KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

10. Hafta – Düzenli İfadeler

Yrd. Doç. Dr. A. Burak İNNER

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

Düzenli İfadeler (RegularExpressions) Nedir ??

- Basit metin işlemleri (tek dizeler üzerinde arama yapma gibi) nasıl yapılacağını bir önceki dökümanda incelemiştik

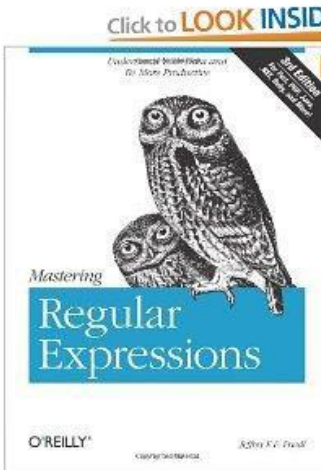
Örneğin ;

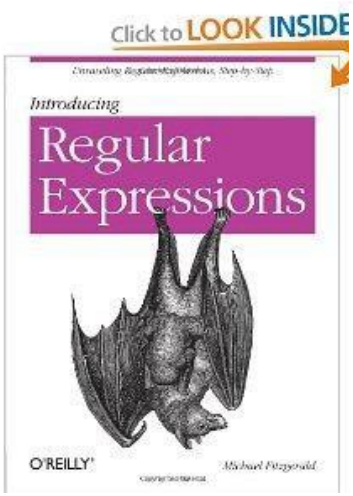
- Dizenin sadece satır başında olduğu durumu arama
- İstenilen bir modelin aranması örneğin telefon numarası,e posta
- Bu düzenli ifadelerin REGEX rolüdür.

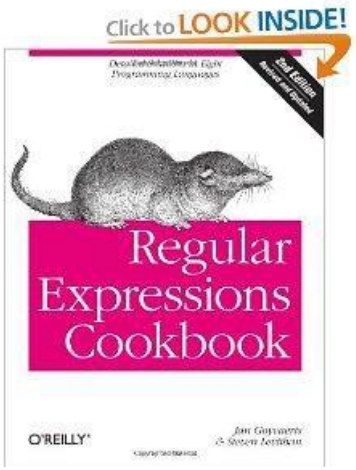
Düzenli İfadeler (Regular Expressions) Nedir ??

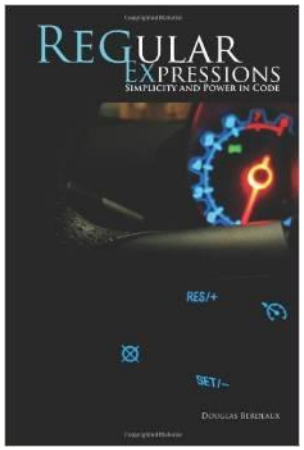
- Dosya adlarını içeren desenleri joker karakterler ve diğer açılımlarla kullanmayı öğrendik ancak bu gelişmeler dosya adlarıyla sınırlıydı.
- Düzenli ifadeler çeşitli linux araçları tarafından kullanılmak üzere metin desenleri oluşturmak için çok güçlü bir araçtır.
  (özellikle metin içinde arama yapmak için)
- Düzenli ifadeler benzer teknikler kullanır ancak genel metin kalıpları için kullanılır.
- Komut dosyası dillerinde ve araçlarda '***perl***' ve'***python***' kullanılır.
- Bunu kullanan ana araçlardan biri grep dir.('Grep' de 're' düzenli ifadeler anlamına gelir)

Regex Karmaşıktır









- Bir çok özel durumlar ve formlar vardır
- Bazı özel araçları kullanıyor,fakat kurallarda ufak değişiklikler vardır.
- Bazı araçların bash de kullanılanın daha zengin seti vardır (**perl gibi**)
- Bu doküman sadece konuyu tanıtmayı hedeflenemektedir, REGEX ile ilgili çok sayıda kaynak vardır.

grep Komutu

$ grep [seçenek] <string/pattern> <dosya(lar)>

Dosya kümesindeki dize veya deseni arar.

seçenek	tam hali	açıklama
-i	**--ignore-case**	Durumu göz ardı et
-v	**--invert-match**	Eşleşmeyen satırları göster
-c	**--count**	Sadece eşleşenlerin sayısını yazdır
-l	**--files-with-matches**	Sadece dosya adlarını yazdır
-L	**--files-without-match**	
-n	**--line-number**	Satır numaralarını ekle
-h	**--no-filename**	Dosya adını ekleme

Literal ve Meta Karakterler

- Literal karakterler , arama modelinde kendilerini temsil eden karakterlerdir.

  $ grep "hata" *.log

  "hata " daki harflerin tümü,tam metin karakterleridir.

```
bilg2017@bilg2017-VirtualBox:~$ grep "hata" dosya.txt
hata
```

- Meta karakterler ,özel anlamları olan karakterlerdir.

  **^ $. [ ] { } - ? * + ( ) | **

  Diğer tüm karakterler literal karakterlerdir.

Literal ve Meta Karakterler

- Normalde kabuk kurallarına göre genişlemeyi önlemek için meta karakterli herhangi bir ifadeyi bir telifin içine koymamız önerilir.
- Meta karakterler kaçındıklarında ,yani eğik çizgiyle başlayarak,harfler olarak ele alınabilirler.
  - Örneğin; **\^ \{ \$ **
- Geri eğik çizgi bazı hazır bilgi karakterlerini bir meta karakter haline dönüştürebilir.
  - Örneğin ; **\d \w**

  - **Düzenli ifadeler,literal ve meta karakterlerin bir karışımından oluşturulan metin desenleridir.**

Düzenli İfadelerin Türleri

- POSIX , iki tür düzenli ifadeyi tanımlar
  - Temel Düzenli İfadeler(Basic Regular Expressions (BRE))
  - Genişletilmiş Düzenli İfadeler(Extended Regular Expressions (ERE))

- Temel düzenli ifadeler(BRE) aşağıdaki meta karakterleri kullanır,diğer tüm karakterlerde harf olarak kabul edilir:

  . ^ $ [ ] *

- Genişletilmiş düzenli ifadeler(ERE) temel kümeye ek olarak aşağıdaki setin kullanılmasını sağlar:

  ( ) { } ? + |

- Ters eğik çizgi bu meta karakterleri (ERE) ve (BRE) de tersine çevirir.
- '***grep***' aracı BRE yi kullanır.
- ERE ye erişmek için, '***egrep***' ya da '***grep –E*** ' kullanılır.

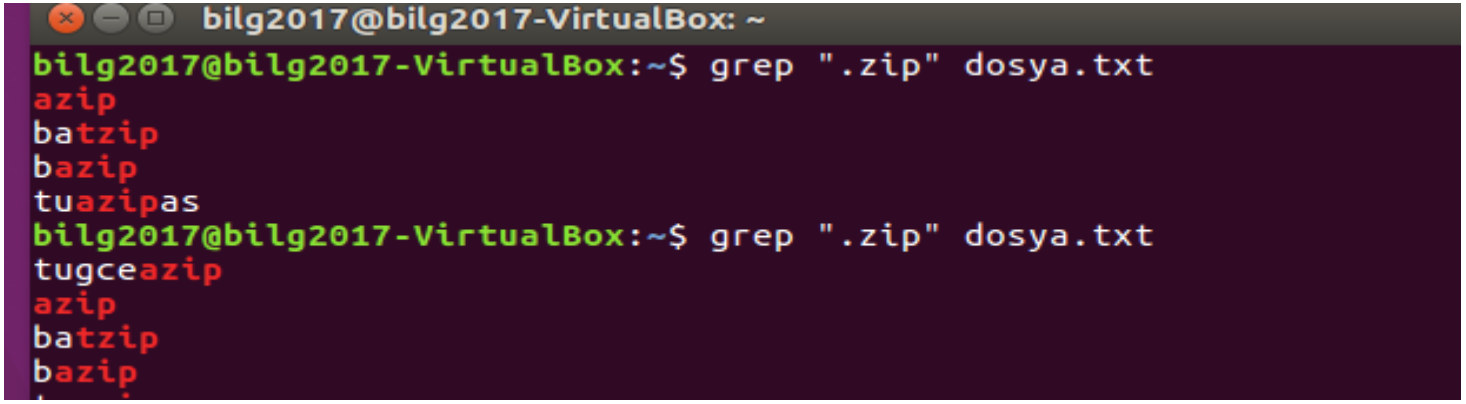
TEMEL DÜZENLİ İFADELER (Basic Regular Expressions (BRE))

“ANY” Karakteri (Nokta karakteri ‘ . ’)

- ‘***.***’ (nokta) tek bir karakteri temsil eden bir meta karakterdir.(NULL karakteri içermez.)
- Örneğin,

  $ grep “.zip” file.log

  Bu örnek herhangi bir karakterle başlayan ve ardından ‘zip’ kelimesini içeren 4 harfli bir metin desenini arar .
  - Sonuçlar : ***gzip***, ***bzip***, ***gnubzip*** , ***rezipped***
  - ***zip*** olmaz.



(^) ve ($) Karakterleri

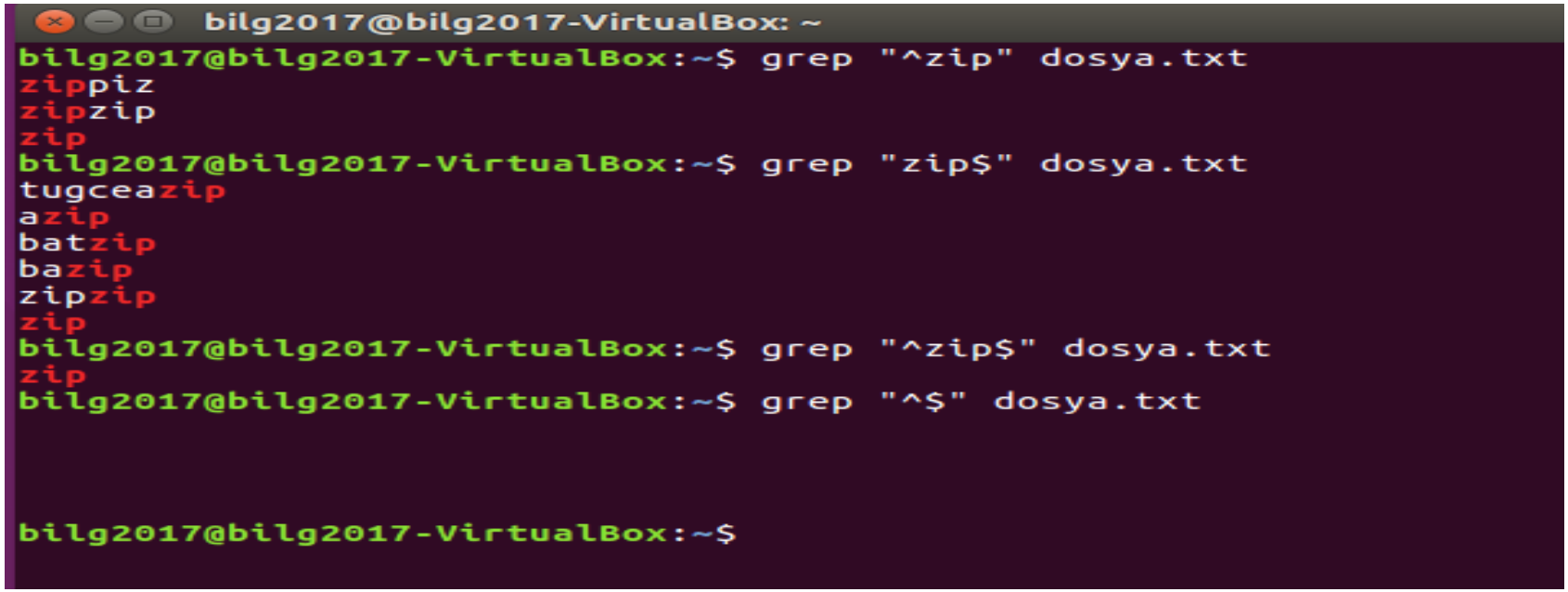
- Dizenin başında bulunan (**^**) ,dizenin satır başında olması gerektiği anlamına gelir.
- Dizenin sonunda bulunan (**$**) ,dizenin satırın sonunda olması gerektiği anlamına gelir.
- örnek:

$ grep "^zip" file.txt ➔ 'zip' ile başlayan herhangi bir satırla sonuçlanır.

$ grep "zip$" file.txt ➔ 'zip' ile biten herhangi bir satırla sonuçlanır.

$ grep "^zip$" file.txt ➔ sadece 'zip' içeren bir satırla sonuçlanır.

$ grep "^$" file.txt ➔ boş satırla sonuçlanır.



Parantez İfadeleri ([ ) and ( ])

- Köşeli parantezler arasında listelenen herhangi bir karakter dizisi için köşeli parantez kullanımı;

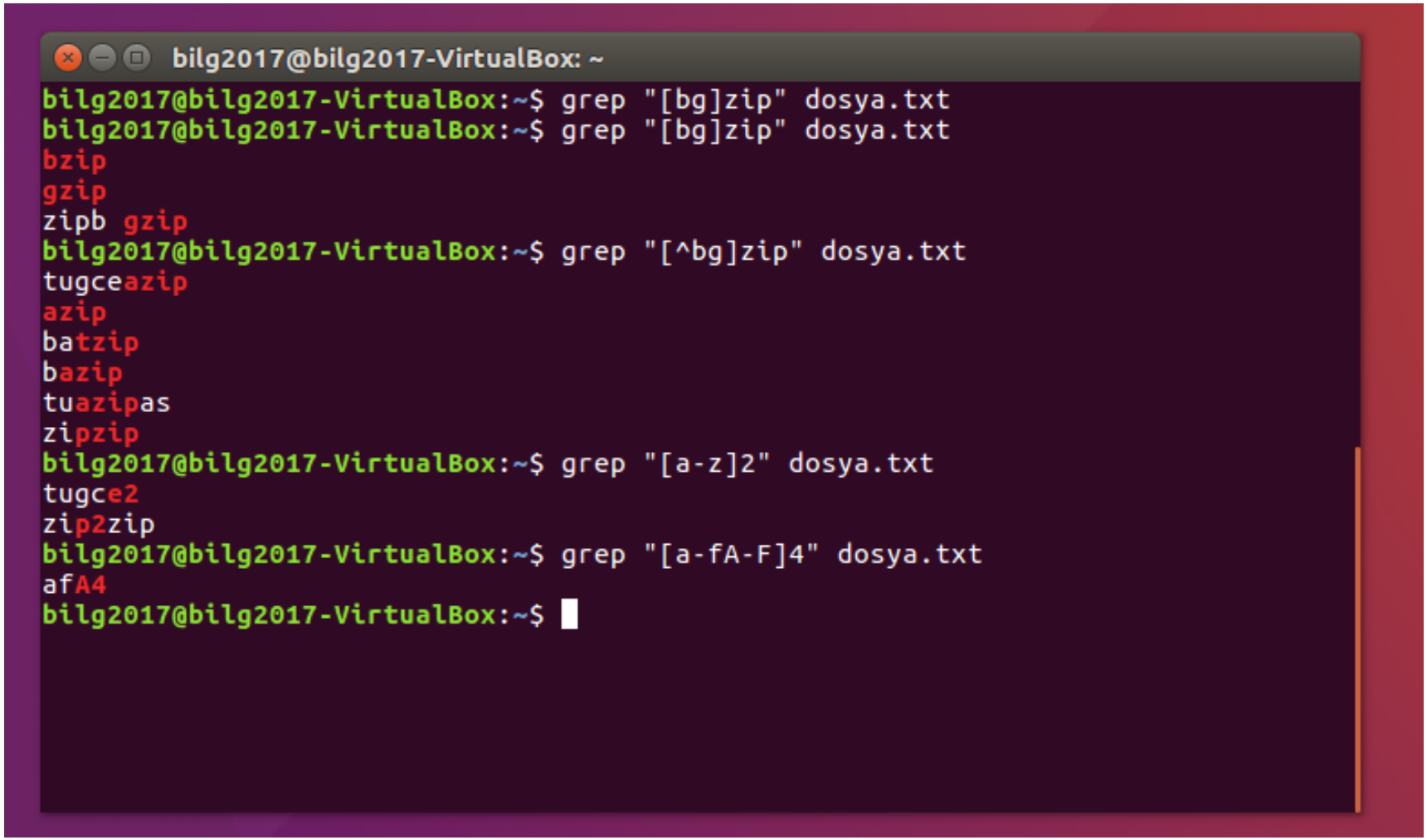
  $ grep "[bg]zip" dict.txt

  Sonuçlar: **bzip**, **gzip**, **aabzip**
- Parantez içindeki herhangi bir karakter haricinde literal olarak kabul edilecektir.
  - (**^**) Eğer başlangıçta gelirse (olumsuzlama olarak kabul edilecektir)
  - (**-**) Eğer ortada gelirse (aralık olarak kabul edilecek)
- Olumsuzlama, haricinde herhangi bir karakterle 'zip' öneri kelimeleri yakalar.

  $ grep "[^bg]zip" dict.txt
  - b veya g veya Null karakteri haricinde herhangi bir karakterle 'zip' öneri kelimeleri yakalar.
- Aralıklar

  $ grep "[a-z]2" dict.txt
  - Herhangi bir küçük harfle başlayan ve ardından 2 gelen kelimeler.

  $ grep "[a-fA-F]4" dict.txt


bilg2017@bilg2017-VirtualBox: ~
bilg2017@bilg2017-VirtualBox:~$ grep "[bg]zip" dosya.txt
bilg2017@bilg2017-VirtualBox:~$ grep "[bg]zip" dosya.txt
bzip
gzip
zipb gzip
bilg2017@bilg2017-VirtualBox:~$ grep "[^bg]zip" dosya.txt
tugceazip
azip
batzip
bazip
tuazipas
zipzip
bilg2017@bilg2017-VirtualBox:~$ grep "[a-z]2" dosya.txt
tugce2
zip2zip
bilg2017@bilg2017-VirtualBox:~$ grep "[a-fA-F]4" dosya.txt
afA4
bilg2017@bilg2017-VirtualBox:~$

Kısayol Karakter Sınıfları

Kısayol karakteri	Açıklama
\w	**[a-zA-Z0-9_]** için standartlar (kelime karakteri)
\s	Boşluk karakteri,sekmeler veya satır sonları için
\t	Sekmeler için(ASCII 0x09)
\r	Başa dönüş desteği (ASCII 0x0D)
\n	Satır besleme (0x0A)
\xnn	ASCII = nn (\xA9 == @) karakterleri temsil eder

Karakter Sınıfları

- Aşağıdaki karakter sınıfları kullanılabilir;

Sınıf	Açıklama
[[:alnum:]]	Alfanümerik **[a-zA-Z0-9]**
[[:word:]]	Alnum ile altçizgi (\ w) ek olarak aynı [a-zA-Z0-9_]
[[:alpha:]]	Sadece harfler **[a-zA-Z]**
[[:digit:]]	Sadece rakamlar **[0-9]**
[[:blank:]]	Boşluk veya Tab (**\s**)
[[:lower:]]	Yalnızca küçük harfler **[a-z]**
[[:upper:]]	Yalnızca büyük harfler **[A-Z]**
[[:space:]]	space
[[:xdigit:]]	Hex digit **[a-fA-F0-9]**

GENİŞLETİLMİŞ DÜZENLİ İFADELER (Extended Regular Expressions (ERE))

(|) Kullanılması

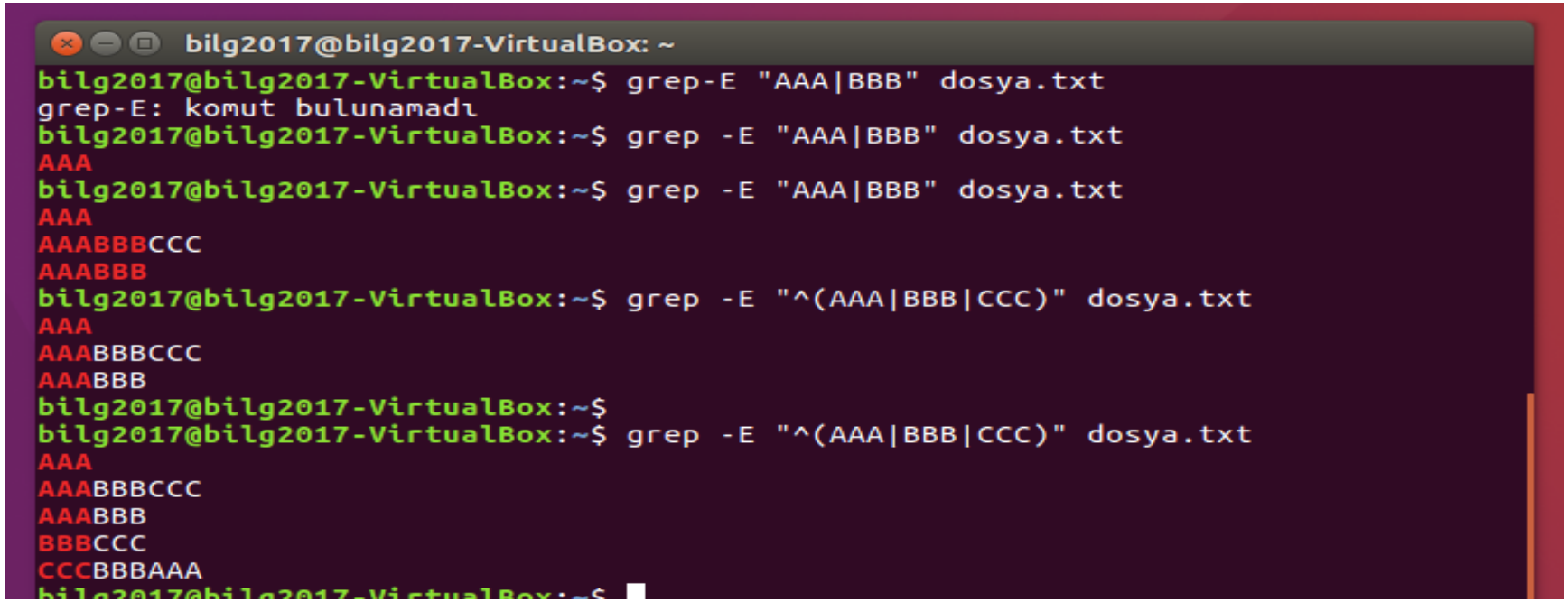
$ grep -E "AAA|BBB" file.txt

Bu AAA veya BBB içeren herhangi bir satırla eşleşir.

- '**()**' değişimi kullanarak normal ifadenin geri kalanından ayırırız.

$ grep –E "^(AAA|BBB|CCC)" file.txt

AAA veya BBB veya CCC ile başlayan herhangi bir satıra uygundur.



Niceleyiciler (*,+, ve ?)

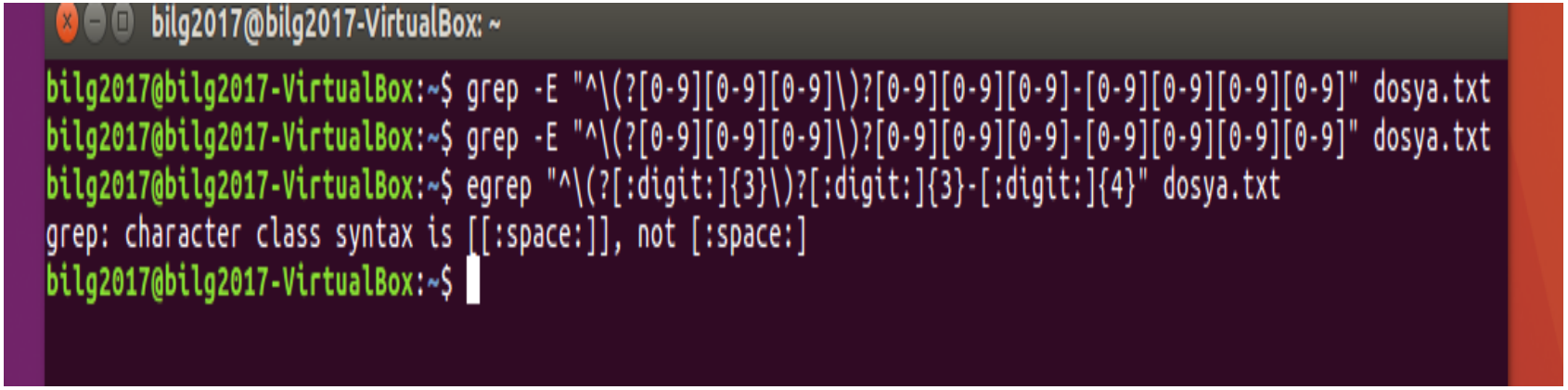
- '**?**' karakteri , önceki elemanın isteğe bağlı (sıfır veya bir kez) olduğunu ifade etmek için kullanılır.
- '*****' karakteri (sıfır veya daha fazla kez) kullanılır
- '**+**' karakteri (bir veya daha fazla kez) kullanılır
- Örneğin,

  Herhangi bir satırı bir telefon numarasıyla başlayarak arıyoruz... bu formatta olabilir

  (nnn) nnn – nnnn veya nnn nnn-nnnn

  $ grep –E "^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]" file.txt
- örnek,
- Hangi satırın uygun bir ifade olduğunu kabul etmek,büyük harfle başlamak,ardından herhangi bir karakter veya boşluk bırakmak veya bir periyotla bitmek istemek;

  $ grep –E "^[[:upper:]][[:upper:][:lower:]]*\.$" file.txt



Eşleme Sayısı ({) ve (})

- ‘**{}**’ belirli bir öğeye belirli sayıda eşleştirmek için kullanılabilir.

Eşleşen Sayısı	Açıklama
{n}	n defa
{n,m}	n to m defa (dahil)
{n,}	n veya daha fazla kez
{,m}	m veya daha az kez

- Örnek : Telefon numarası örneği:

$ egrep “\(?[:digit:]{3}\)?[:digit:]{3}-[:digit:]{4}” file.txt

Örnekler

Örnekler:

$ egrep -i 'Continued\.*' file.txt

$ grep "[0-9a-fxA-FX]abc" file.txt

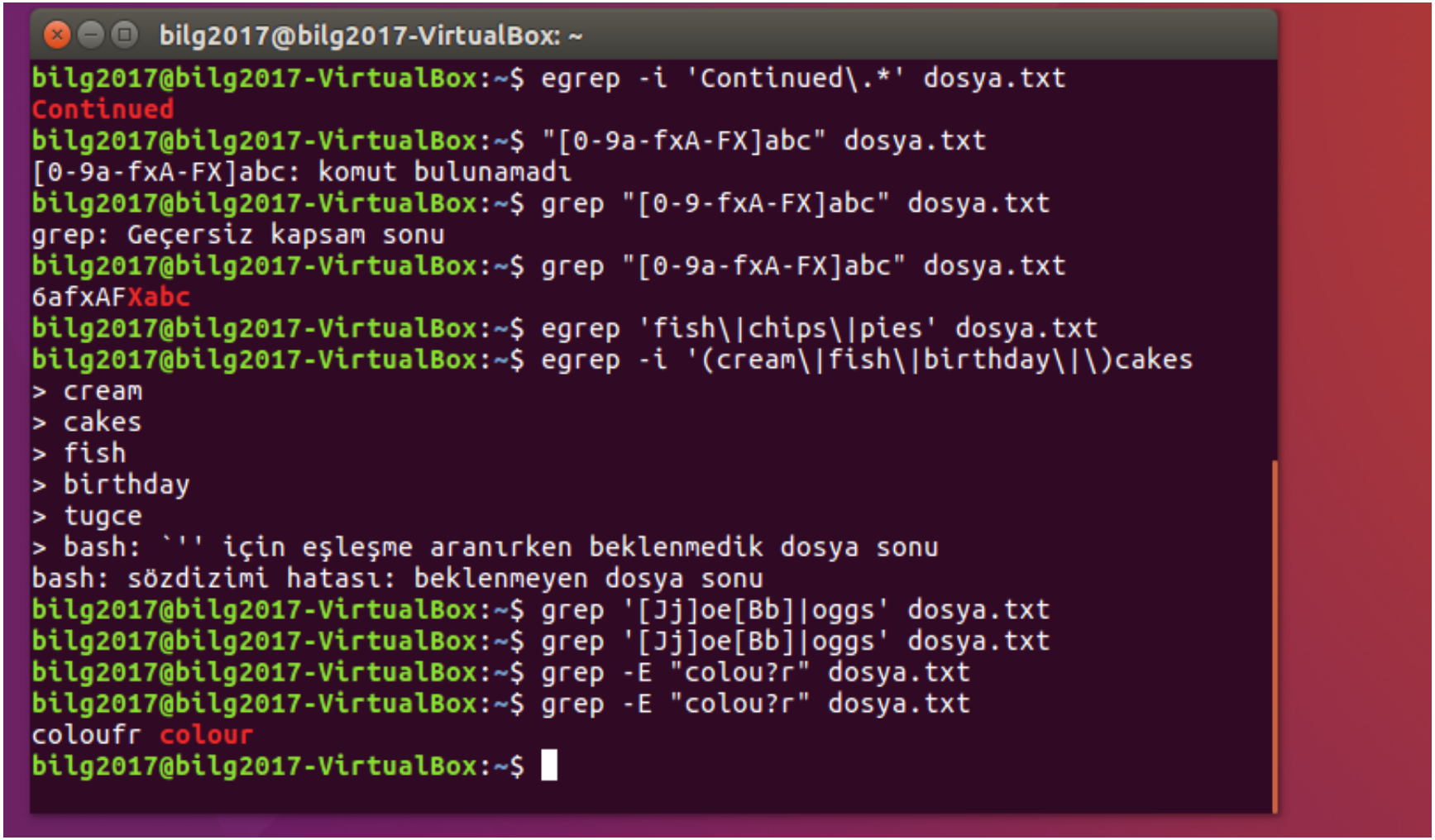
$ egrep 'fish\|chips\|pies' file.txt

$ egrep -i '\(cream\|fish\|birthday\|\) cakes' file.txt

$ grep '[Jj]oe [Bb]loggs' file.txt

$ grep -E "colou?r" file.txt

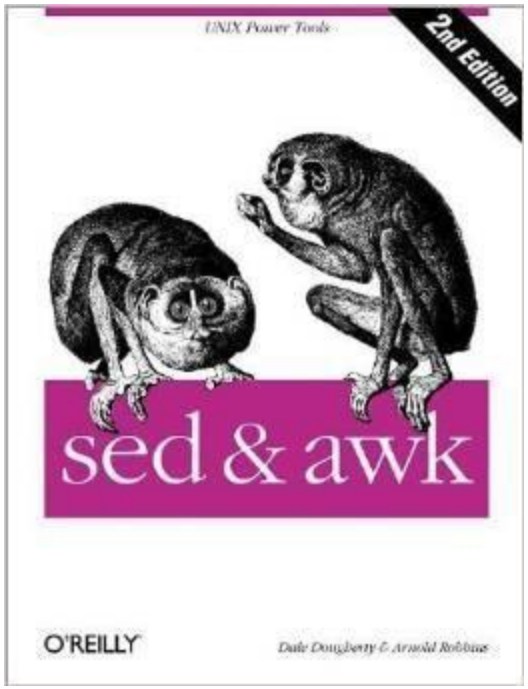
Hatırlatma: Örnek ekran görüntüsü bir sonraki sayfada.


bilg2017@bilg2017-VirtualBox: ~
bilg2017@bilg2017-VirtualBox:~$ egrep -i 'Continued\.*' dosya.txt
Continued
bilg2017@bilg2017-VirtualBox:~$ "[0-9a-fxA-FX]abc" dosya.txt
[0-9a-fxA-FX]abc: komut bulunamadı
bilg2017@bilg2017-VirtualBox:~$ grep "[0-9-fxA-FX]abc" dosya.txt
grep: Geçersiz kapsam sonu
bilg2017@bilg2017-VirtualBox:~$ grep "[0-9a-fxA-FX]abc" dosya.txt
6afxAFXabc
bilg2017@bilg2017-VirtualBox:~$ egrep 'fish\|chips\|pies' dosya.txt
bilg2017@bilg2017-VirtualBox:~$ egrep -i '(cream\|fish\|birthday\|\)cakes
> cream
> cakes
> fish
> birthday
> tugce
> bash: `'' için eşleşme aranırken beklenmedik dosya sonu
bash: sözdizimi hatası: beklenmeyen dosya sonu
bilg2017@bilg2017-VirtualBox:~$ grep '[Jj]oe[Bb]|oggs' dosya.txt
bilg2017@bilg2017-VirtualBox:~$ grep '[Jj]oe[Bb]|oggs' dosya.txt
bilg2017@bilg2017-VirtualBox:~$ grep -E "colou?r" dosya.txt
bilg2017@bilg2017-VirtualBox:~$ grep -E "colou?r" dosya.txt
coloufr colour
bilg2017@bilg2017-VirtualBox:~$

Düzenli ifadeler için diğer komutlar

- Diğer komutların bir çoğu da düzenli ifadelerle çalışır,
  - Dosyaları bulmak için '***find***' ve '***locate***' komutları kullanılır.
  - '***vi***' editorü
  - '***less***' komutu düzenli ifadeleri kullanarak metin araması yapabilir.
  - Düzenli ifadeleri yoğun şekilde kullanan araçlar.
    - ***sed***
    - ***awk***

sed ve awk



SED

- ***sed*** akış editörünü simgeliyor.
- ***sed*** bir program,komut dosyası veya komut satırından metin dosyalarını değiştirmek için kullanılır.
- ***sed*** düzenli ifadeleri kapsamlı biçimde kullanan çok ünlü programlardan biridir.
- ***sed*** zengin bir programdır ve bir çok seçeneğe sahiptir.
- Metin dosyalarının içeriğini değiştirmek için kullanılır

Syntax

- Tüm sed komutları , sed'e aşağıdaki biçimde iletilen bir komut dizesi haline getirilir.

  $ sed 'command string'
- Sed in üzerinde çalıştığı giriş dosyası
  - Giriş yönlendirmesi
  - ***$ sed 'command string' < input-file***
  - Pipes

    $ cat input-file | sed 'command string'
- Çıktı bir dosya da istersek stdout a gider dosyayı yeniden yönlendirmemiz gerekir.

  $ sed 'command string' < input-file > output-file

Yer değiştirme(‘s’ Komutu)

- Metin yerine koymak, sed in en populer komutlarından biridir.
- Belirli bir metin düzeninin tüm tekrarlarını arar(REGEX kullanarak) ve farklı dizge ile değiştirir.
- Örnek:

$ sed ‘s/day/night/’ <old >new

Sed i çalıştır

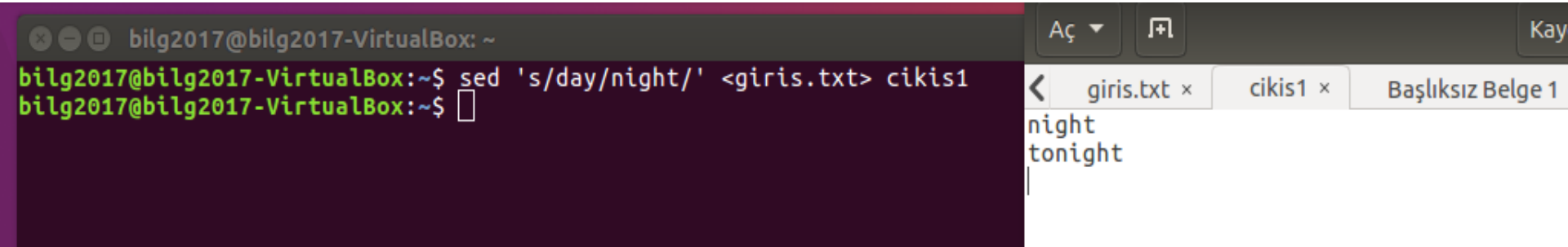
Yer değiştirme komutu

Aranacak model

Yerine koyulacak

Çıktı dosyası

Girdi dosyası



Sınırlayıcı Seçme Karakter

- Sed eğik çizgi,arama deseni ve değiştirme deseni arasındaki ayırıcı(sınırlayıcı) olarak kullanılır.
- Arama deseninde veya değiştirme deseninde bir eğik çizgi kullanmamız gerekiyorsa onu terk etmeliyiz(ters eğik çizgi ile başlayınız)

  $ sed 's/yes\/no/true\/false/' < old-file > new-file

- Başka bir seçenek,başka bir karakteri ayırıcı olarak kullanmaktır.

  $ sed 's:yes/no:true/false:' < old-file > new-file

  $ sed 's|yes/no|true/false|' < old-file > new-file

Genel Seçeneği kullanma

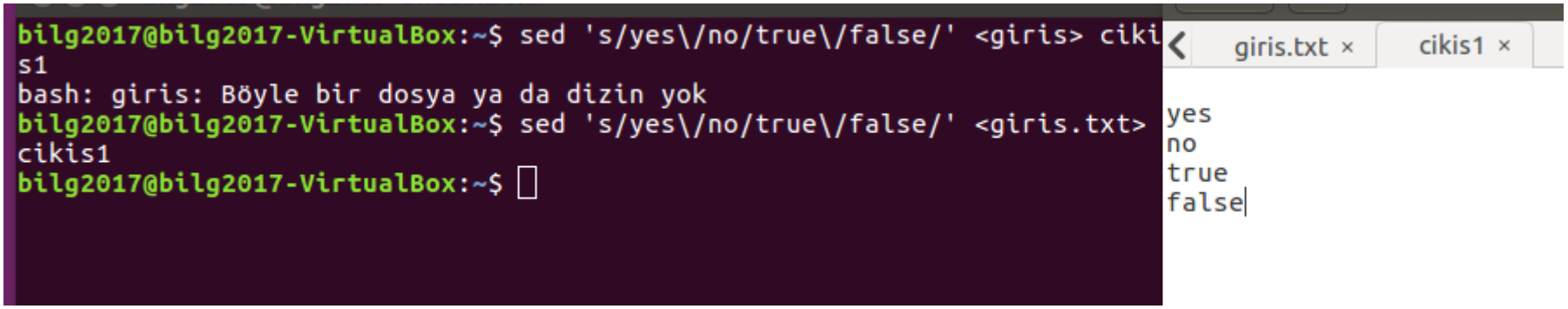
- **sed** dosya satır satırına göre çalışır
- Bu yüzden bu komut

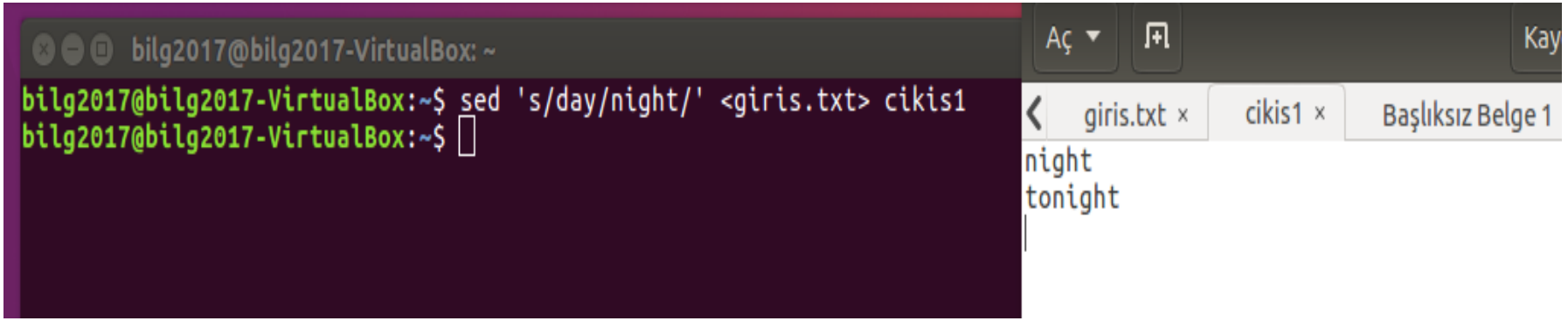
  $ sed ‘s/day/night/’ <old-file >new-file

  - Her satırdaki arama deseninin (day) bir oluşumunu,değiştirme desenine (night)
  - Arama kalıbı aynı satırın içinde birden çok kez mevcutsa,yalnızca ilk ortaya çıkma yerini alacaktır.
- Bunun önlemek için genel bayrağı kullanın

  $ sed ‘s/day/night/g’ <old-file >new-file

Sed komutu ile ilgili denemeler

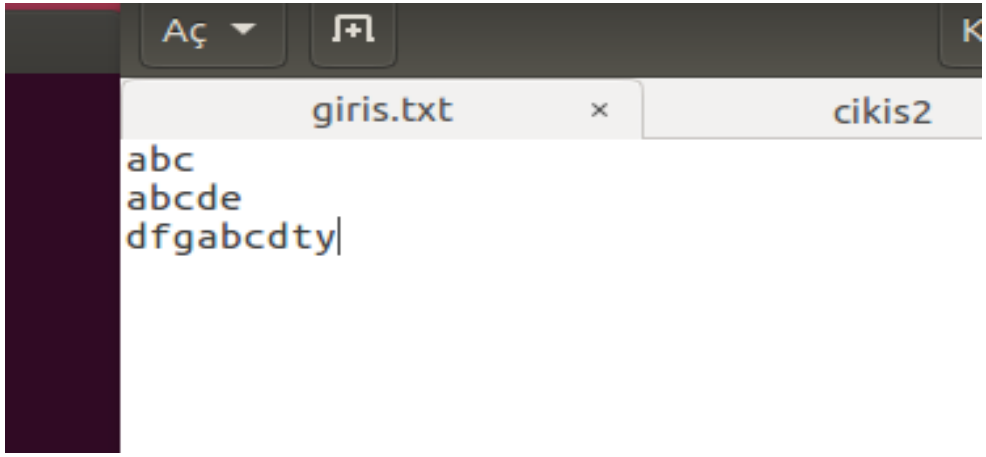


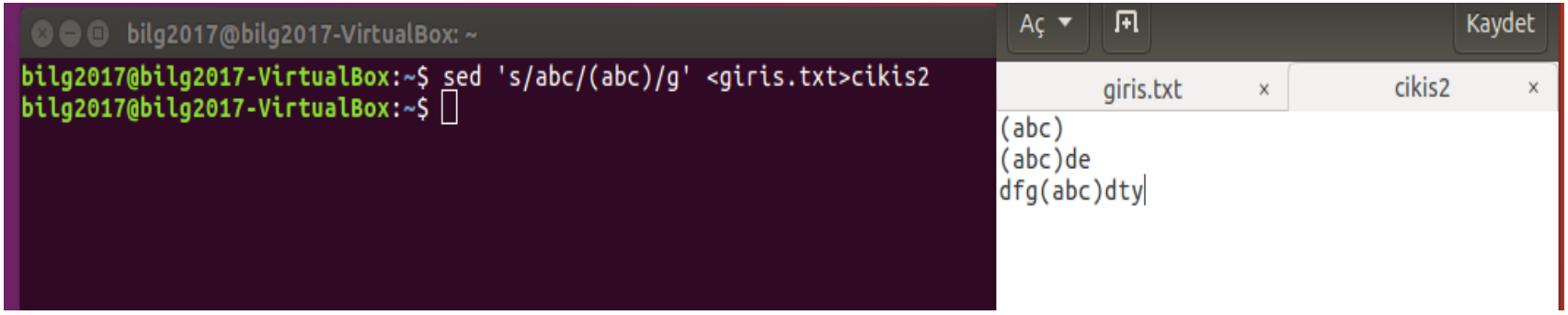


(&) Kullanma

- Bir dizgenin tümüne bakmak istediğimizi ve parantez içine koymak istediğimizi varsayalım.bu komutu kullanırdık.

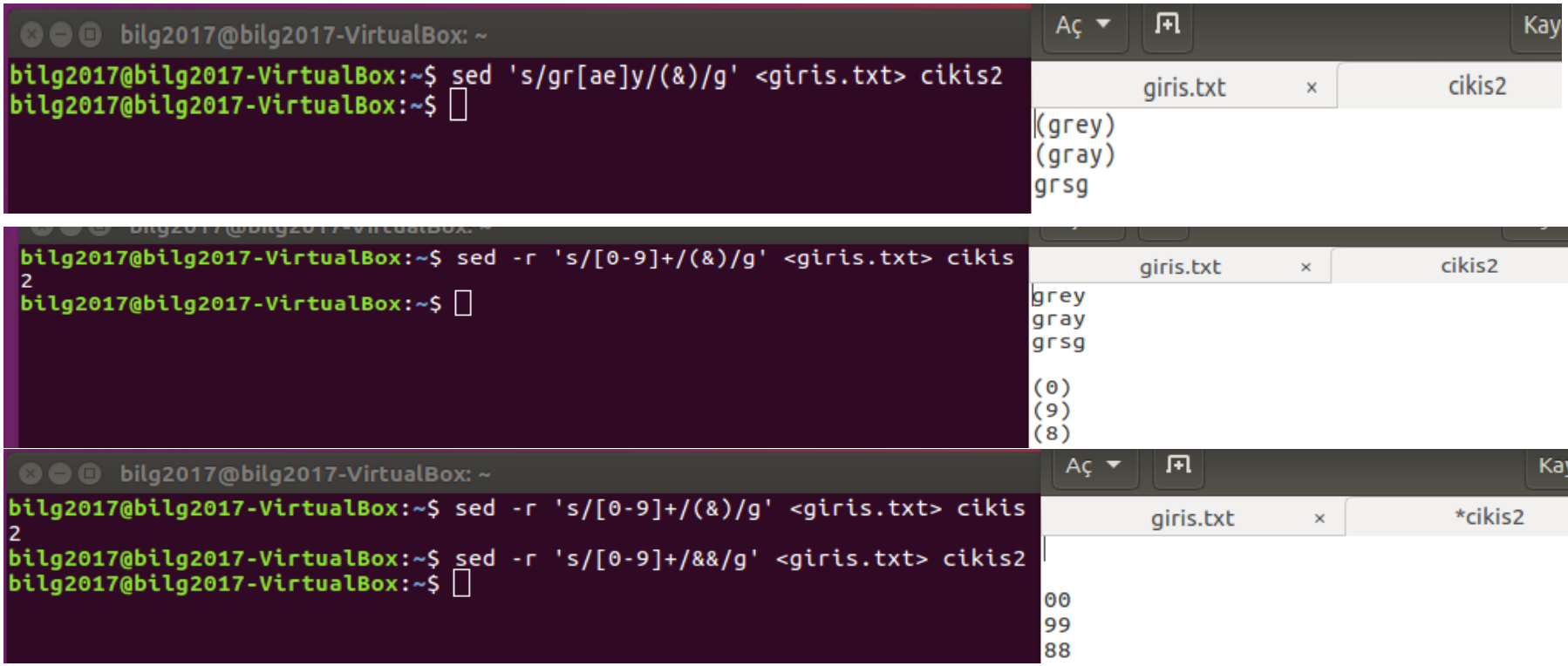
  $ sed 's/abc/(abc)/g' <old-file >new-file





(&) Kullanımı

- Aramıza bir desen kullanmak istersek,meta karakterini & ile eşleşen dizgeyi göstermekiçin kullanabiliriz.
  - ***$ sed 's/gr[ae]y/(&)/g' <old-file >new-file***
  - ***$ sed -r 's/[0-9]+/(&)/g' <old-file >new-file***
- Başka bir örnek dosyadaki herhangi bir sayıyı tekrarlamak istiyoruz.
  - ***$ sed -r 's/[0-9]+/& &/g' <old-file > new-file***
  - Not, son örnekler genişletilmiş REGEX i desteklemek için '***-r***' kullanmaktadır.



awk

- Awk, tablolanmış verileri içeren dosyaları (satır ve sütunlar) işlemek için mükemmel bir araçtır
- Awk ayrıca güçlü dizge işleme işlevlerine sahiptir,bu nedenle belirli dizeleri arayıp çıktıyı değiştirebilir.
- Awk de çok ilişkisel dizeler için destek vermektedir.
- Awk nın "gawk“ adlı bir sürümü vardır.
- “awk” ismi mucitlerinin isimlerinden gelmektedir.

Syntax

- AWK girdi dosyasını satır satır okur
- Her satırda bazı desene dayalı test gerçekleştirir.
- Test başarılı olursa bir işlem gerçekleştirir
- Bir awk ifadesi şöyle görünecek:

  ***pattern* { action }**

- Bazı temel desen,
  - No Pattern: Her satırda eylem gerçekleştirir.
  - ***BEGIN***: Dosya başlangıcı ise eylemi yapın
  - ***END***: Dosya sonu ise eylemi gerçekleştirin

- Örnek :

  BEGIN { print "şimdi dosyayı başlatacağız" }
  { print }
  END { print "Dosya bitti" }

Daha gelişmiş awk

```
BEGIN { print "File\tOwner"}
{ print $9, "\t", $3}
END { print " - DONE -" }
```

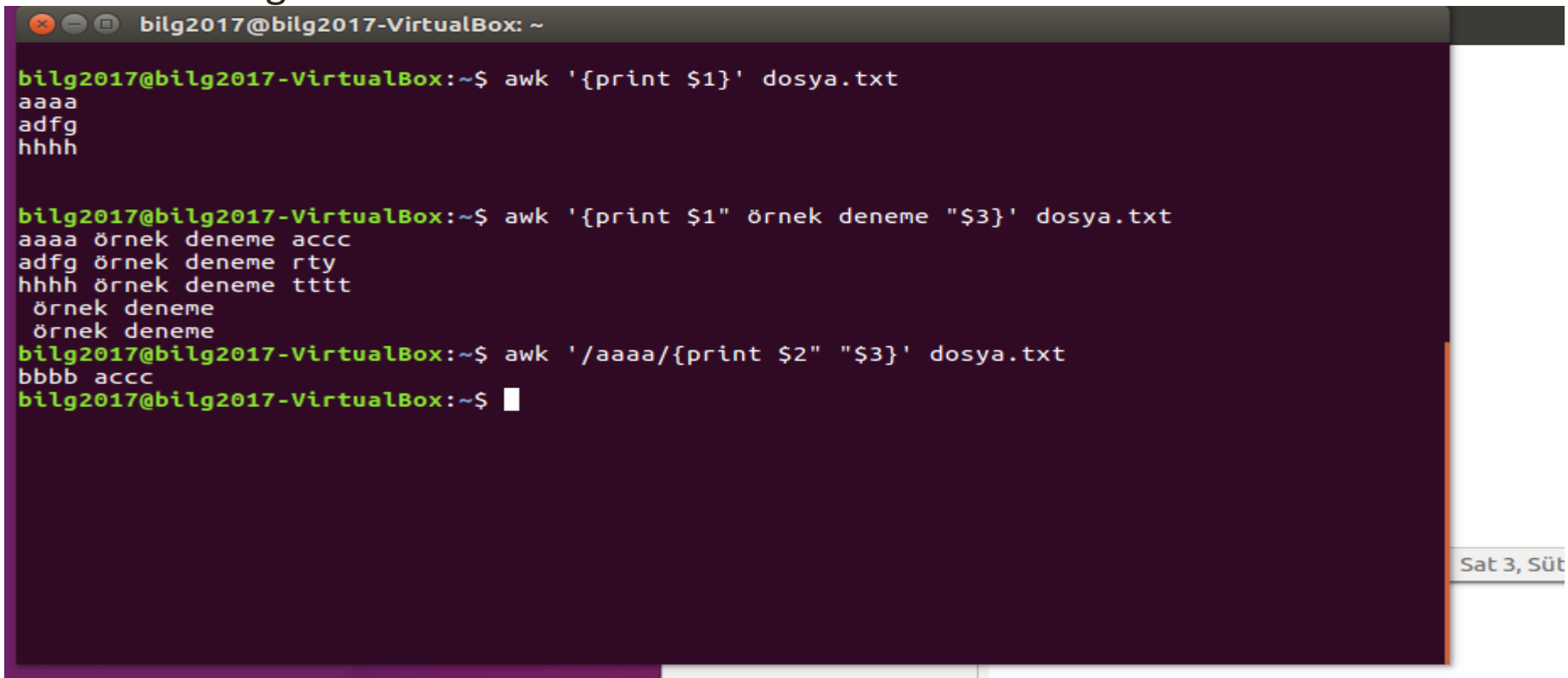
- Bu aşağıdakileri yapacaktır ,
  - Girdi dosyasının ilk satırı okunduğunda komut dosyaları TAB ile ayrıştırılmış “File” ve “Owner” dizelerini çıktılar.
  - Giriş dosyasından okunan her satır için birinci satırda dahil olmak üzere 9. sütundaki ve 3. sütunun verdiği satırlar bir TAB ile ayrışmış olarak yazılır.
  - Girdi dosyası tamamen okunduktan sonra komut dosyaları
  - “- DONE –” yazdırılır.
- Şimdi bu awk deyimlerini şu çıktıya uygularsak :

  $ ls -l

  Tam dosyalar için dosya adlarının ve sahiplerinin adlarının bir listesini yapmalıdır.

AWK

- AWK çok güçlü bir sözdizimi
- Bir bash komut dosyası içinde yazılabilir
- Değişken tanımlayabilir, matematiksel işlemleri gerçekleştirilebilir
- Ayrıca REGEX için güçlü bir destek
- Denediğim bazı örnekler:



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/