KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

9. Hafta – Ortam Değişkenleri

Yrd. Doç. Dr. A. Burak İNNER

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

Ortam Değişkenleri Nedir? ??

- Ortam Değişkenleri, sistem hakkında bilgi depolayan değişkenlerdir.
- Verileri depolamak ve yapılandırma seçeneklerini ayarlamak için kullanılırlar.
- Linux altında kabuk ortamını özelleştirmemize olanak tanırlar.
- Bölünebilirler.
  - Sistem Ortamı Değişkenleri ve
    - Standart İsimler
    - Kabuk tarafından kullanılır
    - Kullanım için kullanıcılar tarafından daha fazla eklenebilir
  - Yerel değişkenler vardır.
    - Kullanıcı tarafından isim seçilebilir
    - Yerelden kabuğa (Çocuk programlara ve kabuğa geçirilmez)
    - Genellikle bütün başlıkları ayırmaktan kaçınılır.

Ortam Değişkenleri Kullanımı

- Ortam değişkenleri kullanım örneği:

- Renkler ve bash komut istemi gibi kabuk görünümünü ve şeklini konfigüre edilebilir
- Saat dilimi, Host adı…
- Çalıştırılabilir dosyalar veya herhangi bir dosya türü için arama yolu olarak Bazı sistem konfigürasyonları için varsayılan değerler
- Özel programlar için bazi konfigürasyon ayarları için kullanılabilir

Proses Ortamı

- Linux'un sistem için global bir ortam değişkeni kümesi tutmadığını veya depolamayacağını unutmayın.
- Çalışan her program kendi ortam ayarlarına sahip olacaktır.
- Bu, farklı işlemlerin farklı ortam ayarlarına sahip olabileceği anlamına gelir.

- Ortamdaki her çalışan işlemin ortam ayarları sistemini /proc/<pid>/environ dosyasını görüntüleyerek listeleyebilirsiniz
- Pid, Süreç Kimliğidir.

- Kabuğun bir süreç olduğunu ve bu nedenle de kendi ortam ayarlarını bulunduğunu unutmayın.
-

Peki, Süreçler Çevre Ayarlarını Nasıl Alıyor ??

Kalıtımla;

- Her işlemin başlatıldığı bir üst süreç olacaktır
- Alt süreç, üst işleminin ortam ayarlarını devralır
- Kabuk içinde başlatılan her programın (işlem) o kabuğun bir alt öğesi olduğunu unutmayın; dolayısıyla, kabuklardan başlatılan işlemler kabuk ortam ayarlarını devralır.
- Ayrıca, açılmamış bir oturum kabuğun açılmış oturum kabuğunun bir alt öğesi olduğunu ve dolayısıyla ortam ayarlarını başlangıçta devraldığını unutmayın.
- Yerel değişkenlerin alt kabuklara veya süreçlere devralınmadığını unutmayın.

Peki süreçler çevre ayarlarını nasıl alıyor?

Başlangıç komut dosyaları tarafından

Bu komut dizileri, üst öğesinden devralınan işlem ayarına eklenen bazı ortam ayarlarını içerebilir.

Bu durumu önceki konu anlatımlarında oturum açma / oturum dışı kabuk başlatma olarak ele almıştık.

Shelle giriş için :

/etc/profile, ~/.bash-profile or ***~/.bash-login*** or ***~/.profile***

- Shelle giriş yapmadan:

  /etc/.bashrc or ***/etc/bash.bashrc***

- Arayüz uygulamaları(Uygulama Arayüz üzerinden başlar.)

  ~/.xinitrc

/etc/profile

- Tüm kabuklara ve tüm kullanıcılara uygulanacak ayarları eklemek için ... / etc / profile içine koymamız gerekiyor.
- Çoğu dağıtımda, / etc / profile'yı doğrudan düzenlememeniz tercih edilir
- Bunu etkinleştirmek için / etc / profile 'nın, /etc/profile.d klasöründe * .sh uzantılı tüm komut dosyalarını kaynaklayan bir döngü vardır.
- Buna göre, yapmamız gereken tek şey, ayarlarımızı bu klasörün içine yeni bir komut dosyasına koymak ve bunu bir something.sh olarak adlandırıp çalıştırılabilir kılmaktır
- Komut dosyamız / etc / profile 'dan çağrılır ve dolayısıyla ayarlarımız giriş kabukları tarafından okunur ve login olmayan kabuklar tarafından devralınır

Peki süreçler çevre ayarlarını nasıl alıyor?

Başlarda argümanlar olarak geçerek

Bazı ortam değişkenleri ayarlarıyla bir programı çalıştırmak isterseniz

$ env VAR=VALUE Command

Veya

$ VAR=VALUE Command

- Örnekler

  $ env EDITOR=vim xterm

  $ EDITOR=vim PATH=$PATH:~/projects myScript.sh

Peki süreçler çevre ayarlarını nasıl alıyor?

Onları Manuel Olarak Ekleyerek

- Kabuğa yerel bir değişken eklemek isterseniz

  $ VARIABLE=VALUE

- Kabukta bir Ortam değişkeni eklemek isterseniz

  $ export VARIABLE=VALUE

- Değerin tırnak işaretleri ile çevrilebileceğini unutmayın
  - Genel olarak isteğe bağlı ama boşluklar içeriyorsa zorunludur!

- Örnekler:

  $ Source_Dir="/usr/share"

  $ export EDITOR=vim

  $ export Project_Dir="~/my project/docs"

"Dışa Aktar" hakkında daha fazlası

- Kabukta yerel bir değişkeni ayarlamak için

  $ My_Var=5

Bu şekilde My_Var, geçerli kabuğun herhangi bir alt öğesine veya işlemine devralınmaz.

- Bir Ortam Değişkenine Dönüştürmek İçin

  $ export My_Var

Bu şekilde My_Var, herhangi bir alt kabuğa veya geçerli kabuk işlemine devralınacak

- Sadece yerel bir değişkene dönüştürmek

  $ export -n My_Var

- Bir Ortam değişkenini sıfırlamak için

  $ export My_Var=

- Değişkeni tamamen kaldırmak için

  $ unset My_Var

Kullanılır.

Ortam Değişkenlerini Kullanma (echo Komutu)

- Şimdi bir değişken değerine erişmek için,

  $ echo $VAR_NAME

  $ echo ${VAR_NAME}

- Aşağıdakileri yapabildiğimizi unutmayın:

  $ PATH=$PATH: $HOME/projects

  $ PATH=${PATH}: ${HOME}/projects

- Bazen {} isteğe bağlıdır ve bazen zorunludur

  $ echo $HOME_and_$USER **(hatalı!..)**

  $ echo ${HOME}_and_${USER}

Ortam Değişkenlerini Kullanma (Printf Komutu)

- Printf komutu sadece echo komutu gibidir, ancak farklı bir format kullanır

  $ printf "$VARIABLE_NAME**\n**“

  $ printf "String %s" $VARIABLE_NAME

  $ printf "Signed Decimal Number %d" $VARIABLE_NAME

  $ printf "Floating Point Number %f" $VARIABLE_NAME

- Örnekler:

  $ printf "$PATH**\n**"

  $ printf "The path is set to %s**\n**" $PATH

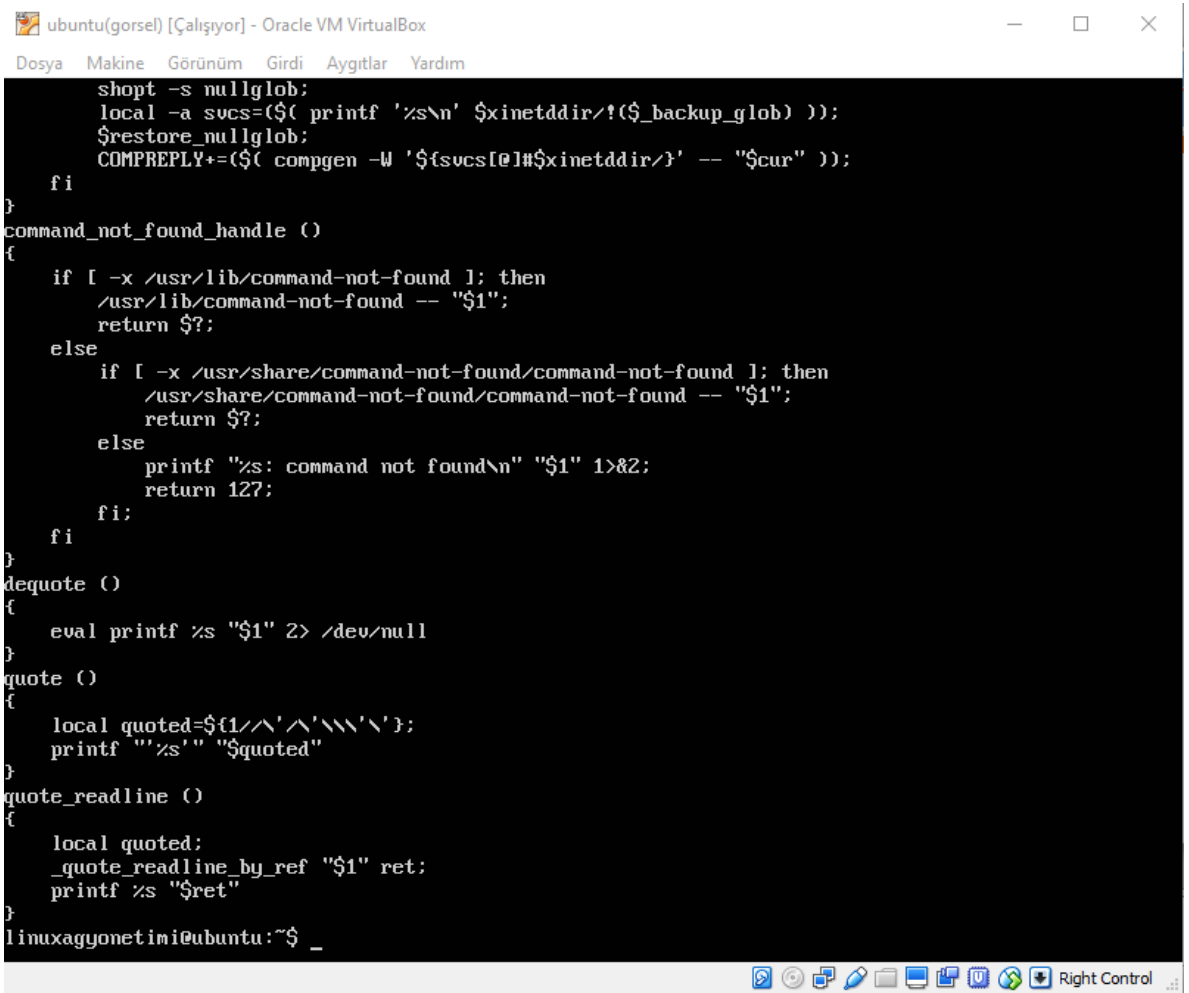
  $ printf "The File count is %d**\n**" $FILE_COUNT

Ortam Değişkenlerinin Listelenmesi (Set komutu)

$ set

- Set komutu, kabuk tarafından görülen tüm değişkenleri (hem yerel hem de ortam değişkenleri) listeler

$ set



Ortam Değişkenlerinin Listelenmesi (Printenv Komutu)

$ printenv

$ env

- Tüm Ortam Değişkenlerinin bir listesini yazdırın (Yerel Vars dahil değildir)

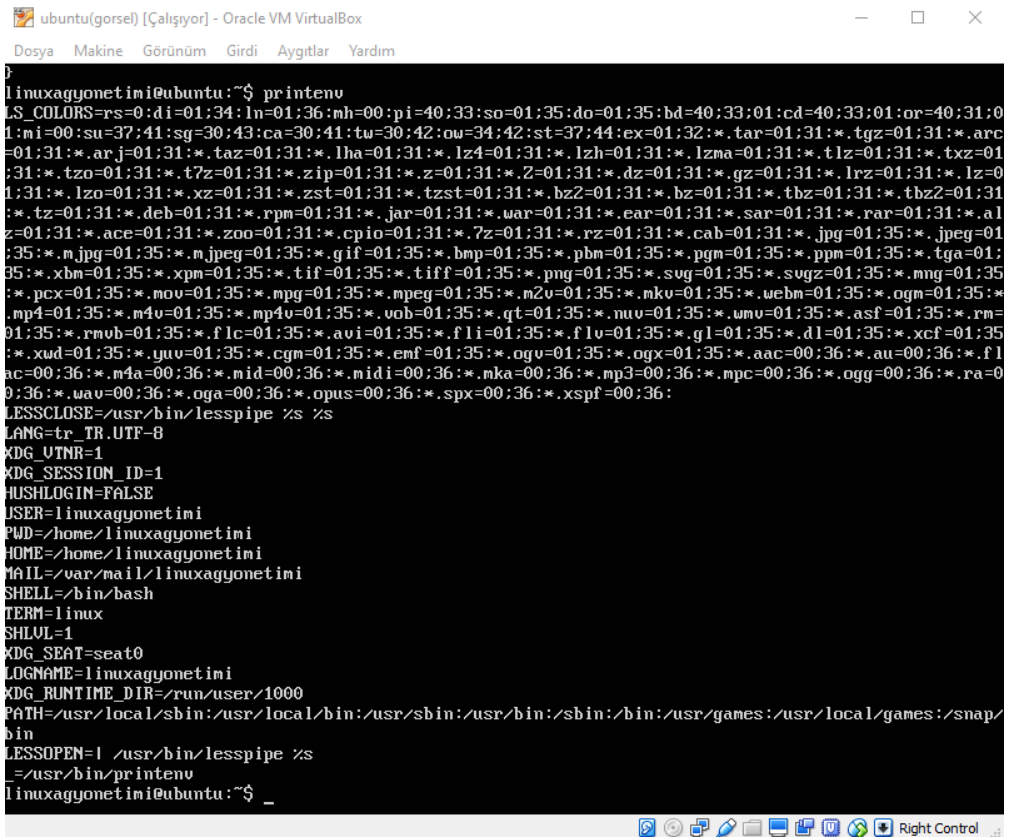
$ printenv <variable>

- Değişkene dolar işareti olmadan koyduğumuza dikkat edin

- Örnekler:

  $ printenv PATH

  $ printenv



$printenv

ORTAK KULLANILAN ÇEVRE DEĞİŞKENLERİ

PATH

- Bu, iki nokta üst üste ":" ile ayrılmış dizinlerin bir listesidir.

/home/tom/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

- Bu liste komutlar ve ikili dosyalar için bir arama yolunu temsil eder, bir komut verirken komutun binary tam yol ile gönderilmesidir.
- Komutu binary adıyla göndererek Linux araması için ayrılmış olarak Linux aramasını bırakın.

  $PATH
- Which komutu, çalıştırılmadan binary dosyanın yolunu bulmak için aynı aramayı gerçekleştirir.
- Geçerli arama yolunu göstermek için

  $ echo $PATH
- Yolun sonuna bir klasör eklemek için

  $ export PATH=$PATH:/usr/bin
- Yolun başlangıcına bir klasör eklemek için

  $ export PATH=/bin:${PATH}

Kullanılır.

Not:

$ ./binary İkili geçerli dizinden çalıştırır

$ binary Arama yoluna dayalı olarak ikili çalıştırır

Geçerli dizini ("./"") yola eklememeniz önerilir

PS1

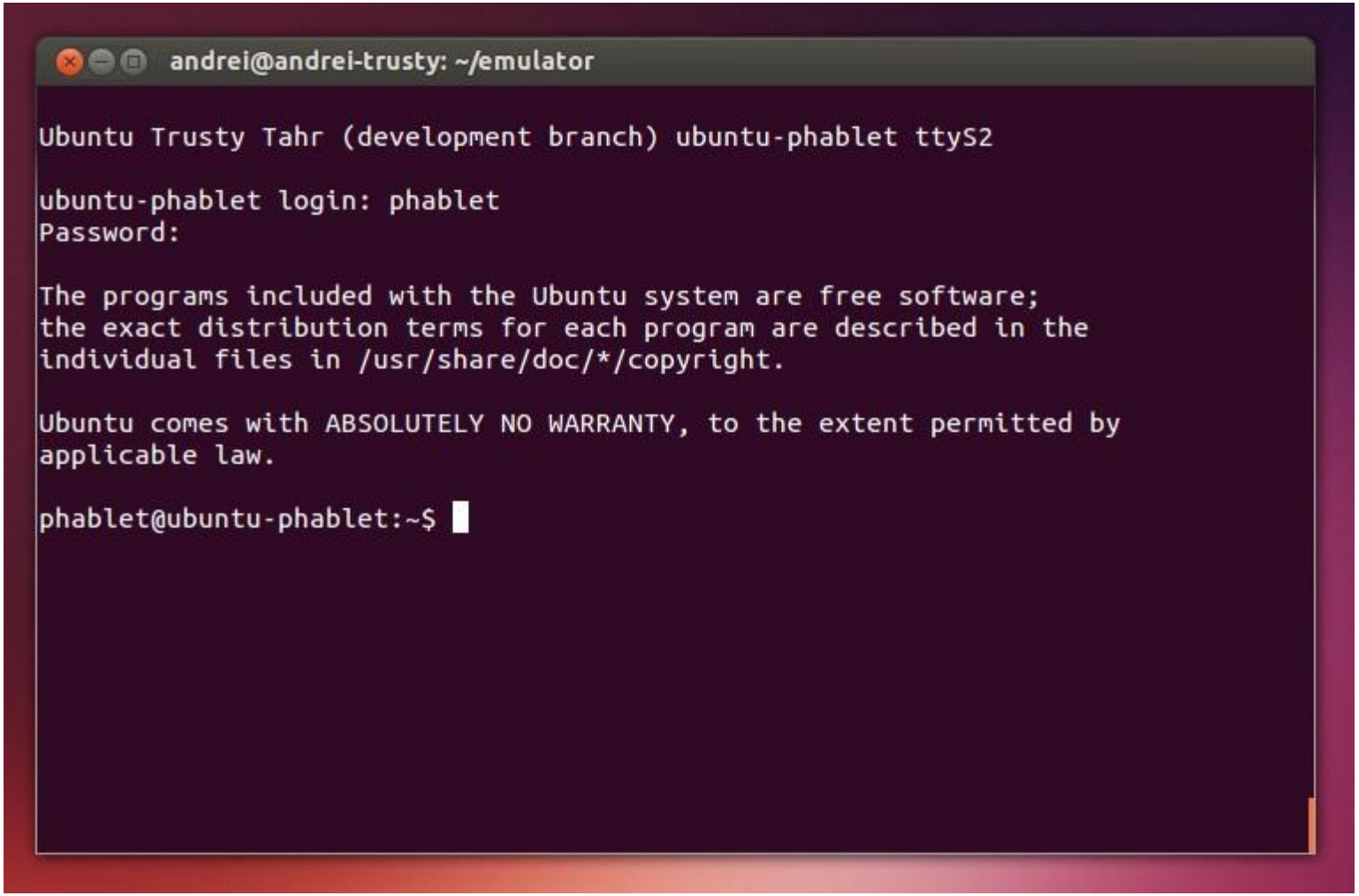
- Kabuk istemini ayarlamaktan sorumludur

- \u ➔ kullanıcı adı
- \h ➔ host adı
- \W ➔ Geçerli çalışma dizini

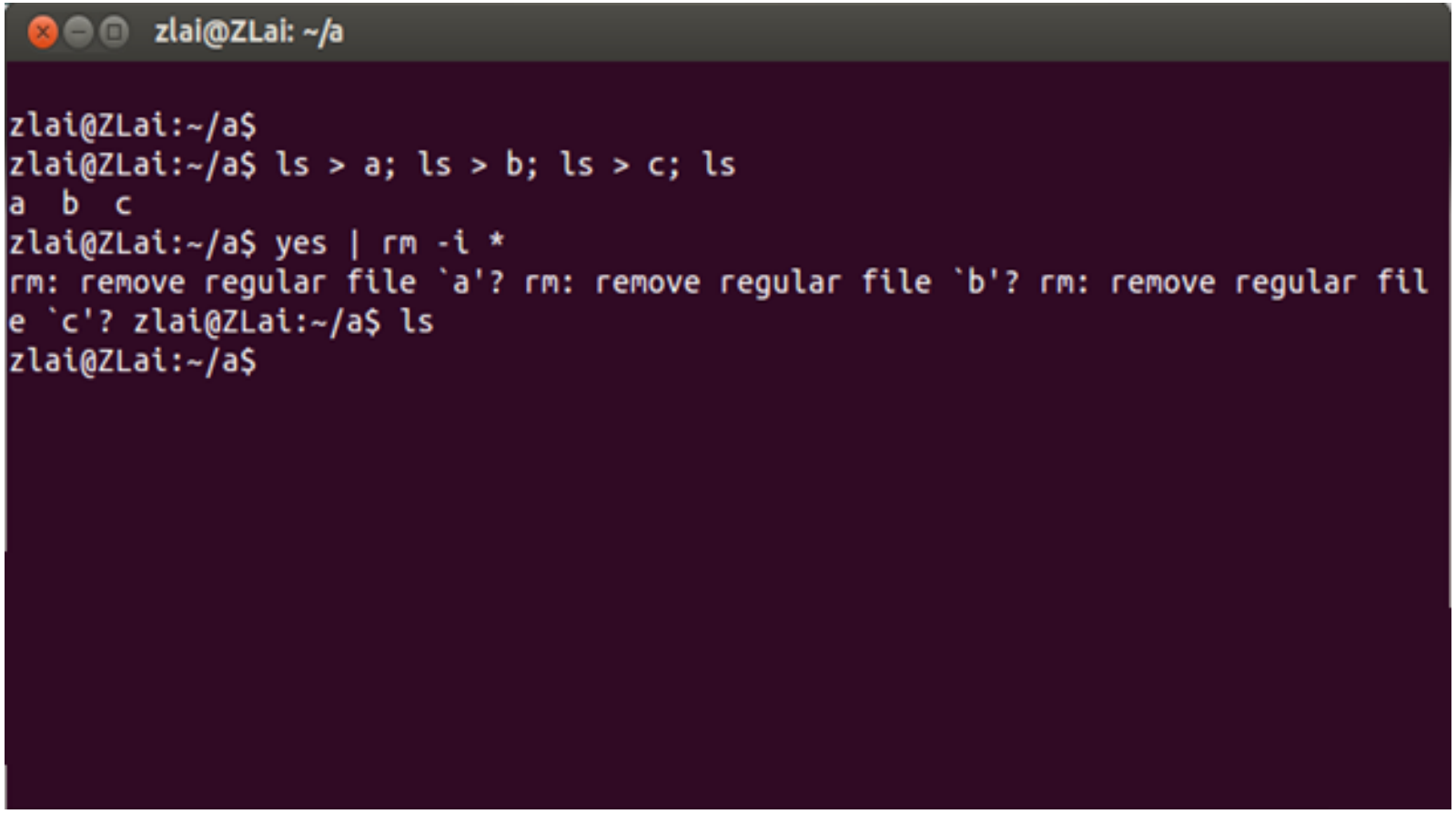
örnek

$ export PS1=[\u@\h \W]\$

PS1


andrei@andrei-trusty: ~/emulator
Ubuntu Trusty Tahr (development branch) ubuntu-phablet ttyS2
ubuntu-phablet login: phablet
Password:
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
phablet@ubuntu-phablet:~$

PS1


zlai@ZLai: ~/a
zlai@ZLai:~/a$
zlai@ZLai:~/a$ ls > a; ls > b; ls > c; ls
a b c
zlai@ZLai:~/a$ yes | rm -i *
rm: remove regular file `a'? rm: remove regular file `b'? rm: remove regular fil
e `c'? zlai@ZLai:~/a$ ls
zlai@ZLai:~/a$

EKRAN

$ DISPLAY

- X ekran adından sorumludur

Bu, sanal bir terminalden (X sunucusunda değil) çalışıyorsanız ayarlanmadığı anlamına gelir

- 3 sayıdan oluşur

  HOST:DISPLAY_NUMBER.SCREEN_NUMBER

- Yerel makine ile uğraşıyorsanız Host alanı boş bırakılır, aksi takdirde x sunucuyu çalıştıran makinenin ana makine adını veya adresini görüntüler
- Ekran numarası, varsayılan ekran için (x-server tty7 üzerinde çalışıyor) 0, diğer sanal terminallerde çalışan diğer x sunucu örnekleri için ise 1,2,3, ... sayı alacaktır.
- Ekran numarası varsayılan olarak 0'dır ve bazen farklı değilse atlanır. Bir ekranın birden fazla ekranı olabilir
- Örnek: ***localhost:4***

  google.com:0

  :0.0

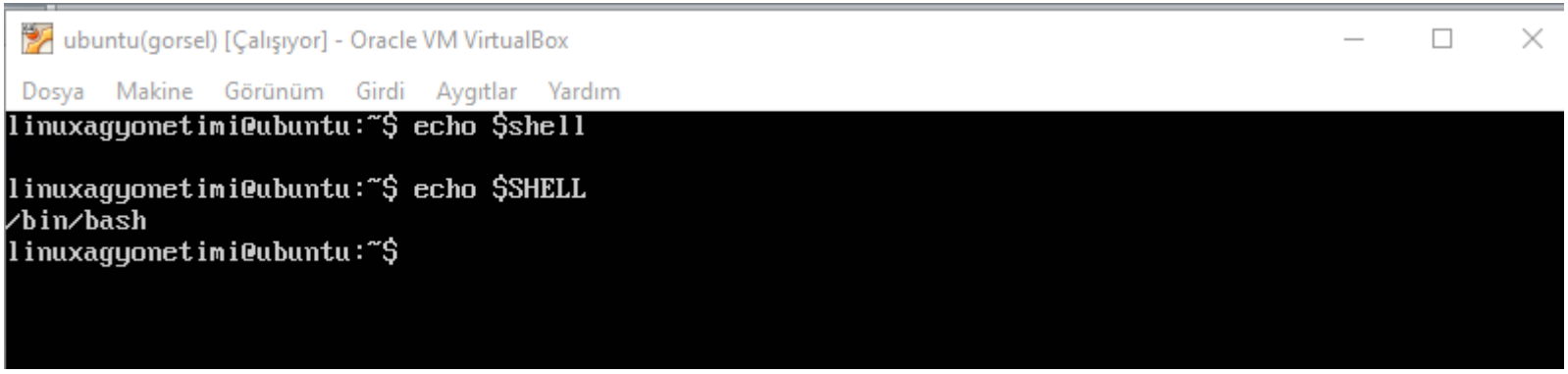
KABUK(shell)

- Giriş kabuğunun yolunu içerir

Örnek:

$ echo $SHELL

/bin/bash



NOT: Büyük/küçük harf duyarlılığı olduğunu hatırlatmakta fayda var

EDİTÖR

- Varsayılan düzenleyicinin adını temsil eder. Eğer herhangi bir editör kurmadıysanız; (vi, herhangi bir gömülü geliştirici için bir zorunluluktur)

```
$ Sudo apt-get install nedit
$Sudo apt-get install geany
$Sudo apt-get install gedit
$ sudo apt-get install alpine-pico
$ sudo apt-get install nano
$ sudo apt-get install vim
$ sudo apt-get install emacs
```

Yazarak editör indirip kurabilirsiniz.

SÜRE

- Kullanılan terminalin türünü temsil eder

Örnekler:

- Benzetilen bir terminalin içinde çalıştırılması durumunda

  $ echo $TERM

  Ekran çıktısı-> xterm

- Bir Linux Sanal terminalinde çalıştırılması durumunda

  $ echo $TERM

  Ekran çıktısı-> linux

```
linuxagyonetimi@ubuntu:~$ echo $term

linuxagyonetimi@ubuntu:~$ echo $TERM
linux
linuxagyonetimi@ubuntu:~$
```

HOME

- Geçerli kullanıcının ana klasörü

Örnek:

$ echo $HOME

/home/linuxagyonetimi

HISTFILE, HISTFILESIZE, HISTSIZE

- $HISTFILE, komut geçmişinin kaydedildiği dosyanın adını taşır.
- $ HISTFILESIZE, geçmiş dosyasında bulunan maksimum satır sayısını taşır
- $ HISTSIZE, komut geçmişinde hatırlanması gereken komut sayısını taşır. Varsayılan değer 500'dür.

```
linuxagyonetimi@ubuntu:~$ echo $HISTFILE
/home/linuxagyonetimi/.bash_history
linuxagyonetimi@ubuntu:~$ echo $HISTFILESIZE
2000
linuxagyonetimi@ubuntu:~$ echo $HISTSIZE
1000
linuxagyonetimi@ubuntu:~$ _
```

HOST ADI

- Makinenizin adını öğrenmenizi sağlar

```
linuxagyonetimi@ubuntu:~$ hostname
ubuntu
linuxagyonetimi@ubuntu:~$
```

Temel Metin İşleme

Neden Metin Dosyaları??

Metin dosyaları Linux'ta büyük rol oynamaktadır!

- Tüm Linux yapılandırma dosyaları metin tabanlıdır.
- Programlar ve komut dosyaları için kaynak kodu barındırır.
- Program günlük dosyaları
- Web Sayfaları (HTML dosyaları)
- XML dosyaları

Bu Dökümanda

- Metin dosyası görüntüleme
- Metin dosyalarını birleştirmek
- Metin dosyası oluşturma
- Bir metin dosyasını boşaltma
- Metin dosyasını düzenleme
- Metin dosyalarını sıralama
- Metin dosyalarında sayma
- Metin dosyalarında arama
- Metin dosyalarını karşılaştırma
- Yamalar Oluşturma ve Kullanma gibi konuları inceleyeceğiz

Bir Metin Dosyasını Görüntüleme (Cat Komutu)

$ cat <file name>

Bu komut, istenen dosyayı, standart olarak çıkış aygıtı (stdout) için gönderir ve bu da varsayılan ekran olur.

$ cat -n <filename>

Bu komut yukarıdakilere benzer, ancak çıktıdaki satır numaralarını da içerir

$ cat <filename> | grep “string”

Bu komut, dosya içeriğini dize arayan grep komutuna gönderir ve bu dizeyi içeren satırları stdout'a gönderir

Dosyaların Birleştirilmesi (cat Komutu)

$ cat file1 file2 file3

Bu komut üç dosyayı birleştirir ve çıktıyı stdout'a gönderir

$ cat file1 file2 file3 > file.total

Üç dosya birleştirilmiş ve file.total içinde saklanır

$ cat * > file.txt

Klasördeki tüm dosyalar tek bir dosyada birleştirilir

Not,

Kısa videolar içeren bir klasöre sahipseniz ve bunları tek bir filmde birleştirmek istiyorsanız,

$ cat *.mpeg > movie.mpeg

Bir Metin Dosyasını Görüntüleme (More / less Komutları)

- Cat komutu çıktılar toplu olarak ekrana gönderildiğinden, uzun dosyalarda sorun var.
- Bunu düzeltmek için, komutları daha fazla ve daha az kullanabiliriz
- Her iki komut da dosyayı sayfa modasına göre bir sayfada görüntüler

  $ less <filename>

  $ more <filename>

- Azalan komut daha iyidir, çünkü yukarı / aşağı kaydırmanın kullanımına izin verir)

- Bir borudaki son komut olarak çok yararlıdır,

  $ ls * | grep "log" | sort | less ls -la |

  less

Metin Dosyasının bir bölümünü görüntüleme (head/ tail Komutları)

$ head <filename>
$ tail <filename>

- Bu komutlar dosyanın yalnızca ilk (veya son) bölümünü listeler
  $ head file1 (Dosyanın ilk 10 satırını görüntüler)
  $ tail file2 (Dosyanın son 10 satırını görüntüler)

- Listelenecek satır sayısını belirleyebilirsiniz
  $ head -100 file1 (Dosyanın ilk 100 satırını görüntüler)
  $ tail -50 file2 (Dosyanın son 50 satırını görüntüler)

- Bir dosyanın güncellemelerini canlı bir şekilde listelemek için (günlük dosyaları, program çalışırken kurdukları gibi izlemek için çok kullanışlıdır)
  $ tail -f file1

- İzlenecek çok yaygın günlük dosyası
  $ tail -f -50 /var/log/messages

Boş bir Metin Dosyası Oluşturma

- Dosyaya boş bir dizgenin yönlendirmesini kullanabilirsiniz

  $ > file.txt

File.txt varsa, bunun içeriğini boşaltacağına dikkat edin (buna dosyanın kesilmesi denir)

- Dokunmatik komutu kullanabilirsiniz.

  $ touch file.txt

File.txt varsa, bu komut dosyanın zaman damgasını içeriğini değiştirmeden güncelleyeceğini unutmayın.

Metin Dosyaları Oluşturma / Ekleme

- Metin editörlerinde dosyalar oluşturabiliriz ancak aşağıdakilerden birini kullanarak dosyaları hızlı bir şekilde oluşturabiliriz.
  - echo Komutu,

    $ echo "Good Morning" > file1

    $echo "Good Morning" >> file2

Garip görünse de bazen bir betik içinde bir dosya oluştururken yararlıdır

  - cat Komutu,

    $ cat > filename

    <put first line of text >

    .....

    <put last line of text>

    ^d

^ D'nin EOF (dosyanın sonu) anlamına geldiğini unutmayın.

Bir Metin Dosyasını Düzenleme

- GUI Tabanlı Editörler:

En ünlüler şunlardır:

Emacs or ***xemacs*** (Çok güçlü ve karmaşık)
gedit (in Gnome) ***Kate*** (in KDE) ***gvim*** (Based on vi)

- GUI Tabanlı Olmayan:

En ünlü olanı vi veya vim'dir (herhangi bir gömülü geliştirici için bir zorunluluktur)

Not:

- Düzenleyici vi (veya vim) ilk kullanımda çok ilkeldir, ancak buna alışırsanız bunun son derece güçlü bir editör olduğunu göreceksiniz
- Ayrıca, bazen, vi katıştırılmış platform hedefindeki tek kullanılabilir düzenleyicidir. Dolayısıyla, gömülü geliştiriciler için buna alışmak çok önemlidir

Metin İçinde arama yapma(Grep Komutu)

$ grep <string or pattern> <files to search>

- Metin aramak için çok güçlü bir araçtır
- Bir dosya, klasör veya bir klasör ağacında belirli bir metin düzenini arayabilir
- 'Grep', arama yeteneğini geliştirmek için güçlü düzenli ifadeleri kullanabilir (ileride yapılacak bir konuşmada tartışılacaktır)

- Örnek

  $ grep "error" ./*.log

  $ grep "fatal error" .

  $ cat *.log | grep "error" | less

Grep Komutu

$ grep -i <pattern> <files> (Büyük küçük harfe duyarlı olmayan arama)

$ grep -r <pattern> <files> (Özyineli arama)

$ grep -v <pattern> <files> (Ters arama, deseni içermeyen satırları gösterin)

$ grep -n <pattern> <files> (Satır numaralarını göster)

Örnekler:

$ cat * | grep "error" | grep –v "fatal"

$ ls -l | grep "project"

$ grep -nr "printf" *.c

$ cat *.h | grep -n "#define"

Metin Sıralama (Sıralama komutu)

$ sort <file>

Bu komut bir dosyadaki satırları / dosyaları alfabetik olarak sıralamak için kullanılır

Örnekler:

$ sort file1 (Dosyanın satırlarını sıralayın ve

$ sort file1 > file2 stdout'a gönderin)

$ sort -r file1 (Ters sıralama)

$ sort -u file1 (Benzer satırları gösterme)

Diğer Seçenekler:

- **-b** OR **--ignore-leading-blanks**
- **-f** OR **--ignore-case**
- **-n** OR **--numeric-sort** (99 is less than 100)

- Tablolaştırılmış veriler için belirli alanlarda karşılaştırmalar yapmak için sıralama, örneğin daha uzun dosya ayrıntıları listesinin dosya boyutunu sıralamak için daha gelişmiş seçeneklere sahiptir

Tekrarlanan Hatları Kaldırma (Uniq Komutu)

$ uniq file1

Bu komut, bir dosyada / dosyalarda tekrarlanan satırları kaldırmak için kullanılır Bu, günlük dosyalarında çok yaygın bir şeydir

Örnekler:

$ uniq file1 (Tekrarlanan satırları kaldırın ve çıktıyı standart çıktıya gönderin)

$ uniq -d file1 (Yalnızca yinelenen satırları göster)

Not:

- Uniq komutu komşu tekrarlanan çizgiler üzerinde çalışır, bu nedenle bitişik olmayan tekrarlanan çizgileri kaldırmak isterseniz

  $ cat file1 | sort | uniq

Metin Dosyalarında Sayma (wc Komutu)

$ wc <file>

Bu komut, bir dosyadaki veya dosyalardaki kelimeleri / satırları / karakterleri saymak için kullanılır

Örnekler:

$ wc file1

$ wc -l file1 (Yalnızca satır sayısı)

$ wc -c file1 (Sadece karakter sayısı)

$ wc -w file1 (Sadece kelime sayısı)

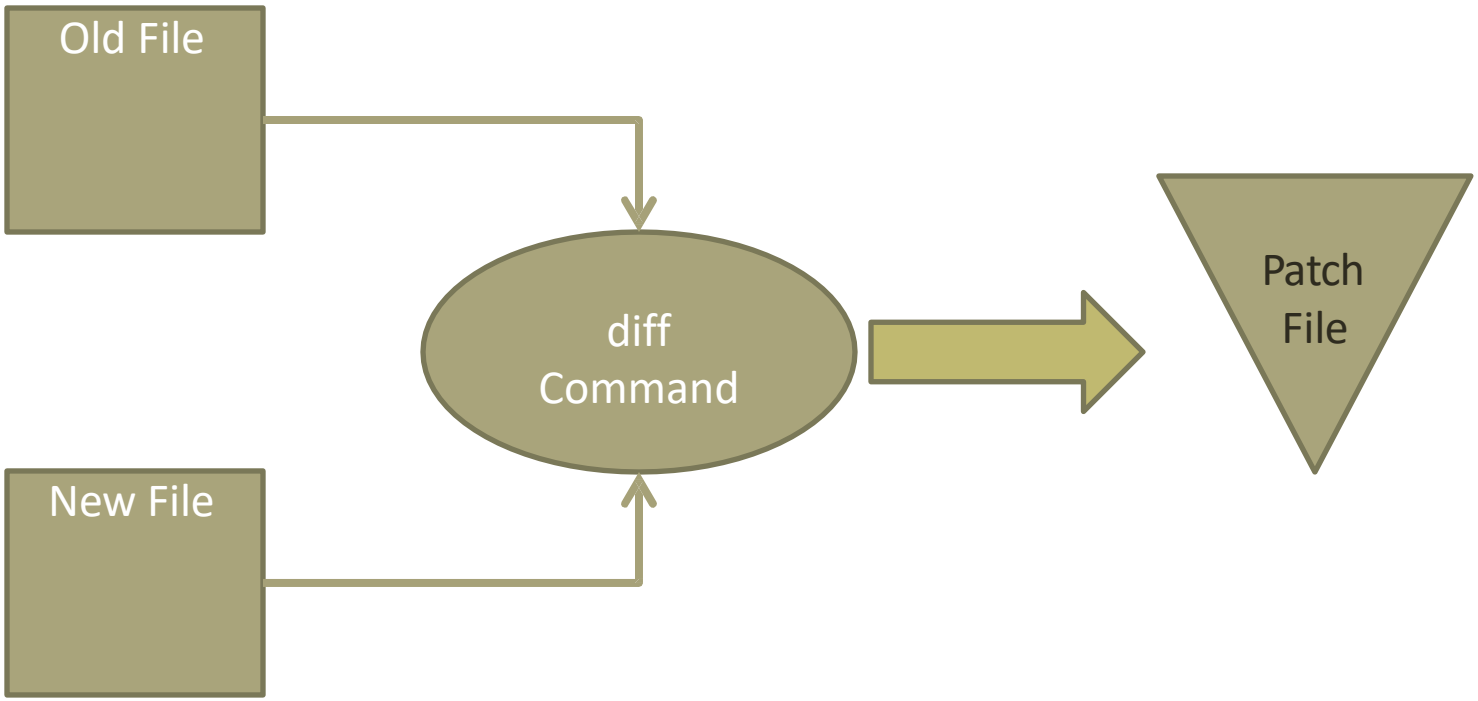
$ cat * | wc -l

Metin Dosyalarını Karşılaştırma

- Linux'ta metin dosyalarını karşılaştırmak için çeşitli GUI tabanlı araçlar vardır,
  - ***Meld***
  - ***Kdiff3***
- Bu araçlar hem iki metin dosyasını karşılaştırmak hem de değişiklikleri üçüncü bir dosyada birleştirmek için uygundur.
- Bu, özellikle aynı dosyanın iki sürümü arasındaki değişiklikleri tanımlamak için kaynak kodu dosyalarında kullanmak için çok kullanışlıdır
- Karşılaştırmayı, komut satırı Arayüzünü komut farkını kullanarak da gerçekleştirebiliriz

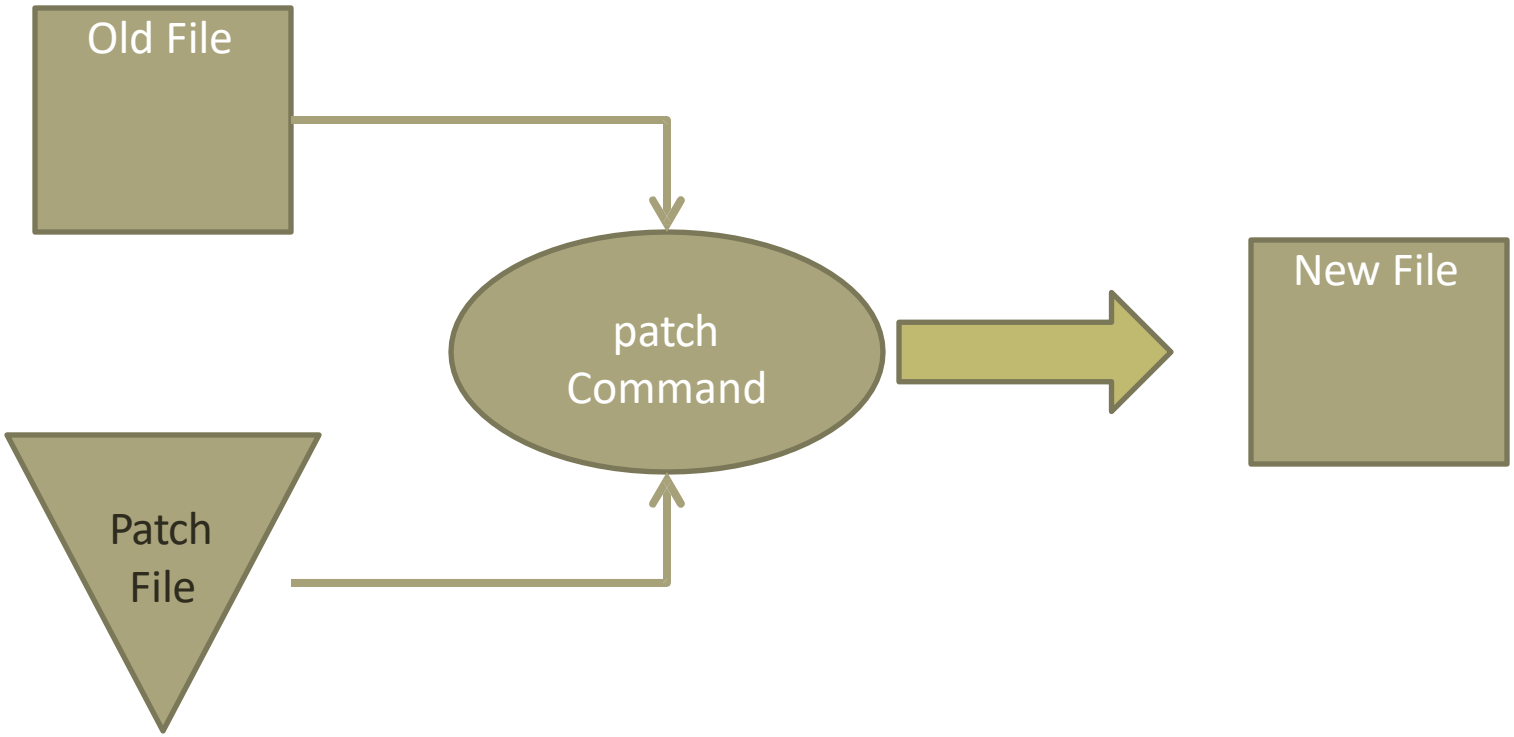
Metin Dosyalarını Karşılaştırma (diff komutu)

- Diff komutu, dosyaların karşılaştırılmasında ve daha sonra kullanılmak üzere yamalar (dosyalar arasındaki deltalar) oluşturulmasında çok yararlıdır.

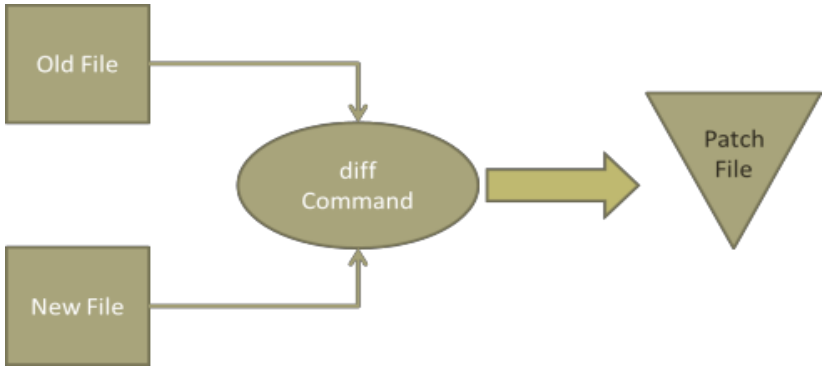


Metin Dosyalarını Düzeltme(Patch komutu)

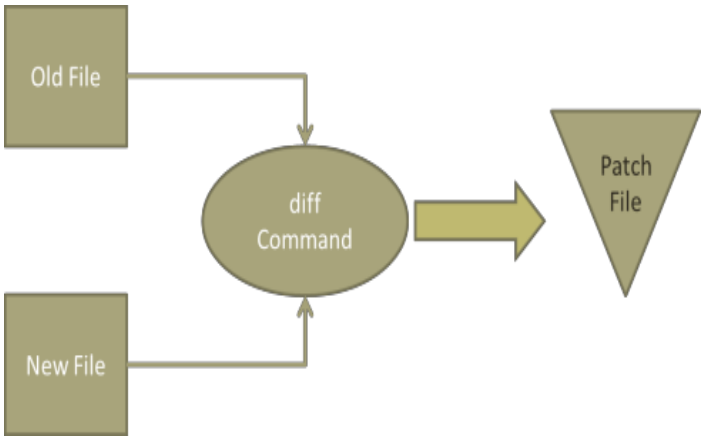
- Patch komutu, eski dosyayı yenisine güncellemek için Düzeltme Dosyası dosyasını eklemek için kullanılır

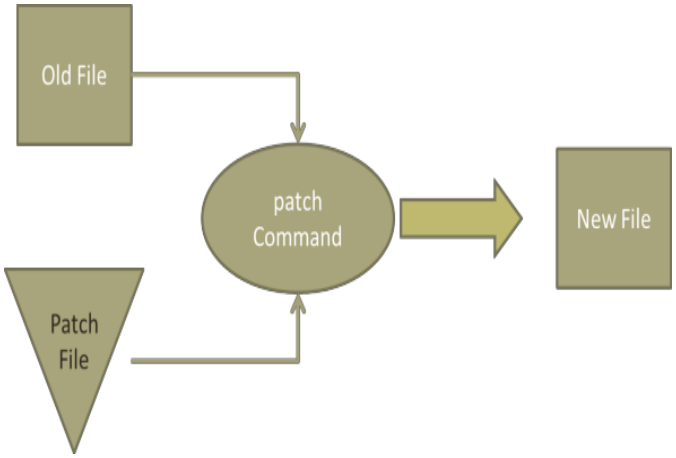


Yamalar Oluşturma ve Kullanma



Yamalar Oluşturma ve Kullanma





Metin Dosyalarını Karşılaştırma (diff komutu)

$ diff from-file to-file

Bu komut, iki dosyayı aşağıdaki gibi karşılaştırır;

- Eğer from-file ve to-file ikisi de dosya isimleriyse, diff bu iki dosyayı karşılaştırır
- From-file ve to-file'tan biri dosya ismi ve diğeri de bir dizin ise, diff, belirtilen dizinde aynı adı taşıyan biriyle karşılaştırır.
- Eğer from-file ve to-file dizinleri ise, diff, iki dizindeki ilgili dosyaları karşılaştırır. "-r" kullanılırsa karşılaştırma özyineli olabilir
- Çıktının stdout'a gideceğini, bir düzeltme eki dosyasına gönderileceğini, çıktıyı yeniden yönlendirmeniz gerektiğini unutmayın.

  $ diff old-file new-file > patch-file

Metin Dosyalarını Karşılaştırma (diff komutu)

Fark komutu çıktısı aşağıdakilerden biri olabilir:

- Normal Format

  $ diff <old-file> <new-file>

- Bağlam Biçimi

  $ diff -c <old-file> <new-file>

- Birleştirilmiş Biçim

  $ diff -u <old-file> <new-file>

'diff' komutu Normal Çıkış Formatı

- Normal çıktı, iki dosya arasında karşılaştırılacak metin bölümlerinden oluşur
- Her bölüm, birinci dosyadaki bir satır veya daha fazla, ikinci dosyadaki bir satır veya daha fazladır.
- Ardından her bölümde, bu bölümdeki ana farkı açıklayan öncü bir çizgi (Değiştirme Komutu denir) bulunur. Bu satır, farkı gösteren bir mektup içerir
  - 'A' harfi APPEND anlamına gelir
  - 'C' harfi CHANGE için
  - 'D' harfi, DELETE ibaresidir.
- Yani Değişiklik Komutanlığı formatına örnekler,

8a12,15

5,7c8,10

5,7d3

'diff' komutu Normal Çıkış Formatı

8a12,15

İlk dosyadaki Satır 8, ikinci dosyadan 12-15 satırla eklenir

5,7c8,10

İlk dosyadaki 5-7 satırlar, ikinci dosyadaki satırlar 8-10 ile değiştirilir.

5,7d3

İlk dosyadan 5-7 arasındaki satırları silin (silinmediyse, ikinci dosyada 3. satırda başlamış olmalıydı).

'diff' komutu Normal Çıkış Formatı

Değiştirme Komutanından sonra, düzeltme eki dosyası, ilk dosyadaki ilgili satırları ve ardından bir ayırıcıyı, ardından ikinci dosyadaki satırları içerecektir

8a12,15
> Bu, eklenecek ilk satıra (ikinci dosyadaki satır 12)
> Bu, eklenecek ikinci satıra (ikinci dosyadaki satır 13)
> Bu eklenecek üçüncü satır (ikinci dosyadaki 14 satır)
> Bu, eklenecek dördüncü satırdır (ikinci dosyadaki satır 15)

5,6c8,10
< İlk dosyadaki 5. satır (silinecek)
< İlk dosyadaki 6. satır (silinecek)

>İkinci dosyanın 8. satırı (eklenecek)
> İkinci dosyanın 9. satırı (eklenecek)
> İkinci dosyanın 10. satırı (eklenecek)

5,7d3
< İlk dosyadaki satır 5 (silinecek)
< İlk dosyadaki 6. satır (silinecek)
< İlk dosyadaki satır 7 (silinecek)

'diff' komutu Bağlam Biçimi

- Bu formatta, diff her dosyanın bağlamını göstermek için daha fazla satır çıktılar. Bu satırlardan bazıları değiştirilmedi, ancak diğer satırlarda bazı değişiklikler var
- Çıktı, her dosya için bir sembol dizesi tanımlamayla başlar
- Bu işlem, sembolü takiben dosya ismi, ardından değiştirilme tarihi koyularak yapılır.
- Ardından, her bölümde dosya değişiklikleri için metin bölümleri olacaktır,
  - Bakmak için satır numaralarını gösterecek her dosya için bir satır.
  - Her dosyadaki satırların bir listesi
  - Her satırın önünde, o satırın durumunu gösteren bir kod bulunur
    - Hiçbir şey: satır her iki dosyada da aynı demektir
    - '!': Satırda değişiklik olduğu anlamına gelir. Bu sembol, her iki dosyada da durumun öncesinde ve sonrasında gösterilir
    - '+': Satır eklendi demektir
    - '-': satır kaldırılmış demektir

'diff' komutu Bağlam Biçimi

*** old-file May 30 12:30:30 2014
--- new-file June 1 07:12:40 2014

****

*** 3,7 ****

- Bu, birinci dosyanın 3. satırıdır (ikinci dosyada silinir)
- Bu, birinci dosyanın 4. satırıdır (ikinci dosyada silinecektir)
- Bu 5 numaralı hattır (değişmeden bırakılacaktır)
- Bu satır # 6 (başka bir şeyle değiştirilecektir)
- Bu 7. satırdır (değişmeden bırakılacaktır)

--- 2,5 -----

Bu ikinci dosyadaki satır # 2 (birinci dosyadaki satır # 5'e benzer)

! Bu ikinci dosyadaki satır # 3 (ilk dosyadaki 6. satırın yerini alacak)

! Bu ikinci dosyadaki satır # 4'dür (ilk dosyadaki satır # 6'nın yerini alacaktır)

Bu, ikinci dosyadaki satır # 5'tir (ilk dosyadaki satır # 7'ye benzer şekilde)

'diff' komutu Birleştirilmiş Biçim

- Bu çıktı biçimi, İçerik Biçimi'nden daha kompakt olmaya çalışıyor
- Listeleme için 2 kopyaya sahip olmak yerine (dosya başına bir tane), her iki dosyadaki satırları ve ilk dosyadan çıkarılan satırları ve bunlara eklenen satırları gösteren bir liste olacaktır
- Çıktı, eski dosyaya bir sembol (normalde '---') ve yeni dosyaya bir sembol (normalde '+++') atanarak başlar
- Bu dosyayı, dosyalar arasında değişiklikler içeren bir grup metin bölümü izler
- Her bölümde,
  - Her dosya için satır numarası aralığı
  - Bu aralıktaki satırların bir listesi,
    - Bir '' ile başlayan, satırın iki dosya arasında paylaşıldığı anlamına gelir
    - Bir '+' ile başlamak, satırı yalnızca ikinci dosyada (eklenmiş)
    - Bir '-' ile başlayan satır, yalnızca ilk dosyada (kaldırıldı)

'diff' komutu Birleştirilmiş Biçim

--- old-file May 30 12:30:30 2014

+++ new-file June 1 07:12:40 2014

@@ -3,5 +2,4 @@

- Bu, birinci dosyanın 3. satırıdır (ikinci dosyada silinir
- Bu, birinci dosyanın 4. satırıdır (ikinci dosyada silinecektir)
- Bu, birinci dosyanın 5. satırıdır (ikinci dosyadaki 2. satırla aynıdır)
- Bu 6. satırda (kaldırılacaktır)

+ Bu, ikinci dosyada 3. satırdır (ilk dosyada mevcut değildir)

+ Bu ikinci dosyadaki satır # 4 (ilk dosyada mevcut değildir)

Bu, ilk dosyadaki 7 numaralı satırdır (ikinci dosyadaki 5 numaralı satır ile aynıdır)

Yamalar Uygulanarak Tek Bir Dosyaya Düzeltme

$ patch <original file> <patch file>

Bu komut yeni dosyayı oluşturmak için düzeltme ekini özgün (eski) dosyaya uygular

- Varsayılan olarak, düzeltme eki dosyasında kullanılan biçimi algılamaya çalışır
- Bu otomatik algılama, aşağıdakileri kullanarak geçersiz kılınabilir:

  $ patch -n <original file> <patch file>

  $ patch -c <original file> <patch file>

  $ patch -u <original file> <patch file>

- Yama orijinal dosyaya uygulanır ve dosya güncellenir
- Düzeltmeyi kaldırmak için (dosyayı düzeltin)

  $ patch -R <updated file> <patch file>

Yamalar Uygulama Dizin Ağacına Düzeltme

$ patch -p<num> < <patch-file>

- Düzeltme eki dosyası, iki dizini karşılaştırarak oluşturulduysa

  $ diff -ru old-dir new-dir > patch-file
- Şimdi eski içeriği olan başka bir dizini güncellemek için yama kullanabiliriz

  $ patch -p1 < patch-file
- Bu örnekte hedef dizininin yama dizini oluşturulurken kullanılan dizin ile aynı derinlik ve adreste olduğunu varsayarak -p1 kullandık ve komutu
- Durum böyle değilse, soymak istediğiniz dizin adının ne kadarına bağlı olarak <num> 'ı seçiyoruz
  - Örneğin, yama dosyasındaki orijinal dizinin yola sahip olduğunu söyleyelim

    /home/tom/projects/alpha-project/src/….
  - Ve hedef dizini şu adreste bulunur:

    /home/bob/src/…
  - Sonra yapacağımız

    $ cd /home/bob

    $ patch -p5 < patch-file

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/