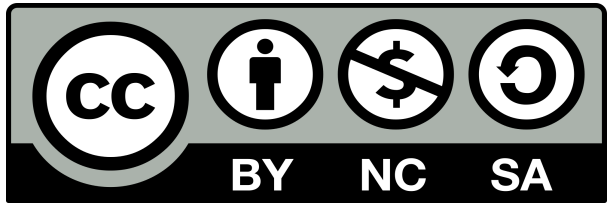

BilkentGNU Linux Eğitimi 2. Hafta

- Text Komutları, I/O Yönlendirmesi ve Pipe

head

- Elinizde uzun bir dosya varsa bunun baştan n adet satırını görüntülemek için kullanılır.

```
bgnu@bilkent:~$ head longtext.txt
```

- Gösterilen satır sayısı için varsayılan değer 10'dur, farklı sayıda satır görüntülemek için "**-n**" seçeneği kullanılır.

```
bgnu@bilkent:~$ head -n 20 longtext.txt
```

tail

- Elinizde uzun bir dosya varsa bunun sondan n adet satırını görüntülemek için kullanılır.

```
bgnu@bilkent:~$ tail longtext.txt
```

- tail komutunda ek olarak "**-f**" seçeneği bulunur, tail ile görüntülediğiniz dosya güncellendikçe eş zamanlı olarak değişimleri görebilirsiniz.

```
bgnu@bilkent:~$ tail –f /var/log/syslog
```

cut

- Bir yazı üzerinde belli kesimler yapmak veya yazının içinden belli bir bölümü almak için kullanılır.

```
bgnu@bilkent:~$ cat cut_sample.txt
Merhaba arkadaşlar, admin alımı var mı?
```

- "**-c**" seçeneği ile yazının içinden istediğiniz karakteri çekebilirsiniz.

```
bgnu@bilkent:~$ cut -c 3 cut_sample.txt
r
```

- "**-f**"(field) seçeneği ile yazının içinden istediğiniz bir alanı çekebilirsiniz.

```
bgnu@bilkent:~$ cut -f 2 -d "," cut_sample.txt
 admin alımı var mı?
```

echo

- Terminal ekranına herhangi bir şey bastırmak için kullanılır.

```
bgnu@bilkent:~$ echo Hello BilkentGNU!
Hello BilkentGNU!
```

Standart Input (stdin)

- Herhangi bir programa girilen veri akışı.(çoğunlukla text)

```
$ echo Hello BilkentGNU!
```

- Burada klavye tarafından "Hello BilkentGNU!" stdin tarafından okunarak programa(echo) iletilir.
- Her programın stdin ile veri okumasına gerek yoktur.

Standart Output (stdout)

- Herhangi bir programın çıktısını yazdığı veri akışı.

```
$ echo Hello BilkentGNU!
Hello BilkentGNU
```

- Bu durumda stdout Terminal ekranı oluyor.

I/O Yönlendirmesi

- Programların stdin veya stdout kaynaklarını değiştirebiliriz.
  - stdout:

```
$ echo Hello BilkentGNU! > hello.txt
```

  - stdin:

```
$ cat < hello.txt > hello2.txt
```

  - Dosyaların üzerine yazmamak için "**>>**" kullanılmalıdır.

```
$ echo Linux Tutorial >> hello.txt
```

Standart Error (stderr)

- Stdout ve stderr birbirinden farklı veri akışlarıdır.

```
$ cat randomfile.txt > test.txt
cat: randomfile.txt: No such file or directory
```

- Bu sebeple stdout yönlendirmesi yaptığımız halde verilen hatayı görebiliriz.
- Veri akışlarını birbirinden ayırmak için tanımlayıcılar kullanılır.
- Dosya tanımlayıcıları
  - stdin: 0
  - stdout: 1
  - stderr: 2

Standart Error Yönlendirmesi

- Stderr yönlendirmesi yapabilmek için ">" seçeneğinden önce dosya tanımlayıcısının belirtilmesi gerekir.(Bu durumda 2)

```
$ cat randomfile.txt 2> test.txt
$ cat test.txt
cat: randomfile.txt: No such file or directory
```

- Stdout yönlendirmesi yaptığınız dosyada bir çok hata mesajı bulunması rahatsız edici olabilir, bu sebeple stderr ve stdout u birbirinden ayırabilmemiz gerekiyor.

```
$ cat randomfile.txt  hello.txt 2>&1 1>test.txt
```

- Stderr çıktılarının tamamen gözden kaybolması için hata mesajlarını "**/dev/null**"' a yönlendirebilirsiniz.

```
$ cat randomfile.txt > test.txt 2> /dev/null
```

Pipe Operasyonu

- Daha önce gördüğümüz stdout ve stdin veri akışlarını birbirine bağlamak için kullanılır.

```
$ cat ls -l | less
```

- İlk komutun çıktısı stdout a gitmek yerine ikinci komuta stdin olarak verilir.

wc

- Dosyada bulunan satır sayısı, kelime sayısı ve byte sayısını gösterir.

```
$ wc longtext.txt
 10     20      500 longtext.txt
```

- Soldan sağa olmak üzere sırasıyla satır sayısı, kelime sayısı, byte sayısı.

sort

- Dosya içeriğini numerik veya alfabetik olarak sıralamk için kullanılır.

```
$ sort sort_sample.txt
```

- Tersten sıralamk için "**-r**", numerik sıralama için ise "**-n**" seçeneği kullanılmalıdır.
- Burada program çıktıyı stdout'a vermektedir, yani "**sort_sample.txt**" dosyasının içeriği değişmedi.
- Sıralanmış dosyayı kaydetmek için stdout yönlendirme bilgimizi kullanalım.

```
$ sort sort_sample.txt > sorted.txt
```

uniq

- Dosya içeriğinde birden fazla bulunan öğeleri ayıklamak için kullanılır.

```
$ uniq uniq_sample.txt
```

- Eğer tekrarlayan öğeler arka arkaya bulunmuyorsa uniq tarafından ayıklanmaz, bu sorunu kolay bir şekilde çözebilmek için pipe bilgimizi kullanalım.

```
$ sort uniq_sample.txt | uniq
```

join

- Aralarında ortak bir numaralandırma sistemi olan iki dosyayı birleştirmek için kullanılır.

```
$ cat quantity.txt
1 50
2 20
3 15
```

```
$ cat products.txt
1 Cips
2 Gofret
3 Su
```

- join komutu çıktıyı stdout'a verdiği için yine stdout yönlendirmesi yapmamız gerekiyor.

```
$ join  products.txt quantity.txt > prodandquant.txt
```

nl

- Dosyanın satırlarını numaralandırır, basit bir işlem gibi gözükse de bir çok alanda işinizi kolaylaştırabilir.
- wc komutu ile satır sayısını öğrenemiyor olsaydık, nl kullanarak bunu öğrenebilirdik.

```
$ nl  longtext.txt | tail -n 1 | cut -f 1
```

- join yapmak istediğimiz fakat ortak olarak numaralandırılmamış dosyaları numaralandırmak için de nl komutu kullanılabilir.

grep

- Herhangi bir dosyanın içinde arama yapmanızı sağlar, açılımı "**Global Regular Expression Parser**"
- Şu anda Regular Expressions konusunu işlemedik fakat grep'e giriş yapıp sonrasında bu konuya değineceğiz.

```
$ grep admin cut_sample.txt
```

- Önceki dersten öğrendiğimiz wildcard'ları hatırlayalım.
  - "?": herhangi bir karakter.
  - "*": herhangi bir karakter(ler).

```
$ grep admin *.txt
```

- grep öğrendiğimiz pipe operasyonu ile sık sık kullanılan bir komut.

```
$ ls -l | grep hello
```

2. Haftanın Sonu

- Tebrikler, BilkentGNU Linux Eğitiminin ikinci haftası sona erdi.
- Bu derste öğrendiğimiz komut ve prensipleri tekrar ederek kafanızda yer etmesini sağlayabilirsiniz.