אתר הקורס:
http://mars.netanya.ac.il/~basicsys

להיכנס למחשב

הפעל תוכנת Xwin32 שנמצאת בתפריט תוכניות

בתפריט הפעלה (Run) הקש
```
telnet mars
```

יופיע חלון telnet שבו יש להקיש מס' חשבון וסיסמה

לפתיחת חלון חדש יש להקיש `xterm &`

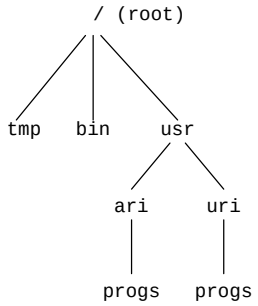
אנו נשתמש ב-shell שנקרא Bourne Again Shell או בקיצור bash

לבדיקת סוג ה-shell הקש
```
> echo $SHELL
/bin/sbash
```


קבצים ומדריכים

יש לכל משתמש חשבון שלו במחשב. חשבון זה מקנה מרחב מסוים על הדיסק שבו רק הוא יכול להשתמש.

ב- Unix, כמו ב MS-DOS, יש מדריכים שבהם ניתן לשמור קבצים קרובים ביחד. ב- Unix מבנה המדריכים נראה כדלקמן:



במערכת הנ"ל ישנם שני משתמשים, ari ו- uri. הם יכולים ליצור מדריכים עם אותם שמות והמערכת תשמור עליהם כמדריכים נפרדים (למשל progs בשרטוט).

המדריך העליון של כל משתמש הוא המדריך home שלו. למשל, המדריך home של ari הוא /usr/ari. כאשר משתמש נכנס לחשבון שלו, הוא נמצא במדריך ה home שלו.

ניתן לציין קבצים (או מדריכים) לפי ***full pathnames*** או ***relative pathnames*** (מהמדריך הנוכחי). למשל, נניח ש- ari נמצא במדריך home שלו, אז השורות הבאות מציינות את אותו קובץ:

/usr/ari/progs/hello.c
progs/hello.c

פקודות מדריכים

pwd - לראות באיזה מדריך הנך נמצא. לדוגמא:

```
>pwd
/home/cs/stud99/u5645627
```

cd <directory name> - לעבור למדריך אחר. לדוגמא, לעבור למדריך progs:
```
>cd progs
>pwd
/home/cs/stud99/u5645627/progs
```

לעבור למדריך מעל המדריך הנוכחי (parent directory):

```
> cd ..
> pwd
/home/cs/stud99/u5645627
```

לחזור ל home directory - **cd** ללא פרמטרים.

ls <directory name> - לראות רשימת הקבצים במדריך. לדוגמא, לראות את הקבצים במדריך הנוכחי:

```
> ls
Desktop  progs  test1
```

לראות את הקבצים בתת-מדריך progs:

```
> ls progs
hello  hello.c
```

לראות רשימת הקבצים במדריך הנוכחי באופן מפורט - הוספת **al-** לפקודה:

```
> ls -al
drwx------   5 u5645627 u5645627     4096 Oct 26 12:06 .
drwxr-xr-x 228 root     root         4096 Oct 24 16:22 ..
-rw-------   1 u5645627 u5645627       39 Oct 25 11:47 .bash_history
-rw-r--r--   1 u5645627 u5645627       24 Oct 24 16:20 .bash_logout
-rw-r--r--   1 u5645627 u5645627      230 Oct 24 16:20 .bash_profile
-rw-r--r--   1 u5645627 u5645627      124 Oct 24 16:20 .bashrc
-rwxr-xr-x   1 u5645627 u5645627      333 Oct 24 16:20 .emacs
drwxr-xr-x   3 u5645627 u5645627     4096 Oct 24 16:20 .kde
-rw-r--r--   1 u5645627 u5645627      435 Oct 24 16:20 .kderc
drwxr-xr-x   5 u5645627 u5645627     4096 Oct 24 16:20 Desktop
drwxr-xr-x   2 u5645627 u5645627     4096 Oct 26 12:13 progs
-rw-r--r--   1 u5645627 u5645627        4 Oct 26 12:02 test1
```

mkdir <directory name> - ליצור מדריך חדש. לדוגמא, ליצור מדריך בשם docs:

```
> mkdir docs
> ls
Desktop  docs  progs  test1
> ls docs
>
```


rmdir <directory name> - למחוק מדריך. לדוגמא, למחוק את המדריך docs:

```
> rmdir docs
> ls
Desktop  progs  test1
```

ניתן למחוק מדריך רק אם הוא ריק.

<u>פקודות קבצים</u>

- **cp** <source file> <destination file> להעתיק קובץ. לדוגמא, ליצור עותק נוסף של test1בשם test2:

```
> cp test1 test2
> ls t*
test1  test2
```

cp <source files> <destination dir> - להעתיק קבצים למדריך. לדוגמא, להעתיק את test1 ו- test2 למדריך progs:

```
> cp test1 test2 progs
> ls progs
hello  hello.c  test1  test2

> mkdir progs1
> cp test1 test2 progs progs1
cp: progs: omitting directory

> ls progs1
test1 test2
```


cp –**r** <source dir> <destination dir> להעתיק מדריך למדריך כולל תתי מדריכים.

```
> cp –r test1 test2 progs progs1
> ls progs1
progs  test1  test2

> ls progs1/progs
hello  hello.c  test1  test2
```

rm <filename-list> - למחוק את כל הקבצים המופיעים ברשימה.

```
> ls progs
hello  hello.c  test1  test2

> rm progs/test1
> ls progs
hello  hello.c  test2
```

רצוי לקבל הודעת אזהרה לפני שמוחקים קובץ לשם כך יש להשתמש ב- `rm –i`

```
> rm –i progs/hello
rm: remove `progs/hello'? y
> ls progs
hello.c test2
```

כדי למחוק מדריך כולל כל תתי המדריכים שלו יש להשתמש בפקודה `rm -r`

```
> ls progs1
progs  test1  test2

> rm –r progs1
> ls progs1
ls: progs1: No such file or directory
```


mv <source file> <destination file> - לשנות את שמו או את מיקומו של קובץ.

```
> mv progs/test2 progs/test3
> ls progs
hello.c  test3
```

לדוגמא, להזיז את הקובץ test1 ל מדריך progs:

```
> mv test1 progs
> ls progs
hello.c test1 test3
```

man <command name> - לקרוא מידע מפורט על פקודה ב - Unix. לדוגמא, לקרוא מידע על הפקודה ls:

```
> man ls
```

לקרוא מידע על הפקודה man עצמה:

```
> man man
```


דואר אלקטרוני - e-mail

אפשר לשלוח ולקבל e-mail מכל משתמש שיש לו חשבון ברשת.

שליחת דואר, שיטה 1

1) הקלד **mail** <login name> כאשר ה login name הוא שם המשתמש שאליו רוצים לשלוח את ההודעה.
2) הקלד את נושא ההודעה.
3) הקלד את לשון ההודעה.
4) הקלד "." (נקודה) בשורה נפרדת אחרי סוף ההודעה.

דוגמא:

```
> mail ari
Subject: Party Tonight
Hi Ari,
   I'm having a party tonight,
   See you there,
    Uri
.
>
```


שליחת דואר, שיטה 2 (הכי פשוטה)

1) הכן קובץ ב- **pico** (או בכל מעבד תמלילים אחר) שמכיל את ההודעה.
2) הקלד:

mail -s <Subject> <login name> < <filename>

כאשר login name זה שם המשתמש שאליו רוצים שההודעה תישלח, Subject זה נושא ההודעה, ו filename הוא הקובץ המכיל את ההודעה שהוכן מקודם.

לדוגמא: (שם הקובץ שהוכן party)

```
> mail -s "Party Tonight" ari < party
```

ניתן גם לשלוח הודעה ללא נושא, לדוגמא:

```
> mail ari < party
```

קריאת e-mail

לקרוא e-mail שקיבלת, הקלד **pine** משורת הפקודה.

פקודות לסקירת תוכן קבצים

cat <file name> - להציג קובץ על המסך. לדוגמא, להציג את תוכן הקובץ prog2.c:

```
> cat prog2.c
```

more <filename> - להציג קובץ על המסך עם עצירה אחרי כל דף.

```
> more  prog2.c
```

הפקודה **head** [-*n*] [*files*]

מדפיסה רק מספר מהשורות הראשונות של קובץ. האופציה *n*- מאפשרת להדפיס *n* מהשורות הראשונות (ברירת המחדל היא 10). לדוגמא: להדפיס את 20 השורות הראשונות בקובץ phone

```
> head -20 phone
```


הפקודה

tail [*options*] [*files*]

מאפשרת להדפיס את 10 השורות האחרונות בקובץ (ברירת מחדל). האופציה *n*- מאפשרת להדפיס את *n* השורות האחרונות בקובץ. האופציה *n*+ מאפשרת להדפיס את כל השורות האחרונות החל מהשורה ה *n* -ית בקובץ.

לדוגמא:

```
> tail main.c
...
> tail -20 main.c
...
> tail +10 main.c
```


הרשאות קבצים

כיוון ש- Unix היא מערכת הפעלה רבת משתמשים רצוי לחסום גישה חופשית של משתמש אחד לקבצים של משתמש אחר. לפיכך, לכל קובץ יש הרשאות שקובעות למי מותרת הגישה לקובץ ולצורך אילו פעולות. כאשר קובץ חדש נוצר רשום בו מי יצר אותו והוא נקרא בעל הקובץ.

יש 3 סוגים של משתמשים שעשויים לגשת לקובץ:

1. user - בעל הקובץ.
2. group - משתמש ששייך לקבוצה של בעל הקובץ.
3. Other - כל שאר המשתמשים במערכת שאינם בנ"ל.

לכל סוג משתמש יש 3 הרשאות שונות:

1. Read - האם מותר לו לקרוא (להעתיק) את הקובץ.
2. Write - האם מותר לו לכתוב על (לשנות) את הקובץ.
3. Execute - האם מותר לו להריץ את הקובץ.

הפקודה

wc [*options*] [*files*]

מאפשרת הדפסת ספירת מספר התווים המילים או השורות בקבצים שונים.

האופציות:

- c- - הדפס את מספר התווים בלבד.
- l- - הדפס את מספר השורות בלבד.
- w- - הדפס את מספר המילים בלבד.

לדוגמא:

```
> wc -l phonebook
...
> wc -w bible
...
> wc -c data
```

לפיכך, יכול בעל קובץ לחסום כל גישה לקובץ ע״י משתמשים אחרים (גם לקריאה וגם לכתיבה) ואפילו לא לאפשר לעצמו לשנותו. מצד שני יכול בעל הקובץ, למשל, לאפשר למשתמשים אחרים לקרוא ולהריץ את קובץ אך לא לשנותו.

<u>שינוי הרשאות קבצים</u>

בעזרת הפקודה chmod יכול בעל קובץ לשנות את ההרשאות של הקובץ. מבנה הפקודה הוא:

chmod *mode files*

כאשר mode הוא שרשור תוים של *who*, *opcode* ו *permission*. *who* הוא אופציונלי (ברירת המחדל היא a). בכול פקודה יש רק *opcode* אחד.

Who
u - User
g - Group
o - Other
a - All

Opcode
+ - Add permission
- - Remove permission
= - Assign permission (and remove permission of the unspecified fields).

Permission
r - Read
w - Write
x - Execute

<u>כללי הרשאות</u>

לקרוא קובץ דרושה הרשאת r לקובץ והרשאת x במדריך בו נמצא הקובץ ובכל המדריכים שמעליו.

לכתוב לקובץ דרושה הרשאת w לקובץ והרשאת x במדריך בו נמצא הקובץ ובכל המדריכים שמעליו.

למחוק קובץ דרושה הרשאת w+x במדריך בו נמצא הקובץ והרשאת x בכל המדריכים שמעליו. שים לב שבמקרה זה אין חשיבות להרשאה בקובץ עצמו, אלא רק להרשאה במדריך בו הוא נמצא.

לבצע פקודת ls למדריך דרושה הרשאת r למדריך והרשאת x לכל המדריכים שמעליו.

לבצע פקודת cp file dir דרושה הרשאת r לקובץ file הרשאת x במדריך בו נמצא הקובץ file ובכל המדריכים שמעליו. בנוסף דרושה הרשאת w+x במדריך dir והרשאת x בכל המדריכים שמעל המדריך dir.

<u>דוגמאות:</u>

הוסף הרשאת הרצה ע״י המשתמש ל file:

```
> chmod u+x file
```

הורד הרשאות כתיבה ע״י משתמשים אחרים מ file:

```
> chmod g-w,o-w file
```

קבע הרשאת קריאה בלבד לכולם בקובץ file:

```
> chmod =r file
```

קבע הרשאת קריאה/כתיבה/הרצה למשתמש, קריאה והרצה לקבוצה וקריאה בלבד לשאר המשתמשים בקובץ file:

```
> chmod u=rwx,g=rx,o=r file
```

הרחבת ~ (Tilde Expansion)

לפני ביצוע פקודה bash מחליף (במידת האפשר) את הסימן ~ לפי הכללים הבאים:

במילה ~ או במילה ~/word מוחלפת ה- ~ בשם המלא של ה- home directory של המשתמש שממנו ניתנת הפקודה. שם זה נלקח מתוך משתנה הסביבה **HOME.**

במילה ~word או במילה ~word/word1 אם קיים משתמש שה- login name שלו הוא word מוחלפת ה- ~word בשם המלא של ה- home directory של המשתמש word. אחרת ה- ~ אינה מוחלפת.

דוגמאות

```
> echo $HOME
/home/cs/stud99/u5645627
> echo ~
/home/cs/stud99/u5645627
> echo ~/yyy
/home/cs/stud99/u5645627/yyy
> echo ~opensys2
/home/cs/courses/opensys2
> echo ~abc
~abc
```


I/O redirection

בכל תוכנית שרצה תחת Unix ערוץ הקלט הסטנדרטי מחובר אל המקלדת, וערוצי הפלט והודעות השגיאה הסטנדרטיים מחוברים אל המסך כברירת מחדל.
ב- bash ניתן לכוון מחדש הן את ערוץ הקלט הסטנדרטי והן את ערוצי הפלט והשגיאה הסטנדרטיים אל ומאת קבצים כלשהם. הכיוון מחדש נקרא redirection.

Input Redirection

<program> < <filename>

תגרום ל- program לקבל את הקלט הסטנדרטי שלה מהקובץ filename.

לדוגמא:

```
> mail opensys2 < letter
```

תגרום ל- mail לקחת את הקלט מהקובץ filename.

Output Redirection

<program> > <filename>

תגרום ל- program לכתוב את הפלט הסטנדרטי שלה לקובץ filename. אם קיים כבר קובץ בשם filename לפני ביצוע הפקודה קימות שתי אפשרויות: אם האופציה שנקראת **noclobber** בתוקף הפקודה לא תתבצע. אחרת הפקודה תתבצע והתוכן הישן של הקובץ filename ימחק.

כדי להמנע מהריסה לא מכוונת של קבצים רצוי לדאוג לכך שהאופציה noclobber תהיה תמיד בתוקף ע״י הוספת השורה

```
set -C
```

לקובץ **.bashrc**

ניתן לעקוף את האופציה noclobber ע״י שימוש בפקודת הכיוון:

<program> >| <filename>

שבכל מקרה הורסת את הקובץ filename וכותבת עליו את הפלט הסטנדרטי של program.

דוגמאות:

```
> ls
file1 file2
> ls > file3
> ls
file1 file2 file3
> cat file3
file1
file2

> set -C
> echo aa > file3
sbash: file3: Cannot clobber existing
file
> echo aa >| file3
> cat file3
aa
```

הפקודה
<program> >> <filename>

מכוונת את הפלט הסטנדרטי של program לקובץ filename אך משרשרת אותו לסוף הקובץ. במקרה שהקובץ filename לא קים יוצר קובץ חדש בשם זה.

לדוגמא :

```
> cat file1
aa
bb
> cat file2
cc
dd
> cat file1 >> file2
> cat file2
cc
dd
aa
bb
```


Multiple Redirection

בנוסף להפנית קלט/פלט סטנדרטיים ניתן להפנות גם את ערוץ השגיאות הסטנדרטי מהמסך אל קבצים.
הפקודה
<command> **>&** <file>
מפנה הן את הפלט הסטנדרטי והן את ערוץ השגיאות הסטנדרטי לקובץ file.

צורה אחרת לכתיבת הפקודה הנ"ל היא :

<command> **1>** <file> **2>&1**

המספר **1** מסמן את קובץ הפלט הסטנדרטי והמספר **2** מסמן את ערוץ השגיאות הסטנדרטי.
בפקודה הנ"ל **1>** מסמן שכתיבה לקובץ שמספרו 1 תופנה לקובץ <file> ו- **2>&1** מסמן שכל כתיבה לקובץ שמספרו 2 תופנה לקובץ שמספרו 1 .
במילים אחרות, כתיבה לפלט הסטנדרטי תופנה לקובץ <file> וכתיבה לערוץ השגיאות תופנה לפלט הסטנדרטי וממנו תופנה לקובץ <file>.
לכן הפקודה הנ"ל מפנה גם את הפלט הסטנדרטי וגם את הקלט הסטנדרטי לקובץ <file>.

הפקודה
<command> **1>** <file1> **2>** <file2>

מפנה את הפלט הסטנדרטי לקובץ file1 ואת ערוץ השגיאות הסטנדרטי לקובץ file2.

דוגמאות

```
> cat prog1
echo example
ttt
 > prog1
example
/home/cs/stud99/u5645627/./prog1: ttt: command not found

> prog1 >| file1
/home/cs/stud99/u5645627/./prog1: ttt: command not found
> cat file1
example

> prog1 1>| file1 2>&1
> cat file1
example
/home/cs/stud99/u5645627/./prog1: ttt: command not found

> prog1 1>| file1 2>| file2
> cat file1
example
> cat file2
/home/cs/stud99/u5645627/./prog1: ttt: command not found
> prog1 1>> file1 2>&1
> cat file1
example
example
/home/cs/stud99/u5645627/./prog1: ttt: command not found
```