pl.: Írja bele, hogy hol kezdi és hol van vége.

```
#!/bin/bash
cat /etc/passwd ␣|␣ awk '
BEGIN {
   print ("Feldolgozás megkezdése");
}
{ print ("Sor: ␣ " $0);
}
END {
     print ("Feldolgozás vége");
}
'
```

./readwrite.sh ␣|␣ more ⇒ oldalakra tördeli

pl.: Írja ki a 4. mezőket minden sorból.

```
#!/bin/bash
cat /etc/passwd ␣|␣ awk ␣ -F: '
{ print ($4);
}
'
```

(-F: elválasztó a kettőspont)

pl.: A 4. mezőben álló számokat adjuk össze.

```
#!/bin/bash
cat /etc/passwd ␣|␣ awk ␣ -F:
BEGIN {
    S=0;
}
{   S = S + $4
}
END {
   print ("Az összeg: ␣ " S);
}
'
```

pl.: Átlag megadása:

```
#! /bin/bash
cat /etc/passwd | awk -F:
BEGIN
 { S=0;
 }
 { S = S+ $4
 }
 END {
     print ("Az összeg : " S } ;
     print ("Az átlag : " S/NR );
     }
```

S = S + $4 <=> S+ = $4

Ha a változót eggyel akarjuk növelni : ++J <=> J = J+1
print (NR) → kiírja a sorok számát

Elágazások

```
#!/bin/bash
awk -v T = "Hello"
Begin {
 if (T == "Hello") {
    print ("Hello");
    }
     else {
     print ("Nem Hello");
    }
}
```

= : értékadás
== : vizsgálat, sikerült-e az értékadás

Ciklus

pl.: Írja ki 0-10-ig a számokat.

```
#!/bin/bash
BEGIN {
for (i=0; i<10; i++) {
   printf ("%d\n", i)
   }
}
```

- (i=0; i<10; i++) → ciklus tulajdonság: kezdeti érték; bentmaradás feltétele; ciklusváltozó növelése. Több változót is lehet vesszővel elválasztani
- printf → formázott print
- %d → azon a ponton egy decimális számot ír ki
  - %h: hexadecimális
  - %s: string
- \n → soremelés

Ha szöveget írатok ki:

```
printf ("neved: %s, korod %d", nev, kor);
```

Asszociatív tömb: az index szöveggel történik, nem számmal

pl.:

```
#!/bin/bash
cat /etc/passwd | awk -F : '
{
szo[$7]++
}
END {
 for (i in szo) print (szo[i], i)
}
'
```

- -F : → feldolgozás mezőnként, ahol a mezőelválasztó a „:".
- ++ → növeljük eggyel
- (i in szo) → minden „szo"-beli „i" elem
- szo[i] → érték
- i → melyik szó fordul elő annyiszor

Ha visszafelé akarjuk kiíratni:

utolsó sora: '

```
| sort -n -r | more
```

- -n → szám szerint rendezve
- -r → visszafelé rendezve

pl.: #!/bin/bash
cat /etc/passwd | awk -F: '
{ for (i=NF; i>0; i--) printf ("%s", $i)
printf ("\n")

$-ral csak a beolvasott dolgokat jelöljük, a sima változókat sima betűkkel.

Az AWK-ra példák

Asszociatív tömbök: bár függvénye van, a rendezéstől olyanig, amilyet máshol nehézkesen használunk.

awk (NR): a feldolgozott sorok száma

linuxdoc.freeweb.hu /-ről lehet letölteni

pl.: keressük meg a leghosszabb sort és írassuk ki!

length: string hosszának lekérdezése

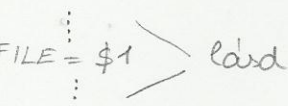
#! /bin/bash
↓ ↳ parancsértelmező (bin: neve; bash: helye)
futtatható programot jelképez

FILE=/etc/passwd
cat $FILE | awk -F: '
BEGIN {
mx=0;
ln=" ";
}
{
if (length($0)>mx){
mx=length($0);
ln=$0;
}
}

$0: a sor
$1: első mező
$2: második mező

```
END {
    printf ("A max hossz: %d\n", mx);
    printf ("A sor: %s\n", ln);
    }
```

pl.: Alakítsuk át, hogy meg tudjuk adni a paramétert, hogy melyik az a file, amiben keresünk.



pl.: A minimumot írja ki!

```
    :
  BEGIN {
mx = 1000;
    :
if (length ($0) < mx);
    :
printf ("Minimum: %d\n", mx);
    :
```

pl.: A shell csak a 40 karakternél hosszabbakat írja ki. (sor!)

```
#!/bin/bash
FILE = $1
if [ ! -f "$FILE" ]; then
    echo "A megadott file nem található".
      exit 1
fi
cat $FILE | awk '
{
  if (length ($0) > 40)
        print ($0);
 }
'
```

pl.: Be kell kérni egy nevet és a hosszt.

```
FILE = $1
HOSSZ = $2
cat FILE | awk -v hossz = $HOSSZ
    if (length ($0) > hossz)
```

pl.: cat szotar.txt | wc

kiírja ? sor ? szó ? karakter

cat szotar.txt | cut -d: -f2 | awk -F ' '

cut: kivágja a 2. oszlopot

```
{ for (i=1; i<NF; i++) {
      print ($1)
  }
}
```

[tr ' mit ' ' mire ' → karakterenként cserél]

tr -d ' karakter ' → kidobálja a névegből

uniq: ami többször fordul elő, abból csak egyet tart meg

Kérdezni, hogy van-e adott domain-nek: host -t .soa $i.hu

Ha a visszatérési érték 0: SIKERES VOLT!

pl.: Írjunk olyan programot, ami kiírja, hogy a 3. mező 0-e.

```
#!/bin/bash
FILE = /etc/passwd
cat $FILE | awk -F:
{ $3 = 0
if ($3 %10 == 0) print $0
}
)
```

10-zel való osztás maradéka 0

pl.: Kiírja az időt.

```
#! /usr/bin/gawk -f
BEGIN {
print strftime ("%Y.%m.%d %H:%M:%S", systime( ));
}
{
}
```

gawk: awk bővítése → bárki használhatja.

év hónap nap óra perc másodperc

Partíciók:

Sovány kliens, erős szerver, de lehet gyenge hardvereket használni
(pl.: NEPTUN: Windows NT terminal server)

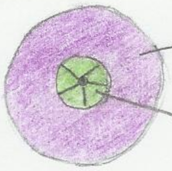
NETWORK COMPUTER: hálózatfüggő, mert nincs benne merevlemez, csak 1 romban lévő frissítő operációs rendszer.

X WINDOW: pl.: linux-nál
ennél vannak X terminálok ⇒ csak sovány kliensek, a grafikus kép jön-megy

Manapság már 120-160 GByte-os merevlemezek kaphatók.
⇓
csak partíciókat lehet létrehozni

PARTICIONÁLÁS:

DOS-ban: FDISK paranccsal karakteres felületen lehetett létrehozni



Gyakran egy partíciót hoznak létre, ez a C:\. Sokkal jobb több partíciót létrehozni, mert a C:\-t formatálhatjuk, amiközül, hogy a minőséges adatok elvesznének, hiszen a D:\-n megőrződnek.

Filerendszerek:
DOS: FAT
WINDOWS: FAT32
WINDOWS NT: NT FS

Formázás:

Start / beáll. / vezérlőpult / felügyeleti eszközök / számgép kezelés
⋮
rendszereszközök
⋮
eseménynaplók

lemezterület

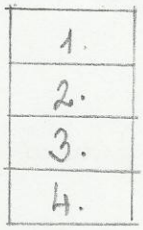
(jobbra alul) lemez 0 } adathordozó ⇒ nem lehet több partícióra
CD ROM } osztani (mágneslemezeket)

Több partíción is lehet operendszer, amelyről betölthetők.
Adott a 0. sáv 0. szektor (legelső sáv és legelső szektor) →
→ itt van a betöltőprogram.
Amelyik partíció aktív, onnan tölti be az operendszert

BOOT MANAGER program: megkérdezi, hogy mit akarunk betölteni,
azt aktívvá teszi és betölti.

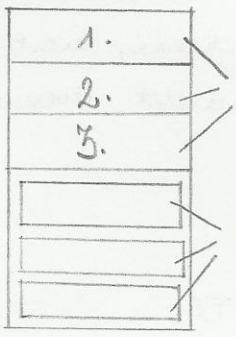
Típusok: elsődleges partíció: többet tud, mint a másodlagos.
Róluk lehet bootolni, és aktívvá tehetők.
másodlagos partíció:

Elsődleges partíciók: → 4 partíció hozható létre



4 partíció hozható létre

VAGY:



elsődleges partíciók

másodlagos partíciók =
= kiterjesztett partíciók

Tetszőleges számú másodlagos partíció rehető fel, ezek nem
bootolhatók (bootolhatóság = aktivitás)

Mit a merevlemezen több partíciót tárolnak, azoknak le kell írni a partíciók határait, hogyha elveszik a partíciótábla, vissza lehessen írni.

Ha megvan a partícionálás, a diszket fel kell formázni.
Linuxon a partícionálást csak a rendszergazda végezheti el.

parancs: FDISK /dev/hda (hd: harddisk, a: első meghajtó)

b: 2. meghajtó
c: 3. –"–
d: 4. –"–

p (print): kiírja, mi a merevlemez állapota
hda 1: első partíció
Egy partíciónak van START (eleje) és END (vége) száma. Ha elveszik a partíciótábla, ezeket kell visszaírni és újra működni fog.

BLOCKS: kiírja, hány blokkból áll
Az aktív partíciót „*" jelzi.
ha a partíció EXTENDED → bővített

Partíció típusok:

FAT 12: floppy
WIN95: FAT32

SOLARIS-nál rögtön másodlagos partíciókat kell csinálni, és ebbe kell mindent beleratni, ez az egészet átfogja.

RAID: disktömb (több lemezt fognak eggyé.)

RAID 0: két disk van

Ha a 0-ba beletesszük egy csíkozott kötetet (amit együtt fognak látni), az adatot kétfelé írja, felét ide, felét oda.

Előny: gyorsabban lehet rá írni, mert minden felesődik.

a winchester teljesítménye: 2,5 MB/s.

Stripe set: csíkozott kötet

Hátrány: Ha az egyik disk elszáll, a másikat nem lehet használni. 2x olyan gyorsan elérhető, de az ADATBIZTONSÁG NEM nagy.

RAID 1:

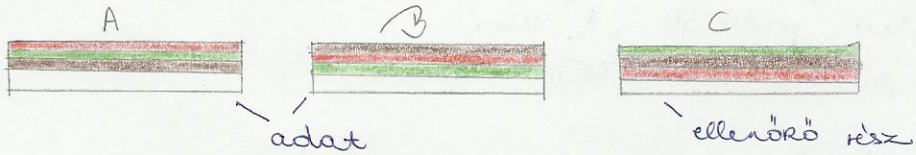
TÜKRÖZÉS: az egyik disk tükörképe a másiknak.

Ha különbség van a két disk között → READ ONLY: újra tükrözi

Terület: a tárterület fele

Sebesség: az eredeti merevlemez sebességével ég. megegyezik.

RAID 5:



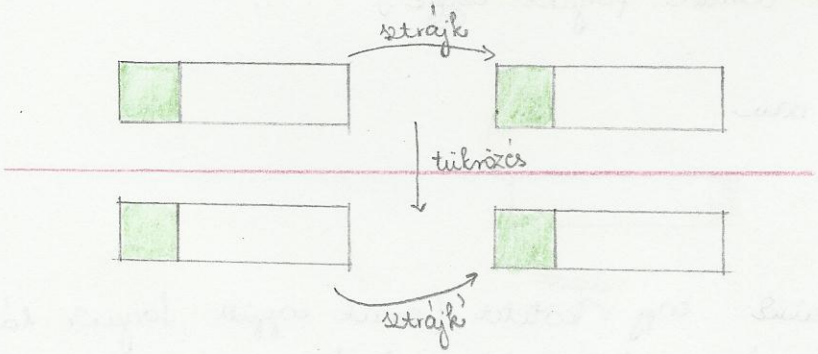
minimum 3 disk kell.

Ha 1 elveszne, az egyik adatból és az ellenőrzőből kiszámítható.

Terület: a tárterület 2/3-a

Sebesség: nő (az adatokat többnyire disk nyeli el)

4. diskkel lehet stripelni (felibe írni) és tükrözni.



Amikor formázunk, úgy érezzük, hogy csak egyet formázunk, de valójában mindet formázzuk