

## תרגיל 1 - מיני מעטפת

מערכות הפעלה, סמסטר ב' תשפ"ה

להגשה: 20.4.2025

הסבר לתרגיל -

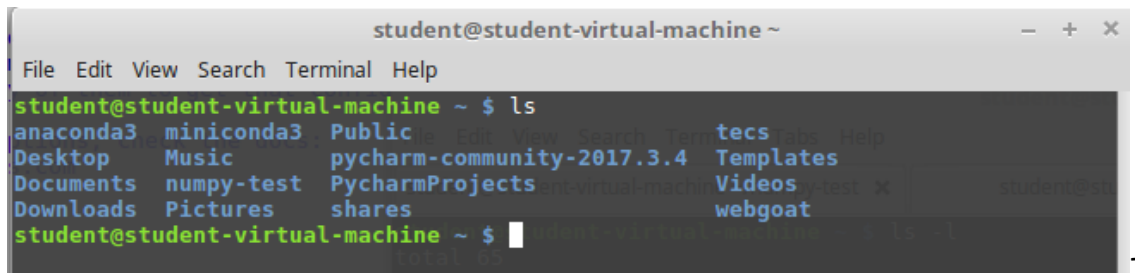
מבוא ורקע:

מה זה ה"shell" ?

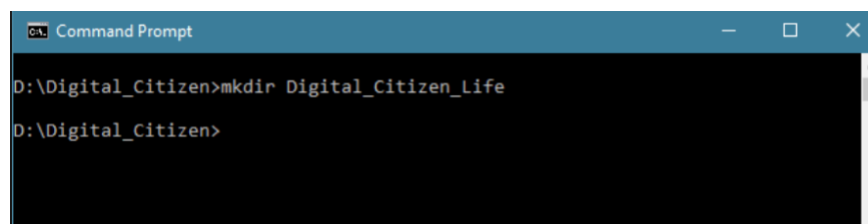
במילים פשוטות, הshell זו תוכנית ממשק אשר קוראת פקודות מהמשתמש ומעבירה אותם בפורמט המתאים למערכת ההפעלה לביצוע. הפקודות בshell הם שורות טקסט אשר מוזנות על-ידי המשתמש. בעבר, זה היה הממשק היחיד למערכת הפעלה. היום כמובן, יש גם את הממשק הגרפי שכולנו מכירים.

לshell יש מספר גירסאות, בגרסאות הראשונות של לינוקס התוכנית נקראה sh תוכנית זו נכתבה על-ידי Steve Bourne. כיום רוב גירסאות הLinux הshell אשר נמצא בשימוש גירסה חדשה יותר של הshell והיא bash שזה ראשי תיבות של Bourne Again Shell. ויש גרסאות נוספות כגון tcsh ואחרות פחות נפוצות: ksh, zsh.

לדוגמא על מנת להורות למערכת הפעלה להציג את תוכן הקבצים בתקיה, בלינוקס נכתוב ls



גם בwindows יש ממשק shell טקסטואלי הנקרא גם command prompt. לדוגמא:



בנוסף, בwindows יש גם את ה power shell.

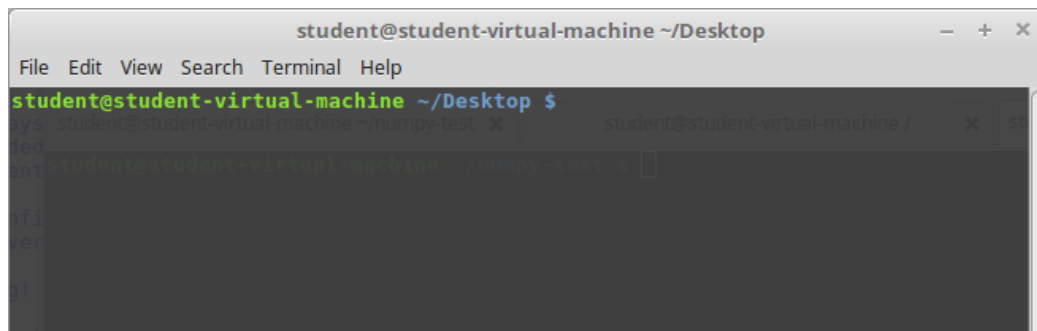
מאפיינים נוספים של ה shell לינוקס

## 1. prompt

בחלון הבא, אנחנו רואים דוגמה לחלון **shell**, שורת ה **prompt** מוצגת בירוק וכחול ומסתיימת ב \$ .

`~/desktop` `( student@stuent-vitual-machine` במקרה שלנו.

אחרי ה **prompt** זה המקום שהמשתמש מקליד את פקודות לביצוע על-ידי מערכת ההפעלה. ( מילים אחרות, ה **prompt** היא אותה מחרוזת אשר מופיעה לפני המקום שבו ניתן להקיש את הפקודה )



## 2. מדידת זמנים

מנגנון מדידת הזמנים מאפשר למשתמש לקבל מידע על משך זמן הריצה של פקודה או סדרת פקודות במעטפת. המדידה שימושית לזיהוי פקודות איטיות, ניטור ביצועים או ניתוח של סקריפטים. את הזמנים ניתן להציג על המסך או לשמור בקובץ לצורך תיעוד וניתוח מאוחר יותר.

## 3. פקודות מסוכנות

מנגנון הבטיחות במעטפת נועד להגן על המשתמש והמערכת מפני הרצת פקודות שעלולות לגרום נזק למערכת ההפעלה או לקבצים חשובים. המעטפת תזהה ותמנע הרצה של פקודות שהוגדרו כמסוכנות, כגון מחיקת קבצים קריטיים או שימוש לא זהיר בהרשאות מנהל. במקרה של זיהוי פקודה מסוכנת, המעטפת תציג למשתמש הודעת אזהרה ברורה ותמנע את ביצועה .

## הגדרת התרגיל

בתרגיל זה נממש בעצמנו מעטפת (shell) פשוטה בשפת C תחת מערכת ההפעלה linux. המעטפת תציג prompt למשתמש תקרא את הפקודות ותשלח אותם למערכת הפעלה לביצוע. וכן את שלושת המאפיינים שפרטו ב "מבוא ורקע" והם (1) **prompt** | (2) **time** | (3)

## dangerous commands

דרישות כלליות מהמעטפת יהיו בדלהלן:

### 1. קלט של פקודה מהמשתמש

- אורך הפקודה המקסימלי כולל הארגומנטים הינו 1024 תווים.
- מספר הארגומנטים המקסימלי הוא 6. אם יש יותר מ 6 ארגומנטים לא כולל שם הפקודה עצמה. הפקודה תוגדר כלא לא חוקית. ויש להדפיס למסך ERR\_ARGS
- לא ניתן להניח הגבלה על מספר התווים של כל ארגומנט (או של שם התכנית) אך סכומם קטן שווה מ 1024
- בין ארגומנט אחד לשני יכולים להיות רווח אד בלבד. אם יש יותר. יש להדפיס למסך ERR\_SPACE

### 2. הרצת הפקודה

הפקודות ישלחו למערכת הפעלה לביצוע באמצעות הפקודות `fork` | `execvp`. ההרצה תהיה על-ידי כך שהתוכנית תייצר תהליך בן. הבן יהיה זה שבפועל יריץ את הפקודה תוך שימוש בפקודה `execvp` – **פרטים נתנו בשיעור ובתרגול**

3. המעטפת תצא כאשר יוקלד המחזורות `done` כאשר המעטפת תצא – היא תדפיס למסך את מספר הפקודות המסוכנות (שנחסמו ושלא נחסמו) שניסו להריץ. פשוט במספר.

### 4. הסבר על מנגנון פקודות מסוכנות

- מנגנון הבטיחות ב shell-מיועד למנוע הרצה של פקודות שהוגדרו כמסוכנות. פקודות אלו ייקראו מקובץ חיצוני שיסופק בארגומנט דרך `argv`. לכל פקודה מסוכנת בקובץ יכול להיות אוסף של ארגומנטים.

- shell תבצע בדיקה בכל פעם שמשתמש מקליד פקודה לפי הכללים הבאים:

- שלבי מימוש המנגנון

### טעינת קובץ הפקודות המסוכנות

בעת אתחול shell, קיראו את הקובץ שסופק לכם ב argv-לתוך מבנה נתונים מתאים.

כל שורה בקובץ מייצגת פקודה מסוכנת ספציפית, עם ארגומנטים מדויקים.

### פורמט לדוגמה של קובץ הפקודות המסוכנות:

```
rm -rf /
```

```
sudo reboot
```

```
shutdown -h now
```

```
mkfs.ext4 /dev/sda
```

הקובץ מכיל פקודות עם ארגומנטים מדויקים ומפורשים.

- כללי השוואה וזיהוי פקודות מסוכנות

כשהמשתמש יקליד פקודה במעטפת, השוו אותה לפקודות המסוכנות לפי הכללים הבאים:

- אם הפקודה שהמשתמש הקליד והארגומנטים זהים בדיוק

המעטפת תמנע לחלוטין את ביצוע הפקודה, ותציג הודעה ברורה:

ERR: Dangerous command detected ("rm -rf /"). Execution prevented.

- אם שם הפקודה זהה, אבל הארגומנטים לא זהים במדויק

המעטפת תציג אזהרה למשתמש שהפקודה שהקליד דומה לפקודה מסוכנת, אך לא תמנע את הביצוע:

WARNING: Command similar to dangerous command ("rm -rf /"). Proceed with caution.

ולאחר מכן הפקודה תרוץ כרגיל.

**אם שם הפקודה אינו מופיע כלל ברשימת הפקודות המסוכנות**  
המעטפת תמשיך לפעול כרגיל ללא התראות או חסימות.

### 1. מימוש מנגנון מדידת זמנים ורישום לקובץ פלט

מטרת המנגנון הזה היא למדוד את זמן הריצה של כל פקודה שהמשתמש מריץ ב **shell**-שלכם, ולשמור את התוצאות בקובץ פלט ייעודי ששמו יועבר כארגומנט בשורת הפקודה (**argv**).

**שלבי המימוש:**

### 6. הסבר על זמנים

#### קבלת שם קובץ הפלט

שם קובץ הפלט יימסר לתוכנית דרך **argv**.  
לדוגמה, הרצת התוכנית תראה כך:

```
./myshell dangerous_commands.txt exec_times.log
```

כאשר **exec\_times.log** הוא שם קובץ הפלט שאליו יירשמו הזמנים.

ו **dangerous\_commands.txt** הוא קובץ ממנו יקראו פקודות מסוכנות

#### מדידת זמן הרצת הפקודה

- השתמשו בפונקציות מערכת כמו **gettimeofday()** או **clock\_gettime()** כדי למדוד זמן באופן מדויק.
- שמרו את הזמן המדויק מיד לפני ביצוע הפקודה ומיד לאחר סיומה, וחישבו את ההפרש.

#### • כתיבה לקובץ הפלט

- פתחו את קובץ הפלט במצב הוספה ("**append**"), כך שכל פקודה חדשה תתועד בשורה חדשה ללא מחיקת התוכן הקודם.

- בכל פעם שמסתיימת הרצת פקודה, כתבו לקובץ הפלט שורה המכילה את הפקודה שבוצעה ואת הזמן שלקח לה לבצע.

(exec\_times.log): מבנה לדוגמה של קובץ הפלט

ls -l : 0.00234 sec

sleep 5 : 5.00012 sec

grep "pattern" file.txt : 0.12345 sec

- כל שורה מתארת פקודה שהורצה, ולאחריה זמן הריצה שלה בשניות, בדיוק של לפחות 5 ספרות אחרי הנקודה.

## 5. מבנה ה-prompt הנדרש בתרגיל

- ה-prompt החדש במעטפת שלכם יכול את האלמנטים הבאים בלבד:

```
#cmd:<number>|#dangerous_cmd_blocked:<number>|last_cmd_time:<time_in_sec>|avg_time:<time_in_sec>|min_time:<time_in_sec>|max_time:<time_in_sec>>
```

- פירוט האלמנטים ב-prompt-

- #cmd:<number>

מייצג את מספר הפקודות החוקיות (**commands**) שהורצו בהצלחה במעטפת מאז תחילת הריצה.  
פקודות חוקיות הן פקודות שהמערכת הצליחה להריץ ללא שגיאות.

- #dangerous\_cmd\_blocked:<number>

מייצג את מספר הפקודות המסוכנות שנחסמו על ידי מנגנון הבטיחות מאז תחילת ריצת המעטפת.  
פקודות מסוכנות הן פקודות שהוגדרו מראש בקובץ המסופק לתוכנית ונחסמו בעקבות התאמה מדויקת (פקודה וארגומנטים זהים)

- last\_cmd\_time:<time\_in\_sec>

מייצג את זמן הריצה בשניות של הפקודה האחרונה שהורצה בהצלחה.

אם עדיין לא הורצה אף פקודה חוקית, יוצג הערך **0.00000**.  
הזמן יוצג בדיוק של 5 ספרות אחרי הנקודה העשרונית לפחות.

● **avg\_time:<time\_in\_sec>**

מייצג את זמן הריצה הממוצע (**Average**) של כל הפקודות החוקיות שהורצו  
בהצלחה מאז תחילת הריצה של המעטפת.

● **min\_time:<time\_in\_sec>**

מייצג את זמן הריצה המינימלי של פקודה חוקית שהורצה בהצלחה עד כה.

● **max\_time:<time\_in\_sec>**

מייצג את זמן הריצה המקסימלי של פקודה חוקית שהורצה בהצלחה עד כה.

● **דוגמה מוחשית ל:prompt-**

```
#cmd:8|#dangerous_cmd_blocked:3|last_cmd_time:0.00235|avg_time:0.00112  
|min_time:0.00002|max_time:0.00300>>
```

● בדוגמה זו:

- שמונה פקודות חוקיות הורצו בהצלחה.
- שלוש פקודות מסוכנות נחסמו.
- הפקודה החוקית האחרונה ארכה 0.00235 שניות.
- זמן הריצה הממוצע של הפקודות עד כה הוא 0.00112 שניות.
- זמן הריצה המינימלי היה 0.00002 שניות.
- זמן הריצה המקסימלי היה 0.00300 שניות.

### עוד פרטים והמלצות טכניות

1. במקרה של שגיאה בהקצאת זיכרון או ביצירת תהליך בן, תודפס שגיאה ERR והמעטפת תצא.
2. התכנית תחכה לסיום תהליך הבן (רמז הפקודה wait) לפני שתציג את שורת הprompt על-מנת לתאפשר קלט נוסף.
3. תוכלו לבדוק את המעטפת על פקודות דוגמת man, echo, ls, cat, more, expr, sleep.
4. הקפידו לשחרר זיכרון!

5. התרגיל הזה יהיה הבסיס לתרגילים הבאים. כתבו אותו בצורה מסודרת, מתועדת, מובנת ומודולרית לא רק בשביל הציון אלא גם בשביל להקל עליכם את התרגילים הבאים.

### שאלות ותשובות

- מה נחשבת פקודה לא חוקית ?
- פקודות לא חוקיות כאשר ערך אם `exec` החזירה ערך של כישלון.
- מה המשמעות "אם יש יותר מ-6 ארגומנטים הפקודה תוגדר כלא חוקית?"
- הכוונה היא שהפקודה המדוברת יכולה לקבל לכל היותר 6 ארגומנטים לא כולל שם הפקודה עצמה. אם מועברים לה יותר מ-6 ארגומנטים, היא ותדווח כשגיאה
- – מה המשמעות של גרשיים מבחינת הקוד, מה צריך לעשות במקרה שיש גרשיים ?
- אפשר להניח שפקודות לא כוללות גרשיים
- יש הגבלות על ה `name` או `command`? שלא יכיל תווים שאינם אותיות או תווים מיוחדים? צריך לבדוק את זה?
- תשובה: לא צריך לבדוק חוקיות של `name` או `command`
- יש הגבלות על ה `name` או `command`? שלא יכיל תווים שאינם אותיות או תווים מיוחדים? צריך לבדוק את זה?
- תשובה: יכול להיות שיהיו שורות ריקות
- "תודפס שגיאה" – מה זה אומר? מה בדיוק אמורים להדפיס? ומה להדפיס אם פקודה נכשלה (`exec` נכשל)?
- תשובה: יש להדפיס: `perror(<command_name>);` לדוגמה:
- `perror("fork"); perror("malloc"); perror("exec");`
- כאשר מספק ארגומנטים גדול מ-6 יודפס `printf("ERR_ARGS")` הפקודה לא תתבצע
- כאשר יותר מרווח אחד יש להדפיס שגיאה. `printf("ERR_SPACE")` הפקודה לא תתבצע
- בכל שאר המקרים של שגיאה יש להדפיס `printf("ERR")`



## הגשה וכללים נוספים:

יש לממש בדיוק ע"פ הגדרות התרגיל – לא יותר ולא פחות

כיתבו את הקוד כולו בקובץ בשם ex1.c

ארוזו בקובץ zip את ה ex1.c וה- README,

לקובץ ההרצה בפקודת הקמפול יש לקרוא ex1. יש להוסיף בקמפול את הדגל -Wall

עבודה עצמאית, "אל תעבדו על עצמכם" מצאתי פתרון ברשת או במקום אחר או אמרו

לי איזה prompt לכתוב – זה לא עבודה עצמאית, ואנחנו גם נבדוק את זה. (בבקשה,

בבקשה! זה רק בשבילכם, לא תעבדו עצמאית לא תלמדו...)

השנה תכלול הבחינה כתיבת קטעי קוד קטנים המבוססים על התרגילים