# 1 概述

## 1.1 版本控制系统工具的选择

采用 Git 开源的版本控制系统。

## 1.2 Git 特点简述

Git 是用于 Linux 内核开发的版本控制工具。与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持，使源代码的发布和交流极其方便。 Git 的速度很快，这对于诸如 Linux kernel 这样的大项目来说自然很重要。 Git 最为出色的是它的合并跟踪（merge tracing）能力。

实际上内核开发团队决定开始开发和使用 Git 来作为内核开发的版本控制系统的时候，世界开源社群的反对声音不少，最大的理由是 Git 太艰涩难懂，从 Git 的内部工作机制来说，的确是这样。但是随着开发的深入，Git 的正常使用都由一些友好的脚本命令来执行，使 Git 变得非常好用，即使是用来管理我们自己的开发项目，Git 都是一个友好，有力的工具。现在，越来越多的著名项目采用 Git 来管理项目开发.

作为开源自由原教旨主义项目，Git 没有对版本库的浏览和修改做任何的权限限制。

# 2 版本控制流程

## 2.1 目标

- 保证各个环境(开发、测试、主干)的独立，避免相互影响。
- 减少最终发布时合并主干出现冲突的概率。
- 降低冲突处理的难度。

## 2.2 原则

多个版本（开发版本，测试版本，发布版本）；多次合并。

## 2.3 流程

- 项目开发编码前从当前主干建立一条开发分支，供项目开发人员使用；
- 开发结束，提交测试的时候，从当前主干建立一条测试分支，将开发分支合并到测试分支上，供测试人员进行测试。这样开发人员对开发分支的修改不会影响测试环境；
- 定时将开发分支的修改合并到测试环境中。
- 回归测试的时候，从当前主干建立一条发布分支，将测试分支合并到该发布分支上，在发布分支上进行回归测试。
- 发布前，将发布分支合并到当前主干。

## 2.4 好处

- 多个版本相互独立，互不影响
- 通过多次与主干的合并，这样发布时候和主干做最后一次合并的冲突会大大减少，并且在与主干多次合并过程中的冲突解决都在测试阶段中得到了测试。

建议：如果项目的周期比较长，和主干进行合并的次数也应该加大，以降低处理冲突的难度。

# 3 Git 版本控制

## 3.1 版本控制目录设置

**每个项目一个主目录，不同用户可在自己的目录下创建项目克隆分支，开发完成后提交合并。**

## 3.2 版本提交流程

（注：PM：项目经理 Project Manager；SE：软件工程师 Software Engineer）

- 版本提交是指：SE 把程序代码、配置脚本、数据库表定义脚本、数据库表基础数据等，提交给 PM；
- 提交流程：
  - PM 确认 building 目录已经备份；
  - SE 克隆 building 目录到自己的工作目录
  - SE 提交代码、配置、数据库等；
  - PM 测试 SE 的 Building；如果 building 有问题重复上一步；
  - Building 正确后，PE 提交确认

# 4 Git 使用简介：

## 4.1 Ubuntu 安装 Git

```
wget http://www.kernel.org/pub/software/scm/git/git-1.7.4.tar.gz
tar -zxvf git-1.7.4.tar.gz
cd git-1.7.4
apt-get install git-core
apt-get install gitk
```

注：PM 用户名：zhouch； SE 用户名：se1

## 4.2 PM 建立 Git 库：

用 pm 用户 SSH 登录 Git_server，服务器地址：192.168.8.40 或 git.rfidcer.org

建立项目目录，如 Hello（VC++演示程序），将项目内容复制到该目录下；

初始化 Git 库：

```
zhouch@GitServer:~$ cd Hello
zhouch@GitServer:~/Hello$ git init
Initialized empty Git repository in /home/zhouch/Hellols /.git/
```

git init 命令用于初始化当前所在目录的这个项目，shell 返回的提示表明已经建立了一个.git 隐藏目录来保存这个项目目前的进展信息。我们可以用 ls -a 看到它。

```
zhouch@GitServer:~/FN_HFD1008Q$ git add .
zhouch@GitServer:~/FN_HFD1008Q$ git commit
```

git add . 这个命令要求 git 给我目前的这个项目制作一个快照 snapshot（快照只是登记留名，快照不等于记录在案，git 管快照叫做索引 index)。快照一般会暂时存储在一个临时存储区域中。

git commit 用于将快照里登记的内容永久写入 git 仓库中，也就是开发者已经想好了要提交自己的开发成果了。

在输入git commit并按回车时会转到一个vi窗口，要求开发者输入这次提交的版本和开发信息。意思就是说这个项目目前的版本是多少，已经完成了哪些功能，还有哪些功能是需要以后完成的等等信息。
至此，一个新项目就诞生了，第一个开发信息（开发日志）也随之诞生。

可输入 git log 查看git 版本日志信息。

## 4.3 PM 提交新版本:

开启一个试验分支(experimental)，如果分支开发成功则合并到主分支（master），否则放弃该试验分支。

$ git branch experimental　　//创建一个试验分支，名称叫 experimental

$ git branch　　//显示当前都有哪些分支，其中标注*为当前所在分支

$ git checkout experimental　//转移到 experimental 分支

通过 SSH 将新版本文件复制到 Hello 目录下；

想检查到目前为止对源码都做了哪些修改（相对于本次工作刚开始之时）：

$ git diff　　//这个命令只在 git add 之前使用有效。如果已经 add 了，那么此命令输出为空

$ git diff –cached　//这个命令在 git add 之后在 git commit 之前有效。

$ git status　　//这个命令在 git commit 之前有效，表示都有哪些文件发生了改动

如果分支开发成功：

$ git commit -a　　//在 experimental 分支改进完代码之后用 commit 在此分支中进行提交

这是一个偷懒的命令，相当于 git add .; git commit;

但是，此处有一点应该注意，那就是 git commit -a 无法把新增文件或文件夹加入进来，所以，如果你新增了文件或文件夹，那么就要老老实实的先 git add .，再 git commit 喽。

$ git checkout master　　//转移回 master 分支

$ git merge experimental　　//经证实分支开发成功，将 exerimental 分支合并到主分支

$ git commit -a　　//彻底完成此次分支合并，即提交 master 分支

$ git branch -d experimental　//因为 experimental 分支已提交，所以可安全删除此分支

如果分支开发失败：

$ git checkout master

$ git branch -D experimental　//由于分支被证明失败，因此使用-D 来放弃并删除该分支

## 4.4　SE 提交新版本:

$git clone /home/zhouch/Hello　Hello1　　//此命令用于克隆 PM 的工作到 se1 的 Hello1 目录下。请注意，此命令有可能会因为/home/zhouch/Hello 的目录权限问题而被拒绝，解决方法是 chmod o+rx /home/zhouch/Hello。

$git commit -a　　//提交自己的改进成果到自己的 git 仓库中，并口头告知 PM 他已经完成了工作。

如果 PM 非常非常信任 se1 的开发能力：

$ cd /home/zhouch/Hello

$ git pull /home/se1/Hello1　//pull 命令的意思是从远端 git 仓库中取出(git-fetch)修改的代码，然后合并(git-merge)到我（Hello）的项目中去。读者要记住一个小技巧，那就是“git pull .”命令，它和 git merge 的功能是一样的，以后完全可以用 git pull .来代替 git merge 哦！请注意，git-pull 命令有可能会因为/home/se1 的目录权限问题而被拒绝，解决方法是 chmod o+rx /home/se1。

如果 PM 不是很信任 se1 的开发能力：

$ cd /home/zhouch/Hello

$ git fetch /home/se1/Hello1 master:se1works　　//此命令意思是提取出 se1 修改的代码内容，然后放到 PM 工作目录下的 se1works 分支中。之所以要放到分支中，而不是 master 中，就是要我先仔仔细细看看 se1 的开发成果，如果我觉得满意，我再 merge 到 master 中，如果不满意，我完全可以直接 git branch -D 掉。

$git whatchanged -p master..se1works　　//用来查看 se1 都做了什么

$git checkout master　　//切换到 master 分区

$git pull . se1works　　//如果我检查了 se1 的工作后很满意，就可以用 pull 来将 se1works 分支合并到我的项目中了

$git branch -D se1works　　//如果我检查了 se1 的工作后很不满意，就可以用-D 来放弃这个分支就可以了

过了几天，se1 如果想继续帮助我开发，他需要先同步一下我这几天的工作成果，只要在其当初 clone 的 myrepo 目录下执行 git pull 即可：

#git pull　　//不用加任何参数，因为当初 clone 的时候，git 已经记住了我（rocrocket）的工作目录，它会直接找到我的目录来取。

# 5 其他 Git 常用命令：

介绍提交用户：（未介绍的话为系统登录用户）

git config --global user.name “Your Name”

git config --global user.email “you@example.com”

删除文件：　git rm main.c

查询开发日志的方法就是 git log

详细开发日志内容：　git log -p

库的逆转与恢复　git reset

命令形式：git reset [--mixed | --soft | --hard] [<commit-ish>]

命令的选项：**(注：命令后必须有 commit 4-6 位 ID 号 以区分需要恢复到哪个版本）**

--mixed

撤销 commit 和 index file，保留 working tree

--soft

只撤销 commit，保留 working tree 和 index file

--hard

撤销 commit、index file 和 working tree，即撤销销毁最近一次的 commit

**参考资料：**

**《看日记学 git》**