CVS 이야기

장우현

louis (at) mizi.co.kr

도운이

박용주

문서 형식을 **LinuxDoc**에서 **DocBook**으로 변환

yongjoo (at) kldp.org

$Date: 2001/08/21 13:01:41 $

여러명이 어떤 공동의 작업을 수행할 때 유용하게 쓸 수 있는 CVS에 관한 이야기를 하고 있습니다. 이 글에서는 아직 CVS 관리에 관한 얘기는 없으며, 순수하게 사용하는 방법만 설명하고 있습니다. 글에 대한 문의사항이 있으면 언제라도 저에게 메일주시기 바랍니다. (참고로 앞으로는 어떻게 될지 모르겠지만, 현재 이글에서는 자세한 내용을 다루고 있지 않습니다. 좀더 자세한 내용을 원하시는 분들은 CVS메뉴얼을 참고하시기 바랍니다. 대부분 CVS를 사용하고픈 사람들은 프로젝트의 소스를 받아오기 위해서, 간단하게나마 참여하고 싶어서 입니다. 그런 분들에게는 이 정도의 글으로도 충분하리라 고 봅니다. 혹시 실제로 쓰다가 보면 유용한 내용인데 이 글에서 빠져있다면 언제라도 메일을 보내주세요.)

고친 과정

고침 1.0 1999/05/05고친이 louis

이 글이 첫 선을 보였습니다.

고침 1.0.12001/08/21고친이 yongjoo

기존의 LinuxDoc 형식을 DocBook으로 변환

1. 일반적인 얘기들

1.1. CVS가 뭐예요?

CVS는 소스의 버전을 관리해 주는 시스템입니다. 이 말을 다시 풀어서 말하면, 프로그램(꼭

프로그램일 필요는 없습니다. 많은 CVS문서에서는 홈페이지의 경우에도 CVS를 통해서 관리할 수 있다고 합니다.) 을 개발하다 보면 각종 파일들을 수정하게 되는데 이 파일들의 버전을 관리해 주는 시스템입니다. 물론 혼자 개발할때도 유용하게 쓰일 수 있지만, 여러 사람이 동시에 하나의 프로그램을 개발할 때 진가를 발휘합니다.

1.2. 알아야 할 용어들

꼭 알아야 사용할 수 있는건 아니지만 알아두는게 여러모로 도움이 될때가 있는 용어들이 있습니다. 대표적인 것으로 revision(이 글에서는 용어에 대해서는 영어를 그대로 쓰도록 하겠습니다. 엉뚱하게 번역을 하면 읽기가 어렵더라구요.) 번호가 있습니다.

각각의 파일들은 자신의 revision 번호를 가집니다. 이 번호는 1.1, 1.2.3.2등과 같이 항상 짝수개의 숫자를 "."으로 연결한 형태가 됩니다. 만일 어떤 파일의 현재 revision 번호가 1.2.3.2 였고, 이 파일을 내가 수정을 했다면 현재 수정한 파일의 revision 번호는 1.2.3.3 이 됩니다. 즉 마지막 숫자가 하나 증가합니다.

또한 프로그램을 개발하다 보면 하나의 중심되는 개발 흐름이 있고, 이와는 별도로 작은 부분에 대한 개발 흐름이 있을 수 있습니다. 이런 중심되는 개발 흐름을 main trunk라고 하며, 이와는 다른 작은 부분에 대한 개발흐름을 branch 라고 합니다. 일반적으로 branch는 뻗어나온 main trunk의 revision번호에 숫자를 두자리 덧붙여서 사용합니다. 아래그림을 보면 제가 언급한 내용을 이해할 수 있을것입니다. (아래그림은 CVS 메뉴얼 (http://www.gnu.org/manual/cvs/index.html)에서 가져왔습니다.)

```
                                                      +-------------+
                     Branch 1.2.2.3.2 ->              ! 1.2.2.3.2.1 !
                                                    / +-------------+
                                                   /
                                                  /
                 +---------+    +---------+    +---------+    +---------+
Branch 1.2.2 -> _! 1.2.2.1 !----! 1.2.2.2 !----! 1.2.2.3 !----! 1.2.2.4 !
               / +---------+    +---------+    +---------+    +---------+
              /
             /
+-----+    +-----+    +-----+    +-----+    +-----+
! 1.1 !----! 1.2 !----! 1.3 !----! 1.4 !----! 1.5 !      <- The main trunk
+-----+    +-----+    +-----+    +-----+    +-----+
                !
                !
                !   +---------+    +---------+    +---------+
Branch 1.2.4 -> +---! 1.2.4.1 !----! 1.2.4.2 !----! 1.2.4.3 !
                    +---------+    +---------+    +---------+
```

또 알아둬야 할 용어중에 repository가 있습니다. (한글로 번역하면 저장소 정도가 되겠죠. 그냥 repository라고 쓰겠습니다.) repository는 쉽게 생각해서 현재 개발중인 소스를 모아둔 곳이라고 생각하시면 됩니다. 저도 더이상은 모릅니다. 많이 알면 머리만 아파요. ^^

2. CVS 사용하기

2.1. 시나리오

설명을 시작하기 전에 현재 독자는 다음과 같은 상황이며, 이 때문에 어쩔 수 없이 CVS를 사용해야 한다고 가정을 해 보도록 하겠습니다. 현재 독자는 프로그래머이며, Qt 2.0개발에 관심이 많습니다. 그래서 현재 개발중인 Qt 소스를 계속 지켜보고 싶으며, 가능하다면 소스를 수정해서 반영도 하고 싶습니다. 그래서 Qt 개발 홈페이지에 갔습니다. 그래서 "저도 참여하고 싶어요" 라고 메일을 썼더니, Qt 개발팀에서 아래와 같은 메일이 왔습니다.

```
...
Please use CVS. CVSROOT is :pserver:xxx@cvs.troll.no:/cvs .
...
```

으잉? CVS가 뭐야? 그리고 또 CVSROOT는 뭐지? 어떻게 쓰지?

2.2. Login

CVS는 위에서 설명한 것 처럼 Concurrent Version System의 약자입니다. -.-;

그럼 CVSROOT는 뭐냐구요? 바로 위에서 이야기 했던 repository입니다. 그냥 그렇구나 하시면 됩니다. 어떻게 쓰냐구요? 그냥 환경변수 CVSROOT를 지정하시거나, cvs를 사용하실 때 지정하시면 됩니다.

```
export CVSROOT=:pserver:xxx@cvs.troll.no:/cvs  (bash, ksh사용자)
setenv CVSROOT :pserver:xxx@cvs.troll.no:/cvs  (csh 사용자)

cvs -d :pserver:xxx@cvs.troll.no:/cvs cvs-command
```

이제 하셔야 할 내용은 CVS에 로긴하는 것입니다. 보통 로긴은 아래와 같이 입력하시면 됩니다.

```
cvs login
```

이 경우 CVS서버에 로긴을 요청하고, 설정에 따라서 비밀번호를 요구하는 경우도 있습니다. 보통의 경우 쓰기권한을 가진 경우에는 비밀번호를 요구합니다. 비밀번호는 메일등으로 전달이 되므로 잘 기억하셨다가 여기에서 입력하면 됩니다.

이렇게 한번 입력된 비밀번호는 CVSROOT와 함께 $HOME/.cvspass 파일에 저장이 되므로 다음번에는 입력할 필요가 없습니다.

2.3. Check out

여기까지 이해하는데 문제가 없었다고 생각하고, 본격적인 일을 해 보도록 하겠습니다. 우선 가장 먼저 해야할 일이 현재 repository에 있는 소스를 꺼내오는 일입니다. 이것을 check out

이라고 합니다. 사용방법은 쉽습니다. 그냥 아래와 같이 입력하시면 됩니다. (현재 개발중인 프로그램의 이름이 qt라고 가정합니다.)

```
cvs co qt
```

위와 같이 입력하여 check out을 마치면 qt라는 디렉토리 아래에 소스들이 들어가 있습니다. 근데 소스 이외에 CVS란 디렉토리가 있습니다. 이것은 CVS가 사용하는 디렉토리 이기 때문에 내부 내용을 절대 지우거나, 변경하지 마세요. 문제가 생기면 책임 못 집니다.

2.4. Update

위에서 check out한 소스코드를 보려는데, 갑자기 일이 터졌습니다. 옆에 처박아 뒀던 서버가 갑자기 말을 안 듣는거예요. 오래간만에 공부쫌 해보려고 했는데... 으<732> 열받아! 터진다. 터져! 마음 같아서는 망치로 문제가 생긴 서버를 요절내고 싶지만 딸린 처자식(물론 저는 아닙니다. ^^) 얼굴들이 생각나서 착한 내가 참는다는 마음으로 서버를 손보기 시작했습니다. 아<732> 그렇게 세월은 흘러갔습니다.

며칠이 지난후 여유가 생겨서 소스코드를 컴파일 해 보려고 자리에 앉았습니다. 순간 이런 생각이 들더군요. 며칠동안 좀더 업그레이드가 되었을꺼야! 그래! 결심했어. 다시 받아오는거야! -.-; 소스코드가 몇줄이 안되고, 가까운 서버에 있는 경우에는 문제가 없지만 만일 소스코드가 엄청나게 크고, 서버는 아주멀리 있는 경우에는 어느세월에 다시 check out을 할까요? 이때는 update명령을 사용하면 됩니다. update하고 싶은 디렉토리에 들어가서 단순히 아래와 같이 입력하시면 됩니다.

```
cvs up
```

만일 qt의 바뀐 소스를 모두 받아오려면 qt디렉토리에서 수행하면 되며, 그게 아니라 example만 다시 받아오고 싶다면 qt디렉토리 아래에 있는 example디렉토리에서 위의 명령을 입력하면 됩니다. 하나의 파일만 update하려면 파일이름을 마지막에 지정하시면 됩니다. 없다면 현재 디렉토리 전체를 update합니다. up명령을 사용하면 아래와 같은 메시지를 출력하면서 update를 수행합니다.

```
? Makefile
cvs server: Updating .
...
U htable.C
U htable.h
...
```

위에서 관심있게 보셔야 할 내용으로 첫번째 칼럼에 나오는 status입니다. 여기에는 여러가지 내용들이 알파벳 한글자로 나타나는데 여기에서 보이는 U 는 Updated의 의미입니다. 즉 자신의 로컬 하드디스크에 있는 소스의 버전보다 repository에 있는 파일의 버전이 높아서 다시 받아왔습니다. 그러나 메뉴얼에서는 위와 같이 설명을 하고 있지만, 제가 실제로 해 본 결과로는 repository에는 있는데 현재 로컬 하드디스크에는 없는경우 U라고 나왔고, 그렇지 않고 repository에 있는 파일의 버전이 높아서 다시 받아온 경우에는 P라고 나왔습니다. 복잡하게

생각하지 말고 U, P의 경우에는 문제없이 repository에 있는 소스를 가져왔다고 생각하시면 됩니다.

그럼 첫번째 줄에 있는 ?는 뭘까요? 이건 repository에는 없는 파일인데 로컬하드에는 있는 파일이라는 말입니다. 여기에서 Makefile은 로컬에서 생성한 파일이기 때문에 그렇습니다. 그냥 무시하시면 됩니다.

이것 이외에 중요한 status로는 M과 C가 있습니다. M은 repository에는 변화가 없는데 자신의 로컬 디스크에 있는 소스는 변한 경우입니다. 보통 개발자가 기능을 추가하기 위해서 소스를 수정한 경우입니다. 이 내용은 나중에 설명드릴 Commit으로 repository의 내용을 갱신시킬 수 있습니다. M 이외에 주의해서 봐야 할 status로 C가 있습니다. C는 Conflict의 의미로, 로컬 디스크의 파일도 변했고 repository의 내용도 변했으며 이 둘을 합칠 수 없는경우입니다. 보통 비슷한 부분을 두 명 이상의 개발자가 고친 경우입니다. 이 경우에는 update한 후 파일을 다시 수정해서 Commit해야 합니다. 이것 외에도 몇가지가 있지만 잘 사용되지 않으므로 필요한 분은 CVS메뉴얼 (http://www.gnu.org/manual/cvs/html_chapter/cvs_toc.html)을 읽어보시기 바랍니다.

2.5. Commit

받아온 소스를 열심히 컴파일 했습니다. 그래서 실행해 보니 짠! 하고 수행되는데 뭔가가 이상하더군요. 그래서 열심히 소스를 분석했습니다. 그래서 문제를 찾았죠. 장하다 대한의 건아! -.-; (오늘이 날씨도 좋은 5월 5일 어린이 날인데, 놀지도 못하고 회사에 있어서 맛이 점점 가고 있습니다. 이해를 해 주시길...)

문제가 되는 부분을 수정하고 이 내용을 repository로 보낼려면 commit을 수행하면 됩니다. commit하기 전에는 반드시 update를 해 보시기 바랍니다. 내가 아닌 누군가가 또 다시 소스를 바꿀 수 있습니다. update를 수행해서 conflict가 된 경우에는 바르게 고치고 다시 commit을 시도하셔야 합니다. commit하는 방법은 아래와 같습니다.

```
cvs ci filename
```

위와 같이 실행하면 갑자기 에디터 화면이 뜹니다. 바로 고친 내용을 적으라고 뜨는겁니다. 적당히 자신이 수정한 내용을 적은 후 저장하고 에디터를 빠져나오면 실재로 commit작업이 수행됩니다. 메시지를 잘 보시면 자신이 수정한 소스파일의 버전이 어떻게 변하는지 볼 수 있습니다.

어떤 사람의 경우 현재 뜨는 에디터가 마음에 안드는 경우가 있습니다. cvs는 기본적으로 CVSEDITOR 환경변수가 있는지 확인한 후 있으면 이 변수에 지정된 에디터를 띄우고, 없다면 EDITOR환경변수를 사용합니다. EDITOR환경변수마저도 없다면 vi를 사용합니다. 또한 에디터를 사용하지 않고, cvs에서 -m옵션을 사용하셔서 change내용을 적을 수도 있습니다.

```
cvs ci -m "나 금방 수정했다. 랄랄라~" filename
```

2.6. Diff

하지만 보통의 경우에는 CVS에 읽기권한만 있는 경우가 많습니다. 이 경우에는 CVS에 쓰기 권한이 있는 사람에게 메일을 통해서 바뀐내용을 보내줘야 하는데 이때 사용할 수 있는 명령어로 diff가 있습니다. 현재 수정한 파일이 driver.c 이며, 이 파일의 diff를 만들려면 아래와 같이 입력하시면 됩니다.

```
cvs diff driver.c > driver.c.diff
```

현재 디렉토리 전체의 diff를 만들려면? 물론 파일이름을 생략하시면 됩니다.

diff 명령의 경우에는 이와같은 용도로 사용할 수도 있지만 cvs repository의 변경된 내용을 확인하고 싶을때도 유용하게 사용됩니다.

2.7. Clean

더이상 qt에 관심이 없어서 그만 사용하고 싶다면 어떻게 할까요? 가장 쉬운 방법은 qt디렉토리를 지워버리면 됩니다. -.-; 무식하긴 하지만 많이들 이렇게 사용합니다. ^^

좀더 세련된 방법으로는 release 명령을 사용할 수 있습니다. 만일 아래와 같이 입력한 경우 현재 수정된 파일이 있는지 찾아주고, 디렉토리의 내용도 지워주므로 유용하게 사용하실 수 있습니다. (이 명령을 쓸 일은 거의 없습니다. 저도 메뉴얼에서만 봤을뿐, 실재로 사용해 본적은 한번도 없습니다. 저 역시 rm명령어를 애용합니다. ^^)

```
cvs release -d qt
```

3. CVS 관리하기

아직 내용이 없습니다.