

## לינוקס - מבוא ומושגים בסיסיים

### ראשי פרקיים

- א. מהו לינוקס.
- ב. כיצד בנויה לינוקס
- ג. GNU
- ד. ההיסטוריה.
- ה. יתרונות לינוקס
- ו. הפצצות לינוקס

**א. מהו לינוקס?**  
 לינוקס היא מערכת הפעלה "חופשית" בעלת קוד פתוח הבנוי על עקרונות מערכת הפעלה יוניקס. היא נוצרה במקור על ידי לינוס טורבלדס בשנת 1991, לינוס, שהוא אז סטודנט פיני ממוצע, שיחק עם מכונה המרצה סוג של יוניקס המכונה Axmin.

لينوس החליט לשכתב את קוד המקור, ולכתוב קרNEL (ליבה) משלו של מערכת שתיה דמויית יוניקס. הוא יצר את הליבה, ושיחרר אותה לעולם דרך האינטרנט, כשהוא פירסם את הליבה, הוא לא חשב על כך שהמערכת תהפוך למאה שהיא הפכה, הוא ראה את זה כחוביותו לא. אנשים התחלו לסייע לו מכל רוחבי העולם, ואת המערכת קיבלה צורה, ויחד עם חלקים נוספים כמו כל'ה השׁח' המובילים, המערכת הפכה למערכת שלמה (כבר לא רק ליבה, אלא גם "מעטפות", מערכת גרפית נוספת בהמשך, קומפיילרים וכו').  
 ביום המערכת היא מערכת שלמה ומלאה, שיכולה לעשות כל מה שמערכת הפעלה אחרת מסוגלת לעשות, ובאזור עיליה, זולה ומעל לכל, וכמובן כשהמערכת מלאת את העיקון של "תוכנה חופשית".

### ב. מבנה מערכת הפעלה UNIX - לינוקס

UNIX היא מערכת הפעלה הבנוייה על פי מודל השכבות הבא

תוכניות שירות - UTILITIES
מעטפת - SHELL
ליבה - KERNEL
חומרה - HARDWARE

חומרה – שכבה זו מכילה את כל רכיבי המערכת האלקטרוניים כגון : מסך, דיסק קשיח, מעבד, זיכרון וכדומה.

### ליבה-גרעין - KERNEL

קרNEL היא ליבת המערכת – החלק המרכזי במערכת הפעלה.

הקרNEL מבצע את העבודה מול החומרה, אחראי על ניהול הזיכרון גישה לדיסק קשיח לאמצעי הקלט והפלט וכדומה.

בשכבה זו מתורגםות פניות המשמשות לרוטיניות שידועות לעבוד מול התקן מסוים.

הקרNEL הוא החלק העיקרי והוא זה שאחראי על ביצועו, מערכת הפעלה ועל יציבותה.

בשל החשיבות הגבוהה של הkernel ליצועי המערכת, הkernel של Linux, בניו ברובו ב C (שנחשבת לשפה מאוד יעילה בבחינת קוד) [ובכמה חלקים קרייטיים יותר ב"אסמבלי"].

הkernel של Linux, זמן להורדה כקוד מקור בחינם, ואלפי מתכנתים ברחבי העולם שוקדים כל העת לשכל ולשפר אותו.

## **SHELL**

שכבה זו ידידותית למשתמש . שכבה זו משתמשת מ��יש פקודה , המעטפת מתרגמת אותה ומספקת את תוצאת הפקודה להתקן כלשהו מערכת הפעלה יודעת לעבור שכבה לשכבה על פי הבקשות השונות.

דוגמה קיימת פקודה המאפשרת לראות רשימת קבצים. המעטפת מתרגמת את תחביר הפקודה לרשימת הקבצים אותה אנו מעוניינים לראות , עוברת לשכבת הליבה שם יודעת המערכת לפנות להתקן חומרה במקורה זה הכוון ולהציג את תוצאת הפקודה במקורה זה רשימת הקבצים . קיימות מערכות הפעלה המספקות מעטפת אחת לעומת זאת יוניקס מאפשרת מספר מעטפות שונות . כמו: bash , C shell.

## **תפקידו סביבת העבודה**

- שימוש אינטראקטיבי'
- התאמת אישית של העבודה בLinux
- תכנות ( סדרה של פקודות עצמאיות המוגדרות לתוכנית אחת נקראת shell script )

## **תוכניות שירות – UTILITIES**

תוכניות הבוססות על מערכת הפעלה ומספקות שירותים נוספים

## **ג. GNU**

המונח ר'ת של **X** is Not **U**nix **i**s **G**nu. זהו ארגון שנקם לו אי שם בשנות ה 80 ושם לעצמו כמטרה לפתח מערכת שתיה דומה ליוניקס, אבל תיתן למשתמש את החירות לשנות את המערכת, לגשת לקוד המקור שלה וכו'... אנשי ארגון זה יצרו באותו ימים כל מה שציריך כדי שתיהה מערכת הפעלה, קומפュילרים, מעטפות וכדומה, רק דבר אחד היה חסר - kernel.

וכאן נפגשו ה **gnu** ו**Linux**: בשנת 1991 לינוס טורבלדו כתב את kernel של Linux, אבל הוא כתב רק kernel. שילוב של הכלים של ה **gnu** יחד עם kernel של Linux, מוביל למערכת הפעלה אחת, שיש מי שמתעקש שהיא צריכה להיקרא **GNU/Linux** ולא **axuchin** בלבד.

רוב הפיציות (מוסבר בהמשך) מבוססות על הגנו. GPL - ה GPL הוא למעשה מסמך של רישיון פתוח, שנכתב על ידי אנשי ה **gnu** ונועד לתת מענה לצרכים השונים: מצד אחד החופש למשתמש, מצד שני, לא לפגוע בזכויותיו של היוצר המקורי.

## **ד. ההיסטוריה.**

yonikos היא מערכת הפעלה שפותחה בשנת 1969 במעבדות בל, בשנת 1974 היא נכתבת מחדש מחדש כולה בשפת ס', מה שהפך אותה למערכת הפעלה הראשונה שנכתבה בשפה סטנדרטית. עם הזמן יוניקוס הגיעו לעולם מספר תנאים ותוכנות שתרמו לרבות לעולם המחשבים כיום וחברות רבות מצאו לה שימוש. יוניקוס היא מערכת הפעלה חזקה יותר מוינדוס וSIMOSית יותר במחשבים גדולים ובסבירות עבודה רבות משתמשים.

עלקב העובדת שיווקס הייתה בעלות אחת מחברות המחשבים המובילות וגם בגין שהוא נכתב בשפה סטנדרטית ושילבה המון רענוןת חדשים שנחפכו לנפוצים עם הזמן, היא היפה לסטנדרט פתוח במערכות הפעלה, סטנדרט שככל אחד יכול לאם, סטנדרט שנקרא XPOSIX, שפירושו הוא Portable Operating System Interface אוXIPOS. בשביל להתחי לסטנדרט זה, מערכת הפעלה צריכה להתאים למספר דרישות. חברות כמו סאן, סיליקון גרפיקס ואאי.בי. אם פיתחו מערכת יוניקס משליהם שתואמת לתקן XPOSIX. [הרחבה במצגת]

לינוקס היא מערכת הפעלה חופשית ופתוחה שפותחה על ידי מפתח בשם לינוס, שייחרר את קוד המקור שלה, ובעזרת האקרים מכל העולם, המערכת התפתחה ועדין מפותחת ונחשבת למערכת מודרנית, לינוקס מכונת אל תakan ה XPOSIX.

וכוללת בתוכה את כל המאפיינים שנכפה לראותה במערכת יוניקס מודרנית.

לינוקס בסיסה פותחה בשביב מחשי 386 אך כיום היא כבר מסוגלת לרוץ כמעט על כל פלטפורמה. יתרונה הגדול של לינוקס על מערכות היוניקס למיניהן היא העלות הנמוכה מאוד (אפשרות בחילק מהמרקרים) והעובדת שלינוקס היא מערכת של הקהילה ובשביל הקהילה ובשל כך היא מערכת חופשית ופתוחה וברוב המקרים גם מסוגלת לתת ביצועים שוים או טובים יותר מערכות יוניקס למיניהן.

## ה. יתרונות מערכת הפעלה

### מלונות

- א. גמישות.
- ב. מחיר – חינם לגמרי או אפסי יחסית למערכות הפעלה האחרות ( תלוי בגרסת ההפצה)
- ג. מערכת מרובה משתמשים אמיתית
- ד. מערכת חזקה הנופלת פעמים מועטות בלבד
- ה. מערכת המכירה את כל המערכות האחרות
- ו. קוד פתוח - מערכת פותחה כתובה ב C ומאפשרת התאמת אישית והוספה כמעט אין סוף התקני חומרה

### חרונות

- א. כושר דיהוי אמצעי חומרה מוגבל
- ב. קשה ללימוד ולהפעלה

### קוד פתוח

קוד פתוח זהו אופן הפצה והעברה של תוכנה, כאשר יש למשתמשים את האפשרות להתבונן בקוד התוכנה עצמה. בקבוצות התוכנות והמערכות בעלות הקוד הפתוח יושבות להן תוכנות עם רישיון BSD ורישיון GPL.

#### רישיון של לינוקס הוא GPL

המנון מתיחס בז"כ למערכות שרישוי התוכנה שלහן הואזכה שימושם למשתמשים לא רק לראות ולחזור את הקוד הממקור, אלא גם לשנות אותו כדי להתאים לצרכיהם, או לפטור באגים שנתקגלו.

שרותי הרשות היפה את מדיהם הקוד הפתוח, משיטה שבה שניים נשלחים בדואר רגיל, ולוקח זמן עד שימושם אל תוך הקוד המוביל (העץ), לשיטה שבה שניים קוררים בין לילה, כאשר אנשים רבים מוטלים חלק בפיתוח ותיקון משותפים בלי קשר למיקום הגיאוגרافي, כך שלמעשה, הקוד פתוח ששורד ומשמר.

חשיבותו של הקוד GPL הוא בכך שהוא "מדקך" כלומר, הקוד שמשתמש בו חייב להיות בקוד פתוח על פי תנאי GPL גם כן (כדי למנוע ניצול לרעה של רישיון הקוד הפתוח).

היות ונושא זה מביא לביעיות מסוימת כמשמעות בספריות פיתוח, שם שלולים את החירות של המשתמש ליצור תוכנית קוד סגור.

## פילוסופיה זו - UNIX

- רבוי משתמשים / רבוי שימושות
- גמישות וחופש
- רבוי כלים קטנים ו שימושיים (ארגון כלים)
- לדבר הוא קובץ
  - גישה לכל התקן חומרה הוא גישה לקובץ בפונקציות סטנדרטיות של גישה לקבצים
- מערכת קבצים איחידה
- פותח ע"י מתכננים לתוכנותם
- לאחרונה קיימת פריחה במשקיים יידידותיים למשתמש הבית.

### 2. הפצת יוניקס

1. מה זו הפצה?  
למעשה "لينוקס" זו רק הליבה (kernel) של מערכת הפעלה. רכיבים רבים דרושים על מנת לפעול מחשב-אישי, לדוגמה, ספריות שונות, יישומים, קבצי אתחול וקבצי תצורה. הפצה עצמה אוספת את כל הנדרש מעבר ללבת המערכת על מנת לצור מערכת הפעלה שלמה ומלאה. וכשمرة היא גם "מפיצה" את האוסף הזה, בין אם על-גבי דיסקים (בד"כ בליעו ספרות ותמייה) ובין אם בעדרת האינטרנט להורדה בצורית קבצי ISO לצריבה עצמית.  
כמו כן לכל הפצה יש את מערכת ההתקנה הייחודית לה.  
כל הפצה פונה לבישה אחרת של משתמש מחשב. ولكن השוני והמגוון הרחב גרסאות הפצה אלו מנוהלות הן על ידי מתנדבים והן על ידי חברות מסחריות המספקות גם תמיכה טכנית, מדריכים למשתמש ומוצרים נוספים. אם כי תמיד ניתן להוריד בהינט ולהיעזר בקבוצות הדין על כל בעיה שמתעורר.

linux-1

70°L

LINUX

*linux-2*

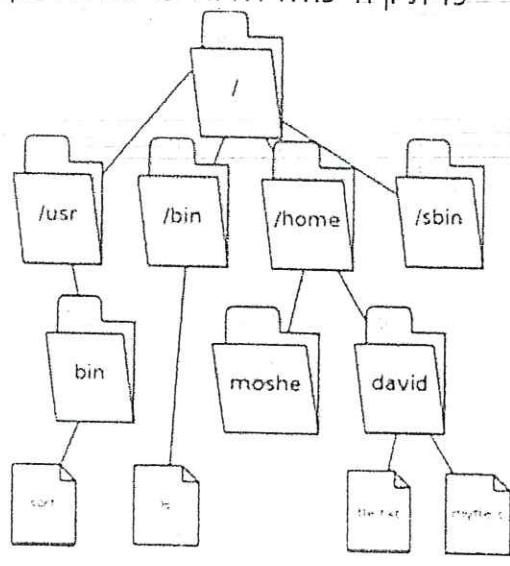
TODAY

## מערכת הקבצים של Unix

היררכיה (דמוי-עץ).

תיקיות וקבצים.

לכל משתמש יש את תת-העץ שלו, שורש העץ הוא "תיקית הבית".  
דמיוון למערכת קבצים אמיתי: ארכיבים, מגירות מלאות, תיקיות, מסמכים.  
כל תיקיה יכולה להיות על מחיצה מקומית /DISK רשת / חיבור חיצוני וכו'.



מערכת הקבצים

סימנים מוסכמים:

- קובץ (FILE)

... ובקבוצה של קבצים יתפצלו הקבצים על ידי סימן סטארט (\*) וסיום (\$) בזווית ימינה (בזווית שמאל בזווית ימינה)

~ פיקוח (LINK)

/ -zeich גזירה

## עוד מידע על קבצים

שמות הקבצים והתיקיות יכולים להכיל אותיות גדולות וקטנות, ספרות ורבים סימני הפיסוק.  
לנקודה בשם הקובץ אין משמעות. נשתמש בנקודה וסימנת כדי לציין שהקובץ נפתח באמצעות  
תוכנה מסוימת (לדוגמא, קובץ C מכיל את הסימנת C. וקובץ TEXT מכיל את הסימנת tex).

## תיקיה הנוכחיות ותיקית בית

התיקיה שבה נמצאים נקראת התיקיה הנוכחיית (current directory) או התיקיה הפעילה (working directory).

כל משתמש יש תיקית בית שהופכת להיות התקינה הפעילה בכל התחברות חדשה. שם המלא של הקובץ הוא פירוט של הנתיב המלא מהשורש אל הקובץ או התקינה. שם זה מתייחס בסימן /,-/ הראשון מייצג את השורש, ואחר המרכיבים מופרדים בעזרת הסימן /.

## כך זה...

אעט היא מערכת תלוית רישיות; כתבו את שם הקובץ בדיק כפוי שהוא מופיע, וזכרו שהקובץ שנקרא 'myfile' שונה מהקובץ שנקרא 'Myfile'.

כל פקודה מזוהה ע"י הקשה על מקש enter

## טיול במערכת הקבצים

הפקודה cd (קיצור ל- Change Directory) משמשת למעבר בין תיקיות. השתמש בפקודה עם שם תיקיה כדי לשנות את התקינה הפעילה לתיקיה הנבחרת.

הפקודה pwd משמשת להציג שמה של התקינה הפעילה.

### למידה ותרגול

צור שתי ספריות בשם mydir ו- friends בעזרת הפקודה mkdir directory .  
 היכנס בספריה mydir וצור קובץ ישם בשם touch myname ע"י הפקודה touch myname  
 עורך את הקובץ שיצרת ב- touch myname ע"י הפקודה pic, ומספר תעודת הוזהות שלך ובמספר הטלבון בבית. שוכר את הבאים, יצחק (פרטי – משפחה), מספר תעודת הוזהות שלך ובמספר הטלבון בבית. שוכר את הקובץ.  
 בעזרת הפקודה ls הציג את רצימת הקבצים המצוינים בספריה הנוכחית.  
 כתעת נציג מידע נוסף על הקובץ שנוצרה שערת קודס, באמצעות הפקודה rm myname -i ls (ls בטעלה = ל- ll)

הציג על הקובץ יופיע במבנה הבא:  
 myghaz grad 1 myghaz.txt 20 oct 24 13:18

באור השדות (מייצgal ליפין): רישאות הקובץ, מספר המצביעים אליו, שם יוצר הקובץ הקבועה אליו (במאתמץ מוגדר ב��יערכת). גודל הקובץ, תאריך ושעה העדכן האחרון שовичח בהובא, ולבסוף - שם הקובץ

בא בספרייה mydir וubah לספרייה הרווחית (שבך) באמצעות הפקודה ~ cd היכנס בספריה friends, וצור בה שליטה הקבצים ישכונותיהם כשות שלושת חברים/chברותיך ע"י במטהה friend1 friend2 friend3

הציג את הקבצים בספריה. טהן הרשות שניתנו לך, כמה תווים הם בוביליסו

פקודת העתקת הקבצים היא cp והמבנה הוא «`cp source target`».  
 כתעת עתיק את הקובץ myname שישרנו בספריה Mydir בספריה הנוכחית (Friends).  
 הפקודה «`cp myname ~ Mydir`», כאשר התו ~ מסמן גוף הפקודה cp בהעתה את הקבצים אותם בילשנו בספריה הנוכחית, בה אנו נמצאים.

אם לא רצח להעתיק קבצים אלא להציג אותן המבנה הוא אותו במבנה כמו של הפקודה cp (העתיק) אבל הפקודה היא עונה (העביר).  
 בעזרת הפקודהcp thou ניתן אף לשנות ישובת קבצים, נניח שיש לנו ( בספריה הנוכחית) קובץ הרעה של תבנית מסוימת אשר יטס קובץ התרעה הוא שם. ואנו רוצים לשנות את שמו ב- hello מהה hello.shcp a.e. אז

נניח שעבדתם בין ניבר בזיניקס וויליאטס וירדתם בז' הקבצים ואינם יודעים באנדרואיד  
 והיכן אתם נמצאים פה או פיאוט טבקדים את הפקודה ps aux (הצדץ/ספריה הנוכחית)  
 שאתם יובדקם בז'

עושים נverb לספריה הראשית באמצעות הפקודה ~ cd, ומעתים את כל הקבצים בספריה Friends לספריה הראשית ע"י הפקודה ./\* Friends בצעו את הפקודה dir , מה היא מבצעת?

בחקו את הספריה Friends באמצעות הפקודה rm -r dir. בדקו שאכן הספריה נמחקה (ז'). מחקו אחד הקבצים שהעתיקתם מהספריה Friends לספריה הראשית ע"י הפקודה rm friend . בדקו שאכן הקובץ נמחק. כדי להחזיר את הקובץ מיש אלא מעבירו אותו לספריה בשם "unrm friend" (מעין של מיחזור). כדי להחזיר את הקובץ ש"מתקן" בוצע את הפקודה rm friend . כדי למחוק את ה- Trashcan מבצעים את הפקודה clean Trashcan

כדי למחוק את הקובץ friend מושך עליו לבצע את הפקודה rm friend

### הרצת פקודות ברצף

ניתן להרצץ שתי פקודות אחת לאחר השניה ע"י הפרדה של " " ביןיהן.

למשל: cd ..; pwd  
(cp game mail; ls mail) cp <source> <target>; ls <target>  
עוד דוגמא -

### הצגת קבצים על המסך

הפקודות cat more ו-less מציגות קבצים על המסך.

הפקודה cat מדפסה קובץ על המסך אך אינה מאפשרת דפדף.

הפקודה less ייעילה לקבצים קטנים.

שימוש פשוט בפקודה <שם קובץ> cat

דוגמא :

- צרו קובץ script myfile בשם myfile :

- בצעו פקודות כלשהן כדי שיוכנסו לקובץ-h script

- סיינו את קובץ-h script ע"י הקלדת exit .

- כדי לבדוק בפועל את הפקודה cat בצויר :

more

הפקודה more מדפסה קובץ על המסך ע"י אפשרותו היא באפשרות דפדף בתחרתיות המסך של הפקודה more מודפס לאחר הקובץ שכבר הוצג על המסך דפדף בקובץ :

space bar דפדף למסך הבא .

6 דפדף יסך לאחר .

Enter דפדף שורה נוספת .

pattern חיפוש מחרוזת (טילים וחלים טיליס)

דוגמא : בוצעו את הפקודה more myfile

less

פקודת less מעצור לאחר הצגת חלון אחד וירפיעו נקודותים (...) בתחתית המסך.

דפוזט בקובץ :

p טיל קדימה.

n טיל קדימה (חצי מסך).

l לכת שורה אחת לאחר.

q less myfile ביצעו את הפקודה

פקודות סיוור בקבצים

פקודות יוניקס מורכבות מ: 1] פרמטרים 2] אופציות 3] שם פקודה

אופציות מסומנות ב "-".

פקודות סיוור בקבצים :

רשימת קבצים בתיקיה.

**ls [file\_name]**

רשימת קבצים עם יותר מידע, הרשות, בעליים, קבוצות וכו'.

**ls -l [file\_name]**

הצגת הקובץ על המסך.

**cat [file-name]**

עריכה קובץ

**pico [file-name]**

הצגת מספר שורות בקובץ.

**head [file-name]**

להציג מספר שורות הראשונות (החלף את X במספר השורות).

**head -X [file-name]**

הצגת השורה האחורית בקובץ.

**tail [file-name]**

להציג מספר שורות אחרונות (החלף את X במספר השורות).

tail -X [file-name]

## התקנים תקניים

לכל תהליך יש:

התקן קלט סטנדרטי -- `stdin`

התקן פלט סטנדרטי -- `stdout`

אלא אם כן נאמר אחרת, התהליך מקבל נתונים מ-`stdin` ושולח נתונים ל-`stdout`.  
בתהליך המופעל ע"י המשמש, מחובר `stdin` למקלדת ו-`stdout` לצג כברירת מחדל.  
קיימים גם התקנים שגיואוט סטנדרטי -- `stderr`

## שימוש בקבצים לקלט או פלט

פונקציית command < file ~  
command > file  
command >> file

## דוגמאות

`ls -l > list`

הפלט של הפקודה יכנס לקובץ בשם `list` וידרשו כל אינפורמציה אחרת אליו הייתה צדאת בקובץ.

`ls -l >> list`

הפלט של הפקודה יכנס לסוף הקובץ בשם `list`

## למידה ותרגול

א.

`file > - ls` כתת החיצון על המסך את הקובץ `file`

ב. `ls >> file` כתת החיצון שוכן על המסך, אך הושק ל-`file`

ג. `ps > file` מה קורחה פאנו ווודה ואט? - 30 שורות כוונתיו  
ה. כתת החיצון על המסך את הקובץ `file`

ד. ערכו קובץ "something" באורך 30 שורות.

ה. הציגו את הקובץ.

ו. הציגו 10 שורות הראשונות מהקובץ.

ז. הציגו 15 שורות אחרונות מהקובץ כשה-`stdout` הוא קובץ.

ח. הוסיף ל-`stdout` מידע על הקובץ `something`.

## צינורות -- pipes

בעזרת סידוק ניתן לבצע מספר פקודות במקביל, כאשר הפלט (stdout) של כל פקודה מודן לקלט (stdin) של הפקודה הבאה.

זהו אחד הכלים השימושיים ביותר ב-AIX.

הפקודות המשוררות מתבצעות במקביל, ולא אחת אחרי השניה. לכן, הפלט מתייחס להתקבל מיידית, גם אם הפקודה הראשונה עדין לא הסתיימה.

מבנה של צינור:

```
command | command | command ...
```

לדוגמה:

```
ls -l | more  
cat myfile | less
```

```
ls | sort -r
```

לדוגמה:

```
atutinb: finger | sort | lpr
```

הפקודה finger מראה את ה-user מוגן של בלוטיסטוס וברכט למשבצון, הפקודה ls ממיינת את הקט שלה וסוציא אונוט בפלט,

והפקודה sort שולחת למדפסת את הקט ישלה. ولكن כל השורה, תזביס למדפסת את רישיות כל האנשים התחוברים בהע למחשב בסי סדר אלףבת של ח-שם name login.

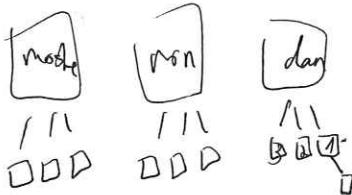
## filters

פילטר היא תוכנית מקבלת את הקט שלה מ-stdin ושולחת את הפלט ל-stdout אלא אם נאמר אחרת.

השילוב בין צינורות לפילטרים הוא כל' חזק ביותר.

תכניות אלו שהכרנו עד עכשוו מהוות פילט רימן:

```
more, less, head, tail, cat
```



תרגיל:

1. צרי עץ תקיות עבור סטודנטים: moshe, ron, dan

2. עבור כל סטודנט צרי 3 תקיות עבור 3 קורסים: dir3Moshe, dir2Moshe, dir1Moshe

3. בתקיתו של דן dir1Dan היכיל קובץ tar1,

4. כאשר את התקיה של דן העתיקי קובץ זה לשאר התקיות של דן,

הפקודה:  $\text{cp dan}/\text{dir1} \sim / \text{dir1}/\text{tar1}$ 

5. צרי העתק נוסף של הקובץ גם בתקיה Dan, dir1Dan

הפקודה:  $\text{cp dan}/\text{dir1} \sim / \text{dir1}/\text{tar1}$ 

6. העתיקי קובץ זה לתקיתו של ron dir1Ron כאשר את התקיה של דן,

הפקודה:  $\text{cp dan}/\text{dir1} \sim / \text{dir1}/\text{ron}/\text{tar1}$ 

7. נתבי את התקיה הנוכחית לתוך התקיה הראשית של משה והעתיקי את הקובץ לתקיה Moshe

הפקודה:  $\text{cp moshe}/\text{dir1} \sim / \text{moshe}/\text{dir1}$ 

8. נתבי את התקיה הנוכחית לתוך התקיה dir2Moshe והעתיקי את הקובץ,

הפקודה:  $\sim$ 

9. העבירי את הקובץ לתוך התקיה dir3Moshe

הפקודה:  $\sim$ 

10. מחקי את הקובץ מהתקיה Moshe, dir3Moshe, מחקי את התקיה dir1Moshe

הפקודה:  $\sim$ 

11. מחקי את כל תקיתו של דן, כולל תת-הספירות והקבצים.

rm -r

מחיקת קובץ	rm	file1	file2	file3 ..
sdade: No such file or directory	rm	sdade		
מחיקת עץ יסטריה עם כל תת-הספירות וקובציים)				rm -r Treenames

מחיקת תקיה ריקה - rmdir

12. השתמשי ב-pipes כדי לשלווה להדפסה את כל הפקודות האחוריות שיצרת.

history | tail

## X-U - הגנה על מערכת הקבצים

לכל קובץ במערכת הקבצים מוגדרים שלושה סוגים הרשאות לשלוש קבוצות משתמשים.

סוגי הרשאות הם :

r - (read) לשריאת תוכן הקובץ (וככל העתקת קובץ).

w - (write) לביצוע ושיימי תוכן הקובץ.

x - (execute) לביצוע או הריצת תוכנית אפסודה.

שלוש קבוצות משתמשים הן :

u - (user) המשמש שיצר את הקובץ.

g - (group) הקבוצה אליה משתייך בעל הקובץ (נקבע ע"י איש ה- system).

o - (other) שאר המשתמשים.

הרשאות מקובצות בקטגוריות של שלוש, בשהלישיות הרשאות מצינה הרשאות עbor המשמש, שהלישיה השנייה מצינה הרשאות עbor המשמש אליה משמשין המשמש והסימזה השלישי מצינה הרשאות עbor שאור המשמשים.

חומר - מסמן שאין הרשאה.

דוגמא :

>> המשטח zmyghaz מפעיל את הפקודה ls -l ומקבל את הפלט הבא על המסך :

ls 15:40 18 feb 1996 46 1 sam year96 1 ---rwxrwxrwx

המו הרואן מהאת סוינ הקובץ, - פרישו שהקובץ הוא קובץ רגיל, פ' זה ספריה.

שלשית החזים הבאים ' u ' מצינום הרשאות קוריאה כתיבה ויצוע עbor על הקובץ ls.

שלשית החזים לאחריהם ' g ' מסמנים של כל המשתמשים האפשרים להבצע של zmyghaz, יפולם לזרוא את הקובץ, אך הם אינם יכולים לבצע לתוכו או לבצע אותו.

שלשית החזים האחרונים ' o ' מסמנם של כל שאר המשתמשים אין הרשאות קוריאה כתיבה או יציע על הקובץ ls.

בעל הקובץ בלבד יכול לשנות את הרשאות לקובץ.

שינוי הרשאות מתבצע באמצעות הפקודה : rm 735 chmod שם הקובץ.

	u	g	o
r/-	r w x 1 1 1	r w x 0 1 1	r w x 1 0 1
	7	5	5

כאשר נפעיל את הפסקודה ls -l

נראה את הרשאות הבאים : rwx - rxr - rwx

אם גנבה chmod 643 xyz

5	4	3
1 1 0	1 0 0	0 1 1
r w -	r -	- w x

מקבל את הרשאות הבאים : r - r -



10

۱۰

26P

לכדי שמשתמש יוכל לראות את כל הלקוחות  
הקיימים בatabase, יתאפשר לו ללחוץ על כפתור  
הקיים ב-Form, ותוצג לו תצוגה של כל הלקוחות  
הקיימים בdatabase. ב-Form ישנו אטטראקטיון  
הנקרא `list`, שמאפשר למשתמש לראות  
כל הלקוחות. ב-Form ישנו אטטראקטיון  
הנקרא `list`, שמאפשר למשתמש לראות  
כל הלקוחות.

chmod [options] filename				שם הרשאות
סוג	x/1	w/2	r/4	
קובץ	הרצה	כתיבה	קריאה	
ספריה	חסימת גישה - כדי שהאחרים יעבדו חכיבים את זה, בנוספַּח חומר גישה לחתת ספריות	להוספה, למחוק או לשינוי קובץ	כתיבה	קריאה
u - user	o - others	g - group	a - all	

```

16~> chmod
usage: chmod [-fR] <absolute-mode> file ...
       chmod [-fR] <symbolic-mode-list> file ...
where  <symbolic-mode-list> is a comma-separated list of
      [ugoa](+|-|=)[rwxXistugo]
17~> chmod 777 test
18~> chmod 888 test
chmod: ERROR: invalid mode
20~> chmod =xwr test
21~> chmod +xwr test
22~> chmod -r test
23~> chmod -R -r test
chmod: WARNING: Permission denied
24~> chmod +or-gw test           /or+, g-w
chmod: ERROR: invalid mode
25~> chmod ugtwx,o-x test

```

הרשאות כתיבה צריכה כתיבה מפורשת של קבוצה ואחרים. בעליים אינם דורש כתיבה מפורשת. a+w,g-w,o+w

צריך עץ ספריות: user1 / user2 / user3

1. הכנסי לספריה user1 וצררי את הקבצים filech1,filech2,filech3

2. עברו הקובץ filech1: [לא בニアרתו]

א. מחקקי את כל הרשאות הקימות:

ב. הגדרי הרשות קריאה עבור כל המשתמשים:

ג. הגדרי הרשות הרצה עבור בעליים וקבוצה:

ד. הגדרי הרשות כתיבה לבעליים בלבד:

3. עברו הקובץ filech2: [לא בニアרתו]

א. מהן הרשות ברירת המחדל של הקובץ?

ב. מה נדרש מיוצר הקובץ לפני שהוא מוצג?

ג. תננה הרשות מלאות לכלום.

ד. הסירiy את כל הרשותות הקימות לשאר משתמשים:

ה. הסירiy הרשות כתיבה לקבוצה:

ו. הסירiy את כל הרשותות הקימות מלבד הרשות כתיבה.

ז. הוסיף הרשות הרצה למשתמש והרשאות כתיבה לקבוצה. (פסיק)

ח. הוסיף הרשות כתיבה לכלום. (א).

ט. הסירiy או הוסיף רק הרשותות מדרשות כדי לקבל:

יב. בעליים- קריאה, כתיבה, הרצה.

יכ. קבוצה- קריאת כתיבה.

יד. משתמשים - קריאת בלבד.

4. עברו הקובץ filech3, הגדרי הרשותות בニアרויות:

rwx,rw,r:

rwx,rwx,rwx:

r--,r--,r--:

w,x,r:

fileHistory (~/) - gedit

Open Save

11 21/18

```
45 b
46 more newfile
47 more myfile
48 less myfile
49 mkdir user1
50 cd user1
51 mkdir user2
52 cd user2
53 mkdir user3
54 cd ~
55 cd user1
56 touch filech1
57 touch filech2
58 touch filech3
59 cd ~
60 cd user1
61 chmod u-rwx filech1
62 chmod g-rwx filech1
63 chmod o-rwx filech1
64 ls -l filech1
65 chmod a+r filech1
66 chmod u+x filech1
67 chmod g+x filech1
68 chmod u+w filech1
69 ls -l filech1
70 ls -l filech2
71 chmod a+wxr filech2
72 chmod o-wxr filech2
73 chmod g-w filech2
74 chmod a-rx filech2
75 chmod u+x,g+w filech2
76 chmod a+w filech2
77 ls -l filech2
78 chmod u+r,g+r,o-W,o+r filech2
79 ls -l filech2
80 chmod 764 filech3
81 chmod 777 filech3
82 chmod 444 filech3
83 chmod 214 filech3
84 ls -l filech3
85 cd ~
86 history>fileHistory
```

Plain Text Tab Width: 2 Ln 62, Col 27 INS

## alias

- ↳ בז'וניקס ניתן ליצור "שמות חינוך" (קיצור) לפקודות עית להקל על השימוש. לדוגמה פקודה ls – יש קיצור והוא ll. כדי ליצור קיצור לפקודה מעתנישים בפקודה alias

`alias "ll"="ls -l"`

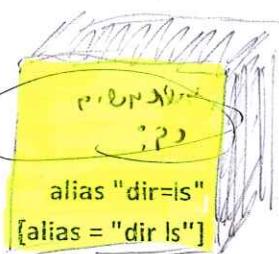
`alias dir "ls -l"`

`alias shalom "kill -1 -1"`

`alias up "cd .."`

`alias cd "cd !*:pwd;ls"`

`alias myecho "echo It's mine !!"`



כעת ניתן להתייחס אל הקיצורים של הפקודות (למשלם) כל פקודות לבב דבר.  
dir יעשה בדיקת מה – ls – ls עשויה (רשימת קבצים עם מידע מורחב).  
shлом תוציא אתכם מהמחשב.  
עפ"ג תעביר אתכם ספרייה אחת למעלה.

נקודות רטיניות בז'וניקס:	alias	כינוי
לצאת מירידם כל בז'וניקס:	exit	
לצאת פירידם כל בז'וניקס:	stty sane	
ליזיידת ב.ג.ז. אפס:	stty sane	
שיטוט בז'וניקס:		
11: No such file or directory		
ביצול כינוי	unalias	
הتلמידות מהיכוני, והרעת הפקורה המקורית	לפקודה	
שא ירים את ls המקורי ולא את הכינוי		

Mkdir->newFolder .1

צרו alias לפקודות הבאות:

1. מחקי את הפקודה `copy`

2. הציגי את הפקודה `alias ls:alias`

3. מחקי את כל ה-salias

Rmdir->deleteFolder .2

Rm -r -> deleteAll .3

רעדן קין גען זיין אט אונלן alias

Cp->copy .4

תהליכיים וקשריהם

מהו תהליך?

תהליך (process) איננו יישום (application) יישום הוא קובץ בדיסק. מנגנון המאפשר לתוכנה לפעול בזיכרון תהליך הוא מופע של יישום, הנמצא כתת מבוצב הפעלה בזיכרון. יכולם להיות מספר תהליכיים של אותו יישום. ניתן להריץ ולהפסיק תהליכיים, ולאחר טבלה של תהליכיים רצים והמשאים המוקצים להם. לכל תהליך יש מספר מזהה ייחודי (pid) המאפשר מעnika מספר מזהה שונה (pid) לכל תהליך שנוצר. כל התיאיותות לתהליך היא דרך ה-pid.

עכ' התהליכים

התהליכים מופעלים בצורה היררכית (עכ'). לכל תהליך שנוצר יש "אב".-Subsystem شמיוני כזו לאף אחד תהליך הראשון שמו פעול בזמן הפעלת המערכת (סס) נקרא init, וה-pid שלו הוא 1. כל התהליכים הם צאצאים של init.

דוגמא:

```
vi <- shell <- login <- init
```

אין גבלה על מספר התהליכים שתהליך מסוגל ליצר.

תהליך האב ממשיך לרווח במקביל עם צאצאיו. סיום של תהליך האב לא משפיע על המשך הביצוע של הצאצא ולהיפך. כאשר הצאצא מסתיים, המערכת מעבירה לאב אינפורמציה על כך. העברה נעשית רק כאשר תהליך האב (במהלך ריצתו) מבקש זאת.

כל עוד הדבר לא נעשה, הצאצא מצוי במצב הקרווי zombie. מושג זה מוגדר כטבלה (ללא פיקוד) אם תהליך אב מסיים את ריצתו לפני הצאצא, הצאצא יומץ ע"י בסיום ריצתו המערכת תעביר ל- init את האינפורמציה על אופן הסיום.

פקודת ps

פקודת ps מציגה רשימה של תהליכים ומזהים.

PID	TTY	STAT	TIME	COMMAND
13183	p3	S	0:00	-tcsh

טווין נט-פואן  
טלון

```

13327 p0 S 0:00 /bin/login
13328 p0 S 0:00 -tcsh
13342 p0 T 0:00 more
13344 p0 R 0:00 ps

```

מצבי תהליכי:

R Runnable

S Sleeping/ Stopped/traced

Z Zombie process

T Terminated

P S

- מוכן

-

-

-

-

-

-

-

-

-

-

אופציית שימושים: מה פירוש כל סימן?

| - פירעון של תהליכים

a - גם תהליכים של משתמשים אחרים

n - מוטן גם שם משתמש וזמן התחלתה

x - מציג גם תהליכים ללא מסוף

w - מציג בפורמט רחוב [לא מוקצת שורות]

f - מודפס גלויים וארצם נאכרים

S - מודפס ערך סיבי ומספר תהליכי

**Job Control**

לשם נוחות המשתמש, ה-shell מותיחס לכל פקודה (pipeline) שהמשתמש יוצר כאיל חטיבה הנקראת job

לכל job יש מזהה (מספר חניוני קטן).

Pid הוא מזהה תחוליך ייחודי למערכת, job הוא מזהה ייחודי למשתמש.

ה-job ייחווית יותר למשתמש.

ה-job יכול להכיל מספר תהליכים.

**דוגמא:**

```
% ls -la | sort > /tmp/delme &
[1] 16980 16981
```

שורה זו תיצור job יחיד המורכב מכמה פקודות. השורה השניה היא ה"תשובה" של ה-shell ומציגה את מספר ה-job

ואת תהליכיים שנוצרו.

הפקודה jobs jobs מוצגה את כל ה-jobs הקיימים.

jobs

הנימוקים הם נספחים ל-`jobs`, ומשדרו נתונים על הפעולות  
לעומת `jobs` (לפחות <sup>15</sup> לינוקס). נספחים ל-`jobs` הם `jobs` (לפחות <sup>15</sup> לינוקס) או `jobs` (לפחות <sup>15</sup> לינוקס).

[1] + Running

emacs

הצגת `jobs` כולל מספר תהליכיים.**פקודות לניהול תהליכיים**

לכל job שמשריצים מוצמד מספר, כמו כן ישנו מספר דרכים לצין תהליך מסוים,

דוגמה	הסבר	syntax
%!	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	%n
%forever	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	%s
%for	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	
%?ever	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	%?s
	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	%%
	המספר המוצמד ל- <code>job</code> (לפחות <sup>15</sup> לינוקס)	%-

fg %1 job

**הפסקת פעולה של job**בעזרת פקודה `kill` ניתן להפסיק גם פעולה של `job` (כל הפעולות ב-`job` ייפסקו).הารוגומנט (מצהה `job`) מתחילה ב-% כדי להבדיל ממזהה תהליכיים.

לדוגמה:

kill %1

**foreground / background**ניתן לסיים `job` בהזית בעזרת צרוף המקשיים ctrl-Z או ctrl-C.ניתן לעצור `job` בהזית ולהעבירו אל רקע בעזרת צרוף המקשיים ctrl-Z.ניתן להחזיר `job` להזית באמצעות הפקודה fg %1.ניתן להעביר `job` שערץ להמשך ריצה ברקע ע"י הפקודה bg %1.ניתן ליצור תהליכי ברקע ע"י הפקודה bash &.ניתן לסיים `job` ע"י הפקודה kill -9.חסול ודאיבוד תהליכי kill -kill -kill (ב-`bash`).כל הפיקודות דלעיל מקבילות מספר `job` כמפורט.שיטסה ב-`shells` היא תכונה של `shells` בצירוף תמייהה ממושכות הפעולה.

kill -9 %1 kill -CONT

done

↓ next/current →	stop	foreground	background
stop		ctrl-Z	kill -STOP pid kill -STOP %jid
foreground	fg %jid		fg %jid

Example:

```

1 > cat ensof.c
main()
{
while(1);
}
2 > gcc -o ensof ensof.c
3 > ensof
^Z - וריאנט של Ctrl-Z
[1] + Suspended ensof
4 > jobs
[1] + Suspended ensof
5 > jobs -l
[1] + 11068 Suspended ensof
6 > ps PID TT S TIME COMMAND
6822 pts/24 S 0:00 -tcs
11068 pts/24 T 0:01 ensof
7 > fg %1
ensof
^Z - וריאנט של Ctrl-Z
[1] + Suspended ensof
8 > bg %1
[1] ensof &
9 > jobs
[1] Running ensof
10 > kill -STOP 11068
[1] + Suspended (signal) ensof
11 > jobs
[1] + Suspended (signal) ensof
12 > bg %1
[1] ensof & -
13 > fg %1
ensof
^C - וריאנט של Ctrl-C
14 > jobs
15 > ensof &
[1] 11173
16 > kill -KILL %1
[1] Killed ensof
17 > jobs
18 > ps
PID TT STAT TIME COMMAND
7348 pts/24 S 0:02 - tcs

```

ההצגה מראה בדרכו ההפוכה של הפקה, המדריכת: תשומן על noden-i ערך של הקישורים אליו נמחקים.

ההצגה מראה בדרכו ההפוכה של הפקה, המדריכת: תשבץ מילויים.

ההצגה מראה בדרכו ההפוכה של הפקה, המדריכת: תשבץ מילויים.

### לינק רכזיאקטי - Hard Link

לינק רכזיאקטי קשור למקום המקורי ב\*)((hard link)) נסמן שטshi שונעתה (חוץ ממתקה) בו משוכן אותו הנקבץ.

במארחת עצמה במקורה של נתיחה, המדריכת: תשומן על noden-i ערך של הקישורים אליו נמחקים. מדריכת מראה בדרכו ההפוכה של הפקה, המדריכת: תשבץ מילויים.

```
% ln file_name link_name
```

כדי לצפות בlienק קשה, משתמשים בו:

```
% ls -i  
1780 link_name  
...  
1780 file_name
```

כך אפשר לדעת שני הקבצים האלו מצביעים לאותו מקום במערכת קבצים, ולשניהם יש את אותו inode.

אנו ניתן לראות שיש שני מצביעים לקובץ.

lienק סימבולי - Soft Link lienק סימבולי הוא מנגנון שמיידר קבץ אחר במערכת וכאן מהיקת הקובץ המקורי יגרום "lienק שבור". כלומר, כאשר נסעה אליו יתגלה

לראש אלlienק נקלט התראה: "הקובץ לא נמצא"

הערה: אם לאחר שמחקנו את הקובץ המקורי ניצור בשלב כלשהו קובץ בעל אותו שם הלינק יצבע על הקובץ החדש

```
% ln -s file_name link_name
```

כדי לצפות בlienק סימבולי, נבצעו:

```
% ls -l  
-rwx----- 1 clark=kiosk=yummi 3 Sep 3 15:30 myfile.new  
lwxrwxrwx 1 clark=kiosk=yummi 10 Sep 3 14:41 mylink ->  
myfile.new
```

כפי שנראה, מופיע חז (>) איפה שישlienק סימבולי.

תרגילים

תרגילים

1.

הරיצי קובץ forever ברקע. forever

2.

הריצי pwd man ברקע. (man זה קובץ עצמה לכל פקודה שהיא. נסוי אותו)

3. כיצד תבדק אילו תהליכי שלך רצים כתע ? ps -a

4.

הציג את כל התהליכי (של כלם) הוצאים כרגע על המחשב ? ps -a

5.

צרי pipes (תהליכי המעבירים מידע ביניהם). איך תודיעי jobs שלך רצים ברגע זה ? ps -a | sort -t|

6.

כיצד ניתן להסיל את הסקריפט forever שרש ברקע ? kill -KILL

7.

העבירי את pwd man לחזית. כיצד ניתן להסיל אותו ? fg

8.

מהו \$ ^ לעומת \$ ^ ? fg

9.

השתמשי בפקודות JOBS PSI כדי לעקוב אחריו התהליכים.

a. הריצי את התהילן forever בחזית. fg

d.

עצרי אותו. fg

לעומת \$ ^

history > file

- העבורי אותו לrinch ברקע.
- צרי תהליך נוסף ברקע (כלשהו כמו vi)
- סימן את התהליך הקודם (PID השתמש בPID)
- הרשי שוב את התהליך ברקע (התהליך מקבל מזהה חדש). vi ↪ bg
- השתמש בPID כדי לעצור אותו ברקע.
- העבורי לחזיות. ↪ kill -KILL
- סימן את התהליך.

קשרו לקובץ:

10. צרי קובץ חדש בשם newmyfile ושים בו את רשימת הפירות האהובה עליו, צרי-link סימבולי לקובץ זה, צפי במה שעשית בעדרת -ls

11. מחקי את הקובץ המקורי myfile (rm) וכמי לעורר את הלינק שיצרת, מה קורה? צפי במצב בעדרת -ls

12. נעת צרי קובץ חדש אחר, וצריו לינק רגיל, בדקי את התוצאות בעדרת -ls מה משמעות הפלט?

13. מה קורה כאשר תשבץ את הקובץ המקורי? ↪ rm myfile\_new

14. מה קורה כאשר תמתקן את הקובץ המקורי? ↪ rm myfile\_new

FIND

syntax: find <local or sub directory> <search criteria> [<action to be taken>]

<criteria>

-name : חיפוש עפ"י שם הקובץ

example : find . -name targil.c

[a wild card search using : \*.c or \\*.c or ???.c]

חיפוש לפי סוג הקובץ :

-type d : directory

-type f : regular file

-type c : character (printer)

-type b : block device ( cdrom... )

-type l : symbolic link

חיפוש לפי גודל הקובץ : size-

syntax: -size <sofar>

-links : hard links.

syntax: -links <links> (מוחפש קבצים שטוף, זה קישורים אליו הם הוא המספר הרשום) > מספר&הו

חיפוש לפי שם ה-user :

-group : group חיפוש לפי שם ה - group

הערה: ניתן לרשום את ה UID או את ה GID במקומם שם ה USER או ה GROUP.

-inum (inode number) : inode חיפוש לפי ה - inum

-atime (access time) (ימיסת ח' כלומר 24 שניות במספר ➔ ח') חיפוש לפי זמן הגישה : n

-ctime (create time) חיפוש לפי זמן יצירה : n

-mtime (modification time) חיפוש לפי זמן שינוי : n



חיפוש עפ"י רמות היררכיות מהספרייה הנווכחית ועד למספר הrama שמצוין - level : level

עורך החיפוש החל מrama level - mindepth <level> : level

example: find . -mindepth 2 -maxdepth 2 (תוקן רמה 2 בלבד)

find -name file-type f -mindepth 2 -maxdepth 2

דוגמה:

בצעו את הפקודה: find -name myfile -print

בדוגמא זו תחילת החיפוש היא מהספרייה הרואית שילכים, מה שבודך כלל עשיים. אם אתם רוצים לטעות חיפוש מהספרייה שבה אתם נמצאים ברגע זה אז במקום / צריך לכתוב (נקודה)

find . -name myfile -print

תרגילים

ברכי מה עשו הפקודה

find /user1/ -size +100k

find /user1 -empty

find ~/ -links +3

-empty -היה שאין לו כל דבר

find . /a -type d -empty  
היה שאין לו כל דבר

## SORT

syntax: sort [-cmnrt] <filename>

Default = z (מיון לפי ה-א' ב')

תאור: מיון שורות של קבצי טקסט. (ניתן למציג שורות של מס' קבצים).

-c (check) (no=0=by 1=yes) בדיקה האם הקובץ כבר ממויין :

הזיפה על המסר בלבד - לא שינוי הקובץ (sort -m f1 f2) (sort -m f1 f2) מיזוג בין שני קבצים שכבר מופיעים

-r (reverse) : sorting from z to a.

-f (Disable Case sensitive)

-n (numeric) מיון של מספרים מהקטן לגודל :

-t (tab) (Default= ) שינוי המפער בין שדות בתוך הקובץ, השינוי תופס לשורת הפקוודה הנווכחית בלבד :

מיון לפי מס' השדה בקובץ (הספרה מתרילה משידה 0): sort +3 (3 is an example, any number..):

מיון לפי שדה <מספר 1> לפי תו <מספר 2> בשידה זה: sort +<מספר 1><מספר 2>

Sort +3 -4 +5 -6= מיון לפי שדה 3 ואת"כ לפי שדה 5 > Sort by x than by y

הפקודה:

sort +2 file

תציג את הקובץ כשהוא ממויין לפי המילה החשישית בכל שורה זו

(פתוחניים למספר את מקומות המילים בשורה פשוט, כמו במערכתם. ולא באחד).

## תרגום

## הפקודה FIND

- find . -name "a"*
- ערכי חפש ע"פ שם קובץ.
  - ערכי חפש למציאת קבצים בלבד.
  - ערכי חפש להקבצים הקטנים מ-5k.
  - ערכי חפש להקבצים הגדולים מ-8k.
  - ערכי חפש לפי שם המשמש שלן.
  - ערכי חפש לפי זמן ייצירה.(קבצים שנוצרו ביום האחרון).
- find -ctime -1*

## 2. הפקודה SORT

## הרטמי בקובץ phon\_list

- sort phone.txt*
- Sort -r phone.txt*
- sort -d phone.txt | sort -n phone.txt*
- מינוי את קובץ.
  - מיית הקובץ בסדר יורד.
  - מיית הקובץ כולל מון מספר לתוכן קובץ. [שים לב להבדיל]
  - מצגי בין הקבצים והן נסרים.
  - מינוי לפי השדרה מס' טלפוני.
- \*\*\*\*\*

## הקובץ phon\_list

a 4  
h 455  
b 82  
t 144  
z 678  
w 333  
044  
c 364

GREP (global regular expression printer)*לען גובץ*

פקודה זו מיועדת לחיפוש בקובץ מסוים (או קבצים) מחרוזת מסוימת.

`$ grep options 'stringname' file-list`

(\*)

*רפרנסים*

אפשרויות - פרמטר זה לא חובה  
 מחרוזת (רצף תווים) מחרוזת שאוטו בראונבו לחפש  
 קובץ (או רשימה של קבצים) שבו מוחפשים את המחרוזת

כasher:  
 - options  
 - stringname  
 - file-list

דוגמא: ב-Home directory נמצא הקובץ הבא בשם net.story:

Local network was installed in 1993. In 1995 global  
 network was installed, while local network was uninstalled. It was a  
 nice time of our company, but soon, the manager reinstalled local  
 network and installation of global network was accomplished.

`$ grep 'local' net.story`

(1)

פקודה

מחפש בקובץ net.story את המחרוזת local. אחר הביצוע של הפקודה grep, UNIX מציגה את השורות המכילות מחרוזת local:

- 2.
- 3.

אם המחרוזת local לא נמצאת בקובץ (קבצים) file-list (קבצים), אז, לאחר הביצוע של הפקודה grep, UNIX לא מציגה שום דבר.

בדוע הפקודה (1) לא מצאת את המחרוזת local בשורה הראשונה של הקובץ net.story. יש להשתמש באופציה -i של הפקודה grep, וכך אם בראונבו לא להבחין בין אותיות גדולות לבין אותיות קטנות, \$ grep -i 'local' net.story

结出的匹配结果是什么？

- 1.
2. network was installed, while **local** network was uninstalled. It was a
3. nice time of our company, but soon, the manager reinstalled **local**!

הערה: פקודה grep סורקת את הקובץ הנתון, שורה אחר שורה, לכן אם מחרוזת שלמה לא נמצאת בשורה אחת, אז פקודה grep לא תמצא את המחרוזת. לדוגמה, פקודה \$ grep

`$ grep -i 'local network' net.story`

תמצוא את המחרוזת local network רק בשורות ה-1 ו-2 ולא תמצא בשורות

**Local network** was installed in 1993. In 1995 global  
 network was installed, while **local network** was uninstalled. It was a

כפי שאמור לעיל, הפקודה grep מוחפש מחרוזת הקובץ. מחרוזת זו יכולה להופיע באמצע  
 מילה, בתחילת המילה או בסופה. לכן, לאחר ביצוע של פקודה:

`$ grep 'installed' net.story`

"找到的短语"

- 1.
- 2.
- 3.

אם בראונבו למצוא מילה שלמה, יש להשתמש באופציה -w (word). לדוגמה, אותה הפקודה עם אופציה -w:

`$ grep -w 'installed' net.story`

找到的短语

בפרמטר file-list אם יכולם לכתוב שם שטחלי סימנים גלובליים: \* ? [ ] .(ambiguous file reference)

במקרה זה יוצגו כפלט את שמות הקצים שבהם נמצאת המחרוזת יחד עם השורות. נניח שב-directory ייחד עם קובץ (\* A is in [1..10] set \*) (\* A in [1..10] then A:=A+1 else A:=A-1  
חישוב תוצאות  
בגון כל היותר גוף סעיפים  
net - מינוס אחד או יותר  
אנו בפער של פקודה)

\$ grep 'in' net.\*

ויצג כפלט

net.story: \_\_\_\_\_

net.pas: \_\_\_\_\_

net.pas: \_\_\_\_\_

### היפוך מתקדם בפקודה grep

דוגמא: נתון קובץ ASCII הבא בשם phonelist

Zachary	703-1993
Lerner	205-1965
David	876-3654
Matthew	876-2765
Cameron	875-4331
Rachel	297-5452
Kathy	345-2765
Megan	346-1372

- פקודה `grep "Kat"` חפש בקובץ phonelist מחרוזת: `Kat`
- סימן ^, תחילת שורה: `>grep '^Kat'` חפש בקובץ phonelist מחרוזת: `Kat`
- סימן \$, סוף שורה: `>grep '$Kat'` חפש בקובץ phonelist מחרוזת: `Kat`
- סימן ^...\$, תחילת וסוף שורה: `>grep '^...$'` חפש בקובץ phonelist מחרוזת: `Kat`
- פקודה `>grep 'er'` חפש בקובץ phonelist מחרוזת: `er`
- פקודה `>grep '...er'` חפש בקובץ phonelist מחרוזת: `er`
- שטוח נקדיות משמעותן לא אומן, מילוי איזו נקיון: `>grep "[KkMm]"` חפש בקובץ phonelist מחרוזת: `KM`
- סימן [] - מפחים על קווים: `>grep "[A-D]"` חפש בקובץ phonelist מחרוזת: `A,B,C,D`
- פקודה `>grep 'mct'` חפש בקובץ phonelist מחרוזת: `mct`
- פקודה `>grep '^..[mct]'` חפש בקובץ phonelist שתי נקדות אחריה הפסיק ^ מחרוזת: `mct`

**GREP** syntax: grep [-cinvw] pattern [file...]

מחפש מחרוזת או חלק מחרוזת  
 -c: (count) : pattern : מ"ס' השורות שמכילות את ה-pattern  
 -i : case sensitive : מבטל הבדיקה בין אותיות  
 -n : (numbering) : pattern : מכרף את מ"ס' השורה שנמצאה בה ה-pattern  
 -w : (word) : pattern : מחפש רק מילה שלמה :pattern  
 -v : pattern : הצגת השורות שלא מכילות את ה-pattern

**(אפשרויות חיפוש מחרוזות כלליות (regular expressions))**

Symbol	Meaning	Examples
.	default: מחפש מחרוזת או חלק מחרוזת	Grep -w ... grep -w ...
^	כל שורה abc יופיע בתחילת השורה: abc	ש-abc
\$	כל בסוף שורה: xyz\$ שורות שמתחלות ונגמרות: ^abc\$	abc במחוזות (כלומר יש בשורה רק abc)
<	האות שבocz ואחריה אחת: A[bc]	<abc
>	האות שבocz ואחריה אחת: A[bc]	Xyz>
[]	האות שבocz ואחריה אחת: A[bc] מהאותיות שבסוגרים: [abc]	A[bc] [abc]
[^]	בכל מקום לא-abc: abc[^abc]	[^ABC]
\	(Like in the shell) \*.*\$	\*.*\$

**(אפשרויות מורחבות(Extended regular expressions))**

(Syntax: grep -E expression or egrep)

Symbol	Description	Examples
*	האות שבocz ואחריה אחת או יותר: *z-a	A[za-z]
?	האות שבocz ואחריה אחת או האיבר שבסוגרים ()	grep -E ab?c → abc or ac
+	האות שבocz ואחריה אחת או יותר: ab+	grep -E [ab]+
{מספר}	הציג את התו בדיק 3 פעמים	grep -E {3}
{n,m}	הציג את התו בדיק n-m פעמים: {n,m} מופעימים: {1,5}	grep -E [+]?{0-9}{3,5}

Example for a regular expression in C language:

Grep -E "([a-zA-Z][a-zA-Z\_0-9]{31})"

הסביר: את ראשונה אותיות קטנות או "\_" את שנייה והלאה יכולה להיות גם סיפה, מקסימום אורך שם = 31.

grep -E a[a-z]\*[a-zA-Z\_0-9]{31} הוכנסו ב- פונקציית grep  
 זו שטוף נגזר

## תרגיל

צורנו מה מהפץ הפקודה, הוסיפו את שורות הפלט מתוך הקובץ המצורף:

```
*****string
moshe
moshe & reuvaen and david and ravid and rachel and rivka
david
ravid rachel
rachel
i meet you
did you met me?
moal and noal
noal and moal

i love banana and cucumber
i dont love melon
dad daaaad dadada dad dabad
```

```
*****phone_numbers
```

```
4
456
234
144
678
333
044
364
```

ביטויים רגולריים ב- grep סטנדרטי:

1. כל תו שאינותו מיחיד מותאים לעצמו.  
דוגמא:

```
grep moshe *
```

מחפש את המילים

2. הסימן ^ בתחילת ביטוי מצמיד את הביטוי לתחילת השורה. הסימן \$ בסוף שורה מצמיד את הביטוי לסוף שורה.  
דוגמאות:

```
grep ^moshe *
```

מחפש את כל היעזרות

```
grep david$ *
```

מחפש את כל השורות

```
grep ^rachels$ *
```

מחפש את כל השורות

```
grep ^$ *
```

מחפש את כל השורות

3. " " (נקודה) מותאמת לכל תו פרט לתו newline (\n) דוגמאות:

```
grep .*
```

מחפש את כל השורות

`grep avid`

ימצא את השורות בהן מופיע avid

4. רצף סימנים בסוגרים מרובעים [] מתאימים לתו אחד מותוכם. אם הסימן ^ מופיע ראשון בתוך הסוגרים המרובעים התכנית מתאימה לכלתו פרט לשאר התווים בתוך ה-[] . הסימן "-" (מיינס) בין שני תווים מציין רצף תווים בין שני התווים שמלצדיו.  
דוגמאות:

`grep [dr]avid *`

يُعْثَرُ أَنَّ الْمُشَوَّهَاتِ شَبَهُنَّ مَوْفَعِيًّا

`grep [1-9][0-9]4`

يُعْثَرُ أَنَّ الْمُشَوَّهَاتِ شَبَهُنَّ مَوْفَعِيًّا رَّجَفِيًّا شَلَّ

`grep ^[^a-zA-Z]`

يُعْثَرُ أَنَّ كُلَّ الْمُشَوَّهَاتِ شَلَّ مَتَّهِيلَةٍ

5. אחרי כל אחד מהניל' יכולת לבוא הוככברת '\*' וביתוי החדש מתאים לרצף של אפס או יותר מופעים של מה שבא לפני הוככברת. יש לשימוש את וביתוי במרקאות כדי לא לבלבל את ה- shell . דוגמאות:

`grep "da*d" *`

ימצא את המילים שבחן מופיע:

`grep *4*4[0-9]*`

ימצא את כל המילים שבחן מופיעים מספרים מסוימים המסתויים ב-4 והם לא 4 (יש לפחות עוד ספרה בהתחלה).

6. אם אחרי כל אחד מהביתויים -1-5 באים סוגרים פשוטים עט מספר בתוכם, הביטוי החודש יחשוף את מה שיש לפניו סוגרים מסוימים ליט' מספר פעמים כבו' שרשום בתוך סוגרים הממולסים. צריך להקדים את סוגרים עם backslash,＼, כפי ש�示 בדוגמאות למטה.  
דוגמאות:

`grep * \"me{2}\\"`

ימצא את "meet" אבל לא את

`grep * \"m{2}\\"{2}\\"`

ימצא את כל המילים המתחילות באות "m"

7. ניתן לשימוש ביטוי רגולרי בסוגרים כדי להתייחס אליהם בהמשך. גם כאן צריך להקדים את סוגרים ב- backslash. החתימות להמה שמצאו בסוגריה היא על ידי \\. הסבר בדוגמה:  
דוגמה:

`grep "\([aouie][aouie]\)\\"`

ימצא מילים מסווג

.\"papa",

כלומר במקרה שחוור בחן הרצוף "עיצורי-תנוועה" פעמיות. לא יתאפשר למשל מילים כמו "sota" שכן הרצוף הראשון והשני שונים, ta != so.

8. סימנים מיוחדים "<" ו- ">" מתאימים לתחילת מילה ולסיומה בהתאם.  
דוגמה:

`grep <[a-zA-Z]\{4\}>`

ימצא את כל המילים שיש בחן מיללים

VI הוא עורך טקסט בלינוקס.

### শ্বি מוצבי עבודה

בניגוד למრבית העורכים בעורך וקיימים שני מוצבי עבודה: מוצב עריכה - במצב זה כל תו שמדובר מתווסף לקובץ בהתאם למיקום הסמן (כמו בעורכים emacs word וכו').

מצב פקודה - במצב זה, כל תו שמדובר מתרפרש ע"י העורך כפקודה או חלק ממנו. כפי שנראה בהמשך, קיימות פקודות רבות וشنויות. חלון מרכיבות מתו אחד בלבד, וחלון מרכיבות מספר תוים או אף ממשפטים שלמים.

דוגמא להמחשה:

הקשה על התו x כאשר העורך במצב עריכה תוסיף את התו x לטקסט במיקום בו הסמן נמצא.

לעומת זאת, הקשה על התו x כאשר העורך במצב עריכה תוסיף פקודה ונמחק את התו הנוכחי, במקרה זה x פועל לפי תפקידו בראשית הפקודות של העורך שהוא מוחיקת אותן. בדפים הבאים נלמד את הפקודות השונות של העורך שיישמשו אותנו בעבודתנו עם העורך.

שים לב: לא לכל התווים יש משמעות במצב פקודה אולם הקשה על то ללא שימוש במצב פקודה לא תשפיע על הקובץ.

### פקודות בסיסיות

כפי שראינו קיימים שני מוצבי עבודה ב:VI -

מצב עריכה ומצב פקודה:

במצב פקודה ניתן לנוט בקובץ באמצעות הקשה על מקשי החיצים (לא ניתן לעשות זאת במצב עריכה, ובד"כ נקבל תווים לא מובנים על המסך אם נעשה כן).

ניר כאן כי במצב פקודה לא ניתן לכתוב טקסט, וכי כל הקשה על מקש נחשבת ע"י העורך כפקודה או חלק ממנו.

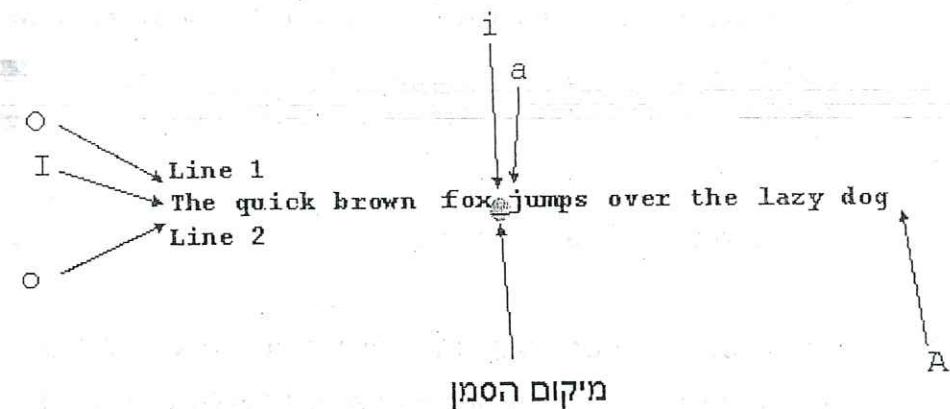
מעבר למצב פקודה מתבצע ע"י הקשה על מקש ה-Escape -

מעבר למצב עריכה מתבצע ע"י הקשה על אחד המKeySpecים הבאים, לפי הצורך:

וין	משמעות
	a
	A

	i
	I
	O
	o

המיצים מסמנים את מיקום הלו הבא, בהתאם לאופציה שנבחרה  
(שאר הטקסט יוז בהתאם)



### שימוש לבן

- א. פקודות ב VI -hn. case sensitive
- ב. בתחילת העבודה, העורך נמצא במצב פקודה.

### עבודה עם קבצים

בשיעור הראשון והשני למדנו על שני מצבים העבודה של העורך ויכד לעבור מצב אחד לשני. בשיעור זה נלמד פעולות בסיסיות של פתיחה שטירה ויציאה מקובץ. vi פתיחת קובץ:

על מנת לפתוח קובץ חדש (או לחילוף, קובץ קיים), נקליד "n" ואת שם הקובץ. לדוגמא, הפקודה:

vi my\_file.txt

פתח קובץ בשם my\_file.txt מהתקיה הנוכחיית. אם הקובץ קיים כבר, תוכנו יוצג על-גבי המסר. אם הקובץ אינם קיים, על המסר יופיע "קובץ ריק".

כעת אנו יכולים לעורר את הקובץ.

על מנת לשמור את הקובץ علينا לעבור במצב פקודה, ולהשתמש באחת האפשרויות

## הבאות:

פקודה	משמעות
w:	
bw:	
:q!	
:w <filename>	
:w! <filename>	

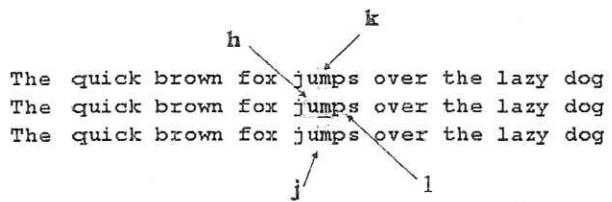
תרגול  
שאלה מ' 1, שאלה מ' 2, שאלה מ' 3

## כיווט בקובץ

פעולה חשובה בעורך היא תזוזה על פני הטקסט או דילוג לחלקים שונים במסמך. לעיתים נדרש להציג את הסמן על פני הטקסט כדי מה אחורה למטה או למעלה באורה פשוטה. לעיתים נטע קדימה או אחורה על פני המסמך בתזוזות של מסך או של חצי מסך בכל פעולה. ולעתים נדרש לדילג לשורה ספציפית במסמך.

תזוזה שלטו בודד בקובץ מתבצעת במצב פקודה ע"י התווים i, j, r, h -

i	
h	
k	
j	



- מיקום ומכה של הסנן
- מיקום הטמן לאחר והשוו על המקש המזון

שימוש לבן

השימוש בחצים שבמקלדת אפשרי ברוב המקרים אך מוטב להתרגל לשימוש בתווים הנ"ל, שאפשר תמיד

תיזוזה בקובץ אינה מעבירה את העורף למצב הוספה אלא פשוט מאפשר טויל בקובץ.

**תנוועה על המסן**  
 באמצעות צירוף המקלשים ctrl + f, d, s - או

n <sup>v</sup>
p <sup>v</sup>
f <sup>v</sup>
b <sup>v</sup>

**דילוג לשורה ספציפית בקובץ**

במצב פקודה יש להקש אט התו נקודתיים (:), ולא חיריהם (בצמוד) את מספר השורה.

לדוגמא: 3:

יעביר את הסמן לתחילה השורה השלישית בקובץ.

**פקודות נוספת לדילוג**

	:\$
	\$

שימוש לבן

אחרי תרגול קצר, התיזוזה בקובץ בעזרת הפקודות הופכת להיות נוכה מאוד.

## מחיקת טקסט

אחת הפעולות הנפוצות בעורך הנהנה מחיקה של טקסט.

כפי שראינו, בערך ובייצוע פעולות מתבצע במצב פקודה. אך, כדי למחוק טקסט, נбурר תחילת במצב פקודה(באמצעות המקש escape - - )

ב-ו-קיימות פקודות שונות עבור צורות מחיקה שונות, החל ממחיקת אחת וכליה במחיקת מספר שורות.

שימוש לבן

מחיקת טקסט ב, ו-מ恰恰ה בערך ע"י התו d במצב פקודה.

להלן התווים המשמשים לפקודות המחיקה השונות :

	x
	dw
	dd
	d\$
	^d

The quick brown fox jumps over the lazy dog

x	The quick brown fox jumps over the lazy dog
dw	The quick brown fox jumps over the lazy dog
dd	The quick brown fox jumps over the lazy dog
d\$	The quick brown fox jumps over the lazy dog
^d	The quick brown fox jumps over the lazy dog

\* - מיקום נכון של הסמן

- מסנן את האותיות שטמךנה לאחר הקשה על הפקשים המתאימים

נעיר כאן כי ניתן להציג פקודות המחיקה גם מספר, שיצין את מספר הפעמים שברצוננו לבצע את הפקודה.

לדוגמה:

8x

יבצע את פקודה "מחיקת האות הנוכחית" שמוגנה פ עמים, כר שלמעשה נמחק 8 אותיות החל מהתו הנוכחי.

דוגמא נוספת:

3dd

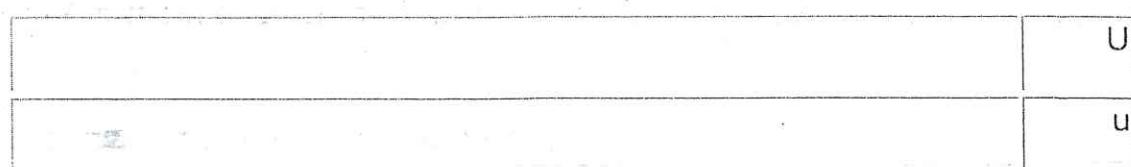
ימחק 3 שורות מהקובץ.

נקדים את המאוחר, ונעיר כאן כי לרוב הפקודות ב-z -ניתן להציג מספר, שיציין את מספר הפעמים שנרצה לבצע את הפקודה.

#### שאלה מס' 4

#### חרטה, חזרה וצירוף

חרטה



דוגמא:

השורה המקורית:

The quick bro fox jumps over the dog

השורה לאחר שינויים (השינויים בוגרים משלאל לימיון, מבליל לעבור לשורה אחרת):

The quick brown fox jumps over the lazy dog

*שינוי תייר*

השורה לאחר ביצועם (ביטול שינוי אחריו):

The quick brown fox jumps over the dog

השורה לאחר ביצועם (ביטול כל השינויים):

The quick bro fox jumps over the dog

חזרה

על-מנת לחזור על ביצוע הפעולה האחורונה יש להקיש על התו 'l' (נקודה).

צירוף שורות

על-מנת לחבר את השורה הנוכחית לזה שאחריה, יש להקיש על התו 'l'.

לפני הקשת l

*הסמן נמצא כאן*

The quick brown fox  
jumps over the lazy dog

אחרי הקשת l (שימוש לב שהמילים fox и jumps מחוברות)

The quick brown foxjumps over the lazy dog

#### שאלה מס' 5

### חיפוש טקסט

חיפוש טקסט יבוצע ע"י הקשה על התו 'ו', ובצמוד אליו את המחרוזת לחיפוש (במצב פקודה, כמובן)

דוגמא:

/Jerusalem

חיפוש המופיע הראשון של המחרוזת Jerusalem בקובץ וידוג אליה. החיפוש יתחיל מהמקום הנוכחי של הסמן.

:se number

כיצד נбурר לモפעים הבאים של המחרוזת בטקסט?

	ו
	N

אם גרצה בכר, יוכל להתחיל מיד בחיפוש בכיוון "אחרה", וזאת ע"י שימוש בתו 'ו' במקום בתו 'ו'.

דוגמא:

?Jerusalem

ידלג למופיע הבא של המילה Jerusalem משמאל לסמן (כלומר - חיפוש בכיוון "אחרה").

הערה: הפיקודה set number תמספר את הקובץ.

### העתקת טקסט

העתקת טקסט מתבצעת בעיקר ע"י התו 'ז' ופקודות ההעתקה מתבצעות במצב פקודה. כפי שLEARן בשיעור על מחייקת טקסט גם להעתקת טקסט קיימות פקודות שונות למצבים השונים- העתקת אות מילה או שורה. כמו כן ניתן לציין מספר שיקבע את מספר הפעמים שהפעולה תתבצע.

להלן התווים המשמשים לפיקודות ההעתקה השונות :

	ז'
	yw
	ז
	y\$
	y}

(ניתן להעתיק טקסט לאוצר ספציאלי וכן לשמור באוצרים שונים מספר ההצעות במקביל).

## הדבקות טקסט

התו **d** משמש להדבקת טקסט:

	<b>d</b>
	<b>P</b>

הפעלת הפקודה תדביק את הטקסט האחרון שנשמר באוגר - זה שהעתיק אחרון או זה שנמחק אחרון, על ידי פקודות העתקה או המחקה של העורך שראינו בשיעורים הקודמים.

העורך ו/או אפשר גם פניה לאוגר ספציפי והעתקה של טקסט אליו או פניה לאוגר ספציפי והדבקה של הטקסט שמאחורי באותו אוגר.

התחבר הוא פניה לאוגר ע"י גרשימים ומספר האוגר ולאחר מכן כתיבת פקודת העתקה או הדבקה.

לדוגמה:

הפקודה הבאה (במצב פקודה - **w5yw9y**) משמשת להעתקה של המילה הנוכחית לתוך אוגר מספר 9.  
הפקודה - **d9** פירושה הדבקה של התוכן שנמצא באוגר מספר 9 (המילה שהעתיק בפקודה הקודמת).

בעזרת השימוש באוגרים אנו יכולים לשמור קטעי טקסט שנזדקק להם מאוחר יותר באוגרים ספציפיים וליעל את העבודה.

## החלפת טקסט

החלפת טקסט מתבצעת במצב פקודה ופעלת על הטקסט החל ממיקומו של הסמן.

שימוש לב!

פקודות ה החלפה הנפוצות עושים שימוש בתו **c** עם תוספות ספציפיות.

עם הפעלת הפקודה מסומנת נקודת הסיום של ההחלפה בתו '**\$**' שיופיע על המסך.

חשוב לזכור פקודות החלפה מעבירות את העורך למצב עריכה וכן יש להקיש escape בסיום ההחלפה.

פקודת החלפה מחליף את השורה הריקה במצב **Insert**.

להלן התווים שמשמשים לפקודות החלפה השונות:

	<b>cl</b>
	<b>cw</b>

	CC
	C\$

**חיפוש והחלפה פשוטים****חיפוש והחלפה חד פעמיים:**

במצב פקודה, יש להזכיר 'ז' מיד אחר-כך את התו '/' את המחרוזת להחלפה, שוב את התו '/' ואת המחרוזת החדשה.

ניתן גם לגביל את טווח הפעולה, כך שפעולות ההחלפה תבוצע רק בקטע מסוים (עדין, רק פעם אחת). עושים זאת ע"י הקולדת מספרי השירות ביןיהן נרצה לבצע את ההחלפה, כשהן מופרדות בפסיק, מיד לאחר פקודת החלפה. דוגמא:

:5,15 s/Jerusalem/Tel-Aviv

תחליף את המופיע הראשון של המחרוזת Tel-Aviv Jerusalem בhetto השורות 5-15 בקובץ הנתון.

לעומת: 5,15 s/Jerusalem/Tel-Aviv: תחליף את המופיע הראשון של המחרוזת Jerusalem בhetto Tel-Aviv בטעות השירות החל משורה 5 עד סוף הקובץ.

**חיפוש בכל המסמך..% -חיפוש בכל המופיעים הראשון...%**

החלפה של כל המופיעים הראשונים בשורה.

חיפוש והחלפה של כל המופיעים של מחרוזת מסוימת בחיפוש והחלפה אלה, נרצה להחליף את כל המופיעים של מחרוזת מסוימת במחרוזת אחרת, על פני כל המסמך או בקטע ממנו.

**s/Jerusalem/Tel-Aviv/g**

אפשר גם:

:g/string\_to\_replace/s//string\_to\_replace\_with/g

כלומר, "g": מיד אחר כך לocusן, מחרוזת להחלפה, לocusן s, שני locusנים והמחרוזת "המחלפה".

גם כן, ניתן להגיד טווח שורות בו תבוצע ההחלפה, וזאת ע"י הוספת "/g" בסוף הפקודה, ומספר השירות על-פנייה יש לבצע את ההחלפה. 10,20g/Jerusalem/Tel-Aviv

דוגמא: 10,20g/Jerusalem/s//Haifa/g: דוגמא להחלפה בתבוצוע אך ורק בטוווח השירות 10-20.

**-חיפוש בכל המסמך – החלפה של המופיעים הראשונים בשורה.**

**.s/Jerusalem/Tel-Aviv/g** – החלפה של כל המסמך.

ניתן גם להשתמש בתו '\$', לציין השורה האחורונה בקובץ, כך שהפקודה הבאה:

\$,g/Jerusalem/s//Haifa/g10:

תבוצע ההחלפה המבוקשת החל משורה 10 ועד סוף הקובץ

## תרגול

## שאלה מס' 1

1. פתחי קובץ חדש בשם tar11.
2. הקלידי את שם משפחתך. (שים לב: עליך לעבור למצב הוסף כדי לבצע הקלהה).
3. עבררי למצב פקודה.
4. הוסיף את שמו הפרטיא לפני שם משפחתך.(או בקיצור בתחילת השורה).
5. צאי מהקובץ כולל שמירה.
6. פתחי את הקובץ tar11 לעבודה.
7. הוסיף בשורה חדשה את העיר בה אתה מתגורר.(השתמש באופציה לפיתוחת שורה חדשה).
8. הוסיף בשורה חדשה בין שתי השורות את כתובבת מגוריך.(כנ"ל רק שהפעם השורה החדשה תהיה מעל השורה שאנו נמצא בה (עט)).
9. צאי מהקובץ tar11 עם שמירה.

## שאלה מס' 2

1. פתחי קובץ חדש בשם tar12.
2. הקלידי את שמו הפרטיא.
3. שמרי על השינויים ללא יציאה מהקובץ.
4. הוסיף בשורה חדשה את שם משפחתך.
5. צאי מהקובץ ללא שמירת השינויים.
6. פתחי את הקובץ tar12 לעבודה.
7. שמרי את השינויים בקובץ חדש tar13.
8. צאי מהקובץ tar12.
9. פתחי את הקובץ tar13.
10. צאי מהקובץ tar13.

## שאלה מס' 3

1. פתחי את הקובץ tar11 לעבודה.
2. הוסיף בסוף השורה הראשונה שתי כוכיות.
3. שמרי את תוכן הקובץ בהובץ tar13 (שים לב tar13: כבר קי"מ).
4. צאי מהקובץ tar11 tar11 ללא שמירת הנתונים.
5. פתחי את הקובץ tar11 לעבודה וצאי ממנו.
6. פתחי את הקובץ tar13 לעבודה וצאי ממנו.

## שאלה מס' 4

1. פתח קובץ חדש בשם tar21.

.2. הקלד את המשפט הבא :

Tzo bzzzze, zzz zor znot zz zzzzto bzzzze

.3. שמר את התוכן בקובץ חדש tar22.

.4. מחק את כל המופעים של האות z بصورة היילה ביותר תוך שימוש יעיל בפקודות המ剔קה

שנלמדו (כולל מספר בפקודה) ותוך שימוש במקשי התנועה .

.5. בטל את כל המחקיות שביצעת بصورة היילה בעזרת פקודה אחת .

.6. מחק את השורה .

.7. בטל את מחיקת השורה .

.8. צא מהקובץ tar21 עם שמירה .

### שאלה מס' 5

בשאלה זו ננסה להקליד את המשפט:

"Summer has always been my favorite time of year"

.1. פתח קובץ חדש בשם tar23.

.2. הקלד את המשפט הבא כפי שהוא מופיע (במספר שורות):

Summer
has has
has has
has
has
always
been

my my my my favorite timmmmmme of year

.3. עברו למצב פקודה .

.4. עברו לשורה 5 .

.5. עברו לשורה ראשונה .

.6. מחק את השורות את המילים ואת האותיות המיותרות, כל זאת بصورة היילה תוך שימוש בפקודה מתאימה ותוך שימוש באופציית המספר (לדוגמא – 2wפ: פקודה למחקת 2 מילים).

.7. צרף את חלקו המשפט לשורה אחת .

.8. צא מהקובץ tar23 עם שמירה .

### שאלה מס' 6

.1. פתח קובץ חדש בשם tar31.

2. הקלד את המשפט הבא לעורך כפי שהוא מופיע

(שים לב: קיימות מספר טעויות במשפט)

I goo to frien friend my visits

3. מיקום של המילים visit i friend הוחלף במשפט תקן את הטעות תוך שימוש בפקודות חיפוש

(למציאת המילה visit גזירה והדבקה למיקומה הנוכחי).

4. גזר והדבק את המילה friends למיקומה הנוכחי במשפט.

5. קיימות שתי שגיאות כתיב במשפט במילה סוף ובמילה visits תקן את השגיאות.

6. ספור, תוך שימוש בפקודות חיפוש כמה פעמים מופיעה האות e במשפט.

7. צרף את המשפט לשורה אחת.

8. העתק והדבק את המשפט פעם נוספת לשורה חדשה.

9. צא מהקובץ tar31 עם שמירה.

### שאלה מס' 7

1. תרגל חיפוש מחוזות ומעבר קדימה ואחוריה על פני המופיעים של המחוזות.

מעבר מחוזות:

אחריה:

קדימה:

2. תרגל שימוש בפקודות העתקה והדבקה תוך שימוש בבאפרים.

אוגר מס' 4:

אוגר מס' 2:

### שאלה מס' 8

#### כתבת תוכנית בשפט C

1. פתח קובץ חדש.

2. כתבי תוכנית בשפט C.

התוכנית מדפסה ע"י לולאה רצף מספרים 1-9.

3. צרי לולאות מקוננות:

העתיק את הלולאה.

החלifi כל מופיע של א-ב-ג.

4. עברו כל כניסה ללולאה הפנימית: הדפיסי מחוזות כלשהי.

5. העתיק את פעולה ההדפסה מחוץ ללולאה: לפבי גוף הלולאה ולאחריו.

6. חפשי את המחוזות שתודפס בתוכנית.

7. עברו על כל המופיעים שלה.

שמרי את הקובץ שוב בסיוםת C.

## תכנות Shell

הזכרנו קיימים סוגים רבים של **Shell** הקידוד מתייחס ל סביבת **bash** .shell bash

### תכנות ב-**bash**

בנוסף לממשק המשתמש **the-shell** הינה שפת תכנות.

לשפה כל המאפיינים הרגילים של שפת תכנות: משתנים, בקרת זרימה, פונקציות, וה坦מישקות חזקה אל מערכת הפעלה.

פקודות בתוכנית יכולות להיות פקודות מערכת חיצונית או פקודות מוגנות (**in-built**) בתוך ה- shell

פקודה המוגנית **echo** מדפסה את רשיימת הפרמטרים שלה.

```
% echo "hello world"
hello world
```

### משתנים ב-**bash**

כמו לכל שפת תכנות, ל-**shell** ניתן להגדיר משתנים ולתת להם ערך.

שם המשתנה יכול להיות כל צירוף של תווים אלפ-נומריים (אותיות ומספרות), שאינם מתחילה בספרה.

השמה נעשית באמצעות האופרטור =

```
myname=sara
```

שימוש בערך של המשתנה נעשה ע"י אופרטור \$.

```
% echo $myname
sara
```

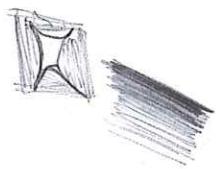
### הפקודה date

מציג את התאריך ניתן לתת פרמטרים ע"מ לקבל את הרצוי  
לדוגמא - היום בשבוע (מלא)

```
% date +%A
Saturday
% date +%d-%m-%Y
22-03-2003
%a %b %e %T %Z %
For example,
```



Fri Dec 23 10:10:42 EST 1988



### לכידת פلت של פקודה למשתנה

הօפרטור - (גרש הפוך) משמש לכידת פلت של פקודה אל תוך משתנה. בטור זוג גרשים הפוכים יכולה להופיע כל פקודת xun.

פלט הפוקודה יהיה ערכו של המשתנה הנקבע בפקודת ההשמה.

```
variable='unix-command'
```

```
% cat backquote
#!/bin/bash
dayinweek=`date +%A`
echo today is $dayinweek
% backquote
today is Sunday vi newfile.txt
```

**כתיבה קבוצי הרצה** קובץ אונס גתקא קזען חטף - אונסן דאלאן דאלאן טב, אונסן  
בדוגמא שלעיל האגנו או'ר מייצרים קובץ בר הרצה, נדרשים שני התנאים הבאים כדי שנוכל להריץ את הקובץ

הקובץ צריך להתחילה (בשורה הראשונה) בציירוף הבא:

```
#!/bin/bash
```

הקובץ צריך להיות בר הרצה כה:

### בקרת זרימה – if

לפקודת if המבנה הבא

```
if condition ; then
    commands
else
    commands
fi
```

התמ" (condition) מורכב ומօפרטור ומօפרודים. אובייה סואן אופרטורים

מחוץתיים

אריתמטיים

אופרטורים לוגיים

אופרטורים עם קבועים

## אופרטורים מחוץתיים

= שווי

!= אי שווי

```
% cat if2
#!/bin/bash
x=file.c
if [ $x != file.c ] ; then
    echo $x is not file.c
else
    echo $x is file.c
fi
% bash if2
file.c is file.c
```

```
% cat if1
#!/bin/bash
x=file.c
if [ $x = file.c ] ; then
    echo $x is file.c
fi
% if1
file.c is file.c
```

## אופרטורים אריתמטיים

-eq, שווה, ושווא בהתאם.

-gt, גדול שווה, קטן שווה, גדול מ-, קטן מ- בהתאם

```
% cat if3
#!/bin/bash
x=6
if [ $x -gt 5 ] ; then
    echo $x is greater than 5
fi

% if3
6 is bigger than 5
```

## אופרטורים לוגיים

!, תוארי לא לוגי!

- א, תנאי גם לוג'
- ו, תנאי או לוג'.

זהים בתחילת לאופרטורים המקבילים בשפהת תכנות.

```
#!/bin/bash
x=5
y=hello
if [ $x -lt 7 -a $y = hello ] ; then
    echo yes
else
    echo no
fi
% if4
yes
```

ה-shell מאפשרת לנו לבדוק תוכנות של קבצים ובהתאם לתוכנות לשנות את זרימת התוכנית.

התוכנות הן:

- א- אם הקובץ קיים
- ב- אם משתמש הרשות קריאה
- ג- האם משתמש הרשות כתיבה
- ה- האם משתמש הרשות ריצה
- ד- האם הקובץ הוא ספרייה
- ה- האם הקובץ הינו קובץ רגיל
- ל- סימבולי (link) האם הקובץ הינו קישור
- ו- האם המשתמש הוא בעל הקובץ

ישנם ארגומנטים נוספים אך לא נverb על כולם

כנich כי אנו בתיקיה מסויימת:

```
% ls -l
total 2
-rw----- 1 yedidia x
-rwxr-xr-x 1 yedidia y
#!/bin/bash
if [ -e xyz ] ; then
echo file xyz exists
else
    if [ ! -d x ] ; then
        echo x is not a directory
```

```

fi
if [ -x y ] ; then
    echo y can be executed
fi
fi

```

?\$

כל פקודה UNIX שה-shell מבצעת מחזירה (עם סימנה) ל-shell אינדיקטור המציין הצלחה או כישלון.

ההגדרה של הצלחה או כישלון תלויות בפקודה.  
ה-shell דואגת להציג במשתנה פנימי ? ערך של 0 (הצלחה) או 1 (כישלון).  
ערךו של המשתנה ? יכול לסייע לנו על מידת ההצלחה של הפקודה الأخيرة.

for

```

for variable in list-of-arguments ; do
    commands
done

```

פקודות ב-commands יבוצעו כמספר האיברים בראשימה. אל הערך הנוכחי של המשתנה הלולאה ניתן להתייחס באמצעות סימן \$variable.

```

% cat for
#!/bin/bash
for i in A B D ; do
    echo $i
    date +%$i
done
% for
A
Monday
B
February
D
02/08/99

```

while

ולאota : while

```
while condition ; do
    commands
done
```

התנאי יכול להיות כל תנאי כמו שהוגדר בפקודת if

דוגמא:

```
num=5
while [ $num -lt 10 ] ; do
    echo "hi"
    num=`expr $num + 1`
done
```

## מערכות

ניתן להגדיר ב-bash מערכים חד ממדיים בצורה:

```
myarray=(arg1 arg2 arg3 ...)
```

האיברים ממושפרים החל מ-0.

האיבר ה-n בראשימה:

```
 ${list[n]}
```

כל האיברים בראשימה:

```
 ${list[@]}

% ar1=(A B C)
% echo ${ar1[@]}
A B C
% echo ${ar1[2]}
C
% ar2=(${ar1[@]} ${ar1[@]})
% echo ${ar2[@]}
A B C A B C
```

## שורט הפקודה

ניתן לקרוא ל-script של bash עם פרמטרים בשורת הפקודה: command line arguments

למשתנים אלו ניתן לגשת דרך המשתנים המיקומיים (positional variables)

...,\$2,\$1\$

המשתנה \$# מכיל את מספר הפרמטרים בשורת הפקודה.

\* הינו כל הפרמטרים שהועברו ל-script.

```
#!/bin/bash
i=0
myarray=( $* )
while [ $i -lt $# ] ; do
    echo ${myarray[$i]}
    i=`expr $i + 1`
done
echo $i
```

## цитוט - quoting

כפי שראינו עד עתה ה-shell מתייחסת אל תווים מסוימים כנתוני הוראות עבורה (&, \*, ?, \$, ).

לעתים אנו מעוניינים למנוע מה-shell את ההתייחסות המיוחדת אל תווים הוראות.

התווים '-' - כאשר הם מופיעים בזוגות בשורת הפקודה מורים ל-shell להתייחס לתווים דלעיל כתווים רגילים.

הבדל בין השניים הוא שבגרשיים הכפולים עדין תבוצע הערכה (evaluation) של משתנים.

```
% x=hello
% echo "what is the value of $x ?"
what is the value of hello ?
% echo 'what is the value of $x '
what is the value of $x
```

הטו \' משמש לציטוט מירידי של התו הבא אחריו. אין צורך בגרשיים \*

```
<pre dir=ltr>
% echo what is this \?
what is this ?
```

## דוגמאות לארשויים

```
% d = "abc"
% echo "$d is a nice string"
abc is a nice string
% echo '$d is a nice string'
$d is a nice string
% echo "`date +%A` is a nice day"
Monday is a nice day
```

## משמעותים מספריים

אם רצאים משתנה מסווני

```
let a = 1
```

ואז ניתן לבצע:

```
let a=$a+1
```

כלומר ב-a יש כרגע הערך 2.

## פונקציות ב-shell

פונקציה נראית כך: ( $\$1$  ו $\$2$  הם הפרמטרים שמועברים לפונקציה)

```
#!/bin/bash

function check {
    grep $1 $2
}

# The next line will search for the string hello in file file.c cause in the
# function it uses grep
check hello file.c
```

## תכנות Shell - תרגול

### תרגיל 1

מה יהיה פלט הרצה של קבצי ה script הבאים:

1. myghaz 1 >cat shc1

```
/ ! # bin/ sh
for i in a b c; do
    echo $i
```

done

**הפלט:**

2. myghaz 3 >cat shc2

```
/ ! #bin/ sh
for i in `ls klum`; do
    echo $i
done
```

**הרצה:**

myghaz 4 >shc2

klum1

klum2

klum3

3. myghaz 5 >cat shc3

```
/ ! #bin/ sh
for i in `ls klum`; do
    echo nothing$i
done
```

**הפלט:**

4. myghaz 7 >cat shc4

```
/ ! #bin/ sh
a=nothing
for i in `ls klum`; do
    echo $a$i
done
```

הפלט:

5. / ! #bin/ sh

```
for i in `ls klum`; do
    if [ $i != klum1 ]; then
        echo $i
    fi
done
```

הפלט:

6. myghaz 15 >cat shc8

```
/ ! #bin/ sh
for i in `ls `; do
    if [ -d $i ]; then
        echo $i
    fi
done
```

הפלט:

7. myghaz 17 >cat shc9

```
/ ! #bin/ sh
for i in `ls klum`; do
    if [ $i = klum1 -o $i = klum2 ]; then
        echo $i
    else
        echo abc$i
    fi
done
```

הפלט:

## 8. myghaz 21 &gt;cat shc11

```
/ ! #bin/ sh
i=0
while [ $i -lt 5 ]; do
    echo $i
    i=`expr $i + 1`
done
```

הפלט:

## 9. myghaz 23 &gt;cat shc12

```
/ ! #bin/ sh
func ()
(echo abc)
func
הפלט:
myghaz 24 >shc12
abc
```

## 10. myghaz 25 &gt;cat shc13

```
/ ! #bin/ sh
i=0
while [ $i -lt 5 ]; do
    i=`expr $i + 1`
    if [ $i -eq 3 ]; then
        continue
    else
        echo $i
    fi
done
```

הפלט:

## 11. myghaz 27 &gt;cat shc14

```
/ ! #bin/ sh
i=0
while [ $i -lt 5 ]; do
    i=`expr $i + 1`
    if [ $i -eq 3 ]; then
        break
    else
        echo $i
    fi
done
echo abc
```

## הפלט

## 12. myghaz 29 &gt;cat shc15

```
/ ! #bin/ sh
i=0
while [ $i -lt 5 ]; do
    i=`expr $i + 1`
    if [ $i -eq 3 ]; then
        exit
    else
        echo $i
    fi
done
echo abc
```

#-\$# מס' הארגומנטים בשורת הפקודה  
 \$0 - שם התוכנית  
 \$1-\$9 - הארגומנטים בשורת הפקודה  
 \*-\$\* - כל הארגומנטים  
 \$\$ - מס'IDI של ה script שרצ בז"ט

### סיכום פקודות בלינוקס

יצירת ספרייה	Mkdir m1
יצירת קובץ	Touch t1
עריכת קובץ	Pico m1
מציג רשימת קבצים בתיקיה הנוכחיית	Ls
מציג מידע נוסף על הקבצים	Ls -l
עליה/ירוד בורגת התקיות עליה לראשי	Cd
עליה ספירה אחורית וירוד לm1	Cd ~
מעתיק את קובץ 1 מולהעתק קורה m2	Cp m1 m2
מעתיק את m1 לקובץ באותו שם בתיקיה הנוכחית Friends	Cp ./m1 .
מעתיק את כל הקבצים שבתיקה למספרה הנוכחית	Cp friends/* .
העברת קבצים בדוגמא זו שינוי השם לקובץ	Mv a helio
מציג את המדריך/ הספריה בה עובדים	Pwd
מחיקת ספריה	Rmdir Friends
מחיקת קובץ	Rm friend1
ביטול מחיקת קובץ	Unrm friend1
ניקוי סל המיחוזר	Clean Trashcan
מחיקה טוטאלית של הקובץ	\rm friend1
מחיקת הספריה וכל הדברים שבתוכה(ע"ז)	Rm -r Friends
הצאת טופט על המסך	Cat f1
מדפסה ומאפשרת דפונף	More f1
מדפסה חלון אחד ומאפשרת דפונף	Less f1
מציג מס' שורות מהתחלת הקובץ	Head f1
מציג 2 שורות מהתחלת הקובץ	Head -2 f1
מציג מס' שורות מסוף הקובץ	Tail f1
מציג 2 שורות מסוף הקובץ	Tail -2 f1
הכנסה של שמות הקבצים לקובץ שמו f1	Ls -> f1 1
כנ"ל שרשור לתרכן שבקובץ f1	Ls -l >>f1 1
קלט	Ls -l <f1 1
2 הפקודות מתחדשות בו זמנית הצגה ואיישור דפונף	Ls -l   more
אפשר כל ההרשאות לכולם	Chmod 777 f1 1
מוסיף להרשאה דראשונה את כל האפשרויות	Chmod +xwr f1 1
מוסיף ל's others' כל קריאה אפשרי ומוריד מהקובוצה את הרשות כתיבה	Chmod o+r,g-w f1 1
מוסיף לקובוצה וככל שימושים כתיבה והרצה ומוריד מאחרים ריצה	Chmod ug+wx,o-x f1 1
מוסיף לכלום הרישאת ריצה (משויזים להוסיף לכלום הרישאת כתיבה לא מספיק w+a אלא צריך לבתוב במפורש לכל סוג )	Chmod a+x f1 1
מידע על עצמי	whoami
קובץ עזרה	Man pwd
	VII
הוספת טקסט מיילין לסמן	a
מעבר לסוף השורה	A
תחילת שורה	I
הוספת טקסט כמקומ הסמן	i

מוסיף שורה מעל	O
מוסיף שורה מתחת	o
שמירת קובץ והמשך עבודה	(Escape + ) :w
שמירה ויציאה	(Escape + ) :wq
יציאה ללא שמירה	(Escape + ) :q!
יצירת גיבוי לקובץ (חדש בלבד)	(Escape + ) :w f22
להחליף קובץ קיים יחליף עריכים	(Escape + ) :w! f22
תווזה של תוו בודד ימינה	(Escape + ) i
" " שמאליה	(Escape + ) h
" " למעלה	(Escape + ) k
" " למטה	(Escape + ) j
מרץ חזי מסך למעלה	(Escape +cntrl + ) u
מרץ חזי מסך למטה	(Escape +cntrl + ) d
מרץ מסך שלם קדימה	(Escape +cntrl + ) f
מרץ מסך שלם לאחוריה	(Escape +cntrl + ) b
יעביר את הסמן לתחילת השורה השלישית	(Escape +: + ) 3
מגיע לסוף הקובץ	(Escape +: + ) \$
מביא לסוף השורה	(Escape +: + ) \$
מחיקת האות שהסמן עומד עליה	(Escape + ) x
מוחק את המילה שעומדת עליה	(Escape + ) dw
מוחק שורה שלימה	(Escape + ) dd
מוחק עד סוף השורה	(Escape + ) d\$
מוחק עד תחילת השורה	(Escape + ) ^d
מחזיר את כל השינויים שעשינו בשורה הנוכחית	(Escape + ) U
מחזיר פעולה אחרונה שעשינו	(Escape + ) u
מבטל החרזרת פעולה אחרונה (בצע שוב)	(Escape + ) .
צירוף שורות בצד זו זהה	(Escape + ) J
חירוש מהירות ראשונה של המילה הנתונה החל מהסמן לפני מטה	(Escape + ) /Jerusalem
רצ קדימה לחירוש המילה הבאה	(Escape + shift + ) n
רצ אחורה לחירוש המילה הבאה	(Escape + shift + ) N
חירוש מהירות ראשונה של המילה הנתונה החל מהסמן לפני מעלה - אחורה	(Escape + ) ?Jerusalem
פקודה למספר הקובץ	(Escape + ) :set number
העתקת האות שהסמן עומד עליה	(Escape + ) yl
העתקת המילה שעומדת עליה	(Escape + ) yw
מעתיק שורה שלימה	(Escape + ) Y
מעתיק עד סוף הישורה	(Escape + ) y\$
מעתיק עד סוף הפסקה	(Escape + ) y}
מדביק העתקה מיימין לטמן	(Escape + ) p
מדביק העתקה משמאלי לטמן	(Escape + ) P
מעתיק את המילה לאוגר מס 3	(Escape +shift + ) '3yw'
הדבקה של התוכן שנמצא באוגר 3	(Escape +shift + ) '3p'
החלפת תוו במקום הסמן	C1
دورס מילה נוכחית	Cw
دورס שורה שלימה	Cc
מחליף עד סוף השורה	C\$
פעם אחת מחליף את מהירות hello world מהסמן	Escape +:s+/hello/world

unit - 52

פעט אחת מחליפה את מחרוזות hello בחרוזות world את המופיע הראשון שבין השורות 5 עד 15	Escape +:5,15 s/hello/world
כנ"ל חיפוש בכ' כל המסקן החלפה של המר פעים הראשונים בשורה	Escape +:% s/hello/world
החלפה תבוצע בטוויה השירות שבין 10 ל-20	Escape +: s/hello/world
החלפה של כל המסקן תבוצע הלהפה מבקשת משורה 10 עד סוף הקובץ	Escape +:10,20g/hello/s//world/g
הדרת שם לפקודה השורה Dir תציג את הקבצים בספרייה הנוכחית hello world shalom החקוב-ps	Alias dir "ls -l"
מחיקת אליאס	Unalias shalom
מציג רשימה תזהליכים ומזהום window-ps	ps
מציג את כל מה שרשן בזאתן-ps	ps -W
מציג תזהליכים ישל משתמשים אחרים-ps	ps -a
מציג גם תזהליכים ללא מטר-ps	ps -x
מציג גם שם משתמש-ps	ps -f
סיכום מלא על הרתולין הנוכחי-ps	ps -s
יצירת job מנמיה פקדות	Ls -ja   sort > /tmp/delme &
מציג את כל ה澤ף-ps	Jobs
כנ"ל כולל מס תזהליכים-ps	-jobs
פריט בטבלון מחוברת עם 25-ps	Wc file
מציג כמה שירותים, מילימ', הווים יש בקובץ file	

new fls dels 27.1

new fls dels 27.1

fls new file 27.1

6 fls new file 27.1

27.1

א. קולגא.

grep avid \*

ימצא את השורות בהן מופיע avid

4. רצף סימנים בסוגרים מרובעים [] מתאים לטו אחד מתוכם. אם הסימן ^ מופיע ראשון בתוך הסוגרים המרובעים התבונתי בהתאם לכל תו פרט לשאר התווים בתוך ה- []. הסימן "-" (פינוס) בין שני תווים מציין רצף תווים בין שני התווים שמלצתיו.  
דוגמאות:

grep [dr]avid \*

يُعْثَلُ أَدِيْدَ [دَرِيْدَ]

grep [1-9][0-9]4 \*

يُعْثَلُ أَدِيْمَ [١٠٩٤]

כאשר הראשונה אינה () והאחרונה היא \*

grep ^[A-Z-A-Z]

يُعْثَلُ كُلُّ الْحُشُورَاتِ شَلَّاءً مَتَّهِلِّيَّةً [أَدِيْدَ - آدِيْدَ]

5. אחרי כל אחד מהן"ל יכול להבא **קוברייה** "י"י וביתוי החדש מתאים לרצף של אפס או יותר מופיעים של מה שבא לפני הcovariation. יש לשים את הביטוי במרקאות כדי לא לבבל את ה- shell  
דוגמאות:

grep "d" \*

ימצא את השורות בהן מופיע: d (אָדָה, כְּדָה, אַדְוִינְדָה (אֲדָוִינְדָה), אַדְוִינְדָה)

grep "[0-9]\*4" \*

ימצא את כל השורות שבחן מופיעים: 4 (אָדָה, כְּדָה, אַדְוִינְדָה (אֲדָוִינְדָה), אַדְוִינְדָה)

6. אם אחרי כל אחד מהbeitotiyim 1-5 באים סוגרים מסוללים עם מספר בתוכם, הביטוי החדש יחש את מה שיש לפני הסוגרים המסוללים מספר פעמים כמו שרשים בתוך הסוגרים המסוללים. צריך להקדים את הסוגרים עם \backslash, כפישמו בדוגמה למטה.  
דוגמאות:

grep "meet" \*

ימצא את "meet" אבל לא את "met"

grep "\{2\}aoeui" \*

ימצא את כל השורות המתחלות באות "m"

7. נתן לשים ביטוי רגולרי בסוגרים כדי להתייחס אליהם בהמשך. גם כאן צריך להקדים את הסוגרים ב- \backslash. ההתייחסות למה שמצאנו בסוגרים היא על ידי \1. הסבר בדוגמא.

דוגמאות: grep "\{1\}aoeui[\aoeui]\{1\}" papa

ימצא מילים מסווג

,"papa",

כלומר כאשר בחן הטרוף "עיזור-תנוועה" פעמיים. לא יתאימו למשל מילים כמו "sota" שכן הטרוף הראשון והשני שונים, ta != so

8. סימנים מיוחדים "<" ו- ">" מתאים לתחילת מילה ולסיומה בהתאם.

דוגמאות:

grep "&lt;[^a-zA-Z]\*&gt;" \*

ימצא את כל השורות שיש בהן מילים

אַהֲרֹן אֶלְקָנָה אֶלְקָנָה

אַהֲרֹן אֶלְקָנָה אֶלְקָנָה  
 אַהֲרֹן אֶלְקָנָה אֶלְקָנָה  
 אַהֲרֹן אֶלְקָנָה אֶלְקָנָה

VI הוא עורך טקסט בלינוקס.

### שני מצבים של עבודה

בניגוד לרובית העורכים בעורך וקיימים שני מצבים של עבודה:  
**מצב עריכה** - במצב זה כל תו שמקש מתווסף לקובץ בהתאם למיקום הסמן (כמו בעורכים emacs וword).

**מצב פקודה** - במצב זה, כל תו שמקלד מתפרש ע"י העורך כפקודה או חלק ממנו. כפי שנראה בהמשך, קיימות פקודות רבות ו שונות. חלקן מורכבות מתו אחד בלבד, וחלקן מורכבות מספר תוים או אף ממשפטים שלמים.  
 דוגמא להמחשה:

הקשה על התו x כאשר העורך במצב עריקה תוסיף את התו x לטקסט במיקום בו הסמן נמצא.

לעומת זאת, הקשה על התו x כאשר העורך נמצא במצב פקודה תמחק את התו הנוכחי, במקרה זה x פועל לפי תפקידו בראשית הפקודות של העורך שהיא מחיקת אותו.  
 בשיעורים הבאים נלמד את הפקודות השונות של העורך שיישמשו אותנו בעבודתנו עם העורך.

שים לב: לא לכל התווים יש משמעות במצב פקודה אולם הקשה על то ללא שימוש במצב פקודה לא תשפיע על הקובץ.

### פקודות בסיסיות

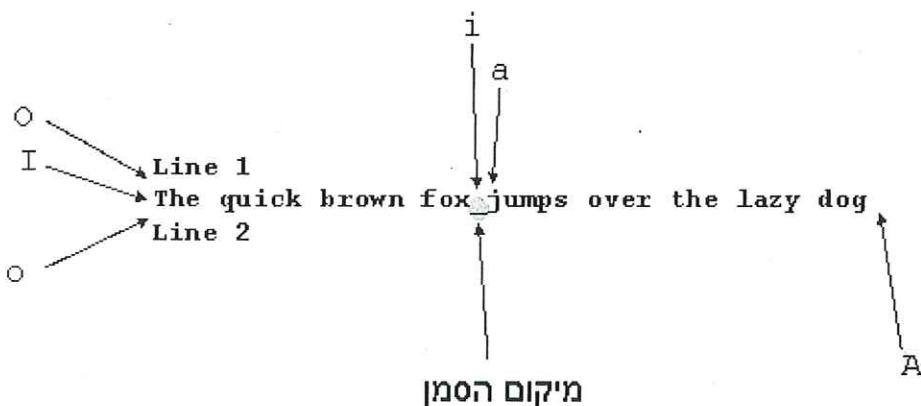
כפי שראינו קיימים שני מצבים של עבודה ב:VI -  
 מצב עריקה ומצב פקודה.

במצב פקודה ניתן לנוט בקובץ באמצעות הקשה על מקשי החיצים) לא ניתן לעשות זאת במצב עריקה, ובד"כ נקבל תוים לא מובנים על המסך אם נעשה כן).  
 בעיר CAN כי במצב פקודה לא ניתן לכתוב טקסט, וכי כל הקשה על מקש נחשבת ע"י העורך כפקודה או חלק منها.

מעבר למצב פקודה מתבצע ע"י הקשה על מקש ה.e - Escape  
מעבר למצב עריקה מתבצע ע"י הקשה על אחד המKeySpecים הבאים, לפי ה蟲ר:

תו	משמעות
a	הוספת טקסט מימין למיקום הסמן
A	הוספת טקסט בסוף השורה בה נמצא הסמן
i	הוספת טקסט במקום בו נמצא הסמן
I	הוספת טקסט בתחילת השורה בה נמצא הסמן
o	הוספת טקסט תוך ייצרת שורה חדשה מתחת לשורה בה נמצא הסמן
O	הוספת טקסט תוך ייצרת שורה חדשה מעל השורה בה נמצא הסמן

החייבים מסמנים את מקום הוויה, בהתאם לאופציה שבסירה  
(שאר הטקסט יושם בהתאם)



### שימוש לבן!

- א. פקודות ב-VI-הן case sensitive. פיקודים יוצרים ערך מסוים.
- ב. בתחילת העבודה, העורך נמצא במצב פקודה.

### עבודה עם קבצים

בשיעור הראשון והשני למדנו על שני מצבים העבודה של העורך ו כיצד לעبور ממצב אחד לשני . בשיעור זה נלמד פעולות בסיסיות של פתיחה שמירה ויציאה מקובץ.VI

פתיחה קובץ:

על מנת לפתח קובץ חדש (או לחילופין, קובץ קיימ), נקליד "vi" ואת שם הקובץ.  
לדוגמה, הפקודה:  
vi my\_file.txt

תפתח קובץ בשם my\_file.txt מהתקיה הנוכחית. אם הקובץ קיים כבר, תוכנו יוצג על-גבי המסר. אם הקובץ אינם קיימים, על המסר יופיע "קובץ ריק".

כעת אנו יכולים לעורר את הקובץ.

על מנת לשמור את הקובץ עליו לעבור למצב פקודה, ולהשתמש באחת האפשרויות הבאות:

פקודה	משמעות	ឧ. קובץ.txt
w	שמירה והמשך עבודה	W ;
wq	שמירה ויציאה מהעורך	Wq ;
q!	יציאה ללא שמירת השינויים	! ;
w <filename>	שמירת הקובץ תחת שם אחר והמשך עבודה	W <filename>

### שאלה מס' 1

1. פתח קובץ חדש בשם tar11.
2. הקלד את שם משפחתך. (שים לב: עליך לעבור למצב הוסף כדי לבצע הקלדה).
3. עברו למצב פקודה.
4. הוסיף את שמו הפרטי לפני שם משפחתך. (או בקיצור בתחלת השורה).
5. צא מהקובץ כולל שמירה.
6. פתח את הקובץ tar11 לעובדה.
7. הוסיף בשורה חדשה את העיר בה אתה מתגורר. (השתמש באופציה לפתיחת שורה חדשה).
8. הוסיף בשורה חדשה בין שתי השורות את כתובות מגוריך. (כנ"ל רק שהפעם השורה החדשה תהיה מעל השורה שאנו נמצא בה כעת).
9. צא מהקובץ tar11 עם שמירה.

### שאלה מס' 2

1. פתח קובץ חדש בשם tar12.
2. הקלד את שמו הפרטי.
3. שומר על השינויים ללא יציאה מהקובץ.
4. הוסיף בשורה חדשה את שם משפחתך.
5. צא מהקובץ ללא שמירת השינויים.
6. פתח את הקובץ tar12 לעובדה.
7. שומר את השינויים בקובץ חדש tar13.
8. צא מהקובץ tar12.
9. פתח את הקובץ tar13.
10. צא מהקובץ tar13.

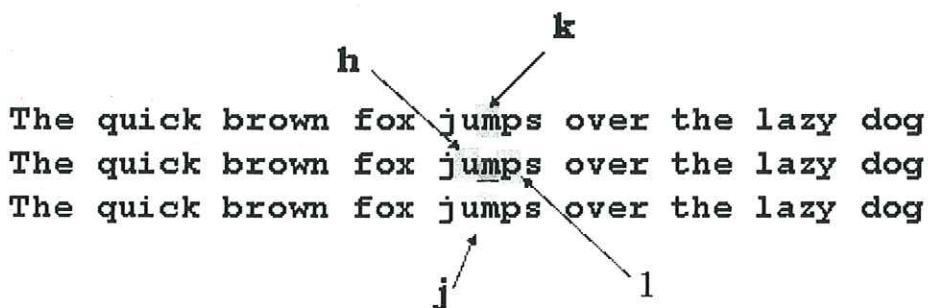
### שאלה מס' 3

1. פתח את הקובץ tar11 לעובדה.
2. הוסיף בסוף השורה הראשונה שתי כוכיות.
3. שומר את תוכן הקובץ בקובץ tar13. (שים לב tar13: כבר קיימ). (שים לב tar13: כבר קיימ).
4. צא מהקובץ tar11 ללא שמירת הנתונים.
5. פתח את הקובץ tar11 לעובדה וצא ממנו.
6. פתח את הקובץ tar13 לעובדה וצא ממנו.

## ניטוט בקוביץ

פעולה חשובה בעורף היא תזוזה על פני הטקסט או דילוג לחלקים שונים במסמך.  
לעתים נדרש להציג את הסמן על פני הטקסט קדימה או אחורית למעלה או למטה בצורה פשוטה. לעתים נדרש לנוע קדימה או אחורית על פני המסמן בתזוזות של מסך או של חצי מסך בכל פעולה. ולעתים נדרש לדלג לשורה ספציפית במסמך.  
תזוזה של TWO בקובץ מתבצעת במצב פקודה ע"י התווים I, k, j, h -

I	new_enew גולגולת של ח'ו גולגולת וואי וו'
h	new_enew גולגולת של ח'ו גולגולת וואי וו'
k	new_enew גולגולת של וואי גולגולת וואי וו'
j	new_enew גולגולת של וואי גולגולת וואי וו'



- מיקום נוכחי של הסמן
- מיקום הסמן לאחר הקשה על המקש המצוין

## שימוש לבן

השימוש בחצים שבמקלדת אפשרי ברוב המקלרים אך מוטב להתרgal להשתמש בתווים הנ"ל, שאפשרי תמיד.  
תזוזה בקובץ אינה מעבירה את העורף למצב הוספה אלא פשוט מאפשר טויל בקובץ.

תנוועה על המסך **u**, **v**, **f**, **ctrl**, **i**, **d**, **s** - אוא

b	הפל עגי נוק צפוי נסיגת
d	הפל עגי נוק צפוי נאיגת
f	הפל נוק צפודזינת
s	הפל נוק צפודזינת

### דילוג לשורה ספציפית בקובץ

במצב פקודה יש להקיש את התו נקודותים (:), ולאחריהם (בצמוד) את מספר השורה.

לדוגמא: 3:

יעביר את הסמן לתחילת השורה השלישית בקובץ.

### פקודות נוספת לדילוג

\$	בגיא גולפֿר אַמְּפּוֹרָה גִּילָּעָן
\$	נאנו גוֹלֵף גַּלְעֵד עֲרָכָתִין

שימוש לב!

אחרי תרגול קצר, התזוזה בקובץ בעזרת הפקודות הופכת להיות נוחה מאוד.

### מחיקת טקסט

את הפעולות הנפוצות בעורך הנה מחיקה של טקסט.

כפי שראינו, בעורך או ביצוע פעולות מתבצע במצב פקודה. לכן, כדי למחוק טקסט, נعبر

תחילה במצב פקודה(באמצעות המקש escape - )

וב-קיימות פקודות שונות עבור צורות מחיקה שונות, החל ממחיקת אחת וכליה  
במחיקת מספר שורות.

שימוש לב!

מחיקת טקסט ב, או-מתבצעת בעיקר ע"י התו d במצב פקודה.

להלן התווים המשמשים לפקודות המחקה השונות :

x	מחיקת כל אות ערך
dw	מחיקת כל אות בסורס ו-0ף נאיגט ערך
dd	מחיקת כל אות ערך
d\$	מחיקת כל אות בסורס ו-0ף גולף כל אות
p^	מחיקת כל אות בסורס ו-גלויגן פאלס

The quick brown fox jumps over the lazy dog

- x The quick brown fox jumps over the lazy dog
- dw The quick brown fox jumps over the lazy dog
- dd The quick brown fox jumps over the lazy dog
- d\$ The quick brown fox jumps over the lazy dog
- <sup>p</sup>d The quick brown fox jumps over the lazy dog

- מיקום וכוחו של הסמן

- מסמן את האותיות שתמחקנו לאחר הקשה על המKeySpec המתאימים

עיר כאן כי ניתן להצמיד לפקודת המחייב גם מספר, שיצין את מספר הפעמים שברצוננו לבצע את הפקודה.

לדוגמא:

8x

יבצע את פקודת "מחיקת האות הנוכחית" שモנה פעמים, כך שלמעשה נמחק 8 אותיות החל מהאות הנוכחית.

דוגמא נוספת:

3ddp

ימחק 3 שורות מהקובץ.

נקדים את המאוחר, ונעיר כאן כי לרוב הפקודות בוו-ניתן להצמיד מספר, שיצין את מספר הפעמים שנרצה לבצע את הפקודה.

שאלה מס' 4

1. פתח קובץ חדש בשם tar21.

2. הקlid את המשפט הבא :

Tzo bzzzze, zzz zor znot zz zzzzto bzzzze

3. שומר את התוכן בקובץ חדש tar22.

4. מחק את כל המופעים של האות z בצורה היילה ביותר תוך שימוש יעיל בפקודות המחייבת שנלמדו (כולל מספר בפקודה) ותוך שימוש במקרים התנועה.

5. בטל את כל המחוקות שביצעת בצורה היילה באמצעות פקודה אחת.

6. מחק את השורה .

7. בטל את מחיקת השורה .

8. צא מהקובץ tar21 עם שמירה .

שאלה מס' 5

בשאלה זו ננסה להקליד את המשפט:

"Summer has always been my favorite time of year"

tar23. 1. פתח קובץ חדש בשם

2. הקlid את המשפט הבא כפי שהוא מופיע (במספר שורות):

Summer

has has

has has

has

has

always

been

my my my my favorite timmmmmme of year

3. עברו למצב פקודה .

4. עברו לשורה 5 .

5. עברו לשורה ראשונה .

6. מחק את השורות את המיללים ואת האותיות המיותרות, כל זאת בצורה היילה תוך שימוש בפקודה מתאימה ותוך שימוש באופציית המספר (לדוגמא – dw2: פקודה למחיקת 2 מילים ).

7. צרף את חלקו המשפט לשורה אחת .

8. צא מהקובץ tar23 עם שמירה .

### חרטה, חזקה וצירוף

חרטה

השורהoriginally appears as	n
השורהoriginally appears as	נ

דוגמא:

השורה המקורית:

The quick brown fox jumps over the dog

השורה לאחר שינויים (השינויים בוצעו משמאל לימין, מבלי לעבור לשורה אחרת):

The quick brown fox jumps over the lazy dog

השלטי האחורני

השורה לאחר ביצוען (ביטול שינוי אחרון):

The quick brown fox jumps over the dog

השורה לאחר ביצוען (ביטול כל השינויים):

The quick brown fox jumps over the dog

### חזקה

על-מנת לחזור על ביצוע הפעולה الأخيرة יש להקיש על התו '' (נקודה).

צרף שורות

על-מנת לחבר את השורה הנוכחית לזה שאחריה, יש להקיש על התו ' '

לפני הקשת ל

The quick brown fox  
jumps over the lazy dog

אחרי הקשת ל (שים לב שהמילים fox - jumps מחוברות)

The quick brown foxjumps over the lazy dog

## חיפוש טקסט

חיפוש טקסט יבוצע ע"י הקשה על התו 'י', ובצמוד אליו את המחרוזת לחיפוש (במצב פקודה, כמוובן)

דוגמא:

Jerusalem

חיפוש המופיע הראשון של המחרוזת Jerusalem בקובץ ודילוג אליה. החיפוש יתחיל מהמייקום הנוכחי של הסמן.

כיצד נועור למופעים הבאים של המחרוזת בטקסט?

n צייר מאום גאנז אונטוף (גניעין נהג. צייר - אין גאנז)

N צייר מאום גאנז אונטוף (גניעין נהג. צייר - נאנז)

(גאנז)

אם נרצה בכך, יוכל להתחילה מיד בחיפוש בכיוון "אחרה", וזאת ע"י שימוש בתו 'י' במקום בתו 'ו'.

דוגמא:

Jerusalem

ידלג למופיע הבא של המילה Jerusalem משמאלי לסמן (כלומר - חיפוש בכיוון "אחרה").

## העתיקת טקסט

העתיקת טקסט מtbody בערך ע"י התו y ופקודות העתקה מתבצעות במצב פקודה. כדי שלמדנו בשיעור על מחיקת טקסט גם להעתיקת טקסט קיימות פקודות שונות למקומות השונים- העתקת אות מילה או שורה. כמו כן ניתן לציין מספר שיקבע את מספר הפעמים שהפעולה תתבצע.

להלן התווים שימושיים לפקודות ההעתיקה השונות:

y	אַלְקָעֵן אֶלְאָמֵן עֲרִיכָה
yw	אַלְקָעֵן אֶלְגָּה עֲרִיכָה
Y	אַלְקָעֵן אֶלְפָּה עֲרִיכָה
y\$	אַלְקָעֵן נְסָאָן בְּזֹאת אֶלְפָּה עֲרִיכָה
y	אַלְקָעֵן בְּזֹאת אֶלְפָּה עֲרִיכָה

(ניתן להעתיק טקסט לאוגר ספציאלי וכן לשמר באוגרים שונים מספר העתקות במקביל.)

## הדבקת טקסט

התו k משמש להדבקת טקסט:

d גַּדְקָה אַיִלְמָאָה גַּוְן

P גַּדְקָה נְנָאָגָה גַּנְדָּה גַּוְן

הפעלת הפוקודה תדביק את הטקסט האחרון שנשמר באוגר - זה שהעתק אחרון או זה שנמחק אחרון, על ידי פוקודת העתקה או המחיקה של העורף שראינו בשיעורים הקודמים. העורף ו' מאפשר גם פניה לאוגר ספציפי והעתקה של טקסט אליו או פניה לאוגר ספציפי והדבקה של הטקסט שמאוחסן באותו אוגר.

התחבר הוא פניה לאוגר ע"י גרשים ומספר האוגר ולאחר מכן כתיבת פוקודת העתקה או הדבקה.

לדוגמא:

הפוקודה הבאה (במצב פוקודה - )"9wy - משמשת להעתקה של המילה הנוכחית לטור אוגר מסטר 9.

הפוקודה - "9k פירושה הדבקה של התוכן שנמצא באוגר מסטר 9 (המילה שהעתקה בפוקודה הקודמת).

בעזרת השימוש באוגרים אנו יכולים לשמור קטעי טקסט שנזדקק להם מאוחר יותר באוגרים ספציפיים וליעל את העבודה.

### **החלפת טקסט**

החלפת טקסט מתבצעת במצב פוקודה ופעלת על הטקסט החל מקומו של הסמן. שימוש לבן

פוקודות ה החלפה הנפוצות עושות שימוש בתו C עם תוספות ספציפיות .

עם הפעלת הפוקודה מסומנת נקודת הסיום של ה החלפה בתו '\$' שיופיע על המסר. חשוב לציין!

פוקודות ה החלפה מעבירות את העורף למצב עrica ולקיש escape בסיום ה החלפה.

להלן התווים שימושיים לפוקודות ה החלפה השונות :

cl	העתק איזון ערכיהם
cw	העתק איזג איזק איזק איזג איזג
cc	העתק איזוואת ערכיהם
c\$	העתק איזק איזג בז גוזג איזוואת

### **חישוב והחלפה פשוטים**

חישוב והחלפה חד פעמיים במצב פוקודה, יש להקיש : s ומיד אחר-כך את התו ' ', את המחרוזת להחלפה, שוב את התו ' ' ואת המחרוזת החדשה.

ניתן גם להגביל את טווח הפעולה, כך שפעולות ה החלפה תתבצע רק בעטוט מסוים (עדין), רק פעם אחת). עושים זאת ע"י הקלדת מספרי השורות בינהן נרצה לבצע את ה החלפה, כשהן מופרדות בפסיק, מיד לאחר פוקודת ה החלפה .

דוגמא:

s/Jerusalem/Tel-Aviv: 5,15:  
תחליף את המופיע הראשון של המחרוזת Jerusalem בחרוזת Tel-Aviv בטווח השורות 5-15 בקובץ הנתון.

חיפוש והחלפה של כל המופעים של מחרוזת מסוימת בחיפוש והחלפה אלה, נרצה להחליף את כל המופעים של מחרוזת מסוימת במחרוזת אחרת, על פני כל המספר או בקטע ממנו.

הפורמט הוא מעט יותר מסובך:

g/string\_to\_replace/s//string\_to\_replace\_with/g:  
כלומר, "g":Nomיד אחר כר לוכסן, מחרוזת להחלפה, לוכסן, s, שני לוכסנים ומחרוזת "המחליפה".

גם כאן, ניתן להגיד טווח שורות בו תבוצע החלפה, וזאת ע"י הוספת "/g" בסוף הפקודה, ומספריו השורות על-פניהם יש לבצע את החלפה.

דוגמא:

g/Jerusalem/s//Haifa/g10,20:  
החלפה תתבצע אך ורק בטווח השורות 10-20.

הערה:

ניתן גם להשתמש בתו '\$', לציין השורה האחורונה בקובץ, כר שהפקודה הבאה:  
\$,g/Jerusalem/s//Haifa/g10:  
תבצע את החלפה המבקשת החל משורה 10 ועד סוף הקובץ

שאלה מס' 6

1. פתח קובץ חדש בשם tar31.

2. הład את המשפט הבא לעורך כפי שהוא מופיע

(שים לב: קיימות מספר טוויות במשפט)

Despite his gud grieving

Father John wos

unable to console

the intensions woman

3. מיקום של המילים grieving ו-intensions הוחלף במשפט תקן את הטוות תור Shimush בפקודות חיפוש (למציאת המילה (intensions גזירה והדבקה למיקומה הנכון).

4. גזור והדבק את המילה grieving למיקומה הנכון במשפט.

5. קיימות שתי שגיאות כתיב במשפט במילה pad ובמילה wos. תקן את השגיאות.

6. ספור, תור Shimush בפקודת חיפוש כמה פעמים מופיעות אותן אאות במשפט.

7. צרע את המשפט לשורה אחת.

8. העתק והדבק את המשפט פעם נוספת לשורה חדשה.

9. צא מהקובץ tar31 עם שמירה.

שאלה מס' 7

1. תרגל חיפוש מחרוזת ומעבר קדימה ואחריה על פני המופעים של המחרוזת.  
2. תרגל שימוש בפקודות העתקה והדבקה תור Shimush בבאפרים.

linux-40

מחוזתיים  
אריתמטיים  
אופרטורים לוגיים  
אופרטורים עם קבועים

## אופרטורים מחוזתיים

= שווין

!= אי שווין

```
% cat if2
#!/bin/bash
x=file.c
if [ $x != file.c ] ; then
    echo $x is not file.c
else
    echo $x is file.c
fi
% cat if2
file.c is file.c
```

```
% cat if1
#!/bin/bash
x=file.c
if [ $x = file.c ] ; then
    echo $x is file.c
fi
% if1
file.c is file.c
```

$\$x \rightarrow$  מבחן  
!=

!=  
file

!=  
file

## אופרטורים אריתמטיים

-eq, -ne, שווה, ושויה בהתאם

-gt, -lt, גודל שווה, קטן שווה, גודל מ-, קטן מ- בהתאם

```
% cat if3
#!/bin/bash
x=6
if [ $x -gt 5 ] ; then
    echo $x is greater than 5
fi

% if3
6 is bigger than 5
```

אופרטורים לוגיים.  
!, תנאי לא לוגי

- t ! -  
-a, תנאי גם לוגי and
- o, תנאי או לוגי or  
-

זהים בתכליות לאופרטורים המקבילים בשפות תכנות.

```
#!/bin/bash
x=5
y=hello      and
if [ $x -lt 7 -a $y = hello ] ; then
    echo yes
else
    echo no
fi
% if4
yes
```

ה-shell מאפשרת לנו לבדוק תכונות של קבצים ובהתאם לתכונות לשנות את זרימת התוכנית.  
התכונות הן:

- e האם הקובץ קיים
- z האם למשתמש הרשות קריאה
- w האם למשתמש הרשות כתיבה
- x האם למשתמש הרשות ריצה
- p האם הקובץ הוא ספרייה
- f האם הקובץ הינו קובץ רגיל
- L סימבולי (link) האם הקובץ הינו קישור
- O האם המשתמש הוא בעל הקובץ

ישנם ארגומנטים נוספים אך לא נועבר על כולם  
נניח כי אנו בתיקיה מסוימת:

```
% ls -l
total 2
-rw----- 1 yedidia x
-rwxr-xr-x 1 yedidia y
#!/bin/bash
if [ -e xyz ] ; then
    echo file xyz exists
else not
    if [ ! -d x ] ; then
        echo x is not a directory
```

ק' יואן טען ג' 32 מ'

```

fi
if [ -x y ] ; then
    echo y can be executed
fi
fi

```

ש? ש? - קידם על האשורה ?\$

כל פקודה UNIX שה-shell מבצעת מחזירה (עם סימנה) ל-shell אינדיקטור המציין הצלחה או כישלון.

ההגדרה של הצלחה או כישלון תלויות בפקודה.  
ה-shell דואגת להציב המשתנה פנימי ? ערך של 0 (הצלחה) או 1 (כישלון).  
ערךו של המשתנה ? יכול לספר לנו על מידת ההצלחה של הפקודה الأخيرة.

for

```

for variable in list-of-arguments ; do
    commands
done

```

הפקודות ב-commands יבצעו כמספר האיברים בראשימה. אל הערך הנוכחי של המשתנה הלולאה ניתן להתייחס באמצעות סימן \$variable.

```

% cat for
#!/bin/bash
for i in A B D ; do
    echo $i
    date +%$i
done
for
A
Monday
B
February
D
02/08/99

```

היום אמצע  
יום אמצע  
לעומת

while

לולאת : while

white      מזמן

```
while condition ; do
    commands
done
```

התנאי יכול להיות כל תנאי כפי שהוגדר בפקודת if

דוגמא:

expr  
השווים איזה נושא

```
num=5
while [ $num -lt 10 ] ; do
    echo "hi"
    num=`expr $num + 1`
done
```

## מערכות

ניתן להציג ב-bash מערכות חד מימדיים בצורה:

```
myarray=(arg1 arg2 arg3 ...)
${myarray[n]}      פונקציית אינדקס
${myarray[@]}      קאcli האגף
```

האיברים ממושפרים החל מ-0.

האיבר ה-n בראשימה:

```
 ${list[n]}
```

כל האיברים בראשימה:

```
 ${list[@]}
% ar1=(A B C)      אוסף נסמן
% echo ${ar1[@]}      אוסף נסמן
A B C
% echo ${ar1[2]}      אוסף נסמן
C
% ar2=(${ar1[@]} ${ar1[@]})      אוסף נסמן
% echo ${ar2[@]}
A B C A B C
```

## שורת הפקודה

ניתן לקרוא ל-script של bash עם פרמטרים בשורת הפקודה:

፳፻፲፭ ዓ.ም

linux- 44

$$\$11 = 7$$

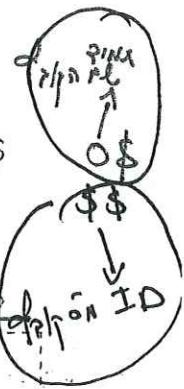
בז"ט

למשתנים אלו ניתן לחשוף דרך המשותפים המיקומיים (positional variables)

go to page 1-3 and write ~~go to page 1-3~~ ... , \$3, \$2, \$1 \$

המשתנה #**איל** את מספר הפרמטרים ששוכנת בהפניה

אנו יכולים כל הפקטורים שהועברו ל-script.



```

#!/bin/bash
i=0
myarray=( $* )
while [ $i -lt ${#myarray[@]} ]; do
    echo ${myarray[$i]}
    i=`expr $i + 1`
done
echo $i

```

## **цитוט - quoting**

כפי שראינו עד עתה ה-**shell** מתייחסת אל תווים מסוימים כנתוני הוראות עברות (&,\$,?,\*).

לעתים אנו מעוניינים למנוע מה-shell את התייחסות המוחדרת אל תמי הוראות.

התוים `|-` – כאשר הם מופיעים בזוגות בשורת הפקודה מורים ל-shell להתיחס לתווים דלעיל כתווים רגילים.

הבדל בין השניהם הוא שבגרשיים הכוונים עדין לתבצע הערכה (evaluation) של משתנים.

```
% x=hello  
% echo "what is the value of $x ?"  
what is the value of hello ?  
% echo 'what is the value of $x '  
what is the value of $x
```

1. "מִלְּפָנֶיךָ סְגִּנְךָ  
בְּרַכְתָּךְ נְסִינְךָ  
וְעַדְךָ"

הטו \ משמש לציטוט מיידי של הינו הכא אחרינו. אינו אורך ברכובים \*

```
<pre dir=ltr>
```

```
% echo what is this \?
```

לונדון גראן צ'רץ'

דוגמאות לארשיים

```
% d = "abc"
% echo "$d is a nice string"
abc is a nice string
% echo '$d is a nice string'
$d is a nice string
% echo "`date +%-A` is a nice day"
Monday is a nice day
```

## משתנים מספריים

values let let נוצרו מפה אם הוויה  
 let a = 1 פונקציית גוף  
 נוצרה דרכו זיהוי.

אם רצים משתנה מספרי:

let a=\$a+1

ואז ניתן לבצע:

כלומר ב-a יש כרגע הערך 2.

## פונקציות ב-shell

פונקציה נראית כך: (\$1 \$2 וכו' הם הפרמטרים שמועברים לפונקציה)

```
#!/bin/bash
function check {
    grep $1 $2
}
# The next line will search for the string hello in file file.c cause in the
function it uses grep .
check hello file.c
```

\$1=hello  
 \$2=file.c  
 פונקציה מקבלת שני פרמטרים  
 EX: hello file.c

## תכנות SHELL

### תרגילים

1. צרי קובץ הרצה script1 שיפלט:

- a. את רשימת הקבצים בספריה.
- b. את שם המשמש.
- c. את כל הקבצים שםם \*.file
- d. צרי 3 תיקיות A B C.

2. כתבי קובץ script2 המגדיר ALIAS לפקודות הבאות  
ALIAS>newFolder  
ALIAS>deleteFolder  
ALIAS>deleteAll  
ALIAS>copy  
ALIAS>exit

3. השתמשתי בפקודות הארגומנטים המיוחדים כדי לראות

- a. את מס' הארגומנטים בשורת הפקודה.
- b. שם הארגומנט הראשון.
- c. שמות כל הארגומנטים.
- d. מס' הAI של script שרצ.

### ולאלה

4. כתבי תוכנית המקבלת כקלט שם קובץ ומדפיסה הודעות על מספר המילים השורות  
והאותיות של הקובץ (WC). -טבלה אawyeh צוי טאל<ה, אלין, גיאן יט קזגף ← סזגף WC

כל נתון מוצג בשורה נפרדת עם הודעה מתאימה.

5. עברי על קבצים בתקיר(lolaha) והדפיסי את שמות הקבצים:

- e. הדפיסי את הקבצים אשר מכילים את שמן.
- f. חפשי קבצים הגדולים מ5KB.

g. הדפיסי את כל הקבצים השונים בשם קובץ TAR1.

h. הציגי את תוכן קובץ TAR1 בכרוף HELLO.

### פעולות אוניברסיטאיות

6. כתבי תוכנית המקבלת 2 פרמטרים ומדפיסה את ההפרש ביניהם. בערך מוחלט.

7. כתבי תוכנית הקולטת גיל ומדפיסה גיל זה בימים ובחודשים.

8. כתבי תוכנית המחשבת ערךת של מספר נתון.

### מערכות ופונקציות

9. כתבי תוכנית המסכמת לטור מערך C, את איברי מערך B בהתאם.

10. כתבי תוכנית המקבלת כקלט מערך של מספרים חיוביים ושליליים, ומדפיסה לכל מספר האם  
הוא חיובי שלילי או 0.

11. כתבי תוכנית המדפיסה לטור קובץ איברי מערך זוגיים.

רין טאגאנך  
אן איזו  
קפאן הו  
אקס  
ליין גולאנש זם  
לט ?

6 נספח  
G נספח ג

```
if [ $1 -lt $2 ]; then
```

```
echo `expr $2 - $1`
```

```
else
```

```
echo `expr $1 - $2`
```

```
fi
```

שאלות  
7

```
echo `expr $1 \* 12`
```

```
echo `expr $1 \* 365`
```

שאלות  
8

```
i=1
```

```
x=1
```

```
while [ $i -lt $1 ]; do
```

```
x = `expr $x \* $i`
```

```
i = `expr $i + 1`
```

```
done
```

```
echo $x
```

a=(1 2 3 4 5)

9 נספח

b=(6 7 8 9 1)

ר' נספח 2

c=(0 0 0 0 0)

i=0

j=0

```
for i in ${a[@]}; do
```

```
c[$j]=`expr ${a[$j]} + ${b[$j]}`
```

```
j=`expr $j + 1`
```

```
done
```

/!#bin/bash

echo#\$ → נספח 3

echo \$1

i=0

for i in \$\*; do

echo \$i

done

echo \$\$ - *ינטגרט*

שאלות  
4

echo \$1

wc \$1

i=0

x=0

for i in `wc \$1`; do

if [ \$x -eq 0 ]; then

echo rows

else if [ \$x -eq 1 ]; then

echo words

else

echo tavim

fi

echo \$i

x=`expr \$x + 1`

done

4 נספח

## יצירת תהליכיים בלינוקו

**fork:** create a new process. The new process (child process) is almost an exact copy of the calling process (parent process). In this method we create an hierarchy structure for the processes, which is similar to the files structure in Unix. The root node of this tree is the process, which is the execution of the init program (pid = 1), is the ancestor of all the system and user processes.

### SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

The fork command copies all of the process (the code = the program, the data = global variables, the stack = automatic variable [local variables in the function] and the program counter).

The new process has a different pid and its ppid is the same as the pid of the calling process.

### RETURN VALUES

Upon successful completion, fork() returns a value of 0 to the child process and returns the process ID of the child process to the parent process.

Otherwise, a value of -1 is returned to the parent process, no child process is created, and the global variable errno is set to indicate the error.

Note: If the process has open files and performs fork(), then any change in the read/write head in anyprocess will entail the same change in the other processes, since the read/write head is shared for all processes because it is managed by the system.

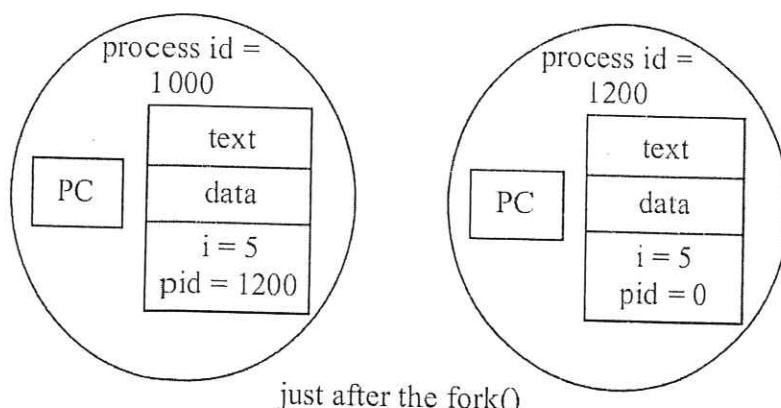
### Example: (2\_1.c)

```
#include <sys/types.h>
int main()
{
    pid_t pid;

    int i;
    i = 5;
    pid = fork();

    i++;

    printf("%d", i);
}
```



**Output:** \_\_\_, but it is unknown which process printed the first 6, since it depends on the CPU the system gave to each process (race condition).

Note that in the child process there are some lines that are never being executed (all the lines before the

fork). In the above example 3 lines are not executed in the child process.

### Examples:

```
(2_2.c)
#include <sys/types.h>
main()
{
    pid_t pid;
    pid = fork();
```

```

if (pid == 0)
    printf("\nI'm the child process");
else if (pid > 0)
    printf("\nI'm the parent process. My child pid is %d", pid);
else
    perror("error in fork");
}

```

**(2\_3.c)**

```

#include <sys/types.h>

main()
{
    pid_t pid;
    int i = 5;
    pid = fork();
    if (pid == 0) // child process
        i++;
    printf("%d", i);
}

```

**Output:** 45 or 56

**(2\_4.c)**

```

#include <sys/types.h>
main()
{
    pid_t pid;
    if ((pid = fork()) == 0)
        printf("1");
    else
        printf("2");
    printf("3");
}

```

**Output:** 2133 or 1233 or 2313 or 1323 (3312 is not possible)

**(2\_5.c)**

```

main()
{
    if (fork() == 0)
        while(1);
    else
        while(1);
}

```

1 > gcc -o example1 2\_5.c

2 > example1 &

[1] 7580

3 > ps -l

UID	PID	PPID	NI	STAT	TT	TIME	COMMAND
-----	-----	------	----	------	----	------	---------

```

8385 4709 4705 20 S
8385 7580 4709 20 R
8385 7581 7580 20 R
4 > kill -KILL %1
[1] Killed
example1

```

```

pts/0 0:01 tcsh
pts/0 0:21 example1
pts/0 0:22 example1

```

(2\_6.c)

```

main()
{
    int i;
    for (i = 0; i < 3; i++)
        if (fork() == 0)
            while(1)
}
1 > gcc -o example2 2_6.c
2 > example2
3 > ps -l

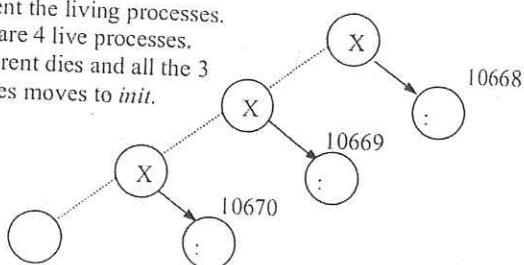
```

```

UID      PID  PPID NI STAT TT
8385 4709 4705 20 S
4 > ps -al
UID      PID  PPID NI STAT TT
8385 4709 4705 20 S
8385 10668 1          20 R
8385 10669 1          20 R
8385 10670 1          20 R
5 > kill -KILL 10668 10669 10670

```

The leaves represent the living processes.  
When  $i = 2$ , there are 4 live processes.  
Afterwards, the parent dies and all the 3 remaining processes moves to *init*.



TIME COMMAND  
pts/0 0:01 tcsh

TIME COMMAND  
pts/0 0:01 tcsh  
pts/0 0:12 example2  
pts/0 0:14 example2  
pts/0 0:13 example2

(2\_7.c)

```

main()

{
    if (fork() != 0)
        while(1);
}
2 > gcc -o example3 2_7.c
3 > example3 &
[1] 11499
4 > ps -al
UID      PID  PPID NI STAT TT
8385 11500 11499 0 Z
8385 4709 4705 20 S
8385 11499 4709 20 R
4 > kill -KILL %1
[1] Killed
example3

```

TIME COMMAND  
0:00 <defunct>  
pts/0 0:01 tcsh  
pts/0 0:44 example3

```
1 mkdir dir2
2 ls
3 mkdir stu
4 cd stu
5 touch a.dat
6 pico a.dat
7 cata.dat
8 cat a.dat
9 head a.dat
10 pico a.dat
11 cat a.dat
12 cls
13 mkdir student1
14 mkdir student2
15 cd student1
16 touch aaa.dat
17 ccd student1
18 pico aaa.dat
19 mkdir do1
20 mkdir do2
21 mkdir do3
22 cd do1
23 touch file1
24 pico file1
25 touch file2
26 pico file2
27 cp file1 file4
28 cp ..
29 cp ~
30 cd ~
31 cp do1 do5
32 cd do1
33 cp file1 ../do2
34 ls -l
/.. * 35 do2
/.. * 36 do3
37 cp * ../do3
38 cd ..
39 cd do2
40 cp ../do1/*
41 history >f
42 cat f
43 sort -r f
44 sort -r f |more
45 sort -r -n f| more
46 sort -r -n f
47 sort -r -n f
48 sort -r -n f |more
49 sort -r -n f > f
50 sort -r -n f |more|lpr
51 cd ..
52 ls -l
53 ls -l |more
54 ls -l |less
55 cleatr
56 clear
57 mkdi r a b c
58 mkdir a b c
```

```
59 cd ~
60 mkdir a b c
61 touch a/file
62 touch file
63 ls
64 touch file1
65 pico file1
66 cat file
67 cat file1
68 cd a
69 ls
70 ls -l
71 cd ~
72 cp a/file b
73 ls b
74 pico file
75 cat file
76 more file
77 less file
78 man history
79 man ls
80 head -30 file
81 head -20 file
82 chmod u-rwx file1
83 chmod u-rwx
84 chmod ugo-rwx file1
85 ls -l file1
86 chmod ug+rwx,o+w file1
87 ls -l file1
88 touch do
89 pico do
90 ls -l do.c
91 chmod u+rwx do.c
92 ls -l do.c
93 gcc -o do.c do
94 gcc -o do do.c
95 do
96 do.c
97 gcc -o do1 do1.c
98 ls -l do1.c
99 chmod u+rwx do1.c
100 ls -l do1.c
101 gcc -o do1 do1.c
102 pico do1.c
103 cat do1.c
104 pico do1.c
105 touch do2.c
106 pico do2.c
107 chmod u+rwx do2.c
108 gcc -o do2 do2.c
109 do2
110 gcc -o do2 do2.c &
111 pico do2.c
112 gcc -o do2 do2.c &
113 pico do2.c
114 gcc -o do2 do2.c &
115 pico do2.c
116 gcc -o do2 do2.c &
```

```
117 gcc -o do2 do2.c
118 ps
119 touch e1.c
120 pico e1.c
121 cd a
122 touch t1 t2 t3
123 ls
124 cd ~
125 cp a/t1 b/
126 cp a/t1 b
127 cd a
*128
129 cp t1 t2 t3 ~/b
130 cp a/file b
131 cp file b
132 cp file ~/b
133 cp * ~/c
134 pico t1
135 cat
136 cat t1
137 more t1
138 less t1
139 head 1 t1
140 head -1 t1
141 tail -2 t1
142 cd..
143 cd ..
144 history > his
145 cls
146 cls
147 man cls
148 ls -l > list
149 ls >> list
150 ps
151 ls | sort >> list
152 ls | sort -r >> list
153 sort list
154 alias "ddd=mkdir"
155 ddd d1
156 alias "cp=cd"
157 cp d1
158 cp ..
159 cp a/t1 d1
/ 160 cp a/t1 d1
\ 161 cp a/t1 d1
162 unalias cp
163 cp a/t1 d1
164 alias
165 history > exampel
```