ארגון ותכנות המחשב

תרגיל 3 - חלק רטוב

המתרגל האחראי על התרגיל: בועז מואב.

שאלות על התרגיל – ב- Piazza בלבד.

~~ התרגיל מורכב משני חלקים בלתי תלויים ~~

הוראות הגשה:

- ההגשה בזוגות.
- על כל יום איחור או חלק ממנו, שאינו באישור מראש, יורדו 5 נקודות.
    - ניתן לאחר ב-3 ימים לכל היותר.
- הגשות באיחור יתבצעו דרך אתר הקורס.
- הוראות הגשה נוספות מופיעות בסוף התרגיל. אנא קראו בעיון.

חלק א – Hacking readelf

מבוא

בחלק זה אנו הולכים לשנות מספר חלקים בקוד של התוכנית readelf, שפגשתם בהרצאה ובתרגול 8.
המטרה של התרגיל מתחלקת לשניים:

1. להרגיש בידיים את המימוש של readelf ובפרט את המבנה של קובץ ELF שנלמד בתרגול.
2. להתמודד בגבורה עם קוד פתוח גדול, ובמבט ראשון אף מאיים.

הכנת סביבת העבודה

1. ודאו כי יש ברשותכם חיבור לאינטרנט
2. בחרו את התיקייה בה תעבדו על חלק זה של תרגיל הבית, פתחו את הטרמינל בתיקייה זו והריצו את הפקודה הבאה, על מנת להוריד את binutils:

```
wget http://ftp.gnu.org/gnu/binutils/binutils-2.36.1.tar.xz -O - | tar -xJ
```

3. בנו את binutils כך (הפקודה make תיקח דקות אחדות):

```
cd binutils-2.36.1
./configure
make
```

4. בדקו כי readelf החדש עובד טוב, על ידי הרצתו. למשל, על הקובץ a.o שסופק לכם:

```
binutils/readelf -s a.o
```

ודאו כי מתקבל הפלט הבא:

```
Symbol table '.symtab' contains 5 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     4: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT    2 x
```

5. כעת ניתן להתחיל לפתור את תרגיל הבית.

שלב ראשון – שינוי ההדפסות ב-Section Headers Table

היזכרו כיצד מדפיסים את טבלת ה-Section Headers. לדוגמה:

```
There are 7 section headers, starting at offset 0xf0:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         0000000000000000  00000040
       0000000000000000  0000000000000000  AX       0     0     1
  [ 2] .data             PROGBITS         0000000000000000  00000040
       0000000000000008  0000000000000000  WA       0     0     1
  [ 3] .bss              NOBITS           0000000000000000  00000048
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .symtab           SYMTAB           0000000000000000  00000048
       0000000000000078  0000000000000018           5     4     8
  [ 5] .strtab           STRTAB           0000000000000000  000000c0
       0000000000000003  0000000000000000           0     0     1
  [ 6] .shstrtab         STRTAB           0000000000000000  000000c3
       000000000000002c  0000000000000000           0     0     1
```

אתם סטודנטים שחברים במועדון המעריצים של אלה לי ורוצים לחגוג את הגעתה הצפויה ליום הסטודנט. לכן, מאחר שאתם יודעים שאלה לי משתמשת אדוקה של readelf, החלטתם לייצר עבורה פלט חדש עבור הדגל שמשמש להדפסה

הנ"ל והוא תוספת של המחרוזת "_<3LEE" לכל שם של section header, שאינו מטיפוס NULL (בלומר sh_type≠NULL), שמודפס למסך בעת ביצוע הפקודה.

אתם תעשו זאת ע"י שינוי הקובץ readelf.c (בקבצים שייבאנו קודם), כך שבעת הרצת **`binutils/readelf`** עם הדגל המתאים להדפסת ה-section header על קובץ ELF כלשהו, יתקבל הפלט עם השינוי שלכם. לדוגמה:

```
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text_<3LEE       PROGBITS         0000000000000000  00000040
       0000000000000000  0000000000000000  AX       0     0     1
  [ 2] .data_<3LEE       PROGBITS         0000000000000000  00000040
       0000000000000008  0000000000000000  WA       0     0     1
  [ 3] .bss_<3LEE        NOBITS           0000000000000000  00000048
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .symtab_<3LEE     SYMTAB           0000000000000000  00000048
       0000000000000078  0000000000000018           5     4     8
  [ 5] .strtab_<3LEE     STRTAB           0000000000000000  000000c0
       0000000000000003  0000000000000000           0     0     1
  [ 6] .shstrtab_<3LEE   STRTAB           0000000000000000  000000c3
       000000000000002c  0000000000000000           0     0     1
```

<u>**הערה**</u>: לפעמים הוספת המחרוזת "_<3LEE" תגרום למחרוזת החדשה לא להיכנס בהדפסה ותקבלו משהו כזה:

```
[Nr] Name              Type             Address           Offset
     Size              EntSize          Flags  Link  Info  Align
[ 0]                   NULL             0000000000000000  00000000
     0000000000000000  0000000000000000           0     0     0
[ 1] .interp_<3LEE     PROGBITS         0000000000000238  00000238
     000000000000001c  0000000000000000   A       0     0     1
[ 2] .note.ABI-ta[...] NOTE             0000000000000254  00000254
     0000000000000020  0000000000000000   A       0     0     4
```

זו ההדפסה הצפויה במקרה כזה. לעומת זאת, אם נשתמש גם בדגל W, נצפה להדפסה הבאה:

```
Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00      0   0  0
  [ 1] .interp_<3LEE     PROGBITS        0000000000000238 000238 00001c 00   A  0   0  1
  [ 2] .note.ABI-tag_<3LEE NOTE            0000000000000254 000254 000020 00   A  0   0  4
  [ 3] .note.gnu.build-id_<3LEE NOTE           0000000000000274 000274 000024 00   A  0   0  4
  [ 4] .gnu.hash_<3LEE   GNU_HASH        0000000000000298 000298 00001c 00   A  5   0  8
```

שלב שני – ספירת הסמלים הגלובליים

אלה נורא אהבה את המחווה שעשיתם עבורה בחלק הראשון של התרגיל, לכן ביקשה ממכם טובה קטנה. אלה רוצה לדעת כמה מהסמלים בקובץ ELF נתון הם סמלים גלובליים. היא מבקשת שתוסיפו שורה חדשה, בסוף ההדפסה של טבלת הסמלים (בגרסת readelf שאנו עורכים), אשר תכיל את אחת מהמחרוזות הבאות:

- אם אין סמלים גלובליים בקובץ – "\nThere are no GLOBAL symbols in this file.\n".
- אם יש סמל אחד גלובלי בקובץ – "\nThere is 1 GLOBAL symbol in this file.\n".
- אם קיים יותר מסמל גלובלי אחד בקובץ – "\nThere are %d GLOBAL symbols in this file.\n". (באשר במקום %d תודפס כמות הסמלים הגלובליים בקובץ כמובן)

דוגמת הרצה:

```
Symbol table '.symtab' contains 5 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     4: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT    2 x

There is 1 GLOBAL symbol in this file.
```

שלב שלישי – הגשה וטסטים

עליכם להגיש את הקובץ readelf.c שאותו ערכתם. הוראות ההגשה המדויקות נמצאות בסוף קובץ זה.

בבדיקת התרגיל אנו נריץ את make עם הקובץ שלכם במקום readelf.c המקורי ונריץ את **`binutils/readelf`** עם כל מיני דגלים שונים ומשונים (מהדגלים שפגשנו בקורס), על כל מיני קבצי ELF שונים ומשונים. הקוד צריך לרוץ בהצלחה (בלומר, כפי ש-readelf המקורי רץ) על כל הדגלים, מלבד אלו שנתבקשתם לשנות. דגל נוסף שכדאי לכם לבדוק הוא הדגל a- (תקראו עליו בתיעוד), שגם הוא צריך להשתנות בעקבות השינויים שעשיתם בשני השלבים הקודמים.

חלק ב – Linker scripts

מבוא

חלק זה בתרגיל דורש היכרות גם עם תרגול 9. הסתכלו על התיעוד של ld, הידוע בשמו ב-GNU Linker, והתמקדו בחלק על Linker Scripts, בו ניעזר בחלק זה של תרגיל הבית.

מה זה Linker Scripts? tl;dr – מדובר בקובץ הגדרות ללינקר, בעזרתו שולטים על הדרך בה הוא מבצע את הקישור ומייצר את קובץ הריצה (כולל הגדרות הטעינה). ל-GNU Linker יש קובץ default בו הוא משתמש בריצה רגילה, אך ניתן להעביר לו קובץ אחר, עם הגדרות אחרות, כך שהקישור יתבצע כפי שאנחנו רוצים (העברה של קובץ אחר מבטלת את השימוש בקובץ ה-default). זה חלק ממה שנעשה בתרגיל.

איך עושים זאת? מייצרים קובץ ld. (קיבלתם אחד כזה, אותו תוכלו לערוך ולשנות בהתאם להוראות התרגיל) ומכניסים אותו להרצת הלינקר עם הדגל T-, כפי שתראו בדוגמאות בהמשך.

בחלק זה נלמד לבצע שני דברים:

1. לייצר קובץ ריצה בו החלק בזיכרון אליו נטען text section הוא בעל הרשאות לכתיבה.
2. להוסיף קובץ "זדוני" בשלב הקישור, שייכנס לתוך קובץ הריצה הסופי וישנה את הקוד בזדוניות בזמן ריצה (וזה יעבוד בזכות מה שנעשה בשלב 1)

נשמע מגניב? גם אם לא, בואו נתחיל. זה יהיה מגניב.

שלב ראשון – קביעת אזורי הטעינה

קיבלתם קובץ בשם hw3.ld. כאמור, זהו קובץ Linker Script. הקובץ הזה, בתצורה הנוכחית, מגדיר ל-Linker איך לבצע קישור מאוד בסיסי של מס' קבצים המכילים sections מסוג text, data ו-bss בלבד. קראו על PHDRS והבינו כיצד לקבוע את הגדרות הטעינה של הלינקר, כך שכל האזורים בקוד (text, data, bss) ייטענו לזיכרון עם הרשאות קריאה, כתיבה והרצה.

דוגמה: לרשותכם 2 קבצים, a.o ו-b.o. שניהם נוצרו בעזרת GNU Assembler (as המוכר והטוב) ומחכים להיקשר יחד לקובץ ריצה. הרצה רגילה של פקודת ld הייתה מייצרת תוצאה כפי שמופיעה משמאל, אך אנו רוצים תוצאה כמו בימין:

```
student@ubuntu18:~/HW3/part2$ ld a.o b.o -o hw3.out
student@ubuntu18:~/HW3/part2$ readelf -l hw3.out

Elf file type is EXEC (Executable file)
Entry point 0x4000b0
There are 2 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x000000000000010a 0x000000000000010a  R E    0x200000
  LOAD           0x000000000000010a 0x000000000060010a 0x000000000060010a
                 0x0000000000000012 0x0000000000000012  RW     0x200000

 Section to Segment mapping:
  Segment Sections...
   00     .text
   01     .data
```

```
student@ubuntu18:~/HW3/part2$ ld a.o b.o -o hw3.out -T hw3.ld
student@ubuntu18:~/HW3/part2$ readelf -l hw3.out

Elf file type is EXEC (Executable file)
Entry point 0x4000b0
There are 2 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x00000000000000b0 0x00000000004000b0 0x00000000004000b0
                 0x000000000000005a 0x000000000000005a  RWE    0x200000
  LOAD           0x000000000000010a 0x000000000060010a 0x000000000060010a
                 0x0000000000000012 0x0000000000000012  RWE    0x200000

 Section to Segment mapping:
  Segment Sections...
   00     .text
   01     .data
```

כלומר, אנו רוצים מצב בו ה-Linker הורץ עם ה-Linker Script שלנו, שגורם לשני ה-segments הנטענים להיות עם הרשאות מלאות של קריאה, כתיבה והרצה.

שלב שני – החדרת קובץ "זדוני" לקישור

קיבלתם קובץ בשם hw3_hook.asm, כאשר ב-text section שלו קיימת פקודת exit(0) בלבד, תחת התווית hook. בשלב זה, נסו להחדיר את הקובץ הזה, כך שהתווית hook תהיה הראשון לרוץ. שימו לב – עליכם לעשות זאת באמצעות ה-Linker Script בלבד, ובפרט מבלי לשנות את הקבצים a.o ו-b.o. בשלב זה, אין צורך לשנות גם את הקובץ hw3_hook.asm. את זה נעשה בשלב הבא.

דוגמה: עבור הקבצים הקיימים, הרצה רגילה נראית כך (התווית _start היא הראשונה לרוץ, כפי שאנו מכירים):

```
student@ubuntu18:~/Spring21/HW3$ as hw3_hook.asm -o hw3_hook.o
student@ubuntu18:~/Spring21/HW3$ ld a.o b.o hw3_hook.o -o hw3.out
student@ubuntu18:~/Spring21/HW3$ ./hw3.out
hello
bye
```

אבל לאחר השינוי שנעשה ב-hw3.ld, שיגרום ל-hook להיות התווית הראשונה לרוץ, נקבל:

```
student@ubuntu18:~/Spring21/HW3$ ld a.o b.o hw3_hook.o -o hw3.out -T hw3.ld
student@ubuntu18:~/Spring21/HW3$ ./hw3.out
student@ubuntu18:~/Spring21/HW3$
```

כלומר, הריצה הסתיימה ללא ההדפסות (כי היא יצאה מיד, בגלל ש-hook הייתה התווית הראשונה לרוץ).

שלב שלישי – היכרות עם a.o ו-b.o

עליכם לערוך היכרות מקרוב עם הקובץ a.o. שימו לב שהקובץ הזה הוא היחיד שמובטח לכם שייראה בדיוק כפי שהוא גם בטסטים (מלבד מה שכתוב בהערות בסוף שלב זה). הקובץ משתמש בקובץ b.o ואולי קבצים נוספים. אלו אינם בעלי מבנה או מימוש ידוע מראש (מלבד העובדה שיכילו רק sections מסוג text, data ו-bss). אתם יכולים להסתכל גם על b.o כדי להבין מה קורה בקוד כולו, אך כאמור זהו אינו קובץ שאתם יכולים להסתמך על תוכנו בזמן הטסטים.

<u>רמז:</u> היעזרו בפקודת objdump. אם תצטרכו, גם ב-hexdump ו-readelf ניתן להשתמש (אך לא הכרחי).

<u>הערות:</u>

1. אסור לשנות את הקובץ a.o, בגלל שאתם לא מגישים אותו. עליכם להשתמש ב-a.o הנתון בלבד.
2. אין התחייבות על תוכן, או אורך המחרוזת msg! להסתמך על אחד מהם בשלב הרביעי יביא לכישלון בטסטים.
3. אין התחייבות על השמות של labels בקבצים אחרים (למשל, השם ending). בטסטים יכול להיות שהשם של הפונקציות בקבצים האחרים ישתנו ובהתאם גם הקריאה ב-a.o תשתנה כדי לא לשנות את הלוגיקה שלה. הפקודות עצמן ב-a.o לא ישתנו, רק יעדן (אולי) ישתנה.

שלב רביעי ואחרון – כתיבת הקוד ה"זדוני"

בשלב הזה נכתוב קוד בקובץ hw3_hook.asm אשר ינצל את העובדה ש-hook היא התווית הראשונה לרוץ (שלב שני), את המבנה של הקובץ a.o (שלב שלישי) ואת העובדה שכעת ניתן לכתוב בזמן ריצה לכל האזורים (שלב ראשון), על מנת לייצר התנהגות חדשה.

בקובץ hw3_hook.asm יש מחרוזת ב-data section. המחרוזת הזו נשלחה אליכם מאגודת המעריצים של אלה, לאחר שראו את הביצועים המרשימים שלכם בחלק א' של תרגיל זה.

עליכם להצליח לייצר את ההתנהגות הבאה:

1. תחילה, יודפס למסך התוכן של msg מהקובץ a.o (כפי שגם קורה ב-start_ בהתחלה).
2. כעת תודפס המחרוזת שלכם, זו שנתונה ב-msg בקובץ hw3_hook.asm.
3. ואז תתבצע הפונקציה ending (כפי שגם קורה ב-start_. במקור זה קורה מיד לאחר שלב 1, אך אנו נדחוף את שלב 2 לפני). **אתם לא צריכים לקרוא ל-ending לבד. רק לחזור לקוד המקורי שיעשה זאת בכל מקרה.**
4. יציאה תקינה מהתוכנית (כפי שקורה ב-start_ לאחר הקריאה ל-ending גם כך).

את ההתנהגות הזו תייצרו בעזרת שינויים ב-Linker Script (שכבר עשיתם בשלבים קודמים) ובעזרת עריכת ה- text section של hw3_hook.asm לצרכים שלכם.

עבור הקבצים שקיבלתם, ההתנהגות הצפויה היא כזו –

```
$ ld a.o b.o -o hw3.out
$ ./hw3.out
hello
bye
$ as hw3_hook.asm -o hw3_hook.o
$ ld a.o b.o hw3_hook.o -o hw3.out -T hw3.ld
$ ./hw3.out
hello
This code was hacked by Ella Lee's gang
bye
```

הידד! הקוד שלכם נכנס לקובץ הריצה הסופי ועשה את עבודתו בצורה נקיה, מבלי לפגוע בקבצים המקוריים. בכך סיימתם את התרגיל (והוכחתם לאלה שאתם מעריצים נאמנים 😊)

כעת עליכם להגיש שני קבצים בדיוק – את hw3.ld ואת hw3_hook.asm. יחד הם מחזיקים את כל מה שעשיתם בחלק ב' של תרגיל זה.

חלק ג' - הוראות הגשה לתרגיל בית רטוב 3

אם הגעתם לכאן, זו בהחלט סיבה לחגיגה. אך בבקשה, לא לנוח על זרי הדפנה ולתת את הפוש האחרון אל עבר ההגשה – חבל מאוד שתצטרכו להתעסק בעוד מספר שבועות מעכשיו בערעורים, רק על הגשת הקבצים לא כפי שנתבקשתם. אז קראו בעיון ושימו לב שאתם מגישים את כל מה שצריך ורק את מה שצריך.

עליכם להגיש את הקבצים בתוך zip אחד:

hw3_wet.zip

בתוך קובץ zip זה יהיו 2 תיקיות:

- part1
- part2

ובתוך כל תיקייה יהיו הקבצים הבאים (מחולק לפי תיקיות):

- part1:
  - readelf.c
- part2:
  - hw3.ld
  - hw3_hook.asm

בהצלחה!!!