הטכניון – מכון טכנולוגי לישראל

מעבדה במערכות הפעלה 046210

תרגיל בית מס' 1

הקדמה

ברוכים הבאים לתרגיל מס' 1 במעבדה במערכות הפעלה.

במסגרת תרגיל זה, נכיר את סביבת העבודה בקורס, ונבנה מנהל התקן פשוט.

סביבת עבודה

במעבדה זאת נריץ GNU/Linux ("לינוקס") על מכונה וירטואלית. לשם כך תצטרכו:

- VMWare Player מותקן על המחשב.
- מכונה וירטואלית שמכילה מערכת הפעלה עם גרעין של Linux.

VMWare Player הינו "נגן" של מכונות וירטואליות שמריץ מערכת הפעלה אחרת כאפליקציה על המחשב שלכם. מכונה וירטואלית היא אוסף קבצים בו שמור המצב של מערכת ההפעלה שרצה כאפליקציה במחשב שלכם.

הכנה של סביבת העבודה

כשעובדים בבית

צריך לעשות את ההכנה רק פעם אחת.

1. יש להוריד את קובץ ההתקנה של VMWare Player מהאתר של VMWare, ולהתקינה על המחשב.
2. יש להוריד את קובץ המכונה הווירטואלית RedHat VMware Image, מאתר הקורס.
3. יש לפתוח את ה zip לתיקייה עם הרשאות כתיבה (במחשב שלכם).

כשעובדים בחוות מחשבים

צריך לעשות את ההכנה כל פעם מחדש כאשר נכנסים למחשב עם חשבון שלכם.

1. יש להוריד את קובץ ההתקנה של VMWare Player מהאתר של VMWare, ולהתקינה על המחשב.
2. יש להוריד את קובץ המכונה הווירטואלית RedHat VMware Image, מאתר הקורס.

3. יש לפתוח את קובץ המכונה הווירטואלית RedHat.zip אל תיקייה temp שבדיסק C (רק בתיקייה temp יש מספיק מקום והרשאת כתיבה).

שימו לב: שינויים שבצעתם במכונה הווירטואלית לא נשמרים אחרי הניתוק מהחשבון! שימרו את הקבצים שעליהם עבדתם בתיקייה תחת דיסק Z, כך תוכלו להמשיך לעבוד על הקבצים המעודכנים אחרי שתתחברו מחדש.

הרצה של מערכת הפעלה

אחרי שהכנתם את סביבת העבודה אפשר להריץ את המכונה הווירטואלית.

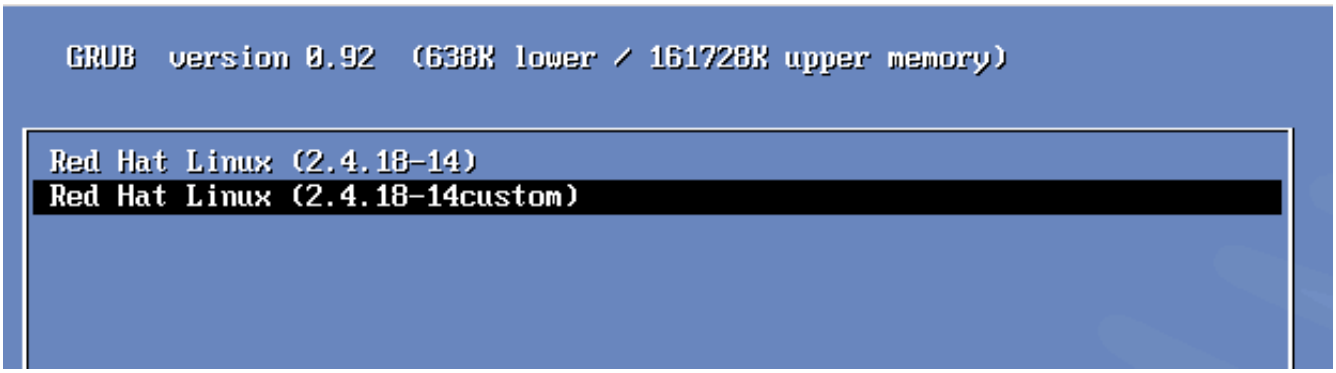
1. הריצו את VMWare Player.
2. בחרו ב-Open כדי לפתוח את המכונה הווירטואלית.
3. בחרו את התיקייה בה נמצאת המכונה הווירטואלית ופתחו את הקובץ המזוהה ע"י VMWare Player.
4. אם בעת ההפעלה אתם נשאלים אם העתקם או העברתם את הImage, בחרו ב"I Copied it".

שימוש במכונה הווירטואלית

אחרי שהרצתם את המכונה הווירטואלית תקבלו חלון בו מודפסות הודעות שונות בזמן טעינת מערכת ההפעלה.

בחירה של גרעין (***Kernel***):

התפריט הראשון שמוצג, נקרא GRUB menu, מפרט את הגרעינים שמותקנים במערכת ההפעלה הווירטואלית.



השורה ראשונה היה הגרעין מקורי, ללא שינויים. נשתמש בגרעין זה כדי לקבל סביבה יציבה, בה יהיה נוח לערוך ולקמפל שינויים בגרעין הניסיוני. השורה שנייה הינה custom kernel , הגרעין בו תעשו שינויים במסגרת המעבדה. אליו תבחרו להיכנס כדי לבדוק את השינויים שעשיתם.

שימו לב – ייתכן ובעקבות השינויים שתבצעו במערכת (התקנת גרעין ששיניתם), סדר השורות יתהפך.

כניסה למערכת ההפעלה

אחרי שבחרתם את הגרעין, יודפס פירוט של תהליך ה-boot למסך. בסיום התהליך, תתבקשו להתחבר למערכת. ניתן להיכנס כמשתמש "רגיל" תחת שם משתמש: user. ניתן להיכנס כמשתמש "מנהל" תחת שם משתמש: root. הסיסמא עבור שני המשתמשים הינה: 046210.

שימוש בתיקיות משותפות

בשביל לפשט את העברת המידע בין המכונה הווירטואלית והמחשב המארח, מערכת ה-VMWare מממשת מנגנון של תיקיות משותפות. תיקיה משותפת הינה תיקייה שתוכנה זהה גם במערכת ההפעלה הווירטואלית (RedHat Linux) וגם במערכת ההפעלה של המחשב שלכם (ה"מארח" בעולם המונחים של VMWare). כל שינוי שיבוצע בתיקייה, במערכת ההפעלה של ה"מארח" או במערכת ההפעלה הווירטואלית, יופיע מידית בתיקייה במערכת ההפעלה השנייה.

בשביל שהתיקייה המשותפת תעבוד, צריכים להתקיים מספר תנאים:

- במכונה הוירטואלית (ה-Linux) צריכים להיות מותקנים הדרייברים של VMWare. קבצים אלו כבר מותקנים על המכונה הווירטואלית שברשותכם.

- ה-VMWare Player צריך להיות מוגדר למצב של "תיקיות משותפות". בתפריט של VMWare Player, תחת הסעיף Shared Folders, יש לסמן את Always enabled ותחת properties לבחור את התיקייה במחשב שלכם (המארח) שאותה אתם רוצים לשתף.

במקרה והמכונה כבר פעילה, ייתכן והמערכת לא תאפשר לעדכן את הגדרות השיתוף. במקרה זה, בצעו אתחול למכונה הווירטואלית (ב shell הקישו "reboot"). לאחר האתחול, מיפוי התיקיות אמור להתעדכן.

בתוך מערכת ההפעלה הווירטואלית התיקייה המשותפת מופיעה תחת:

/mnt/hgfs/Shared/

במחשב שלכם זאת התיקייה שבחרתם בהגדרה של תיקיות משותפות.

לפעמים נוח לעבוד על קבצים בתיקייה המשותפת מתוך המחשב המארח עם editor אליו אתם רגילים, ואחר כך להשתמש בקבצים הנ"ל מתוך המערכת הווירטואלית. יחד עם זאת, אנחנו ממליצים לתרגל שימוש בעורכי טקסט מבוססי יוניקס, כגון emacs.

שימוש ב ***shell***

אפשר לפתוח חלון shell ע"י קליק ימני בשולחן העבודה ובחירה ב-New Terminal.

כדי להיזכר בסביבה של shell אתם יכולים לקרוא תזכורת קצרה בשקפי התרגולים של הקורסים מערכות הפעלה וממ"ת. בנוסף, קיים חומר רב בנושא ה- shell באינטרנט.

חומר רקע

Linux Kernel

הגרעין הינו החלק העיקרי של מערכת ההפעלה, ומהווה את הגשר בין האפליקציות שרצות במחשב לחומרת המחשב. בין היתר הגרעין אחראי על ניהול התהליכים (Processes/Tasks), ניהול הזיכרון, רכיבי החומרה השונים והתקשורת. גרעין לינוקס החל בתחילת שנות התשעים כפרויקט של סטודנט בשם לינוס טורבאלדס. לינוס היה מעוניין לפתח מערכת הפעלה תואמת Unix על גבי מעבד 386. לינוס הפיץ את קוד הגרעין בצורה חופשית ועד מהרה הקוד צבר משתמשים רבים בכל רחבי העולם. כיום גרעין הלינוקס מפותח כפרוייקט קוד חופשי על ידי מספר מפתחים בכל העולם כשלינוס משמש כמפתח הראשי. קיימות גרסאות של הגרעין למספר רב של פלטפורמות (אפילו שעוני יד) והוא נחשב לאחת הדוגמאות המוצלחות של פרויקט קוד חופשי (לשם קנה מידה, לאחרונה הוערכה עלות פיתוח של גרעין לינוקס בקרוב לשני מיליארד דולר). הפצות לינוקס השונות (Ubuntu, Fedora, Suse וכדומה) משתמשות בגרעין לינוקס (עם שינויים קלים) עליו הן מוסיפות ממשק גרפי (Gnome & KDE desktops וכדומה) וכן אפליקציות שונות.

מרחבי ריצה

הקוד הרץ על מחשב יכול לרוץ תחת שתי רמות הרשאה שונות :

- הרשאות של מרחב המשתמש.
- הרשאות של מרחב הגרעין.

במרחב המשתמש (***User Space***):

תוכנות רגילות (עורכי תמלילים, נגנים וכלים נוספים) רצים במרחב המשתמש. הרשאות של קוד הרץ במרחב המשתמש:

- יש גישה למרחב הזיכרון של האפליקציה – אין גישה למרחב הזיכרון של גרעין.
- קבוצה מצומצמת של פקודות שהמעבד רשאי לבצע.
- יש הגבלות על הגישה לרגיסטרים מסוימים (למשל לרגיסטרים שמגדירים הרשאות של הקוד).
- אין גישה ישירה להתקני חומרה חיצוניים.

במרחב הגרעין (***Kernel Space***):

קוד של גרעין מערכת ההפעלה רץ תחת הרשאות מרחב הגרעין. קוד הגרעין כולל מימוש של קריאות מערכת, וכן של מנגנוני ליבה שונים, כגון מתזמן תהליכים (scheduler) ומנהלי התקני חומרה (device drivers).

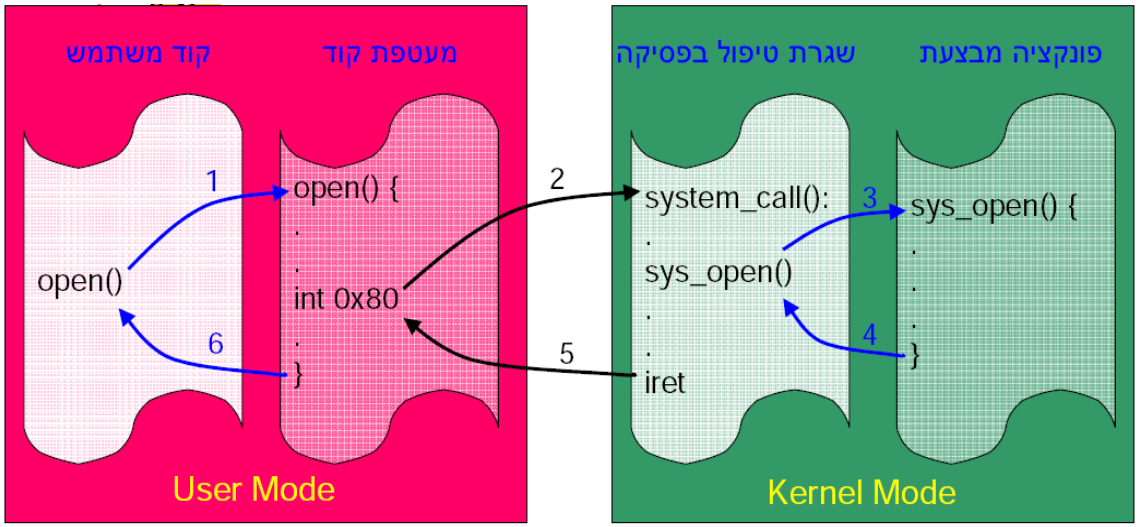
- יש גישה למרחב הזיכרון של הגרעין. לרוב יש גם גישה למרחב הזיכרון של האפליקציה.
- המעבד רשאי לבצע מספר גדול יותר של פקודות (לרוב, כל הפקודות הנתמכות).
- כמעט ואין מגבלות על הגישה לרגיסטרים.
- יש גישה מלאה להתקני החומרה השונים.

ממגוון סיבות, בעיקר אבטחה ויציבות המערכת, רוב הקוד שאנו עושים בו שימוש רץ במרחב המשתמש. לפעמים הקוד צריך שירות שאפשר לבצע רק עם הרשאות של מרחב הגרעין. למשל קריאה/כתיבה לקובץ, דגימת הזמן הנוכחי, יצירת תהליך חדש וכדומה. כדי לעבור ממרחב המשתמש למרחב הגרעין הקוד עושה שימוש במנגנון *קריאת מערכת.*

קריאות מערכת

בציור הבא, הקוד בריבוע האדום רץ עם הרשאות של מרחב המשתמש. הקוד בריבוע הירוק רץ עם הרשאות של מרחב הגרעין. לקוד זה יש גישה להתקני חומרה ולמבני נתונים של גרעין.

במערכות x86 ישנות (כמו זו שאנו נבצע עליה את המעבדה), שינוי הרשאות *מרחב משתמש* ↔ *מרחב הגרעין* נעשה ע"י פקודות *int 0x80* ו *iret*. (פעולות 2 ו 5 בציור). במעבדים מודרניים, ישנן פקודות מיוחדות לביצוע המעבר הנ"ל – SYSENTER ו-SYSEXIT.



כתוצאה מביצוע פקודה *int 0x80* המעבד מחליף אוטומטית את ההרשאות של הקוד להרשאות של מרחב הגרעין ומעדכן מצביע של ביצוע קוד לתחילת הפונקציה *system_call*. כאשר גרעין מערכת ההפעלה מסיים את פעולתו, הוא משתמש בפקודה *iret* כדי להחליף את ההרשאות של הקוד להרשאות של מרחב המשתמש ומעדכן מצביע של ביצוע קוד לפקודה הבאה אחרי *int 0x80*.

LKM – Loadable Kernel Modules

הגרעין של מערכת ההפעלה linux בנוי בצורה מודולרית. ישנו הגרעין הבסיסי, base kernel, הממש פונקציות בסיסיות כגון ה-Scheduler, וישנן תכניות נוספות הנקראות מודולים, kernel modules. המודולים משמשים להרחבת הפונקציונליות של מערכת ההפעלה, כגון חיבור לחומרה חיצונית, גישה למערכות קבצים שונות וכדומה. חלק מהמודולים מקושרים לגרעין ונטענים

אתו. אחרים, Loadable Kernel Modules, נטענים ומוסרים בצורה דינאמית בהתאים לצורך. הדבר מאפשר לשמור על גרעין בסיס מצומצם הדורש משאבים מעטים בלבד מהמערכת ולהרחיבו רק כשיש בכך צורך.

בתרגיל בית זה, אתם נדרשים לכתוב LKM פשוט הממש מנהל התקן מסוג Char Driver. הדבר יאפשר לכם להתנסות בעקרונות תכנות של Linux kernel, ללא צורך להדר את הגרעין עצמו (בניגוד לתרגילים הבאים).

מנהלי התקן, קבצי התקן ו-*IOCTL*

ניתן להוסיף ממשק לגרעין המערכת על ידי הוספת קריאת מערכת (system call). אולם, הוספת קריאת מערכת דורשת הרשאות, הידור מחדש של הקרנל, ויכולת לצלול את הקוד של מערכת ההפעלה. לכן, נהוג לעתים להוסיף ממשקים לגרעין המערכת באופן שונה – על ידי יצירת "קבצי התקן" (device files). הסיבה שמנהלי ההתקן ממומשים כקבצים נובעת מהגישה של מערכת ההפעלה יוניקס: כל דבר (כמעט) הוא קובץ. קבצי ההתקן מייצגים התקנים שונים (זיכרון נשלף, מסך, רמקולים ועוד), וניתן (בתלות במימוש מנהל ההתקן) לפתוח אותם, לקרוא מהם, לכתוב אליהם ולבצע עליהם פעולות נוספות. לשם ביצוע פעולות נוספות, הוגדר במערכות Unix (שלינוקס תואמת להן) ממשק בשם IOCTL. ממשק זה מאפשר לתוכנת המשתמש להעביר בקשות "חריגות" למערכת ההפעלה, שלא נכון היה להעביר אותן כפעולת כתיבה או קריאה.

למשל, באם יש לנו קובץ אשר מתאר כרטיס קול, ברור כי הממשק הטבעי הוא שכתיבה תוציא קול מהכרטיס, לפי המידע שכתבנו, קריאה תעביר לתוכנה שלנו את דגימות הקול מהמיקרופון המחובר לכרטיס. כדי להגדיר פרמטרים אחרים של כרטיס הקול (כגון קצב הדגימה, הפורמט שבו אנו דוגמים ועוצמת הנגינה) עושים שימוש במנגנון ה-IOCTL. מנגנון זה הוא למעשה syscall, שמקבל 3 פרמטרים:

1. מזהה של הקובץ אליו אנחנו רוצים לבצע IOCTL (לדוגמא, הקובץ של כרטיס הקול).
2. מספר (מזהה) של ה-IOCTL שאנחנו רוצים לבצע (לדוגמא, קביעת קצב הדגימה).
3. מצביע למבנה נתונים המכיל מידע נוסף לשימוש מנהל ההתקן (לדוגמא, קצב הדגימה בו אנו מעוניינים).

מבני נתונים של הגרעין

כאשר אנחנו במרחב הגרעין, אפשר לגשת למבני נתונים של הגרעין. לדוגמא, לכל תהליך במרחב הגרעין ישנה רשומה (מתאר תהליך) מטיפוס task_struct שמחזיקה את הידע על התהליך:

1. מזהה התהליך (PID)
2. מצב התהליך ועדיפותו
3. מצביע למתארי תהליכים נוספים (רשימה מקושרת)
4. מצביע לטבלת אזורי זיכרון התהליך
5. מצביע לטבלת קבצים הפתוחים של התהליך
6. ועוד...

את ההגדרה של טיפוס הנתונים task_struct המחזיק מידע זה אפשר לראות בקובץ:

/usr/src/linux-source-2.4.18custom/include/linux/sched.h

ניתן לגשת למבנה הנתונים של התהליך הנוכחי (התהליך שמבצע את הקוד) באמצעות המאקרו current. לדוגמה:

```
task_struct* p = current;
```

כדי להשתמש במאקרו currentצריך לצרף את השורה הבאה לקובץ התוכנית:

```
#include <linux/sched.h>
```

פרוט התרגיל שיש להגיש

בתרגיל זה יש ליצור מנהל התקן מסוג Char Device אשר יטען ב-LKM. לפני המשך קריאת פרוט התרגיל יש לקרוא את ההסבר על מנהלי ההתקנים (תחת סעיף מידע שימושי שבהמשך).

מנהל ההתקן ישמש לצורך ערבול (hashing) של תוכן טקסטואלי. פעולת הערבול תתבצע באמצעות MurmurHash אשר מסופקת כחלק מקבצי העזר לתרגיל. ערבול טקסט הינה שיטה לבצע חיפוש יעיל בHashTables. לדוגמא ב- MurmurHash משתמשים כדי לחפש מפתחות בmemcached, אפליקציית שרת פופולארית.

מנהל ההתקן ינהל כתיבה וקריאה אל תוך חוצץ (buffer) בזיכרון. גודל החוצץ הינו קבוע, 4096 byte. לכל קובץ התקן

(device file) המשויך למנהל ההתקן יוקצה buffer נפרד בצורה דינמית. משך חיי ה buffer-הוא כאורך חיי קובץ ההתקן. על מנהל ההתקנים ליישם את הממשק הבא אשר מאפשר כתיבות לתוך הbuffer וקריאות אשר מחזירות את ערך הhash של התוכן הנקרא משורשר לתוכן הנקרא עצמו:

- כתיבה - כל כתיבה תירשם לתוך ה-buffer בהמשך לכתיבה הקודמת (ללא דריסת כתיבות קודמות).

  אין לאפשר כתיבה מעבר לגודל הbuffer. בעת ניסיון לכתוב מעבר לגודל ה-buffer יש להחזיר הודעת שגיאה -ENOMEM (Out of memory). אין לבצע כתיבה חלקית במקרה זה.

  לא ניתן לכתוב מחרוזת באורך 0. במקרה של ניסיון לכתוב מחרוזת באורך 0 יש להחזיר הודעת שגיאה: EINVAL -.

  ההתקן יחזיר את מספר הנתונים שהועתקו ל-buffer.

- קריאה – כל קריאה מתוך ה-buffer תתחיל מסוף הקריאה הקודמת ועד לגודל המבוקש. שימו לב כי <u>אין לאפשר קריאה מעבר לנתונים שנכתבו עד כה ל-buffer</u>. ההתקן יעתיק את כמות הנתונים שנתבקש או את כמות הנתונים שנכתבה (ועדיין לא נקראה), הקטנה מבניהן. ההתקן יחזיר את ערך הhash עבור התוכן הנקרא משורשר לתוכן הנקרא מה-buffer.

  במקרה של שגיאה בהעתקה בין מרחב ה-User וה-Kernel יש להחזיר הודעת שגיאה בעלת ערך -ENOMEM (אין זיכרון). אין לבצע קריאה חלקית במקרה זה.

  ההתקן יחזיר את סכום מספר הנתונים שנקראו מהbuffer וגודל תוצאת פונקציית הhash.

- פעולת RESTART מאתחלת את מיקום <u>הקריאה</u> האחרונה לתחילת ה-buffer. כלומר, הקריאה הבאה ממנהל ההתקן תתחיל מתחילת ה-buffer. הנתונים הקיימים ב-buffer לא משתנים.

- פעולת RESET מאתחלת את פעולות <u>הכתיבה והקריאה</u> אל ה-buffer. כלומר פעולות הכתיבה והקריאה הבאות יתחילו מתחילת ה-buffer. הנתונים הקיימים ב-buffer 'ימחקו' (אין צורך במחיקה של ה-buffer).

- פעולת SET_SEED קובעת את הSEED שפונקציית הhash משתמשת בו. עליכם לאתחל את ערך הseed ל-0 עד שהוא משתנה ע"י הפעולה SET_SEED.

יש לממש את פעולות ה-RESET, SET_SEED וה-RESTART על ידי מנגנון ה-IOCTL לפי הפרוט הבא (ראה קישור תחת מידע שימושי) :

```
#define MY_MAGIC 'r'
#define MY_RESET _IOW(MY_MAGIC, 0, int)
#define MY_RESTART _IOW(MY_MAGIC, 1, int)
#define MY_SET_SEED _IOW(MY_MAGIC, 2, int)
```

שם מנהל ההתקן צריך להיות s19_device. מספר major של ההתקן צריך להינתן באופן דינמי. קבצי התרגיל מכילים שלד (my_module.h, my_module.c) עליו ניתן לבנות את מנהל ההתקן.

מידע שימושי

- הסבר על מנהלי התקן מסוג Char Driver ניתן למצוא בתירגול 10 של מבנה מערכות הפעלה בקישור הבא וכן בקישור הבא.
- אפשר למצוא הסבר על איך להוסיף תמיכה ב- IOCTL בקישור הבא.
- ניתן, ורצוי ,לדבג את המודול בעזרת פקודת printk. אפשר לקרוא על שימוש ב-printk בקישור הבא.
- הגרעין לא מקופל אל מול הספריות הסטנדרטיות (GNU CLib). לכן לא ניתן לעשות שימוש בפונקציות מתוך 'string.h', 'stdio.h', 'stdlib.h' וכדומה.
- הגרעין עושה שימוש בזיכרון פיזי. לשם הקצאת זיכרון דינמי בגרעין יש לעשות שימוש בפקודות kmalloc ו-kfree (השימוש בהן זהה לשימוש בפקודות malloc ו-free).
- זיכרון השייך למרחב הגרעין עובר תרגום שונה מזיכרון השייך למרחב המשתמש. לכן, קוד מנהל ההתקן לא יכול לגשת ישירות למידע הנמצא במרחב הזיכרון של המשתמש.

  בעקבות זאת עליכם לעשות שימוש בשגרות copy_to_user, copy_from_user על מנת להעביר מידע בין המרחבים. ניתן לקרוא עליהן בקישור הבא.

בדיקה של מנהל ההתקן

כדי לקמפל את מנהל ההתקן יש להשתמש בפקודה הבאה (לחלופין ניתן להשתמש בקובץ Makefile המצורף):

```
gcc –c –I/usr/src/linux-2.4.18-14/include –Wall my_module.c
```

לשם בדיקת מנהל ההתקן יש להתקין אותו ולבדוק האם הוא מבצע נכון את כל הפעולות הנדרשות ממנו. טעינת מנהל ההתקן נעשית בעזרת הפקודה:

```
insmod ./my_module.o
```

כדי ליצור קובץ התקן המשויך למנהל ההתקן ניתן להיעזר בפקודה mknod. לדוגמא, הפקודה הבאה תיצור קובץ התקן בשם s19_device המזוהה על ידי מספר מג'ורי 0:

```
mknod /dev/s19_device c major 0
```

במקום major יש לרשום את מספר ה-major שנבחר להתקן שלכם. ניתן למצוא את מספר ה-major שנבחר להתקן בקובץ /proc/devices.

הסרת ההתקן נעשית בעזרת הפקודות:

```
rm -f /dev/s19_device
rmmod my_module
```

מומלץ לכתוב סקריפטים שיבצעו את הפעולות הנ"ל בצורה אוטומטית.

בדיקת תפקוד מנהל ההתקן תתבצע על ידי פתיחת קובץ ההתקן, ביצוע הפעולות המפורטות לעיל ובדיקת התוצאות שלהן. ניתן לבצע זאת בעזרת תכנית בשפת C. כמו כן ניתן גם לעשות זאת בשפת סקריפט כגון Python שמאפשרת פיתוח מהיר ונוח של תכניות. תיעוד שפת Python נמצא בקישור הבא, הסבר על גישה לקבצים נמצא בקישור הבא ועל ביצוע פניות IOCTL ניתן לקרוא בקישור הבא. בין קבצי התרגיל מצורף קובץ בדיקה לדוגמא: test.py.

אין צורך להגיש את התכנית בה השתמשתם לבדיקת מנהל ההתקן שלכם.

הוראות הגשה לתרגיל:

- תאריך הגשה: 17 באפריל 2019 עד השעה 23:55.
- הגשה אלקטרונית דרך אתר הקורס.
- הגשה בזוגות בלבד. אפשר להשתמש בפורום באתר הקורס למציאת שותפים.
- יש להגיש דרך חשבון של אחד השותפים בלבד.(אין להגיש פעמיים - מכל חשבון).

ההגשה בקובץ **zip** בשם id1_id2.zip , כאשר id ,1id2 הם מספרי ת.ז. של השותפים. הקובץ יכיל:

- קבצי התוכנית: my_module.h, my_module.c
- קובץ טקסט submitters.txt עם הנתונים של המגישים לפי המתכונת הבאה:

first_name1 last_name1 id1
first_name2 last_name2 id2

יש להקפיד שקובץ ה zip לא יכיל קבצים נוספים, לרבות תיקיות. דהיינו, על ההגשה להיות בצורה הבאה:

```
zipfile -+
         |
         +- submitters.txt
         +- my_module.c
         +- my_module.h
         |
```

דגשים לגבי הציון

- משקל התרגיל הינו 25% מהציון הסופי.
- הקפידו על מילוי הדרישות לשמות הקבצים והממשקים. כל טעות הגשה תוריד **5** נקודות מהציון על התרגיל.
- התרגילים יבדקו אוטומטית. עבודה אשר אינה עוברת את הבדיקה האוטומטית, הציון עבורה יהיה 0.
- יש להקפיד על סדר ותיעוד הקוד.
- על כל יום איחור יופחתו 4 נקודות, עד לתקרה של 24 נקודות.