repo项目管理与分支演进

1. patch文件备份

首次创建仓库

注意修改yourname为你的名字，有两处。

```
$ git clone ssh://yourname@10.20.40.19:29418/freemeos/mt6750/ALPS-MP-N0.MP7-V1_DROI6755_66_N/patch && scp -p -P 29418 yourname@10.20.40.19:hooks/commit-msg patch/.git/hooks/
```

提交

示例如下。

```
$ cd your-patch-git-dir
$ git pull
$ cp the-patch-zip-files-from-mtk ./
$ git add the-patch-zip-files-from-mtk
$ git commit -m "[patch/backup] ALPS-MP-N0.MP7-V1_DROI6755_66_N: modem p3-p5, ALPS p3"
$ git push origin HEAD:refs/for/master
```

2. mtk版本合并patch

为了减少干扰，建议使用完全干净的源码树执行patch合并操作。

拉取代码

首次合并patch，请拉取代码树，示例代码如下

```
$ mkdir alsp_50n_mtk
$ cd alsp_50n_mtk
$ repo init --no-repo-verify -u ssh://yourname@10.20.40.19:29418/freemeos/manifest -m ALPS-MP-N0.MP7-V1_DROI6755_66_N/mtk.xml
$ repo sync
$ repo start --all mtk
```

如果已经有这样的一棵树，请更新这棵树，保证代码最新。

```
$ repo sync
```

合并patch

创建新的本地分支patch用于执行本次的patch合并

```
$ repo start --all patch
```

将patch解压到源码目录，确保该步骤执行正确。然后查看当前的改动，添加-o选项，如果有未被任何git仓库跟踪的文件也列出来（例如.repo目录目录下新增文件）

```
$ repo status -o
```

```
$ repo forall -pc git status --ignored | awk '/^project/{a=1; project=$2
}a==1&&$0~/git/{projects=projects project " ";a=0}END{print projects}'
test/ test2/
```

注意该命令输出将会在接下来的在pcb分支合并patch时使用。

添加代码，并本地提交

```
$ repo forall <projects1 projects2 ...> -pc 'git add . && git commit -m "[patch]
you-should-say-something-here"'
```

执行完毕后使用再次确认是否遗漏文件。

```
$ repo forall git status --ignored
```

如果有，那么手动进入该目录下，执行git add -f .，git commit --amend 后直接保存退出vim窗口。

说明：git add配合-f参数，可以强制添加所有文件（即使是在.gitignore被忽略的文件）。

确认完毕后，上传代码到服务器

```
repo upload
```

gerrit上合并提交

打开gerrit网页，http://10.20.40.19:8080，登陆，确认代码提交成功，将代码review并submit到仓库中。

本地分支清理

删除本地分支

```
$ repo abandon patch
```

3. 将patch合并到pcb分支

拉取pcb分支代码

```
$ mkdir alsp_50n_pcb_oversea
$ cd alsp_50n_pcb_oversea
$ repo init --no-repo-verify -u ssh://yourname@10.20.40.19:29418/freemeos/manifest -
m ALPS-MP-N0.MP7-V1_DROI6755_66_N/pcb_oversea.xml
$ repo sync
$ repo start --all pcb_oversea
```

从mtk合并提交

更新仓库，确保获得仓库最新代码。

```
$ repo sync
```

创建新的本地分支patch用于执行本次的patch合并

```
$ repo start --all patch
```

从mtk分支合并patch，其中<projects1 projects2 ...>参数，请使用前面生成的project list替换，注意该命令需要在项目根目录下执行。

```
$ repo forall <projects1 projects2 ...> -pc git cherry-pick origin/mtk
```

如果有仓库冲突，进入仓库，执行git mergetool执行三方合并，然后手动git add，git commit，然后重新执行上述命令，注意project list中去掉已经执行完毕的project。

如果patch分支已经全部合并了origin/mtk分支提交。接下来对project list使用git commit --amend -m重写提交信息。由于git cherry-pick会将某提交完整的"pick"过来，包括其commit message，其中被使用的change-id，使用git commit --amend -m重写提交信息，确保每个提交都生成新的的change-id。

```
$ repo forall <projects1 projects2 ...> -pc git commit --amend -m "[patch] you-
should-say-something-here"
```

上传代码

```
$ repo upload
```

本地分支清理

删除本地分支

```
$ repo abandon patch
```