מערכות הפעלה - תרגיל בית 2

אוניברסיטת חיפה

סמסטר אביב תשפ"ד

לתרגיל זה שני חלקים - תיאורטי ומעשי.

1. חלק תיאורטי (24 נקודות), בו יהיה עליכם לענות על מספר שאלות תיאורטיות.
2. חלק מעשי (76 נקודות), בו יהיה עליכם לממש דרייבר באמצעות מודול.

חלק תיאורטי

ניתן לנמק את התשובות, אך אינכם מחויבים.

1. תהי P קבוצת כל ה-pid-ים של התהליכים הפעילים ברגע נתון (לינוקס). נגדיר $G = (P, E)$, כך ש-$(p_1, p_2) \in E$ אמ"מ התהליך עם pid p_1 הוא ההורה (real parent) של תהליך עם pid p_2. קבעו את נכונות כל אחת מהטענות הבאות:
   (א) G הוא בהכרח DAG.
   (ב) במידה ונתעלם מכיווני הקשתות, G הוא בהכרח עץ.
   (ג) במידה ונתעלם מכיווני הקשתות, G הוא בהכרח יער.
   (ד) במידה ונריץ BFS מהקודקוד 0, ייתכן שלא נבקר בכל הקודקוד.
2. יהי x משתנה גלובלי שערכו 0, ויהיו p_1, p_2 תהליכים (בלינוקס) שמריצים את הפקודה $\text{printf}\,(\text{"}\%d\text{"}, \&x)$. קבעו את נכונות כל אחת מהטענות הבאות:
   (א) ה-pid של שני התהליכים בהכרח זהה.

(ב) ה-pid של שני התהליכים בהכרח שונה.

(ג) הפלט של שני התהליכים (stdout) בהכרח זהה.

(ד) הפלט של שני התהליכים (stdout) בהכרח שונה.

חלק מעשי

בתרגיל זה תממשו device driver עבור התקן הצפנה (encryption device) באמצעות LKM. בעולם האמיתי, ייתכן שהתקן ההצפנה הוא רכיב חומרתי, שמבצע הצפנה מורכבת באופן מהיר. בתרגיל זה תממשו הצפנה בסיסית שתדמה זאת, ומהירות ההצפנה אינה משנה לנו.

הצפנות

צופן קיסר (Caeser Cipher)

- בהינתן מחרוזת s ומפתח הצפנה k, ההתקן מזיז (shift) את ערכו של כל תו ב-s לערך הנמצא k מקומות אחריו בטבלת ה-ASCII, באופן ציקלי.
- כלומר, עבור מחרוזת s באורך n ומפתח הצפנה k, פעולות הצפנה ופענוח תבוצענה כמתואר באיור 1.

```
// encode
for (i = 0; i < n; i++) {
        s[i] = (s[i] + k) % 128;
}
// decode
for (i = 0; i < n; i++) {
        s[i] = (s[i] + 128 - k) % 128;
}
```

איור 1: צופן קיסר: הצפנה ופענוח.

צופן XOR (XOR Cipher)

- ⊕ בהינתן מחרוזת s ומפתח הצפנה k, ההתקן מחליף כל תו c ב-s בתוצאת ה-XOR של c ו-k: $c \oplus k$.
- ⊕ כלומר, עבור מחרוזת s באורך n ומפתח הצפנה k, פעולות הצפנה ופענוח תבוצענה כמתואר באיור 2.

```
// encode + decode
for (i = 0; i < n; i++) {
        s[i] = s[i] ^ k;
}
```

איור 2: צופן XOR: הצפנה ופענוח.

מימוש מנהל ההתקן

בתרגיל תממשו מנהל התקן שיתמוך בשני סוגי התקני תווים: התקן מסוג Caesar Cipher והתקן מסוג XOR Cipher. עבור כל אחד משני סוגי ההתקנים, מנהל ההתקן ינהל חוצץ נתונים (data buffer) שאליו משתמש הקצה יכתוב מידע כדי להצפין מחרוזות, וממנו יקרא מידע כדי לפענח מחדש את המידע המוצפן. בנוסף ליכולת לכתוב ולקרוא מההתקן, נרצה ששני ההתקנים יאפשרו למשתמש לבצע את הפעולות המיוחדות הבאות:

- ⊕ Set Encryption Key: הגדרת מפתח ההצפנה שישומש להצפנה/פענוח.
  - ⊗ **שימו לב**, תכונה זו צריכה להיות מוגדרת ב-file object שמתאר את הקשר העבודה של המשתמש עם ההתקן.
  - ⊗ כלומר, אם פתחנו את אותו ההתקן (למשל, את Caesar Cipher) באמצעות שתי קריאות open נפרדות (כל אחת החזירה fd שונה), אז יש לאפשר להגדיר encryption key ייחודי עבור כל אחד מה-file objects שמוצבעים על ידי ה-file descriptors.

⊕ Set Read State: עלינו לאפשר למשתמש להגדיר האם כאשר תבוצע קריאה מההתקן, ייקרא המידע המוצפן ללא פענוח (raw read), או שיקרא את המידע המוצפן לאחר פענוח (decrypt read).

⊗ **שימו לב**, גם תכונה זו צריכה להיות מוגדרת ב-file object שמתאר את הקשר העבודה של המשתמש עם ההתקן, בדומה לתכונה הקודמת.

⊕ Zero Buffer: עלינו לאפשר למשתמש לבקש לאפס את חוצץ הנתונים של ההתקן.

עליכם לכתוב kernel module שירשום מנהל התקן חדש בעל מספר major המוקצה דינמית, לממש שני file operation sets: אחד עבור Caesar Cipher, אשר יוגדר ב-Minor 0, ואחד עבור XOR Cipher, שיוגדר ב-Minor 1. להלן קווים מנחים למימוש המודול (תוכלו לממש כרצונכם):

1. הגדירו module parameter בשם memory_size.

⊕ המשתנה יכיל את גודלם של שני החוצצים שיאכסנו את המחרוזת המוצפנת, ויקבע בעת טעינת המודול.

2. ממשו את הפונקציה init_module.

⊕ כפי שראינו בתרגול, פונקציה זו נקראת כאשר המודול נטען לגרעין (למשל עם insmod). מרכיבי המימוש:

(א) רישום מנהל ההתקן ושיוך למספר Major באמצעות register_chrdev.

(ב) הקצאת מקום לשני חוצצים שיאכסנו את המידע שנכתב לשני ההתקנים (גודלם נקבע ע"פ memory_size).

(ג) **שימו לב**, המודול עובד בתוך הגרעין: להקצאת זיכרון בגרעין השתמשו ב-kmalloc (מידע נוסף כאן).

3. ממשו את הפונקציה cleanup_module.

⊕ כפי שראינו בתרגול, פונקציה זו נקראת כאשר מודול נפרק מהגרעין (למשל עם rmmod). מרכיבי המימוש:

(א) הסרת מנהל ההתקן ממספר ה-Major המשויך אליו באמצעות unregister_chrdev.

(ב) שחרור הזיכרון שהוקצה לחוצצים של ההתקנים.

(ג) **שימו לב**, לשחרור זיכרון שהוקצה דינמית בגרעין השתמשו ב-kfree (מידע נוסף כאן).

4. ממשו file operations עבור כל התקן.

⊕ לכל התקן, עליכם לממש את הפונקציות open, release, write, read, ioctl.

⊕ **שימו לב**, למרות שלכל התקן יהיה file operations משלו, ניתן (ואף נוח יותר) לממש את הפעולות באופן כללי כך שיתאימו לשני ההתקנים (למשל, המימוש של open, release ו-ioctl זהה לחלוטין בשני ההתקנים).

מימוש file operations

⊕ מימוש open. עליכם לבצע את הפעולות הבאות:

1. לבחור את אובייקט ה-file_operations המתאים (Caesar Cipher או XOR Cipher), בהתאם ל-Minor ששמור ב-inode object.
במידה וה-minor אינו תקין, יש להחזיר ENODEV-.

2. להקצות מקום ב-private data של ה-file object, בו נאחסן את ה-key ואת ה-read state (עליכם ליצור מבנה נתונים משלכם שמאחסן את שני השדות, ולגרום ל-private data להצביע על מופע שלו).

⊕ מימוש release. עליכם לבצע את הפעולות הבאות:

1. לשחרר את המקום שהוקצה ב-private data של ה-file object.

⊕ מימוש ioctl. עליכם לבצע את הפעולות הבאות (סוג הפקודה והערכים המתאימים מתקבלים כפרמטרים של ioctl):

1. לבדוק האם סוג הפקודה שנשלחה הינו change key. אם כן, יש לשמור בשדה key של ה-private data את ה-key החדש.

2. לבדוק האם סוג הפקודה שנשלחה הינו change read state. אם כן, יש לשמור בשדה read state של ה-private data את ה-read state החדש.

3. לבדוק האם סוג הפקודה שנשלחה הינו zero. אם כן, יש לאפס את חוצץ הנתונים של ההתקן. שימו לב: אם תהליך כלשהו פתח את אותו התקן מספר רב של פעמים (מספר קריאות ל-open), לאחר שהתבצעה פקודת איפוס, ניסיון קריאה מההתקן ע"י כל אחד מה-file descriptors השונים יחזירו אפסים. במידה וה-minor של ההתקן אינו תקין, יש להחזיר ENODEV-.

⊗ בקובץ encdec.h מוגדרים קבועים (התבוננו בו): עליכם להשתמש בהם כדי להבדיל בין הפקודות השונות והארגומנטים שלהם.

⊕ מימוש גרסה של write עבור Caesar Cipher. עליכם לבצע את הפעולות הבאות:

1. לכתוב את תוכן **החוצץ שהמשתמש העביר**, לתוך **החוצץ של התקן Caesar Cipher** באופן מוצפן כמתואר לעיל. שימו לב, עליכם לכתוב את המידע החל מהמיקום הבא לכתיבה (מתקבל כפרמטר loff_t *f_pos לפונקציה write).

2. אם לא ניתן לבצע כתיבה נוספת להתקן ההצפנה, משום שערכו של *f_pos שווה ל-memory_size של חוצץ ההצפנה של ההתקן, אז יש להחזיר את השגיאה ENOSPC-.

3. כאשר אתם מבצעים את העתקת המידע מהחוצץ של המשתמש (שנמצא ב-user space) אל חוצץ ההתקן (שנמצא ב-kernel space), עליכם להשתמש בפונקציה copy_from_user כאשר מאפשרת להעתיק כך מידע באופן בטוח (מידע נוסף כאן).
   במידה ו-copy_from_user נכשלת, על הפונקציה להחזיר EFAULT-.

4. לאחר סיום הכתיבה, עליכם להוסיף ל-*f_pos את מספר התווים שכתבתם לחוצץ, כך שבפעם הבאה נכתוב במקום הפנוי הבא, ולא נדרוס תווים שכבר נכתבו.

5. **שימו לב**, גם אם אותו התקן נפתח ע"י תהליך מספר רב של פעמים (כמה קריאות ל-open), כל פעולות הכתיבה שיופנו ל-file descriptors שהתקבלו

מקריאות ה-open יכתבו כולן לאותו החוצץ של ההתקן: יש רק שני חוצצים, אחד לכל התקן.

⊕ מימוש גרסה של write עבור XOR Cipher, באופן דומה ל-Caesar Cipher.

⊕ מימוש גרסה של read עבור Caesar Cipher. עליכם לבצע את הפעולות הבאות:

1. לקרוא את תוכן **החוצץ של התקן Caesar Cipher**, לתוך **החוצץ של המשתמש** באופן **מפוענח** (decrypted) או **חשוף** (raw), בהתאם ל-read state שנמצא ב-private data של ה-file object. שימו לב, עליכם לקרוא את המידע החל מהמיקום הבא לקריאה.
2. אם לא ניתן לבצע קריאה נוספת מהתקן ההצפנה, משום שערכו של *f_pos שווה ל-memory_size של חוצץ ההצפנה של ההתקן, יש להחזיר את השגיאה EINVAL-.
3. כאשר מבצעים העתקת מידע מהחוצץ של ההתקן (שנמצא ב-kernel space) אל החוצץ של המשתמש (שנמצא ב-user space), עליכם להשתמש בפונקציה copy_to_user אשר מאפשרת להעתיק מידע מ-kernel space אל user space באופן בטוח (מידע נוסף כאן).
   במידה ו-copy_to_user נכשלת, על הפונקציה להחזיר EFAULT-.
4. לאחר סיום הקריאה, עליכם להוסיף ל-*f_pos את מספר התווים שנקרא לתוך החוצץ.
5. **שימו לב**, גם אם אותו התקן נפתח ע"י תהליך מספר רב של פעמים (כמה קריאות ל-open), כל פעולות הקריאה שיופנו ל-file descriptors שהתקבלו מקריאות ה-open יקראו כולן מאותו החוצץ של ההתקן: יש רק שני חוצצים, אחד לכל התקן.

⊕ מימוש גרסה של read עבור XOR Cipher, באופן דומה ל-Caesar Cipher.

קומפילציה והרצה

סופקו לכם הקבצים הבאים:

1. encdec.c: שלד לכתיבת התרגיל - השתמשו בו.
2. encdec.h: מכיל הגדרות של קבועים שישמשו אתכם במימוש (אין לערוך קובץ זה).
3. Makefile:
   - ⊕ קובץ המגדיר כיצד לקמפל את התרגיל.
   - ⊕ להידור המודול, עליכם להריץ את הפקודה make ב-shell, וכך gcc יהדר את המודול כמוגדר ב-Makefile.
   - ⊕ אין לערוך קובץ זה.
4. load: shell script שטוען את המודול לגרעין (באמצעות insmod).
   - ⊕ בנוסף, הסקריפט יוצר שני קבצי התקנים: אחד בשם "/dev/encedc0" עם minor 0, והשני "/dev/encdec1" עם minor 1.
   - ⊕ **שימו לב**, load מקבלת פרמטר בשם memory_size, שמועבר בהמשך למודול שכתבתם. אם תריצו "./load memory_size=100", load תאתחל את המודול כך ש-memory_size יהיה 100.
   - ⊕ אין לערוך קובץ זה.
5. unload: shell script שמבצע את הפעולה ההופכית של load:
   - ⊕ מסיר את המודול שנטען ע"י load מהגרעין (באמצעות rmmod).
   - ⊕ מוחק את שני קבצי ההתקנים שנוצרו ע"י load.
   - ⊕ אין לערוך קובץ זה.
6. test: תוכנית שבאמצעותה תוכלו לבדוק את תקינות מנהל ההתקן שתכתבו. התוכנית מאפשרת למשתמש להזין את הפקודות הבאות:

(א) **open #device_id #reference_id #flags**. פקודה זו תגרום לתהליך לקרוא ל-open עבור פתיחת התקן:

i. #device_id הוא 0 או 1, ובאמצעותו ייפתח "/dev/encdec0" או "/dev/encdec1" בהתאמה.

ii. #reference_id הינו מספר המשויך ל-fd האמיתי שהתקבל כתוצאה מקריאה ל-open.

iii. #flags הוא אחד מ-"read", "write", ו-"read|write".
למשל: "open 0 2 read": פתח את "/dev/encdec0" לקריאה, שייך את ה-fd למספר 2.

(ב) **write #reference_id "#string"**. פקודה זו תגרום לתהליך לקרוא ל-write עבור כתיבה להתקן:

i. #reference_id הינו ה-fd לכתיבה.

ii. #string היא המחרוזת אותה אנו מעוניינים לכתוב להתקן (באופן מוצפן).
למשל: "**write 1 "Hello World"**": לכתוב "Hello World" אל ה-fd שמזוהה עם המספר 1.

(ג) **read #reference_id #count**. פקודה זו תגרום לתהליך לקרוא ל-read עבור קריאה מהתקן:

i. #reference_id הינו ה-fd לקריאה.

ii. #count הינו מספר הבתים אנו מעוניינים לקרוא.
למשל: "**read 1 5**": לקרוא 5 תווים מה-fd שמזוהה עם המספר 1.

(ד) **lseek #reference_id #pos**. פקודה זו תגרום לתהליך לקרוא ל-lseek כדי להזיז את ה-seek pointer של ה-fd שמזוהה עם ה-reference id הנתון:

i. #reference_id הינו ה-fd הרלוונטי.

ii. #pos הינו המיקום החדש של ה-seek pointer.
למשל: "**lseek 0 0**": הזז את ה-seek pointer של ה-fd שמזוהה עם המספר 0, כך שיצביע לבית הראשון בחוצץ של ההתקן.

(ה) **ioctl #reference_id #cmd #arg**. פקודה זו תגרום לתהליך לקרוא ל-ioctl:

i. #reference_id הינו ה-fd הרלוונטי.

ii. #cmd הינו מזהה הפקודה שברצוננו לבצע: "change_key", "change_read_state" או "zero".

iii. #arg הינו ארגומנט הפקודה שברצוננו לספק, אופציונלי:

- $\oplus$ אם הפקודה היא change_key, יכיל את ערך המפתח החדש.
- $\oplus$ אם הפקודה היא change_read_state, יכיל את הערך decrypt או raw.

למשל: "**ioctl 2 change_key** 5", או "**ioctl 2 change_read_state raw**".

(ו) **close #reference_id**. פקודה זו תגרום לתהליך לקרוא לפונקציה close עבור ה-fd המזוהה עם reference id:

i. #reference_id הינו ה-fd הרלוונטי.

(ז) **exit**. פקודה זו מסיימת את ריצת התוכנית.

7. test.c: קובץ המקור של test. אנא בדקו שאתם מבינים כיצד הפונקציה execute_command (המוגדרת בו) עובדת, להבנת התרגיל באופן טוב יותר (אין לערוך קובץ זה).

8. $\{\text{test} \circ i \circ \text{.in}\}_{i=1}^{5} \cup \{\text{test} \circ i \circ \text{.out}\}_{i=1}^{5}$: קבצי בדיקה.

כאשר אתם עובדים על התרגיל, וודאו כי כל הקבצים נמצאים באותה הספרייה. כאשר תרצו לבדוק את המודול שכתבתם, בצעו את השלבים הבאים:

1. הריצו את התוכנית unload כדי להסיר את גרסת המודול הנוכחית שמותקנת בגרעין.
2. הריצו את התוכנית make כדי להדר מחדש את המודול.
3. הריצו את התוכנית load כדי לטעון מחדש את המודול המהודר.
4. הריצו את התוכנית test כדי לעבוד מול מנהל ההתקן שהותקן ע"י המודול (באמצעות בדיקה ידנית, קבצי הבדיקה הנתונים או קבצי בדיקה שתצרו בעצמכם).

לנוחיותכם, ניתן לראות כאן דוגמא לשימוש באופן העבודה המפורט לעיל.

הגשה

הגשה במודל, לפי הפורמט הבא:

1. עליכם ליצור קובץ zip (השתמשו ב-zip או gzip בלבד) בשם hw2_id1_id2.zip, כאשר id1, id2 הם מספרי תעודות הזהות של המגישים.
2. קובץ ה-zip מכיל **אך ורק** את הקבצים הבאים, **ללא תתי-ספריות**.
   - ⊕ encdec.c.
   - ⊕ dry.pdf, שמכיל את התשובות לחלק התיאורטי.
   - ⊕ submitters.txt, שמכיל את מספרי הזהות והשמות של מגישי התרגיל, מופרדים ע"י פסיק. למשל:

```
Bill Gates,bill@microsoft.com,123456789
Linus Torvalds,linus@gmail.com,234567890
```

3. צרו את קובץ ה-zip באמצעות הפקודה

```
zip hw2_id1_id2.zip encedc.c dry.pdf submitters.txt
```

4. הגישו את ה-zip דרך המודל.