

```

basicsys@mars~/lec9>cat F1
yyy moshe1
moshe2
no mosh
abc

basicsys@mars~/lec9>awk '/moshe/{print;print}' F1
yyy moshe1
yyy moshe1
moshe2
moshe2

basicsys@mars~/lec9>awk '{print;print}' F1
yyy moshe1
yyy moshe1
moshe2
moshe2
no mosh
no mosh
abc
abc

basicsys@mars~/lec9>cat F2
aa  bb  cc
dd  ee  fff gg

basicsys@mars~/lec9>awk '{print $3 $1}' F2
ccaa
ffffd

basicsys@mars~/lec9>awk '{print $3,$1}' F2
cc aa
fff dd

basicsys@mars~/lec9>awk '{print $3" "$1}' F2
cc aa
fff dd

basicsys@mars~/lec9>awk '{print $0,NF}' F2
aa  bb  cc 3
dd  ee  fff gg 4

```

```

basicsys@mars~/lec9>awk '{print $0,$NF}' F2
aa  bb  cc cc
dd  ee  fff gg gg

basicsys@mars~/lec9>awk '{print $NF}' F2
cc
gg

basicsys@mars~/lec9>awk '{print NR,$0}' F1 F2
1 yyy moshe1
2 moshe2
3 no mosh
4 abc
5 aa  bb  cc
6 dd  ee  fff gg

basicsys@mars~/lec9>awk '{print FNR,$0}' F1 F2
1 yyy moshe1
2 moshe2
3 no mosh
4 abc
1 aa  bb  cc
2 dd  ee  fff gg

basicsys@mars~/lec9>cat F4
aa 8 2
bb 7 6
cc 2 4

basicsys@mars~/lec9>awk '$3>2 {print $0,$2*$3}'
F4
bb 7 6 42
cc 2 4 8

basicsys@mars~/lec9>awk '$3>2 {print $0,x*1}' F4
bb 7 6 0
cc 2 4 0

basicsys@mars~/lec9>awk '$3>2 {print $0,x+1}' F4
bb 7 6 1
cc 2 4 1

```

```

basicsys@mars~/lec9>awk '$3>2 \
> {x=5;print $0,x+1}' F4
bb 7 6 6
cc 2 4 6

basicsys@mars~/lec9>awk '$3>2 {print $3/3}' F4
2
1.33333

basicsys@mars~/lec9>echo ${4/3}
1

basicsys@mars~/lec9>cat F3
1999 12 ab Fiat
200000 130 cc Subaru

basicsys@mars~/lec9>awk '{printf "year %8d \
> no %4d car %-8s good\n", $1, $2, $4}' F3
year 1999 no 12 car Fiat good
year 200000 no 130 car Subaru good

The following example shows how to align to the
right depending
on a value of a variable (variable x in this
example).

x=-10

basicsys@mars~/lec9>awk '{printf "year %'$x'd \
> no\n", $1}' F3
year 1999 no
year 200000 no

basicsys@mars~/lec9>cat F1
ab cd ef gh
ab cd
d e f g h

basicsys@mars~/lec9>awk 'NF>=3 {print "3", $0} \
> NF==5 {print "5", $0}' F1
3 ab cd ef gh
3 d e f g h
5 d e f g h

```

```

basicsys@mars~/lec9>echo "ab cd" ef | \
> awk '{print $2}'
cd

basicsys@mars~/lec9>echo ab cd ef | \
> awk '{print $2}'
cd

basicsys@mars~/lec9>x=5.5

basicsys@mars~/lec9>y=3.2

basicsys@mars~/lec9>echo ${x+y}
-bash: 5.5+3.2: syntax error: invalid arithmet

basicsys@mars~/lec9>echo $x $y | \
> awk '{print $1+$2}'
8.7

basicsys@mars~/lec9>cat B1
{print "hi"}
{print $1+$2}
{print $1*$2}

basicsys@mars~/lec9>cat F1
10 20 30
40 45
3.2 5.5 6.3

basicsys@mars~/lec9>awk -f B1 F1
hi
30
200
hi
85
1800
hi
8.7
17.6

basicsys@mars~/lec9>cat B1
#!/bin/awk -f
{print "hi"}
{print $1+$2}

```

```

{print $1*$2}

basicsys@mars~/lec9>chmod u+x B1

basicsys@mars~/lec9>B1 F1
hi
30
200
hi
85
1800
hi
8.7
17.6

basicsys@mars~/lec9>cat B1
{print hi}
{print $1+$2}
{print $1*$2}

basicsys@mars~/lec9>awk -f B1 F1

30
200

85
1800

8.7
17.6

basicsys@mars~/lec9>cat B1
BEGIN {print "hi"}
{print $1+$2}
{print $1*$2}
BEGIN {print "hello"}
END {print "NR="NR}
BEGIN {print "ok"}
END {print "END"}

basicsys@mars~/lec9>cat F1
10 20 30
40 45
3.2 5.5 6.3

```

```

basicsys@mars~/lec9>awk -f B1 F1
hi
hello
ok
30
200
85
1800
8.7
17.6
NR=3
END

basicsys@mars~/lec9>cat P1
BEGIN {print ARGV}

basicsys@mars~/lec9>awk -f P1 F1 F2 F3
4

basicsys@mars~/lec9>cat P1
BEGIN {OFS="xy"; print "ab","cd"}

basicsys@mars~/lec9>awk -f P1
abxycd

basicsys@mars~/lec9>cat P1
BEGIN {OFS="xy"; print "ab","cd"
      print "de","fg"}

basicsys@mars~/lec9>awk -f P1
abxycd
dexyfg

basicsys@mars~/lec9>cat P2
BEGIN {OFS="xy"; ORS="zz"
      print "ab","cd"
      print "de","fg"}

basicsys@mars~/lec9>awk -f P1
abxycdzzdexyfgzzbasicsys@mars~/lec9>

```

```

basicsys@mars~/lec9>cat F1
a b c
de
fg

basicsys@mars~/lec9>cat P1
{ nc = nc + length($0) + 1
nw = nw + NF
}
END {print NR, "lines", nw, "words", nc, "chars"}

basicsys@mars~/lec9>awk -f P1 F1
3 lines 5 words 12 chars

basicsys@mars~/lec9/examples>cat F1
abc def
ghe
a b c d

basicsys@mars~/lec9/examples>cat F2
100
20000
25

basicsys@mars~/lec9/examples>awk '{print \
> FILENAME,$0}' F1 F2
F1 abc def
F1 ghe
F1 a b c d
F2 100
F2 20000
F2 25

basicsys@mars~/lec9/examples>cat F3
abc

basicsys@mars~/lec9/examples>cat A1

BEGIN { for (i=0; i<length(ARGV); i++) {
        print "ARGV["i"]="ARGV[i]
        }
}

```

```

basicsys@mars~/lec9/examples>awk -f A1 F1 F2 F3
ARGV[0]=awk
ARGV[1]=F1
ARGV[2]=F2
ARGV[3]=F3

basicsys@mars~/lec9/examples>awk -f A1 F1 F2 R3
ARGV[0]=awk
ARGV[1]=F1
ARGV[2]=F2
ARGV[3]=R3

basicsys@mars~/lec9/examples>cat A1

BEGIN { for (i=0; i<length(ARGV); i++) {
        print "ARGV["i"]="ARGV[i]
        }
}
{print FILENAME,$1}

basicsys@mars~/lec9/examples>awk -f A1 F1 F2 R3
ARGV[0]=awk
ARGV[1]=F1
ARGV[2]=F2
ARGV[3]=R3
F1 abc
F1 ghe
F1 a
F2 100
F2 20000
F2 25
awk: A1:6: (FILENAME=F2 FNR=3) fatal: cannot open
file `R3' for reading
file or directory)

```

```

basicsys@mars~/lec9/examples>awk -f A1 F1 F2 F3
ARGV[0]=awk
ARGV[1]=F1
ARGV[2]=F2
ARGV[3]=F3
F1 abc
F1 ghe
F1 a
F2 100
F2 20000
F2 25
F3 abc

basicsys@mars~/lec9>cat P1
{ nc = nc + length($0) + 1
nw = nw + NF
}
END {print NR, "lines", nw, "words", nc, "chars"}

basicsys@mars~/lec9>cat F1
abc
de
a x

basicsys@mars~/lec9>awk -f P1 F1
3 lines 4 words 11 chars

basicsys@mars~/lec9>cat F1
abc
de
a x

basicsys@mars~/lec9>cat F2
abdd
e f

basicsys@mars~/lec9>awk -f P1 F1 F2
5 lines 7 words 20 chars

```

```

basicsys@mars~/lec9>cat P2
BEGIN {fname=ARGV[1]}
(FILENAME == fname) {
    nc=nc+length($0)+1
}
(FILENAME != fname) {
    print fname, nc, "chars"
    fname=FILENAME
    nc=length($0)+1
}
END {
    print fname, nc, "chars"
}

basicsys@mars~/lec9>cat F1
abc
de
a x

basicsys@mars~/lec9>cat F2
abdd
e f

basicsys@mars~/lec9>awk -f P2 F1 F2
F1 11 chars
F2 9 chars

basicsys@mars~/lec9>cat A1
BEGIN {A[12]="abc";A["abc"]=100;A["de"]="w100z";
    for (y in A) {
        print y,"A["y"]="A[y]"
    }
}

basicsys@mars~/lec9>awk -f A1
de A[de]=w100z
abc A[abc]=100
12 A[12]=abc

```

```

basicsys@mars~/lec9>cat A2
BEGIN {A[12]="abc";A["abc"]=100;A["de"]="w100z";
  print "length of A is:", length(A)
  for (y in A) {i++}
  print "i=", i
}
basicsys@mars~/lec9>awk -f A2
length of A is: 3
i= 3

```

```

basicsys@mars~/lec9>cat A3
# examples of usage of substr
BEGIN { s="abcdef";
  print "substr(s,2,3)="substr(s,2,3)
  print "substr(s,2)="substr(s,2)
  print
  "substr(\"abcdef\",2,3)="substr("abcdef",2,3)
}

```

```

basicsys@mars~/lec9>awk -f A3
substr(s,2,3)=bcd
substr(s,2)=bcdef
substr("abcdef",2,3)=bcd

```

```

basicsys@mars~/lec9>cat A4
{
  if (NF > max_nf) {
    max_nf = NF
  }
  for (i=1 ; i <= NF ; i++ ) {
    if (length($i) > A[i] ) {
      A[i] = length($i)
    }
  }
}
END { i=1
  while (i <= max_nf) {
    s = s " " A[i]
    i++
  }
  s = substr(s,2)
  print s
}

```

```

basicsys@mars~/lec9>cat F3
abcddd eee ffffff abc
g hhhh abc z
a b c d e f g h

```

```

basicsys@mars~/lec9>awk -f A4 F3
6 5 6 3 1 1 1 1

```

הפעלת awk

ישנן שלוש אפשרויות להפעלת awk

1. הפעלה משורת הפקודה באופן הבא:

רשימת קבצים {...} {סדרת פקודות}2 תנאי {סדרת פקודות}1 תנאי '1 awk

2. הפעלה על ידי קריאה לתוכנית awk שנמצאת בקובץ נפרד (לדוגמה קובץ בשם B) באופן הבא:

awk -f B רשימת קבצים B

התוכנית awk שנמצאת בקובץ B הינה במבנה הבא:

```

1תנאי {
  סדרת פקודות לביצוע1
}
2תנאי {
  סדרת פקודות לביצוע2
}

```

...

3. הפעלת תוכנית סקריפט ב- awk שנמצאת בקובץ נפרד (לדוגמה קובץ בשם C) באופן הבא:

רשימת קבצים C

וכן הלאה...

בשלב הסופי (אחרי שכל שורות הקלט נקראו):

אם תנאי 1 הוא END מתבצעת סדרת פקודות לביצוע 1,

אם תנאי 2 הוא END מתבצעת סדרת פקודות לביצוע 2, וכן הלאה...

משתנים ב-awk

המשתנים ב-awk מחולקים לשני סוגים: משתני מערכת (בדרך כלל

באותיות גדולות) ומשתנים שהמשתמש מגדיר.

הטיפול במשתנים הוא בדומה לשפת C (ולא כמו ב-bash) כך שלקבלת

ערך של משתנה לא מוספים \$ לפני המשתנה. משתנה שלא קיבל ערך

התחלתי מקבל ערך 0 כאשר משתמשים בו כבעל ערך מספרי, (לדוגמה

בביטוי $x+5$) ומקבל ערך מחרוזת ריקה כאשר משתמשים בו כמחרוזת.

משתני מערכת

\$0 מכיל את שורת הקלט הנוכחית

\$1 מכיל את המילה הראשונה בשורת הקלט הנוכחית

\$2 מכיל את המילה השנייה בשורת הקלט הנוכחית

וכן הלאה...

עבור משתנה i $\$i$ מכיל את המילה ה- i בשורת הקלט הנוכחית.

NF מכיל את מספר המילים בשורת הקלט הנוכחית

NR מכיל את מספר שורת הקלט הנוכחית (מספור שורות הקלט מתחיל מ-1)

FNR מכיל את מספר שורת הקלט הנוכחית בתוך הקובץ הנוכחי. (מספור

השורות מתחיל מ-1).

FILENAME מכיל את שם קובץ הקלט הנוכחי

ARGV מערך שמכיל את רשימת הפרמטרים לתוכנית (בדרך כלל זו רשימת

הקבצים שמועברים בקריאה לתוכנית ה-awk), כאשר

תוכנית ה-awk שנמצאת בקובץ C הינה במבנה כפי שתוארה התוכנית

שנמצאת בקובץ B כאשר השורה הראשונה חיבת להיות:

```
#!/bin/awk -f
```

השורה הנ"ל מסמנת למערכת שהקובץ מכיל תוכנית בשפת awk.

הלוגיקה של תוכנית awk

תוכנית awk בנויה מסדרת זוגות של תנאים ופקודות לביצוע כמו במבנה הבא:

```
{
```

```
    סדרת פקודות לביצוע 1
```

```
}
```

```
{
```

```
    סדרת פקודות לביצוע 2
```

```
}
```

```
...
```

הקלט לתוכנית מגיע מרשימת הקבצים שהועברו בזמן הפעלת awk (בכל

אחת משלושת הצורות להפעלת awk ישנה רשימת קבצים שמועברת). במידה

ורשימת הקבצים לא קיימת אז הקלט לתוכנית ה-awk מגיע מהקלט

הסטנדרטי (ברירת מחדל: מקלדת).

תוכנית ה-awk מתבצעת באופן הבא:

בשלב ההתחלתי (לפני שקוראים שורות קלט):

אם תנאי 1 הוא BEGIN מתבצעת סדרת פקודות לביצוע 1,

אם תנאי 2 הוא BEGIN מתבצעת סדרת פקודות לביצוע 2,

וכן הלאה...

בשלב האמצעי מתבצע מעבר על כל שורות הקלט ועבור כל שורת קלט:

אם תנאי 1 מתקיים מתבצעת סדרת פקודות לביצוע 1,

אם תנאי 2 מתקיים מתבצעת סדרת פקודות לביצוע 2,

awk מכיל את המחרוזת ARGV[0]

ARGV[1] מכיל את הפרמטר הראשון לתוכנית

ARGV[2] מכיל את הפרמטר השני לתוכנית, וכן הלאה...

ARGC מכיל את מספר הפרמטרים לתוכנית ועוד 1

תנאים ב- awk

ביטוי רגולארי/ התנאי מתקיים אם שורת הקלט הנוכחית מתאימה לביטוי (לפי חוקים של ביטויים רגולאריים).

כל התנאים הבאים אפשריים (המשמעות של כל תנאי ברורה):

מספר1 > מספר2 מספר1 >= מספר2

מספר1 < מספר2 מספר1 <= מספר2

מספר1 == מספר2 מספר1 != מספר2

מחרוזת1 == מחרוזת2 מחרוזת1 != מחרוזת2

תנאי1 && תנאי2 || תנאי1

ניתן להשתמש בסוגריים (אין צורך בהקדמת \ לסוגריים) בהרכבת תנאים

כמו למשל: (תנאי1 || תנאי2) && (תנאי3 || תנאי4)

ניתן להשתמש ב- ! לשלילת תנאי כמו למשל (תנאי1) !

BEGIN התנאי הזה מתקיים רק בשלב ההתחלתי שלפני קריאת שורות הקלט.

END התנאי הזה מתקיים רק בשלב הסופי שלאחר קריאת שורות הקלט.

הפקודה print ב- awk

מחרוזת1, מחרוזת2, print מדפיסה את המחרוזות כאשר בין המחרוזות מודפס הערך של משתנה המערכת OFS (ברירת מחדל: תו רווח אחד), בנוסף מודפס בסוף הפלט את הערך של משתנה המערכת ORS (ברירת מחדל: \n).

כאשר הפרמטרים לפקודה print לא מופרדים על ידי פסיקים כמו באופן הבא:

print 1 מחרוזת2 מחרוזת1

המחרוזות יודפסו ברצף (ללא תו הפרדה בין המחרוזות) ובסוף יודפס ערך משתנה המערכת ORS.

הפקודה printf ב- awk

...פרמטר1, פרמטר2, printf המרה

מדפיסה את המחרוזת לאחר החלפת הביטוי ההמרה בפרמטרים המתאימים להם לפי הסדר משמאל לימין. לדוגמה בפקודה

printf "ab%5dcd%-6d",123,456

ביטוי ההמרה %5d יוחלף על ידי הדפסת 123 בתוספת שני רווחים מצד שמאל,

וביטוי ההמרה %-6d יוחלף על ידי הדפסת 456 בתוספת 3 רווחים מצד ימין.