

```
basicsys@mars~/lec10>cat F3
abcdddabc eee fffffff abc
g hhhhh abc z
```

```
basicsys@mars~/lec10>cat P1
{
  print $0
  sub("abc",111)
  print $0
}
```

```
basicsys@mars~/lec10>awk -f P1 F3
abcdddabc eee fffffff abc
111dddabc eee fffffff abc
g hhhhh abc z
g hhhhh 111 z
```

```
basicsys@mars~/lec10>cat P1
{
  print $0
  gsub("abc",111)
  print $0
}
```

```
basicsys@mars~/lec10>awk -f P1 F3
abcdddabc eee fffffff abc
111ddd111 eee fffffff 111
g hhhhh abc z
g hhhhh 111 z
```

```
basicsys@mars~/lec10>cat P1
{
  print $0
  gsub("abc",111,$1)
  print $0
}
```

```
basicsys@mars~/lec10>awk -f P1 F3
abcdddabc eee fffffff abc
111ddd111 eee fffffff abc
g hhhhh abc z
g hhhhh abc z
```

```
basicsys@mars~/lec10>cat P1
BEGIN {
    s="abcdef"
    gsub("[adf]",111,s)
    print s
}
```

```
basicsys@mars~/lec10>awk -f P1
111bc111e111
```

```
basicsys@mars~/lec10>cat P1
BEGIN{FS="(ab) +";OFS="@"}
{print $1,$2,$3,$4,NF}
```

```
basicsys@mars~/lec10>echo "cdababxyabab  zabwr" \
> | awk -f P1
cd@xy@  z@wr@4
```

```
basicsys@mars~/lec10>cat P1
BEGIN{FS=" ";OFS="@"}
{print $1,$2,$3,$4,NF}
```

```
basicsys@mars~/lec10>echo abcd | awk -f P1
a@b@c@d@4
```

```
basicsys@mars~/lec10>cat P1
BEGIN{FS=""}
{
    for (i=1 ; i <= NF ; i++ ) {
        print $i
    }
}
```

```
basicsys@mars~/lec10>echo abcd | awk -f P1
a
b
c
d
```

```
basicsys@mars~/lec10>cat P1
BEGIN {
  n=split("1aaabc2abc3",A,"a")
  for (i in A) {
    print "A[" i "]=" A[i]
  }
  print "n=" n
  for (i=1; i<=length(A); i++){
    print "A[" i "]=" A[i]
  }
}
```

```
basicsys@mars~/lec10>awk -f P1
A[4]=bc2
A[5]=bc3
A[1]=1
A[2]=
A[3]=
n=5
A[1]=1
A[2]=
A[3]=
A[4]=bc2
A[5]=bc3
```

```
basicsys@mars~/lec10>cat P1
BEGIN {
  x["ab"]=8;split("abcd",x,"")
  for (i in x) {
    print "x["i"]=" x[i]
  }
}
```

```
basicsys@mars~/lec10>awk -f P1
x[4]=d
x[1]=a
x[2]=b
x[3]=c
```

```

basicsys@mars~/lec10>cat P1
BEGIN {
  for (i=1; i<=100; i++ ){
    A[i]=i*2
  }
  print "before length(A)=" length(A)
  split("",A)
  print "after length(A)=" length(A)
}

```

```

basicsys@mars~/lec10>awk -f P1
before length(A)=100
after length(A)=0

```

```

basicsys@mars~/lec10>cat P1
BEGIN {
  split("123  abc  ge",A)
  for (i=1; i<=length(A); i++ ){
    print "A[" i "]= " A[i]
  }
}

```

```

basicsys@mars~/lec10>awk -f P1
A[1]=123
A[2]=abc
A[3]=ge

```

```

basicsys@mars~/lec10>cat P1
BEGIN {
  FS="a"
  split("123  abc  ge",A)
  for (i=1; i<=length(A); i++ ){
    print "A[" i "]= " A[i]
  }
}

```

```

basicsys@mars~/lec10>awk -f P1
A[1]=123
A[2]=bc  ge

```

```

basicsys@mars~/lec10>cat P1
BEGIN {ORS=""}
{ for (j=1;j <=NF; j++) {
  A[NR,j]=$j
}
}
END { for (i=1; i<=NR; i++) {
  for (j=1; j<= NF; j++) {
    print A[i,j]
  }
  print "\n";
}
}

```

```

basicsys@mars~/lec10>cat F3
abcddd eee ffffff abc
g hhhh abc z
a b c d e f g h

```

```

basicsys@mars~/lec10>awk -f P1 F3
abcddeeefffffabcd
ghhhhhabcz
abcdefgh

```

```

basicsys@mars~/lec10>cat P1
BEGIN {x=1;
  print "before f1: x=" x
  f1(x)
  print "after f1: x=" x
}
function f1(x) {
  print "in f1: x=" x
  x=8
}

```

```

basicsys@mars~/lec10>awk -f P1
before f1: x=1
in f1: x=1
after f1: x=1

```

```

basicsys@mars~/lec10>cat P1
BEGIN {x=1;
    print "before f1: a=" a " x=" x
    f1(x)
    print "after f1: a=" a " x=" x
}
function f1(a) {
    print "in f1: a=" a " x=" x
    a=8
}

```

```

basicsys@mars~/lec10>awk -f P1
before f1: a= x=1
in f1: a=1 x=1
after f1: a= x=1

```

```

basicsys@mars~/lec10>cat P1
BEGIN {x=1;
    print "before f1: x=" x " z=" z
    f1(x);
    print "after f1: x=" x " z=" z
}
function f1(a) {
    print "in f1: a=" a " x=" x
    a=8
    x=9
    z=10
}

```

```

basicsys@mars~/lec10>awk -f P1
before f1: x=1 z=
in f1: a=1 x=1
after f1: x=9 z=10

```

```
basicsys@mars~/lec10>cat P1
```

```
BEGIN {x[1]=8; x["ab"]=10
  print "before f1: x[1]=" x[1] " x[ab]=" x["ab"]
  f1(x)
  print "after f1: x[1]=" x[1] " x[ab]=" x["ab"]
}
function f1(a) {
  print "inside f1 a[1]=" a[1] " a[ab]=" a["ab"]
  print "inside f1 x[1]=" x[1] " x[ab]=" x["ab"]
  a[1]=12;
  a["ab"]=13;
  x[1]=14
}
```

```
basicsys@mars~/lec10>awk -f P1
```

```
before f1: x[1]=8 x[ab]=10
inside f1 a[1]=8 a[ab]=10
inside f1 x[1]=8 x[ab]=10
after f1: x[1]=14 x[ab]=13
```

```
basicsys@mars~/lec10>cat P2
```

```
{
  for (i=1; i<= NF; i++) {
    A[NR,i]=$i
  }
}
END {
  for (i=1; i <=NR/2; i++) {
    s=A[i,i];
    for (j=i; j <= NR-(i-1); j++) {
      if (A[i,j] != s) { print "NO" ; exit}
      if (A[j,i] != s) { print "NO" ; exit}
      if (A[NR-(i-1),j] != s)
        { print "NO" ; exit}
      if (A[j,NR-(i-1)] != s)
        { print "NO" ; exit}
    }
  }
  print "YES"
}
```

```
basicsys@mars~/lec10>cat F1
```

```
2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 2
```

```
basicsys@mars~/lec10>awk -f P2 F1
```

```
YES
```

```
basicsys@mars~/lec10>cat P3
```

```
{
  for (i=1; i<= NF; i++) {
    A[NR,i]=$i
  }
}
END {
  for (i=1; i <=NR/2; i++) {
    if (check_row(A,i,i,NR-(i-1))=="NO")
      { print "NO" ; exit}
    if (check_row(A,NR-(i-1),i,NR-(i-1))=="NO")
      { print "NO" ; exit}
    if (check_column(A,i,i,NR-(i-1))=="NO")
      { print "NO" ; exit}
    if (check_column(A,NR-(i-1),i,NR-(i-1))=="NO")
      { print "NO" ; exit}
  }
  print "YES"
}
```

```
function check_row(A,i,j1,j2) {
  s=A[i,j1]
  for (p=j1; p<= j2; p++) {
    if (A[i,p] != s) { return "NO" }
  }
  return "YES"
}
```

```
function check_column(A,i,j1,j2) {
  s=A[j1,i]
  for (p=j1; p<= j2; p++) {
    if (A[p,i] != s) { return "NO" }
  }
  return "YES" }

```

```
basicsys@mars~/lec10>cat F1
```

```
2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 2
```

```
basicsys@mars~/lec10>awk -f P3 F1
```

```
YES
```

```
basicsys@mars~/lec10>cat P4
```

```
{
for (i=1; i<= NF; i++) {
  A[FILENAME,FNR,i]=$i
}
}
END {
  for (x=1; x < length(ARGV); x++) {
    if (check_file(A,ARGV[x],FNR)=="NO")
      { print ARGV[x] ": NO" }
    else
      {print ARGV[x] ": YES"}
  }
}

```

```

function check_file(A, file, n) {
  for (i=1; i <=n/2; i++) {
    if (check_row(A, file, i, i, n-(i-1))=="NO")
      { return "NO" }
    if (check_row(A, file, n-(i-1), i, n-(i-1))=="NO")
      { return "NO" }
    if (check_column(A, file, i, i, n-(i-1))=="NO")
      { return "NO" ; exit }
  }
  if (check_column(A, file, n-(i-1), i, n-(i-1))=="NO")
    { return "NO" }
  }
  return "YES"
}

```

```

function check_row(A, file, i, j1, j2) {
  s=A[file, i, j1]
  for (p=j1; p<= j2; p++) {
    if (A[file, i, p] != s) { return "NO" }
  }
  return "YES"
}

```

```

function check_column(A, file, i, j1, j2) {
  s=A[file, j1, i]
  for (p=j1; p<= j2; p++) {
    if (A[file, p, i] != s) { return "NO" }
  }
  return "YES"
}

```

```

basicsys@mars~/lec10>cat F1

```

```

2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 2

```

```
basicsys@mars~/lec10>cat F2
2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 5
```

```
basicsys@mars~/lec10>awk -f P4 F1 F2
F1: YES
F2: NO
```

```
basicsys@mars~/lec10>cat P5
{
for (i=1; i<= NF; i++) {
    A[FILENAME,FNR,i]=$i
    B[FILENAME]=FNR
}
}
END {
    for (x=1; x < length(ARGV); x++) {
        if (check_file(A,ARGV[x],B[ARGV[x]])=="NO")
            { print ARGV[x] ": NO" }
        else {print ARGV[x] ": YES"}
    }
}
```

```
function check_file(A,file,n) {
    for (i=1; i <=n/2; i++) {
        if (check_row(A,file,i,i,n-(i-1))=="NO")
            { return "NO"}
        if (check_row(A,file,n-(i-1),i,n-(i-1))=="NO")
            { return "NO"}
        if (check_column(A,file,i,i,n-(i-1))=="NO")
            { return "NO" ; exit}
    }
    if (check_column(A,file,n-(i-1),i,n-(i-1))=="NO")
        { return "NO" }
    }
    return "YES"
}
```

```

function check_row(A, file, i, j1, j2) {
  s=A[file,i,j1]
  for (p=j1; p<= j2; p++) {
    if (A[file,i,p] != s) { return "NO" }
  }
  return "YES"
}

```

```

function check_column(A, file, i, j1, j2) {
  s=A[file,j1,i]
  for (p=j1; p<= j2; p++) {
    if (A[file,p,i] != s) { return "NO" }
  }
  return "YES"
}

```

```

basicsys@mars~/lec10>cat F1

```

```

2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 2

```

```

basicsys@mars~/lec10>cat F2

```

```

2 2 2 2 2 2
2 1 1 1 1 2
2 1 3 3 1 2
2 1 3 3 1 2
2 1 1 1 1 2
2 2 2 2 2 5

```

```

basicsys@mars~/lec10>cat F3

```

```

2 2 2 2
2 1 1 2
2 1 1 2
2 2 2 2

```

```

basicsys@mars~/lec10>awk -f P5 F1 F2 F3

```

```

F1: YES
F2: NO
F3: YES

```

```
basicsys@mars~/lec10>cat P6
{
  split($0,A,":")
  split(A[2],B,",")
  for (x in B) {
    C[B[x]]= C[B[x]] " " A[1]
  }
}
END {
  for (y in C) {
    print y ":" C[y]
  }
}
```

```
basicsys@mars~/lec10>cat F4
dan:C,C++,JAVA
uri:JAVA,Perl
yossi:Perl,Bash,C
```

```
basicsys@mars~/lec10>awk -f P6 F4
Bash: yossi
JAVA: dan uri
C: dan yossi
C++: dan
Perl: uri yossi
```

## מערכים ב-awk

מערכים ב-awk הם אוסף לא מסודר של זוגות מהצורה: ערך, אינדקס. כאשר האינדקסים והערכים הם מחרוזות. מערכים מהסוג הזה נקראים מערכים אסוציאטיביים.

איתחול איבר במערך מתבצע על ידי פקודה:

ערך = [אינדקס] שם המערך

לדוגמה ההפקודה `x["ab"]=12` מאתחלת את המערך `a` כך שבאינדקס `ab` הערך הוא `12`.

הוצאת איבר ממערך מתבצעת על ידי פקודה:

[אינדקס] שם המערך `delete`

לדוגמה הפקודה `delete x["ab"]` מוציאה את האיבר בעל אינדקס `ab` מהמערך. מחיקת כל אברי המערך מתבצעת על ידי פקודה:

שם המערך `delete`

לדוגמה הפקודה `delete x` מוחקת את כל אברי המערך `x`.

לבדיקה האם איבר נמצא במערך אפשר להשתמש בפקודה:

{...} ( שם המערך in אינדקס ) `if`

לדוגמה `{...} if ("ab" in x)` בודק אם האיבר `ab` נמצא במערך `x`.

לקבלת מספר אברי המערך אפשר להשתמש בפקודה: `length( שם המערך )`

לסריקת כל אברי המערך אפשר להשתמש במבנה הבא:

{ ... } ( שם המערך in שם משתנה ) `for`

בכל מעבר בלולאה המשתנה יקבל ערך של אינדקס של איבר במערך.

הסדר של אברי המערך נקבע על ידי המערכת ולכן אין לדעת באיזה סדר יסרקו אברי המערך.

### הפונקציה length ב-awk

מבנה הפונקציה: (שם משתנה) length

אם המשתנה מכיל מחרוזת הפונקציה מחזירה את מספר התווים במחרוזת,  
אם המשתנה מכיל מערך הפונקציה מחזירה את מספר האיברים במערך.

### הפונקציה substr ב-awk

מבנה הפונקציה: (מספר 2, מספר 1, מחרוזת 1) substr

מחזירה את תת המחרוזת של מחרוזת 1 החל מתו מספר 1 (מספור התווים מתחיל מ-1) כאשר לוקחים מספר 2 תווים. לדוגמה אם המשתנה s מכיל את המחרוזת abcde אזי substr(s,2,3) מחזירה bcd.

### הפונקציה sub ב-awk

מבנה הפונקציה: (מחרוזת 2, מחרוזת 1, ביטוי רגולארי) sub

משנה את מחרוזת 2 על ידי החלפת תת המחרוזת (הראשונה משמאל) של מחרוזת 2 שמתאימה לביטוי הרגולארי במחרוזת 1. לדוגמה אם המשתנה s מכיל את המחרוזת cdababd אזי לאחר ביצוע sub("(ab)+", "xyz", s) המשתנה s יכיל cdxyzd

אם מחרוזת 2 לא קיימת, במקומה מופיע \$0 (דהינו השורה הנוכחית ש-awk עובד עליה), ואז השורה הנוכחית משתנה בהתאם לכללים הנ"ל.

### הפונקציה gsub ב-awk

מבנה הפונקציה: (מחרוזת 2, מחרוזת 1, ביטוי רגולארי) gsub

הפונקציה פועלת באופן דומה לפונקציה sub אלא שמבצעת ההחלפות של כל תתי המחרוזת של מחרוזת 2 שמתאימות לביטוי הרגולארי. לדוגמה, אם המשתנה s מכיל את המחרוזת cdababwababz אזי לאחר ביצוע gsub("(ab)+", "xyz", s) המשתנה s יכיל cdxyzwxyz

## המשתנה FS ב- awk

המשתנה FS מכיל ביטוי רגולארי שלפיו awk מפריד את שדות שורת הקלט. דהינו הערך של המשתנה הזה משפיע על האופן שבו יוצבו ערך למשתנים NF \$1 \$2 וכו' לדוגמה אם ערך המשתנה הוא (ab)+ ושורת הקלט היא zababcdcdababef אזי \$1 יקבל ערך z, \$2 יקבל ערך cdc \$3 יקבל ערך ef ו- NF יקבל ערך 3.

כברירת מחדל המשתנה FS מכיל תו רווח בודד ואז (באופן מיוחד) awk מפריד את שדות הקלט לפי מילים (דהינו מצמצם רצפים של רווחים ומכנים ל- \$1 את המילה הראשונה, ל- \$2 את המילה השניה וכו').

אם המשתנה FS מכיל מחרוזת ריקה אז awk (באופן מיוחד) מפריד את שדות הקלט לפי תווים. לדוגמה אם שורת הקלט מכילה abc והערך של FS הוא מחרוזת ריקה, אזי \$1 יכיל a, \$2 יכיל b, \$3 יכיל c ו- NF יכיל 3.

משנה את מחרוזת 2 על ידי החלפת תת המחרוזת (הראשונה משמאל) של מחרוזת 2 שמתאימה לביטוי הרגולארי במחרוזת 1. לדוגמה אם המשתנה s מכיל את המחרוזת cdababd אזי לאחר ביצוע `sub(/(ab)+/, "xyz", s)` המשתנה s יכיל cdxyzd

## הפונקציה split ב- awk

מבנה הפונקציה: `split(ביטוי רגולארי, שם מערך, מחרוזת)`

מפרידה את המחרוזת לחלקים בהתאם לביטוי הרגולארי ומכניסה את החלקים שהופרדו כאברי המערך ששמו ניתן בפרמטר השני (האינדקסים של אברי המערך הם מספרים שלמים החל מ- 1).

לדוגמה לאחר הקריאה לפונקציה `split("xababcdabe", A, "(ab)+")` המערך A יכיל את שלושת האיברים הבאים בלבד: `A[1]=x, A[2]=cd, A[3]=e` אם המערך A לא היה קיים לפני הקריאה לפונקציה, אזי הוא נוצר על ידי הפונקציה. אם המערך A היה קיים לפני הקריאה לפונקציה, אזי האיברים שהיו בו לפני הקריאה לפונקציה ימחקו ולאחר הקריאה לפונקציה הוא יכיל רק את שלושת האיברים הנ"ל.

אם בקריאה לפונקציה לא מופיע ביטוי רגולארי, אזי ההפרדה של המחרוזת לחלקים תהיה לפי הערך של המשתנה FS, דהינו כפי שמתבצעת ההפרדה של שורת הקלט הנוכחית למשתנים \$1,\$2,\$3...

## פונקציות שמוגדרות על ידי המשתמש ב-awk

המבנה של הגדרת פונקציה:

(רשימת ארגומנטים) שם הפונקציה function

{

גוף הפונקציה

}

המשתנים ברשימת הארגומנטים מקבלים ערך בזמן הקריאה לפונקציה.

משתנים שאינם מערכים מועברים by value ומשתנים מסוג מערך

מועברים by reference. המשתנים שברשימת הארגומנטים הם לוקליים לפונקציה, ביציאה מהפונקציה הם נעלמים.

לדוגמה, אם  $x$  הוא משתנה בעל ערך 5 ונקרא לפונקציה  $f1(x)$  ובהגדרת הפונקציה רשום:

```
function f1(a)
```

אזי המשתנה  $a$  בתחילת הפונקציה יקבל ערך 5. השמת ערך 6 למשתנה  $a$  בתוך הפונקציה לא תשנה את ערך המשתנה  $x$ . לכן העברת פרמטרים כזו נקראת העברה by value.

אם  $x$  הוא משתנה מסוג מערך ו- $x[1]$  יש ערך 5 ונקרא לפונקציה  $f1(x)$  ובתוך הפונקציה יהיה רשום  $a[1]=8$  אזי ביציאה מהפונקציה הערך של  $x[1]$  יהיה 8. לכן העברת פרמטרים כזו נקראת העברה by reference.

הפקודה `return` יוצאת מהפונקציה ומחזירה ערך. אם לא רשום ערך

אז הערך המוחזר מהפונקציה הוא `null`. אם פונקציה מסתיימת ללא פקודת

`return` אזי הערך המוחזר מהפונקציה אינו מוגדר.