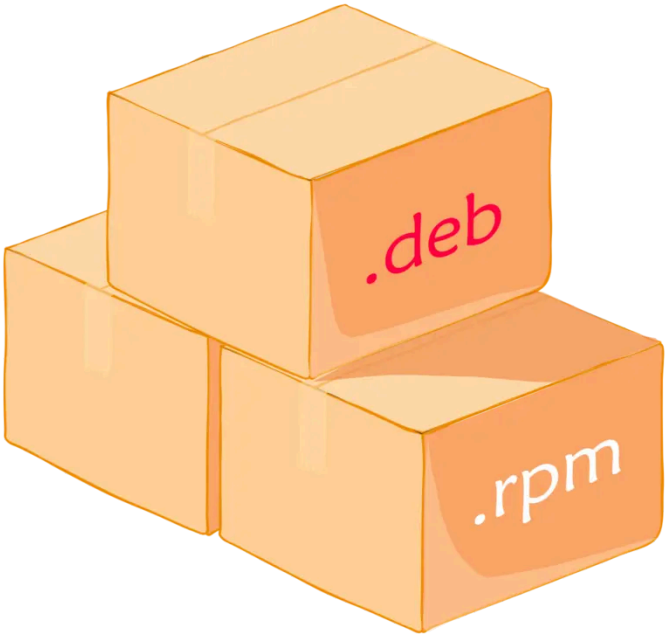

Linux Paket Yönetimi

Debian Tabanlı Sistemlerde Paket Yönetimi

> `dpkg -i <paket_adı.deb>` : Yalnızca indirmiş olduğumuz yani lokal olarak bilgisayarımızda mevcut olan ".deb" uzantılı paketlerin kurulması için `dpkg` komutunun "install" yani "kurma" anlamına gelen `i` seçeneğinin ardından kurmak istediğimiz paketin ismini girmemiz gerekir.
>
> Not : Aracın doğru şekilde çalışması için gereken ek paketler yani bağımlılıkları tek tek kurmamız gerekiyor.
>
> Kurulan paketin kurulum yerlerini detaylı görüntülemek için:

```
dpkg -L <paket_adı>
```

> `dpkg -r <paket_adı>` : Sistemimize kurmuş olduğumuz paketi silmek istersek `dpkg` aracının "remove" yani "silmek - kaldırmak" ifadesinin kısaltmasından gelen `r` seçeneği kullanılır.
>
> Not : Kaldırılan paket başka araç tarafından kullanılıyorsa hata alırız. Yine de aracın bozulması pahasına paketi kaldırmak istiyorsanız `--force-all` yani zorlama seçeneğini kullanarak `dpkg --force-all -r <paket_adı>` komutu ile ilgili paketi kaldırmaya zorlayabilirsiniz.
>
> `dpkg -P <paket_adı>` : Aracın konfigürasyon dosyaları da dahil sistemden tamamen tüm dosyalarının kaldırılmasını istersek "purge" yani "arındırmak" anlamındaki `P` seçeneği kullanılır.
>
> `dpkg -I <paket_adı.deb>` : Henüz paketi kurmadan önce paketin içeriği hakkında bilgi almak istersek (boyut, versiyon, bağımlılıkları vb...) "info" ifadesinin kısaltmasından gelen `I` karakteri kullanır.

> `dpkg -S dosya_yolu` (veya uzun haliyle `dpkg --search`) komutu, bir dosyanın hangi debian paketi tarafından kurulduğunu bulmak için kullanılır.

> **Kullanım Şekli :** `dpkg -S /dosya/yolu` **(örn :** `dpkg -S /usr/bin/firefox`**)**

> `dpkg -l` **: Sistemde yüklü bulunan tüm paketleri listelemek için "list" yani "listelemek" ifadesinin kısalmasından gelen** `l` **seçeneği kullanılır.**
>
> `dpkg -l <paket_adı>` **: Belirtilen paketin sistemde kurulu olup olmadığını sorgulamak için bu komut kullanılır.**
>
> `dpkg -l | grep <paket_adı>` **:** `grep` **komutu ile belirtilen paketin adında yada açıklamasının herhangi bir yerinde geçen paket yada paketlerin sistemde kurulu olup olmadığını sorgular.**

> `dpkg-reconfigure <paket_adı>` **: Aracı kurduktan sonra konfigürasyonları hatalı veya eksik uygulandıysa tekrar ilgili aracı baştan kurmadan yalnızca konfigürasyonların tekrar yapılmasını sağlamak, konfigürasyon dosyaları bozulmuş veya konfigürasyonu için sorulan sorulara yeniden farklı şekilde yanıt vererek yeniden konfigure etmek için kullanılır.**

APT Komutu İle Paket Listesinin Güncellenmesi

> `apt-get update` | `apt update` **: Repolardaki paketler kurulmadan evvel en güncel index bilgisini almak için kullanılır. Yani paket listesinin en güncel halini alıyoruz.**

> `apt-get upgrade` | `apt upgrade` **: Yazılım paketlerini en güncel sürümlerine yükseltmek için kullanılır. Yani paketleri güncellemek için kullanıyoruz.**
>
> **Not : Eğer amacınız tüm paketleri değil de spesifik olarak bazı paketleri güncellemek ise, güncellemek istediğiniz paketi tekrar kurmak üzere apt install <paket_adı> şeklinde komutunuzu girebilirsiniz. Bu sayede ilgili aracın en son sürümüne güncelleme yapılacaktır. Zaten** `apt` **aracı sistemde aynı isimli paket olduğunu fark edeceği için yalnızca ilgili paketi üst sürüme yükseltmeyi teklif ediyor.** `apt --only-upgrade install <paket_adı>` **komutuda sadece tek bir paketi günceller.**
>
> `apt-cache search <paket_adı>` | `apt search <paket_adı>` **: Depoda paket arama, yani bir paketi kurmadan önce ilgili paketin repoda hangi isimde tutulduğunu öğrenmek için kullanılır.**
>
> `apt-cache show <paket_adı>` | `apt show <paket_adı>` **: Paket hakkında ayrıntılı bilgi almamızı sağlar.**

> `apt-get install <paket_adı>` | `apt install <paket_adı>` **: Depo üzerinden paketin bağımlılıkları ile beraber online kurulum yapmak için kullanılır.**

> `apt-get remove <paket_adı>` | `apt remove <paket_adı>` **: Sistemimize kurmuş olduğumuz paketi kaldırmak için kullanılır.**

> **Not : Belirtilen paketin, başka bir araç tarafından kullanılmayan, artık gerek duyulmayan bağımlılıklarının da kaldırılması için `apt autoremove <paket_adı>` komutu kullanılır. Eğer bu komutun sonuna `-y` argümanını eklemiş olsaydım bana sorulmadan ilgili paket ve paket ile ilişkili artık gerekli olmayan paketler de silinmiş olacaktı.**

> **`apt-get autoremove --purge <paket_adı>` : Paketi konfigürasyon dosyaları ve kullanılmayan bağımlılıklarıda dahil tüm dosyaları sistemden tamamen kaldırma.**

> **`apt --fix-broken install` | `apt-get install -f` : APT'yi mevcut kırık paketleri düzeltmeye ve farkında olmadan bozduğumuz ya da sildiğimiz paketleri gerekirse eksik bağımlılıkları yüklemeye yönlendirir, bağımlılıkları çözülmemiş veya eksik olan paketleri belirleyip tekrar yükler.**

> **`apt-get dist-upgrade` : Komutu ile sistemde yüklü bulunan bir paketin bağımlılıkları arttıysa veya azaldıysa güncelleme yapılırken aynı zamanda varsa yeni paketlerin kurulması ve ayrıca artık gerekli olmayan paketlerin de kaldırılması mümkün oluyor.**
>
> **Not : `apt full-upgrade` komutu sayesde güncelleme esnasında bağımlılık sorunlarının ilgili paket için otomatik olarak çözülmesi sağlanıyor.**

Gereksiz Paketlerin Silinmesi

> **İndirilen paketler daha sonra tekrar kullanılma ihtimaline karşı diskte tutuluyorlar. Yani biz bir aracı kurmak için komut girdiğimizde o aracın paketi tekrar kullanılmak üzere diskte tutuluyor. Bu paketler `/var/cache/apt/archives/` dizini altında tutuluyor. Bunları silmek için de yine `apt` aracını kullanabiliriz. Eğer `apt-get clean` ya da `apt clean` komutlarını kullanırsak bu paketlerin hepsi silinmiş olacak.**

> **Eğer indirmiş olduğumuz `.deb` uzantılı paketi `apt` aracı ile kurarsak, internet bağlantımız da olduğu için `apt` aracı bu paketin bağımlılıklarını da otomatik çözümleyip kuracak. Yani lokal olarak bulunan paketleri dahi `apt` aracı ile kurabiliyoruz.**

```
apt install ~/Downloads/<paket_adı.deb>
```

> **`apt-cache depends <paket_adı>` : Paketin çalışması için gerekli olan bağımlılıkları listeler.**

> **`.deb` dosyasını kurmadan bağımlılıkların sistemde eksik olup olmadığını kontrol etmek için:**

```
sudo apt-get install -f ./<paket_adı.deb> --dry-run
```

- **`--dry-run` : Kurulum yapmaz, sadece simülasyon yapar.**
- **Eksik bağımlılıkları listeler.**

> Bağımlılık ağacını detaylı görüntülemek için aşağıdaki komut kullanılır.

```
debtree ./<paket_adı.deb>
```

> `apt list` | `apt-cache pkgnames` : Depodaki mevcut tüm paketleri listeler.
> `apt list --upgradable` : Sistemdeki güncellenebilir paketleri listeler.

> `apt download <paket_adı>` : Depodan ismi verilen paketi kurmadan indirme işlemi yapar.

Kaynak Listesi

> APT aracının doğru paketleri bulabilmesi için, APT aracının ilgili repo adreslerini biliyor olması lazım. İşte bizler bu repo adreslerini sistem üzerindeki "sources.list" yani "kaynak listesi" dosyasında belirtiliyoruz. APT aracı bu kaynak listesine bakıp sorgulama yapacağı repo adreslerini öğreniyor.

> Debian tabanlı dağıtımlarda kaynak listesi `/etc/apt` dizini altındaki `sources.list` isimli dosyadır. Bu dosyada apt aracının paketleri edinmek için hangi adreslere bakması gerektiğini belirten bağlantılar vardır. Yani repoların adresi bu `sources.list` dosyası içinde tanımlanmıştır.

Red Hat Tabanlı Dağıtımlarda Paket Yönetimi

> Debian tabanlı dağıtımlarda kullandığımız `dpkg` ve `apt` araçlarının Red Hat tabanlı dağıtımlardaki karşılığı sırasıyla `rpm` ve `yum` araçlarıdır. Debian tabanlı dağıtımlar için hazırlanmış olan paketler `.deb` uzantılı iken, Red Hat tabanlı dağıtımlar için hazırlanmış olan paketler `.rpm` uzantılıdır. `.rpm` uzantılı paketleri yönetmek için de `rpm` aracını kullanıyoruz. `rpm` aracı tıpkı `dpkg` aracı gibi paketlerin lokal olarak yönetilebilmesini sağlıyor. `yum` aracı ise tıpkı `apt` aracı gibi repolar üzerinden paketlerin ve bağımlılıkların kolayca yönetilebilmesini sağlıyor. `yum` aracı da aslında arkaplanda `rpm` aracını kullanarak repolardan paketlerin bulunması bağımlılıkların otomatik olarak çözümlenmesi gibi pek çok faydalı işlevi sunan üst seviyeli bir paket yönetim aracıdır.

> Kurulu tüm paketleri görmek için:

```
rpm -qa | less
```

> `rpm -i <paket_adı.rpm>` : Lokalde var olan `rpm` uzantılı bir paketi kurmak için kullanılır.

> Sistemde kurulu olan bir paketi kaldırmak için `rpm` komutunun `-e` seçeneğinden sonra ilgili paketin ismini girmemiz yeterli. Buradaki `e` seçeneği "erase" yani "silmek" ifadesinin kısaltmasıdır.

> Eğer işlemler hakkında detaylıca çıktı almak istersek "verbose" ifadesinin kısaltması olan `v` seçeneğini kullanabiliriz. Eğer bu seçeneği eklemezseniz araç silinir ancak herhangi bir çıktı almazsınız.

YUM ve DNF

> `yum` aracı tıpkı `apt` aracı gibi paketlerin bulunması, kurulması, bağımlılıklarının otomatik olarak çözümlenmesi, güncellenmesi, kaldırılması gibi paket yönetimi işlerini bizler için kolay hale getiren Red Hat tabanlı dağıtımlarda kullanılan kararlı yapıdaki paket yönetim aracıdır. Fakat bu aracın daha gelişmiş versiyonu olan `dnf` aracını öğrenmek daha makul bir yaklaşım olacaktır.

> Repolardaki paketlerde araştırma yapmak için `dnf search <paket-adı>` komutu kullanılır.

> Depodan paket kurmak için `dnf install <paket-adı>` şeklinde komut girebiliyoruz.

> `dnf check-update` : Sistemde kurulu paketlerin güncellemelerini kontrol etmek için kullanılır. Tüm paketleri kontrol etmek yerine dilersek `check-update` komutundan sonra paket ismi girip spesifik paket güncellemesini de kontrol edebiliriz.

> Eğer yalnızca kontrol etmek yerine güncellemelerin yüklenmesini de istiyorsak `dnf update` komutunu kullanabiliyoruz.
>
> Spesifik olarak tek bir paketi güncellemek istersek `sudo dnf install <paket-adı>` komutu ile varsa ilgili aracın güncelleştirilmesini sağlayabiliriz.

> `dnf remove paket-adı` : Paket kaldırmak için bu komut kullanılır.

> Gereksiz paketler Kurulmak üzere indirilen ve artık ihtiyaç duyulamayan paketlerin silinmesi için `sudo dnf clean all` komutunu kullanabiliyoruz.

> Alien komutu ile deb/rpm paket dönüşümü yapılabilmektedir.
> Bir yazılımın `rpm` paketi var fakat `deb` formatında paketi yoksa `alien` komutu sayesinde `rpm` paketinden `deb` paketine dönüşüm yapılabilir. Tam tersi olarak `deb` paketinden de `rpm` paketi yapılabilmektedir.

İşlem	Komut	Açıklama
`.deb` → `.rpm` dönüştürme	`sudo alien -r paket.deb` veya `sudo alien -cr paket.deb`	`-r` rpm üretir, `-c` scriptleri korur
`.rpm` → `.deb` dönüştürme	`sudo alien -d paket.rpm` veya `sudo alien -cd paket.rpm`	`-d` deb üretir, `-c` scriptleri korur

---

Kaynak Koddan Derleyerek Kurulum

> Kuracak olduğumuz yazılımın `.tar.gz` uzantılı arşiv dosyasını temin etdikten sonra dosyayı klasöre çıkarıyoruz. Burada "README" ve "INSTALL" gibi isimlerde metin dosyaları bulunuyor. İstisnalar hariç neredeyse tüm araçların kaynak kodlarında, aracın kurulumu ve konfigürasyonları ile ilgili bilgi sunan bu tür dosyalar zaten geliyor. Genel olarak kurulumu ele alıyorum ancak daha önce de söylediğim şekilde en doğru bilgiyi geliştiricinin sunduğu doküman ve `install` veya `readme` gibi dosyalardan öğrenebilirsiniz. Burada listelenen dosyalar elbette ilgili yazılıma göre değişiklik gösterir. Ancak genel olarak bilgi içeren metin dosyalarının yanında kurulum için ön ayarlamaları yapan `configure` dosyası ve kurulum işlemini kolaylaştıran genellikle `install.sh` isminde kurulum betiği ile karşılaşırsınız. Konfigürasyonlar için `configure`

> dosyasını çalıştırıyoruz. Ayrıca buradaki `makefile` dosyaları da gerekli konfigürasyon ayarlamaları yapıldıktan sonra ilgili aracın derlenip kurulması için kullanılıyor.

> - İlk olarak sıkıştırılmış dosyayı açıyoruz. Açılan klasörün içine girip, orada ilk olarak `./configure` komutu ile "configure" dosyasını çalıştırıyoruz.
>   - İlk olarak konfigürasyon dosyasını çalıştırdığımız için mevcut sistemin derleme işlemine uygun olup olmadığı kontrol ediliyor. Dolayısıyla uyumlu değilse hata çıktısında belirtilen uyarıları araştırıp çözdükten sonra derleme adımlarına devam etmelisiniz.
>   - Bu işlem sonucunda bulunulan dizinde inşa işleminin nasıl yürüyeceğini tarif eden `Makefile` adlı bir dosya oluşur.
> - `make` komutu ile derleme işlemini gerçekleştiyoruz.
>   - Burada aslında `./configure` komutu ile oluşan `Makefile` adlı dosyayı `make` adlı bir program aracılığıyla çalıştırmış oluyoruz. `make` bir sistem komutudur. Bu komutu yukarıdaki gibi parametresiz olarak çalıştırdığımızda `make` komutu, o anda içinde bulunduğumuz dizinde bir `Makefile` dosyası arar ve eğer böyle bir dosya varsa onu çalıştırır. Eğer bir önceki adımda çalıştırdığımız `./configure` komutu başarısız olduysa, dizinde bir `Makefile` dosyası oluşmayacağı için yukarıdaki `make` komutu da çalışmayacaktır. O yüzden derleme işlemi sırasında verdiğimiz komutların çıktılarını takip edip, bir sonraki aşamaya geçmeden önce komutun düzgün sonlanıp sonlanmadığından emin olmamız gerekiyor.
>   - `make` komutunun yaptığı iş, programın sisteminize kurulması esnasında sistemin çeşitli yerlerine kopyalanacak olan dosyaları inşa edip oluşturmaktır.
> - Şimdi derlenmiş olanları kurmak için `sudo make install` komutunu girmeliyiz.
>   - Kuracak olduğumuz programın eski sürümü de sistemde kalsın istiyorsak `make install` yerine `make altinstall` komutu kullanılır. `make altinstall` komutu, program kurulurken klasör ve dosyalara sürüm numarasının da eklenmesini sağlar. Böylece yeni kurduğunuz program, sistemdeki eski sürümü silip üzerine yazmamış olur ve iki farklı sürüm yan yana varolabilir. Eğer `make altinstall` yerine `make install` komutunu verirseniz sisteminizde zaten varolan eski bir sürüme ait dosya ve dizinlerin üzerine yazıp silerek o sürümü kullanılamaz hale getirebilirsiniz.
> - Kurulum için derlenmiş ama artık ihtiyaç duymadığımız dosyaları `make clean` komutu ile temizleyebiliriz.

> Kaynak koddan kurulum yaparken `--prefix` parametresiyle programı istediğin yere kurabilirsiniz:

```
./configure --prefix = $HOME/
make
sudo make install
```

Linux'ta programın sisteme nasıl kurulduğuna göre dosyalar farklı dizinlere gider.

🧩 1.Depodan (APT, DNF vs.) Kurulan Programlar:

```
sudo apt install <paket_adı>
```

🔹 **Bu programlar paket yöneticisi tarafından sistem standart dizinlerine kurulur:**

Tür	Dizin	Açıklama
Çalıştırılabilir dosyalar	`/usr/bin/` veya `/bin/`	Terminalden `program-adı` yazınca çalışan dosya burada olur
Kütüphaneler	`/usr/lib/` veya `/lib/`	Paylaşılan `.so` dosyaları
Yapılandırma dosyaları	`/etc/`	Sistem genel ayar dosyaları
Veri / kaynak dosyaları	`/usr/share/`	İkonlar, temalar, yardım dosyaları
Man sayfaları	`/usr/share/man/`	`man program-adı` komutu için içerikler

> 💡 Paket yöneticisi (ör. `apt`, `dnf`) tüm dosyaları bilir, bu yüzden kaldırmak için:

```
sudo apt remove <paket_adı>
```

📦 2. `.deb` Dosyasından Kurulan Programlar:

```
sudo dpkg -i <paket_adı.deb>
```

🔹 `.deb` paketleri de aynı dizin yapısını kullanır, çünkü `dpkg` sistemin kendi paket yöneticisidir. Yani genelde yine şu klasörler kullanılır:

- `/usr/bin/` → çalıştırılabilir dosyalar
- `/usr/share/` → ikonlar, dil dosyaları
- `/usr/lib/` → kütüphaneler
- `/etc/` → ayarlar

> 💡 `.deb` dosyası sistemde hangi dosyaları nereye koyduğunu görmek için:
>
> ```
> dpkg -c <paket_adı.deb>
> ```
>
> veya kurulduktan sonra:

```
dpkg -L <paket_adı>
```

⚙️ 3.Kaynak koddan (örneğin `./configure && make && make install` adımlarıyla) derleyip kurduğun programlar:

```
./configure
make
sudo make install
```

🔹 **Bu yöntem sistem paket yöneticisini bypass eder (haber vermez), bu yüzden nereye kurulduğunu manuel takip etmek gerekir. Varsayılan dizinler genelde:**

Tür	Dizin	Açıklama
Çalıştırılabilir dosyalar	`/usr/local/bin/`	**Sistemdeki kullanıcı yazılımları için ayrılmıştır**
Kütüphaneler	`/usr/local/lib/`	
Ayarlar	`/usr/local/etc/`	
Veri / kaynak dosyaları	`/usr/local/share/`	

🔹 **Kurulum dizininde `uninstall` varsa yani `Makefile` içinde bir `uninstall` hedefi varsa kurulan dosyaları sistemden kaldırır.**

```
sudo make uninstall
```

> 📁 Genellikle kaynak koddan derlenen programlar `/usr/local/` altına kurulur. Kurarken hangi dosyalar nereye gittiğini görmek için:

```
sudo make install > install.log
```

🧠 **Eğer program `sudo make checkinstall` komutuyla kurulduysa (yani `.deb` paketi oluşturup yükler dolayısıyla aşağıdaki komutla kaldırabilirsiniz):**

```
sudo apt remove <paket_adı>
```

> 🔹 Bu yöntem Debian/Pardus/Ubuntu tabanlı sistemlerde çok daha kontrollüdür, çünkü sistem paket yöneticisine kaydedilir.
