Pisi kullanımı

pisi

Published
with GitBook

İçindekiler

Pisi CLI kullanımı

Bu belge pisi'nin komut satırında (pisi-cli) kullanımı hakkında çeşitli bilgiler içerir.

Depo işlemleri

Depo eklemek

Depo eklemek için **add-repo** komutunu depo adı depo adresi parametreleriyle çalıştırıyoruz.

```
pisi add-repo depo_adı http://depo.adresi.com/depo/dizini/pisi-index.xml
```

'depo adresi' parametresinin alabileceği değerler:

- Yerel adresler (örn /home/groni/pisi/depom/pisi-index.xml)
- İnternet adresleri (http://groni.com/pisi-index.xml ya da ftp://groni.com/pub/pisi-index.xml)

Depo indeks dosyasının adı öntanımlı olarak **pisi-index.xml'dir**, fakat tercihe bağlı olarak aynı dosyanın sıkıştırılmış hali pisi-index.xml.xz adıyla sunulur, bu sayede bant genişliğinden tasarruf edilir.

PiSi'nin 148'inci yayımından itibaren depo eklerken dağıtım sürümü ve mimarisi kontrolü desteği gelmiştir. Özellikle bu desteği kapamak istiyorsanız --ignore-check parametresini kullanabilirsiniz.

Depo kaldırmak

Depo kaldırmak için **remove-repo** komutunu sadece bir depo adıyla birlikte çağrıyoruz,

```
pisi remove-repo depo_adı
```

eğer birden fazla depoyu birden kaldırmak istiyorsak depoların isimlerini boşlukla ayırarak yazıyoruz.

```
pisi remove-repo birinci_deponun_adı ikinci_deponun_adı
```

Depoları listelemek

Depoları listelemek için **list-repo** komutunu kullanıyoruz.

```
pisi list-repo
```

Burada aktif olan depolar yeşil, pasif olan depolarsa kırmızı renkte görülecektir.

Depoları pasifleştirmek

Sisteminizdeki depoları isterseniz kapatabilirsiniz. Kapatılan depolar işlemlerde dikkate alınmaz. Depoları kapatmak için **disable-repo** komutunu kullanıyoruz.

```
pisi disable-repo depo_adı
```

Eğer birden fazla depoyu birden kapatmak istiyorsak depoların isimlerini boşlukla ayırarak yazıyoruz.

```
pisi disable-repo birinci_deponun_adı ikinci_deponun_adı
```

Depoları etkinleştirmek

Kapattığınız depoları açmak içinse **enable-repo** komutunu kullanıyoruz.

```
pisi enable-repo depo_adı
```

Eğer birden fazla depoyu birden açmak istiyorsak depoların isimlerini boşlukla ayırarak yazıyoruz.

```
pisi enable-repo birinci_deponun_adı ikinci_deponun_adı
```

Depo güncellemek

Bir depoyu güncellemek istiyorsak **update-repo** komutunu kullanıyoruz,

```
pisi update-repo
```

Herhangi bir parametre vermezseniz etkin olan bütün depolar güncellenir. PiSi'nin normal davranışı, mevcut veritabanı depodakiyle aynıysa depodaki index dosyasını indirmemek şeklinde tasarlanmıştır. Veritabanını her halükârda güncellemek istiyorsanız --force parametresini kullanabilirsiniz. Eğer bir ya da daha fazla depoyu güncellemek istiyorsak depo isimlerini boşlukla ayırarak komutun sonuna ekliyoruz. Örneğin belirli bir depo için:

```
pisi update-repo depo_adı
```

ve birden fazla depo için:

```
pisi update-repo depo_adı diger_bir_depo_adı
```

Bileşenleri listelemek

Tüm bileşenleri listelemek için **list-components** komutunu kullanıyoruz,

```
pisi list-components
```

Çıktının bileşenler hakkında daha ayrıntılı bilgi vermesini istiyorsanız --long parametresini verebilirsiniz.

```
pisi list-components --long
```

eğer belirli bir depodaki bileşenleri listelemek istiyorsak --repository depo_adı parametresini veriyoruz.

```
pisi list-components --repository depo_adı
```

Depo indeksi oluşturmak

Eğer herhangi bir kaynak ya da ikili paket deposunun indeksini oluşturmak istiyorsak ana dizine gidip **index** komutunu veriyoruz.

```
pisi index --skip-signing
```

Bu komut bir dizinde PiSi dosyaları arar, onlardan PiSi etiketlerini çıkarır ve bilgiyi pisi-index.xml adında bir XML dosyasında toplar.

Birden fazla dizin verirseniz, komut yine çalışır, fakat herşeyi tek bir katalog dosyasına koyar.

index komutunun alabileceği parametrelerden bazıları ve açılamaları şu şekildedir:

- --absolute-urls Kataloglanan dosyalar için mutlak bağlar sakla.
- --output arg Katalog çıktı dosyası
- --compression-types arg Katalog dosyası için virgülle ayrılmış sıkıştırma türleri
- --skip-sources PiSi spec dosyalarını kataloglama.
- --skip-signing Kataloğu imzalama.

2.Paket işlemleri

Paket kurmak

Pisi paketlerini kurmak için **install** komutunu kullanıyoruz, eğer birden fazla paket kuracaksak isimlerini boşlukla ayırarak komutun sonuna ekliyoruz,

```
pisi install paketadı başka-bir-paket-adı
```

eğer yerel bir dosyadan kuracaksak dosyanın tam adını giriyoruz, birden fazla paket kuracaksak isimlerini boşlukla ayırarak yazıyoruz.

```
pisi install /home/groni/pisi_depom/paketadı.pisi
```

Halihazırda kurulu paketleri yeniden kurmak için --reinstall parametresini kullanıyoruz.

```
pisi install --reinstall paketadı
```

Dağıtım sürümü ve mimarisi kontrolünü atlayarak paket kurmak için --ignore-check parametresini kullanıyoruz.

```
pisi install --ignore-check paketadı
```

Dosya çakışmalarını gözardı ederek paket kurmak için --ignore-file-conflicts parametresini kullanıyoruz.

```
pisi install --ignore-file-conflicts paketadı
```

Paket çakışmalarını gözardı ederek paket kurmak için --ignore-package-conflicts parametresini kullanıyoruz.

```
pisi install --ignore-package-conflicts paketadı
```

Paket kaldırmak

Pisi paketlerini kaldırmak için **remove** komutunu kullanıyoruz.

```
pisi remove paketadı
```

Eğer birden fazla paket kaldırmak istiyorsak isimlerini boşlukla ayırarak komutun sonuna ekliyoruz. Ayrıca --component parametresiyle paket isimleri yerine bileşen de belirtilebilir, bu durumda bileşen altındaki tüm paketler kaldırılır. Çomar yapılandırma ajanını es geçerek paket kaldırmak için --ignore-comar parametresi kullanılabilir.

```
pisi remove --ignore-comar paketadı
```

Bağımlılık bilgilerini dikkate almadan paket kaldırmak için --ignore-dependency parametresi kullanılabilir.

```
pisi remove --ignore-dependency paketadı
```

Gerçekte hiçbir eylem gerçekleştirmeden, sadece neler olacağını görmek için --dry-run parametresi kullanılabilir.

```
pisi remove --dry-run paketadı
```

Değiştirilen ayar dosyaları dahil paketin tüm dosyalarını kaldırmak için --purge parametresi kullanılabilir.

```
pisi remove --purge paketadı
```

Emniyet mandalını yok sayarak paket kaldırmak için --ignore-safety parametresi kullanılabilir.

```
pisi remove --ignore-safety paketadı
```

Paket güncellemek

Paket güncellemek için **upgrade** komutunu kullanıyoruz, eğer birden fazla paketi güncelleyeceksek isimlerini boşlukla ayırarak komutun sonuna ekliyoruz.

```
pisi upgrade paketadi
```

Depodaki tüm güncellemeleri yapmak istiyorsak upgrade komutunu parametresiz kullanıyoruz.

```
pisi upgrade
```

Paket hakkında bilgi almak

Paket bilgilerini görüntülemek için **info** komutunu kullanıyoruz, birden fazla paketin bilgisini görmek istiyorsak isimlerini boşlukla ayırarak komutun sonuna yazıyoruz.

```
pisi info paketadı
```

Paketteki dosyaların bir listesini görmek için --files parametresini kullanıyoruz.

```
pisi info --files paketadı
```

Paketteki dosyaların sadece yollarını görmek içinse --files-path parametresini kullanıyoruz.

```
pisi info --files-path paketadı
```

Kaynaktan paket inşa etmek

Bir paketi kaynaktan inşa etmek istediğimizde **build** komutunu kullanıyoruz. Yerel veya uzak bir adresteki pspec.xml dosyasının adresinin verilmesi yeterlidir, PiSi gerekli dosyaları indirip paketi inşa edecektir. Kaynak depo kullanıyorsanız, doğrudan kaynak depoda bulunan bir

paketin adını vererek PiSi'nin o paketi inşa etmesini sağlayabilirsiniz. Örneğin yereldeki bir paketin pspec.xml dosyasını kullanarak şu şekilde paket inşası yapılabilir.

```
pisi build /home/groni/ornek/pisi/pspec.xml
```

Bir sunucudaki paket inşa edilmek isteniyorsa da şöyle bir komut kullanılabilir:

```
pisi build http://sunucu.adresi.com/paketler/pspec.xml
```

İnşa işleminin değişik safhalarını işletmek istediğinizde ya da inşa işleminin sizin kontrolünüzde gerçekleşmesini istediğinizde çeşitli parametreler kullanabilirsiniz:

- --fetch Kaynak arşivi indirdikten sonra inşayı sonlandır.
- --unpack Kaynak arşivini açtıktan, sha1sum denetimi yaptıktan ve yamaları uyguladıktan sonra inşayı sonlandır.
- --setup Yapılandırma adımından sonra inşayı sonlandır.
- --build Derleme adımından sonra inşayı sonlandır.
- --check Test adımından sonra inşayı sonlandır.
- --install Kurulum adımından sonra inşayı sonlandır.
- --package PiSi paketi oluştur. inşa seçenekleriyle ilgili bazı önemli parametrelerse şu şekildedir:
- --ignore-dependency Bağımlılık bilgilerini dikkate alma.
- --ignore-action-errors ActionsAPI kaynaklı hataları yoksay.
- --ignore-safety Emniyet mandalını yoksay.
- --ignore-check Test adımını yoksay.
- --ignore-sandbox İnşa işlemini inşa klasörüyle sınırlama.

Kaynak depodan paket kurmak

Kaynak depodaki bir paketi inşa edip kurmak istediğimizde **emerge** komutunu kullanıyoruz, bu komut paketi tüm bağımlılıklarıyla birlikte kuruyor.

```
pisi emerge paketadı
```

Bir bileşen altındaki tüm paketler de derlenip kurulabilir. Örneğin:

```
pisi emerge -c game
```

Depodaki paketleri listelemek

list-available komutu, belirtilen depolarda yayınlanan PiSi paketlerinin kısa bir listesini verir. Eğer tek bir depodaki paketleri listelemek istiyorsak list-available komutuna bir depo adını parametre olarak veriyoruz,

```
pisi list-available depo_adi
```

eğer tüm açık depolardaki paketleri listelemek istiyorsak list-available komutunu parametresiz çalıştırıyoruz.

```
pisi list-available
```

Sadece kurulu olmayan paketleri göstermek için de --uninstalled parametresini kullanıyoruz.

```
pisi list-available --uninstalled
```

Verilen bileşendeki paketleri listelemek içinse --component parametresi kullanılabilir. Örneğin:

```
pisi list-available --component game
```

Güncellemeleri listelemek

Güncellemeleri listelemek için **list-upgrades** komutunu kullanıyoruz,

```
pisi list-upgrades
```

Kurulu paketleri listelemek

Kurulu paketleri listelemek için **list-installed** komutunu kullanıyoruz.

```
pisi list-installed
```

Sadece verilen makine tarafından inşa edilmiş paketleri listelemek için --with-build-host parametresi kullanılabilir. Örneğin:

```
pisi list-installed --with-build-host localhost
```

Sadece verilen bileşen altındaki kurulu paketleri listelemek için --component parametresi kullanılabilir. Örneğin:

```
pisi list-installed -c game
```

Ayrıca --install-info parametresiyle detaylı kurulum bilgisi, --long parametresiyle de uzun biçimli çıktılar alınabilir.

Bekleyen paketleri listelemek

Yapılandırılmayı bekleyen paketleri listelemek için **list-pending** komutunu kullanıyoruz.

```
pisi list-pending
```

Kaynakları listelemek

Kullanılabilir kaynak paketleri listelemek için **list-sources** komutunu kullanıyoruz.

```
pisi list-sources
```

Kalan paketleri yapılandırmak

Eğer kurulum sırasında bazı paketlerin ÇOMAR yapılandırması atlandıysa, o paketler yapılandırılmayı bekleyen paketler listesine eklenir. ÇOMAR yapılandırması yapılmamış bu tip paketleri yapılandırmak için**configure-pending** komutunu kullanıyoruz.

```
pisi configure-pending
```

Paket aramak

Paket aramak için **search** komutunu kullanıyoruz. Bu komut; özet, açıklama ve paket ismi alanlarında belirtilen kavramı içeren paketlerin listesini çıkarır. Öntanımlı arama paket veritabanında yapılır. Kurulum veritabanında arama yapmak için --installdb parametresi ve veya kaynak veritabanında arama yapmak için --sourcedb parametreleri kullanılabilir.

```
pisi search anahtar_kelime diğer_anahtar_kelime
```

Arama sadece belli bir depoda da yapılabilir.

```
pisi search anahtar_kelime --repository depo_adı
```

Aramalar --name ile sadece paket isimlerinde, --summary ile paket özetlerinde veya --description ile paket açıklamalarında da yapılabilir.

Kurulumu denetlemek

Yüklenen her dosya için bir kontrol toplamı tutulur. **check** komutu paketin kurulumunun doğruluğunu kontrol etmek için bu toplamları kullanır. Paketlerin adlarını vermeniz yeterlidir. Eğer hiç paket adı verilmemişse, kurulu durumdaki bütün paketler doğrulanır.

```
pisi check paketadı
```

check komutu -c parametresiyle birlikte kullanılarak bir bileşendeki tüm paketlerin kontrolünü de yapılabilir.

```
pisi check -c bileşenadı
```

Diğer işlemler

PiSi veritabanlarını yeniden inşa etmek

PiSi veritabanlarını yeniden inşa etmek için **rebuild-db** komutunu kullanıyoruz.

```
pisi rebuild-db
```

Önbellek dosyalarını temizlemek

Kaynaklar, paketler ve geçici dosyalar /var dizinine kaydedilir. Bu dosyalar uzun vadede çok yer kaplayabilir. Önbellekteki bu dosyaları silmek için **delete-cache** komutunu kullanıyoruz.

```
pisi delete-cache
```

Dosya aramak

Bir dosyanın hangi pakete ait olduğunu bulmak için **search-file** komutunu kullanıyoruz.

```
pisi search-file /path/dosya
```

Paket ilişkilerinin grafiğini çıkarmak

Verilen paketlerden başlayarak, bağımlılık ve paket çakışmalarını da dikkate alarak paket ilişkilerinin grafiğini çıkarmak için **graph** komutunu kullanıyoruz. graph komutu, öntanımlı olarak depo paketleri arasındaki ilişkileri çıkarır ve dosyayı graphviz biçiminde 'pgraph.dot' dosyasına yazar.

```
pisi graph paketadi
```

Kullanılmayan kilitleri temizlemek

PiSi veritabanı erişimini yönetmek için dosya kilitleri kullanır. Kullanılmayan kilit dosyalarını veritabanı dizininden kaldırmak için **clean** komutunu kullanıyoruz.

```
pisi clean
```

Yardım almak

Komut satırında pisi kullanımı hakkında yardım almak için **help** komutunu kullanıyoruz.

```
pisi help
```

Eğer belirli bir komut için yardım almak istiyorsak şu şekilde kullanıyoruz.

```
pisi help komut
```