# פקודות בסיסיות

כשרוצים עזרה עם פקודה מסוימת נרשום "man"ולאחר מכן שם הפקודה הרצויה. לדוגמה: man echo.

## פקודות על קבצים ותיקיות

### ניווט

pwd - כדי לגלות באיזה תיקיה (path) אני נמצא.

cd [path] - כדי לשנות תיקייה ואז את כתובת התיקייה הרצויה.

cd direc1/direc2/direc3

cd~ - להגיע לתיקיית home"".

cd.. - לחזור לתיקייה הקודמת.

### הצגת קבצים בתיקייה

כדי למנות את כל הקבצים בתיקייה נוכחית נרשום "ls" (קיצור של list). ניתן להרחיב את הפקודה ls באמצעות הפקודות הבאות. ניתן גם לחבר פקודות אלו.

ls -a – מראה גם קבצים נסתרים.

ls -l – נותן עוד פירוט על כל קובץ, כמו הרשאות ותאריך יצירה.

ls -t – ממיין לפי תאריך שינוי.

ls -s – ממיין לפי גודל.

ls -h – מציין את גודל הקובץ בצורה קריאה.

ls -r – הופך את הסדר.

Ls [filename] – כדי לקבל פרטים על קובץ ספציפי.

ls -\*.filesignture – כדי לקבל פרטים על כל הקבצים שיש להם סיומת מסוג מסוים.

man ls – לעוד אפשרויות.

### פעולות על תיקיות

mkdir [direcname] - ייצור תיקייה חדשה בשם direcname.

rmdir [direcname] - ימחק תיקייה עם שם שכתבנו. אפשר להשתמש גם ב-rm שמיועדת למחיקת קבצים.

### פעולות על קבצים

touch [filename] - ולאחריו שם הקובץ ייצור קובץ חדש. חשוב לא לשכוח לציין סיומת לקובץ.

rm [filename] - ולאחריו שם הקובץ ימחק אם הקובץ אם קיים. ניתן להוסיף דגל -r כדי למחוק תיקיות בלבד.

rm -r 'folder to delete'

cp [origin] [copy] - משכפל קובץ. נרשום את השם והסיומת של הקובץ שאנו רוצים לשכפל ואז את השם שאנו רוצים שיהיה של הקובץ המשוכפל.

cp filename.txt filename1.txt

mv [filename] [path] - מעביר קובץ לתיקייה אחרת. נרשום את השם והסיומת של הקובץ, ולאחר מכן את כתובת התיקייה הרצויה.

mv filename.txt ./new\_directory/

נשתמש בפקודה זו גם כדי לשנות שם קובץ.

mv filename.txt newname.txt

nano [filename] - עריכת קובץ מתוך הטרמינל.

## הרשאות

לכל קובץ בלינוקס יש רמת הרשאה. נוכל להשתמש בפקודה ls -l filename"" כדי גלות את רמת ההרשאה.

"r" – פירושו ניתן לקריאה.

"w" – פירושו ניתן לכתיבה.

"x" – פירושו ניתן לבצע.

ההרשאה מתחלקת ל-3 חלקים. השמאלי פונה למשתמש הראשי, האמצעי לקבוצה, והימני לעולם. לדוגמה: -rw-rw-r- פירושו ניתן לקריאה וכתיבה למשתמש ולקבוצה, ורק לקריאה לעולם.

### שינוי הרשאה

ניתן לשנות הרשאה באמצעות הפקודה "chmod":

* להורדת הרשאה: chmod [user/group/others/all]-[r/w/x] [file(s)]
* להוספת הרשאה: chmod [user/group/others/all]+[r/w/x] [file(s)]

## Piping

ניתן לחבר בין פקודות וקבצים בדרכים הבאות:

1. command > out – הפלט של הפקודה נכתב בקובץ ששמו out. כל מה שהיה בקובץ נדרס.
2. command >> out – הפלט של הפקודה נכתב בקובץ ששמו out, אך מתווסף לקובץ ולא נדרס.
3. command < in – הקלט של הפקודה נקרא מקובץ בשם in.
4. command1 | command2 – הפלט של פקודה command1 הופך להיות הקלט של פקודת command2. ניתן לשרשר כך כמה תוכניות.

command1 | command2 | command3 | …

## קריאה וכתיבה

### קריאה

cat [filename] - שלאחריו נרשום שם קובץ טקסט, ידפיס למסך את כל מה שיש בקובץ.

cat filename.txt

### כתיבה

echo [string] - מדפיסה טקסט שנרשום אחריו. הברירת מחדל היא להדפיס למסך. נוכל להפנות את הטקסט מ-echo לקובץ טקסט באמצעות '<<' או '<'. '<<' משרשר לסוף קובץ הטקסט, ואילו '<' דורס כל מה שכתוב בו וכותב מה שציינו.

echo hello world

echo hello world > filename.txt

echo hello world >> filename.txt

cat << EOF >> [filename] - פקודה המאפשרת לנו לכתוב לתוך קובץ ישירות דרך הטרמינל, עד שנכניס את המחרוזת "EOF". היה אפשר לבחור בפקודה כל מחרוזת שנרצה ולא דווקא EOF.

tee [string] – כמו echo, אלא שגם יכתוב לקובץ וגם יציג למסך מה נכתב.

## wc

שלאחריו שם של קובץ, יחזיר את מספר התווים, מילים ושורות בקובץ. ניתן להוסיף דגלים כדי לבחור רק אחד מהן.

-m מספר התווים.

-w מספר המילים.

-l מספר השורות.

## sort

- ממיין את filename1 ומכניס את הסדר הממוין ל-filename2.

sort [filename1] > [filename2]

# פקודות עריכת וחיפוש טקסט

## ביטוי רגולרי

ביטוי המייצג מחרוזת טקסט. בדרך כלל נשתמש כדי לחפש טקסט בקובץ.

'\*' שלאחריו או לפניו מחרוזת – מייצג את כל הקבצים שמתחילים או מסתיימים באותה מחרוזות.

echo \*.c //all files ended with '.c'

'[]' שבתוכו מחרוזת – בשביל לחפש קובץ שאחת מהתווים בתוך האופרטור נמצאים בו.

echo [a-df]xy.[cC] //starts with one of 'a,b,c,d,f', then 'xy.', then one of 'c,C'

'?' – יכול להיות כל תו במקום זה.

cat fileno?.txt //all files starts with 'fileno', then every char

\b – תוחם מילה. מציין שהמילה שהטקסט שאנו מחפשים מתחיל/מסתיים במילה חדשה., תלוי היכן מיקמנו את \b. אפשר גם להציב התחלה וסוף.

## grep

מחפש טקסט בתוך הקובץ. מערכת ההפעלה תעבור שורה אחר שורה בקובץ, ותחזיר כל שורה שבה מופיע המחרוזת שהכנסנו. בדרך כלל משתמשים בביטוי רגולרי כדי לחפש מחרוזות מורכבות.

grep [string] < [filename]

-i גם אותיות גדולות וגם קטנות בטקסט שהזנו.

-v יחזיר את כל השורות שאינן מכילות את הטקסט.

-n מדפיס גם את מספר השורה שבה נמצא מה שחיפשנו.

-w מחפש רק מילים שלימות ולא חצאי מילים.

## Sed

דומה ל-grep אלא שמבצע פעולות נוספות של עריכת טקסט. פקודת sed משמשת לחיפוש, החלפה, הכנסה ומחיקה. השימוש העיקרי אמנם הוא מציאה והחלפה. גם כן תומך בביטויים רגולריים. הסינטקס של הפקודה הוא:

sed OPTIONS... [SCRIPT] [INPUTFILE...]

### החלפה

כדי לבצע החלפה יש להשתמש באפשרות s (substitution). לאחר מכן 3 '/' כאשר בין הראשון לשני המילה שאנו רוצים לחפש ולהחליף, ובין השני לשלישי המילה שתחליף את המילה הקודמת. הברירת מחדל היא שפקודה זו מחליפה את ההופעה הראשונה של המילה שציינו בכל שורה. אם יש יותר מהופעה אחת, היא לא תימחק.

sed 's/unix/linux/' file.txt

כדי להחליף את ההופעה ה-n בכל שורה, נרשום לאחר ה-/ השלישי את מספר המופע שרוצים להחליף. כדי להחליף את כל ההופעות בכל שורה, נרשום במקום מספר את האות 'g' (global). וכדי להחליף מההופעה ה-n ועד סוף השורה, נרשום את מספר ההופעה ולאחריה 'g'.

sed 's/unix/linux/3' file.txt // replace 3 instance

sed 's/unix/linux/g' file.txt // replace all instance

sed 's/unix/linux/3g' file.txt // replace 3-end line instance

כדי שהפעולה תתבצע רק עבור שורות מסוימות, נרשום את מספר השורה לפני ה-s. כדי לקבל טווח של שורות נבדיל בפסיק את מספרי השורות. כדי שהפעולה תתבצע החל משורה מסוימת, נרשום במספר השורה האחרונה $.

sed '2 s/unix/linux/' file.txt

sed '2,4 s/unix/linux/' file.txt

sed '2,$ s/unix/linux/' file.txt

כדי להדפיס רק את השורות שבהן נעשה שינוי נוסיף את האפשרות -s לאחר פקודת ה-sed ו-p לאחר מכן. אם נרשום p בלבד יודפסו כל השורות, אך שורות בהן היה שינוי יודפסו פעמיים.

sed -n 's/unix/linux/p' file.txt

### מחיקה

כדי למחוק מילים ניתן להשתמש בפקודת החלפה לעיל. הפקודה תחפש מילים ותחליף אותן במילה ריקה.

sed 's/unix//' file.txt

כדי למחוק שורה, נרשום את מספר השורה ולאחריה d (delete). אפשר למחוק טווח שורות באמצעות פסיק המפריד בין השורות. הושרה האחרונה מיוצגת באמצעות $.

sed '5d' filename.txt

sed '2,5d' filename.txt

sed '2,$d' filename.txt

### דגלים

-i גם אותיות גדולות וגם קטנות בטקסט שהזנו.

## awk

הרחבה של sed, ומשמשת גם כן לעריכת טקסט. Awk עובדת בצורה כזו שהיא מחפשת טקסט בכל שורה. עבור השורות שקיים בהן הטקסט, היא מאפשרת להריץ פקודות שלימות עם תנאים ומשתנים.

כדי לחפש שורה שבה נמצא טקסט כלשהו נרשום פעמיים '/', שבתוכן יהיה הטקסט לחיפוש. יכול להיות גם ביטוי רגולרי. אפשר גם להכניס תנאי רגיל. אם לא מציינים טקסט לחיפוש או תנאי אחר, הברירת מחדל היא לעבור על כל השורות בקובץ. לאחר החיפוש נפתח סוגריים מסולסלות ובהן נרשום את כל הפקודות bash שנרצה להריץ עבור כל שורה שמקיימת את התנאי. נוכל להוסיף גם תנאים. פונקציה שימושית היא print, שאם לא מקבלת כלום מדפיסה את כל השורה.

awk '/unix/ {print}' file.txt

כל שורה בטקסט מחלקת את המילים על ידי רווח. נוכל לגשת למילה ה-n באמצעות $n. הערך $0 מייצג את השורה כולה. אם ניגש בשורה למילה ה-n, כאשר בשורה זו יש פחות מ-n מילים, תודפס שורה ריקה.

$awk '{print $1,$4}' file.txt //print only the 1,4 words in every row

### משתנים מיוחדים

NR - בתוך כל איטרציה מחזיר את מספר השורה שאנו נמצאים בה.

ORS - סוכם את הפלט של כל שורה ללא ירידת שורה. לבסוף מדפיס אוטומטית את מה שמכיל בתוכו. משתנה זה מאפשר לנו להדפיס מספר שורות בקובץ כשורה אחת.

# הרצת תוכניות

## שלבי קימפול תוכנית

תהליך הקימפול מחולק לשלושה חלקים: קדם מעבד (preprocessor), מהדר (compile), ומחבר (linkage). הקדם מעבד והמהדר לוקחים את כל קבצי הקוד (file.c) בתכנית שלנו, ומחזירים במקומם קבצי object (file.o). המחבר לוקח את כל קבצי object ומחבר אותם לקובץ אחד (בשם של קובץ main) המוכן להרצת המחשב.

באמצעות gcc ניתן לבצע שתי פעולות אלו בנפרד. אם לדוגמא יש לנו שני קבצי קוד, אחד בשם main.c והשני בשם square.c. השורה הראשונה תיצור שני קבצי object – main.o ו-square.o. השורה השנייה תיצור את הקובץ המאוחד המוכן להרצה.

$ gcc –c Square.c –o Square.o // preprocessor and compile only

$ gcc –c Main.c –o Main.o

$ gcc Square.o Main.o –o Main // linkage only

אם נעדכן קובץ אחד בתוכנית שלנו, באמצעות פקודות אלו לא נצטרך לקמפל את כל הקבצים שוב אלא רק את אותו קובץ.

## הפעלת תוכנית

כדי להריץ תוכנית פשוט צריך לרשום את ה-path שמסתיים בקובץ שאנו רוצים להריץ. אם אנו בתיקייה של הקובץ נרשום "./" ואז את שם הקובץ והסיומת שלו. כדי לסיים הרצת תוכנית נלחץ ctrl+c.

./filename.txt

## פלט וקלט מתוכנית לקובץ

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

* ערוץ הקלט הסטנדרטי (ערוץ מספר 0). מחובר למקלדת.
* ערוץ הפלט הסטנדרטי (ערוץ מספר 1). מחובר אל המסך.
* הערוץ של הודעות השגיאה (ערוץ מספר 2). מחובר אל המסך.

### קלט

כדי לקלוט ערכים מקובץ נשתמש באופרטור > בהרצת הקובץ. כל הערכים ישמרו בבאפר, עד שנחליט בתוכנה למשוף אותם באמצעות פונקציות כמו scanf או cin.

/.a.out < input.txt

### פלט

כשמריצים תוכנית בלינוקס אנו יכולים להפנות את הפלט התקני לקובץ אחד ואת פלט השגיאה התקני לקובץ אחר. למשל, בפקודות הבאות:

./a.out

./a.out > out.txt

./a.out 2> err.txt

./a.out > out.txt 2> err.txt

הפקודה הראשונה כותבת גם את הפלט התקני וגם את פלט השגיאה התקני למסך. הפקודה השנייה כותבת את הפלט התקני לקובץ, ואת פלט השגיאה התקני למסך (שימושי כשיש הרבה פלט שלא רוצים לראות על המסך אלא לשמור בקובץ לעיון מאוחר יותר, אבל עדיין רוצים שהודעות שגיאה יגיעו למסך באופן מידי). הפקודה השלישית כותבת את פלט השגיאה התקני לקובץ ואת הפלט הרגיל למסך. והפקודה הרביעית כותבת את שני סוגי הפלטים לשני קבצים שונים.

כל הפקודות לעיל משתמשות ב-'<', ולכן הפלט ידרוס כל מה שבקובץ ויכתוב מחדש. כדי להוסיף בסוף הקובץ ולהימנע מדריסה נשתמש באותן הפקודות רק ב-'<<'.

### קבצים מיוחדים

יש שני קבצים מיוחדים שאפשר לכתוב/לקרוא מהם תוך כדי ריצת תוכנית:

* /dev/null - קובץ זה מיועד לכתיבה בלבד. מידע שנכתב אליו לא נשמר בשום מקום, ולא ניתן לקריאה. שימושי כאשר אנחנו מעוניינים "להשתיק" פלט מתוכנה.
* /dev/zero - קובץ זה מיועד לקריאה בלבד. קובץ שתמיד ניתן לקרוא ממנו אפסים והוא לעולם לא "נגמר". שימושי בעיקר במקרה שהתוכנה דורשת קובץ קלט, ואין לנו קובץ קלט לתת לה.

## קבלת ערכים לתוכנית מהטרמינל

כדי שנוכל להכניס ערכים לתוכנית דרך שורת הפקודה נגדיר את פונקציית main בדרך הבאה:

Int main (int argc, char\* argv[]) { ...

לאחר הגדרה זו אם נרשום בשורת הפקודה:

$ prog1 –u danny –p 1234

כל ביטוי שיש אחריו רווח, יקלט בתוכנית בתור איבר במערך argv:

argc == 5   
argv[0] == “prog1”   
argv[1] == “-u”  
 ...   
argv[4] == “1234"

ניתן להבחין שיש דמיון ל-JAVA מבחינת הגדרת הפונקציה main. אמנם בניגוד ל-JAVA ב-C האיבר הראשון הוא שם התוכנית עצמה.

### הגדרת קבוע

ניתן להגדיר שקבוע מוגדר לערך כלשהו דרך פקודת gcc, במקום להשתמש ב-#define. כותבים:

gcc -D <**const**> = <val> <source\_file> -o <executable>

כאשר const הוא הקבוע, val הוא ערכו, source\_fileהוא שם קובץ הקוד, ו executable-הוא שם קובץ התוכנית המתקבלת. לדוגמה, הפקודה:

gcc -D RED=1 red\_test.c -o red\_test.out

תהדר את הקובץ red\_test.c, ובכל מקום בו תזהה בו את הרצף RED היא תחליפו ל-1.

כדי להגדיר שקבוע כלשהו סתם מוגדר (לא לערך כלשהו), נבצע את אותה הפעולה בלי val.

gcc -D DEBUG red\_test.c -o red\_test.out

# מערכת הפעלה

## תכונות מערכת ההפעלה

top – לראות ביצועים ותהליכים פועלים במחשב בזמן אמת.

ps – מראה אילו תהליכים (process) רצים כעת במערכת הפעלה.

kill – הריגת תהליכים לפי מספר.

killall – הריגת תהליכים לפי שם.

## סביבות עבודה

יש הרבה סביבות עבודה שאפשר להריץ בהם מערכת הפעלה של לינוקס

### AWS Cloud 9

הפעלת סביבה וירטואלית של לינוקס באתר של AWS.

### Virtual Machine

מכונה וירטואלית שעליה ניתן להריץ איזה סביבה וירטואלית שנרצה. צריך להקצות לה זיכרון HD ו-RAM.

### Dual Boot

התקנה מלאה של לינוקס על המחשב ביחד עם ווינדוס.

### Docker

תוכנה מינימלית להרצה של טרמינל בלבד. אין צורך להקצות לה משאבים, אלא לוקחת מינימום של משאבים פנויים.

בעזרת Docker image ניתן לייצר מצב של מערכת הפעלה, לשלוח ולהפעיל אותה ב-Docker ממחשבים אחרים. שימושי מאוד בתעשייה. להתקנה של Docker ראה מצגת תרגול 1 בקורס מערכות הפעלה.

### Ubunto Desktop

אפליקציה של ווינדוס להרצת אובונטו על המחשב.