一样造成



修改为

即可

再进行提交：

$ git add holdmc-git.md

$ git commit -m "fix confilct from HOLDMC"

[master c172754] fix confilct from HOLDMC

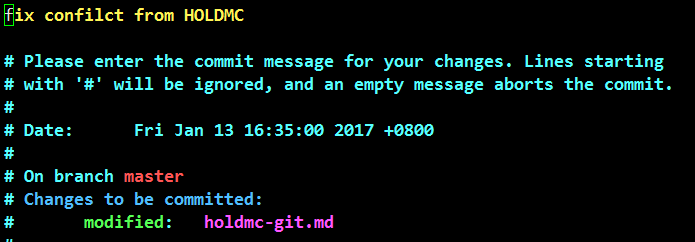
### git commit --amend——修改提交信息

将上一条提交信息记为了 " fix confilct from HOLDMC "，但它其实是解决合并时发生的冲突只是过程之一，这样标记实在不妥。

于是，我们要修改这条提交信息。

$ git commit --amend

出现：

修改为：



$ git log --graph

\* commit f8dfdf81c6173e38c3847519036b69dd6c88e295

|\ Merge: 4a7e097 b14b737

| | Author: chen qigan <chenqiganmail@foxmail.com>

| | Date: Fri Jan 13 16:35:00 2017 +0800

| |

| | Merge branch 'fix-B'

| |

| \* commit b14b737775379ee4e7f09b84ffa31d7e03fb6a8e

| | Author: chen qigan <chenqiganmail@foxmail.com>

| | Date: Fri Jan 13 16:20:04 2017 +0800

| |

| | Fix-B from HOLDMC

| |

\* | commit 4a7e0972c6f06da819529726df63b7800eca7d98

|\ \ Merge: 79bf69c aaaa7f6

| |/ Author: chen qigan <chenqiganmail@foxmail.com>

|/| Date: Fri Jan 13 15:43:42 2017 +0800

| |

| | Merge branch 'feature-A'

| |

| \* commit aaaa7f61e6466fe151ed8a4de0dcfe43b191ad6a

|/ Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:36:33 2017 +0800

|

| Add from feature-A branch by HOLDMC

|

\* commit 79bf69cc883d262986adb7b46be937fbb2bfbd43

| Author: chen qigan <chenqiganmail@foxmail.com>

| Date: Fri Jan 13 15:18:05 2017 +0800

|

| Add index from HOLDMC

|

\* commit a6bde1e1d78fdbea59821ff0622774356a1c7cf7

Author: chen qigan [chenqiganmail@foxmail.com](mailto:chenqiganmail@foxmail.com)

## 建立uboot 的git库

### 初始化git库：

$git init

Initialized empty Git repository in /home/holdmc/work/old\_dev\_env/u-boot-2009.08/.git/

$ git add \*

The following paths are ignored by one of your .gitignore files:

System.map

patches

u-boot

u-boot.bin

u-boot.lds

u-boot.map

u-boot.srec

Use -f if you really want to add them.

$git commit -m "raw source"

[master (root-commit) 94c17a5] raw source

6630 files changed, 1910786 insertions(+)

create mode 100644 CHANGELOG

create mode 100644 CHANGELOG-before-U-Boot-1.1.5

create mode 100644 COPYING

create mode 100644 CREDITS

。。。

（skip）