TEMEL LİNUX KOMUTLARINA GİRİŞ (4)

DERS İÇERİĞİ

1. Düzenli İfadeler
2. Bilgi Alma Komutları
3. Bash Kabuğunda Başlıca Kısa Yollar
4. Süreç Oluşturma ve Sonlandırma

1. DÜZENLİ İFADELER (REGULAR EXPRESSION-REGEX)

- Linux'ta ve birçok Unix benzeri sistemde, düzenli ifadeler **metin desenlerini eşleştirmek** veya **aramak** için kullanılır.
- Bunun haricinde **programlamada** ve **bazı linux kabuk işlemlerinde kullanılır.**
- **Verilerden** istenilen bilgilerin çekilmesi, kullanıcı girdilerinin denetlenmesi vb. **birçok işlem için kullanılır.**
- Sadece arama işleminde değil aranılan ifadelere uygulanacak işlemler için de **düzenli ifadeler kullanılır.**

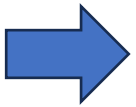
❖ **Düzenli ifadeler, bir dizi özel karakter ve kurallar içerir.**

❖**Bunlar aşağıda özetlenmiştir**.

[ ] : Belirli karakter sınıflarıyla eşleşmeyi sağlar.

Örneğin kullanıcı isimlerinin tutulduğu bir dosyada **A** ile başlayıp **h** ya da **l** ile devam eden isimleri bulmak için

```
root@bilmuh:~# cat users.txt
Ali
Ahmet
Arif
Zeynep
Turan
A123
A345
```



```
root@bilmuh:~# grep A[hl] users.txt
Ali
Ahmet
```

❖Sadece metinlerden oluşan bir arama yaptırılmak istenirse:

```
root@bilmuh:~# grep A[a-z] users.txt
Ali
Ahmet
Arif
```

❖Sadece rakamlardan oluşan bir arama yaptırılmak istenirse:

```
root@bilmuh:~# grep A[0-9] users.txt
A123
A345
```

[^] : Kümenin içerisindeki karakter haricinde (dışında) bir karaktere karşılık gelir.

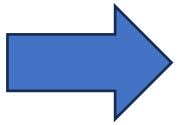
```
root@bilmuh:~#  grep A[^0-9] users.txt
Ali
Ahmet
Arif
```

=

```
root@bilmuh:~# cat users.txt | grep A[^0-9]
Ali
Ahmet
Arif
```

. : Nokta ile başlayan yere herhangi bir karakter geleceğini ifade eder.

```
root@bilmuh:~# cat liste.txt
a216test
a2t
araba
16araba
123test
kalem
kaleci
kelam
asd123son
```



```
root@bilmuh:~# cat liste.txt | grep a.t
a2t
```

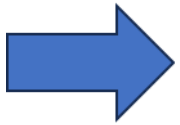
***: Herhangi bir sayıda herhangi bir karaktere karşılık gelir.**

^: Satırın başına karşılık gelir.

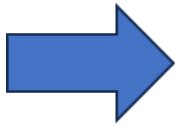
$: Satırın sonuna karşılık gelir.

Örnek 1:

```
root@bilmuh:~# cat liste.txt
a216test
a2t
araba
16araba
123test
kalem
kaleci
kelam
asd123son
```



```
root@bilmuh:~# cat liste.txt | grep ^a2
a216test
a2t
```



```
root@bilmuh:~# cat liste.txt | grep test$
a216test
123test
```

Örnek: Satır başı **a2** ile **başlayan** ve **devamında herhangi bir harf** ya da **karakterler** geçen ve **t** ile **sonlanan satırları bulur.**

```
root@bilmuh:~# cat liste.txt | grep ^a2.*t$
a216test
a2t
```

\ : Özel karakterlerin normal karakter olarak algılanmasını sağlar.

\ |: Kendinden bir **önceki** veya **bir sonraki karaktere karşılık gelir.**

\?: Kendisinden **önceki karakterin 0** ya da **bir kez bulunduğunu** gösterir.

\+: Kendisinden **önceki karakterin 1** ya da **daha fazla bulunduğunu** gösterir.

\(..\): Grup olarak düzenli deyimleri tanımlar.

\{n\} : Kendisinden önceki karakterin **n kez tekrarlandığını gösterir.**

\{n,m\}: Kendisinden önceki karakterin **en az *n* en çok *m*** kez **tekrarlandığını gösterir.**

Örnek: liste2.txt dosyasında **zzz** ile **başlayan satırların bulunması**

```
root@bilmuh:~# cat liste2.txt | grep ^zzz
zzz nnbnmn
zzz 67869 lkşlkkl
```

=

```
root@bilmuh:~# cat liste2.txt | grep "^z\{3\}"
zzz nnbnmn
zzz 67869 lkşlkkl
```

```
root@bilmuh:~# cat liste2.txt | grep  "\<zzz"
zzz nnbnmn
zzz 67869 lkşlkkl
```

Örnek : Kendisinden **önce** veya **sonra** ***opera*** kelimesinin geçtiği satırların listelenmesi

```
root@bilmuh:~# cat liste2.txt | grep  "\opera\+"
opera 192.103.212.15
192.103.212.15 opera
```

❖ sed (Stream Editor)

- **Metin üzerinde işlem yapan bir programdır.**
- Genel olarak programların **çıktılarını düzenlemek** veya **pek çok dosyayı otomatik olarak düzenlemek** için **kullanılır.**
- Metin üzerinden **tek geçişte istenen kriterlere uygun değişiklikleri yapabildiğinden** oldukça **etkin** ve **hızlı bir araçtır.**
- **Kullanımı:**

```
$ sed [seçenekler] dosya […]
```

sed komutunun birden çok parametresi vardır:

-n: Sadece belirtilen satırlara uygulama yapar.

-e: Bir sonraki komutun bir düzenleme komutu olur.

-f: Bir sonraki ifade bir dosya adı (yani dosyadan girdi alacaksa) olur.

-p: Yazdırma

-d: Silme

-i: dosyaları yerinde düzenler. Yani kaydeder.

❖Bu parametrelerin tamamına –help komutu ile erişilebilir.

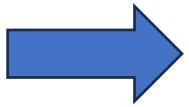
❑**sed komutu,**

s/ değişecek ifade / yerine gelecek ifade /g

şeklinde yazılarak **metin dosyası içerisinde ifadeler değiştirilebilir.**
g eklenmezse dosyanın tamamı yerine **her satırdaki kalıp aranır.**

Örnek: Bir metin dosyasındaki kelimeleri bulup değiştirsin.

```
root@bilmuh:~# cat a1.txt
bir iki bir iki üç dört
iki üç beş bir
üç beş yedi bir bir on
bir beş yedi bir on beş
```



```
root@bilmuh:~# sed 's/bir/BIR/'<a1.txt
BIR iki bir iki üç dört
iki üç beş BIR
üç beş yedi BIR bir on
BIR beş yedi bir on beş
```

- Yukarıda her bir satırda ilk bulunan bir (pattern) kelimesi değiştirilerek BİR yapılmıştır.

❖ Dosyanın tamamında değişiklik yapılacaksa aşağıdaki gibi bir komut verilmesi gerekir.

```
root@bilmuh:~# sed 's/bir/BIR/g'<a1.txt
BIR iki BIR iki üç dört
iki üç beş BIR
üç beş yedi BIR BIR on
BIR beş yedi BIR on beş
```

```
root@bilmuh:~# sed 's/bir/BIR/g' -i a1.txt
root@bilmuh:~# cat a1.txt
BIR iki BIR iki üç dört
iki üç beş BIR
üç beş yedi BIR BIR on
BIR beş yedi BIR on beş
```

❖ **Sonucu ekrana yazdırdı! Dosyada değişiklik olmadı!**

❖ **Dosyada değişiklik oldu.**

❖ sed komutu düzenli ifadeler ile birlikte kullanılabilir.

❖ **Örneğin Satır başındaki boşlukları (SPACE ve Tab) silmek için:**

```
$ cat space.txt
 satir 1
    satir 4
satir   3
$ sed 's/^[ \t]*//' space.txt
satir 1
satir 4
satir   3
```

❖ awk

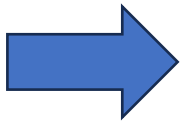
- **Güçlü bir desen arama aracıdır.**
- **sed** gibi **benzerlerinden ayıran özellik** ise **matematik işlemler**, **şartlı ifadeler**, **değişkenler** ve **dosya G/Ç işlemleri** gibi script dillerinde olan **temel işlevlere sahip olmasıdır.**
- **Genelde yapılandırma dosyalarını, okumak** veya **komut çıktılarını işlemek için kullanılır.**
- Dosyalarla işlem yapmak için **– f** parametresi kullanılır.

❖Çağrılma şekli:

```
$ awk -f awk_script dosya
$ awk desen {aksiyon} dosya
```

Örnek 1 : Belirtilen dosyadaki **2** ve **3** numaralı **sütunu ekrana basmak için** (sütunlar $2 ve $3 ile belirtildi):

```
root@bilmuh:~# cat a1.txt
BIR iki BIR iki üç dört
iki üç beş BIR
üç beş yedi BIR BIR on
BIR beş yedi BIR on beş
```



```
root@bilmuh:~# awk <a1.txt '{print $2 , $3}'
iki BIR
üç beş
beş yedi
beş yedi
```

Örnek 2 : Bir ile başlayan satırların bulunması için aşağıdaki komut yazılır.

```
root@bilmuh:~# awk '/^BIR/' a1.txt
BIR iki BIR iki üç dört
BIR beş yedi BIR on beş
```

❖ **awk komutu ile çok daha özel ve karmaşık aramalar yapılabilir.**

2. BİLGİ ALMA KOMUTLARI

❖lsb_release : Kullanılan linux dağıtımını öğrenmek için kullanılır.

```
root@bilmuh:~# lsb_release  -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 12 (bookworm)
Release:        12
Codename:       bookworm
```

❖Sistemin sadece adını öğrenmek için **cat /etc/*release** kullanılır.

```
root@bilmuh:~# cat /etc/issue
Debian GNU/Linux 12 \n \l
```

❖Sistemle ilgili daha ayrıntılı bilgi almak içim **cat /etc/*release** kullanılır.

```
root@bilmuh:~# cat /etc/*release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

❖ uname

❖Kullandığı **çekirdek (Kernel) versiyonunu** öğrenmek için kullanılır.

```
root@bilmuh:~# uname
Linux
root@bilmuh:~# uname -a
Linux bilmuh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64
GNU/Linux
```

❖ hostname

Sunucunun adını yani host adını verir.

```
root@bilmuh:~# hostname
bilmuh
```

❖ who-whoami

❖Sistemde kim aktif, kim login olmuş, o an hangi kimlikle çalışılmakta gibi **bilgileri görmek için aşağıdaki komutlar kullanılır.**

- **who**, Sistemde hangi kimlikle giriş yapıldığı

```
root@bilmuh:~# who
seckin   tty2         2023-11-18 18:00 (tty2)
```

- **whoami,** sistemde o anda hangi kullanıcı ile çalışıldığını verir.

b parametresi **en son ne zaman girildiği,** ***r*** parametresi **açılış seviyesini verir.**

```
root@bilmuh:~# whoami
root
```

```
root@bilmuh:~# who -b
         sistem önyüklemesi 2023-11-18 17:56
```

```
root@bilmuh:~# who -r
        açılış_düzeyi 5 2023-11-18 17:56
```

- **w,** hangi kullanıcı o anda hangi uygulamaları/komutu çalıştığını gösterir.

```
root@bilmuh:~# w
 19:47:04 up  1:50,  1 user,  load average: 0,00, 0,03, 0,00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
seckin   tty2     tty2             18:00    1:50m  0.02s  0.02s /usr/libexec/gnome-ses
```

❖CTRL+D

- Bir kullanıcıdan çıkış yapmak ya da sistemi kapatmak için kullanılır.

❖ uptime

- Sistemin ne kadar süre açık olduğunu verir.

```
root@bilmuh:~# uptime
 19:55:00 up  1:58,  1 user,  load average: 0,01, 0,02, 0,00
```

❖ date

- Sistemin o anki tarih ve saat bilgisini verir.

```
root@bilmuh:~# date
Cts 18 Kas 2023 19:58:14 +03
```

❖ vmstat

- Sistemin genel durumunu görmek için

```
root@bilmuh:~# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 2262396  31908 787124    0    0    22    11   23   17  0  0 100  0  0
```

❖ stat

- **Dosya** ve **dizin durumu** hakkında **bilgi almak için kullanılır.**

```
root@bilmuh:~# stat /root
 Dosya: /root
 Boyut: 4096            Bloklar: 8          Kimlik bloku: 4096   dizin
Aygıt: 8,1      İndeks: 1046529     Bağlar: 7
Erişim: (0700/drwx------)  Uid: (    0/    root)   Gid: (    0/    root)
    Erişim: 2023-11-13 02:45:19.146514640 +0300
Değiştirme: 2023-11-13 03:02:05.167863495 +0300
Değişiklik: 2023-11-13 03:02:05.167863495 +0300
     Doğum: 2023-10-14 23:27:38.177479029 +0300
root@bilmuh:~# stat -f  /root
 Dosya: "/root"
Kimlik: 8d226756d58fdb9b Ad uzunluğu: 255     Tür: ext2/ext3
Blok boyutu: 4096       Temel blok boyutu: 4096
Bloklar: Toplam: 7450336    Boş: 6190965    Kullanılabilir: 5806172
İndeksler: Toplam: 1905008    Boş: 1742347
```

❖ which

- Parametre olarak verilen **bir komutun tam yolunu verir.**

```
root@bilmuh:~# which cat
/usr/bin/cat
root@bilmuh:~# which ls
/usr/bin/ls
```

❖ whereis

- Bir dosyayla ilgili çalıştırılabilir dosyanın, **kaynak** ve **yardım dosyalarının konumunu öğrenmek için kullanılır.**

```
root@bilmuh:~# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
root@bilmuh:~# whereis cat
cat: /usr/bin/cat /usr/share/man/man1/cat.1.gz
root@bilmuh:~# whereis -b cat
cat: /usr/bin/cat
```

❖ dmidecode

- Sistemin **donanım bileşenlerini** ve **BIOS ile ilgili bilgileri göstermek** için kullanılır.

```
root@bilmuh:~# dmidecode
# dmidecode 3.4
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 456 bytes.
Table at 0x000E1000.

Handle 0x0000, DMI type 0, 20 bytes
BIOS Information
        Vendor: innotek GmbH
        Version: VirtualBox
        Release Date: 12/01/2006
        Address: 0xE0000
```

❖ history

- Bir kullanıcının yazmış olduğu komut geçmişini verir.

```
root@bilmuh:~# history
    1  su man
    2  su -help
    3  clear
    4  exit
    5  lsb_release
    6  lsb_release  -a
    7  cat /etc/*release
```

↑ ↓

* Komut geçmişini temizlemek için

```
root@bilmuh:~# history -c
```

- Bir kullanıcının önceki yazmış olduğu komutları görmek için kullanılır.

❖ clear

- Ekranı temizler. Komutlar ve yazılanlar sadece ekranda gözükmez. Burada komut geçmişi, history komutu ile geri getirilir.

❖ fdisk –l : Sistemdeki disk bölümlerini gösterir.

```
root@bilmuh:~# fdisk -l
Disk /dev/sda: 30 GiB, 32212254720 bayt, 62914560 sektör
Disk model: VBOX HARDDISK
Birimler: sektör'i 1 * 512 = 512 baytın
Sektör boyutu (montıksal/fiziksel): 512 bayt / 512 bayt
G/Ç boyutu (en düşük/en uygun): 512 bayt / 512 bayt
Disketikeri tipi: dos
Disk belirleyicisi: 0xbb143735

Aygıt      Açılış Başlangıç      Son   Sektör Boyut ld Türü
/dev/sda1  *           2048 60913663 60911616   29G 83 Linux
/dev/sda2          60915710 62912511  1996802  975M  5 Ek
/dev/sda5          60915712 62912511  1996800  975M 82 Linux takas / Solaris
```

❖ Proc Sanal Dosya Sisteminden Bilgi Almak

❖**CPU** ve **RAM** hakkında ayrıntılı bilgi edinmek için sanal dosya sistemi olan **proc** kullanılabilir.

❖**CPU için cat /proc/cpuinfo**

```
root@bilmuh:~# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 141
model name      : 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz
stepping        : 1
cpu MHz         : 2496.002
cache size      : 24576 KB
```

❖ **Hafıza için cat /proc/meminfo**

```
root@bilmuh:~# cat /proc/meminfo
MemTotal:        4008672 kB
MemFree:         2458272 kB
MemAvailable:    2944416 kB
Buffers:           33072 kB
Cached:           655164 kB
SwapCached:            0 kB
Active:           445832 kB
Inactive:         880420 kB
Active(anon):       1372 kB
Inactive(anon):   655456 kB
Active(file):     444460 kB
```

3. BASH KABUĞUNDA BAŞLICA KISA YOLLAR

❖ CTL+C

- Bir komuttan veya sürekli çalışan veya herhangi bir hata ile karşılaşıldığında çıkış yapmak için kullanılır.

```
root@bilmuh:~# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.059 ms
^C
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3900ms
rtt min/avg/max/mdev = 0.020/0.047/0.059/0.016 ms
```

❖ CTL+D

- **exit** komutu ile aynıdır. Terminali sonlandırır. **Bir kullanıcıdan çıkış yapar.**

❖ CTL+L

- Clear komutu ile aynıdır. Ekranı temizler.

❖ Tab Tuşu

- Otomatik tamamlama yapar.

❖ CTL+Z

- **Çalışan uygulamayı arkaplana atar**. **fg** komutu ile ön plana alınır. Bir kullanıcıdan çıkış yapar.

❖ <CTL+ALT><Fn> n=1,2,3,...6 tuşları

- Birden fazla sanal terminal açılmasını sağlar. 6 tane sanal terminal açılır.

❖ CTL+ALT+F7

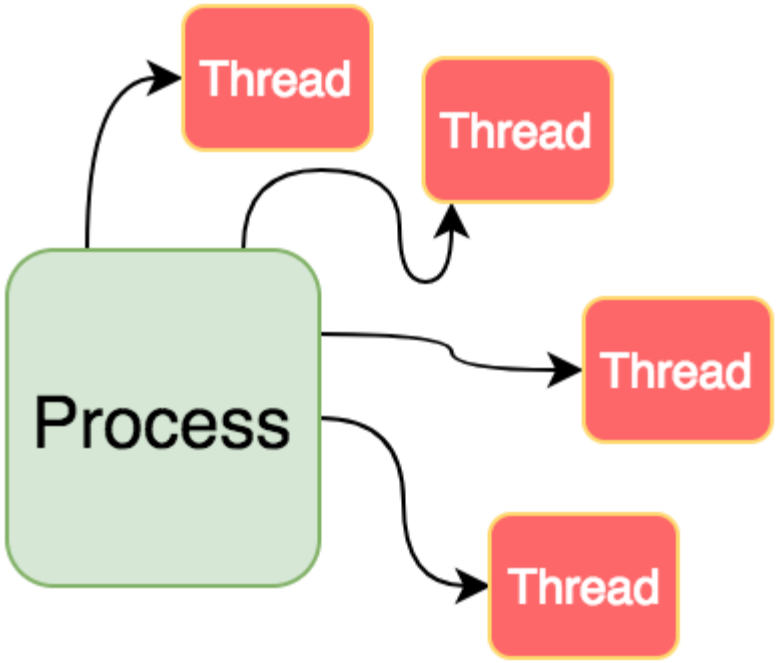
- Açılan bir oturuma geri dönmek için kullanılır.

4. ÇALIŞAN SÜREÇLER

- **Süreç (process)** genel tanımla **çalışan bir programdır.**
- **Süreçler,** komut satırından çalıştırılan kısa süreli komut olabileceği gibi **işletim sisteminin açık olduğu süre** boyunca çalışan bir ağ servisi de olabilir.
- **Her sürece ait bir PID numarası vardır.**
- Süreçle ilgili yapılacak işlemler bu PID numarası üzerinden gerçekleştirilir.
- Linux altında çalışan ilk süreç, **init olup PID numarası** her zaman **1**'dir.

```
  PID TTY          TIME CMD
 2676 pts/0    00:00:00 su
 2681 pts/0    00:00:00 bash
 2740 pts/0    00:00:00 ps
```

- **Init ilk süreç olduğu** için kullanıcı tarafından başlatılan diğer süreçlerin aksine çekirdek tarafından başlatılır.
- **Bazı süreçler eş zamanlı çalışan alt thread'lere sahip olabilir.**
- Thread'ler aynı **PID altındadır**, kendilerine ait bir PID numarası olmaz.
- Örneğin **Mozilla Firefox**'ta **bir thread**, **DNS sorgularını çözerken diğer thread sunucuyla konuşur.**



❖ ps

❖Çalışan kendine ait süreçleri görüntülemek için temel olarak **ps komutu kullanılır.**

```
  PID TTY          TIME CMD
 2676 pts/0    00:00:00 su
 2681 pts/0    00:00:00 bash
 2740 pts/0    00:00:00 ps
```

❖**Sistemdeki çalışan tüm süreçleri görüntülemek için kullanılır.**

```
root@bilmuh:~# ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 167936 12500 ?        Ss   21:29   0:00 /sbin/init
root           2  0.0  0.0      0     0 ?        S    21:29   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   21:29   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   21:29   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   21:29   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   21:29   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   21:29   0:00 [kworker/0:0H-event
root          10  0.0  0.0      0     0 ?        I<   21:29   0:00 [mm_percpu_wq]
```

-a: Bütün süreçler

-u: Belirtilen kullanıcıya ait süreçler

-t: Belirtilen terminale ait kullanıcılar

❖ ps –u seckin

```
root@bilmuh:~# ps -u seckin
    PID TTY          TIME CMD
   1104 ?        00:00:00 systemd
   1106 ?        00:00:00 (sd-pam)
   1121 ?        00:00:00 pipewire
   1124 ?        00:00:00 wireplumber
   1127 ?        00:00:00 pipewire-pulse
   1135 ?        00:00:00 dbus-daemon
   1137 ?        00:00:00 gnome-keyring-d
   1139 ?        00:00:00 gvfsd
   1151 ?        00:00:00 gvfsd-fuse
```

❖**Çalışan süreçleri, ağaç yapısı şeklinde göstermek** için aşağıdaki komut **kullanılır.**

```
$ ps axjf
```

```
    1  2635  2635  2635 ?           -1 Ss       0   0:00 /usr/sbin/sshd
 2635  6077  6077  6077 ?           -1 Ss       0   0:00  \_ sshd: simsek [priv]
 6077  6081  6077  6077 ?           -1 S      501   0:00      \_ sshd: simsek@pts/0
 6081  6082  6082  6082 pts/0     6207 Ss     501   0:00          \_ -bash
 6082  6207  6207  6082 pts/0     6207 R+     501   0:00              \_ ps axjf
    1  2653  2653  2653 ?           -1 Ss       0   0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
    1  2669  2669  2669 ?           -1 SLs     38   0:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
```

❖ PSTREE

❖Çalışan kendine ait süreçleri ağaç yapısı şeklinde göstermek için kullanılır.

```
root@bilmuh:~# pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───2*[dbus]
        ├─dbus-daemon
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b───3*[{gnome-sessi+
        │      │                 │                 └─2*[{gdm-wayland-ses}]
        │      │                 └─2*[{gdm-session-wor}]
        │      └─2*[{gdm3}]
        ├─low-memory-moni───2*[{low-memory-moni}]
```

❖ top

❖**Süreçlerin anlık değişimleri ile birlikte verir.**

❖**Çeşitli alt parametreleri vardır. Bunlar:**

-d: Değerlerin güncellenme aralığını belirlemek için

-p PID: Yalnızca verilen PID numarasına sahip süreci izlemeye alır.

-q: Değerler herhangi bir bekleme olmadan güncellenir.

-C: Birden fazla işlemci olan sunucularda tek tek CPU değerini göstermek yerine toplam CPU değerlerini gösterir.

-c: Sadece süreç adını değil tam komut satırı parametrelerini gösterir.

-H: Thread'leri göstermek için

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1260 seckin    20   0 4548944 269200 129488 S   2,3   6,7   0:10.67 gnome-shell
 2053 seckin    20   0  550044  51408  38868 S   0,3   1,3   0:01.24 gnome-terminal-
    1 root      20   0  167936  12500   9248 S   0,0   0,3   0:00.61 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
```

❖top –d 10 yazılına komutun süreçler her 10 saniyede bir gösterir.

❖**Varsayılan değer, 3 saniyedir.**

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1260 seckin    20   0 4548960 269076 129488 S   0,2   6,7   0:11.70 gnome-shell
    1 root      20   0  167936  12500   9248 S   0,0   0,3   0:00.61 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
    5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 slub_flushwq
```

❖top –p PID

- Yalnızca verilen bir süreci izlemek için kullanılır.
- Aşağıda 1260 numaları süreci izlemek için top –p 1260 komutu çalıştırılmıştır.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1260 seckin    20   0 4551040 269092 129488 S   0,7   6,7   0:13.68 gnome-shell
```

❖pgrep

- Hafızada çalışan süreçlerin belirli kriterlere göre listelenmesi için kullanılır.

Örnek: http ile ilgili yürütülen süreçleri görmek için

```
# pgrep http
3077
3531
3532
3533
3534
```

Örnek: user4, kullanıcısına ait yürütülen süreçleri görmek için

```
# pgrep -u user4
6081
6082
```

❖free

- Kullanılan bellek miktarını gösterir.
- Aynı zamanda swap kullanımı hakkında da bilgi verir.

```
root@bilmuh:~# free
               total        used        free      shared  buff/cache   available
Mem:         4008672     1073192     2458352       13548      717312     2935480
Swap:         998396           0      998396
```

❖nohup

- Verilecek komutları herhangi bir kesintiye uğramadan çalıştırmayı sağlar.
- Bütün sinyalleri devre dışı bırakarak **çalışan sürecin sinyallerden etkilenmesini engeller.**
- **Kullanımı: nohup komut &**
- Örneğin çok uzun sürecek bir işlem arkaplana atılmıştır.

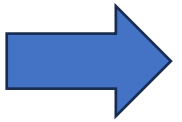
```
# nohup find / -xdev -type f -perm +u=s -print &
[1] 19567
nohup: appending output to `nohup.out'
```

- Nohup komutu sadece arka planda sinyallere maruz kalmadan çalışmayı engeller.

❖jobs

- Arkaplana gönderilmiş görevleri görmek için kullanılır.

```
# jobs
[1]   Running                 bash script.sh &
[2]-  Running                 sample-soft &
[3]+  Done                    browser .
```



Yürütülen görevlerin sırası, [1],[2], [3], … şeklinde gösterilmektedir.
Bu numaralar bir görev veya süreç numarası değildir.

❖ fg

- Arkaplanda çalışan bir komutu geri almak (ön plana) çıkarmak için kullanılır.

```
root@bilmuh:~# jobs
[1]+  Çalışıyor               sleep 30 | echo "merbaba" &
root@bilmuh:~# fg %1
sleep 30 | echo "merbaba"
```

4.SÜREÇ OLUŞTURMA VE SONLANDIRMA

Süreçler iki şekilde oluşturulur:

1. **Çatallanma (fork):**
2. **Yeni süreç başlatma (exec):**

1. Çatallanma (fork)

- **Ana süreç kendi kopyasını çıkartarak yeni iş bu kopyaya yaptırtılır.**
- **İlk çalışan süreç olan init kendi kopyasını çıkartarak yeni işleri bu kopyaya yaptırtır.**
- Bu şekilde bir ağaç yapısı şeklinde yeni süreçler oluşturulur.

2. Yeni süreç başlatma (exec)

- **exec komutuyla yeni bir süreç başlatılır.**
- Bu süreç ana sürecin yerini alır.
- Komut satırından **exec ile bir komut çalıştırıldığında**, **komut bitince tekrar komut satırına dönülmez.**
- Çünkü **yeni süreç, ana sürecin yerini almıştır**, **artık ana süreç kalmamıştır.**

- Yeni süreç başlatmaya örnek olarak **komut satırından exec komutuyla ssh komutunu çalıştırmak verilebilir.**
- **Karşı sunuya bağlantı kuran ssh komutu, exec ile çalıştırıldığından kendi ana süreci olan kabuğun yerini almıştır.**
- **Karşı taraftan exit yapılıp ssh sonlandırıldığında konsol penceresi kapanır.**
- **Bash kabuğu komut** çalıştırırken **fork + exec** yapar.
- **Yani kendisinin bir klonunu çıkartır ve bu klon exec yaparak yeni süreci başlatır.**
- **Yeni süreç klonun yerini alır.**

- **Ana süreç oluşturduğu alt sürecin sonlanmasını wait(2)** veya **waitpid(2)** **sistem çağrısıyla yapar.**
- Bir süreç zombi durumuna geçebilir. Zombi süreçler sistem kaynağı tüketmez.
- **Zombi süreçler,** işletim sistemi tarafından **sonlandırılmış** (tamamen sona ermiş) ancak **ebeveyn süreci** tarafından henüz tam olarak **sonlandırılmamış bir süreçlerdir.**
- Bütün süreçlerin atası olan **init** süreci **belli aralıklarla zombi süreçleri sahiplenerek yok eder.**

❖ Sinyaller

- **Sinyallerin her birinin özel bir anlamı olup süreçle haberleşmeyi sağlar.**
- Örneğin **bir süreç**, **SIGFPE** (Floating Point Error) almışsa **sıfıra bölmek gibi anlamsız bir işlem yaptığını haber verir.**
- **Sinyaller eğer program tarafından yakalanmazsa genellikle programın aniden sonlanmasına neden olur.**
- Bu nedenle **sistem programcıları**, **kendilerine gelen sinyali yakalayıp programı düzgün bir şekilde kapatırlar.**
- **Sinyali işleyen bu kısma, programcılıkta sinyal işleyici (signal handler) denir.**

- **Aşağıdaki tabloda sinyallerin bir kısmı verilmiştir.**

Sinyal	Değer	Aksiyon	Açıklama
SIGHUP	1	Term	Terminali askıya al yada süreci sonlandır
SIGINT	2	Term	Klavyeden gelen kesme
SIGQUIT	3	Core	Klavyeden gelen çıkış kesmesi
SIGILL	4	Core	Tanımsız CPU işlemi
SIGABRT	6	Core	Abort
SIGFPE	8	Core	Matematik işlemci hatası
SIGKILL	9	Term	Süreci öldür
SIGSEGV	11	Core	Geçersiz bellek adreslemesi
SIGPIPE	13	Term	Okuyucusu olmayan kırık pipe'a yazma hatası
SIGALRM	14	Term	Zamanlama sinyali
SIGTERM	15	Term	Sonlandırma sinyali
SIGUSR1	30,10,16	Term	Programcının tanımladığı özel sinyaller
SIGUSR2	31,12,17	Term	Programcının tanımladığı özel sinyaller
SIGCHLD	20,17,18	Ign	Alt süreç durdu veya sonlandı
SIGCONT	19,18,25	Cont	Durmuşsa devam et
SIGSTOP	17,19,23	Stop	Durdur

Term: Süreci sonlandır.
Ign: Sinyali gözardı et
Core: Süreci sonlandır ve bellek dump'ı al.
Stop: Süreci durdur

❖ Kill

❖**Süreci sonlandırmak için kullanılır.**

❖**kill <pid> :** Süreci pid numarası ile sonlandırır.

❖**kill –i <pid> :** Parametre değerine göre işlem üzerinde işlem yapar. Aşağıda **kill –l** komutu ile parametre değerleri öğrenilebilir.

```
seckin@bilmuh:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

- **Örnek olarak SIGHUP, süreci askıya alır**.
- **SIGKILL, süreci sonlandırmaya zorlar.**

ÖRNEK

- **Sleep** komutu ile **bir süreç başlatılmıştır** ve **arka plana alınmıştır.**
- Böylece tek terminalde kill komutu yazılabilmiştir ve süreç sonlandırılmıştır.

```
root@bilmuh:~# sleep 300&
[1] 4174
root@bilmuh:~# kill -9 4174
```

❖ pkill

❖**Sürecin adı** veya **sahibi** gibi **değişik pattern**'ler verilerek **belirtilen sürece sinyal göndermek için pkill kullanılır.**

❖Aşağıdaki komut, **user4** kullanıcısının sahip olduğu **tüm süreçleri sonlandırır.**

```
# pkill -STOP -u user4
```

❖Aşağıdaki komut ise adında **http** geçen bütün süreçleri sonlandırır.

```
# pkill http
```

❖ pgrep

❖**Verilen kritere süreç araması yaptırmak** için kullanılır.

❖Böylece verilen komuttan etkilenecek süreçlerin istenen süreçler olduğundan emin olunur.

❖Aşağıda bir kullanıcı tarafından yürütülen süreç verilmiştir.

```
root@bilmuh:~# pgrep -u seckin
1107
1108
1123
1126
```

```
root@bilmuh:~# pkill -u seckin
```

Örnek

- Aşağıda bir kullanıcı tarafından bir sonsuz döngü oluşturan bir bet,k yazılmıştır.

```
  GNU nano 7.2                    /home/seckin/sonsuz_dongu.sh
#!/bin/bash
while true;do
echo "sonsuz dongu calisiyor"
sleep 1
done
```

- Oluşturulan bu betik aşağıda verildiği gibi çalıştırılmıştır.

```
seckin@bilmuh:~$ ./sonsuz_dongu.sh
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
sonsuz dongu calisiyor
```

- Çalışan bu tarz bir betik sistemde gereksiz işlem ve kaynak kullanımına yol açacaktır.
- Bunu root sonlandırmak isterse PID numarasını bilmelidir.
- Bunun için pgrep komutundan faydalanılır.

```
root@bilmuh:~# pgrep  sonsuz_dongu.sh
4206
```

- Yukarıda verildiği gibi süreç numarası (PID), bulunmuştur.

```
root@bilmuh:~# kill -9 4206
```

❖ killall

❖**Bir kullanıcı yada bir servisle ilgili tüm süreçleri sonlandırmak için bu komuttan faydalanılır.**

❖**Bu komutun birçok parametreleri vardır.**

❖**Aşağıda bir kullanıcıya ait olan süreçlerin sonlandırılması verilmiştir.**

```
root@bilmuh:~# pgrep -u seckin
1107
1108
1123
1126
1132
1134
1135
1142
1156
```

```
root@bilmuh:~# killall -u seckin
```

❖**Aşağıda bir servis ile ilgili süreçlerin sonlandırılması verilmiştir.**

killall apache2

UYGULAMA SORULARI

1. **Kullanici_adi.txt adında bir dosya oluşturunuz.** Bu dosyanın içeriğine 10 tane kişi ekleyiniz. Burada 1. sütunda aynı isimden 3 tane isim olsun. 3 tane de benzer (Kamil, Kadir vs.) isimler olsun. 4 tane de farklı isimler olsun. 2. sütunda bu kullanıcıların e-posta adresleri olsun. Bunlardan 4 tanesinin e-posta adresi, e-posta tanımlama kurallarına (Örn: ax12@btu, 123, bm12.com vb.) uymasın.

a) **Aynı isme sahip satırları bulan komutu yazınız.**

b) **Benzer isimleri veren satırları bulan komutu yazınız.**

c) **com ile biten satırları bulduran komutu yazınız.**

d) **Sadece e-posta sütunun ekranda gösteren ve de bu sonucu; eposta.txt dosyasına kaydediniz**.

e) **eposta.txt dosyasında e-posta tanımlama kurallarına uyan ve uymayan satırları bulacak komutları yazınız.**

f) **Kullanıcı_adi.txt dosyasında aynı olarak tanımladığınız isimleri bulup bunların hepsini büyük veya küçük harfe çevirerek dosyanın içeriğini güncelleyen komutu yazınız.**

2. Sonsuz döngüden oluşan ve ekrana 'Merhaba' yazacak bir betik yazıp bu betiği ikinci bir terminalde çalıştırınız.

Bu terminalde ekrana yazılanları gözlemleyiniz.

Daha sonra bu betiği sonlandıracak komutu sisteme giriş yaptığınız terminalde sonlandırınız.

3. Root olarak giriş yaptığınız bir sunucuda bir kullanıcının tüm süreçlerini sonlandırınız.

4. Root olarak giriş yaptığınız bir sunucuda bir kullanıcının tüm süreçlerini askıya alınız.

5. Kullanıcı olarak giriş yaptığınız bir sistemde başka kullanıcıların yürüttüğü süreçleri görebilir ve sonlandırabilir misiniz?

6. ping 127.0.0.1 komutunu çalıştırıp 10 saniye sonra bu komutun yürütülmesini kısa yol tuşu ile sonlandırınız.

7. ping 127.0.0.1 komutunu çalıştırınız ve arkaplana atınız. Bu süreci sonlandırmaya çalışınız.

KAYNAKLAR

- Pardus LPI-Sertifikasyon-Kitabı
- Linux Komut Satırı, Kemal DEMİREZ, Abaküs Kitap Yayın Dağıtım Hizmetleri, 2019