REPO 설치

다운로드

```
$ mkdir ~/bin
$ curl https://android.git.kernel.org/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

~/.bashrc 파일의 끝에 다음과 같이 추가하여 ~/bin 폴더를 PATH에 추가

```
PATH=~/bin:$PATH
```

안드로이드 소스 받기

안드로이드 소스를 받을 디렉터리 생성, 여기서는 aosp-gingerbread. git clone 과 달리 디렉터리를 만들어 주지 않음을 주의한다.

```
$ mkdir aosp-gingerbread
$ cd aosp-gingerbread
```

초기화

-u 옵션으로 manifest git 저장소의 주소를 -b 옵션으로 특정 브랜치를 지정한다.

```
$ repo init -u git://android.git.kernel.org/platform/manifest.git -b gingerbread
```

init이 끝나면, 실제로는 .repo 디렉터리가 다음과 같은 형태로 생성된다.

```
$ tree -L 1 .repo
.repo
├── manifest.xml -> manifests/default.xml
├── manifests
└── manifests.git
```

```
├── project.list
├── projects
└── repo
```

-u 인자의 주소인 manifest의 git 저장소의 -b 인자의 브랜치가 .repo/manifests로 저장되며 그 중 default.xml 파일을 .repo/default.xml 로 심볼릭 링크한다.

.repo/default.xml에 안드로이드를 구성하는 프로젝트들 (약 180개)의 각각의 remote 주소와 브랜치(또는 태그) 이름 clone 될 경로가 적혀 있다. 예;

```
<project path="bionic" name="platform/bionic" />
<project path="cts" name="platform/cts" />
```

AOSP의 경우 대부분 같은 remote와 브랜치를 사용하므로 파일의 위쪽에 다음과 같이 기본 값으로 설정되어 있다.

```
<remote  name="korg"
         fetch="git://android.git.kernel.org/"
         review="review.source.android.com" />
<default revision="gingerbread"
         remote="korg" />
```

그 결과 위의 bionic 프로젝트는 다음 명령어를 실행한 것과 같아진다.

```
git clone git://android.git.kernel.org/platform/bionic -b gingerbread ./bionic
```

동기화

```
$ repo sync
```

모든 프로젝트들을 clone 하기 때문에 엄청 오래 걸림. 실행하지 말고 다음 장의 TIP을 확인하라!

팁!

AOSP 미러에서 받기

인시그널의 git 서버에서 미러하는 AOSP를 이용. repo init 후에, .repo/manifests.xml파일을 다음과 같이 수정한다.

```
   <remote  name="korg"
            fetch="git://android.git.kernel.org/"
            review="review.source.android.com" />
+  <remote  name="insignal"
+           fetch="gitosis@210.219.52.136:"
+           review="review.source.android.com" />
   <default revision="gingerbread"
-           remote="korg" />
+           remote="insignal" />
```

git://android.git.kernel 대신 미러 주소인 git@git.insignal.co.kr:mirror/aosp/를 사용하게 바꾸었다. 저장하고 sync를 수행한다.

y

```
$ repo sync -j3
```

매 프로젝트마다 ssh 암호를 묻는 경우 다음 명령어로 개인키를 잠시 메모리에 올려두면 편하다.

```
$ ssh-add
```

미러된 저장소들은 읽기 권한만 있음. 참고용으로 꼭 한 벌 받아둔다.

이미 받은 repo를 활용

작업용 안드로이드를 받을 때 다음 명령어 처럼 --reference 옵션으로 기존에 받은 repo를 지정하면 중복된 커밋들은 다시 받지 않고 로컬 카피하므로 훨씬 빠르다.

```
$ repo init -u git@git.insignal.co.kr:sen-a107/gingerbread-manifest \
--reference /home/suapapa/workspace/mirrors/aosp/
```

> 주의! reference 삼은 repo는 지우면 안된다.

자주 사용하는 명령어들

git과 동일하게 모든 명령어는 help로 도움말을 볼 수 있다. 예;

```
$ repo help list
```

대부분 여러 프로젝트에서 수행해야하는 git 명령어를 한번에 실행하기 위해 만들어진 명령어이다.

sync

repo에 속한 모든 프로젝트를 갱신한다. 각 프로젝트는 독립적이므로 -j 옵션을 사용해 한번에 다운 받는 개수를 늘릴 수 있다.

```
$ reop sync -j3
```

list

repo에 속한 모든 프로젝트를 나열한다. 프로젝트 이름과 경로가 출력된다.

```
$ repo list
```

start

repo sync가 끝난 후의 프로젝트들은 detatch 된 상태. 아무런 브랜치에도 속해 있지 않다.

```
$ repo start master --all
```

모든 git 프로젝트들을 로컬 master 브랜치로 -없으면 생성해서- 이동. 따라서 브랜치간 이동 역시 start 명령을 사용

branch

repo 로 생성한 브랜치 목록을 보여준다.

```
$ repo branch
*  master                   | in all projects
```

abandon

start로 생성한 브랜치를 삭제한다. 생성(start)시 --all로 모든 프로젝트들에 생성한 브랜치도 한 번에 삭제 됨.

```
$ repo abandon test
```

forall

repo에서 지원하지 않는 기능을 수행할 수 있다. 예로 모든 git 저장소의 user.email을 수정하려면;

```
$ repo forall -c "git config user.email suapapa@insignal.co.kr"
```

references

- http://source.android.com/source/version-control.html