Verziókezelés határok nélkül – A Bazaar és az Olive felület

A verziókezelésről a Linux felhasználók jó része a CVS-re asszociál, érthető módon, hiszen rengeteg nyílt forráskódú projekt tárolja ebben a fejlesztés eredményét. Léteznek azonban kicsit más elgondolás alapján működő rendszerek is. A cikk célja kettős: egyfelől a verziókezelő rendszerek fontosságára akarja felhívni a figyelmet, másrészt pedig egy friss fejlesztéssel is megismerkedünk.

Kik jelentik a célcsoportot?

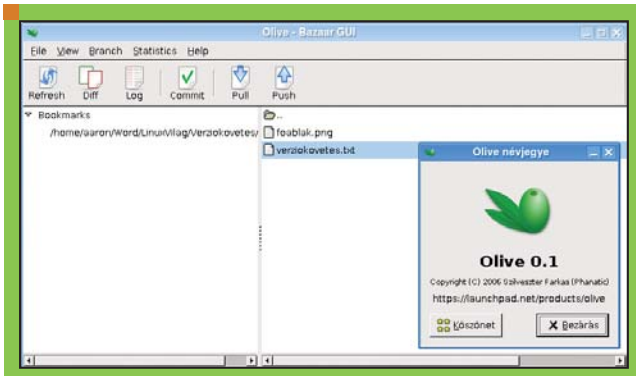
Minden olyan felhasználó, aki termelékeny munkát végez a számítógéppel. Dokumentumokat írunk, digitális képeket retusálunk, prezentációkat készítünk, a rendszer beállítófájljait hangoljuk, programokat fejlesztünk és eközben sok esetben semmilyen verziókezelő rendszert nem használunk. De miért is kellene? Talán velünk is megesett, hogy munkánk során kiderült, hogy tegnap még jó irányba haladtunk, de egy apró módosítás miatt mára sikerült tönkretenni az egészet. Milyen jó lenne előszedni a tegnapi verziót!
Sőt, még jobb lenne, ha azt is láthatnánk, hogy mi az a bizonyos módosítás, amire már nem is emlékszünk.
Ez mindenféle munka során előfordulhat. Ezekre az esetekre (meg még számtalan másra) találták ki a verziókezelést! Egy bizonyos összetettségi szint alatt az életünket csak megnehezíti a verziókezelő rendszer használata, semmint, hogy megkönnyítené. Ezt a szintet magunknak kell megállapítani, hogy mikor ér meg egy kis plusz odafigyelést az, hogy az egyes munkafázisokat pontosan nyomon tudjuk követni és elő tudjuk szedni. Például egy cikk megírásakor nem szoktam verziókezelő rendszert használni. Ezzel a cikkel kivételt teszek a képernyőfotók kedvéért, hogy lássuk, hogyan alakul a szöveg.
Most mindjárt be is rakom az aktuális állapotot a verziókezelő rendszerbe!
Elkészült az első verzió. Hol is tartottunk? Talán az a tévhit él a verziókövetésről, hogy szöveges adatokkal használhatóak csak. Ez valamilyen szintig igaz is, hiszen egy bináris fájlon megnézni, hogy mi változott az egyes változatok között, nem valami vidám mulatság, de a legfontosabb előnyt nem veszítjük el: a rendszer helyettünk gondoskodik arról, hogy bármikor elő tudjuk szedni a megadott állapotot. Ezt persze akár magunk is megtehetjük egy precíz fájlelnevezési konvencióval, de azért az jóval nehézkesebb és több odafigyelést igényel. Ráadásul minden egyes fájlverzióhoz megjegyzést is fűzhetünk, amiből kiderül az éppen aktuális módosítás célja. Talán mostanra már mindenki kedvet kapott a kipróbáláshoz, úgyhogy nézzünk egy konkrét megvalósítást!

Bazaar – elosztott verziókezelés

Ha analógiát akarunk keresni, akkor azt mondhatjuk, hogy a fájlcserélők után a verziókezelő rendszerek is beálltak a sorba: a decentralizálásnak mindenütt vannak előnyei. Gondoljunk csak az internetre, ott is a decentralizálás gondolatával kezdődött a fejlesztés. Mit is ígér egy verziókezelő, ha elosztott? A legnagyobb előnyként azt említi a *Bazaar* leírása, mely a ➲ *http://bazaar-vcs.org* címen érhető el, hogy élő internetkapcsolat nélkül is lehet dolgozni. Értve ezt mind a kliens illetve a szerver kapcsolatára. Hiszen ahogyan mondjuk egy repülőn ülve nem szokott internethozzáférés lenni, úgy a szerver kiesése is okozhatja egy centralizált rendszer működésképtelenségét. A saját gépen lévő lerakatba (*repository*) lehet beküldeni (*commit*) a munka eredményét. További előnyöket is felsorol, melyek leginkább szoftverfejlesztőknek lehetnek hasznosak. Lehet azonban centralizált módon is dolgozni a *Bazaarral*, ekkor nagyon hasonlít a működése a *CVS*-re. Egyébként sem különösebben tér el, már azokban a dolgokban ami a felhasználó felé jelentkezik. Ez jól is van így, aki használt már változatkezelő rendszert, gyorsan megbarátkozik ezzel is.
A *Bazaar* egy karakteres felületű alkalmazás, így a bzr(1) man oldalon részletes leírást találhatunk róla. Én viszont a cikkeket grafikus felületen szoktam megírni és sokan vannak olyan Linux felhasználók, akik csak akkor nyúlnak parancssorhoz, ha másképp nem lehet megoldani a problémát, így most is így teszünk!
A *Summer of Code* keretében elkészült az *Olive* felület a *Bazaarhoz*, így a verziókezelés használata nemcsak hasznos, de egyszerű is!

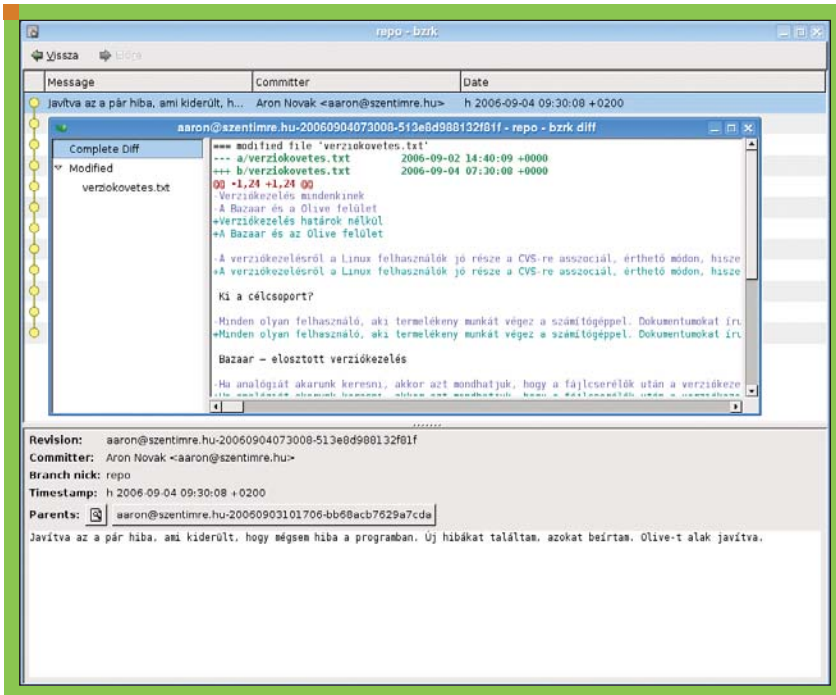
Olive – egy újabb magyar fejlesztés

Tehát grafikus felületen használnánk a Bazaar által felkínált lehetőségeket? A ➲ *http://bazaar-vcs.org/Olive* címről tudjuk letölteni a legfrissebb verzióját *Farkas Szilveszter* fejlesztésével elkészült programnak.



1. ábra Az Olive képernyője

A szoftver ***Pythonban*** íródott és a ***GTK***-t használja. A 0.10.0-es verzió volt a legfrissebb a cikk megírásakor. Ahhoz, hogy az ékezeteink rendesen működjenek, nem árt, ha a rendszerünk ***UTF-8***-as karakterkódolásra van állítva. A csomag letöltése után a `setup.py` paranccsal tudjuk telepíteni a szoftvert. Ezután az `olive-gtk` kiadásával indul az alkalmazás.
Ahhoz, hogy el tudjuk kezdeni használni az ***Olive*** lényegi funkcióit, először létre kell hoznunk egy tárolót. Tipikusan egy tárolóban egy adott munkához tartozó fájlokat tárolunk, például ennél a cikknél a szövegfájl és a képernyőfotók tartoznak ide. Abban a könyvtárban, ahol dolgoztunk vagy dolgozni fogunk, adjuk ki a `bzr init`-nek megfelelő parancsot az ***Olive***-ban: a Branch/Initialize menüvel tehetjük ezt meg. Ez létrehozza azt a struktúrát, amelyre a ***Bazaarnak*** szüksége van a verziók tárolásához (ez az adott könyvtárbeli rejtett ***.bzr*** könyvtárban látható).
Az `init` parancs nem adta a tárolóhoz hozzá önműködően az összes benne lévő fájlt és könyvtárat, ezt nekünk kell megtenni az `add` utasítással. Ezután nincs más hátra, mint nekilátni a munkának. Gépeljünk, kódoljunk, tervezzünk, rajzolgassunk és mikor úgy érezzük, hogy most már készen van egy részfeladat, vagy mondjuk vége a munkanapnak akkor vegyük elő az ***Olive***-ot és adjuk ki a fájlra (fájlokra) a `commit` parancsot. Ezzel mondjuk meg, hogy az aktuális állapotot vegye fel a rendszer egy új verzióba. Ezzel készen is volnánk. Tulajdonképpen a plusz munka, mely szükséges ahhoz, hogy használhassuk a verziókezelést, ennyivel véget is ér. Most lássuk, mit kapunk cserébe!
Két változat között láthatjuk a különbségeket, ez nyilvánvalóan csak szöveges fájloknál használható. A ***Bazaar*** oldala alapján azonban tervezik azt a lehetőséget a fejlesztők, hogy külső programmal bizonyos bináris fájlok közti különbség mégis szemléletes legyen. Most a ***diff*** csak annyit árul el két bináris fájlról, hogy azok különböznek-e vagy sem. Elképzelhető, hogy később például ***OpenOffice.org*** fájlok esetén egy külső programmal láthatóvá válik két beküldött ***odt*** fájl közötti különbség is.
Ezt a ***Statistics/Logs*** menünél a ***Parents*** mellett lévő nagyítóval kérhetjük. Pont azokat a változásokat fogjuk megkapni, amin éppen állunk.
Ha egy régebbi verzióra van szükségünk, akkor azt a ***Branch/Get*** menü alatt tudjuk kérni. Beírjuk a ***Branch*** elérési útvonalát (ez a `bzr inittel` előkészített mappa), kiválasztjuk, hogy hova akarjuk létrehozni az adott verziót, aztán a ***Revision Numberrel*** pedig azt adjuk meg, hogy hányadik módosításig alkalmazzuk a változásokat. Ennyi az egész.
Szoftverfejlesztéshez ennél jóval összetettebb módon is lehet használni a ***Bazaart***, de alapvetően a verziókezelő rendszerek ezeket a szolgáltatásokat nyújtják. Az ***Olive*** pedig segít nekünk, hogy minél kisebb energia ráfordításával tudjunk részesülni mindezen előnyökből.
A ***Bazaart*** jó eséllyel megtaláljuk a disztribúciónk csomagjai között, az ***Olive*** pedig már bekerült az ***Ubuntu Universe*** csomagjai közé, remélhetőleg hamarosan a többi disztribúcióhoz is lesz csomag.



2. ábra A különbségek listája (diff) és a napló, melyet magunk írhatunk az egyes verziókhoz

Novák Áron
(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.