

דוגמאות שהוצגו בהרצאה 10 בקורס יסודות מערכות פתוחות

דוגמה 1

דוגמאות של פונקציות ב-awk שמראות שהעברת פרמטרים של משתנים פשוטים היא by value והעברת פרמטרים של מערכים היא by reference וכן דוגמאות שמראות שמשתנים שמופיעים בארגומנטים של הפונקציה (זאת אומרת בתוך הסוגריים של הגדרת הפונקציה) הם משתנים לוקליים.

הדוגמאות שהוצגו בכיתה ניתנות לצפייה בחלק 1 של סרטון 10. דוגמאות נוספות נמצאות בחוברת בעמודים 198-201.

דוגמה 2

התכנית הבאה מבצעת את המשימה של דוגמה 5 בהרצאה בהרצאה 9 תוך שימוש בפונקציה. נזכיר את הדרישה:

התכנית P1 הבאה היא תכנית ב-awk שמקבלת קלט רשימת שמות קבצים שמכילים מספרים (לא בהכרח מטריצה ריבועית) ומדפיסה לפלט שורה עבור כל קובץ שמכילה את שם הקובץ ואת סכום המספרים בקובץ שגדולים מהמספר המתאים להם בשורה הראשונה. (אם למספר מסוים אין מספר מתאים לו בשורה הראשונה אז לא לוקחים אותו). על שורות הפלט להיות ממוינות לפי הסכומים שמופיעים בפלט, בסדר מספרי עולה.

לדוגמה, עבור הקבצים שתוארו בדוגמה 5 של הרצאה 9 לאחר הפעלת התכנית על ידי הפקודה:

P1 F1 A1 B1

מתקבל הפלט:

B1:15

F1:43

A1:80

```
#!/bin/awk -f
function calc_file(file,i,j,A,B) {
    i=1
    while (getline<file == 1){
        for (j=1;j<=NF;j++) {
            A[i,j]=$j
        }
        B[i]=NF
        i++
    }
    close(file)
    n=i-1
    s=0
    for (i=1;i<=n;i++){
        for (j=1;j<=B[i];j++){
            if (A[1,j]!=" " && A[i,j]>A[1,j]){
                s+=A[i,j]
            }
        }
    }
    return s
}
BEGIN {
    for (x=1;x<ARGC;x++) {
        file=ARGV[x]
        print calc_file(file) "file > "tmp"
    }
    system ("sort -n tmp >| tmp1")
    while (getline<"tmp1"==1) {
        print $2":"$1
    }
}
```

להלן התכנית בפורמט טקסט:

```
#!/bin/awk -f
function calc_file(file,i,j,A,B) {
    i=1
    while (getline<file == 1){
        for (j=1;j<=NF;j++) {
            A[i,j]=$j
        }
        B[i]=NF
        i++
    }
    close(file)
    n=i-1
    s=0
    for (i=1;i<=n;i++){
        for (j=1;j<=B[i];j++){
            if (A[1,j]!="" && A[i,j]>A[1,j]){
                s+=A[i,j]
            }
        }
    }
    return s
}
BEGIN {
    for (x=1;x<ARGC;x++) {
        file=ARGV[x]
        print calc_file(file) " "file > "tmp"
    }
    system ("sort -n tmp >| tmp1")
    while (getline<"tmp1"==1) {
        print $2:"$1
    }
}
```

דוגמה 3

התכנית P1 הבאה מבקשת קלט מהמשתמש ומחכה לתו `\n`. לאחר שהמשתמש מקליד את התו `\n` התכנית מדפיסה `Hello world` (בתוספת ירידת שורה) ומסתימת.

להלן התכנית:

```
#!/usr/bin/expect
set timeout 20
expect "\n" {send "Hello world\n"}
```

ובפורמט טקסט:

```
#!/usr/bin/expect
set timeout 20
expect "\n" {send "Hello world\n"}
```

דוגמה 4

התכנית P1 הבאה מקבלת שני פרמטרים מהמשתמש ומבקשת קלט מהמשתמש ומחכה שהמשתמש יקליד מחרוזת שמתאימה לביטוי שהועבר בפרמטר הראשון (לפי חוקי גלוב). לאחר שהמשתמש מקליד מחרוזת שמתאימה לביטוי שהועבר בפרמטר הראשון התכנית מדפיסה לפלט את הפרמטר השני (בתוספת ירידת שורה).

לדוגמה, לאחר הפעלת התכנית על ידי הפקודה:

```
P1 hi hello
```

התכנית תחכה שהמשתמש יקליד `hi` ולאחר מכן התכנית תדפיס `.hello`.

להלן התכנית:

```
#!/usr/bin/expect
set x [lindex $argv 0]
set y [lindex $argv 1]
expect "$x" {send "$y\n"}
```

ובפורמט טקסט:

```
#!/usr/bin/expect
set x [lindex $argv 0]
set y [lindex $argv 1]
expect "$x" {send "$y\n"}
```

דוגמה 5

התכנית P2 הבאה (תכנית ב-bash) מבקשת מהמשתמש 2 מספרים ומדפיסה את סכומם.
התכנית P1 הבאה (תכנית expect) מפעילה את התכנית P2 וכשהתכנית P2 מבקשת 2 מספרים מהמשתמש, התכנית P1 שולחת את המספרים האלה במקומו (גמד קטן שעושה את העבודה במקום המשתמש).

להלן התכנית P2:

```
echo -n "Enter number 1: "; read x
echo -n "Enter number 2: "; read y
echo "The sum of the numbers is: $[${x+$y}]"
```

להלן התכנית P1:

```
#!/usr/bin/expect
spawn P2
expect "number 1" {send "10\n"}
expect "number 2" {send "30\n"}
expect "The" {exit}
```

לאחר הפעלת התכנית P1 מתקבל הפלט:

```
spawn P2
Enter number 1: 10
Enter number 2: 30
The sum of the numbers is: 40
```

ובפורמט טקסט
התכנית P2:

```
echo -n "Enter number 1: "; read x
echo -n "Enter number 2: "; read y
echo "The sum of the numbers is: $[${x+$y}]"
```

והתכנית P1:

```
#!/usr/bin/expect
spawn P2
expect "number 1" {send "10\n"}
expect "number 2" {send "30\n"}
expect "The" {exit}
```

דוגמה 6

התכנית P1 הבאה מבצעת אותה משימה כמו התכנית הקודמת אבל דואגת לכך שההודעה spawn P2 לא תודפס לפלט.

```
#!/usr/bin/expect
log_user 0
spawn P2
log_user 1
expect "number 1" {send "10\n"}
expect "number 2" {send "30\n"}
expect "The" {exit}
```

ובפורמט טקסט:

```
#!/usr/bin/expect
log_user 0
spawn P2
log_user 1
expect "number 1" {send "10\n"}
expect "number 2" {send "30\n"}
expect "The" {exit}
```

דוגמה 7

התכנית P1 הבאה ממשתמשת בשרת `rainmaker.wunderground.com` כדי לקבל את נתוני מזג האוויר של עיר שהקידוד שלה (ב-3 אותיות) מתקבל בפרמטר לתכנית.

לדוגמה, לאחר הפעלת התכנית על ידי הפקודה:

P1 NYC

מתקבל הפלט:

```
NYC
Weather Conditions at 03:51 AM EST on 03 Jan 2018 for New York JFK, NY.
```

Temp (F)	Humidity (%)	Wind (mph)	Pressure (in)	Weather
=				
18	52%	WEST at 16	30.36	Partly Cloudy

```
Forecast for New York, NY
344 am EST Wed Jan 3 2018
```

```
...Winter Storm Watch in effect from late tonight through
Thursday evening...
```

.Today...Sunny. Highs in the upper 20s. West winds around 5 mph, becoming south this afternoon.
.Tonight...Mostly cloudy. A chance of snow in the evening, then snow after midnight. Snow accumulation around an inch. Lows in the lower 20s. East winds 5 to 10 mph, becoming north 10 to 15 mph with gusts up to 25 mph after midnight. Chance of snow 80 percent.
.Thursday...Snow. Total snow accumulation of 3 to 5 inches. Blustery with highs in the upper 20s. Northwest winds 15 to 25 mph with gusts up to 40 mph. Chance of snow 90 percent.
.Thursday night...Mostly cloudy with a chance of snow in the evening, then partly cloudy after midnight. Lows around 10 above.
Press Return to continue, M to return to menu, X to exit: x

להלן התכנית:

```
#!/usr/bin/expect
set city [lindex $argv 0]
log_user 0
spawn telnet rainmaker.wunderground.com
expect "Press Return to continue:" {send "\n"}
expect "city code" {send "$city\n"}
log_user 1
expect "exit" {send "x\n"}
expect "x" {exit}
```

ובפורמט טקסט:

```
#!/usr/bin/expect
set city [lindex $argv 0]
log_user 0
spawn telnet rainmaker.wunderground.com
expect "Press Return to continue:" {send "\n"}
expect "city code" {send "$city\n"}
log_user 1
expect "exit" {send "x\n"}
expect "x" {exit}
```

דוגמה 8

התכנית P1 הבאה מפעילה את התכנית הקודמת (בהנחה ששמה של התכנית הקודמת הוא B1) ומדפיסה את הטמפרטורה בעיר שהתכנית מקבלת כפרמטר. התכנית משתמשת ב- `awk` כדי לעשות את החישוב של המרה מפרנהייט למעלות לפי הנוסחה:

$$(x-32) * 5/9$$

מאחר והחישוב אינו בשלמים התכנית משתמשת ב- `awk` לצורך החישוב. כדי שבתוצאת החישוב תוצג רק ספרה אחת אחרי הנקודה התכנית מציבה ערך "1f.%" למשתנה `OFMT` של `awk`.

לדוגמה לאחר הפעלת התכנית על ידי הפקודה:

```
P1 NYC
```

מתקבל הפלט:

```
The temperature in New York is -7.8 degrees celsius.
```

להלן התכנית:

```
B1 $1 >| tmp
city=$(cat tmp | head -2| tail -1| awk '{print $12,$13}')
faren=$(cat tmp | head -5| tail -1| awk '{print $2}')
cels=$(echo $faren | awk 'BEGIN{OFMT="%.1f"}{print ($1-32)*5/9}')
echo "The temperature in $city is $cels degrees celsius."
```

ובפורמט טקסט:

```
B1 $1 >| tmp
city=$(cat tmp | head -2| tail -1| awk '{print $12,$13}')
faren=$(cat tmp | head -5| tail -1| awk '{print $2}')
cels=$(echo $faren | awk 'BEGIN{OFMT="%.1f"}{print ($1-32)*5/9}')
echo "The temperature in $city is $cels degrees celsius."
```


דוגמה 9

התכנית P1 הבאה משתמשת ב- `elinks` כדי לגלוש באתר שמציג שעות של ערים בעולם. התכנית נכנסת לאתר, קופצת לשדה מספר 9 באתר (בשדה זה מקלידים את שם העיר לחיפוש), ולאחר מכן התכנית מכניסה Netanya לשדה החיפוש, ולאחר מכן התכנית מבצעת את הפקודה `interact` שמעבירה את השליטה למשתמש. לאחר הפעלת התכנית על ידי הפקודה: P1 מוצג המסך הבא שבו המשתמש יכול להמשיך לגלוש באמצעות `elinks`.

```
Netanya Time and Date | Israel | CheckTimes.com (1/8)
Link: [1]canonical
      [2] [IMG]

* [3]Home
* [4]U.S. Directory
* [5]World Directory
* [6]About

Current Time in Netanya, Central District, Israel

12:23
PM
IST

Wednesday, January 3rd 2018

[7]Date and Time Info
http://www.checktimes.com/world/asia/il/central_district/netan[-----]
להלן התכנית:
```

```
#!/usr/bin/expect
spawn elinks http://www.checktimes.com/
sleep 3
expect "Current" {send "9\n"}
expect "Value" {send "\nNetanya\n"}
expect "form" {send "\n"}
interact
```

ובפורמט טקסט:

```
#!/usr/bin/expect
spawn elinks http://www.checktimes.com/
sleep 3
expect "Current" {send "9\n"}
expect "Value" {send "\nNetanya\n"}
expect "form" {send "\n"}
interact
```

התכנית P1 הבאה מקבלת כפרמטר שם עיר ומשתמשת ב-
elinks כדי לגלוש באתר שמציג שעות של ערים בעולם.
התכנית נכנסת לאתר, קופצת לשדה מספר 9 באתר (בשדה זה
מקלידים את שם העיר לחיפוש), לאחר מכן התכנית מכניסה
את העיר שהתכנית קיבלה כפרמטר לשדה החיפוש, ולאחר מכן
התכנית שומרת את התוצאה שהאתר מציג על המסך בקובץ tmp
והתכנית יוצאת מ- elinks.

לאחר הפעלת התכנית על ידי הפקודה:

P1 Netanya

לא מתקבל פלט על המסך. אבל קובץ tmp מכיל את הנתונים
עבור נתניה.
למשל כשמבצעים את הפקודה more tmp מקבלים במסך
הראשון:

Link: [1]canonical
[2][IMG]

- * [3]Home
- * [4]U.S. Directory
- * [5]World Directory
- * [6]About

Current Time in Netanya, Central District, Israel

12:33
PM
IST

Wednesday, January 3rd 2018

[7]Date and Time Info

--More-- (5%)

להלן התכנית:

```
#!/usr/bin/expect
file delete -force tmp
#set timeout 2
set city [lindex $argv 0]
set Alt "\033"
set Ctrl_u "\025"
spawn elinks http://www.checktimes.com/
sleep 4
expect "Current" {send "9\n"}
expect "Value" {send "\n${Ctrl_u}${city}\n"}
expect "form" {send "\n"}
sleep 4
expect "Time" {send "${Alt}fvtmp\n"}
expect * {send "q"}
expect * {send "y"}
expect
```

ובפורמט טקסט:

```
#!/usr/bin/expect
file delete -force tmp
#set timeout 2
set city [lindex $argv 0]
set Alt "\033"
set Ctrl_u "\025"
spawn elinks http://www.checktimes.com/
sleep 4
expect "Current" {send "9\n"}
expect "Value" {send "\n${Ctrl_u}${city}\n"}
expect "form" {send "\n"}
sleep 4
expect "Time" {send "${Alt}fvtmp\n"}
expect * {send "q"}
expect * {send "y"}
expect
```