דוגמה: התוכנית get_id קולטת מס׳ תעודת זהות מהמשתמש. לאחר קבלת אישור מהמשתמש שהמס׳ שקלטה נכון היא מוסיפה אותו לקובץ id_file.

```
> cat get_id
while true
  do
   echo "Please enter your teudat zehut"
   echo -n " number (9 digits): "
   read tz
   case $tz in
   [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9])
      break
      ;;
    *)
      echo "You entered $tz but you"
      echo " have to enter 9 digits. Please try again."
      continue
      ;;
   esac
 done
while true
 do
  echo -n "OK to update your details: (y/n)? "
  read flag
  case $flag in
    y)
      break
      ;;
    n)
       echo "Your details were not updated"
       echo " try again later, bye."
       exit
       ;;
     *)
       echo "You have to enter y or n"
       continue
       ;;
   esac
 done
echo "$tz" >> id_file
```


דוגמה להרצת התוכנית get_id:

```
> get_id
Please enter your teudat zehut
 number (9 digits): u123
You entered u123 but you
 have to enter 9 digits. Please try again.
Please enter your teudat zehut
 number (9 digits): 6666666666
You entered 6666666666 but you
 have to enter 9 digits. Please try again.
Please enter your teudat zehut
 number (9 digits): 777777777
OK to update your details: (y/n)? n
Your details were not updated
 try again later, bye.
```


פרמטרים לתוכנית script

ניתן להעביר פרמטרים לתוכנית script. ניתן להתיחס לפרמטרים אלה בתוך תוכנית ה-script כדלקמן:

$0 - הפקודה עצמה

$1 - הפרמטר הראשון

$* - רשימת כל הפרמטרים הנתונים לפקודה

גם $@ מציין את רשימת הפרמטרים הנתונים לפקודה. אבל בתוך גרשיים כפולים ישנו הבדל בין "$@" לבין "$*" כפי שמתואר בדוגמה הבאה:

```
>  cat display_parm1
#!/bin/sh
for u in "$*"
  do
    echo "$u"
  done
>  cat display_parm2
#!/bin/sh
for u in "$@"
  do
    echo "$u"
  done
>
> display_parm1 “123   abc”  “456     def”  789
123   abc 456     def 789
> display_parm2 “123   abc”  “456     def”  789
123   abc
456     def
789
```


הרחבת מסלול (Pathname Expansion)

ב- Bash קיימות פקודות המקבלות מספר קבצים כפרמטרים. לכן, קיימת אפשרות לציין תבנית מסוימת, כך שפעולה תתבצע על כל הקבצים אשר שמותיהם מתאימים לתבנית. את התבנית בונים בעזרת תווים הנקראים Wildcards.

קיימים שלושה Wildcards:
* - מתאים לאפס או יותר תווים כלשהם.
דוגמאות:

*.c

כל הקבצים עם סיומת c.

AD

קבצים שבשמותיהם צירוף האותיות AD.

הפקודה:

```
cat *.c
```

תציג את תוכן כל הקבצים עם הסיומת .c

? - מתאים לבדיוק תו אחד כלשהוא. דוגמאות:

```
prog?
```
כל הקבצים ששמותיהם מתחילים ב- prog, ולאחר מכן יש בדיוק תו אחד. הפקודה:
```
rm prog1.?
```
תמחק את כל הקבצים ששמותיהם מתחילים ב prog1. ואחרי הנקודה יש בדיוק תו אחד.

[<characters>] - מתאים לאות אחת מתוך characters. התוים יכולים להיות מצויינים מפורשות, או יכולים להיות מצויינים כתחום של אותיות. דוגמאות:

```
file[13]
```
מציין את כל הקבצים ששמותיהם מתחילים ב- file ושאחרי אותיות אלו מופיע או 1 או 3.

```
file[A-Za-z13]
```
מציין את כל הקבצים ששמותיהם מתחילים ב- file ושאחרי אותיות אלו מופיעה אחת מאותיות ה-א"ב האנגלי או אחת מהספרות 1 או 3.


```
file[0-9][0-9]
```
מציין את כל הקבצים ששמותיהם מתחילים ב- file ושאחרי אותיות אלו מופיעות שתי ספרות.

אם לא קיים אף קובץ ששמו מתאים לתבנית לא תתבצע הרחבת מסלול והמחרוזת תישאר כפי שהיא. לדוגמא אם אין אף קובץ שמסתיים ב- .c אז:

```
> echo *.c
*.c
```
בתוך גרשיים כפולים (או רגילים) לא מתבצעת הרחבת מסלול. לדוגמה:

```
> ls *
file1 file2
> ls "*"
ls: *: No such file or directory
```

בהשמה של ערך למשתנה לא מתבצעת הרחבת מסלול. לדוגמה לאחר ההשמה *=v הערך של v הינו * .

```
> v=*
> echo "$v"
*
> echo $v
file1 file2
```


בפקודה האחרונה מתבצעת תחילה variable expansion בו מוחלף $v ב- * ולאחר מכן מתבצעת הרחבת מסלול המחליפה את ה- * ב- file1 file2 שהיא המחרוזת המועברת לפקודה echo.

Command Aliases

alias <new name> <command name> - נותן את השם new name לפקודה command name. דוגמה:

```
> alias rm="rm –i"
> rm file2
```

פקודה זו מוחקות את הקובץ file2 לאחר קבלת אישור מהמשתמש. שים לב שהחלפת ה- alias מתבצעת פעם אחת בלבד ולכן לאחר החלפת rm ב- rm –i לא מתבצעת החלפה נוספת של rm.

כדי להמנע מהחלפת alias יש להוסיף \ להתחלת הפקודה. לדוגמה \rm file תבצע מחיקה של הקובץ file ללא בקשת אישור מהמשתמש.

alias - מציגה את כל ה aliases. דוגמא, נניח כי רק ה alias הנ"ל הוגדר:

```
> alias
alias rm='rm -i'
```
הפקודה unalias <name> מבטלת את ה- alias name. לדוגמה unalias rm מבטלת את ה- alias הנ"ל.

Pipelining

הפקודה
<program1> | <program2>

גורמת לכך ש program2 לוקחת כקלט את הפלט של program1. ניתן לבצע שרשור מסוג זה למספר תוכניות, כך שכל תוכנית לוקחת כקלט את הפלט של קודמתה, ומעבירה את הפלט שלה לבאה אחריה ברשימה.

לדוגמה:
>ls –l | more

תציג למסך את רשימת הקבצים במדריך הנוכחי, עם עצירה בכל פעם שהמסך מתמלא.

```
> who | sort
```

תציג למסך את רשימת כל האנשים המחוברים כרגע למחשב לפי סדר אלף-בת של ה- login name.

```
> cat file1 | tr “[a-z]” “[A-Z]”
```

תציג למסך את התוכן של file1 באותיות גדולות.

בד"כ ה- Standard error לא עובר ב- pine. כדי להעביר גם את ה- Standard error ב- pipe יש להשתמש בסימון 2>&1 המעביר את ה- Standard error ל- Standard output. לדוגמה:

```
> cat x y
cat: x: No such file or directory
This is file y
> cat x y | tr “[a-z]” “[A-Z]”
…
> cat x y 2>&1 | tr “[a-z]” “[A-Z]”
…
```


פקודות שימושיות לעבודה עם pipes

הפקודה
sort [*options*] [*files*]

ממיינת את שורות הקבצים [*files*], בדרך כלל בסדר א"ב.

האופציות:

- **-b** התעלם מרווחים וסימני טבולציה בתחילת השורה.
- **-d** בצע מיון בסדר מילוני (התעלם מפיסוק).
- **-f** התעלם מהבדלי uppercase/lowercase.
- **-m** merge sorted input files.
- **-n** השוה בסדר מספרי.
- **-o*file*** הדפס את הפלט לקובץ *file*.
- **-r** הפוך את סדר המיון.
- **-u** שורות זהות תופענה בפלט פעם אחת בלבד.
- **+*n*** לפני ביצוע המיון דלג על *n* המילים הראשונות. ז"א מיין מהמילה ה- n+1 עד סוף השורה. (מילה היא אוסף תוים ללא רווחים).
- **-*m* +*n*** מיין מהמילה ה- n עד המילה ה- m (כולל את מילה m).

הפקודה tee

הפקודה
tee [*options*] [*files*]

משכפלת את כל מה שנכנס לקלט הסטנדרטי שלה ושולחת אותו הן לפלט הסטנדרטי והן לכל אחד מהקבצים ב [*files*].

לדוגמא:

```
> ps | tee proc_list list
  PID TTY          TIME CMD
 3124 pts/1    00:00:00 sbash
 3199 pts/1    00:00:00 ps
 3200 pts/1    00:00:00 tee

> more proc_list
  PID TTY          TIME CMD
 3124 pts/1    00:00:00 sbash
 3199 pts/1    00:00:00 ps
 3200 pts/1    00:00:00 tee

> more list
  PID TTY          TIME CMD
 3124 pts/1    00:00:00 sbash
 3199 pts/1    00:00:00 ps
 3200 pts/1    00:00:00 tee
```

האופציה a- מאפשרת שרשור של הפלט לסוף הקבצים [*files*] במקום מחיקתם.

דוגמאות :

```
> cat short
Pear
Pear
apple
pear
Apple
> sort short

> sort –u short

> sort –uf short

> sort –u +0f +0 short

> cat cars
nisan sunny   1993    120000   40500
fiat  600     1961    300000    1500
fiat  600     1960    300000    2000
Honda civic   1997    50000    80000
honda civic   1995      3000   70000

> sort cars

> sort –f +0 –1 cars

> sort -f +0 -1 +3 cars

> sort –fu +0 –1 cars

> sort –fu +1 –2 cars

> sort +3 cars

> sort +3n cars
```


הפקודה grep

הפקודה
grep [*options*] <*reg-expr*> [*files*]

מחפשת בקבצים [*files*] שורות בהן ישנה התאמה לביטוי הרגולרי reg-expr וכותבת אותם לפלט הסטנדרטי.

כאשר ברשימת הקבצים מופיע יותר מקובץ אחד יתוסף לכל שורה בפלט שם הקובץ אליו היא שייכת. ניתן לבטל את התוספת הזו ע"י שימוש באופציה h– .

הערה: אם רוצים לחפש שורות בהן מופיע ביטוי המורכב ממספר מילים הנפרדות ע"י רווחים מכילים את הביטוי בין גרשיים.

לדוגמא:

```
>grep "It is" file
```

מחפשת בקובץ file שורות בהן מופיע הביטוי It is.

האופציות:

- **-c** הדפס רק את מספר השורות שנמצאו.
- **-h** הדפס את השורות עצמן ללא שמות הקבצים, בהם השורות נמצאו.
- **-i** התעלם מהבדלי uppercase/lowercase.
- **-l** הדפס רק את שמות הקבצים בהם השורות נמצאו ללא הדפסת השורות עצמן.
- **-n** הדפס את השורות ואת מספרן בקבצים.
- **-v** הדפס את כל השורות, בהן אין התאמה לביטוי הרגולרי reg-expr .

להלן דוגמאות לחוקי ההתאמה של הביטוי הרגולרי reg-expr .

צרוף של תוים רגילים יתאים לכל שורה המכילה צרוף זה במקום כלשהוא בתוך השורה. לדוגמה:

```
> (echo “spring 1998”; echo ringing) | \
  grep ring
```

נקודה מתאימה לתו כלשהו. לדוגמה:

```
> (echo “will talk”; echo “may balk”) | grep .alk
```

סוגריים מרובעים מציינים קבוצה של תווים. לדוגמה:

```
> (echo 12ab22; echo “ 12a”; echo “ 12Ab") | \
  grep “ 12[a-zA-Z]”
```

כאשר הסימן ^ מופיע כתו ראשון בתוך הסוגרים המרובעים הוא מסמן שקבוצת התוים המוגדרת ע"י הביטוי [^exp] הינה כל התוים שאינם בקבוצה המוגדרת ע"י הביטוי [exp]. לדוגמה:

```
> (echo ” 12a”;echo “ 12Ab”;echo “ 12.”) | \
   grep “ 12[^a-zA-Z]”
```

הסימן * מתאים ל-מס' כלשהוא של חזרות של התו שלפניו. לדוגמה:

```
> (echo aabbbbc; echo ac; echo abc; \
   echo ababc) | grep “ab*c”

> (echo “a  c”; echo ac; echo axxxyyc; \
   echo ababc) | grep “a.*c”
```

הסימן ^ מציין את התחלת השורה והסימן $ מסמן את סוף השורה. לדוגמה:

```
> (echo 1234567; echo “ 123”; echo 2123) \
   | grep ^123

> (echo “ab123 c”; echo “Ab123 C”; \
   echo nab123cc) | grep “^[aA].* c$”
```


דוגמאות לשימושים באופציות של grep:

```
> ls
ex/      ex2      ex4      file2~
ex1      ex3      file1~
> ls | grep ex
ex/
ex1
ex2
ex3
ex4
> ls | grep -c ex
5
> more file1
Arik !
Have you heard news ?
If not , listen !
> more file2
Tomorrow we have a new exam !
So we cannot go to Ran's birthday .
By .
> grep -v -n new file1 file2 | sort -n | head -3
file1:1:Arik !
file1:3:If not , listen !
file2:2:So we cannot go to Ran's birthday .
```


הפקודה uniq

הפקודה

uniq [*options*] [*file1* [*file2*]]

מורידה מהקובץ הממוין ***file1*** את כל השורות הזהות הסמוכות, ושולחת עותק אחד של כל שורה לקובץ ***file2*** (או בהעדר הקובץ ***file2***, לערוץ הפלט הסטנדרטי).

האופציות :

- **-c** הדפס כל שורה פעם אחת בלבד וספור עותקים של כל שורה.
- **-d** הדפס שורות המופיעות יותר מפעם אחת, אך לא שורות המופיעות פעם אחת בלבד.
- **-u** הדפס רק את השורות המופיעות פעם אחת בדיוק.
- ***-n*** התעלם מ- ***n*** המילים הראשונות (לפני קבלת ההחלטה אם שתי מילים זהות).

הערה: השתמש בו זמנית רק באחת משלוש האופציות הראשונות.