KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

16. Hafta – Paket Yönetimi

Yrd. Doç. Dr. A. Burak İNNER

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr

Linux'ta Yazılım Nasıl Kurulur?

- Kaynak koddan derleme
  - Yazılımı indirip, ikili dosyaları/kütüphaneleri üretmek için derlemeniz gerekiyor.
  - Genellikle, bu basit bir görevdir ve bazen birkaç püf noktası gereklidir.
  - Bazı durumlarda, bu tek seçenektir (yalnızca kaynağımız vardır).
  - Resmi olmayan sürümler kurarken yararlıdır.
  - Bazı gömülü hedeflere yüklemede kullanışlıdır (x86 olmayan işlemciler için geçerli)

- Önceden hazırlanmış bir paket yükleyin
  - Yazılım, kuruluma hazır bir pakette önceden hazırlanmış olabilir.
  - Yalnızca resmi sürümlerde geçerlidir
  - Paketler kullanılan Linux dağıtımına bağlıdır
  - Debian tabanlı dağıtımlar (Ubuntu gibi) .deb gelirler.
  - Red Hat tabanlı dağıtımlar (Fedora gibi) içeri girer. rpm

Kaynak Kod Oluşturma

Yazılımı Yükleme Kaynak kodu

- Kaynak kodunu indirin.
- Bir zip veya rar dosyasının basit bir şekilde indirilmesi, ardından kaynak kodunun ayıklanması
- Bazı yazılım yapılandırma aracı aracılığıyla indirebilirsin (svn, git, ...)
- Paketi oluşturup kurmak için verilen talimatlar için kaynak koduyla birlikte verilen readme dosyasını kontrol edin.
- Readme dosyası, kaynağı oluşturmadan önce yapılması gereken ayarlamaları ana hatlarıyla belirtmelidir.
- Ayrıca, bu kodun çalıştırılması için gereken bağımlılıkların da ana hatlarıyla belirtilmesi gerekir.

CM Yazılım Araçlarını Kullanma

- Kaynak kodu normal olarak bazı Yapılandırma Yönetim Aracı'nda saklanır
- Bu araçlar değişiklikleri ve düzeltmeleri korumak için kullanılır ve
- Birden çok kullanıcı tarafından erişim
- En sık kullanılan araçlar git, SVN (Subversion)
- Bazzar, Mercurial, vb. Gibi başka araçlar var.
- Açık Kaynak Kod Barındırma Ortak Siteler
  - Projeler,
    - GitHub (https://github.com/)
    - Google Code (https://code.google.com/)

- Bu araçlardan kaynak kodunu indirmek için özel komutlara ihtiyacınız vardır.
- $ Git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
- $ Svn checkout http://svn.wikimedia.org/svnroot/mediawiki/branches/REL1_23/phase3

Kaynak Koddan Yazılım Yükleme

- Tipik adımlar şunlardır:
  - Yapılandırma değişiklikleri yapın
  - Makefile'yi içeren uygun dizine gidin
  - Yapılandırmaların yürürlüğe girmesi için çalıştırın,

    $ Make config
  - Kodu şu yolla oluşturun:

    $ make
  - İkili dosyaları ve kitaplıkları ağaç hiyerarşisinde uygun yerlere yükleyin,

    $ Sudo make install

Not:

- Bu adımları uygulayabilmek için, takım zincirinin düzgün bir şekilde kurulmuş olması ve kurulum yapması gerekir.
- Yüklü ikili, işleminde bazı paylaşılan kitaplıklara ihtiyaç duyduğundan hala çalışmayabilir.

Paylaşılan Kütüphaneler Nedir?

- Normalde programın işlevselliği şunlardadır:
  - Programın kaynak kodu (bu programa özgü işlevsellik)
  - Önceden derlenmiş bazı Kütüphaneler (yazdırma, dosyalara yazma gibi birden fazla program tarafından kullanılan işlevler...)

- Programın görevini yerine getirmesi için bazı kütüphanelere bağlanması gerekir.
  - Bu bağlantı Statik olabilir
  - Dinamik olabilir (Çalışma zamanında, program kitaplığa bağlıdır)

Statik Link vs Dinamik Link

- Paylaşılan Kütüphanelerde neden dinamik link kullanıyoruz ??
  - İkili görüntü boyutunu daha küçük tutmak için
  - Birden fazla program aynı kütüphaneyi kullanıyor olabileceği için her biri için dahil etmek mantıklı değildir.
  - Ayrıca, bellekte yüklü farklı programlar daha az bellek kaplar.
  - Kütüphanenin işlevselliği ve hata tespitinde yükseltmek için bu kütüphaneyi kullanarak tüm programların yeniden oluşturulmasını gerektirir.

- Neden Statik link kullanıyoruz ??
  - Bağımlılıkları kaldırmak için (programı çalıştırılması gereken her şeye sahiptir)
  - Kitaplığın belirli bir sürümünü kullandığımızdan (çakışmaları önlemek için) emin olmak için...

Dinamik Bağlantı Nasıl Olur?

- Program çalıştırıldığında, program ikili dinamik bağlantı linki kontrol edilir.(ld.so)
- Dinamik bağlayıcı, paylaşılan kitaplıklara bağımlılık olup olmadığını kontrol edilir.
- Ayrıca, bu bağımlılıkları, gerekli kitaplıkların yerini belirleyerek ve onu program ikili dosyasına bağlayarak çözmeye çalışır
- Dinamik bağlayıcı bir kitaplık bulamazsa, programı çalıştırma da başarısız olur.

Dinamik linker gerekli kütüphanleri nasıl buluyor?

Dinamik Linker Kütüphaneleri Bulur

- **Kütüphane Yolu ile Ortam Değişkeni:**
  - LD_LIBRARY_PATH çevre değişkenini denetler (kitaplıkları içeren iki nokta üst üste işaretiyle “:” ayrılır)

- **Dinamik Linker Önbelleği:**

  - Dinamik linker önbellek ikili bir dosyadır (/etc/ld.so.cache)
  - Kütüphanelerin bir dizini ve konumlarını içerir
  - Dinamik linker için belirtilen dizinlerde kitaplıkları bulmak için hızlı bir yöntem sağlar.
  - Dinamik linker önbelleklerine bir dizin eklemek için;
    - /etc/ld.so.conf dosyasına dizin yolunu ekleyin
    - İkili önbellek oluşturmak için yardımcı programı ldconfig çalıştırın

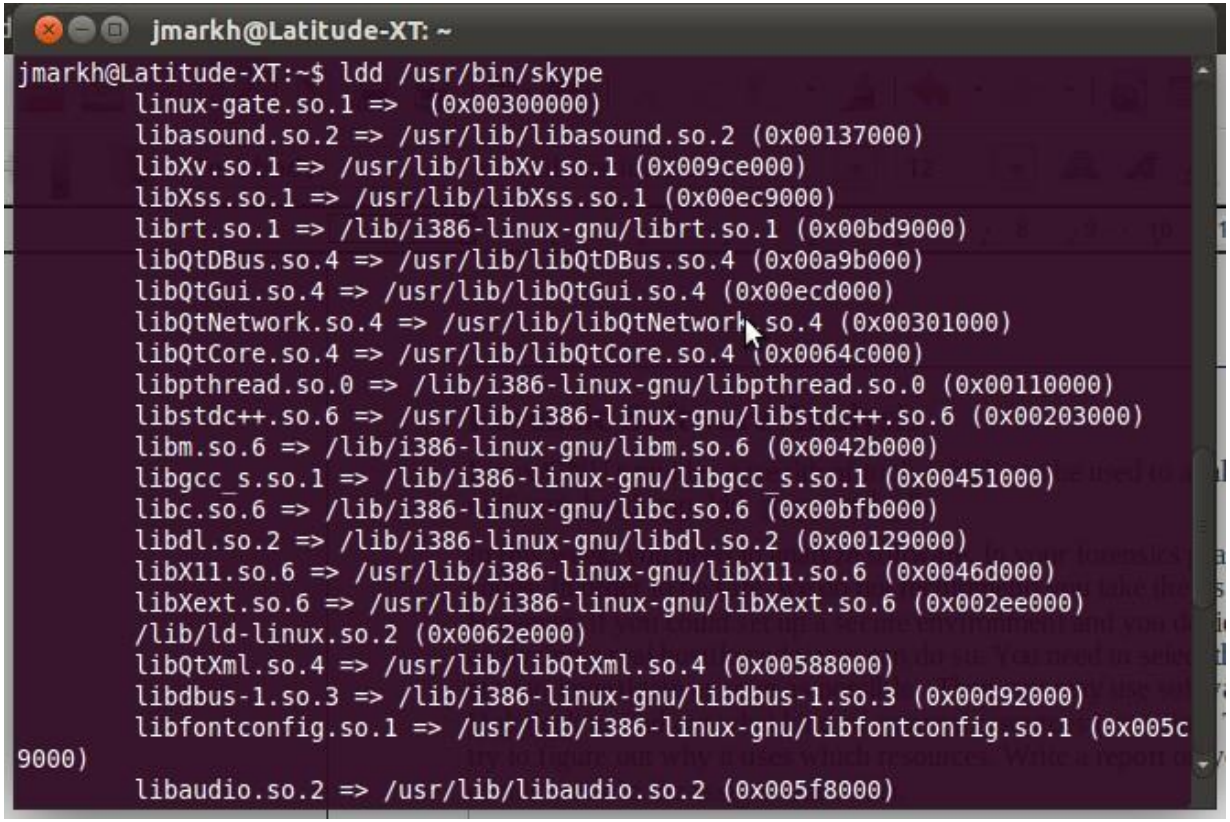
EKBİLGİ: Önbellek çalışma zamanı bağlayıcı, tarafından kullanılan *ld.so* veya *ld-linux.so* . **Ldconfig** , hangi sürümlerin güncelleneceğini belirlerken karşılaştığı kitaplıkların başlıklarını ve dosya adlarını kontrol eder.

Gerekli Kütüphaneleri Bulma (Ldd Komutu)

Komut için birden fazla program belirtilebilir

$ Ldd <program>

Bu komut, bu program için gerekli kitaplıkları (konumlarıyla birlikte) listeler



Dinamik Linker Önbellek Dosyasını Yönetme (Ldconfig Komutu)

$ Ldconfig

$ Ldconfig <Kitaplık Dizinleri>

- Bu komut, içeriği görüntülemek veya Dinamik Linker Önbellek dosyasını (/etc/ld.so.cache) oluşturmak için kullanılır.
- Önbellek içeriğini görüntülemek için

  $ Ldconfig -p
- /etc/ld.so.conf dosyasından önbellek oluşturmak için (ek olarak
- / Lib ve / usr / lib)
- $ Ldconfig
- Yukarıdaki gibi önbellek oluşturmak için, / usr / local / lib eklenerek,
- $ Ldconfig / usr / yerel / lib

Hazırlanmış Yazılım Paketi Kullanma

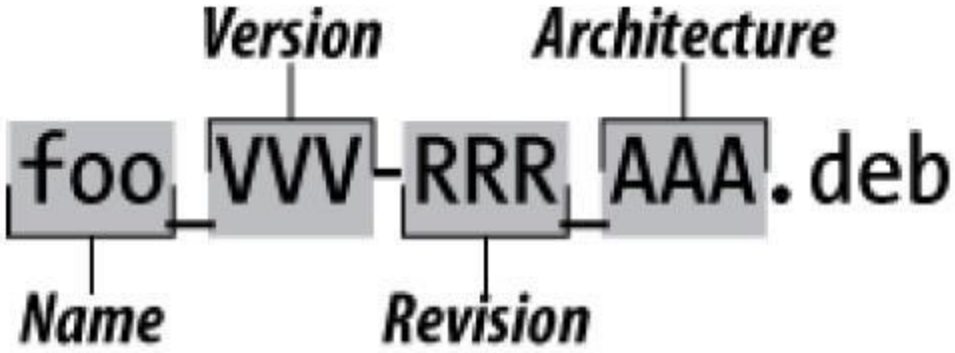
Yazılım Paketleri

- Bir Yazılım Paketi, dosyaların bulunduğu bir arşivdir;
  - Yüklenecek ikili dosyaları(önceden oluşturulmuş)
  - Uygulama için gereken yapılandırma dosyalarını
  - Pakete ilişkin meta veriler i(bağımlılıkların bir listesi de dahil olmak üzere)
  - Kurulum öncesi / sonrası betiklerini içerir

- Paketler hazırdır,
  - Bireysel olarak dahili sitelerde (.deb uzantılı bir paket dosyası veya .rpm)
  - Ortak depolar içindeki bir grup arasında (paketlerin toplanması) bulunabilir.

- Araçlar ve paket biçimi Linux dağıtımına bağımlıdır (bu dökümanda Debian tabanlı dağıtımlara odaklanacağız)

Debian Paketi Dosya Adı Biçimi



- Paket adı normalde kısa çizgilerle ayrılmış kelimeleri içerir
- Paketin versiyonu normal olarak ayrılmış 3 rakamdan oluşur
- Nokta ile, **major.minor.patch biçiminde**
- Mimari, normalde, bu paketin hedeflediği işlemci türlerini belirtir.

- Örnekler:
  gedit-common_3.10.4-0ubuntu4_i386.deb
  gnome-kullanıcı guide_3.8.2-1_all.deb
  Libfile kopye özyinelemeli-perl_0.38-1_all.deb

.deb Paket Dosyası Nasıl Yüklenir (Dpkg Komutu)

$ Dpkg -i <paket dosyası>
$ Dpkg -r <paket adı>

- .deb dosyasını kurmak için dpkg aracını kullanırız.
- " my-package " adlı bir paketimiz varsa ve pakette dosya adı my-package_1.8.0_i386.deb
- Paket dosyasını indirirsek, şu şekilde kurabiliriz:

  $ Sudo dpkg -i *my-package_1.8.0_i386.deb*
- Paketi daha sonra kaldırmayı kararlaştırırsak,

  $ Sudo dpkg -r *my-package*

Paket Bilgileri Gösteriliyor

- Yüklü tüm paketleri listelemek için

  $ Dpkg –list

- Bir paketin içindeki tüm dosyaları göstermek için

  $ Dpkg -L <paket adı>

- Bir paketin yüklenip yüklenmediğini belirlemek için

  $ Dpkg –status <paket adı>

- Hangi paketin yüklendiğini öğrenmek için

  $ Dpkg –arama <dosya adı>

- Dpkg'nin bilgilerini dizinde bulunan dosyalarda sakladığını unutmayın

  / Var / lib / dpkg

- Örneğin,

/ Var / lib / dpkg / available Kullanılabilir paketlerin listesi

/ Var / lib / dpkg / status Paketlerin durumu (kurulu, kaldırma işaretli, ..)

Harika.... Peki catch nedir???

- Gösterdiğimiz gibi, dpkg aracı paket dosyasını yüklemekle ilgilenir.

- Ancak, bir sorun var,
  - Bir sürü paket bağımlılıkları vardır, paket A'nın çalışması için belirli bir revizyona sahip B paketinin kurulması gerekir
  - Kullanıcı, bağımlılıkları bilmek ve istenen paketi yüklemeden önce gerekli ön koşulları yerine getirmelidir.
  - Bu, işlemi çok karmaşık hale getirir ve çok hata eğilimli olur

- Bağımlılık çözümüne önem veren yüksek seviyeli bir araca ihtiyacımız vardır.
  - Gerekli bağımlılıkları sorgulayabilmelidir
  - Ardından gerekli kurulumları gerçekleştirmelidir
  - Sonra istenilen paketi yükleyebilir
  - Kullanıcı müdahalesi olmaksızın tüm bunlar yapılabilmelidir.

- Bu, "Gelişmiş Paketleme Aracı" (Advanced Packaging Tool)veya apt aracıyla sonuçlandı

"Apt" nedir?

- " Apt " yazılımı yüklemek için yüksek seviyeli araçlar kümesidir
- Debian tabanlı Linux dağıtımlarında paketler
- Birkaç araçtan oluşur;
  $ Apt-get
  $ Apt-cache
- Kullanıcıdan çok fazla bilgi almasına gerek yoktur,
  - Kullanıcının paket dosyasını indirmesi gerekmez
  - Kullanıcı, paket dosya adını veya yayın numarası bilgisine bile ihtiyaç duymaz
  - Tüm kullanıcıların bilmeleri gereken "paket adı" yeterli bilgidir
  - Kullanıcının paket bağımlılıklarını bilmesine ya da onlar için tatmin edecek herhangi bir şey yapmasına gerek yoktur
  - Tümü kullanıcı için şeffaf bir şekilde yapılır

Apt kullanarak paketleri yükleme (Apt-get install Komutu)

- Belirli bir programı veya kütüphaneyi kurmak paketin adı yeterlidir.
- Bunu bilmiyorsanız, paketin adını webte arayarak kurabilirsiniz.

  $ Sudo apt-get install <paket adı>

- Apt aracı daha sonra gerekli tüm adımları yerine getirir
- İçrerik olarak;
  - Yüklenecek paketin en son sürümünü tanımlar
  - Paket için ön koşulları tanımlar (her birinin gereken sürüm numarası ile birlikte)
  - Paketin kurulumu için ne kadar disk alanı gerektiğini hesaplar
  - Kullanıcının kurulumu tamamlanmadan onaylamasını ister.
  - Kullanıcı yüklemeyi onaylarsa, araç yerel olarak makine üzerinde mevcut değilse gerekli tüm dosyaları internette indirir
  - Daha sonra kurulum prosedürü gerçekleştirilir
- Peki, apt tüm bunları nasıl yapar?

Yazılım Deposu

- Normalde paket dosyaları izole edilmiş olarak depolanmaz.
- Paketler "Paket Depoları" olarak adlandırılan büyük arşivlerde gruplandırılır.
- Depo, onları organize etmek için bir dizin dosyası ile birlikte bir paket koleksiyonudur.
- Apt aracı istenilen paketi aramak için hangi depoların izini tutar.
- Bu, bir yapılandırma dosyası /etc/apt/sources.list aracılığıyla yapılır.
- Bu yapılandırma dosyası, paketleri aramak için farklı depoları içeren sunucular için URL'lerin bir listesini içerir.

Ubuntu Deposu

- Ubuntu örneğin bir depolar setiyle birlikte geliyor
- Bu repoları içeren yansıtma sunucularının URL'leri /etc/apt/sources.list dosyasında listelenmiştir.
- Ubuntu depolarındaki yazılım paketleri **4** kategoriye ayrılmaktadır,
  - Ana: ubuntu tarafından desteklenen açık kaynaklı ücretsiz paketleri içerir ve dağıtım ile birlikte gelir.
  - Sınırlı: Ubuntu tarafından ihtiyaç duyulan mülkiyet yazılımını içerir. Buna, açık kaynaklı bir yedek yerine sahip olmayan donanım sürücüleri dahildir.
  - Evren: Kullanılabilir tüm açık kaynak yazılımını içerir. (Ubuntu tarafından desteklenmiyor)
  - Multiverse: Ubuntu tarafından desteklenmeyen tescilli yazılımlar içeriyor

/etc/apt/sources.list

```
deb http://tr.archive.ubuntu.com/ubuntu/ zesty universe
# deb-src http://tr.archive.ubuntu.com/ubuntu/ zesty universe
deb http://tr.archive.ubuntu.com/ubuntu/ zesty-updates universe
# deb-src http://tr.archive.ubuntu.com/ubuntu/ zesty-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://tr.archive.ubuntu.com/ubuntu/ zesty multiverse
# deb-src http://tr.archive.ubuntu.com/ubuntu/ zesty multiverse
deb http://tr.archive.ubuntu.com/ubuntu/ zesty-updates multiverse
# deb-src http://tr.archive.ubuntu.com/ubuntu/ zesty-updates multiverse

## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
deb http://tr.archive.ubuntu.com/ubuntu/ zesty-backports main restricted universe multiverse
# deb-src http://tr.archive.ubuntu.com/ubuntu/ zesty-backports main restricted universe multiverse

## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.
# deb http://archive.canonical.com/ubuntu zesty partner
# deb-src http://archive.canonical.com/ubuntu zesty partner

deb http://security.ubuntu.com/ubuntu zesty-security main restricted
# deb-src http://security.ubuntu.com/ubuntu zesty-security main restricted
deb http://security.ubuntu.com/ubuntu zesty-security universe
# deb-src http://security.ubuntu.com/ubuntu zesty-security universe
deb http://security.ubuntu.com/ubuntu zesty-security multiverse
# deb-src http://security.ubuntu.com/ubuntu zesty-security multiverse
linuxagyonetimi@ubuntu:/etc/apt$
```

/etc/apt/sources.list

- Apt araçları, yazılım paketlerini aramak için depoları tanımlamak için bu dosyayı kullanır.
- Kullanıcı, bu dosyayı düzenleyerek depoları ekleyebilir/kaldırabilir
- Dosya satırların bir listesini içerir
  - deb ile başlamak bu ikili paketler demektir
  - deb-src ile başlayan bu kaynak kod paketleri (geliştiriciler için yararlıdır)
  - Depo için URI'nin ardından,
    - Dağıtımın adı
    - Kullanılabilen bileşenler(ana, kısıtlanmış, evren, çok evrenli...)

- Not:
  - Dağıtım Sürümünün adını öğrenmek için aşağıdaki komutu kullanın:
    $ Lsb_release -sc
  - Örnekler:
    - Ubuntu 14.04 ismi TRUSTY TAHR (veya **TRUSTY**)
    - Ubuntu 12.04, PRECISE PANGOLIN (**PRECISE**) ismini taşır.

Değiştirme(/etc/apt/sources.list)

- Dosyayı bir metin düzenleyicisinde açarak değiştirebilirsiniz

  $ Sudo vi /etc/apt/sources.list

  - Sonra yeni depolar ekleyin
  - Eski depolar hakkında yorum yapın veya kaldırın
- Ancak dosyaya bir depo eklemek için bu komutu kullanmak (dosyanın bozulmasını önlemek için) önerilir.

  $ Sudo add-apt-repository "<dosyaya eklenecek satır>«

- Örnek

  $ Sudo add-apt-repository "deb http://us.archive.ubuntu.com/ubuntu/ hassas evren"

- Bununla birlikte, dosyanın düzenlenmesinin hemen değişikliklerle sonuçlanmadığını unutmayın
- Etkinleştikten sonra paketlerin apt veritabanını güncellemeniz gerekecek

Paket veritabanını güncelleme(apt-get güncelleme)

$ Apt-get güncelleme

- Bu komut, apt'yi, paket veritabanını yeniden oluşturmasına neden olur
- /etc/apt/sources.list dosyasına gider
  - İçeriği bulunan paketler için her depoları sorgular.
  - Her bir paket için,
    - Son sürüm numarası
    - Boyut
    - Bağımlılık listesi
- Sonra dahili veritabanını bu bilgi ile oluşturur
- Bir kurulum gerçekleştirilmesi gerektiğinde apt, iç veritabanına bakar
- Buna göre, en son paket setini kullandığımızdan emin olmak için her seferinde bir güncelleme yapılması önerilir.
- Ayrıca, liste her değiştirildiğinde bir güncelleme gerçekleştirmeliyiz.

Yüklü Paketleri Yükseltme (apt-get upgrade Command)

$ Apt-get upgrade

- Bu komut
  - İç veritabanını kontrol eder
  - Yüklü paketlerin sürüm numaraları ile veritabanındaki sürüm numaralarını karşılaştırır.
  - Yüklü paketlerden herhangi biri eski ise daha yeni versiyonu kuracaktır.
  - Yeni sürüm yeni bir bağımlılık veya çatışma içeriyorsa, yükseltmez (yeni bağımlılık yüklenmez)
- Bu komut, apt iç veritabanını kullandığından, komutu çağırmadan önce yenilemeniz önerilir
- Bu, çağrı yoluyla yapılır

$ Sudo apt-get update

Yüklü Paketleri Yükseltme (apt-get upgrade Command)

$ Apt-get dist-upgrade

- Bu iki komut biribirine çok benzer:

$ Apt-get upgrade

- Tek fark, dist-upgrade yüklü paketin yeni sürümünün yeni bir bağımlılık gerektirdiğini veya artık belirli bir bağımlılığa ihtiyaç duymadığını bulursa, değiştirilen bağımlılıkları yükleyecektir/kaldıracaktır.
- Bu, bazı paketlerin yüklü paketlerin güncellenmesi sonucunda kurulacağı veya kaldırılacağı anlamına gelir
- Gerçekten önce hangi eylemin yapılması gerektiğini bilmek istiyorsanız, önce bu komutu kullanın,

$ Sudo apt-get check

- Bu bir güncelleme gerçekleştirir ve daha sonra kopmuş bağımlılıklar için bir tanılama yapar

Paketleri Kaldırma

- Bir paketin Kurulumdan Çıkarılması ve Yapılandırma dosyalarının tutulması (gelecekteki yeniden yükleme için)

  $ Sudo apt-get remove <paket adı>

- Bir paketi kaldırın ve yapılandırma dosyalarını kaldırın

  $ Sudo apt-get temizleme <paket adı>

- Artık ihtiyaç duyulmayan paketleri kaldırmak için(bağımlılık olarak yüklenmiş ancak artık gerekli değil)

  $ Sudo apt-autoremove olsun

Paket Bilgileri Almak

- Paketleri bir anahtar kelime için aramak için
  $ Apt-önbellek arama <anahtar kelime>

- Belirli bir paket hakkında bilgi almak için
  $ Apt-cache show <paket adı>
- Sürüm, boyut, bağımlılıklar, çakışmalar gibi bilgiler

- Yüklü tüm paketlerin listesini almak için
  $ Apt-cache pkgnames

- Paketin politikasını almak için ... ana, evren, ...
  $ Apt-önbellek ilkesi <paket adı>

Paket Arşivi (/var/cache/apt/archives)

- Bu klasör, **apt** aracıyla paket dosyaları için bir önbellek olarak kullanılır
- Bir paketi kurarken, **apt**, paket dosyasını bu dizinde indirir
- Buna göre ve paket dosyaları büyük olduğundan, disk alanından kazanmak için bazen bu dosyaları silmemiz gerekiyor

$ Sudo apt-autoclean almak

- Paketler için .deb dosyalarını / var / cache / apt / arşivlerinden kaldırır.
- Artık gerekli değil

$ Sudo apt-get clean

- Tüm .deb dosyalarını / var / cache / apt / arşivlerinden kaldırır.
- Bazı paketlerin yüklenmeye ihtiyaç duyulduğunda yeniden indirilmesi gerekeceğinden bu durum istenmeyebilir

Dosya Arama

Linux ve İkili Dosyalar

$ Ls

- *Bir komut verirken, Linux yürütülecek ikili dosyayı nasıl bulur?*
- *Linux, arama yapmak için dizinlerin bulunduğu ortam değişkenini (PATH) kullanır*

Örnek:

PATH = / usr / bin: / bin: / usr / local / bin

Peki bir kullanıcı ikili bir dosyayı Linux ile aynı şekilde arayabilir mi ??

İkili Dosyalar Arama(which komutu)

$ Which <ikili dosya>

- Eğer ikili bir dosyayı çalıştırabiliyor ve bu ikili dosyanın nerede olduğunu bilmek istiyorsan

  $ which shutdown

- Bu komutun ikili dosyasını ararken **$ PATH** çevre değişkeni
  - Bu, $ PATH'ın ikili dosyanın klasörü yoksa which bunu bulamayacaktır
  - Eğer komutu çalıştıramayacaksanız which onu bulamayacaktır

Basit Arama (locate Komutu)

$ Locate <dosyaadı>
$ Locate <yolun veya dosyanın bir parçası>

- Locate komutu, sistemdeki tüm dosyaların önceden hazırlanmış bir veritabanına ve bunların tam yoluna dayalı gerekli dosyayı arar.
- Locate komutunu kullanarak arama yaparken, 'veritabanı' geçen dize stringte aranır
- Bu nedenle, yalnızca dosya yolunun bir parçası olsanız bile herhangi bir dizeye dayalı olarak arama yapabilirsiniz

$ Locate in / zi

Bu sonuçta,

/ Usr / bin / zip
/ Usr / bin / zipcloak
/ Usr / bin / zipsplit

“locate” için veritabanı

- Locate komutu tarafından kullanılan veritabanı, updatedb programı tarafından oluşturulur
- Bu program, periyodik olarak (örneğin bir kere günlük) veritabanını günceller
- Buna göre, veritabanında yanlış veya eksik bilgi olabilir,
- Yeni oluşturulan dosyalar bazen veritabanında olmayabilir
- Son zamanlarda silinen dosyalar hala veritabanında olabilir
- Yeni taşınan dosyalar, eski konumunu kullanarak veritabanında olabilir.

- Bu, yapılmış değişikliklerin aramada kaçırılabileceği anlamına gelir.

- Dizini manuel olarak güncellemek isterseniz,

  $ Sudo updatedb

Locate Komutunun Sınırları

- Locate komutu çok kullanışlı bir komuttur, Çünkü:
  - Çok basit
  - Adını, adının bir kısmını, yolunun bir bölümünü gibi temel bilgilerle bir dosyayı arayabilirsiniz.
  - Çok hızlı bir komuttur (önceden hazırlanmış bir veritabanı kullandığı için)
- Bununla birlikte, bazı kısıtlamalarla birlikte gelir,
  - Yalnızca dosya adına (veya yol adı) göre arama yapabilir
  - Yeni dosyaları bulamayabilir
  - Geçerli ağacı, arama sırasında kullanmadığı için yanlış sonuçlarla sonuçlanabilir (bunun yerine ağacın eski bir anlık görüntüsünü kullanır, çünkü veritabanında yer alan bilgileri kullanır)
- Locate komutu, sistem dosyalarını veya geleneksel Linux dosyalarını aramada kullanışlıdır ancak bir çok başka senaryoda kullanılamaz.

Daha Akıllı Bir Arama (find komutu)

- Find, farklı arama filtrelerini kullanarak dosyaları arayabilen çok zengin bir komuttur,
  Mesela;
  - Bir dosyayı adına göre arayın
  - Bir dosyayı boyutuna göre arayın
  - Son değiştirilme tarihine dayanan bir dosya arayın
  - Türüne göre bir dosya arayın (normal dosya, dizin, sembolik bağ ...)
  - Başka kriterlere göre bir dosya arama
  - Bu arama kriterlerinin bir kombinasyonu
  - Arama kapsamını belirli bir dizine (ve alt dizinlerine) sınırlayın.
  - Bazı alt dizinleri arama alanından atla

- Bu komut, arama ölçütleriyle eşleşen tüm dosyalar üzerinde bir eylem gerçekleştirme olanağı verir

Arama Kapsamını Seçme

- Find komutu, aramayı belirli kapsam dahilinde gerçekleştirir Örneğin:

  $ find ~
- Bu, giriş dizinindeki ve alt dizinindeki tüm dosyaları listeler

  $ find ~ | wc -l

- Ev dizinimin altındaki dosyaları say

  $ find /

- Sistemdeki tüm dosyaları kök dizininden başlayarak listeleyin. Tüm dosyaları arayabilmek için bu komutu **sudo** ile çalıştırmanız gerekebilir

  $ find /bin /sbin /usr/bin

- Bu 3 dizindeki tüm dosyaları listeleyin (kapsam, birden çok dizini içerebilir)

  $ find ~/Documents | grep “.pdf”

- Belgeler klasörü altındaki tüm pdf dosyalarını bulun

  $ find . > file-list.txt

- Belirtilen dosyadaki geçerli dizinin altındaki dosyaların listesini saklayın

Aramaya Filtreler Ekleme

- Şimdiye kadar find komutu, belirtilen yoldaki tüm dosyaları bulur ve bunları listeler.
- Aramayı bazı dosyalarla sınırlandırmak pipe kullanarak için sonucu grep komutuna bağlayabiliriz.
- Locate komutuna benzer bir etki sağlar (güncel olması dışında)
- Ancak arama sonuçlarına göre arama seçeneklerine daha fazlasına ihtiyacımız var.
- Dosya adı veya yolu

- Bunu yapmak için find komutuna Arama Filtreleri eklemeliyiz.

Dosya Türüne Göre Arama

$ Find <kapsam> -type <type>

Kullanıcı giriş dizininde bir dizin listesi oluşturmak için

$ Find ~ -type d

Yalnızca kullanıcı giriş dizininde bir dosya listesi oluşturmak için

$ Find ~ -type f

Sistemdeki tüm sembolik bağlantılar için bir liste oluşturmak için,

$ Find / -type l

Aygıt dosyaları için

$ find / -type c (Karakter cihazları)

$ find / -type b (blok cihazı)

Dosya ismi ile arama

$ find <scope> -name *<file name pattern>*

$ find <scope> -iname *<file name pattern>*

- Bir dosyayı adıyla aramak için

  $ find ~ -name my-file.txt

- Dosya adı için bir desen (veya normal bir ifade bile kullanabilirsiniz)

  $ find ~ -name "*.xml"

  $ find ~ -name *\.xml

- Büyük küçük harfe duyarsız arama için,

  $ find / -iname "*.html"

  $ find / -iname *\.html

Dosya Boyutuna göre Arama

$ find <scope> -size <file Size filter>

- Dosya boyutuna göre arama yapabilirsiniz

  $ find ~ -size +1M

  - \+ Daha büyük anlamına gelir
  - -daha küçük anlamına gelir

  Ne demek, tam da bu boyut

  - M, Megabayt demektir
  - G, Gigabyte anlamına gelir
  - K kilobayt anlamına gelir
  - w sözcük anlamına gelir (2 bayt)
  - c karakterleri (1 bayt)
  - b, bloklar (512 bayt) anlamına gelir; bu, herhangi bir birim geçilmezse öntanımlıdır

Ve Daha Fazlası

- Find komutu çok zengin bir komuttur ve kullanımı açıklanan senaryolarla sınırlı değildir
- Filtreler aynı zamanda oluşturma tarihi, değiştirilme tarihi, kullanıcı sahibi olma, grup sahibi olma, izinler vb.

- Örnekler:

$ find ~ -perm 777

$ find . -user root

$ sudo find / -mtime -50

$ sudo find / -atime +50

Arama Kritelerinin çoklu kullanımı

- Aşağıdaki gibi daha sofistike bir arama ölçütü elde etmek için birden fazla arama filtresi kullanabilirsiniz.
- Birden fazla koşulun birlikte "AND" olması, bu, bir maçın yalnızca tüm koşulların yerine getirilmesi durumunda gerçekleşeceği anlamına gelir.

- Birden fazla "OR" koşullarının birlikte olması, bunun anlamı
- Maç, koşullardan herhangi birinin yerine getirilmesi durumunda olur
- Herhangi bir şartın tersine çevrilmesini sağlayabilirsiniz
- Kullanarak daha karmaşık bir arama ifadesine sahip olabilirsiniz.
- parantez

Birden fazla Filtre Kullanma ("AND")

- Bir dosya eşleşmesinin tüm ölçütleri karşılaması gereken "AND" filtrelerine sahip olabilirsiniz
- Filtreleri geriye arkaya koy

- ***$ find ~ -type f -name "*.jpg" -size +10M***
  - Kullanım '**-a**'

    $ find ~ -name '*.jpg' -a -size +10M
  - Kullanım'**-and**'

    $ find ~ -name '*.jpg' -and -size +10M

Birden fazla Koşul Kullanma(ORed)

- Bir dosya eşleşmesinin herhangi bir filtreyi karşılaması gerektiği "OR" filtrelerine sahip olabilirsiniz
- Kullanım '**-o**'

  $ find ~ -name '*.jpg' -o -size +10M

  - Kullanım'**-or**'

    $ find ~ -name '*.jpg' -or -size +10M

Tersine Çevrilen Koşullar

- Arama kriterlerine ters çeviren filtrelere sahip olabilirsiniz.
- Doğru kullanım
  - ***$ find ~ \! -name "*.pdf"***
- Yanlış kullanım

  $ find ~ -not -name "*.pdf"

Karıştırma & eşleme

- Daha karmaşık bir filtre ifadesi elde etmek için şu ana kadar açıklanan filtrelerden herhangi birini karıştırıp eşleyebilirsiniz.
- İfade daha karmaşıklaştığı için parantez "( )" kullanılması gereklidir
- Parantezlerin kaçılacağını unutmayın.
- Kullanım \(expression \)
- *Örnek:*

  *$ find ~ \(-name tmp -o -name *.txt \)*

  $ find ~ \(-name tmp -o \(-not -user root\) \)

Eylemi Bul ve Gerçekleştir

- Find komutu için varsayılan eylem, arama kriterlerine uyan dosyaların listesini yazdırmaktır (dosya adı tam yol ile)
- Bununla birlikte, arama ölçütlerini karşılayan bu dosyalara uygulanacak başka bir işlem ayarlayabilirsiniz
- Adları yolla yazdır (Varsayılan eylem)**$ find ~ -type d -print**
  - Dosya Silme

    $ find ~ -name "*.pdf" -delete
  - Dosyaların ayrıntılı bir listesini yapma
  - **$ find ~ -size +10M -ls**
  - İlk maçtaki aramadan çıkın. Herhangi bir dosyanın bu ölçütleri karşılıyorsa, aramanın amacı atılırsa, bu kullanılabilir

    $ find ~ -size +100M -quit

Dosya Atlama (-prune aksiyonu)

- Find komutu, arama işlemini tüm alt dizinlerde tekrar tekrar gerçekleştirir
- Bazı arama sonuçlarını atlamamız durumunda, diğer arama filtreleriyle birlikte -prune eylemiyle bir arama filtresi kullanabiliriz.

- Örnek:
- ***$ find ~ -type l -prune -o -name "*.txt" -print***

  $ find . -path ./misc -prune -o -name '*.txt' -print

  $ find . -type d \(-path dir1 -o -path dir2 -o -path dir3 \) -prune -o -print

Kullanıcı Tanımlı İşlemler (-exec)

- Şu ana kadar, bulma için tanımlanmış bazı eylemlerle birlikte çalıştık.
- komuta
- Sonuç ile başka bir şey yapmak istersek
- Aramamız mı var?
- Aramanın sonucunu komuta argüman olarak vermeliyiz.Bu ***-exec*** komutu ile gerçekleştiriliyor
- Bu seçenekle,
- Find komutu öncelikle arama ölçütlerini karşılayan dosyaların listesini alır
- Ardından, bu dosyaları (birer birer) bir argüman olarak
- komuta
- Find komutu, her argüman için bir alt kabuk başlatır ve içindeki komutu çalıştırır

Kullanıcı Tanımlı İşlemler(-exec komutu)

- Örnek:***$ find ~ -name "*.BAK" -exec rm {} ';'***
- Not {} ve;
- {}, Dosya adını komuta nereye yerleştireceğini gösteren bir yer tutucudur.
- ; Komutunın sona ermesinin bir göstergesidir
- ; Tarafından yorumlanmasını önlemek için alıntı yapılmalıdır veya kaçılmıştır.
- kabuk
- {} Komutunda birden çok kez kullanılabileceğini unutmayın,***$ find ~ -name "*.java" -exec cp {} {}.old \;***

Kullanıcıya Bilgi İstemek(–ok)

- -exec eylemi find komutu tarafından yapılan aramadan oluşan dosya listesinde belirtilen komutu gerçekleştirir.
- Ancak, bunların hepsi kullanıcı müdahalesi olmaksızın gerçekleşir.
- Arama filtrelerini ayarlarken yapılan herhangi bir hata, yanlış dosya grubunda işlem yapılmasına neden olabilir.
- Bunu hafifletmenin bir yolu, önce aramanın sonucunu doğrulamak için -print veya -ls eylemiyle denemek, daha sonra komutu diğer daha tehlikeli komutu kullanarak uygulamaktır.
- Başka bir yol da, find komutunun, herhangi bir dosya üzerinde eylem gerçekleştirilmeden önce kullanıcıya bir evet / hayır cevabı vermesini istemesini sağlamaktır
- Bu -exec eylemi yerine -ok kullanılarak yapılır

$ find ~ -name "*.java" -ok mv {} {}.old ';'

Yürütmeyi Hızlandırma

- Find komutu, dosyaların uzun bir listesine neden olabilir
- -exec ve -ok işlemlerini tartıştığımız yol, her girişte ayrı olarak komut çalıştırılmasına neden olur.
- Komut, arama sonucunda çıkan dosyalardan birinde yürütülürken, bir alt kabuk başlatılır ve komutu onun içinde yürütülür
- Bu işlem çok işle sonuçlanabilir
- Bazı Komutlar birden fazla bağımsız değişkeni kabul eder, bu nedenle, tüm arama sonucunu komutu yerine getirmek ve sonuçlanan her dosya için komutu çalıştırmak yerine sadece bir kez çalıştırmak çok daha verimli olacaktır.
- Bunun yerine; A + ile

  $ find ~ -name "*.java" -exec rm {} '+' (Teklifinin isteğe bağlı olduğunu unutmayın)

Düzenli ifadelerin çalıştırılması (regex ve –regex seçenekleri)

$ find <scope> -regex <Düzenli ifade>

$ find <scope> -regextype posix-egrep <Genişletilmiş Düzenli İfade >

$ locate -r <Regular Expression>

$ locate --regex <Extended Regular Expression>

- *Locate ve find komutu birlikte çalıştırılabilir.*
- *find komutuyla kullanımı*

  $ find ~ -regex '.[^-_./0-9a-zA-Z].*'*
- *Locate komutuyla kullanımı*

  $ locate --regex 'bin.(bz|gz|zip)'

Kaynakça

- Ahmed ElArabawy, Linux for Embedded Systems for Arabs

Teşekkürler.

Dersin Sonu

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr/