CVS oinarriak

Victor Lascurain Artolazabal

ingeniaritza ELEKA linguistikoa

Aurkibidea

1 Sarrera.

Dokumentu honetan GNUko CVSaren erabilerari buruzko zenbait ohar ematen dira. 2. atalean oinarrizko aginduen zerrendatxo bat dago (zerrenda hau ez da osoa) eta 3.ean agindu horien erabilera ingurune praktiko batean ikusi ahal izateko, adibide gisa erabiliko dugun proiektu bat garatuko dugu. Adibidean zehar komando lerroko komandoak erabiliko ditugu, bezero grafikoak egon badauden arren (Unix/Linux, Mac eta Windowserako).

2 Oinarrizko CVS aginduak.

2.1 cvs checkout / cvs co.

Komando hau erabiltzen da CVSko fitxategien laneko kopia bat lortzeko.

2.2 cvs commit.

Komando hau erabiltzen da laneko kopian egin ditugun aldaketak CVSan gordetzeko. “-m” aukera erabili ezean, editore bat irekiko zaigu mezu bat sartu ahal izateko. Aldatutako fitxategien kopia berri bat sortuko da CVSan eta bertsio zenbakia handituko da.

2.3 cvs update.

Komando honen bidez gure laneko kopia eguneratu dezakegu, beste erabiltzaileek egin dituzten aldaketak gure laneko kopiara ekarriz. Gatazkak sortzen diren kasuan (ikusi 3.7. eta 3.8. puntuak) CVSk egoera honen berri emango digu.

2.4 cvs export.

“checkout”en eragin antzekoa du. Diferentzia zera da, “checkout”k laneko kopia bat sortzen duela, gure lanaren parte diren fitxategiez gain CVSk bere lanerako behar dituenak ere sortzen direlarik eta “export”ek ez dituela azken hauek sortzen. Komando hau erabilgarria da bezeroei edo, orokorrean, CVSan aldaketak egiteko beharrik/baimenik ez duten entitateei gure fitxategien kopia bat bidaltzeko.

2.5 cvs import.

Aurrekoaren kontrakoa, nolabait esateko. Fitxategi multzo bat hartu (direktorio eta azpidirektorioekin) eta CVSan denak batera “1” bertsio gisa sartzeko erabiltzen da, besteak beste, komando hau.

2.6 cvs add.

Komando honen bidez lanean ari garen bitartean sortzen ditugun fitxategi berriak CVSan sar ditzakegu. Komando honek gure laneko kopian bakarrik izango du eragina, CVSa “commit” exekutatzen dugunean eguneratuko da.

2.7 cvs remove

Komando honen bidez erabiltzen ez ditugun eta beharrezkoak ez diren fitxategiak ezabatu ditzakegu. Hau egiten ez badugu, “checkout” bat edo “update” bat exekutatzen dugun bakoitzean, fitxategi horiak berriz agertuko zaizkigu. Ezabatutako fitxategien kopia zaharrak ez dira ezabatzen.

3 Kasu praktikoa.

Atal honetan aplikazio txiki baina erreal bat CVSaren laguntzaz nola garatuko genukeen ikusiko dugu. Gure aplikazio honen helburua Euskarazko hitzak silabatan zatitzeko gai den programatxo bat sortzea izango da, HYPHENATOR deituko duguna. Adibidea garapenean zehar gerta daitezkeen egoera arruntenak zeintzuk diren eta beraiei nola aurre egin erakusteko bereziki aukeratutako pausotan dago banatuta. CVSaren oinarrizko funtzio guztiak erakusten saiatu gara, baina, aldi berean, errealitatean gertatuko ez liratekeen egoerak artifizialak baztertuz. Pausoen aukera egokia izango dela espero dugu.

3.1 Proiektuaren hasiera.

Bezeroek eskatuta edo barne beharrei erantzuteko, hitzak silabatan zatitzen dituen programa baten beharra sumatu da talde barruan. Behar honi erantzuteko hitzak zatitzeko python programatxoa idatzi du Gregok: `zatitu_hitzak.py`.

3.2 CVSan 1. bertsioa sartu.

Gregok programatxoa CVSan sartzen du, aldaketak kudeatu ahal izateko. Komando honek: `cvs import -m'Hitz zatitzaile proiektuaren hasiera.' HYPHEN Bittor start` CVSan HYPHEN direktorioa sortzen du uneko direktorioaren edukiarekin. Lehenengo bertsio honi “start” etiketa jartzen dio eta “Bittor”ek sartu duela adierazi. “-m” erabiltzen da datu basean gordeko den mezua komandoarekin batera idazteko.

```
$cvs import -m'Hitz zatitzaile proiektuaren hasiera.' HYPHEN
Bittor start
N HYPHEN/zatitu_hitzak.py

No conflicts created by this import
```

Orain, “zatitu_hitzak.py” CVSan dago. Bertsio kudeaketak funtziona dezan beharrezkoa da CVStik ateratako laneko kopietan lan egitea. Horretarako “checkout” bat egin behar da.

3.3 Bi langile kode berean.

Ibonek programatxoan zenbait hobekuntza egin ditzakeela uste du eta laneko kopia bat lortzen du.

```
$cvs co HYPHEN
cvs checkout: Updating HYPHEN
U HYPHEN/zatitu_hitzak.py
```

Agindu horrek HYPHEN direktorio bat sortuko du Gregok idatzitako `zatitu_hitzak.py` programatxoa eta CVS izeneko direktorio batekin.

```
./HYPHEN
./HYPHEN/CVS
./HYPHEN/CVS/Root
./HYPHEN/CVS/Repository
./HYPHEN/CVS/Entries
./HYPHEN/zatitu_hitzak.py
```

Direktorio hori CVSak bere lana egiteko behar du, ez da gure proiektuaren parte baina EZ DA EZABATU BEHAR.

3.4 Fitxategi ugalketa.

Ibonek fitxategia bi zatitan banatzea egokia dela erabakitzen du: liburutegi bat eta programatxoa bera. Aldaketak egin eta CVSari aldaketen berri ematen dio.

```
$cvs add hyphenator.py
cvs add: scheduling file `hyphenator.py' for addition
cvs add: use `cvs commit' to add this file permanently

$cvs commit -m 'Programa eta liburutegia banatuta.'
cvs commit: Examining .
RCS                              file:                              /
home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/hyphenator.py,v
done
Checking in hyphenator.py;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/hyphenator.py,v     <--
hyphenator.py
initial revision: 1.1
done
Checking in zatitu_hitzak.py;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/zatitu_hitzak.py,v
<--  zatitu_hitzak.py
new revision: 1.2; previous revision: 1.1
done
```

3.5 Kodearen banaketa.

Programaren Ibonen bertsioa (aurreko puntuko aldaketekin lortutakoa) Alatzneri eman nahi diogu, bere lanerako erabil dezan. Alatzne ez da garatzaileen taldekoa. Horretarako Ibonek CVSko azken bertsioa etiketatzen du eta gero esportatu egiten du.

```
$cvs tag hyphenator_1-0
cvs tag: Tagging .
T hyphenator.py
T zatitu_hitzak.py
$cvs export -r hyphenator_1-0 HYPHEN
cvs export: Updating HYPHEN
U HYPHEN/hyphenator.py
U HYPHEN/zatitu_hitzak.py
```

Orain argi ikus dezakegu laneko kopia eta esportatuko kopiaren arteko diferentzia. Esportatutako kopian ez dago CVS direktorioaren aztarnarik ere.

```
./HYPHEN
./HYPHEN/hyphenator.py
./HYPHEN/zatitu_hitzak.py
```

Etiketak gauza askotarako dira erabilgarriak. Kasu honetan, kanpoko entitate bati ematen diogun kodea zein den zehazki jakin ahal izateko.

3.6 Lana paraleloan.

Gregok lanean jarraitu nahi du. Bere laneko kopia eguneratzen du, programa nagusiari erabilerari buruzko mezua gehitzen dio (parametro gabe exekutatzen denerako), eta CVSan sartzen ditu aldaketak.

```
$cvs update
cvs update: Updating .
U hyphenator.py
U zatitu_hitzak.py
```

Ibonek sortutako fitxategi berria (hyphenator.py) automatikoki agertzen da Gregoren laneko direktorioan. Gregok bere lana egiten du eta CVSan gorde. Logetan gelditukо den mezua ere gehitzen du noski: “Erabilerari buruzko mezua gehituta.”.

```
$cvs commit -m 'Erabilerari buruzko mezua gehituta.'
cvs commit: Examining .
Checking in zatitu_hitzak.py;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/zatitu_hitzak.py,v
<--  zatitu_hitzak.py
new revision: 1.3; previous revision: 1.2
done
```

3.7 Lan paraleloaren arazoa: bertsio gatazka.

Ibonek ere ideia berbera izan du eta bere aldaketak egiten dizkio programari (hor sortzen da gatazka). Ibonen "commit"ak errorea ematen du, “1.3” zenbakia duen “zatitu_hitzak.py” fitxategiaren kopia bat badagoelako.

```
$cvs commit -m'Erabiltzaileari laguntzeko mezua gehituta.'
cvs commit: Examining .
cvs commit: Up-to-date check failed for `zatitu_hitzak.py'
cvs [commit aborted]: correct above errors first!
```

3.8 Gatazken konponbidea.

Ibonek bere laneko kopia eguneratu behar du. CVSa bertsio gatazkaz ohartuko da eta ahal duen neurrian arazoan konpontzen saiatuko da.

```
$cvs update cvs update: Updating .
RCS                          file:                          /
home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/zatitu_hitzak.py,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into zatitu_hitzak.py
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in zatitu_hitzak.py
C zatitu_hitzak.py
? hyphenator.pyc
```

Beltzez dagoen zatiak erakusten du CVSak egin duena. CVSko bertsioa (1.3) eta Ibonen laneko direktoriokoa (1.2) konbinatu. Ezin izan du modu seguruan egin, eta Ibonek lan pixka bat egin beharko du.

```
import sys, hyphenator
<<<<<<< zatitu_hitzak.py

if len(sys.argv) == 1:
      print """Aizu, hori ez da programtxo honi deitzeko modu egokia.
Mesedez, eman ezazu gutxienez zatitu beharreko hitz bat. Horrela
egiten ez duzun bitartean mezu hau jasoko duzu behin eta berriz."""
      sys.exit(1)
=======

if len(sys.argv) == 1:
      print >>sys.stderr, "Erabilera: %s hitza_1 hitza_2 ... hitza_n"%sys.argv
[0]
>>>>>>> 1.3
for word in sys.argv[1:]:
      print hyphenator.splitWord(word)
sys.exit(0)
```

“<<<<<<<”, “=======” eta “>>>>>>>” bidez dago zati arazotsua mugatuta. “=======” gaineko partean Ibonen laneko kopia dago, eta azpian Ibonek CVStik ateratako bertsioa.

```
import sys, hyphenator

if len(sys.argv) == 1:
      print  >>sys.stderr,  "Erabilera:  %s  hitza_1  hitza_2  ...  hitza_n"%
sys.argv[0]
      sys.exit(1)
for word in sys.argv[1:]:
      print hyphenator.splitWord(word)
sys.exit(0)
```

Bertsio bien konbinaketa egindakoan, goian ageri den bezalako emaitza lor dezakegu. Ikusiko dugun bezala, orain ez dago inolako arazorik “commit” egiteko ordua.

```
$cvs commit -m'Itzulera balioak gehituta (exit bidez).'
cvs commit: Examining .
Checking in zatitu_hitzak.py;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/zatitu_hitzak.py,v
<--  zatitu_hitzak.py
new revision: 1.4; previous revision: 1.3
done
```

Ikusten den bezala, 1.4 bertsioa da orain unekoa. Normala den moduan, logerako mezua ere aldatu du Ibonek.

3.9 Aldaketa handietarako prestatu.

Programatxoa C++en inplementatzeko erabakia hartzen dute. Aldaketa handiak egitea aurreikusten denez, "release" berri bat egiteko une egokia dela dirudi gainera. Horretarako uneko bertsioak etiketatuko ditugu. Oro har, garapenean zehar “puntu garrantzitsuetan” CVS datu-basea etiketatzea komenigarria da.

```
$cvs tag hyphenator_1-1
cvs tag: Tagging .
T hyphenator.py
T zatitu_hitzak.py
```

3.10 Fitxategiak gehitu eta ezabatu.

Fitxategiak nola gehi daitezkeen ikusi dugu. Batzutan ordea, beharrezkoa izaten da fitxategiak ezabatzea ere. Horretarako daukagu “remove” komandoa.

```
$cvs add configure.ac GDB hyphenator.cpp hyphenator.h Makefile.am
zatitu_hitzak.cpp
cvs add: scheduling file `configure.ac' for addition
cvs add: scheduling file `GDB' for addition
cvs add: scheduling file `hyphenator.cpp' for addition
cvs add: scheduling file `hyphenator.h' for addition
cvs add: scheduling file `Makefile.am' for addition
cvs add: scheduling file `zatitu_hitzak.cpp' for addition
cvs add: use `cvs commit' to add these files permanently
$rm hyphenator.py zatitu_hitzak.py
$cvs remove hyphenator.py zatitu_hitzak.py
cvs remove: scheduling `hyphenator.py' for removal
cvs remove: scheduling `zatitu_hitzak.py' for removal
cvs remove: use `cvs commit' to remove these files permanently
```

Fitxategiak CVStik ezabatu aurretik laneko direktoriotik ezabatu behar dira (bestela CVS kexatu egiten da). Mezuak dioen bezala, aldaketak ez dira behin-betikoak “commit” exekutatu

arte. Orduan ere, hurrengo bertsiotik aurrera izango dute eragina, hau da, ezabatutako fitxategiak ez dira galtzen, ezabatu aurreko azken bertsioa datu-basean geratzen da.

CVSan automatikoki sortzen diren fitxategiak sartzea ez da, oro har, ideia ona. Har dezagun adibide gisa gure programatxoaren C++ bertsioa. Hauek dira CVSan sartu ditugun fitxategiak: iturburu kodea eta konpilaziorako fitxategiak.

```
configure.ac  GDB             hyphenator.h  zatitu_hitzak.cpp
CVS           hyphenator.cpp  Makefile.am
```

Konparatu dezagun hori konpilazio prozesuan sortze diren fitxategi multzoarekin.

```
aclocal.m4      config.status  hyphenator.cpp    libtool      mkinstalldirs
autom4te.cache  config.sub     hyphenator.h      ltmain.sh    stamp-h
config.guess    configure      hyphenator.lo     Makefile     stamp-h.in
config.h        configure.ac   hyphenator.o      Makefile.am  zatitu_hitzak
config.h.in     CVS            install-sh        Makefile.in  zatitu_hitzak.cpp
config.log      GDB            libhyphenator.la  missing      zatitu_hitzak.o
```

3.11 Adarren zergaitia.

Alatznek akats bat aurkitu du bere programatxoan: "t[aeiou]" duten hitzak ez dira ondo banatzen. Iboni heltzen zaio konpontzeko enkargua BAINA orain C++ inplementazioa da daukaguna. Alatznek ez du C++ bertsioa nahi, Pythonekoa nahi du, bertsio zaharra, zuzenduta noski. Horretan laguntzen digute adarrek, bertsio zaharren gaineko aldaketak egiten, azken bertsioekin nahasi gabe. Esperimentuak egiteko ere baliagarriak dira (adar esperimentalak sortuz). Lehenengo, ikus dezagun zein den akatsa.

```
$zatitu_hitzak.py kaixo agur zatitu zatiketa
kai*xo
a*gur
zatitu
zati*keta
```

3.12 Adarra sortu.

Ibonek adar bat sortzen du (branch) Alatzneri pasatako bertsioaren "tag"ean (kasu honetan hurrengoan) oinarrituta. Aldaketak egiten ditu, berriz etiketatu (1.1.1 edo) eta Alatzneri pasa.

Lehenengo pausoa lanerako bertsio egokia lortze da.

```
$cvs co -r hyphenator_1-1 HYPHEN
cvs checkout: Updating HYPHEN
U HYPHEN/hyphenator.py
U HYPHEN/zatitu_hitzak.py
```

“checkout” komandoaren “-r” parametroaren bidez egiten da hori. Defektuz, “checkout”ek azken bertsioak ateratzen ditu. “-r etiketa” parametroarekin “etiketa” etiketa duten bertsioa atera dezakegu.

Bigarren pausoa adarra sortzea da, laneko bertsioan oinarrituz (hau ez da modu bakarra, baina agian errazena da).

```
$cvs tag -b hyphenator_1-1-zuzenketak HYPHEN
cvs tag: Tagging HYPHEN
T HYPHEN/hyphenator.py
T HYPHEN/zatitu_hitzak.py
```

Adarra etiketa mota berezi bat dela esan dezakegu. Diferentzia “-b” marka da (branch).

Hirugarren pausua laneko direktorioa adarrera pasatzea da. Oso akats arrunta da pauso hau ahaztea. Kontuan hartu behar da “tag -b”k adarra CVS datu-basean sortzen duela, baina ez ditu laneko direktorioko CVS azpidirektorioan beharrezko aldaketak egiten. Hori dela eta, beharrezkoa da pauso hau.

```
$cvs update -r hyphenator_1-1-zuzenketak
cvs update: Updating .
```

Aldaketak egin eta zuzendutako bertsioa ondo dabil.

```
$zatitu_hitzak.py kaixo agur zatitu zatiketa
kai*xo
a*gur
za*ti*tu
za*ti*ke*ta
```

3.13 “commit” adar batean.

Laneko direktorioa adar batekoa bada, “commit” egiten dugunean ez da azken bertsioa aldatzen, adarreko azken bertsioa baizik. Ikus dezagun komandoa zein den.

```
$cvs commit -m'ta, te, ti, to, tu silabe akatsa zuzenduta.'
cvs commit: Examining .
Checking in hyphenator.py;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/Attic/hyphenator.py,v
<--  hyphenator.py
new revision: 1.1.2.1; previous revision: 1.1
done
```

Ikusten den bezala, 1.1 bertsiotik ez gara 1.2ra pasa, 1.1.2.1era baizik, hots, 1.1.2 adarreko lehenengo bertsiora.

“tag” komandoa erabiliz uneko bertsioa etiketatzen denez, 1.1.2.1 bertsioa etiketatzen du komando honek.

```
$cvs tag hyphenator_1-1-1
cvs tag: Tagging .
T hyphenator.py
T zatitu_hitzak.py
```

Orain prest gaude bertsio berri bat exportatzeko.

3.14 Azken bertsiora itzuli.

C++ bertsioa ere konpondu behar da noski! Check out bat egin, zuzendu eta aurrera. Horretarako nahikoa da inongo parametrorik gabe “checkout” bat exekutatzea.

```
$cvs co HYPHEN
cvs checkout: Updating HYPHEN
U HYPHEN/GDB
U HYPHEN/Makefile.am
U HYPHEN/configure.ac
U HYPHEN/hyphenator.cpp
U HYPHEN/hyphenator.h
U HYPHEN/zatitu_hitzak.cpp
```

Egin beharreko aldaketak egin eta CVS datu-basean sartu.

```
$cvs commit -m'ta, te, ti, to, tu silabe akatsa zuzenduta.'
cvs commit: Examining .
Checking in hyphenator.cpp;
/home/victor/cvs_aurkezpena/CVS_DIR/HYPHEN/hyphenator.cpp,v   <--
hyphenator.cppnew revision: 1.2; previous revision: 1.1
done
```

Ikusten den bezala, bertsio zenbakia 2 luzerakoak dira berriz ere.

4 Bukaerako oharrak.

Azalpen hau lagungarria izango zela espero dugu. Dena dela, badaude azaldu ez diren CVSaren ezaugarri mordo bat, besteak beste “watch”ak, fitxategiak blokeatzeko aukera, plantilak, “diff”/“status”/”log” komandoak, fitxategi bitarrak kudeatzeko moduak, erabiltzaileen kudeaketa... Guzti hauen berri izateko jo CVSaren dokumentaziora (info).