עבודה עם Shell

לאחר התחברות ל-t2 לצורך עבודה אינטראקטיבית, מורצת תכנית (תהליך) מיוחדת הנקראת Shell. לאחר הרצת ה-Shell, מופיע **prompt** המסמן כי המחשב מוכן לקבל פקודות מהמשתמש.

מהו shell?
Shell הוא כינוי לתוכנה המקשרת בין המשתמש לבין גרעין מערכת ההפעלה.
בפועל הshell מקבל פקודה מהמשתמש, מבצע החלפות טקסט בפקודה בהתאם לתכונות ה-shell ולבסוף שליחת הפקודה המעובדת למערכת ההפעלה.

ישנם כמה סוגי Shell אפשריים, כגון bash, ksh, sh, csh, tcsh. במחשב ה-t2 מוגדר לכם ה- Shell להיות ***tcsh***. (זהה ל- c-shell).

חשוב!

- עם הכניסה למחשב יש להריץ את הפקודה bash כדי לעבוד עם ה- shell הנכון. ניתן לקבוע את bash לברירת המחדל ע"י הפקודה chsh.
- הורידו את הקובץ bashrc. (התו הראשון בשם הקובץ הוא נקודה) מאתר הקורס ומקמו אותו בתיקיה הראשית של חשבונכם. קובץ זה מכיל פקודות שרצות בעליית bash ויוצרות סביבת עבודה נוחה יותר.

lion:eesoft> **cp ~eesoft/bash/.bashrc .**

יציאה מה-Shell

כדי לצאת מה-Shell יש להשתמש בפקודה logout:

lion:eesoft> **logout**

מערכת הקבצים ב-UNIX

מערכת ההפעלה אחראית בין השאר על ארגון הקבצים במחשב.
מערכת הקבצים ב-UNIX מורכבת מקבצים ומדריכים.
קובץ הוא אוסף סדור של תווים, ומדריך מאפשר שמירת מספר קבצים ותיקיות נוספות בצורה מסודרת.
לכל קובץ או תיקייה יש שם, לא חייבת להיות סיומת.
נשים לב ששמות קבצים ותיקיות לא יכולים להכיל את התו '/' מאחר והוא משמש לסימון מסלולים בתיקיות במערכת הקבצים.

זכויות גישה לקובץ (הרשאות)

מאחר שמערכת ההפעלה UNIX הינה **מרובת משתמשים**, יש צורך למנוע גישה חופשית של משתמשים לא רצויים – הן לצורך שמירה על פרטיות, והן לצורך Security.
לצורך נוחות הוגדרו **3 סוגי (ורמות) משתמשים**:

- **user** – משתמש בודד.
- **group** – קבוצת משתמשים כלשהי אליה שייך המשתמש.
- **other** – כל שאר המשתמשים.

- לכל סוג משתמש יש **3 הרשאות שונות** בלתי תלויות:
- **read** – הרשאת קריאה של המידע (הצגת שמות קבצים במדריך).
- **write** – הרשאת כתיבת מידע (הוספה/מחיקת קבצים במדריך).
- **execute** – הרשאת הרצת תכנית ("כניסה" למדריך).

- ב-UNIX מוגדרים לכל קובץ **3 פרמטרים** הקשורים בהרשאות:
- **user(owner)** – שם המשתמש בעל הקובץ.
- **group** – שם קבוצה אליה שייך הקובץ.
- **mode** – ערך המציין את אופי ההרשאות של הקובץ. דוגמא ל-mode:

מציין כי זהו מדריך עם הרשאות קריאה ו"כניסה" לכולם, והרשאת כתיבה לבעל המדריך בלבד.	`drwxr-xr-x`
מציין כי זהו קובץ עם הרשאות קריאה לכולם, והרשאת כתיבה לבעל הקובץ בלבד.	`-rw-r--r--`

3

דוגמה ל3 פרמטרים בUnix הקשורים בהרשאות:
על ידי כתיבת הפקודה ls –l
מקבלים:

```
-rwxr-xr-x  1 shirel users  6502 Feb 20 11:45 a.out
-rw-r--r--  1 shirel users     0 Feb 13  2013 books.txt
drwxr-xr-x  2 shirel users  4096 Feb 13  2013 Central
```

במקרה זה אנו רואים כי user(owner) זה shirel
ה-group זה users
גודל הקבצים רשום בעמודה החמישית.

העברת קלט/פלט ב-bash

לכל תוכנית שרצה ב-UNIX נתונים מראש שלושה ערוצי קלט/פלט:

- ערוץ **הקלט הסטנדרטי** (ערוץ מס' 0) מחובר למקלדת.
  ערוץ זה מאופיין ע"י stdin ב-C (למשל הפקודה scanf קוראת את הנתונים מערוץ זה).
- ערוץ **הפלט הסטנדרטי** (ערוץ מס' 1) מחובר אל המסך.
  הפקודה printf ב-C מדפיסה אל ערוץ זה.
- הערוץ של **הודעות השגיאה** (ערוץ מס' 2) מחובר אל המסך.
  ערוץ זה מאופיין ע"י stderr ב-C. למשל: fprintf(stderr, "warning\n");

ב-bash ניתן ***לכוון מחדש*** כל אחד מהערוצים הנ"ל, כך שהמידע ייקלט מהקובץ או ייפלט לקובץ או הודעות השגיאה ייפלטו לקובץ.
העברה מחדש של ערוץ נקראת: **redirection**.

העברת קלט מהקובץ: **program < filename**, למשל:

```
lion:eesoft> mail ilana@ee < letter
```

הסבר: תוכנית ה-mail מצפה לקבל את הקלט שלה מהקלט הסטנדרטי, ז"א שהמשתמש יקליד במקלדת את מה שהוא רוצה לשלוח. הפקודה לעיל מעבירה את הקובץ letter כקלט ל- mail. (זו פונקציה שמקבלת כפרמטר את כתובת המייל)

העברת פלט לקובץ: **program > filename**, למשל:

```
lion:eesoft> ls > ex_list
lion:eesoft> cat ex_list
ex1        ex2        ex3
ex1~       exer/
lion:eesoft>
```

הפקודה הראשונה העבירה את הפלט של ls לקובץ ex_list, והשניה הציגה את תוכנו.

הוספת פלט לקובץ (append): **program >> filename**, למשל:

```
lion:eesoft> ls  >> ex_list
lion:eesoft> cat ex_list
ex1        ex2        ex3
ex1~       exer/
ex1        ex2        ex3
ex1~       exer/
lion:eesoft>
```

העברת שגיאות לקובץ: **program 2> filename**

הוספת שגיאות לקובץ: **program 2>> filename**

העברת פלט לקובץ אחד ושגיאות לקובץ אחר:

program > file1 2> file2

העברת פלט ושגיאות לקובץ: **program > filename 2>&1**

הסבר: הסימון הראשון מעביר את הפלט הסטנדרטי לקובץ, והשני אומר לפלט השגיאה לעבור אל הפלט הסטנדרטי (שכבר מועבר לקובץ) הסדר חשוב.
נשים לב כי 2>1 היה מפורש כהעברת stderr לתוך קובץ ששמו "1".

הוספת פלט ושגיאות לקובץ: **program >> filename 2>&1**

קבצים מיוחדים

שני קבצים מיוחדים ביוניקס משמשים כמעט רק לצורך העברת קלט\פלט:

/dev/null

קובץ זה מיועד לכתיבה בלבד. מידע שנכתב אליו לא נשמר בשום מקום, ולא ניתן לקריאה. שימושי כאשר אנחנו מעוניינים "להשתיק" פלט מתוכנה.

program_with_alot_of_uninteresting_output > /deb/null

/dev/zero

קובץ זה מיועד לקריאה בלבד. קובץ שתמיד ניתן לקרוא ממנו אפסים והוא לעולם לא "נגמר". שימושי בעיקר במקרה שהתוכנה דורשת קובץ קלט, ואין לנו קובץ קלט לתת לה.

צינורות (pipes)

שימו לב: פעולת הצינור שונה בתכלית מפעולת ה-redirection. הפקודה:

```
aluf:nb>  program1 > program2
```

תגרום לפלט של program1 לדרוס את הקובץ שמכיל את הקוד של program2!
אך ניתן לראות שפעולת הצינור שקולה לפקודות הבאות:

```
aluf:nb>  program1 > tmpfile
aluf:nb>  program2 < tmpfile
aluf:nb>  rm tmpfile
```

כדי לחסוך את הטיפול בקבצים זמניים, ניתן להשתמש בpipeline המחבר את הפלט של פקודה אחת לקלט של פקודה שניה ישירות.

הפקודה: program1 | program2 גורמת לכך, ש-program2 לוקח כקלט את הפלט של program1: ניתן לשרשר כך מספר תוכניות, כך שכל תוכנית לוקחת כקלט את הפלט של קודמתה ומעבירה את הפלט שלה לבאה אחריה ברשימה.
לדוגמא:

```
aluf:nb>  finger | sort | lpr
```

הפקודה **finger** מראה את ה login name של כל מי שמחובר כרגע למחשב, הפקודה **sort** ממיינת את הקלט שלה ומוציאה אותו כפלט, והפקודה **lpr** שולחת למדפסת את הקלט שלה. ולכן כל השורה, תדפיס למדפסת את רשימת כל האנשים המחוברים כרגע למחשב לפי סדר אלף-בת של ה-login name.

הפקודה: program1 2>&1 | program2: שולחת הן את הפלט הסטנדרטי והן את פלט השגיאות של program1 כקלט של program2, לדוגמא:

```
aluf:nb> make 2>&1 |  more
```

7

פקודות שימושיות ב- Unix

הקדמה: כל גירסה של מערכת הפעלה Unix מספקת אוסף גדול של תוכניות שימושיות מאוד, הפורמט של הפקודות הוא די אחיד בכל הגרסאות של ה- Unix, אך לעתים, עלולים להיות הבדלים בינן – ההבדלים הם בעיקר ברשימת האופציות שהפקודות מקבלות ובצורת הפלט.

סימון: רוב הפקודות יכולות לקבל מספר פרמטרים , שחלקם הוא אופציונלי ולא הכרחי – הם יופיעו בתוך סוגריים מרובעים [..]. פרמטרים שהם הכרחיים יופיעו בין סוגריים משולשים < .. >.

הפרמטרים ב- Unix מופיעים אחרי סימן מינוס "-", וניתן להעביר מספר פרמטרים בו זמנית (אלא אם מצוין אחרת). למשל, אם כתוב

ls [-al]

המשמעות היא כי הפקודה **ls** מקבלת פרמטרים **a** ו-**l**, וניתן להפעיל אותה בכל אחד האופנים הבאים: **ls -la , ls -a, ls -l.**

הפקודות:

להלן סט פקודות שימושיות ב-Unix. מקוצר הזמן והחסרון במקום, לא מוצגות כל האופציות של הפקודות, כמו כן לא ניתנות דוגמאות על כל הדברים הרשומים.

עליכם ללמוד ולבדוק את כל הפקודות ב- t2 ולהבין מה כל אופציה עושה !

מומלץ ללמוד את הפקודות בעזרת התוכנה **man**:

lion:eesoft> **man command**

נותנת הסבר על הפקודה command.

כדי להתייחס לקובץ יש להשתמש בשמו.
בכל זמן קיימת תיקייה המוגדרת כתיקייה נוכחית, למשל:

```
[shirel@t2 ~]$ pwd
/homet2/shirel
```

התו "/" משמש בסימן מפריד להגדרת מסלול במערכת הקבצים.
ניתן להתייחס לקובץ על ידי שם מוחלט או על ידי שם יחסי מהתיקייה הנוכחית:

```
[shirel@t2 ~]$ cat /homet2/shirel/hello_world.txt
Hello World
[shirel@t2 ~]$ cat hello_world.txt
Hello World
```

8

פקודות בסיסיות הקשורות במערכת הקבצים

1. **cd <directory>**: לעבור למדריך אחר במערכת הקבצים.
. – נקודה אחת מייצגת את המדריך הנוכחי.
.. – שתי נקודות מייצגות את מדריך האב. (כלומר המדריך בו נמצא המדריך הנוכחי)

2. **pwd**: מציגה את **שם** המדריך הנוכחי.

3. **mkdir/rmdir <directory>**: יוצר (mk) או מוחק (rm) מדריך.

4. **ls [-altR] [filelist]** : מציגה את תוכן המדריכים או את רשימת הקבצים המתאימים לתיאור של filelist.

ההצגה הרגילה תראה בצורה:

```
lion:eesoft> ls
ex1        ex2        hmw
```

להציג את תוכן המדריך hmw/:

```
lion:eesoft> ls hmw
hmw0       hmw1
```

להציג רק את הקבצים במתחילים ב- ex:

```
lion:eesoft> ls  ex*
ex1        ex2
```

הסבר ל*:
ניתן להשתמש בקיצור * כדי להתייחס למספר קבצים בבת אחת ע"י שימוש בתבניות, למשל:
* מתייחס לכל הקבצים
*.txt יותאם לכל הקבצים ששמם מסתיים ב-"txt."

להציג את רשימת הקבצים עם פרטים מלאים:

```
lion:eesoft> ls  -l
-rw-r--r--  1 eesoft   90112    Mar  4    1999    ex1
-rw-r--r--  1 eesoft   714      Apr  11   1999    ex2
drwx------  2 eesoft   512      May 11    1999    hmw
```

אופציות חשובות אחרות של ls:

a מציגה גם את הקבצים שמתחילים ב "." (נקודה). (פריט נסתר ב-UNIX הוא כל קובץ או מדריך ששמם מתחיל בנקודה)

t מציגה רשימת הקבצים ממוינת לפי סדר כרונולוגי (לפי זמנים)

R מציגה את כל הקבצים במדריך ובתוך התת-מדריכים (רקורסיבית)

l מדפיס פלט כך שכל רשומה מופיעה בשורת פלט נפרדת עם מידע נוסף

5. **chmod <modes> <files>:** משנה את הרשאות הגישה של קובץ. הרשאות הגישה הן מהצורה: who-operator-permission, לדוגמא: u+w – מוסיף הרשאת כתיבה למשתמש.

```
lion:eesoft> ls  -ld  hmw
drwx------ 2eesoft      512          May 11      1999 hmw
lion:eesoft> chmod o+x hmw
lion:eesoft> ls  -ld  hmw
drwx-----x 2 eesoft     512          May 11      1999 hmw
lion:eesoft> chmod o+r,g+r hmw
lion:eesoft> ls  -ld  hmw
drwxr--r-x 2 eesoft     512          May 11      1999 hmw
```

6. **find <start-dir> [options]:** פקודה לחיפוש בתוך מערכת הקבצים. החיפוש יכול להיות לפי שם, סוג, תאריך וכו'. (חייב לציין תיקייה שהחיפוש יתחיל בה)

7. **cp <file1> <file2>:** יוצר עותק של <file1> בשם <file2>.

8. **cp <files> <dir>:** יוצר עותקים של הקבצים <files> בתוך <dir>.

9. **mv <file1> <file2>:** מחליף את השם של <file1> ל- <file2>.

10. **mv <files> <dir>:** מעביר את הקבצים <files> לתוך <dir>.

11. **rm <files>:** מוחק את הקבצים <files>.

הערות:

- לפקודות cp, mv, rm ניתן להוסיף את הפרמטרים הבאים:
  - i- – בקש אישור לפני מחיקת (או דריכה על) קובץ קיים.
  - f – force- – אל תשאל שאלות במקרה דריכה/בעיית הרשאות.
  - r- – העתקה/מחיקה רקורסיבית של מדריכים.
- ב-UNIX אין אפשרות לשחזר קבצים שנמחקו.

פקודות בסיסיות העוסקות בתוכן של קבצים

12. **cat filename1 [filename2] …** מדפיסה למסך את כל הקבצים filename1,filename2… אחד אחרי השני.

13. **less [filename]:** מציגה על המסך את הקלט שלה או filename מסך-מסך.
מאפשרת גלילה בשני הכיוונים, ואין צורך לקרוא את כל הinput לפני התחלת הפעולה. למשל:

```
lion:eesoft> man less | less
```

14. **wc [-c | -l | -w] [filename1] ....:** ללא אופציות, מדפיסה את מספר השורות, מספר המלים ומספר האותיות בקלט שלה או בקבצים filename1, filename2, לדוגמא:

```
aluf:nb> wc  BSTmain.c    .login
    84     258    2165 BSTTmain.c
     2       9     131  .login
    86     267    2296 total
```

האופציות:

c: מדפיסה את מספר התוים בלבד.

l: מדפיסה את מספר השורות בלבד.

w: מדפיסה את מספר המילים בלבד.

15. **sort [options] [files]:** ממיינת את הקלט שלה או הקבצים [files], בדרך כלל בסדר א"ב. יש לקרוא man sort כדי לקבל את כל הפרמטרים האפשריים ב-[options].
אם הפקודה מופעלת על מספר קבצים מודפסת גם שורת סיכום.
דוגמא:
נראה תחילה איזה קבצים יש לנו במדריך הנוכחי:

```
aluf:nb>  ls
data        ex1          ex1~
ex2
```

נספור את מספר השורות בכל קובץ (ע"י wc –l), ונמיין אותם (ע"י sort):

```
aluf:nb>  wc –l * | sort –n –r
   47 total
   27 ex1~
   16 ex1
    4 data
    0 ex2
aluf:nb>
```

האופציה n- מסדרת את הקלט לפי הערך המספרי (למשל 4<27), בלי ה –n הסידור היה נעשה לפי ערך ה- ascii ואז "4"<"27".
והאופציה r- הופכת את סדר ההדפסה: מהגדול אל הקטן.

האופציה k F- ממיינת לפי השדה ה- Fיי בכל שורה, דוגמא:

```
aluf:nb> cat bank
Nafea      100   60
Mustafa    70    100
Ilana      200   700
aluf:nb> sort –n –k 2 bank
Mustafa    70    100
Nafea      100   60
Ilana      200   700
aluf:nb> sort –r –n –k 3 bank
Ilana      200   700
Mustafa    70    100
Nafea      100    60
```

בפקודה האחרונה, המיון נעשה לפי השדה השלישי בכל שורה, לפי ערך מספרי ובסדר יורד.

16. **grep [options] expression [files]** : מחפשת בקבצים [files] או בקלט הסטנדרטי שורות בהן מופיעה המילה word ומוציאה אותם לפלט.

הערה: אם רוצים לחפש שורות בהן מופיע ביטוי המורכב מכמה מילים, שמים גרשיים, למשל grep "It is" file. יש לקרוא man grep כדי לקבל את כל הפרמטרים האפשריים ב-[options].

דוגמא 1:

```
aluf:nb> cat file
Habibi, tomorrow we have an exam!
Yalla, bye!
aluf:nb> cat file | grep bye
Yalla, bye!
```

מדפיסה את השורות בהן מופיעה מלה bye.

דוגמא 2:

האופציה n– grep, מוסיפה לכל שורה שהיא מדפיסה את שם הקובץ ומספר השורה בקובץ.

האופציה v– expression grep, מדפיסה את כל השורות שלא מכילות את expression:

```
aluf:nb> cat file1
Ahi!
Have you heard news?
If not, listen!
```

```
aluf:nb> cat file2
Tomorrow we have a new exam!
So we cannot go to Ron's birthday.
By.
aluf:nb> grep –v –n new * |  head –3
file1:1:Ahi!
file1:3:If not, listen!
file2:2:So we cannot go to Ron's birthday.
aluf:nb>
```

הסבר: הודפסו רק שלוש שורות בגלל ה head –3 , השורות ממוספרות לפי שם קובץ ומספר שורה בקובץ בגלל ה grep –n , והשורה file1:2 לא הודפסה היות והיא מכילה new בתוכה.

דוגמא 3:

```
aluf:nb> cat file1 file2
Ahi!
Have you heard news?
If not, listen!
Tomorrow we have a new exam!
So we cannot go to Ron's birthday.
Bye.
 aluf:nb> cat file2 file1 | grep new  > news
 aluf:nb> cat news
Have you heard news?
Tomorrow we have a new exam!
aluf:nb>
```

דגלים נוספים:

i מתעלם מהבדלי uppercase/lowercase

w מדפיס את כל השורות בהן <expression> מופיע בדיוק (לא כתת מחרוזת)

l הדפס רק את שמות הקבצים בהן נמצאו שורות מתאימות.

c הדפס רק את כמות השורות שנמצאו בכל קובץ ללא הדפסת השורות עצמן.

grep מקבל regular expression.

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

17. **head [-n] [filename]:** מדפיס n שורות (או 10) מראשית הקובץ filename או מהקלט שלה, דוגמא:

```
aluf:nb> head –1 jargon
Habibi
```

הפקודה מדפיסה את השורה הראשונה בקובץ jargon.

18. **tail [-n | -f] [filename]:** דומה ל head, אך מסוף הקובץ.

```
aluf:nb> ls –l | tail -2
-rw-r--r-- 1 nafea    3789    Mar  4    1998    ex1~
drwx----- 2 nafea    512     May 11    1998 exer/
aluf:nb>
```

הפקודה מדפיסה את שתי השורות האחרונות בפלט של ls.

האופציה f- מציגה את סוף הקובץ, ולא יוצאת מהתכנית !! ואז כל הוספה לקובץ (למשל ע"י תהליך רקע) תראה על המסך. כדי לצאת מ tail –f יש להשתמש ב CTRL-C.

19. **diff [file1] [file2]**: משווה את התוכן של שני קבצים ומדפיסה את ההבדלים ביניהם. אם לא הודפס דבר, הקבצים זהים (תוכלו להשתמש בתוכנית זו כדי לוודא שהפלט של התוכניות שלכם זהה לדוגמאות פלט שאנחנו ניתן לפני שתגישו שיעורי בית).

20. **uniq [options] [file1 [file2]**: מורידה מהקובץ הממוין file1 את כל השורות הזהות הסמוכות, ושולחת עותק יחיד של כל שורה לקובץ file2 או לפלט הסטנדרטי. יש לקרוא man uniq כדי לקבל את כל הפרמטרים האפשריים ב- [options].

21. **cut –c list [filename]...:** מקבלת את הקלט שלה מקובץ או מהקלט הסטנדרטי, ומדפיסה מכל שורה אך ורק את התווים המופיעים ב list, ה- list יכולה להראות:

5-10: התווים 5 עד 10.

-7,8,14,34: התווים 7 ו- 8 ו- 14 וכל התווים מ- 34 ואילך.

דוגמא:

```
aluf:nb> cat computers
The computer name is aluf.technion.ac.il
The computer name is t2.technion.ac.il
The computer name is tx.technion.ac.il
aluf:nb> cut –c5-13,22- computers
```

computer aluf.technion.ac.il
computer t2.technion.ac.il
computer tx.technion.ac.il

22. **cut –f list [-d] [filename…]:** מקבלת את הקלט שלה מקובץ או מהקלט הסטנדרטי, ומדפיסה מכל שורה אך ורק את השדות המופיעים ב list. שדה הוא רצף תווים המופרד ע"י tab. ניתן להגדיר מפריד (delimiter) חדש יחיד בין השדות ע"י –d. למשל:

```
aluf:nb> cut –f2,5  –d" "  computers
computer aluf.technion.ac.il
computer t2.technion.ac.il
computer tx.technion.ac.il
```

אם נרצה להפריד בין השדות ע"י נקודה ולהדפיס את השדה שני בלבד:

```
aluf:nb> cut –f2 –d.  computers
technion
technion
technion
```

סיכום:
ניתן להשתמש במגוון תכניות סטנדרטיות ב-Unix הפועלות משורת הפקודה ומבצעות פעולות בסיסיות.
ניתן להדפיס את תחילת או סוף הקלט ע"י head ו-tail.
ניתן למיין את הקלט לפי שורות בעזרת sort.
ניתן להשמיט שורות זהות בעזרת uniq.
ניתן לחפש שורות מסוימות בקבצים בעזרת grep.
ניתן להפריד עמודות ספציפיות מהקלט בעזרת cut.
ניתן לספור את מספר התווים המילים והשורות בקלט בעזרת wc.

הרצת ובקרת תהליכים ב-UNIX

UNIX היא מערכת הפעלה היכולה לשרת מספר רב של תהליכים (תוכניות) בו-זמנית.

יצירת תהליכים

ישנן רק 2 צורות להריץ תהליכים:

- **בחזית** (foreground)
- **ברקע** (background)

הרצה בחזית

הרצת תוכנית בחזית מתבצעת פשוט ע"י כתיבת פקודה:

<command>

פקודה המורצת בחזית גורמת ל-prompt להיעלם עד לסיום הפקודה ולא ניתן להקליד ולהריץ פקודה נוספת כל עוד הפקודה הנוכחית לא הסתיימה. ולכן **בחזית יכולה להתבצע פקודה אחת לכל היותר.**

דוגמא: קומפילציה של תוכנית בעלת קובץ אחד-

```
lion:eesoft> gcc –o myprog main.c
```

הרצה ברקע

הרצת תוכנית ברקע מתבצעת ע"י כתיבת פקודה והוספת & בסוף:

<command> &

לדוגמא, הפקודה הבאה מורצת בחזית:

```
lion:eesoft> find /usr –name 'tcsh'
```

לעומת זאת, הפקודה הבאה מורצת ברקע:

```
lion:eesoft> find /usr –name 'tcsh' &
lion:eesoft>
```

ואנו עשויים לקבל את ה prompt , **לפני** סיום התהליך.
על מנת לבצע מספר תוכניות בו-זמנית, יש להריצן ברקע. לאחר תחילת ביצוע פקודה ברקע, מופיע ה-prompt חזרה מיד, וניתן לתת למערכת פקודות נוספות שתתבצענה במקביל לפקודה הראשונה.
הרצה ברקע שימושית לתוכניות המתבצעות לאורך זמן רב שאינן מדפיסות כלום לערוץ הפלט, אינן קולטות כלום מערוץ הקלט או לתוכניות גרפיות!

חשבו כיצד בכל זאת ניתן להריץ ברקע תוכניות אשר עושות שימוש בערוצי קלט/פלט. (רמז: redirection).

ניתן לבצע העברת קלט מתוך קובץ גם לתהליך שנמצא ברקע למשל באופן הבא:

```
program < input_file &
```

מעקב אחרי תהליכים

הצגת תהליכים

בעזרת הפקודה **jobs** ניתן לבדוק אילו תהליכים קיימים כרגע ומה מצבם:

```
lion:eesoft> jobs
[1] + Running     cc prog.c
[2] – Running     dvips –o doc.ps doc.tex
```

דוגמא נוספת: נניח שיש לנו תוכנית שמדפיסה שורה ורצה לעד

```
#include <stdio.h>
void main( ) {
        printf("I am going to run forever\n");
        while(1)
                ;
}
```

נריץ אותה ברקע:

```
aluf:nb> forever &
[1] 587
```

אנו נקבל את prompt בחזרה, ובמקרה, התוכנית תדפיס את שורתה:

```
aluf:nb> I am going to run forever
aluf:nb>
aluf:nb> jobs
 [1] + Running     forever
```

לכל תהליך שמריצים מוצמד מספר , כמו כן, ישנם מספר דרכים לציין תהליך מסוים:

- ***%n*** – תהליך מספר n
- ***%s*** – תהליך אשר שורת פקודתו מתחילה במחרוזת s
- ***%?s*** – תהליך אשר שורת פקודתו מכילה מחרוזת s
- ***%%*** – התהליך הנוכחי
- ***%-*** – התהליך הקודם

לדוגמא, כל המחרוזות הבאות מתייחסות לאותו תהליך מהדוגמא הקודמת:

- ***%1***

- ***%for***
- ***%?ever***

העברת תהליך לחזית

הפקודה: fg [%jobIds] מעבירה את התהליך האחרון שהורץ ברקע או [jobIds] לחזית, למשל:

```
aluf:nb> cat try_input.c
main() {
   char str[20];
   printf("I shall try to eat something");
   scanf("%s", str);
   printf("%s", str);
}
aluf:nb> try_input &
[1] 828
aluf:nb> I shall try to eat something
[1] + Suspended (tty input)   try_input
aluf:nb> jobs
[1] + Suspended (tty input)   try_input
aluf:nb> fg %1
try_input
goodies
goodies
aluf:nb>
```

העברת תהליך לרקע

הפקודה: **bg [%jobIds]** מעבירה את התהליך האחרון שהושעה או [jobIds] לרקע, למשל:

```
aluf:nb> jobs
[1] + Suspended        forever
aluf:nb> bg %1
[1]   forever &
aluf:nb> jobs
[1] + Running          forever
```

השעיית תהליך

Ctrl-Z משעה את התהליך המתבצע כרגע בחזית. תהליך מושעה ניתן להעביר לחזית או לרקע ע"י שימוש בפקודות fg או bg.

הפקודה: **kill –s SIGTSTP [%jobIds]** משהה את התהליך האחרון שהורץ ברקע או [jobIds], לדוגמא:

```
aluf:nb> emacs &
[1] 982
aluf:nb> forever &
[2] 991
aluf:nb> jobs
[1] + Running          emacs
[2] – Running          forever
aluf:nb> kill -s SIGTSTP %2
 [2] – Suspended (signal)  forever
aluf:nb> jobs
[1] - Running          emacs
[2] + Suspended (signal)  forever
aluf:nb>
```

הריגת תהליכים

Ctrl-C הורג את התהליך המתבצע כרגע בחזית.

הפקודה: **kill –9 %jobId** הורגת את התהליך jobId.

דוגמא פשוטה להרצת תהליכים:

```
lion:eesoft> netscape&
[1] 585
```

(ה-prompt חוזר ונפתח חלון גרפי של netscape)

```
lion:eesoft> jobs
[1]  + Running              netscape
lion:eesoft> kill –9 %1
lion:eesoft>
[1]    Killed               netscape
```