Yeni Geliştirici Kılavuzu

Barış Metin <baris@uludag.org.tr>, A. Murat Eren <meren@uludag.org.tr>

30 Mayıs 2005

Özet

Bu belge yeni bir geliştiricinin Uludağ projesi geliştirme sürecine dahil olmak için bilmesi ve yapması gerekenleri tarif etmektedir.

İçindekiler

1 Giriş

Bu belgede Ulusal Dağıtım projesi bünyesinde geliştirilen Pardus İşletim Sistemi'ne katkıcıların nasıl destek verebileceğini ve bir Pardus geliştiricisi olmak için gerekenleri anlatmaya çalışacağız. Özelde Uludağ geliştirme modelini anlatıyor olsak da, sanırız belgede anlatılanlar hemen hemen tüm dağıtım projelerini tarif edebilir.

Bu belgenin konusu dışında kalan tüm bilgiye Ulusal Dağıtım web sayfalarından ulaşabilirsiniz. Web sayfalarından belgeler, kullanılan araçlar ve servisler ile ilgili gerekli bilgiyi alabilirsiniz. Ulusal Dağıtım web sayfaları `http://www.uludag.org.tr` adresindedir.

2 Geliştirici kimdir?

Kaba hatları ile projenin iki tür görevi yerine getirmesi gerektiğini söylemek yanlış olmayacaktır. Her dağıtımın yaptığı gibi yazılımları bir araya getirmek ve bunun için gerekli alt yapıyı hazırlama ve ayakta tutma işlerinin yanı sıra dağıtımın ruhunu oluşturacak yeni araçlar ve teknolojiler geliştirmek yapmamız gerekenler arasındadır.

Geliştirici ile yalnızca program yazan kişileri değil, bir yazılım projesini başarılı kılmak için yapılması gereken herhangi bir işi yerine getiren kişileri tarif ediyoruz. Bir yazılım projesini hayata geçirmek için programlama yanında, belgeleme, hata kontrolleri, görsel materyaller, çeviri çalışmaları, vb. pek çok alanda çalışma yapılmasının gerektiğinin farkındayız. Pardus içerisinde de tüm bu çalışmalara ihtiyacımız var.

Nasıl başlayacağım?

Öncelikle gidişata ayak uydurmaya çalışmak gerekiyor. Bunun için bir süre izlemek ve işlerin nasıl yürüdüğünü gözlemlemek genellikle işleyen bir yöntemdir. Uludağ web sayfalarından ulaşılabilen e-posta listelerine üye olarak tartışmaları ve yapılanları izlemek, yayınlanan dökümanları incelemek, yapılan hata raporlarını ve hatalara getirilen çözümleri gözlemlemek faydalı olacaktır.

Daha sonra tanımlı yöntemlere göre geliştirmeye yardımcı olabilirsiniz. Yazılımları test ederek hata raporlarına yorumlarınızı ve çözüm önerilerinizi ekleyebilir, bulduğunuz yeni hataları bildirebilirsiniz. Yenilikçi teknolojilerin geliştirilmesine katkıda bulunabilir, yeni özellikler ekleyebilirsiniz. Ya da süreçte işlemediğini (yavaş işlediğini) düşündüğünüz bir adıma destek olabilir, onu hızlandırabilirsiniz. Fakat her zaman diğer geliştiriciler ve katkıcılar ile iletişim halinde olmalısınız. Birden fazla kişinin çalıştığı her işte olduğu gibi burada da birbirimizden haberdar olmamız gerekiyor.

İhtiyaçlarımız aşağıdaki maddeler ile özetlenebilir.

2.1 Yazılım geliştirme, hata ayıklama

Yazılım geliştirme konusundaki bilginizi ve emeğinizi bu alanda çalışan kaynak temizleme ve yenilikçi teknolojiler ekiplerine yardımcı olarak aktarabilirsiniz. Onlara nasıl yardımcı olabileceğinizi en iyi kendileri söyleyeceklerdir. `http://liste.uludag.org.tr/cgi-bin/mailman/listinfo` adresi yardımı ile üye olabileceğiniz Uludağ e-posta listeleri size bu konuda yardımcı olacaktır.

Web sayfalarından ulaşılabilecek Uludağ hata takip sistemini kullanarak raporlanmış hataları inceleyebilir ve çözüm getirebilirsiniz.

2.2 Çeviri

Özgür yazılımların yerelleştirilmesi, doğru ve eksiksiz bir Türkçe ile kullanılabilmeleri bir diğer ihtiyacımız. Bunun için hemen her büyük özgür yazılım projesi için oluşturulmuş olan yerelleştirme ekipleri ile birlikte çalışabilirsiniz. Öncelikli olarak Uludağ için seçtiğimiz yazılımların yerelleştirme çalışmalarına destek olmanız bize doğrudan fayda sağlayacaktır.

Kısaca yerelleştirme çalışmalarına destek olun; ham çevirinin ötesinde çevirilerin kalitelerini de arttırmaya çalışın. Çeviri çalışmalarına destek olmakla sadece Ulusal Dağıtım'a değil, mevcut tüm Linux dağıtımlarına da sorunsuz Türkçe'ye sahip paketler kazandırarak önemli bir misyonun parçası olabilirsiniz.

Belli başlı çeviri gruplarına yardım etmek için `http://cekirdek.uludag.org.tr/~baris/tmp/ceviri-calismalari.html` adresindeki yerelleştirme çalışmaları listesinden ilgi duyduğunuz yerelleştirme çalışmalarına ulaşabilirsiniz.

2.3 Test ve hata bildirimleri

Seçtiğimiz yazılımları test ederek hata bildiriminde bulunabilirsiniz. Hataları diğer dağıtımlarda kontrol ederek durumu raporlayabilirsiniz. Diğer dağıtımlarda da bu sorunun olduğunu bilmek doğru bir çözüm üretmek konusunda bize yardımcı olabilir. Eğer sorun test ettiğiniz bir dağıtımda çözülmüş (veya hiç yok) ise, çözüme ulaşmamız o dağıtımın yazılımla ilgili yaptığı çalışmaları inceleyerek daha hızlı olacaktır. Hata takip sistemimize `http://bugs.uludag.org.tr/` adresinden erişebilirsiniz.

2.4 Grafik Tasarım, Çoklu ortam

Eğer bu konuda yeteneğiniz varsa dağıtım içerisinde kullanılacak ikon setleri, yazı tipleri, renk temaları gibi grafik bilgisi gerektiren konularda yardımcı olabilirsiniz. Çalışmalarınızda ihtiyaç duyabileceğiniz görsel materyal konusunda e-posta listelerimiz yoluyla bizden yardım isteyebilirsiniz.

2.5 Belgeleme

Yürütülmekte olan projelerin belgelemesine destek olabilirsiniz. Kullanıcı belgelerinin yanı sıra projeye yeni katılacak geliştiriciler için “Nasıl yapılır” belgeleri hazırlayabilirsiniz. Web sayfalarımızın güncellenmesinde bize yardımcı olabilirsiniz ya da sayfalarımızın İngilizce, Almanca, İspanyolca gidi yabancı dillerdeki çevirilerinin güncel olmasını sağlayabilirsiniz.

2.6 Tanıtım

Projenin tanıtımına destek olarak daha çok insanın haberdar olmasını sağlayabilirsiniz.

Muhtemelen yardım edebileceğiniz konuların listesi yukarıdakiler ile sınırlı değildir. E-posta listelerindeki proje ile ilgili tartışma konularında fikirlerinizi belirterek, önerilerinizi sunarak ve yapıcı eleştiriler getirerek, dağıtımı ilgilendiren gelişmelerden haber ederek yardımcı olabileceğiniz yeni alanlar bulabilirsiniz.

3 Geliştiricinin sorumluluğu nedir?

Aslında bu sorunun cevabı üzerinde çalışmaya başladığınız konuya göre oldukça değişken olabilir. Fakat genel olarak tüm geliştiricilerin temel sorumluluklarından bahsedebiliriz. Aslında burada tarif edilen sorumluluklar temelde birden fazla geliştiricinin bir arada, uyum içerisinde çalışabilmesi için gerekli olan maddelerdir. Bu yüzden aşağıdaki maddeler genel geçer olarak kabul edilebilse de, çalışma ortamınıza ve arkadaşlarınıza alıştıkça birlikte çalışmak için bir yöntem bulabilirsiniz.

3.1 Süreklilik

Bir geliştiricinin üzerinde çalıştığı projelerde sürekli olarak çalışabilmeyi göze alması gerekiyor. Burada süreklilik ile ifade edilen 7/24 bir çalışma değil, fakat üzerinde çalışılan konunun devamlılığının sağlanması. Bir işletim sistemi projesinde parçalar birbirleri ile sıkı bir şekilde bağlı olduğu için geliştiricilerin de birbirlerinin hızlarına ayak uydurabilmeleri gerekiyor.

3.2 Doğruluk

Proje içerisinde birden fazla geliştirici ile aynı konu üzerinde çalışıyor olabilirsiniz. Bunun yanında sizin çalışmalarınızdan etkilenen başka geliştiriciler de olabilir. Bu yüzden çalışmalarınızın yalnızca sürekliliği değil, diğer geliştiricileri etkilediği oranda doğruluğu da önemlidir. Eğer çalışmanızı ana geliştirme depolarından biri üzerinde yürütüyorsanız, işin her aşamasında sizinle birlikte çalışan ve çalışmalarınızdan etkilenen diğer geliştiricileri de göz önüne almalısınız. Yaptığınız bir değişikliğin, az da olsa, diğer geliştiricilerin işlerini yapmasına engel olmamasını sağlamalısınız.

3.3 Kararlılık

Üzerinde çalıştığınız konuda eğer karar verme yetkisi sizde ise kararlarınızı çok sık bir şekilde değiştirmeniz diğer geliştiricilerin gelişimi takip etmesini, fakat daha önemlisi size bağımlı olan geliştiricilerin çalışmalarını güçleştirecektir. Bu yüzden kararlarınızı iyi düşünerek ve diğer geliştiricilerin de fikirlerini alarak vermeniz faydalı olacaktır.

Yine de bir noktada kararınızı değiştirmeniz gerektiğini hissederseniz, bunu diğer geliştiricilere de haber vererek ve eski çalışmalarınız eğer kullanılıyorsa onlara zarar vermeyerek, ayrı bir "deneysel" alanda gerçekleştirebilirsiniz.

3.4 İletişim

Aldığınız kararlardan, attığınız adımlardan küçük de olsa yaptığınız değişikliklerden diğer geliştiricileri haberdar etmek faydalı olacaktır. Böylelikle üzerinde çalıştığınız konuya (her zaman ihtiyaç duyulan) yeni geliştiricileri daha kolay adapte edebilir, birlikte çalıştığınız geliştiriciler ile daha uyumlu ve hızlı çalışabilirsiniz. Bu sayede ileride yardıma ihtiyacınız olduğunda başka bir geliştirici yaptıklarınızdan haberdar olduğu için sorununuza daha hızlı çözüm üretebilmesini de saylayabilirsiniz. İletişim için e-posta listelerini, hazırlayacağınız belgeleri veya yaptığınız değişikliklere ekleyeceğiniz açıklama bilgilerini (subversion gönderim mesajları gibi) kullanabilirsiniz.

4 Yeni geliştirici başvurusu

Yeni bir geliştiriciye her zaman yer vardır. Siz de bir Pardus geliştiricisi olmak için başvurabilir ve resmi bir geliştirici olabilirsiniz. Yeni bir geliştirici olmanın sorumluluğunu kabul etmek ön şart olmakla birlikte her zaman yeterli olmayacaktır. Fakat sorumluluk dağıtmak ve yeni geliştiricileri kabul etmek konusunda hevesli olduğumuzu açıkça belirtebiliriz. Detaylar için okumaya devam edin...

4.1 Kim başvurabilir?

Pardus için çalışmakta olan ve bu belgede tarif edilen geliştirici sorumluluklarını kabul eden herkes *yeni geliştirici başvurusu* yapabilir. Genellikle iletişim e-posta listeleri üzerinden yazılı olarak gerçekleştiriliyor; geliştiricinin başvurusunun kabul edilebilmesi için kararlılığının ve çalışma yönteminin gözlenmiş olması gerekmektedir. Bu nedenle başvurmadan önce kendiniz için uygun bir iş seçip üzerinde çalışmaya başlayabilirsiniz. Çalışmalarınızı diğer geliştiriciler ile paylaşarak yaptıklarınızın incelenmesini sağlamanız diğer geliştiricilerin sizi tanımalarını sağlayacaktır.

Örneğin, şu anki paketler deposundaki yazılımlar üzerinde çalışmak, yama ve çözüm önerileri getirmek, diğer çözüm önerilerini inceleyerek test etmek ve bir rapor halinde sunmak iyi bir başlangıç noktası olabilir.

Bunun yanında, projeye destek vermek isteyenlerin kendilerini tanıtıp özelliklerine uygun bir iş seçmek için kullanabilecekleri kalite e-posta listesi[1] bulunmaktadır.

4.2 Başvuru nasıl yapılır?

Başvuruyu gelistirici@uludag.org.tr adresine bir e-posta ile yapabilirsiniz.

Başvuru e-postanızda;

1. Pardus üzerinde çalışmakta olduğunuz geliştirme konularını
2. çalışmak istediğiniz diğer konuları
3. hata kontrol siteminde kullandığınız e-posta adresini/kullanıcı adını
4. uzmanlık alanlarınızı
5. sizin çalışmalarınızdan haberdar olan ve size referans olabilecek bir Pardus geliştiricisini

bildirmeniz gerekiyor.

Başvuru e-postasına hesabınız için kullanacağınız kullanıcı adı ve parolanızın şifrelenmiş bir halini (htpasswd ile oluşturulmuş) eklemelisiniz. Bu sayede düz metin halini sadece sizin bildiğiniz parolanızın şifrelenmiş halini bize göndermiş olursunuz ve kimlik denetimi gerektiren yerlere kullanıcı adınızı ve şifrenizi ekleyebiliriz.

4.2.1 htpasswd ile parola oluşturma

Parola ve kullanıcı adınızı htpasswd programını kullanarak oluşturabilir ve çıktı dosyasını e-postanıza ekleyebilirsiniz. Bunun için aşağıdaki komut kullanılabilir.

```
$ htpasswd -c parola_dosyasi kullanici_adi
New password:
Re-type new password:
Adding password for user kullanici_adi
```

Sonuç olarak oluşturulacak çıktı dosyasını (parola_dosyasi) e-postanıza ekleyebilirsiniz.

[1] Kalite e-posta listesine http://liste.uludag.org.tr/cgi-bin/mailman/listinfo/kalite adresinden ulaşılabilir.

4.2.2 perl ile parola oluşturmak

Bunun için aşağıdaki komutu kullanabilirsiniz.

```
perl -e "print crypt('parolanız','xy'),\"\n\";"[2]
```

Komut yalnızca parolanızı oluşturacaktır. E-postanızda kullanmak istediğiniz kullanıcı adını da belirtmeniz gerekir.

4.2.3 python ile parola oluşturmak

Aynı işlemi python ile aşağıdaki komut ile yapabilirsiniz.

```
python -c "import crypt; print crypt.crypt('parola', 'xy')"
```

Yine çıktı olarak yalnızca parolanızın gölgelenmiş hali verilecektir. E-postanızda kullanmak istediğiniz kullanıcı adını belirtmeniz gerekir.

5 Subversion depoları

Uludağ kapsamındaki geliştirme süreci Subversion sürüm kontrol sistemi üzerinden gerçekleştirilmektedir. Subversion açık kaynak kodlu bir sürüm takip sistemidir. Bir yazılım projesi üzerinde birden fazla uygulama geliştiricisinin birbirlerinin yaptıkları değişiklikleri bozma kaygısı olmadan bir arada çalışabilmelerini sağlayan bir geliştirme altyapısıdır. Bu sayede herhangi bir yazılımın gelişim süreci geriye dönük takip edilebilmekte, zaman içerisinde yapılan değişiklikler gözlenebilmekte, herhangi bir zamanki versiyona kolayca dönülebilmektedir.

Uludağ bünyesinde şu anda kullandığımız iki adet subversion deposu bulunuyor.

5.1 Uludağ Deposu

Uludağ deposu, proje içerisinde geliştirilmekte olan ürünlerin tutulduğu depodur. Pardus için geliştirilen tüm yazılımlar Uludağ deposu içerisinde tutulur.

[2] 'xy' crypt() fonksiyonuna, gölgelemede kullanılmak üzere parametre olarak verilecek olan rastgele iki karakterdir.

5.2 Paketler Deposu

Paketler deposu https://svn.uludag.org.tr/paketler adresinde hizmet vermekte ve Pardus'un ilk sürümüne girecek olan yazılımlar için, kuluçka dağıtım kurallarına uygun bir şekilde[3], temel bir Portage deposu oluşturmaktadır.

Paketler deposu şu an için yalnızca 'Kaynak Temizleme Kılavuzu' belgesinde anlatıldığı şekli ile yazılımların dağıtıma girmeden önce çözülmesi gereken sorunlarını çözmek için kullanılmaktadır.

Paketler deposu Pardus için yeni bir paket yönetim sistemi hazırlanana kadar kullanılacak, bundan sonra işlevini bu hali ile yitirecek ve büyük bir ihtimal ile yeni bir yapı oluşturulacaktır. Fakat şu anda işletim sistemine eklenecek yazılımlara dair yapılan iyileştirmeler bu depo kullanılarak yapacaktır.

5.3 Depo dizin sıra düzenleri

Her bir Uludağ subversion deposu aşağıdaki dizin sıra düzenine (hiyerarşi) sahiptir.

Depo içerisinde üç ana dizin bulunur; trunk, tags ve repos.

5.3.1 trunk/

Trunk dizini sürekli çalışmanın yapıldığı dizindir.

Her proje modülü (belge, web sayfaları, yazılım projeleri, vb.) trunk/ altında kendi dizinine sahiptir.

5.3.2 tags/

Tags dizini, her hangi bir modül için, trunk altında yapılan işlerin etiketlenerek (taglanarak) kopayalandıkları yerdir. Bu dizin de kendi içinde 3 dizin barındırır.

- **tags/RELEASE/**: Yazılımların (veya modüllerin) kendi sürüm numaralarını etiketlemek içiin kullandıkları dizindir. Örneğin, tasma'nın 0.2 sürümü, tags/RELEASE/tasma-0.2 dizininde etiketlenir.
- **tags/BLACKHOLE**/: Geliştirilmesi durdurulmuş (üzerinde çalışacak bir geliştirici olmadığı için veya artık o projeye ihtiyaç duyulmadığı için) projelerin "atıldığı" bir kara deliktir. Buradaki projeler tekrar kullanılmak istendiğinde trunk/ altına kopyalanarak üzerinde çalışılır.
- **tags/RESTRUCTURED/**: Yazılım (veya modül) tümü ile yeniden yapılanmaya girmiş ve eski dosyalar artık kullanılmayacaksa modülün atılacağı dizindir. Örneğin, yazılımına 28 Mayıs 2005'de yeniden başlanmış "abc" projesi tags/RESTRUCTURED/abc-2005-05-28/ altına taşınır.

[3] Kuluçka dağıtım tanımı ve seçim kriterleri 'Kuluçla Dağıtım Seçimi' belgesinde anlatılmaktadır.

5.3.3 repos/

Repos dizini geliştiricilerin trunk/ altında yapılan çalışmadan farklı, deneysel çalışmalarını, aynı modül üzerinde çalışan diğer geliştiricileri rahatsız etmeden yürütebilecekleri dizindir. Geliştirici repos/ altına kendi dizinini oluşturarak bu dizin içerisinde deneysel çalışmalarını yürütebilir.

(Yalnızca paketler deposu için geçerli olan bir kural olarak, repos/doc/ dizini altında bu proje ile ilgili belgeler bulunur.)

6 Subversion kullanımı

Subversion'ın çok ayrıntılı bir kullanım kitabı[4] mevcuttur, ayrıca subversion projesinin kendi sitesindeki[5] sıkça sorulan sorular sayfasından[6] da proje hakkında bilgi alınabilir. Bu kısımda pratik kullanıma ilişkin sık ihtiyaç duyulan komutlar örneklerle anlatılmaya çalışılacaktır.

6.1 Sistemimde subversion olup olmadığını nasıl anlayabilirim?

Sisteminizde subversion olup olmadığını en hızlı şekilde "**svn –version**" komutunun çıktısına bakarak öğrenebilirsiniz. Şuna benzer bir şeyler görmeniz iyiye işarettir:

```
evreniz@jaco:~$ svn --version
svn, version 1.0.3 (r9775)
  compiled May 19 2004, 21:28:49
Copyright (C) 2000-2004 CollabNet.
Subversion is open source software, see http://subversion.tigris.org/
This product includes software developed by CollabNet (http://www.Collab.Net/).
The following repository access (RA) modules are available:
* ra_dav : Module for accessing a repository via WebDAV (DeltaV) protocol.
  - handles 'http' schema
  - handles 'https' schema
* ra_local : Module for accessing a repository on local disk.
  - handles 'file' schema
* ra_svn : Module for accessing a repository using the svn network protocol.
  - handles 'svn' schema
evreniz@jaco:~$
```

Eğer yoksa, `http://subversion.tigris.org/project_packages.html` adresinden dağıtımınız ya da işletim sisteminiz için hazırlanmış olan paketi alarak sisteminize kurabilirsiniz.

[4] `http://svnbook.red-bean.com/`
[5] `http://subversion.tigris.org/`
[6] `http://subversion.tigris.org/project_faq.html`

6.2 Depo nedir?

Depo (repository), herkesin üzerinde çalıştığı yazılım(lar)ın son sürümünün, son sürümden önceki tüm sürümlerinin ve sürümler arası değişikliklerin kullanıcı, tarih ve sebep bilgileri ile beraber saklandığı ve çeşitli yöntemlerle erişilebilen bir disk alanıdır.

6.3 Depoda hangi dizinlerin olduğunu nasıl görebilirim?

Sizin de belirttiğiniz gibi, bir depo, içerisinde birden fazla dizin içerebilir. Deponun hiyerarşisi, diskimiz üzerindeki bir dizinin içi gibidir. Böylece deponun tamamını diskinize almadan gezip, sadece üzerinde çalışmak ya da göz atmak istediğiniz kısmın kopyasını diskinize alabilirsiniz. Depo içerisindeki dizinler ve dosyaların listesini "**svn ls depo_adresi**" formatı ile görüntülüyoruz:

```
$ svn ls http://svn.uludag.org.tr/uludag
repos/
tags/
trunk/
$ svn ls http://svn.uludag.org.tr/uludag/trunk
COMAR/
comar_prototip_old/
web/
$ svn ls http://svn.uludag.org.tr/uludag/trunk/COMAR
COMAR-1.sxw
COMARd/
CSL/
OM/
SlicerAPI/
confparser/
$ svn ls http://svn.uludag.org.tr/uludag/trunk/COMAR/confparser
GenericParser.py
README
branchedParser.py
comar_configparser.png
config_files/
confparser.py
flatParser.py
sectionedParser.py
$
```

6.4 Depodaki bir dizinin kopyasını nasıl alabilirim?

Deponun bir kopyasının oluşturulması esnasında "**svn co**" komutu kullanılır. Kopya oluştuktan sonra bir daha bu komut kopya üzerinde işlem yapılmaz.

```
$ svn co http://svn.uludag.org.tr/uludag
A uludag/trunk
...
...
```

Depoya bir URI olarak davranabilirsiniz. Bu şekilde depo içerisindeki herhangi bir alt dizini de alabilirsiniz.

```
$ svn co http://svn.uludag.org.tr/uludag/trunk/COMAR
A COMAR/COMAR-1.sxw
A COMAR/CSL
...
...
```

6.5 Bendeki kopyanın güncel olup olmadığını nasıl anlayabilirim?

Son değişikliklerden haberdar olmak ve son sürümü takip etmek için düzenli olarak deponun sizdeki kopyasını "**svn update**" komutu yardımı ile güncellemeniz gerekir. Komutu tek başına çağırdığınız taktirde bulunduğunuz dizinin içindeki dosyalar ve dizinlerin tamamı güncellenir. Ayrıcı komutun sonuna güncellenmesini istediğiniz dizinin ya da tek dosyanın adresini ekleyebilirsiniz.

```
~/work/uludag/[...]/uludag/trunk $ svn update
U tasma/modules/tasmanet/device.cpp
U tasma/modules/tasmanet/devicesettings.cpp
U tasma/modules/tasmanet/device.h
U tasma/modules/tasmanet/devicesettings.h
Updated to revision 158.
~/work/uludag/[...]/uludag/trunk $
```

6.6 Dosyaların yanlarındaki işaretler ne anlama geliyor?

SVN ile çalışırken, güncelleme, sorgulama gibi işlemler esnasında dosyaların yanında bir önceki örnekte de olduğu gibi kendinden sonraki dosya ile ilgili ne gibi bir değişiklik olduğunu size haber vermek içindirler.

Dosyaların yanında **U**, **D**, **A**, **C** ya da **G** harflerinden birisini görebilirsiniz:

- **A** Eklenmiş.
- **D** Silinmiş
- **U** Güncellenmiş

- **G** Birleşmiş (depodan aldığınız son güncelleme sizin yerel değişiklik yaptığınız bir dosya ile birleştirilmiş)
- **C** Çakışmış (depodan aldığınız son güncelleme sizin yerel yaptığınız değişiklikler ile çakışmış)

6.7 Bir takım dosyayı değiştirdim, şimdi ne yapacağım?

Kendi kopyanızda ne gibi değişiklikler yaptığınızı görmek istediğinizde "**svn status**" komutunu kullanabilirsiniz. Diğer tüm komutlar gibi bu da sonuna ekleyeceğiniz bir URI ile çalışabilir. Aşağıda deponun son güncel kopyasına bir dosya eklendiği, bir dosya silindiği, iki dosyanın da değiştirilmiş olduğu görünüyor:

```
~/work/[...]/trunk/COMAR/comar $ svn status
A COMARd/csl/degisiklik
D COMARd/csl/loader.py
M COMARd/COMARValue.py
M comar-call/rpc.c
~/work/[...]/trunk/COMAR/comar $ svn status COMARd/csl/COMARValue.py
M COMARd/COMARValue.py
~/work/[...]/trunk/COMAR/comar $
```

Ayrıca değiştirdiğiniz dosyalarda neyi değiştirdiğinizi de "**svn diff**" komutu ile öğrenebilirsiniz:

```
~/work/[...]/trunk/COMAR/comar $ svn diff comar-call/rpc.c
Index: comar-call/rpc.c
===================================================================
--- comar-call/rpc.c (revision 158)
+++ comar-call/rpc.c (working copy)
@@ -146,6 +146,7 @@
  if (len == 0) break;
  if (len == -1) {
          puts("baglanti erken kesildi");
+         //bambaska bir degisiklik
          break;
  }
  printf("RECV[%s]\n\n", buf);
~/work/[...]/trunk/COMAR/comar $
```

6.8 Yeni bir dosya ekledim fakat yanında "?" var...

Deponun kopyası üzerinde çalışırken yeni bir dosya yaratmak istediğinizde, "**svn add**" yardımı ile (bunun "**svn copy**", "**svn del**" gibi kardeşleri de vardır) lokal kopyanızı

bu dosyanın depoya eklenmesini istediğinize dair haberdar etmeniz gerekir. Buna neden gerek olduğunu şöyle açıklamaya çalışalım: Örneğin lokal kopyanızdaki uygulamayı derleyip denemek istiyorsunuz, bu durumda çalışma kopyanızda, asıl depoya göndermeyi istemeyeceğiniz Makefile'ler, *.m4 dosyaları gibi sadece sizi ilgilendiren dosyayalar oluşacaktır. Böyle durumlarda eklediğiniz dosyaların, depoya da eklenmemesi büyük avantaj ve kolaylık sağlar, çünkü siz yazılımın kaynak kodunu değiştirip, yeniden derleyip uygun olduğunda depoya göndermeye karar verdiğinizde diğer dosyaların depoya gitmeyeceğini bilirsiniz. "svn add" komutu ile *eklenmesini istediğiniz* dosyaları depoya eklersiniz. "**svn del**" ayrıca açıklanmayacaktır.

```
~/work/[...]/COMARd/csl/sample $ svn status
~/work/[...]/COMARd/csl/sample $ touch yenibetik.csl
~/work/[...]/COMARd/csl/sample $ svn status
? yenibetik.csl
~/work/[...]/COMARd/csl/sample $ svn add yenibetik.csl
A yenibetik.csl
~/work/[...]/COMARd/csl/sample $ svn status
A yenibetik.csl
~/work/[...]/COMARd/csl/sample $
```

6.9 Değiştirdiğim dosyalar eski hallerine dönsünler istiyorum.

Yaptığınız değişiklikleri istediğiniz an "**svn revert**" komutu yardımı ile son kopyadaki orjinaline döndürebilirsiniz:

```
~/work/[...]/trunk/COMAR/comar $ svn status
A COMARd/csl/degisiklik
D COMARd/csl/loader.py
M COMARd/COMARValue.py
M comar-call/rpc.c
~/work/[...]/trunk/COMAR/comar $ svn revert comar-call/rpc.c
Reverted 'comar-call/rpc.c'
~/work/[...]/trunk/COMAR/comar $ svn status
A COMARd/csl/degisiklik
D COMARd/csl/loader.py
M COMARd/COMARValue.py
~/work/[...]/trunk/COMAR/comar $
```

Ayrıca dosyaların tümünü özyinelemeli şekilde eski hallerine döndürmek de mümkündür...

```
~/work/[...]/trunk/COMAR/comar $ svn revert . -R
Reverted 'COMARd/csl/degisiklik'
Reverted 'COMARd/csl/loader.py'
```

```
Reverted 'COMARd/COMARValue.py'
~/work/[...]/trunk/COMAR/comar $ svn status
~/work/[...]/trunk/COMAR/comar $
```

6.10 Değiştirdiğim dosyaları göndermek istiyorum?

Değiştirdiğiniz dosyaların son hallerinden eminseniz, depoya değişikliklerinizin yansıması için "**svn commit**" komutunu kullanabilirsiniz. Bu komut ile, -diğer tüm komutlarda olduğu gibi- bir tek dosyayı, bir tek dizin ve altındakileri ya da yaptığınız tüm değişiklikleri depoya gönderebilirsiniz. svn commit dediğinizde, svn size neyi neden değiştirdiğinizi başkalarının da görebilmesi ve depoda geriye dönük izlemelerde görüntülenmek üzere loglanması için yaptığınız değişikliklerin içerisinde yazdığı bir dosyayı favori metin editörünüzde açar. Favori metin editörü olarak açtığı metin editörünü değiştirmek için svn'in kullandığı SVN_EDITOR isimli ortam değişkeninden faydalanabilirsiniz:

```
~/work/[...]/COMARd/csl/sample $ SVN_EDITOR="vi" svn commit
~/work/[...]/COMARd/csl/sample $ SVN_EDITOR="mcedit" svn commit
~/work/[...]/COMARd/csl/sample $ SVN_EDITOR="kwrite" svn commit
.
.
```

Metin editörüne değişiklikleri yazıp, yazdıklarınızı kaydedip editörü kapattığınız anda svn yerel kopyanızdaki değişiklikleri depoya göndermeye başlar.

6.11 Başka hangi komutlar var?

Subversion'ı subversion komutlarını öğrenmek için de çalıştırabilirsiniz. "**svn help**" size kullanabileceğiniz komutların bir listesini verirken, "**svn help komut_adi**" size *komut_adi* ile ilgili ayrıntılı bilgi döndürür.

```
$ svn help
usage: svn <subcommand> [options] [args]
Type "svn help <subcommand>" for help on a specific subcommand.
Most subcommands take file and/or directory arguments, recursing
on the directories. If no arguments are supplied to such a
command, it will recurse on the current directory (inclusive) by
default.
Available subcommands:
add
blame (praise, annotate, ann)
cat
```

```
checkout (co)
cleanup
commit (ci)
copy (cp)
delete (del, remove, rm)
diff (di)
export
help (?, h)
import
info
list (ls)
log
merge
mkdir
move (mv, rename, ren)
propdel (pdel, pd)
propedit (pedit, pe)
propget (pget, pg)
proplist (plist, pl)
propset (pset, ps)
resolved
revert
status (stat, st)
switch (sw)
update (up)
Subversion is a tool for version control.
For additional information, see http://subversion.tigris.org/
```

Bir Subversion komutu ile ilgili ayrıntılı bilgi almak için;

```
$ svn help add
add: Put files and directories under version control, scheduling them for
addition to repository.  They will be added in next commit. usage: add PATH...
Valid options:
--targets arg           : pass contents of file ARG as additional args
-N [--non-recursive]    : operate on single directory only
-q [--quiet]            : print as little as possible
--config-dir arg        : read user configuration files from directory ARG
--force                 : force operation to run
--auto-props            : enable automatic properties
--no-auto-props         : disable automatic properties
```

komutları kullanılabilir veya `http://subversion.tigris.org/project_faq.html` adresinden Sıklıkla Sorulan Soruları yada `http://svnbook.red-bean.com` adresinden Subversion kitabı okunabilir.

7 Subversion kullanım kuralları

Subversion deposu tüm gelişticilerin ortak olarak kullandıkları bir alandır. Bir arada çalışabilmek için geliştiricilerin depoyu etkin, doğru ve bir düzen içerisinde kullanabiliyor olmaları gerekir.

Subversion kullanım kuralları, Uludağ depolarına yazma hakkı olan geliştiricilerin uymaları gereken kurallardır.

7.1 Her zaman güncel bir depo ile çalışın

Geliştirici sayısı arttıkça subversion deposu üzerindeki güncellemeler de sıklaşacaktır. Diğer gelişmelerden haberdar olmak ve yaptıklarınızın diğer yapılanlar ile çakışmasını önlemek için çalışmaya başlamadan önce mutlaka *svn update* komutu ile deponuzu güncelleyin.

7.2 Göndermeden önce düşünün

Yaptığınız değişiklikleri Subversion deposuna göndermeden[7] önce iki defa düşünün. Depoya gönderdiğiniz veriler tüm geliştiricilere ulaşacak ve onların çalışmalarını etkileyecektir. Bu yüzden aşağıdaki maddelere uyulması büyük önem taşır.

1. Çalışmayan bir kodu subversion deposuna göndermeyin.
2. Göndermeden önce mutlaka son değişiklikleri almak için *svn update* ile deponuzu güncelleyin. Yaptığınız değişikliklerin diğerleri ile çakışmadığına emin olun.
3. Ne gönderdiğinize dikkat edin. Bunun için gönderimden önce mutlaka *svn diff* komutu ile gönderdiğiniz değişiklikleri kontrol edin.
4. Yaptığınız değişiklikleri mutlaka test edin. Hatta iki defa test edin.

7.3 Gönderimlerinize açıklayıcı mesajlar ekleyin

Gönderimlerde kullanılan açıklama mesajları yapılan değişikliğe odaklanmalı ve mümkün olduğunca açıklayıcı olmalıdır. Mümkün olduğunca yalnızca üzerinde değişiklik yaptığınız dosyalar ile ilgili açıklama mesajları eklemeye çalışın. Bununla birlikte, konu dahilinde, svn diff komutunun çıktısından elde edilemeyecek tüm bilgiyi açıklama mesajınızda anlatabilirsiniz.

Doğru bir açıklama mesajı eklememek yapılan değişikliğin anlaşılmasını güçleştirecektir.

[7] gönderim= commit. svn commit komutu ile gerçekleştirilen eylem.

7.4 Çalışma planlarına uyun

Eğer dağıtım genelinde veya üzerinde çalıştığınız bileşenin ana geliştiricisi bir çalışma/zaman planı ortaya koymuşsa gönderimlerinizde bu plana uyun.

Örneğin bir uygulama geliştiricisi belirli bir zamanda uygulamaya yeni özellik eklemeyi durdurup yalnızca bilinen hataların giderilmesi üzerinde çalışmak isteyebilir. Yaptığınız değişikliğin bu kurala uyması beklenir.

Eğer yaptığınız değişikliğin plana uyup uymadığını kestiremiyorsanız mutlaka ilgili e-posta listesine veya ana geliştiriciye başvurun.

7.5 Birden fazla bileşeni etkileyen bir değişiklik yaptıysanız tüm geliştiricileri bu değişiklikten haberdar edin.

Tüm geliştiricilerin yaptığınız “büyük” güncellemeden haberdar olmaları için konu ile ilgili e-posta listesine mutlaka bir duyuru iletisi gönderin.

7.6 Yaptığınız değişikliğin sorumluluğunu alın

Eğer yaptığınız güncelleme herhangi bir sorun çıkartıyorsa bunun sorumluluğunu alın ve çözülmesini kendiniz veya yardım alarak sağlayın.

7.7 Genel kabullere saygı duyun

Geliştirici tartışmalarında kabul edilen genel kurallara uyun ve yaptığınız değişikliğin bu kuralları bozmadığına emin olun. Emin olmadığınız durumlarda her zaman “iletişim” yolunu seçebilirsiniz.

7.8 Hata kontrol sistemindeki bir hatayı çözerken hata numarasını girin.

Eğer yaptığınız güncelleme raporlanmış bir hatayı çözüyorsa, hata kontrol sistemini depodaki güncellemeler ile senkron tutabilmek için hangi hatayı çözdüğünüzü bildirin ve daha sonra hata kontrol sistemindeki hatayı kapatın.

7.9 Kendi sorumluluğunuzda olan dosyaları güncelleyin

Yalnızca kendi sorumluluğunuzda olan dosyaları güncelleyin. Eğer bir başka geliştiricinin sorumluluğunda olan dosyalarda bir hata bulduysanız, göndermeden önce sorumlu geliştiriciye doğrudan ulaşarak veya e-posta listelerinde diğer geliştiricilerin fikirlerini alarak depoyu güncelleme yoluna gidin. Eğer sorumlu geliştirici yaptığınız değişiklikleri kabul etmez ise saygı gösterin.

7.10 Otomatik oluşturulan dosyaları depoya eklemeyin

Derleme araçlarının sonradan oluşturduğu Makefile, Makefile.in, configure betikleri, vb. dosyaları depoya eklemeyin. Bu dosyalar her geliştiricinin makinasında farklı şekillerde yeniden oluşturulacak ve diğer geliştiriciler tarafından bir güncelleme olarak algılanacaktır. Bu dosyaların depoya eklenmesi genellikle hata olarak algılanır.

7.11 Güncellemeleri atomik yapın

Bir iyileştirme/güncelleme ile ilgili tüm değişiklikleri bir anda gönderin, subversion birden fazla dosyayı aynı anda göndermenize izin verecektir. Gönderimleri ayrı ayrı yapmak diğer geliştiricilerin kafalarını karıştırabilir ve yaptığınız iyileştirmeleri kaçırmalarına neden olabilir.

Kaynaklar

[1] Metin, Barış & Onur, Çağlar (Kasım 2004). Ulusal Dağıtıma Nasıl Yardım Ederim? http://www.uludag.org.tr

[2] Metin, Barış (Kasım 2004). Paketler Deposu Yeni Geliştirici Başvurusu. http://www.uludag.org.tr

[3] Eren, A. Murat (Kasım 2004). Subversion Deposu Kullanma Kılavuzu. http://www.uludag.org.tr

[4] Barth, Andreas (2005). Debian Developer's Reference. http://www.debian.org/doc/manuals/developers-reference/

[5] Fox, Tammy & Pennington, Havoc (2003). Fedora Project Developer's Guide. http://fedora.redhat.com/participate/developers-guide/

[6] KDE (2004). Applying For a KDE CVS Account. http://developer.kde.org/documentation/misc/applycvsaccount.php

[7] KDE (2004). KDE CVS Commit Policy. http://developer.kde.org/policies/commitpolicy.html