תרגיל בית רטוב 4

תאריך הגשה: יום שישי 19/06/2015, 12:30 בצהריים

המתרגל האחראי על התרגיל: עידו זמירי

שאלות ותשובות לתרגיל בית זה יינתנו כרגיל בפיאצה.
אנו ממליצים להתחיל לעבוד על תרגיל הבית מוקדם ככל האפשר.
המסמך מנוסח בלשון זכר (אך מיועד לשני המינים).

הקדמה

בתרגול למדתם את העקרונות הבסיסיים של כתיבת מנהל התקן. בתרגיל זה תשתמשו בהם לצורך כתיבת מנהל התקן מסוג char device.
מטרתכם בתרגיל זה היא לממש מנהל התקן המדמה משחק מחשב דמוי snake עבור שני שחקנים כפי שיתואר באופן כללי בהמשך. ההתקן מנהל את המשחק מול התהליכים שפונים אליו (הם השחקנים במשחק). תהליך יכול לבקש לבצע מהלכים או לקרוא מהו מצב הלוח הנוכחי.
מומלץ לעיין בקובץ מהאתר (תרגיל במבוא למדעי המחשב) לשם הבנה של כללי המשחק. לשם המחשת המשחק, הינכם מוזמנים להדר את התוכנה המצורפת (מימוש של המשחק בuserspace) ולשחק בה.
אינכם צריכים להבין את המימוש המצורף לעומקו, אלא לתקן אותו כך שיתאים להתקן. למשל, סביר שבכל מקום שבו יש קריאה לprintf תרצו לשנות אותה להחזרת ערך שגיאה למשתמש (פרט לפונקציה Print במימוש הנתון), ובאופן דומה, את הקריאות לscanf סביר שתרצו לשנות לקבלת ערך מwrite (ראו הסבר עבור write בהמשך). הפונקציות העיקריות אותם אתם אמורים להבין הן Update, GetInputLoc (ובעיקר בהקשר של פעולת הscanf) Print1 על מנת לקבל את מצב הלוח (ובעיקר בהקשר של פעולת הprintf).
במשחק משתתפים שני שחקנים, האחד שחור והשני לבן (במימוש מיוצג על ידי הטיפוס player המקבל אחד משני הערכים המוגדרים בdefine. שימו לב שאין לשנות ערכים אלה כדי שהמימוש לא ישבר). השחקן הלבן מבצע את המהלך הראשון על פי הגדרת התרגיל.
בכל תור מתקבל תו יחיד (במימוש בפונקציה GetInputLoc) המציין את כיוון התנועה של הנחש (ערכי התוים ומשמעותם מוגדרים באמצעות define במימוש).
הדפסת הלוח מתבצעת בפונקציה Print: נחש מחולק לכמה segmentים, כאשר כל segment מודפס כמספר. ראש הנחש מסומן במספר 1 עבור השחקן הלבן, ו1- עבור השחקן השחור. המשכי הנחשים (הsegmentים הבאים של גופי הנחשים) מסומנים על ידי הגדלת הערך המוחלט ב1 לעומת הsegment הקודם ללא שינוי הסימן (+\-).
שימו לב, המימוש נועד לעזור לכם. אתם יכולים לשנות אותו כרצונכם, ואפילו לכתוב מימוש שונה לחלוטין (לא מומלץ).
בעזרת כתיבה להתקן ניתן לבצע מהלך במשחק. כדי לבצע מהלך יבקש התהליך לבצע את המהלך על ידי שליחת התו המתאים באמצעות הפונקציה write (ניתן לקבץ כמה בקשות על ידי שליחת יותר מתו אחד). בקשה זו תחסום את הקריאה עד שלתהליך (לשחקן) יהיה מותר לבצע את המהלך על פי כללי המשחק (כלומר, זהו תורו של השחקן שרוצה לבצע את המהלך).

מנהל ההתקן – תיאור מפורט

ההתקן ומנהל ההתקן

שמו של מנהל ההתקן הוא snake, והוא ימומש כמודול. המספר הראשי של מנהל ההתקן יוקצה באופן דינמי. ניתן להוריד מאתר הבית (יחד עם הגדרות תרגיל זה) את הקובץ snake.h. עליכם לממש את מנהל ההתקן בקובץ בשם snake.c, על סמך ההגדרות המופיעות ב-header file.

ניתן לייצר מספר משחקים שינוהלו על-ידי מנהל ההתקן. מספר המשחקים שעל מנהל ההתקן לתמוך בהם יועבר לו כפרמטר בעת יצירתו (כלומר בביצוע insmod). שם הפרמטר הינו max_games וטיפוסו int. ניסיון לייצר יותר ממספר ההתקנים המותר לא יבדק. הminor של ההתקנים יהיה סדרתי החל מ0. כל התקן כזה יעבוד בצורה הבאה:

- כל התקן יחזיק באופן בלתי תלוי בהתקנים האחרים משתנים עבור המשחק עליו הוא אחראי.
- בעת פתיחת התקן, יוגרל לוח משחק ושני תהליכים יוכלו לפתוח במשחק מול ההתקן (שימו לב שעליכם לשנות את המימוש של הגרלת המספר, מכיוון ש()rand אינו נתמך בkernel).

פונקציות נתמכות

להלן תיאור הפונקציות שבהן מנהל ההתקן שלכם צריך לתמוך. כל הפעולות הבאות ישפיעו אך ורק על ההתקן עליו בוצעה הפעולה. ייתכן שרשימת ערכי החזרה אינה מכסה את כל המקרים והשגיאות האפשריים. במידה ואיתרתם מקרים נוספים, בחרו ערך חזרה מתאים (יש לתעד את זה בתיאור שאתם מגישים). שימו לב שערכי השגיאה המפורטים הם אלה שתוכניות משתמש רואות במשתנה errno (ראו תרגול 2).
שימו לב שבתרגיל זה עליכם לממש פונקציות read/write עם התנהגות שונה מהסטנדרטית, חשוב לזכור שזה למטרת התרגיל בלבד ובאופן כללי התקן תווים לא ממומש כך, אלא יש שימוש במנגנונים שיועדו לכך (ioctl).

1. open – פעולה זו היא רישום למשחק. אם לא קיים שחקן שנרשם אז הקריאה תחסום עד שיגיע שחקן נוסף. אם קיים כבר שחקן אחד שנרשם עבור משחק (התקן) זה המשחק יתחיל (והשחקן הראשון יצא מהחסימה). כל תהליך נוסף שיבצע open להתקן ייכשל. הראשון (אנו נדאג למנוע race. אם יתקיים, אינכם צריכים להבטיח הוגנות) שמבצע את בקשת הopen הוא השחקן הלבן, והשני הוא השחקן השחור.
2. release – כל תהליך שמבצע release עוזב את המשחק. עזיבת המשחק על ידי כל משתתף משמעה שכל קריאה לפונקציה (פרט לrelease) על ידי השחקן (התהליך) היריב תכשל ותחזיר ערך שגיאה מתאים.
3. read – ההתקן יחזיר לתהליך הקורא את מצב הלוח הנוכחי (ממומש בפונקציה Print). בקשה זו היא אסינכרונית למשחק עצמו, כלומר, אם ירצה התהליך, הוא יוכל להכיל שני threadים, ומאחד מהם לקרוא לread על מנת להציג את הלוח, ומהשני לבצע פעולות write.
אינכם צריכים לממש קריאה חלקית של הלוח – כלומר, בכל קריאה אתם רשאים להניח שגודל החוצץ המסופק הוא גדול או שווה למספר התווים הנדרשים להדפסת הלוח. יש לרפד את שארית החוצץ ב'\0'. אם תקבלו ערך קטן ממספר התווים הנדרשים להדפסת הלוח, יש להדפיס את הלוח חלקית. בקריאה הבאה לread, כתבו את מצב הלוח מתחילתו.
4. write – משתמש יקרא לפונקציה זו כדי לבצע מהלכים במשחק. כל מהלך ניתן כתו אחד, כאשר התווים החוקיים הינם 2, 4, 6 ו8. אם מתקבל תו שאינו חוקי, ההתקן יחזיר שגיאה ויגרום להפסד אוטומטי של התהליך שניסה לכתוב את התו הלא-חוקי (שימו לב שדבר זה מהווה שינוי מהמימוש בפונקציה GetInputLoc).
5. llseek – פעולה זו לא תמומש ע"י מנהל ההתקן. אולם, מאחר ומערכת ההפעלה מספקת מימוש ברירת מחדל לפעולה זו, יש למנוע זאת ע"י כתיבת פונקציה

המחזירה תמיד ENOSYS-.

6. ioctl – מנהל ההתקן יתמוך בפעולות הבקרה הבאות, המוגדרות ב-snake.h (הפעולות, כמו פעולת read, הן אסינכרויות למשחק עצמו):
   א. SNAKE_GET_WINNER – פעולת בקרה זו תחזיר 4 אם השחקן הלבן ניצח במשחק, 2 אם השחקן השחור ניצח במשחק או 1- עבור משחק שעוד לא נגמר. ערך שלילי אחר יוחזר עבור שגיאה.
   ב. SNAKE_GET_COLOR – פעולת בקרה זו תחזיר 4 אם השחקן הוא השחקן הלבן, 2 אם השחקן הוא השחקן השחור או ערך שלילי עבור שגיאה.

<u>סינכרוניזציה:</u>

יש להתייחס לבעיות סינכרוניזציה העלולות להיגרם עקב עבודה עם התקן מסוים במקביל ממספר תהליכים (יכולים להיות ניסיונות פתיחה מהרבה תהליכים, ובנוסף יכולים להיות שני תהליכים שחקנים), לצורך כך השתמשו בסמפור המוגדר בגרעין include/asm-i386/semaphore.h. כמו כן עליכם לתמוך באפשרות שכמה תהליכים יפתחו למשחק את אותו ההתקן, במקרה זה המשחק יהיה משותף לכל התהליכים שפתחו את ההתקן למשחק. בנוסף, יש לאפשר עבודה עם יותר מהתקן אחד (/dev/snake0, /dev/ snake1, …). לדוגמה התכנית הסדרתית הבאה חוקית ועליכם לדעת להתמודד איתה:

1. פתח התקן 1.
2. פתח התקן 2.
3. קרא מהתקן 2.
4. כתוב להתקן 1.
5. סגור התקן 2.
6. סגור התקן 1.

הערות והמלצות

- כל הקוד צריך להיות בשפת C.
- הפתרון שאתם מגישים ייבדק על VMware המריץ Red Had Linux 8.0.
- חלק מהבדיקה ייעשה באמצעות בדיקות אוטומטיות.
- מומלץ לבנות את מנהל ההתקן בשלבים:

1. צרו תחילה מנהל התקן בסיסי שניתן לטעון ולהסיר (init_module ו-cleanup_module), המסוגל לתמוך במספר משתנה של התקנים.
2. הוסיפו תמיכה בפעולות open ו-release העובדות כנדרש.
3. לבסוף הוסיפו את הטיפול בפעולות הקריאה והכתיבה.

שימו לב שבכל שלב ייתכן שתידרשו לעדכן פונקציות מהשלבים הקודמים.

הנחיות הגשה

ההגשה של תרגיל זה מכילה שני חלקים:

- הגשה אלקטרונית – עליכם ליצור קובץ zip עם התכולה הבאה (ללא תתי ספריות!):
  - כל קבצי המקור המהווים מימוש של מנהל ההתקן שלכם, <u>ללא</u> הקבצים מהאתר.
  - קובץ makefile בשם Makefile שמהדר את הקבצים ויוצר מודול בשם snake.o.
  - קובץ בדיקה בסיסי בשם test_snake.c שמשתמש בפונקציות של ההתקן.
  - קובץ בשם submitters.txt שמכיל את מספרי הזהות והשמות של כל מגישי התרגיל במבנה הבא:

Bill Gates bill@t2.technion.ac.il 123456789

Linus Torvalds linus@gmail.com 234567890

Steve Jobs jobs@os_is_best.com 345678901

- **הערה חשובה:** קובץ הzip צריך להיות בצורה המתוארת בדיוק. קובץ הzip צריך להכיל אך ורק קבצים ולא תיקיות. אתם יכולים ליצור קובץ zip (בvmware) ע"י הרצת הפקודה הבאה:

zip final.zip snake.c <any additional files> submitters.txt

קובץ הzip צריך להיראות בצורה הבאה:

```
zipfile -+
         |
         +- Makefile
         |
         +- test_snake.c
         |

         +- snake.c
         |
         +- …
         |
         +- submitters.txt
```

1. הגשה **מודפסת** הכוללת הסברים על מבנה המודול שלכם, איך בחרתם לממש אותו וכל פרט אחר שתמצאו כרלוונטי.

עלו והצליחו, צוות הקורס