סימון: בתרגול זה וביתר התרגולים עד סוף הסמסטר, יינתנו דוגמאות לעבודה בסביבת c-shell תחת UNIX. אנו נצמד לסימון הבא: פקודות שהמשתמש צריך לרשום מופיעות **באותיות מודגשות (bold)**, ואילו הפלט של ה c-shell/UNIX יופיע *באותיות לטיניות*.

בעבודה אינטראקטיבית מול c-shell, ה c-shell נותנת **prompt** בכל במקום היא מוכנה לקבלת פקודה, במשך התרגול נשתמש ב **prompt** הבא:

aluf:nb>

ה- **prompt** יכול להשתנות מחשבון לחשבון, ולכן בחשבון שלכם ב- t2 אתם עשויים לראות **prompt** שונה.

1 העברת קלט/פלט ב-C-Shell

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

- ערוץ **הקלט הסטנדרטי** (ערוץ מס׳ 0) מחובר למקלדת
- ערוץ **הפלט הסטנדרטי** (ערוץ מס׳ 1) מחובר אל המסך
- הערוץ של **הודעות השגיאה** (ערוץ מס׳ 2) מחובר אל המסך

ב-C-Shell 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:

aluf:nb> **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.

העברת פלט לקובץ: program > filename, למשל:

```
aluf:nb> ls ex* > ex_list
aluf:nb> cat ex_list
ex1         ex2         ex3
ex1~                exer/
aluf:nb>
```

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

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

```
aluf:nb> ls ex* >> ex_list
aluf:nb> cat ex_list
ex1         ex2         ex3
ex1~                exer/
ex1         ex2         ex3
ex1~                exer/
aluf:nb>
```

העברת פלט ושגיאות לקובץ: program >& filename

הוספת פלט ושגיאות לקובץ: program >>& filename

העברת פלט לקובץ אחד ושגיאות לקובץ אחר: (program > file1) >& file2
למשל:

```
aluf:nb> (make > out) >& err
```

את הפלט של make נראה ע"י:

```
aluf:nb> more out
gcc try_input.c –o try_input
*** Error code 1
```

והודעות השגיאה נראה ע"י:

```
aluf:nb> more err
try_input.c: In function ‘main’:
try_input.c:4: parse error before ‘printf’
make: Fatal error: Command failed for target ‘try_input’
aluf:nb>
```

2 צינורות (pipes)

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 aluf, 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 |& program2: שולחת הן את הפלט הסטנדרטי והן את פלט השגיאות של program1 כקלט של program2, לדוגמא:

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

3 פקודות שימושיות ב Unix

הקדמה: כל מערכת 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.

אנו נעבוד בקורס לפי מערכת ה- Unix שנמצאת במחשב ה- t2 הנקראת solaris ver. 2.6 שהיא תוצר של חברת Sun Microsystems. הגרסאות אחרות שנפוצות של מערכות unix הן: Linux , BSD, Digital unix, AIX.

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

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**.

1. **man:**

```
aluf:nb> man command
```

נותנת הסבר על הפקודה command.

2. **more [filename]:** מציגה על המסך את הקלט שלה או filename מסך-מסך.

3. **ls [-altR] [filelist]** : מציגה את תוכן המדריך הנוכחי או את רשימת הקבצים המתאימים לתיאור של filelist.

ההצגה הרגילה תראה בצורה:

```
aluf:nb> ls
ex1         ex2         ex3
ex1~                exer/       mail/
netscape/ hmw/        classes/
aluf:nb>
```

להציג את תוכן המדריך hmw/:

```
aluf:nb> ls hmw
hmw0/     hmw1/     hmw2/
hmw3/     readme    submission-instructions.pdf
aluf:nb>
```

להציג רק את הקבצים במתחילים ב- ex:

```
aluf:nb> ls  ex*
ex1         ex2         ex3
ex1~                exer/
aluf:nb>
```

להציג את רשימת הקבצים עם פרטים מלאים:

```
aluf:nb> ls  -l  ex*
-rw-r--r--      1 nafea   90112     Mar 4     1998      ex1
-rw-r--r--      1 nafea   714       Apr  11   1998      ex2
-rw-r--r--      1 nafea   298316    Jun  17   1998      ex3
-rw-r--r--      1 nafea   3789      Mar  4    1998      ex1~
drwx-----2 nafea   512         May 11    1998      exer/
aluf:nb>
```

אופציות חשובות אחרות של ls:

-a מציגה גם את הקבצים שמתחילים ב ".." (נקודה).

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

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

4. **tee [options] [files]** : משכפלת את כל מה שנכנס לקלט הסטנדרטי שלה ושולחת אותו הן לפלט הסטנדרטי והן לכל אחד מהקבצים ב-[files], למשל:

```
aluf:nb> ls | tee file_list list
```

האופציה –a מאפשרת שרשור (appending) של הפלט לסוף הקבצים [files] במקום דריסת תוכנם.

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

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

האופציות:

-c : מדפיסה את מספר התוים בלבד.

-l : מדפיסה את מספר השורות בלבד.

-w : מדפיסה את מספר המילים בלבד.

6. **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].

דוגמא:

נראה תחילה איזה קבצים יש לנו במדריך הנוכחי:

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

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 "27"<"4".

והאופציה r– הופכת את סדר ההדפסה: מהגדול אל הקטן.

האופציה k F– ממיינת לפי השדה ה- F"י בכל שורה, דוגמא:

```
aluf:nb> more 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.

7. **spell [options] [files]** : משווה מילים מהקבצים [files] למילים במילון של המערכת (/usr/dict/words) ומדפיסה על המסך כל המילים שלא נמצאו במילון.

האופציה **wordlist+** מורה להשתמש **בקובץ הממוין** wordlist כבמילון נוסף של המשתמש,

למשל:

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

נראה אילו מלים המערכת לא מכירה, נמיין אותם ונשמור אותם בקובץ:

```
aluf:nb> spell file | sort > jargon
aluf:nb> more jargon
Habibi
Yalla
aluf:nb> spell +jargon file
aluf:nb>
```

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

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

9. **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.

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

11. **grep [options] word [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 grep –n, מוסיפה לכל שורה שהיא מדפיסה את שם הקובץ ומספר השורה בקובץ.

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

```
aluf:nb> more file1
Ahi!
Have you heard news?
If not, listen!
aluf:nb> more 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> more news
Have you heard news?
Tomorrow we have a new exam!
aluf:nb>
```

12. **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].

13. **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:

10-5: 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
```

14. **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 רווח או tab.

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

ניתן להגדיר מפריד (delimiter) חדש בין השדות ע"י d–, למשל אם ארצה להפריד בין השדות ע"י נקודה ולהדפיס את השדה שני בלבד:

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

4 בקרת תהליכים ב-UNIX

4.1 Unix כמערכת הפעלה:

UNIX היא מערכת הפעלה, עם המאפיינים הבאים:

1. רבת תהליכים (Multi-Processes): יכולה לשרת מספר רב של תהליכים בו-זמנית.
2. רבת משתמשים (Multi-Users): יכולה לעבוד מול כמה משתמשים בו-זמנית, (וכל משתמש יכול להריץ כמה תהליכים בו זמנית): למשל, כאשר כולכם עבדתם על מחשב ה t2 , עבדתם עם מערכת הפעלה Unix שרצה על המחשב t2 בסביבה רבת משתמשים.
3. רבת מעבדים(Multi-Processors): רצה מעל מספר מעבדים(CPUs) באותו מחשב, בו זמנית, ובצורה שקופה למשתמש: סביר להניח שלא הבחנתם בעובדה זו כשהרצתם על ה t2, למרות שה t2 מכיל בעצם יותר ממעבד אחד !!

בפרק זה נלמד כיצד משתמש יכול לייצר מספר תהליכים ולשלוט על אופן ביצועם.

4.2 יצירת תהליכים

ב-Unix ישנן רק 2 צורות להריץ תהליכים:

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

4.2.1 הרצה בחזית

הרצת תוכנית בחזית מתבצעת פשוט ע"י כתיבת פקודה:

<command>

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

דוגמא: כאשר הרצתם את הפקודה pine לקריאת דואר אלקטרוני, או קראתם ל gcc –c BSTmain.c וכו', בכל הפקודות הנ"ל הרצתם תהליך וחיכיתם עד סיומו, לפני הרצת תהליך אחר.

4.2.2 הרצה ברקע

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

```
<command> &
```

לדוגמא, הפקודה הבאה מורצת בחזית:

```
aluf:nb> gcc Calc.c –o Calc
```

לעומת זאת, הפקודה הבאה מורצת ברקע:

```
aluf:nb> gcc Calc.c –o Calc &
aluf:nb>
```

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.

4.3 מעקב אחרי תהליכים

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

```
aluf:nb> 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
aluf:nb>
```

4.4 פקודות לניהול תהליכים

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

4.4.1 העברת תהליך לחזית

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

```
aluf:nb> more 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>
```

4.4.2 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
```

4.4.3 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: **stop [%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> stop %2
 [2] – Suspended (signal)  forever
aluf:nb> jobs
[1] - Running           emacs
[2] + Suspended (signal)  forever
aluf:nb>
```

4.4.4 הריגת תהליכים

Ctrl-C הורג את התהליך המתבצע כרגע בחזית

הפקודה: **kill –9 %jobId** הורגת את התהליך jobId, למשל:

```
aluf:nb> forever &
[1] 587
aluf:nb> I am going to run forever
aluf:nb> kill –9 %fo
aluf:nb> jobs
aluf:nb>
```