# Git cmd

github: yhong061+0714

==============================

git reflog //list git operation and commit id

git branch camera\_0912 b8192b1 //create brance with commit id b8192b1

==============================

https:

echo "# self" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/yhong061/self.git

git push -u origin master

==============================

ssh:

echo "# self" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:yhong061/self.git

git push -u origin master

==============================

git remote add origin https://github.com/yhong061/self.git

git push -u origin master

==============================

git remote add origin git@github.com:yhong061/self.git

git push -u origin master

==============================

==============================

git clone https://github.com/yhong061/sample.git

cd sample

vim git.txt

git add git.txt

git commit -sm "add git.txt"

cat .git/config

git push origin master

==============================

git remote -v //list remote name

==============================

git blame kernel/xc6130\_reg/readme.txt //list file's modify author per lines

==============================

git log --oneline //display log in one line

git log --graph //display log in a graph mode

git log -all //display all branchs log

==============================

git reflog //list HEAD pointer's trajectory

==============================

git add -p readme.txt //separate readme.txt's modification to serial parts

//s(separate) --> y(yes) --> n(no)

==============================

git rebase -i HEAD~2 //compress the last 2 commits to 1

==============================

git stash //stash current work env.

git stash list //list stash

git stash apply //restory stash, default is stash@{0}

git stash apply stash@{2} //restory stash's name stash@{2}

==============================

git fsck --lost-found //list the lost commit

==============================

//clone remote a branch not remote all branchs

git init

git remote add -t BRANCH\_NAME\_HERE -f origin REMOTE\_REPO\_URL\_PATH\_HERE

git checkout BRNACH\_NAME\_HERE

==============================

//save the changed filename to "update.zip" between serial commits

git archive -o ../updated.zip HEAD $(git diff --name-only HEAD^) //save the changed filename between HEAD and HEAD^

git archive -o ../updated.zip NEW\_COMMIT\_ID $(git diff --name-only OLD\_COMMIT\_ID NEW\_COMMIT\_ID) //save the changed file between NEW and OLD commit ID

# Git cmd

git blame arch/arm64/boot/dts/gt7\_ebox.dts

//查看文件的每一行的提交者信息

git blame -L 160,+10 arch/arm64/boot/dts/gt7\_ebox.dts

//查看文件的每一行的提交者信息，指定行

-------------------------------------------------------------

repo sync android/device/actions/common/frameworks/video

vim /home/yhong/work/GS700E\_5110\_AR\_160907/android/.repo/manifest.xml

-------------------------------------------------------------

patch -p1 < ../video2/0001-Add-monofreq-and-multifreq-PLL-Setting.patch

git status

git add .

git status

git am --continue -s

git commit -as --amend

传统的打patch方式

-------------------------------------------------------------

git whatchanged .

查看commitID涉及到的文件修改

git format-patch -1

生成commitID对应的patch, 1代表生成倒数第一个commitID的patch, 2代表生成倒数第一个和倒数第二个commitID的patch

git am –abort

放弃掉以前的am信息，这样才可以进行一次全新的am

git am ../video2/0001-Add-monofreq-and-multifreq-PLL-Setting.patch

可以一次合并一个文件，或者一个目录下所有的patch

-------------------------------------------------------------

git cherry-pick 8168fb32d64fff496f1b6db7063e8d1bd4a5147c

//将其他分支的指定commit-id分支同步到当前分支

-------------------------------------------------------------

593 git status

594 git commit -asm "add driver xc6130"

595 git status

596 git log

597 git branch

598 git checkout dev

599 git log

600 git show -p

601 rm test

602 git status

603 git rm test

604 git status

605 git commit -asm "revert to top tree "

606 git log

607 git pull

608 git branch

609 git checkout yhong

610 git log

------------------------------------------------

git push gl5209 yhong1:review/TB\_ipc\_ov2718

531 git branch

532 git status

533 git diff

534 git reset --hard

535 git status

536 git fetch gl5209

537 git branch -a | grep ov2718

538 git checkout -b ov2718remotes/gl5209/review/TB\_ipc\_ov2718

539 git checkout -b ov2718 remotes/gl5209/review/TB\_ipc\_ov2718

540 git branch

541 git branch -D ov2718remotes/gl5209/review/TB\_ipc\_ov2718

542 git branch

543 git log

------------------------------------------------

639 git branch

640 git status

641 git commit -as

642 git log

643 git branch

644 cat .git/config

645 git branch

646 git push gl5209 yhong:review/TB\_IPC\_yhong

647 git branch -a | grep yhong

648 git branch -a

649 git branch -a | grep yhong

------------------------------------------------

595 git checkout -b ipc remotes/gl5206/sh/GS700C\_5110\_IPC\_20160825

596 git branch

597 git checkout yhong

598 git push gl5206 yhong:review/TB\_ipc\_camera

599 git log

600 git checkout ipc

601 git cherry-pick f4d9b3104974382ff172288c669bfb81cd73ab23

602 git log

603 git pull

604 git push gl5206 ipc:sh/GS700C\_5110\_IPC\_20160825

=====================================================================

git fetch gl5209

git rebase

=====================================================================

HEAD: 表示当前版本

HEAD^: 表示上一个版本

HEAD^^: 表示上上一个版本

HEAD~100: 表示当前版本网上100个版本

=====================================================================

git add:

git add <path> //添加 created or modified 文件，不包括 deleted 文件

git add -u <path> //添加 modified or deleted 文件，不包括 created 文件

git add -A <path>

git add -i //查看文件状态

git rm path/file //删除通过add命令提交到暂存区的文件

=====================================================================

git branch:

git checkout branch //检出分支

git checkout -b branch //创建分支并检出分支

git branch -d dev //删除dev分支

关联远程分支：

git checkout -b dev; //创建本地分支dev

git pull origin dev //关联远程分支origin和本地分支dev，dev和origin将保持一致

git push origin dev:tmp //将本地分支dev的commit内容推送到远程仓库origin的tmp分支

git push origin :tmp //删除远程仓库origin的tmp分支

git fetch

git branch --set-upstream dev gl5206/sh/TB\_multistream

//--set-upstream: 跟踪远程分支

//本地分支名：dev

//远程分支名：remotes/gl5206/sh/TB\_multistream

//最终结果：在 .git/config 文件中增加关联信息

[branch "dev"]

remote = gl5206

merge = refs/heads/sh/TB\_multistream

=====================================================================

git checkout/reset:

git reset --hard commit-id //回退到指定版本号的地方

git reset HEAD file //取消暂存区的修改，并将暂存区的修改返回到工作区

git checkout -- filename //取消工作区的修改

git checkout branch -- filename //用branch所指向的提交中的filename替换暂存区和工作区的文件

git checkout . //用暂存区的所有文件覆盖本地文件

git checkout -- . //用暂存区的所有文件覆盖本地文件

git checkout //汇总当前目录及子目录的文件在工作区，暂存区，HEAD的差异

git checkout HEAD //汇总当前目录及子目录的文件在工作区，暂存区，HEAD的差异

=====================================================================

git diff:

git diff readme.txt //比较本地文件跟git仓库的不同

git diff HEAD -- readme.txt //比较工作区文件跟git仓库(本地分支)的不同

=====================================================================

git merge:

get merge dev //将dev分支合并到当前分支，假定当前处于master分支

get merge --no-ff -m "no-ff" dev //将dev分支合并到当前分支，并为dev分支创建commit-id

--no-ff表示不是用Fast forward方式合并分支

fast forward方式合并只是移动了HEAD指针，并没有做内容拷贝

可通过git log查看分支历史

=====================================================================

git init //创建空的版本库

git add readme.txt //将代码修改提交到暂存区

git commit -m "add readme.txt" //把暂存区的修改文件提交到git仓库

git status //查看git仓库修改状态

git diff readme.txt //比较本地文件跟git仓库的不同

git diff HEAD -- readme.txt //比较工作区文件跟git仓库的不同

git log //查看git仓库的修改记录

git log --pretty=oneline //查看git仓库的修改记录, 在一行显示

git reset --hard HEAD^ //获取上一个版本

HEAD: 表示当前版本

HEAD^: 表示上一个版本

HEAD^^: 表示上上一个版本

HEAD~100: 表示当前版本网上100个版本

git reset --hard commit-id //回退到指定版本号的地方

git reset HEAD file //取消暂存区的修改，并将暂存区的修改返回到工作区

git checkout -- readme.txt //取消工作区的修改

rm test.txt;git rm test.txt;git commit -m "remove test.txt"

//删除工作区文件，删除暂存区文件，删除git仓库文件

rm test.txt;git rm test.txt;git checkout -- test.txt

//文件误删：删除工作区文件，删除暂存区文件，恢复删除的文件

//创建远程库

登陆GitHub网站

点击"Create a new repo",创建远程库

"Repository name"中输入"learngit"

git remote add origin git@github.com:michaelliao/learngit.git //关联本地git仓库到远程

origin: 远程库名

michaelliao: GitHub的账户名,自己创建

git push -u origin master //将本地库内容推送到远程仓库, 第一次推送

git push origin master //将本地库内容推送到远程仓库

git clone git@github.com:michaelliao/gitskills.git //克隆远程库，gitskill为远程库名

git brance dev //创建dev分支

git checkout -b dev //创建dev分支，并切换到dev分支

git branch //查看所有分支，当前分支前用"\*"表示

git checkout dev //切换到dev分支， 即将HEAD指针指向dev

git checkout master //切换到主分支， 即将HEAD指针指向master

get merge dev //将dev分支合并到当前分支，假定当前处于master分支

get merge --no-ff -m "no-ff" dev //将dev分支合并到当前分支，并为dev分支创建commit-id

--no-ff表示不是用Fast forward方式合并分支

fast forward方式合并只是移动了HEAD指针，并没有做内容拷贝

可通过git log查看分支历史

git branch -d dev //删除dev分支

//解决冲突

//修改feature1分支上的readme.txt文件

git checkout -b feature1

修改readme.txt文件

git add readme.txt

git commint -m "AND simple"

//修改master分支上的readme.txt文件

git checkout master

修改readme.txt文件

git add readme.txt

git commint -m "& simple"

git merge feature1 //产生冲突

git status //查看冲突的文件

直接查看工作区中的readme.txt，并修改冲突

//文件中<<<<<<< HEAD, =========, >>>>>>> feature1标记了冲突的内容，直接修改即可

git add readme.txt //提交修改冲突后的文件

git log --graph --pretty=oneline --abbrev-commit //查看分支的合并情况

git brance -d feature1 //删除feature1分支

//分支储藏: 临时插进来的任务,如bug的修改

git stash //将当前分支修改内容储藏起来

git checkout master //切换到master分支

git checkout -b issue //创建新的分支issue

修改issue分支工作区的readme.txt //修改bug

git add readme.txt //提交修改内容

git commit -m "fix bug 101" //提交修改内容

git checkout master //切换到master分支

git merge --no-ff -m "merged bug fix 101" issue //合并issue分支到主分支

git brance -d issue //删除issue分支

git checkout dev //切换到dev分支

git status //显示dev分支的修改记录： 无

git stash list //查看分支的储藏情况

git stash apply //恢复stash的内容，但stash内容不删除

git stash apply stash@{0} //恢复指定的stash的内容，针对多次stash

git stash pop //恢复stash的内容，stash内容删除

git stash drop //删除stash的内容

//远程库冲突

//远程库名: origin

git remote //查看远程库的信息

git remote -v //查看远程库的详细信息

git push origin master //将master分支上所有本地信息提交到远程库

git push origin dev //将dev分支上所有本地信息提交到远程库

//第一位修改者

git clone git@github.com:michaelliao/learngit.git //克隆远程库learngit

git branch //本地库将生成master分支

git checkout -b dev origin/dev //创建远程origin的dev分支到本地

修改dev分支文件： hello.py

git add hello.py

git commit -m "add coding: /usr/bin/env"

git push origin dev

//第二位修改者

git clone git@github.com:michaelliao/learngit.git //克隆远程库learngit

git branch //本地库将生成master分支

git checkout -b dev origin/dev //创建远程origin的dev分支到本地

修改dev分支文件: hello.py

git add hello.py

git commit -m "add coding: utf-8 "

git push origin dev //与第一位修改者产生冲突

//修改远程冲突

git brance --set-upstream dev origin/dev //建立本地dev与远程origin/dev的链接

//--set-upstream解决"git push"报错：no tracking information

git pull //将最新的提交从origin/dev上抓取下来

本地修改冲突的文件

git add hello.py

git commit -m "merge & fix hello.py"

git push origin dev

git tag V1.0 //给当前分支打上标签

git tag V1.0 commit-id //在commit-id处打上标签

git tag //查看当前分支标签

git show V1.0 //查看标签信息

git tag -a v0.1 -m "version 0.1 released" commit-id //在commit-id处打上标签，并添加信息

git tag -d V1.0 //删除V1.0的标签

git push origin v1.0 //给远程库origin打标签

git push origin --tags //将本地所有的标签推送给远程库

git push origin :refs/tags/v1.0 //删除远程库上的标签

# END