

הרצאה מספר 12

מחיקת אברי מערך

הפקודה: [אינדקס] שם מערך delete מוחקת איבר אחד מהמערך.

הפקודה: שם מערך delete מוחקת את כל אברי המערך.

```
basicsys@mars~/lec12>cat P1
BEGIN{ A["ab"]=12;A["cd"]=13;A["ef"]=18
  print length(A)
  delete A["cd"]
  print length(A)
  A["ef"]=" "
  print length(A)
  delete A
  print length(A)
}
```

```
basicsys@mars~/lec12>awk -f P1
3
2
2
0
```

פונקציות ב-Bash

הגדרת פונקציה:

שם הפונקציה {
function

גוף הפונקציה
}

צורה נוספת להגדרת פונקציה:

שם הפונקציה () {
גוף הפונקציה
}

אפשר לקרוא לפונקציה רק לאחר שהפונקציה הוגדרה.

המשתנים של הפונקציה

המשתנים של הפונקציה שלא הוגדרו על ידי הפקודה local (שתואר בהמשך) מוכרים גם בסביבה החיצונית, והפונקציה יכולה לשנות אותם.

```
basicsys@mars~/lec12>cat P1
x=8
function f1 {
  echo "inside f1 x=$x"
  x=9
  y=10
}
echo "outside before calling f1 x=$x y=$y"
f1
echo "outside after calling f1 x=$x y=$y"
```

```
basicsys@mars~/lec12>P1
outside before calling f1 x=8 y=
inside f1 x=8
outside after calling f1 x=9 y=10
```

הפקודה local

הפקודה: שם משתנה local

מגדירה משתנה מקומי שמוכר רק בתוך הפונקציה, ונעלם ביציאה מהפונקציה.

```
basicsys@mars~/lec12>cat P1
x=8
function f1 {
  echo "inside f1 x=$x"
  local x
  local y=10
  echo "inside f1 after local statement x=$x y=$y"
  x=9
}
echo "outside before calling f1 x=$x y=$y"
f1
echo "outside after calling f1 x=$x y=$y"
```

```
basicsys@mars~/lec12>P1
outside before calling f1 x=8 y=
inside f1 x=8
inside f1 after local statement x= y=10
outside after calling f1 x=8 y=
```

העברת פרמטרים לפונקציה

העברת הפרמטרים לפונקציה מתבצעת על ידי המשתנים `$#, $@, $*, $1, $2, ...` באופן דומה להעברת פרמטרים לסקריפט. בתוך הפונקציה אין גישה לפרמטרים של הסקריפט. לדוגמה, כאשר רושמים בתוך הפונקציה `$1` הוא יוחלף בפרמטר הראשון בקריאה לפונקציה (ולא בפרמטר הראשון בקריאה לסקריפט שבתוכו מוגדרת הפונקציה).

```
basicsys@mars~/lec12>cat P1
echo $1 $2
f1() {
  echo $1 $2
  for x in "$@"
  do
    echo "$x"
  done
}
f1
f1 "20 30" 40
echo $1 $2

basicsys@mars~/lec12>P1 400 5
400 5

20 30 40
20 30
40
400 5
```

השפעת הפעלת פונקציה על ידי קריאה מהסוג (שם הפונקציה)

כאשר קוראים לפונקציה בצורה הבאה: (שם הפונקציה) נפתח תת תהליך שבו מופעלת הפונקציה הקריאה (שם הפונקציה) תוחלף על ידי הסורק בפרט של תת התהליך הזה. המשתנים של תת התהליך הזה נעלמים כאשר תת התהליך הזה מסתיים.

```
basicsys@mars~/lec12>cat P1
f1() {
  echo [$1**2 + $2**2]
  y=[$1**2 + $2**2]
}
f1 2 3
echo y=$y
x=$(f1 3 4)
echo x=$x
echo y=$y
echo "$(f1 6 7)"
echo y=$y
f1 8 9
echo y=$y
```

```
basicsys@mars~/lec12>P1
13
y=13
x=25
y=13
85
y=13
145
y=145
```

```

basicsys@mars~/lec12>cat P1
f1() {
  echo ${$1**2 + $2**2}
  y=${$1**2 + $2**2}
  f2
}
f2(){
  echo y inside f2=$y
}
f1 2 3
echo y=$y
x=$(f1 3 4)
echo x=$x
echo y=$y
echo "$(f1 6 7) "
echo y=$y
f1 8 9
echo y=$y

```

```

basicsys@mars~/lec12>P1
13
y inside f2=13
y=13
x=25 y inside f2=25
y=13
85
y inside f2=85
y=13
145
y inside f2=145
y=145

```

סקריפט ופונקציה בעלי שם זהה

כאשר יש קובץ סקריפט ופונקציה בעלי שם זהה תיקרא הפונקציה (ולא הסקריפט). כדי לקרוא לסקריפט ולא לפונקציה יש להוסיף ./ לפני הקריאה.

```

basicsys@mars~/lec12>cat f1
echo "I am file f1"

```

```

basicsys@mars~/lec12>cat P1
f1() {
  echo I am f1 in P1
}
f1
./f1
f1

```

```

basicsys@mars~/lec12>P1
I am f1 in P1
I am file f1
I am f1 in P1

```

שימוש בפונקציה בתוך exec של find

כדי לקרוא לפונקציה בתוך exec יש להצהיר עליה כסוג export ויש להוסיף bash -c מיד לאחר ה-exec כפי שמתואר בדוגמה הבאה.

```

basicsys@mars~/lec12>tree
~basicsys/win14/ex7/d2/d2
/home/cs/segel/basicsys/win14/ex7/d2/d2
|-- A
`-- d3
    |-- E
    `-- d4
        `-- S
2 directories, 3 files

```

```

basicsys@mars~/lec9>cat P1
get_file_name(){
  echo $1 | tr "/" "\n" | tail -1
}
export -f get_file_name
find $1 -type f -exec bash -c "get_file_name {}" \;

```

```

basicsys@mars~/lec12>P1 ~basicsys/win14/ex7/d2/d2
A
E
S

```

שימוש ב- {שם משתנה}

הצורה שם משתנה\$ היא קיצור ל- {שם משתנה}\$. כאשר יש שתי אפשרויות שונות לפרוש שם משתנה\$ מומלץ להשתמש ב- {שם משתנה}\$.

```
basicsys@mars~/lec12>x=abcde
basicsys@mars~/lec12>echo $x
abcde

basicsys@mars~/lec12>echo ${x}
abcde

basicsys@mars~/lec12>xx=5
basicsys@mars~/lec12>echo ${x}x
abcdex

basicsys@mars~/lec12>echo ${xx}
5

basicsys@mars~/lec12>echo $xx
5
```

פעולות על מחרוזת ב- bash

{שם משתנה#}\$ מוחלף במספר התווים שנמצאים במחרוזת שהמשתנה מכיל.

```
basicsys@mars~/lec12>x=abcdef
basicsys@mars~/lec12>echo ${#x}
6
```

הפקודה `expr` מאפשרת לבצע פעולות שונות על מחרוזות כפי שיתואר להלן.

מחרוזת `expr length`

הפקודה מדפיסה את מספר התווים שנמצאים במחרוזת.

```
basicsys@mars~/lec12>expr length $x
6

basicsys@mars~/lec12>expr match abcdef ab.[cd]
4
```

ביטוי רגולארי מחרוזת `expr match`

אם בביטוי הרגולארי אין סוגריים הפקודה מחזירה את מספר התווים בחלק של המחרוזת שהתאים לביטוי הרגולארי, אם בביטוי הרגולארי יש סוגריים אז הפקודה מחזירה את תת המחרוזת שהתאימה לסוגריים הראשונים שבביטוי הרגולארי. את ההתאמה מבצעים על התחלת המחרוזת בלבד, והביטוי הרגולארי הוא בסגנון ביטוי רגולארי בסיסי (כמו של `grep`) ולכן צריך להקדים \ לסימנים כמו () { } + וכו'.

```
basicsys@mars~/lec12>expr match abcdef a*.[cd]
3

basicsy~/lec12>expr match abcdef '\(a*.[cd]\)'
abc

~/lec12>expr match ababdcde '\(ab\)\+\(cd\)\+'
ababcdcd

~/lec12>expr match ababdcde '\(ab\)\+\)\(cd\)\+'
abab
```

קבוצת תווים מחרוזת `expr index`

מחזירה את ההופעה הראשונה במחרוזת של תו מתוך קבוצת התווים. אם התו לא מופיע במחרוזת מחזירה 0.

```
basicsys@mars~/lec12>expr index abcde db
2

basicsys@mars~/lec12>expr index abcde rc
3

basicsys@mars~/lec12>expr index abcde rt
0
```

מספר 2 מספר 1 מחזורי expr substr

מחזירה את תת המחזורות החל מתו מספר 1 כאשר לוקחים מספר 2 תווים. מספור התווים במחזורות מתחיל מ-1.

```
basicsys@mars~/lec12>expr substr abcdefg 3 4
cdef
```

```
basicsys@mars~/lec12>expr substr abcdefg 3 7
cdefg
```

{ מספר 2 : מספר 1 : שם משתנה }

מוחלף בתת המחזורות שמתקבלת מהמחזורות שהמשתנה מכיל החל מתו מספר 1 כאשר לוקחים מספר 2 תווים. במידה ומספר 2 לא קיים אז לוקחים תווים עד סוף המחזורות. מספור התווים מתחיל מ-0.

```
basicsys@mars~/lec12>x=abcdefg
```

```
basicsys@mars~/lec12>echo ${x:3:4}
defg
```

```
basicsys@mars~/lec12>echo ${x:3}
defg
```

```
basicsys@mars~/lec12>y=${x:2}
```

```
basicsys@mars~/lec12>echo "$y"
cdefg
```

{ ביטוי בסגנון גלוב # שם משתנה }

מוחלף בתת המחזורות שמתקבלת מהמחזורות שהמשתנה מכיל לאחר קיצוץ מצד שמאל של המחזורות את החלק (הקצר ביותר) שהתאים לביטוי לפי חוקי גלוב. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>echo ${x#a??}
defg
```

```
basicsys@mars~/lec12>echo ${x#a*}
bcdefg
```

{ ביטוי בסגנון גלוב ## שם משתנה }

מוחלף בתת המחזורות שמתקבלת מהמחזורות שהמשתנה מכיל לאחר קיצוץ מצד שמאל של המחזורות את החלק (הארוך ביותר) שהתאים לביטוי לפי חוקי גלוב. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>echo ${x##a*}
```

{ ביטוי בסגנון גלוב % שם משתנה }

מוחלף בתת המחזורות שמתקבלת מהמחזורות שהמשתנה מכיל לאחר קיצוץ מצד ימין של המחזורות את החלק (הקצר ביותר) שהתאים לביטוי לפי חוקי גלוב. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>echo ${x%e*}
abcd
```

```
basicsys@mars~/lec12>x=fabcdab
```

```
basicsys@mars~/lec12>echo ${x%ab*}
fabcd
```

{ ביטוי בסגנון גלוב %% שם משתנה }

מוחלף בתת המחזורות שמתקבלת מהמחזורות שהמשתנה מכיל לאחר קיצוץ מצד ימין של המחזורות את החלק (הארוך ביותר) שהתאים לביטוי לפי חוקי גלוב. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>echo ${x%%ab*}
f
```

{מחרוזת 1/ביטוי בסגנון גלוב/ שם משתנה }

מוחלף בתת המחרוזת שמתקבלת מהמחרוזת שהמשתנה מכיל לאחר החלפה אחת של תת המחרוזת הראשונה משמאל שמתאימה לביטוי (כאשר לוקחים את החלק הארוך ביותר שמתאים) לפי חוקי גלוב במחרוזת 1. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>x=ababcdab
```

```
basicsys@mars~/lec12>echo ${x/ab/zzz}
zzzabcdab
```

```
basicsys@mars~/lec12>echo ${x/ab*/zzz}
zzz
```

{מחרוזת 1/ביטוי בסגנון גלוב// שם משתנה }

מוחלף בתת המחרוזת שמתקבלת מהמחרוזת שהמשתנה מכיל לאחר החלפות של כל תתי המחרוזות שמתאימות לביטוי לפי חוקי גלוב במחרוזת 1. ערך המשתנה נשאר ללא שינוי.

```
basicsys@mars~/lec12>echo ${x//ab/zzz}
zzzzzzcdzzz
```

הפקודה rev

מבנה הפקודה: רשימת שמות קבצים rev

מדפיסה את תוכן הקבצים לאחר ביצוע reverse על השורות שלהם.

```
basicsys@mars~/lec12>cat F1
abc def
gh 123 45
```

```
basicsys@mars~/lec12>rev F1
fed cba
54 321 hg
```

```
basicsys@mars~/lec12>echo abc | rev
cba
```

הרחבת סוגריים מסוללים (brace expansion)

הביטוי {מחרוזת1,מחרוזת2,מחרוזת3,מחרוזת4} {מחרוזת5,מחרוזת6,מחרוזת7} מוחלף על ידי הסורק בכל האפשרויות של הצרופים שלהם.

```
basicsys@mars~/lec12>echo a{b,c}e
abe ace
```

```
basicsys@mars~/lec12>echo a{b,c}{1,2,3}r
ab1r ab2r ab3r ac1r ac2r ac3r
```

הביטוי {מספר1..מספר2} מוחלף על ידי הסורק בסדרת המספרים החל ממספר 1 ועד מספר 2 (כולל).

```
basics~/lec12>for x in {1..10}; do echo $x ; done
1
2
3
4
5
6
7
8
9
10
```

```
basi~/lec12>for x in $(seq 10); do echo $x ; done
1
2
3
4
5
6
7
8
9
10
```

```
basicsys@mars~/lec12>echo {1..10}
1 2 3 4 5 6 7 8 9 10
```

```
basicsys@mars~/lec12>echo {3..10}
3 4 5 6 7 8 9 10
```

```
basicsys@mars~/lec12>echo {c h}
{c h}
```

הביטוי {ת01 .. ת02} מוחלף על ידי הסורק בסדרת התווים החל מת01 ועד ת02 (כולל).

```
basicsys@mars~/lec12>echo {c..h}
c d e f g h
```

הפקודה eval

מבנה הפקודה: רשימת פרמטרים eval

הסורק עובר על הפרמטרים ויוצר מהם מחרוזת. לאחר מכן הפקודה eval גורמת לכך שהמחרוזת שנוצרה מופעלת כפקודת bash.

```
basicsys@mars~/lec12>x=3
```

```
basicsys@mars~/lec12>y=8
```

```
basicsys@mars~/lec12>echo {$x..$y}
{3..8}
```

```
basicsys@mars~/lec12>echo "{$x..$y}"
{3..8}
```

```
basicsys@mars~/lec12>echo \{$x..$y\}
{3..8}
```

```
basicsys@mars~/lec12>eval echo {$x..$y}
3 4 5 6 7 8
```

```
basicsys@mars~/lec12>echo $(echo {$x..$y})
{3..8}
```

מערכים ב-bash

הגדרת מערך: (רשימת אברי המערך מופרדים על ידי רווח) = שם המערך

אברי המערך יאותחלו באופן הבא: האיבר הראשון ברשימה יהיה באינדקס 0 האיבר השני ברשימה יהיה באינדקס 1 וכן הלאה..

```
basicsys@mars~/lec12>a=20
```

```
basicsys@mars~/lec12>echo $a
20
```

```
basicsys@mars~/lec12>a=(20 30 40)
```

```
basicsys@mars~/lec12>echo $a
20
```

{[אינדקס]שם מערך}

מוחלף בערך של האיבר במערך שנמצא באינדקס הנתון.

```
basicsys@mars~/lec12>echo ${a[0]}
20
```

```
basicsys@mars~/lec12>echo ${a[1]}
30
```

```
basicsys@mars~/lec12>echo ${a[2]}
40
```

{[שם מערך]}

מוחלף ברשימת כל אברי המערך (לאחר צמצום רווחים).

```
basicsys@mars~/lec12>echo ${a[@]}
20 30 40
```

```
basicsys@mars~/lec12>echo "${a[@]}"
20 30 40
```

"\${@} שם מערך"

מוחלף ברשימת כל אברי המערך (ללא צמצום רווחים).

```
basicsys@mars~/lec12>a=("ab 10" "cd 20" 40)
```

```
basicsys@mars~/lec12>echo "${a[@]}"
ab 10 cd 20 40
```

```
basicsys@mars~/lec12>echo ${a[@]}
ab 10 cd 20 40
```

```
basicsys@mars~/lec12>a[1]="40 1111"
```

```
basicsys@mars~/lec12>echo "${a[@]}"
ab 10 40 1111 40
```

"\${#@} שם מערך"

מוחלף במספר האיברים במערך.

```
basicsys@mars~/lec12>b=({2..5}{a..c})
```

```
basicsys@mars~/lec12>echo "${b[@]}"
2a 2b 2c 3a 3b 3c 4a 4b 4c 5a 5b 5c
```

```
basicsys@mars~/lec12>echo "${#b[@]}"
12
```

```
basicsys@mars~/lec12>echo "${#a[@]}"
3
```

```
basicsys@mars~/lec12>echo ${#a[@]}
3
```

```
basicsys@mars~/lec12>echo ${a[@]}
ab 10 40 1111 40
```

```
basicsys@mars~/lec12>echo "${b[@]}"
2a 2b 2c 3a 3b 3c 4a 4b 4c 5a 5b 5c
```

"מספר : מספר 1 : @] שם מערך"

מוחלף ברשימה שמתקבלת מאברי המערך החל מאיבר מספר 1 כאשר לוקחים מספר איברים. מספור אברי המערך מתחיל מ-0.

```
basicsys@mars~/lec12>echo "${b[@]:2:3}"
2c 3a 3b
```

```
basicsys@mars~/lec12>b=({a..c}{1..2}{d..f})
```

```
basicsys@mars~/lec12>echo "${b[@]}"
ald ale a1f a2d a2e a2f b1d b1e b1f b2d b2e b2f
c1d c1e c1f c2d c2e c2f
```

```
basicsys@mars~/lec12>echo ${#b[@]}
18
```

"\${#} שם מערך"

מוחלף במספר התווים באיבר הראשון במערך (זאת אומרת האיבר שנמצא באינדקס 0).

```
basicsys@mars~/lec12>echo ${#b}
3
```

```
basicsys@mars~/lec12>echo ${b}
ald
```

"ביטוי בסגנון גלוב # / @] שם מערך"

מוחלף ברשימת אברי המערך לאחר ביצוע קיצוץ משמאל בכל אחד מאברי המערך של החלק (הארוך ביותר) שמתאים לביטוי הרגולארי.

```
basicsys@mars~/lec12>echo ${b[@]/#?}
1d 1e 1f 2d 2e 2f 1d 1e 1f 2d 2e 2f 1d 1e 1f 2d
2e 2f
```

"ביטוי בסגנון גלוב % / @] שם מערך"

מוחלף ברשימת אברי המערך לאחר ביצוע קיצוץ מימין בכל אחד מאברי המערך של החלק (הקצר ביותר) שמתאים לביטוי הרגולארי.

```
basicsys@mars~/lec12>echo ${b[@]/%?}
a1 a1 a1 a2 a2 a2 b1 b1 b1 b2 b2 b2 c1 c1 c1 c2
c2 c2
```


באופן דומה ניתן לבצע את שאר הפעולות על מחרוזות על כל אחד מאברי המערך כפי שמוצג בדוגמאות הבאות.

```
basicsys@mars~/lec12>echo ${b[@]/%?}
ald ale alf a2d a2e a2f b1d b1e b1f b2d b2e b2f
c1d c1e c1f c2d c2e c2f
```

```
basicsys@mars~/lec12>echo ${b[@]/a/z}
zzz1d zzz1e zzz1f zzz2d zzz2e zzz2f b1d b1e b1f
b2d b2e b2f c1d c1e c1f c2d
c2ec2f
```

```
basicsys@mars~/lec12>b=({a..c}1{a..c})
```

```
basicsys@mars~/lec12>echo ${b[@]}
ala alb alc b1a b1b b1c c1a c1b c1c
```

```
basicsys@mars~/lec12>echo ${b[@]/a/z}
zzz1a zzz1b zzz1c b1zzz b1b b1c c1zzz c1b c1c
```

```
basicsys@mars~/lec12>echo ${b[@]//a/z}
zzz1zzz zzz1b zzz1c b1zzz b1b b1c c1zzz c1b c1c
```