PART 01

关于Git

初识Git

安装完 Git 应该做的第一件事就是设置用户名称与邮件地址。每一个 Git 的提交都会使用这些信息，并且它会写入到每一次提交中，不可更改

```
$ git config --global user.name "名字"  //设置用户名
$ git config --global user.email "邮箱"   //设置用户邮箱
$ git config --list    //查看用户信息
```

常用命令

- git init
- git config --global user.name XXX
- git config --global user.email XXX
- git config --list
- git add
- git commit –m “XXXX”
- git status
- git push origin 本地分支名:远程分支名
- git pull
- git fetch

本地库与远程库建立连接的方式

1.连接远程库

$ git init 初始化一个本地库

本地库连接远程库

$ git remote add origin https://github.com/lixuehe/aaa.git

2.克隆远程库

$ git clone https://github.com/lixuehe/aaa.git

分支管理

小问题：大家来猜一猜gitbranch会出现什么结果 .

分支管理

因为git的分支必须指向一个commit，没有任何commit就没有任何分支。

提交第一个commit后git自动创建master分支”。

分支管理

创建分支(1.本地 git branch 分支名 2.远程分支1.可以在github创建 2.可以在本地分支推到远程仓库)

删除分支（1.本地 git branch -d（D） 分支名 2.远程 git push origin --delete 分支名）

查看分支（git branch -a ）

切换分支（git checkout 分支名 ）

合并分支（git merge 分支名）

分支管理

比较fetch加merge 和 pull 。

结果上来看 确实 fetch 加merge 等于 pull

但是 fetch是让你本地所关联的远程端的commit id 版本号更新到了最新，
然后在本地进行合并远程。

git pull 会将本地库更新至远程库的最新状态

https://www.cnblogs.com/sunshisonghit/p/6806255.html

冲突出现（本地）

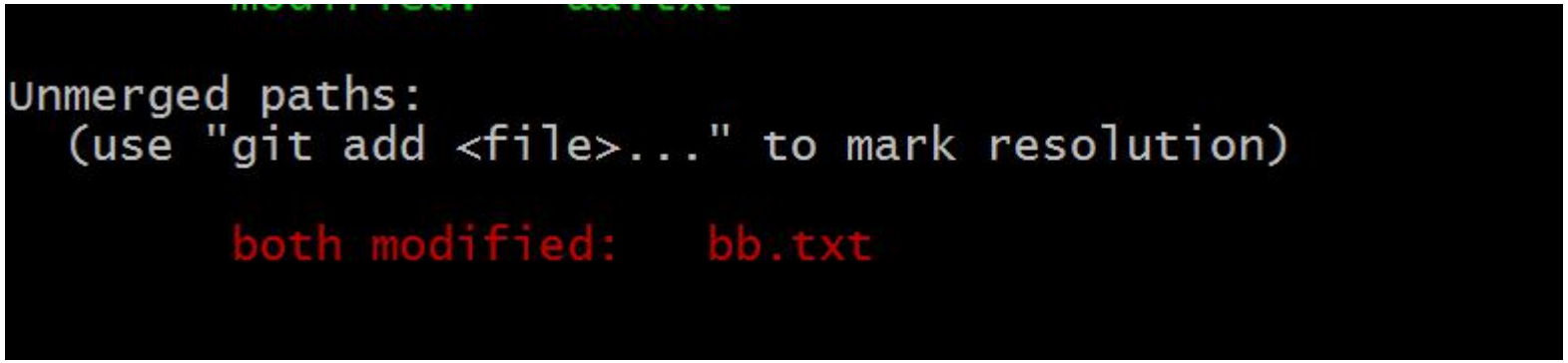
git checkout dev
git vim bb.txt

修改一些内容并保存

git checkout devliu
git vim bb.txt

在devliu的分支上修改一些内容并保存

git checkout dev
git merge devliu



冲突出现（本地）

1.git diff
2.git status

```
lixuehe@lixuehe-ThinkPad-T420:~/liuchuan$ git diff
diff --cc bb.txt
index 4253234,96b48bd..0000000
--- a/bb.txt
+++ b/bb.txt
@@@ -1,1 -1,1 +1,5 @@@
++<<<<<<<< HEAD
 +不好,世界!
++========
+ 你好,世界!
++>>>>>>>>> dev_test
lixuehe@lixuehe-ThinkPad-T420:~/liuchuan$ git status
位于分支 dev
您有尚未合并的路径。
  (解决冲突并运行 "git commit")
  (使用 "git merge --abort" 终止合并)

未合并的路径:
  (使用 "git add <文件>..." 标记解决方案)

        双方修改:    bb.txt

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
```

冲突出现（本地）

在bb.txt 中改内容，改成想要的内容

你好，世界！

git commit -am”修改了bb.txt”

冲突出现（本地）

我们可以直接查看readme.txt的内容：

```
Git is a distributed version control system.
Git is free software distributed under the GPL.
Git has a mutable index called stage.
Git tracks changes of files.
<<<<<<< HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> feature1
```

Git用`<<<<<<<`，`=======`，`>>>>>>>`标记出不同分支的内容，我们修改如下后保存：

```
Creating a new branch is quick and simple.
```

线上冲突

模拟本地冲突的原因就是为了给模拟远程冲突做铺垫
　那么我们来模拟一下远程冲突

　组员A和组员B同时对一个文件进行修改，然后分别提交到远程，组长在第二天合并代码的时候就会产生冲突

合并无关的历史

拒绝合并无关的历史 添加–allow-unrelated-histories
没有共同的祖先历史，比如，
我初始化仓库并连接线上仓库，创建了一个文件，进行add和commit，也就是说让仓库知道了我文件的存在，那么我去拉线上我分支的东西会出现这个拒绝.就是因为，线上的仓库有记录日志，而线下的这个新仓库没有线上文件的记录，没有共同的历史祖先.

git init
touch hhh.js
git add -A
git commit -m “某某某”
git pull origin lixuehe
==> *fatal: 拒绝合并无关的历史*
git pull origin lixuehe --allow-unrelated-histories

版本回退

git reset --hard *版本号*
git reset --hard HEAD^

版本回退

查看版本号 git log 查看commit提交历史

git reflog 可以查看所有分支的所有操作记录

gitlog gitreflog有什么不同

小技巧

解决每次都需要输入密码的问题

git bash进入你的项目目录，输入git config --global credential.helper store

在你的本地记录下了账号和密码

工作流

1.组员的工作:

组员保持每天本地dev和远程dev保持同步，本地自己的分支和自己dev分支保持同步，然后在自己分支进行工作，最后提交到自己远程分支上.

2.组长的工作:

组长定期将dev分支合并到master分支；每天将远程组员分支的代码检查无误后合并到本地dev分支上，然后将dev分支更新。

PART 02

算 法 示 例

算法示例

敲黑板，先 留 作 业 ！

a = 6　b = 5
a + b = ?

怎么求？

将运算过程发送至超姐邮箱　329611148@qq.com

算法示例

要求：不使用运算符 + 和 - ,计算两整数 a , b 之和。

案例：

输入： a = 5, b = 4

算法示例

java代码实现：

```
public int getSum(int a,int b){
        return b==0 ? a:getSum(a^b,(a&b)<<1);
}
```

```
public int getSum(int a, int b){
        while (b != 0){
        int sum=a^b;
        int carry=(a&b)<<1;
        a = sum;
        b = carry;
        };
        return a;
};
```

算法示例

异或运算：
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

`相同为0，不同为1`

与运算：
0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

两位同时为1，结果为1，否则为0
`同真为真`

在位运算操作中

`异或` 的一个重要特性是 `无进位加法`
`与运算` 将获得进位

算法示例

异或运算

a = 5 = 0101
b = 4 = 0100

a ^ b 如下：

0 1 0 1
0 1 0 0
---
0 0 0 1

与运算

a = 5 = 0101
b = 4 = 0100

a & b 如下：

0 1 0 1
0 1 0 0
---
0 1 0 0

然后使用左移运算符进位，得到：
1 0 0 0

算法示例

异或运算

a = 5 = 0101
b = 4 = 0100

a ^ b 如下：

0 0 0 1

与运算

a & b 如下：

然后使用左移运算符进位，得到：

1 0 0 0

再对两个数字进行`异或运算`、`与运算`

1 0 0 1

0 0 0 0

算法示例

1. a + b 的问题拆分为（a和b的无进位结果）+(a和b的进位结果)

2. 无进位加法使用异或运算计算得出

3. 进位结果使用与运算和移位运算计算得出

4. 循环此过程，直到进位为0

算法示例

敲黑板，留作业！

a = 6　b = 5
a + b = ?

怎么求？

将运算过程发送至超姐邮箱　329611148@qq.com

心得

引用史铁生说过的一句话：**过程是目的**。

在和伙伴琢磨Git工作流的过程是最重要的，过程中会产生分歧，能学到东西。

最后，希望大家在之后的小组宣讲中越来越好！

End