תרגיל בית יבש 4

תאריך הגשה: יום שישי19/06/2015 , 12:30 בצהריים

המתרגל האחראי על התרגיל: עידו זמירי

שאלות ותשובות לתרגיל בית זה יינתנו כרגיל ב[פיאצה](https://piazza.com/technion.ac.il/spring2015/234123/home).

אנו ממליצים להתחיל לעבוד על תרגיל הבית מוקדם ככל האפשר.

המסמך מנוסח בלשון זכר ([אך מיועד לשני המינים](http://hebrew-academy.org.il/2010/10/04/%D7%90%D7%99%D7%9A-%D7%A4%D7%95%D7%A0%D7%99%D7%9D-%D7%9C%D7%A7%D7%91%D7%95%D7%A6%D7%94-%D7%A9%D7%A8%D7%95%D7%91%D7%94-%D7%A0%D7%A9%D7%99%D7%9D/)).

**חלק 1**

1. לאחר שלמדו על סיגנלים התווכחו ביניהם שני סטודנטים בקורס מערכות הפעלה.  
   סטודנט א' טען: "בלינוקס אין קינון של סיגנלים ולכן לעולם לא נצטרך להשתמש במנגנוני סנכרון בעת מימוש הנדלר של סיגנל".  
   ואילו סטודנט ב' טען: "יתכנו מצבים בהם כן נצטרך להשתמש במנגנוני סנכרון בעת מימוש הנדלר של סיגנל"  
   הסבירו מי משניהם צודק.  
   נניח יש 2 חוטים עם מבנה נתונים X בזכרון משותף, וב-SIGNAL HANDLER שלהם כותבים ל-X. נניח חוט A מקבל סיגנל. ה-SIGNAL HANDLER רץ ב-USER MODE ולכן ייתכן ותתקבל פסיקת שעון, וייתכן שכתוצאה מפסיקה זו יוחלף ההקשר לחוט B. נניח גם B מקבל כעת סיגנל (זה לא סותר את אי-קינון הסיגנלים, כי חוט A לא קיבל את הסיגנל) וכותב גם הוא ל-X. כך, יש 2 כותבים אסינכרוניים לאותו מבנה X! לכן סטודנט ב' צודק, יש לסנכרן.
2. לאיזה פקודתshell דומה התוכנית הבאה? הסבירו.
3. close(0);
4. open(“./my\_file”, O\_RDONLY);
5. execv(“/bin/cat”);

פקודה זו שקולה לביצוע cat ./my\_file. הפעלת הפקודה cat ללא פרמטר נוסף גורם לו להמתין לקלט מ-stdin, ובכל תהליך stdin מוצבע ע"י רשומה עם ערך file\_descriptor=0. בביצוע close(0) התהליך מפנה את הרשומה הזו, ובביצוע open האלגוריתם מחפש את הרשומה הפנויה הראשונה שהיא כרגע רשומה 0. לכן, stdin הוחלף ב-my\_file, ולכן cat יבצע בפועל cat ./my\_file.

1. בתכנות בC מניחים שכתובת הזיכרון 0, המסומנת בהרבה מקרים על ידי המאקרו NULL, אינה כתובת חוקית לגישה. הסבירו כיצד מערכת הפעלה יכולה לגלות גישה לכתובת זו, ללא בדיקה בתוכנה של כל כתובת בה משתמשים.

כאשר מערכת ההפעלה מקצה אזורי זיכרון לתהליך, היא קובעת את אזור הזיכרון המכיל את כתובת 0 כאזור לא חוקי לגישה. כך, כפי שבדיקת חוקיות גישה מתבצעת עבור אזורי זכרון אחרים, בדיקה כזו תתבצע עבור גישה לכתובת 0 והמעבד יודיע למערכת ההפעלה שהתבצעה גישה לא חוקית. בשגרת הטיפול, מערכת ההפעלה תבדוק ותבין שהייתה גישה לכתובת 0 ותפעל בהתאם.

1. המאקרו access\_ok מוודא כי טווח הכתובת [to, to+size) מוכל במרחב הכתובת של המשתמש. הסבירו מדוע בדיקה זו הכרחית ולא מספיק להסתמך על מנגנון חריגות הדף.

נניח אנחנו בקוד גרעין עם משתנה X המכיל כתובת וירטואלית שנשתמש בה בביצוע פקודת copy\_to\_user. נניח כתובת X אינה חוקית לכתיבה ע"י המשתמש, אבל חוקית לכתיבה ע"י הגרעין – אם ננסה לבצע copy\_to\_user אז המעבד לא ימנע זאת, כי קוד גרעין ביצע את הפקודה ולכן יש לו הרשאה – אבל קריאת המערכת שביצעה copy\_to\_user התבצעה מקוד משתמש! אם היינו סומכים על מנגנון ה-Page fault, משתמש היה יכול לעבוד עלינו ולהשתמש בקריאות מערכת ע"מ לכתוב לכתובות שאינן חוקיות עבורו.  
לכן, יש להשתמש ב-access\_ok כדי לבדוק מראש האם הכתיבה חוקית למשתמש.

**2 – נכון\לא נכון**

ענו נכון/לא נכון ונמקו. תשובה ללא נימוק לא תתקבל.

1. תהליך בהרשאת root יכול להרוג מודול באמצעות הפקודה kill.

סמנו: נכון \ **לא נכון**  
kill היא פקודה ששולחת סיגנלים לתהליך, לא למודול. להרוג מודול ניתן רק על ידי פקודת rmmod. ניתן להרוג את כל התהליכים שמריצים קוד של מודול עם פקודות kill, אבל זה לא "הורג את המודול"

1. אם התבצעו n קריאות ל - close אז התבצעו גם n קריאות ל - release.

סמנו: נכון \ **לא נכון**  
קריאת release מתבצעת פעם אחת ותפקידה לשחרר את מבנה הנתונים שאותחל open. אבל כל חוט שמשתמש במודול יכול לבצע close. רק בclose האחרון תתבצע release

1. כדי לקבל את ה - pid של התהליך הנוכחי מודול יכול להשתמש בפקודה getpid אך מקובל להשתמש במאקרו current כי זה יעיל יותר.

סמנו: נכון \ **לא נכון**  
getpid היא קריאת מערכת ומוגדרת ב-USERSPACE. היא אינה מוגדרת במרחב הגרעין, וקוד של מודול רץ במרחב הגרעין. לכן, ניתן להשתמש ב-current בלבד.

1. ניתן להשתמש במספרי minor שונים על מנת לבחור איזה אובייקט file\_operations יועבר ל - register\_chrdev על ידי init\_module ובכך להשיג פונקציונליות שונה לפעולות read ו - write.

סמנו: נכון \ **לא נכון**  
לפני ביצוע init\_module עוד לא קיבלנו מספר major. לפני ביצוע register\_chrdev, לא ניתן לבצע פקודת mknod כי המודול עוד לא רשום במערכת. לפני ביצוע mknod, אין מספרי minor בכלל! לכן ל-init\_module לא ניתן בכלל להעביר מספרי minor, שלא לדבר על לבחור אובייקט fops בהתאם.  
כביכול, אפשר לשלוח פרמטרים נוספים ל-init\_module שיכולים "במקרה" להיות מספרים בין 0 ל-255, ושבהמשך השימוש במודול הפרמטרים שנשלחו יהיו קשורים למספרי minor שיווצרו בהמשך. במקרה זה, כן יכול להיות קשר בין מספרי minor (שעוד לא באמת קיימים) לבין פרמטרים שנשלחים ל-init\_module ומשפיעים על איזה אובייקט fops לחבר, **אבל**: זו תהיה פעולה חד פעמית, לכן אחר כך כל תהליך שיפתח התקן יקבל את אותו fops (אלא אם הוא ישתנה בפונקציה אחרת, אחרי init\_module).

1. בעת טעינת מודול, נוצר אוטומטית קובץ ב/dev שאיתו הוא יעבוד. בעת הסרת המודול, הקובץ נמחק גם כן אוטומטית.

סמנו: נכון \ **לא נכון**

טעינת מודול לא פותחת קובץ /dev, זה קורה רק בפתיחת התקן חדש. באותו האופן לגבי ההסרה. הקובץ של ההתקן נמחק כאשר ההתקן נמחק. לא המודול.