Linux Kullanıcıları Derneği Seminerleri

RPM Paket Yöneticisi

Mayıs 2005

Devrim GÜNDÜZ
LKD, TDMSoft

http://seminer.linux.org.tr
http://www.gunduz.org
http://www.tdmsoft.com.tr

devrim@gunduz.org

2005

Giriş

Linux' un gelişmesi, kullanıcı sayısının artması ve kullanıcı profilinin değişmesi nedeniyle zaman içinde çok büyük yenilikler bu dünyaya kazandırılmıştır. Bunlardan bir tanesi de yazılımların paketlenmesidir. “tarball”lardan sonra bugün RPM, paket yönetim sistemleri içinde en çok kullanılanıdır.

RPM Red Hat tarafından Red Hat Linux için geliştirilmiş, ancak güçlü özellikleri nedeniyle bir çok Linux dağıtımı tarafından da kendilerine uyarlanmıştır ve kullanılmaktadır.

RPM, RPM Package Manager' in kısaltılmasıdır. *(Burada yanlış bir bilgiyi de düzeltmek gerekli: RPM, Red Hat Package Manager' in kısaltması* ***değildir)***. RPM temel olarak üstte de bahsedildiği gibi yazılımların paketlenmesini sağlar. RPM sayesinde sistemde kurulu olan paketler takip edilebilir. Böylece hangi paketin kurulu olduğunu ve hangi dizine hangi dosya ya da dosyaları yerleştirdiği de ileride sorgulanabilir olur. Paket bağımlılık sistemi ile bir yazılım paketinin kurulması için gereken diğer paketlerin de kurulması sağlanır. Benzer şekilde, paketlerin sistemden kaldırılması da aynı ölçüde kolaydır. Son olarak da yazılımların güncelleme aşamasında sistem yöneticilerini büyük dertlerden kurtarır. Bu yazıda bu konu ile ilgili ayrıntıları okuyabileceksiniz.

Bu yazıda “RPM” RPM Paket Yönetim Sistemini, “*rpm”* de RPM' in komut satırı aracını belirtecektir.

İpucu:

Öntanımlı olarak tüm RPM işlemlerinin yapılması için root olmak gerekmektedir. Ancak belirli durumlarda ve gerekli veritabanının hazırlanması durumunda herhangi bir kullanıcı da RPM kurabilir ya da yapılandırabilir. Bu konu daha sonra işlenecektir.

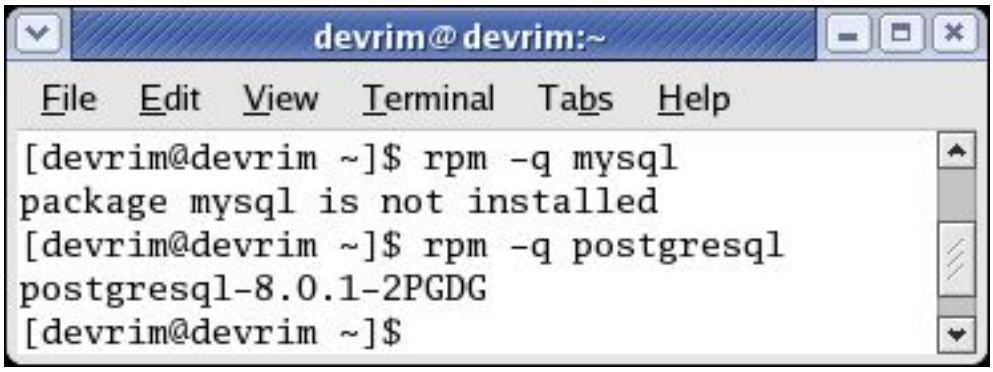
RPM ile tarball arasındaki farklar

Tarball ile RPM arasında bir çok fark bulunmaktadır. Bunlardan en önemlilerinden bir tanesi RPM' deki sürüm bilgisidir. Her RPM bir sürüm numarasına (version) *(yazılımın sürüm numarası, genelde RPM' in içindeki Açık Kaynak Kodlu yazılımın sürüm numarasıdır) ve* release bilgisine (paketin kaçıncı kez build edildiği) sahiptir.

Bir diğer fark da bağımlılık takibidir *(dependency tracking)* Paket bağımlılığı, bir paketin yapılacak işlem için *(paket oluşturulması, kurulması ya da silinmesi)* bir ya da daha fazla RPM paketi ile ilişkisinin olmasıdır. Eğer bu bağımlılıklar yerine getirilmezse, RPM kurulmayacaktır.

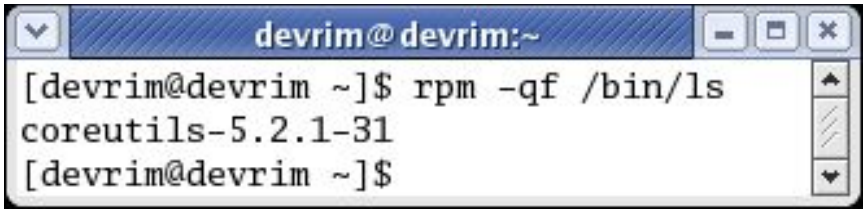
RPM dünyasına giriş : Sorgulama

RPM dünyasında yapılabilecek iş işlemlerden bir tanesi, RPM veritabanında bir paketin var olup olmadığını kontrol etmektir. Bunun için *rpm*' e -q parametresi verilir. Resim-1' e bakınız:



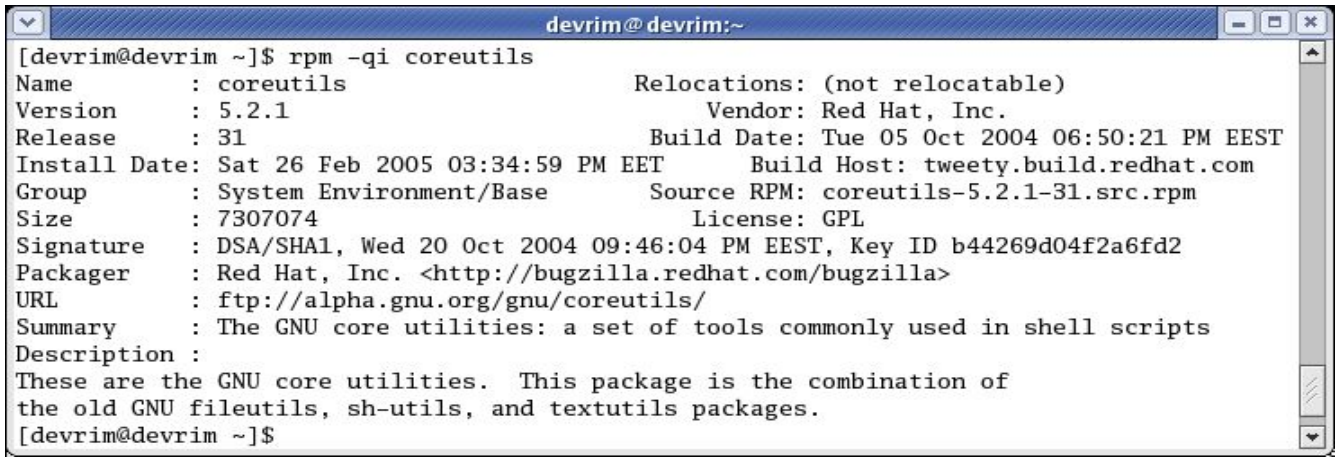
Resim 1: rpm sorgulama (-q)

RPM veritabanında sorgu yapmanın başka şekilleri de vardır. Örneğin disk üzerindeki /bin/ls dosyasının hangi paket tarafından kurulduğunu bulalım. Bunun için *rpm* komutuna -qf parametresi verilir (Şekil 2). *rpm*, ilgili dosyanın hangi paket tarafından kurulduğunu, paketin adı, sürümü ve release bilgisi ile birilikte verir.



Resim 2: Bir dosyanın hangi rpm tarafından sağlandığını sorgulama (-qf)

Peki, coreutils paketi tam olarak nedir? Bunu öğrenmek için *rpm*' e -qi parametresinin verilmesi yeterlidir. Resim 3' te bunun bir örneği verilmiştir.



Resim 3: rpm -qi örneği

Burada dikkatinizi çekmiştir; coreutils' in sürüm numarası ya da diğer bilgilerini -qi ile sorgulama yaparken kullanmadık. Eğer verdiğiniz paket adı birden fazla paketi belirtmiyorsa *(örneğin kernel)*, sürüm numarasının yazılmasına *gerek kalmaz*.

rpm' deki -q parametresine eklenebilecek parametrelerin listesi ise şöyledir:

* **-p paket_adı** : Paket dosyasının sorgulanmasını sağlar.
* **-f /sorgulanacak/dosya_adı** : Dosyanın hangi pakete ait olduğunun sorgulanmasını sağlar.
* **-a** : RPM veritabanındaki tüm paketlerin listesini çıkartır.
* **-i paket_adı** : İlgili paket hakkındaki bilgiyi verir.
* **-l paket_adı** : Verilen paketin dosyalarının listelenmesini sağlar.
* **-c paket_adı** : Verilen paket ile ilgili ayar dosyalarını listeler.
* **-d paket_adı** : Verilen paket ile ilgili belge dosyalarının listelenmesini sağlar.
* **--whatrequires** : Verilen pakete (varsa) hangi paket ya da paketlerin bağlı olduğunu listeler

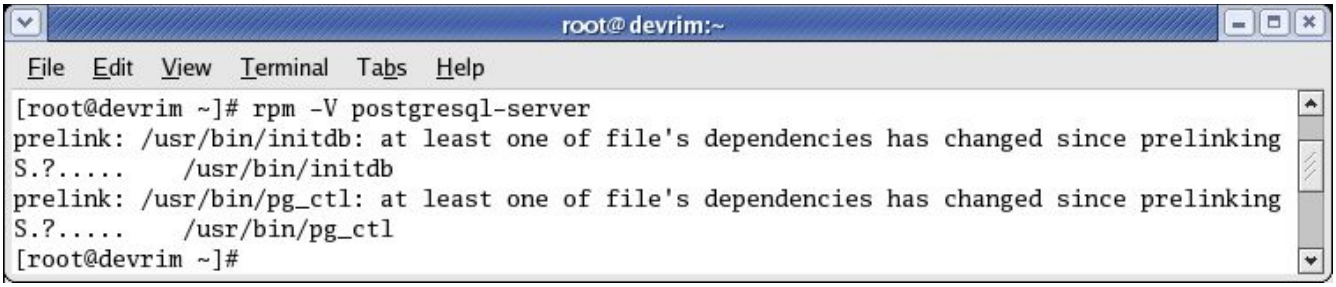
İpucu:

RPM' in kurulum için olan -i parametresi ile -q -i karıştırılmamalıdır. -q -i ile -qi aynıdır. -q' nun kullanıldığı yerde tüm işlemler sorgu amaçlı yapılır.

Paket bütünlük kontrolü

Sorgulama sadece paketin varlığı ya da sisteme kurduğu dosyalar ile sınırlı değildir. RPM' in sisteme kurulan dosyaların bilgisini sakladığını yazmıştık. Burada RPM' in bir başka üstün yönünü göreceğiz: *rpm* ile, bu dosyaların değiştirilip değiştirilmediğini de kontrol etmek mümkündür. Bu işlem basit anlamda paket silme işlemlerinde değiştirilmiş ayar dosyalarının sistem üzerinde bırakılması için kullanılır. Daha ileri seviyede düşünülürse, bir şekilde değiştirilmiş ikili (binary) dosyalar da kontrol edilebilir ve böylece güvenlik ile ilgili olası sorunlar da belirlenebilir.

Bunun için *rpm*' e -V parametresi eklenir. Sorgu için de bir paket adı verilir. Örnek bir sorgu için Resim 4' e bakınız:



Resim 4 : Paket bütünlük kontrolü

Eğer herhangi bir paketin herhangi bir dosyasının değişip değişmediği sorgulanmak isteniyorsa, *rpm*' e -Va parametresi verilir:

```
# rpm -Va
S.5....T  c /etc/pam.d/system-auth
SM5....T  c /etc/sysconfig/rhn/up2date
S.5....T  c /etc/sysconfig/rhn/up2date-uuid
...
```

Bu komutlar ile oluşan çıktının açıklaması da aşağıdaki tabloda verilmiştir:

Sıra No	Sembol	Anlamı
1	S	Dosya boyutu değişmiş
2	M	Mod ya da izinleri değişmiş
3	5	Checksum değişmiş
4	D	Aygıt majör ya da minör numaraları değişmiş
5	L	Kısayol değişmiş
6	U	Kullanıcı sahipliği değişmiş
7	G	Grup sahipliği değişmiş
8	T	Son düzenlenme tarihi değişmiş

Eğer ilgili kolonda . işareti varsa, o bilginin değişmediği belirtilmiş olur.

Yeni paket kurma

RPM Paket Yöneticisinde paket yüklemek yine *rpm* aracılığı ile yapılır. RPM için birkaç tane grafik arayüz bulunmasına rağmen, bunların hiçbirisi *rpm*' in tüm özelliklerini barındırmamaktadır. Bu nedenle, yazıda *rpm* ile ilgil işlemlere ağırlık verilecektir.

RPM kurmak için *rpm*' e -i parametresi verilir. Örnek:

rpm -i postgresql-8.0.1-2PGDG.i686.rpm

Paket adı ile ilgili ayrıntıları da bu aşamada verebiliriz. Paket adının ilk kısmı, o yazılımın adını ya da o yazılımın bir alt bölümünü belirtir:

postgresql-8.0.1-2PGDG.i686.rpm # PostgreSQL RPM' i
postgresql-libs-8.0.1-2PGDG.i686.rpm # PostgreSQL istemci programlarının gereksinim duyduğu kütüphaneleri (**lib**rarie**s**) sağlayan RPM

Paket adının arkasındaki rakamlar ise ana sürüm, alt sürüm ve yapım (build) numarasını gösterir. Üstteki örnekte, PostgreSQL 8.0.1 sürümünü kullandığımızı ve bizdeki paketin bu sürüm için yapılmış ikinci sürüm olduğunu görüyoruz. Yapım numarasının yanındaki metin ise her pakette olmayabilir ve o paket ile ilgili fazladan bilgiler verir (paketi kim yaptı, hangi özelliklere sahip gibi). Sonraki bölümde de paketin hangi mimari için özelleştirildiği belirtilir. i686, Pentium 2 ve üzerindeki mimariyi temsil eder. x86_64 ise 64-bitlik mimariyi belirtir. noarch ise o paketin herhangi bir mimari için özelleştirilmediğini belirtir. Burada olabilecek diğer alternatifler de i386, i486, i586 ve src (Source RPM)' dir.

Paket kurulurken, paketin bağımlılık listesi kontrol edilir. Paket bağımlılıkları, paketi yapan tarafından belirlenir. Bu bağımlılıklar sağlanmıyorsa o paket yüklenmez. Fedora / Red Hat sistemlerde rpmdb-{fedora||redhat} paketi yüklenirse, ilgili bağımlılığın ya da bağımlılıların hangi paket(ler) tarafından sağlanabileceği bilgisi de size verilir. Ancak bu liste içerisinde sadece dağıtım ile gelen paket listesinin bulunduğu ve 3. parti uygulamaların bu listeye dahil olmadığını anımsatmakta fayda var.

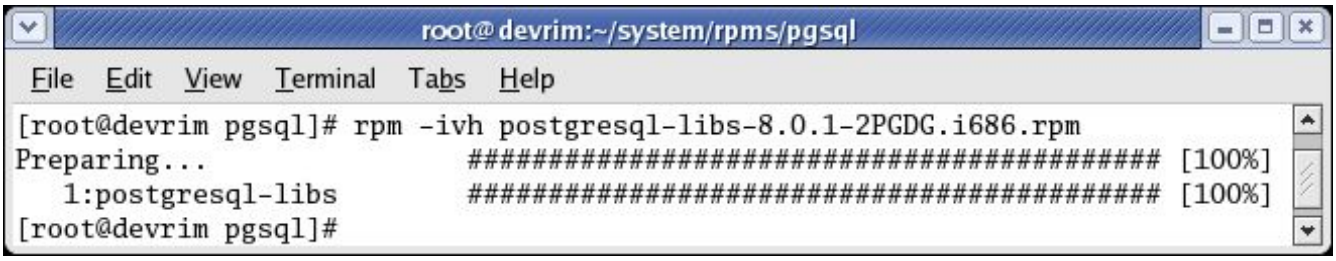
Paket kurarken -i ile birlikte kullanılabilecek diğer parametreler de şunlardır:

* **-h** : Hash marking (kurulum düzey işaretini gösterir)
* **-v** : Verbose modu
* **--test** : Kurulumu sadece deneysel olarak yapar.
* **--force** : Zorla kurulum (dosya ve paket çakışmalarında kullanılabilir)
* **--nodeps** : Dependency (bağımlılık) önemsenmeden kurulum yapılır (Bu işlemi yaparken ne yaptığınızı bilmeniz ve ne yaptığınıza emin olmanız gerekmektedir. Aksi taktirde kurduğunuz yazılım çalışmayabilir.
* **--replacefiles** : Aynı dosyalardan varsa yerine yenisini yazar.

İpucu:

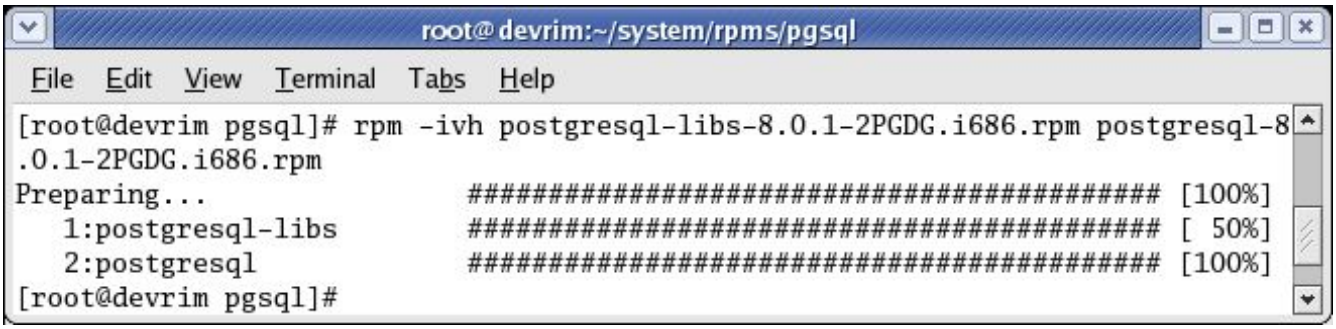
RPM, aynı adlı paketlerin aynı sürümlerinin kurulmasına izin vermez. Ancak bunun bazı istisnaları olabilir. Kernel paketi bunlardan bir tanesidir.

Örnek bir paket kurulumu için Resim-5' e bakınız. Burada, -i' nin yeterli olduğunu, -v ve -h parametrelerinin seçimsel olduğunu tekrar anımsatalım.



Resim 5 : -ivh ile paket kurulumu örneği

Aynı anda birden fazla paket kurulması mümkündür. Böyle bir gereksinim anında tüm paket isimleri ardarda yazılmalıdır. Örnek için Resim-6' ya bakabilirsiniz.



Resim 6 : Aynı anda birden fazla paket kurulumu örneği

> **İpucu:**
>
> Birbirine bağımlı (dependent) paketleriniz varsa, bunların bağımlığını çözmek için hepsini aynı anda yükleyebilirsiniz.

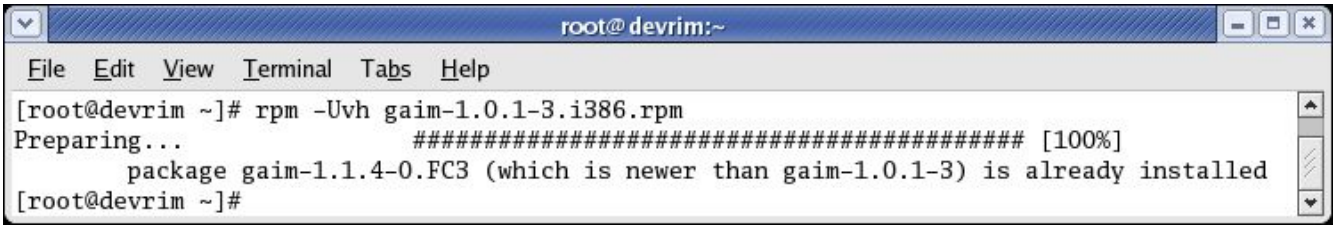
Paket güncelleme

Daha önce de bahsedildiği gibi RPM, sistem yöneticisi için yazılım yönetimini oldukça kolaylaştırır. Bir yazılımın yeni bir sürümü çıktığında yazılımın güncellenmesi çok kısa bir sürede tamamlanabilir.

Paket güncellemek için *rpm*' e verilecek temel parametre -U' dur. Bu işlem, sistemide kurulu olan yazılımın yeni bir sürümünün kurulmasını sağlar. Bu işlem sırasında öncelikle eski paket silinir ve ardından yeni paket kurulur. Paketin ayar dosyasına dokunulmaz; ancak gerekirse yeni ayar dosyası sonunda .rpmnew olacak şekilde yaratılır *(Örnek: httpd.conf.rpmnew)* .

-i parametresi ile birlikte kullanılan -v, -h ve –test parametreleri -U ile de aynı yöntemle kullanılabilir.

-U ile sürüm yükseltirken paketin sürüm numaraları kontrol edilir. Örneğin sistemde 8.0.1-1 sürümü kurulu iken 8.0.1-2 sürümüne geçiş yapılabilir. Eğer tersi bir durum söz konusu olursa, Resim 7' deki gibi bir uyarı verilir:



Resim 7: Daha yeni bir sürüm kurulu olduğu halde eski sürüme geçilmek istediğinde verilen mesaj

Ancak eğer yine de bir alt sürüme geçmek istenirse, -U' ya –oldpackage parametresi eklenebilir:

 # rpm -Uvh –oldpackage paketin.eski.sürümü

Uyarı:

Her RPM paketi için güncelleme işlemi uygun sonuç vermeyebilir. Bir yazılımı güncellemeden önce ilgili yazılımın sürüm yükseltme/azaltma yönergelerini mutlaka okuyunuz. Aksi takdirde veri kaybı yaşayabilirsiniz.

Paket kaldırmak (silmek)

Yazının girişinde de belirtildiği gibi, RPM paket yönetim sistemi sayesinde paketlerin sistemden kaldırılması da kolaydır.

Paket kaldırma sürecinde, paket kurulma sürecinin benzeri ama tersi bir süreç izlenir. Öncelikle, o pakete bağımlı olan başka paket ya da paketlerin olup olmadığı RPM veritabanından kontrol edilir. Eğer bu kontrol sonucunda veritabanında paket(ler) bulunursa bunlar kullanıcıya bildirilir. Eğer bağımlılıklarda sorun yoksa, preun *(pre-uninstall)* betiği *(script)* çalıştırılır. Örneğin bir sunucu paketi kaldırılmadan önce o sunucunun durdurulması gerekecektir. Ardından, değiştirilmiş ayar dosyaları, dosya adının sonuna .rpmsave yazılıp saklanır. Varsa postun *(post-uninstall*) betiği çalıştırılır *(örneğin, bir lib paketinin kaldırılmasından sonra ldconfig çalıştırılabilir)*Bu aşamalardan sonra paket sistemden kaldırılmış olur.

Kurulum sürecinde olduğu gibi --test parametresi ile paket silme işleminin sadece denenmesi sağlanabilir. Böylece olası sonuçlar önceden görülebilir. --nodeps parametresi de kurulum sürecindekine benzer olarak kaldıracağınız pakete olan bağımlılıkların gözardı edilmesini sağlar.

Eğer üstte belirtilen preun ve postun betiklerinin çalıştırılmaması isteniyorsa, *rpm*

komutuna –noscripts parametresi verilir.

Diğer RPM işlemleri

rpm ile yapılabilecek işlemler bunlarla sınırlı değildir. RPM' in altyapısına yönelik bazı parametreler de mevcuttur. --rebuilddb bunlardan birtanesidir. RPM veritabanının bir şekilde zarar görmesi durumunda çalıştırılabilir. --vv ile de verbose mode kullanılabilir:

```
# rpm --rebuilddb -vv
D: rebuilding database /var/lib/rpm into /var/lib/rpmrebuilddb.7984
D: creating directory /var/lib/rpmrebuilddb.7984
D: opening old database with dbapi 3
D: opening  db environment /var/lib/rpm/Packages joinenv
...
```

Bir başka parametre de --initdb'dir. Bu parametre çok dikkatli kullanılmalıdır; zira tüm RPM veritabanı ilklendirilecektir!!!

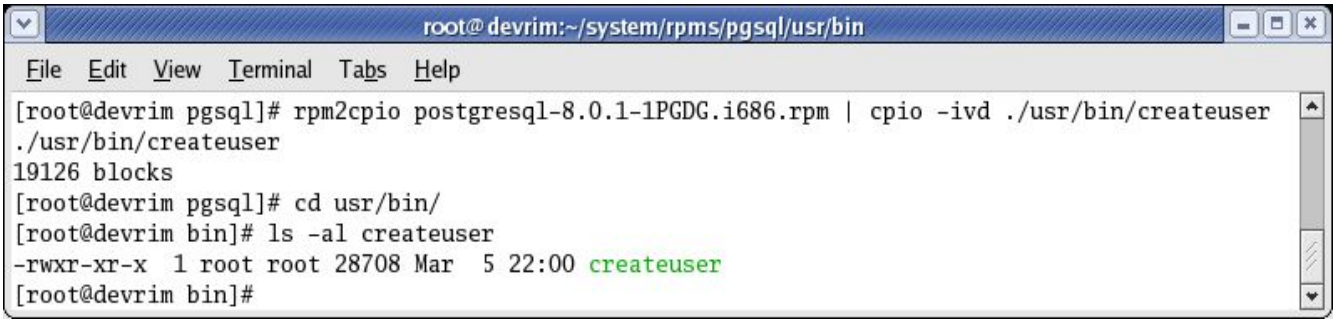
Bir RPM içindeki herhangi bir dosyayı listelemek ve o dosyayı almak

Bazı durumlarda bir RPM'in içindeki sadece bir dosyayı almak gerekebilir. Bu tür durumlarda rpm2cpio komutu imdadımıza yetişir. rpm2cpio komutu bir RPM dosyasını cpio dosyasına çevirir.

Öncelikle o RPM'in içinde hangi dosyaların olduğunu ve bunların nereye kurulacağını öğrenelim. Benzer işlemi rpm -qlp ile de yapabileceğimizi unutmayalım:

```
# rpm2cpio postgresql-8.0.1-2PGDG.i686.rpm |cpio -t
./usr/bin/clusterdb
./usr/bin/createdb
./usr/bin/createlang
./usr/bin/createuser
 ...
```

Şimdi, biz bu çıktıdaki /usr/bin/createuser dosyasını paketi kurmadan alalım. Bunun için cpio'ya farklı bir parametre vereceğiz:



Böylece “bulunduğumuz dizinin” içinde usr/bin dizini oluştu ve o dizine de createuser dosyası yerleştirildi. Bu yöntem kullanılarak istenilen dosya ya da dosyalar ilgili paket kurulmadan o paketten alınabilir.

Bu yazıda, RPM paket yönetim sisteminin kullanımında hazır paketler ile neler yapılabileceğini anlatmaya çalıştık. Bol Linux'lu günler dileğiyle.