הטכניון – מכון טכנולוגי לישראל

מעבדה במערכות הפעלה 046210

תרגיל בית מס' 3

תאריך הגשה: 8 באפריל 2024 עד השעה 23:55

הקדמה

במסגרת תרגיל זה נבנה מנהל התקן פשוט שתומך בפעולות קריאה, כתיבה ובמנגנון ioctl (ראו הסבר על מנגון זה בהמשך). 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).

תיאור כללי

מטרתכם היא לייצר התקן עבור מערכת צ'אט. קריאה וכתיבת הודעות יתבצעו ע"י קריאה וכתיבה לקובץ בשליטת ההתקן. על ההתקן לתמוך במספר חדרים שפעילים במקביל, כל אחד בקובץ התקן משלו. לכל חדר פעיל, ההתקן ישמור רשימה מקושרת של ההודעות (ראו פירוט מה לשמור לכל הודעה בהמשך). כל תהליך שפותח את הקובץ יוכל לקרוא הודעות בקצב שלו – ההתקן יעקוב אחרי המיקום בתוך רשימת ההודעות של כל תהליך בנפרד. בנוסף כל תהליך יוכל לשאול את ההתקן כמה הודעות נשארו לו לקרוא ולבצע חיפוש להודעה הבאה שנשלחה מהתהליך מסוים.

לאורך כל התרגיל, אפשר להניח כי אין פעולות הנעשות במקביל – אם למשל שני תהליכים קוראים/כותבים לאותו קובץ התקן, בכל רגע נתון רק אחד מהם פועל. לכן אין צורך להשתמש בנעילות או בטכניקות אחרות לתכנות מקבילי.

לאורך התרגיל יש להשתמש בקרנל הרגיל (לא ה-custom).

פירוט על המימוש

בסוגריים מופיע שם הפונקציה המתאימה ב-file_operations:

- פתיחה של קובץ ההתקן (open) - כאן יש לבצע איתחול של החדר. יש ליצור ולאתחל מבנים שתשתמשו בהם ולהקצות זיכרון אם צריך. עבור שגיאה באתחול יש להחזיר EFAULT- (שימו לב למינוס). ההתקן יחזיר 0 בעת אתחול נטול שגיאות.

  שימו לב: פתיחה של אותו קובץ התקן (מזוהה ע"י אותו מספר מג'ורי ואותו מספר מינורי) מספר פעמים תקרה במצב שבו כמה תהליכים נמצאים ביחד באותו החדר, ולכן כולם כותבים וקוראים מאותה רשימת הודעות. לכן, החל מהפתיחה השניה של החדר אין צורך לבצע איתחול ראשוני, כי התהליך רק מצטרף לחדר ולא יוצר חדר חדש.

- סגירה של הקובץ (release) – אחרי שה-file descriptor האחרון של החדר נסגר, יש לשחרר את רשימת ההודעות ומשאבים נוספים שהקצתם למען פעילות תקינה של החדר. ניתן להשתמש בעיקרון reference counting על מנת לעקוב אחרי מספר ה-file descriptors הפתוחים לכל קובץ התקן.
- קריאה (read) – כל קריאה תעתיק לחוצץ שהעביר המשתמש מספר הודעות לפי גודל החוצץ. אתם יכולים להניח כי החוצץ שאתם מקבלים הוא מסוג message_t[] (מערך של message_t, ראו פירוט למטה) ושפרמטר הגודל הוא כפולה שלמה של sizeof(message_t). אם אין מספיק הודעות, יש למלא את החוצץ חלקית. לבסוף יש להחזיר את מספר *הבתים* (לא ההודעות) שנכתבו לחוצץ, שיהיה שווה למספר ההודעות שנכתבו כפול sizeof(message_t). למשל, אם קיבלתם חוצץ בגודל 3*sizeof(message_t) אבל יש רק הודעה אחת בחדר, יש לכתוב רק הודעה אחת לחוצץ ולהחזיר 1*sizeof(message_t). בנוסף יש לעקוב אחרי ההיסט (offset) – אם יש שתי הודעות בחדר ומתבצעות שתי בקשות קריאה מאותו התהליך, כל אחת להודעה אחת, הקריאה הראשונה תחזיר את ההודעה הראשונה והקריאה השניה תחזיר את ההודעה השניה. עושים זאת בעזרת הפרמטר f_pos – בתחילת הקריאה הוא יציין את המיקום הנוכחי (בבתים, ז"א כדאי לחלק ב sizeof(message_t)) של התהליך בתוך רשימת הקריאות, ובסוף בקריאה יש לקדם אותו במספר הבתים שקראנו. אם המצביע לחוצץ הוא NULL יש להחזיר EFAULT-. אם יש בעיה בלהעתיק נתונים למרחב המשתמש יש להחזיר EBADF-. בכל שגיאה אחרת יש להחזיר EFAULT-.
- כתיבה (write) – כל כתיבה תוסיף את ההודעה (מחרוזת ללא סיומת NULL) לסוף רשימת ההודעות. על ההודעות להופיע ברשימה לפי הסדר שבו ההתקן קיבל אותן. יכול להיות שמספר תהליכים יבצעו כתיבה, אך אפשר להניח כי הן לא נעשות במקביל (אין צורך להשתמש בנעילות או בטכניקות אחרות לתכנות מקבילי). על ההתקן לוודא שאורך ההודעה (ללא תו NULL) לא עולה על MAX_MESSAGE_LENGTH. ההתקן יצרף להודעה את ה-PID של התהליך ששלח אותה ואת הזמן שבו ההודעה נשלחה. את הזמן יש לשמור כ time_t (מספר שלם המייצג שניות מאז 1 בינואר 1970) בעזרת פונקציית העזר gettime() בקובץ השלד שקיבלתם. לבסוף יש להחזיר את מספר הבתים שנכתבו להתקן (במקרה שלנו זה יהיה תמיד אורך החוצץ שקיבלתם). במקרה של הודעה ארוכה מדי, יש להחזיר - ENOSPC. במקרה של חוצץ NULL יש להחזיר EFAULT-. במקרה של שגיאה בקריאה מחוצץ המשתמש יש להחזיר EBADF-. בכל שגיאה אחרת יש להחזיר EFAULT-.

- שינוי היסט (llseek) – תהליך יוכל לזוז בתוך רשימת ההודעות קדימה ואחורה. עליכם לתמוך בסוג ההזזה SEEK_SET (ערך 0 לפרמטר whence), שמשנה את המיקום הנוכחי ברשימת ההודעות ביחס להתחלה. המיקום שאתם תקבלו יהיה בבתים (לכן שוב, כדאי לחלק ב(sizeof(message_t כדי להשיג מיקום ברשימה) כמספר אי-שלילי. אם כתוצאה משינוי ההיסט המיקום חורג מעבר לסוף הרשימה, יש לאפס את המיקום לאחרי ההודעה האחרונה (ז"א שקריאה תחזיר 0 עד הכתיבה הבאה). לבסוף יש להחזיר את המיקום החדש (בבתים) של התהליך. אם קיבלתם סוג הזזה לא מוכר, יש להחזיר EINVAL-.
- COUNT_UNREAD – פעולת ioctl. תחזיר את מספר ההודעות שעדיין לא נקראו – מספר ההודעות מהמיקום הנוכחי של התהליך ברשימה עד הסוף. למשל, אם יש 3 הודעות ברשימה והמיקום שלנו הוא 1 (ז"א, לפני ההודעה השניה), COUNT_UNREAD תחזיר 2.
- ~~SEARCH – פעולת ioctl. מקבלת מספר תהליך ומחזירה את המיקום (בבתים) של ההודעה הבאה (לפי המיקום הנוכחי של התהליך) שנשלחה ממספר התהליך הזה. למשל – נניח ויש 3 הודעות שנשלחו מתהליכים A, B, A. אם תהליך נמצא כרגע במיקום 0 ועושה SEARCH לתהליך A הוא יקבל 0. אם הוא נמצא במיקום 1 ועושה את אותו הSEARCH הוא יקבל (2*sizeof(message_t. במקרה שהתהליך לא נמצא יש להחזיר ENOENT.~~

הערות :

- בכל פעולה ניתן להחזיר את השגיאה ENOMEM במקרה וניסיתם להקצות זיכרון דינאמי וההקצאה נכשלה.
- השתמשו בהגדרה הבאה עבור message_t:

```
struct message_t {
   pid_t pid;
   time_t timestamp;
   char[MAX_MESSAGE_LENGTH] message;
}
```

- שימו לב שמכיוון שיש אורך מקסימלי ידוע ל-message, יש להוסיף תו NULL בסוף רק אם אורך ההודעה שאתם כותבים קטן מ-MAX_MESSAGE_LENGTH.
- את פעולת COUNT_UNREAD יש לממש על ידי מנגנון ה-IOCTL לפי הפרוט הבא (ראו מידע על כללי על ioctl למטה):

```
#define MY_MAGIC 'r'
#define COUNT_UNREAD _IO(MY_MAGIC, 0)
```

שם מנהל ההתקן צריך להיות chat. מספר major של ההתקן צריך להינתן באופן דינמי. קבצי התרגיל מכילים שלד (chat.h, chat.c) עליו ניתן לבנות את מנהל ההתקן. בנוסף, בקבצי השלד ישנם פונקציות עזר לשליפה של הזמן הנוכחי ב-time_t מהמערכת ולקבל את מספר התהליך הנוכחי. שימו לב שהקבצים שקיבלתם לא בהכרח מכילים שלד לכל הפונקציות שעליכם לממש. היעזרו בחומרי העזר המצוינים למטה לפי הצורך.

מידע שימושי

- הסבר על מנהלי התקן מסוג Char Driver + דוגמאות ניתן למצוא ניתן למצוא בקישור הבא וכן בקישור הבא.
- אפשר למצוא הסבר על איך להוסיף תמיכה ב- IOCTL בקישור הבא.
- ניתן, ורצוי ,לדבג את המודול בעזרת פקודת printk.
- הגרעין לא מקופל אל מול הספריות הסטנדרטיות (GNU CLib). לכן לא ניתן לעשות שימוש בפונקציות מתוך 'stdlib.h', 'stdio.h', 'string.h' וכדומה. עם זאת, לרוב הפונקציות השימושיות יש תחליף ומימוש מתאים בגרעין.
- הגרעין עושה שימוש בזיכרון פיזי. לשם הקצאת זיכרון דינמי בגרעין יש לעשות שימוש בפקודות kmalloc ו-kfree אשר מקצות זיכרון באופן דינאמי. שימו לב, עליכם לדאוג לשחרר זיכרון שאתם מקצים, דליפות זיכרון יחשבו כשגיאה.
- זיכרון השייך למרחב הגרעין חייב להיות מופרד מטעמי אבטחה מזיכרון המשתמש. לכן, קוד מנהל ההתקן לא יכול לגשת ישירות למידע הנמצא במרחב הזיכרון של המשתמש באופן ישיר!

  בעקבות זאת עליכם לעשות שימוש בשגרות copy_to_user, copy_from_user, strnlen_user על מנת להעביר מידע בין המרחבים. ניתן לקרוא עליהן בקישור הבא ובקישור הבא
- בקובץ ההתקן ישנו שדה בשם private_data בו אתם יכולים להשתמש על מנת לשמור מידע שיישאר לכל אורך חיי ההתקן (כל עוד הוא לא נסגר בעזרת פונקציית ()close או כאשר התהליך שפתח את קובץ ההתקן נסגר בעצמו). לדוגמא, תוכלו לשמור מבנה אשר מנהל את החוצץ בשדה זה.
- המיקום הנוכחי בקובץ פתוח (מסוג struct file) נשמר בשדה f_pos. ניתן לראות את ההגדרה המלאה של המבנה בקובץ include/linux/fs.h.

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 chat.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 ./chat.o
```

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

```
mknod /dev/chat 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/chat
rmmod chat
```

מומלץ לכתוב סקריפטים שיבצעו את הפעולות הנ"ל בצורה אוטומטית על מנת לחסוך זמן וטעויות אנוש.

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 התוכנית בה השתמשתם לבדיקת מנהל ההתקן שלכם.

הוראות הגשה לתרגיל:

- 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 id1, id2 הם מספרי ת.ז. של השותפים. הקובץ יכיל:

- קבצי התוכנית: chat.h, chat.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
         +- chat.c
         +- chat.h
         |
```

דגשים לגבי הציון

- משקל התרגיל הינו 30% מהציון הסופי.
- התרגילים יבדקו אוטומטית. ירדו ניקוד לעבודה שלא עומדת בהוראות ההגשה.
- יש להקפיד על סדר ותיעוד הקוד.

נספח: הסבר על מנגנון ה-*ioctl*

צורת השימוש בהרבה התקנים (דיסקים חיצוניים, כרטיסי קול, מקלדת וגם ההתקן בתרגיל שלנו) היא של העברת מידע מ- ואל ההתקן, בדומה לקובץ. מהסיבה הזאת בחרו להכליל את המנגנון הזה, ולממש העברת מידע בין המשתמש להתקן דרך קובץ שאליו קוראים וכותבים. קריאה וכתיבה לקבצים הוא מנגנון פשוט ויעיל, ומשתמשים כדי לעשות את הפעולות הפשוטות ו\או הפעולות שדורשות להעביר מידע בקצב מהיר. הבעיה היא שמנגנון זה מוגבל רק לפעולות קריאה וכתיבה. במקרה וההתקן רוצה לחשוף כלפי המשתמש עוד פונקציות, הוא יכול לעשות זאת בעזרת מנגנון ioctl – קריאת מערכת כללית שמריצים על קובץ התקן

ומעבירים לה מספר פונקציה ופרמטר. מנגנון זה מאפשר להתקן להגדיר פונקציות נוספות, להצמיד להם מספר ולהודיע למשתמש (למשל בתיעוד של ההתקן) מה המשמעות של כל מספר פונקציה ומה הפרמטר שהוא צריך להעביר. המשתמש לאחר מכן ישתמש בקריאת המערכת ioctl על הקובץ של ההתקן ביחד עם מספר הפונקציה כדי לבצע את הפעולות הנוספות.

דוגמה פשוטה לשימוש ב-ioctl הוא כרטיס קול. ההתקן של כרטיס הקול יצור קובץ התקן, אשר קריאה ממנו תבצע הקלטה (דרך שקע המיקרופון של הכרטיס) וכתיבה אליו תשמיע צליל (דרך שקע הרמקולים). אבל אם המשתמש רוצה לשנות הגדרות של הכרטיס, כמו למשל קצב הדגימה של המיקרופון, הוא יפתח את קובץ ההתקן ויבצע ioctl. במספר הפונקציה הוא יכתוב את מה שכתוב בתיעוד של מנהל ההתקן, ובפרמטר הוא יכול להעביר את קצב הדגימה כמספר חיובי. עבור פונקציה אחרת (של אותו ההתקן או התקן אחר), הפרמטר יכול להיות מספר שלילי או אפילו מצביע. מבחינת ioctl עצמה אין משמעות לפרמטר – המימוש במנהל ההתקן יתייחס לפרמטר לפי הטיפוס המתאים למספר הפונקציה שהמשתמש ביקש להריץ.

שימו לב שמספר הפונקציה הוא שרירותי לחלוטין ופרטי למנהל ההתקן שאתם כותבים. למרות זאת, הוגדרו קונבנציות איך ליצור את המספרים האלה ויש פונקציות עזר לשם כך. זאת המשמעות של שורות ה-define בקובץ השלד – להגדיר את המספרים של פונקציות ה-ioctl בתרגיל לפי הקונבנציות הנהוגות בקרנל.