**代码管理及规范**

**2021-04-06 chenglongqiao**

## 背景

所谓版本控制系统（Version Control System）， 从狭义上来说，它是软件项目开发过程中用于储存我们所写的代码所有修订版本的软件，但事实上我们可以将任何对项目有帮助的文档交付版本控制系统进行管理。

如果在开发团队中没有使用版本控制， 多个开发人员在开发过程中将会引入很多问题， 如软件代码的一致性、 软件内容的元余 、 软件过程的事物性、 软件开发过程中的并发性、 软件源代码的安全性， 以及软件的整合等问题。

版本控制的目的是实现开发团队并行开发、 提离开发效率的基础。其目的在于对软件开发进程中文件的发展过程提供有效的追踪手段， 保证在需要时可回到可用的版本， 避免文件的丢失、 修改的丢失和功能的覆盖。

版本控制的功能在于跟踪记录整个软件的开发过程， 包括软件本身和相关文档，以便对不同阶段的软件及相关文档进行表示并进行差别分析， 对软件代码进行可撤销的修改， 便于汇总不同开发人员所做的修改， 辅助协调和管理软件开发团队。

## 使用范围

本规范文档适用于软件研发部门 、 项目成员等人员。

## git规范及要求

### 账号

开发人员遵循一人一 号原则， 不得向他人透漏， 严禁使用他人账号进行各项操作。

### 提交规则

提交代码应严格遵守 “ 先更新再提交” 原则。

当完成了一个小功能，能够编译并且通过自己测试之后，谨慎地提交。

如果在修改的期间别人也更改了仓库工程 的对应文件，那么 Commit 就可能会失败。 如果别人和自己更改的是同一个文件，那么 Update 时会自动进行合并，如果修改的是向 一行，那么合并时会产生冲突，这种情况就需要同之前的开发人员联系 ， 两个人一起协商解决冲突，解决冲突之后，需要两人一起测试保证解决冲突之后，程序不会影响其他功能。

在更新时注意所更新文件的列表，如果提交过程中产生了更新，则也是需要重新编译并且完成自己的一些必要测试，再进行提交。

不能提交与代码无关的文件及影响他人测试、 运行的代码。 例如Eclipse 中的. classpath 文件，Windows 生成的缩略图 Thumbs.db,目编译生成的临时文件 obj, . class， 测试功能的临时本机配置文件等。提交了这样的文件后，别人在更新后就可能与本地的环境冲突从而影响大家的工作。

不能提交自己不明白的代码， 你的代码将被项目成员所分享。 如果提交了你不明白的代码，你看不懂，别人也看不懂，如果在以后出现了问题将会成为项目质量的隐患。因此在自己写代码或引入任何第三方代码之前，确保你对这个代码有一个很清晰的了解。

不能别人未知情的情况下，强行替换、 删除别人的代码。别人写的代码也是劳动成果，而且有可能与其它代码关联 、 交互，强行替换 、 删除会造成系统运行的稳定，同时造成对方迷惑。处于尊重、 个人修养方面， 也不应该未知情下这样操作。

### 提交细则

见《git提交规范细则》

## 编码规范

见《C++语言编程规范》

## 开发文档

* 要求在提代码交后的一天完成文档提交，要用公司模板表示（使用说明）
* 开发记录，形式暂时不限制，内容也不限制，记录下开发中的一些细节，哪些坑
* 存放在ECM中
* 使用文档，尽可能详细，可持续完善，做到给出去后不用再来问，有图有文
* 测试要求及coding指派

## 开发环境及辅助工具

1. ubuntu 18.04 开发环境（虚拟机环境）

2. qtcreator 上层程序开发IDE 编译器Clang 6.0（虚拟机环境已配置）

3. keil mdk5 单片机IDE v5.06

4. smartgit 仓库管理