להחזיר לתא מס' 87

מערכות הפעלה
234123

תרגיל בית יבש מספר : 4

הוגש ע"י :

אלכסנדר בוגצ'נקו	312867187	shadow89@t2
שם	מספר זהות	דואר אלקטרוני
סרגיי בוכמנס	321188898	serbuh@t2
שם	מספר זהות	דואר אלקטרוני
דני אנשלס	311727200	sdanshel @t2
שם	מספר זהות	דואר אלקטרוני

ציון

שאלה 1 - פסיקות, סיגנלים ותקשורת בין תהליכים

א. 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"
מי משניהם צודק? הסבר/הסברי.

סטודנט ב' צודק - במקרה ויש לנו מבנה נתונים מורכב אנחנו יכולים לקבל סיגנל לביצוע פעולות מסויימות במבנה הנתונים בעת הטיפול במבנה הנתונים. הטיפול עדיין לא הסתיים ואילו הסיגנל מבצע פעולות/ שינויים וכתוצאה מכך אנו עלולים לקבל תוצאות שגויות ולכן יש צורך בסינכרון.

ב. אם ביט ה - if ב - eflags כבוי, השפעתה של פקודת המכונה sti מתחילה רק לאחר סיום פקודת המכונה שבאה אחריה.
האם יש הבדל בין קטעי הקוד הבאים? אם כן מהו, אם לא הסבר/הסברי מדוע.

i.

```
1.          2.
sti         sti
nop         cli
cli
```

במקרה זה יש הבדל בין קטעי הקוד - במקרה 1 בדגל יהיה מכובה לאחר ביצוע פקודות המכונה מכיוון ש- cti ידליק את הדגל לאחר פקודת nop ואילו cli יכבה אותו לאחר מכן.
בקרה ה-2 לעומת זאת הדגל יהיה דלוק מכיוון שלאחר ביצוע cli, sti ידליק את הדגל ולכן בסיום הפעולות הדגל יהיה דלוק.

ii.

```
1.          2.
sti         Iret
iret
```

במקרה זה אין הבדל בין קטעי הקוד – שני קטעי הקוד יחזרו מהשגרה וישחזרו את הרגיסטרים, בפרט את eflag . לאחר ביצוע הפעולה הנ"ל (חזרה מהשגרה) אנחנו נמצאים בסביבת המשתמש ולכן אין משמעות לפקודת ה-sti.

ג. לאיזה פקודת shell דומה התוכנית הבאה? הסבירו.

```
close(0);
open("./my_file", O_RDONLY);
execv("/bin/cat");
```

הפקודה הנ"ל דומה לפקודה – cat < ./my_file כאשר הפקודות:
Close(0) - סוגרת stdin, קלט סטנדרטי
open("./my_file", O_RDONLY) – פותחת את הקובץ לקריאה בלבד
execv("/bin/cat") – מציגה את תוכן של הקובץ על המסך

שאלה 2 - זיכרון וירטואלי

השאלה מסתמכת על השינוי הבא במערכת הפעלה Linux: נניח כי כעת מערכת ההפעלה משתמשת בשני זיכרונות פיזיים ראשיים. כל נתון בזיכרון הוירטואלי יכול להופיע באחד הזיכרונות או בשניהם. אם הוא מופיע בשניהם אז הוא יופיע באותה כתובת פיזית. כעת במקום ביט present אחד יהיה שימוש בשני ביטים של present. אם שניהם כבויים אז הדף לא נמצא בזיכרון. אם הביט הראשון דלוק, אז הדף נמצא בזיכרון הראשון. אם השני דלוק, אז הדף נמצא בזיכרון השני. אם שניהם דלוקים אז הדף נמצא בשני הזיכרונות. אנו עובדים עם מערכת של 32 ביט וגודל דף של 4K כרגיל.

א. מה הוא גודל הזיכרון הוירטואלי המקסימאלי שאליו יכול לגשת תהליך במבנה החדש (תחת ההנחה שתהליך יכול להשתמש בכל מרחב הכתובות)? הסבירו.

4GB – המערכת שאנו עובדים איתה היא 32 ביט וגודל דף של 4K ולכן גודל הזכירון הוירטואלי הוא 4GB .

ב. האם צריך לבצע שינוי במנגנון התרגום של כתובת לפי טבלאות הדפים עבור השינוי? אם כן תארו מה השינוי? אחרת, הסבירו מדוע לא.

כן – עלינו לדעת עם איזה זיכרון פיזי אנחנו עובדים לכן עלינו לשמור מידע עבור ה-present של הזיכרון הראשי הנוסף (ברגיסטר cr3 או במשתנה גלובלי מסוים) כך שהרגיסטר cr3 עם המידע הנוסף ידעו להפנות את ה- PGD לזיכרון המתאים.

ג. כיצד יתבצע מנגנון התרגום של כתובת וירטואלית עבור המבנה החדש של הזיכרון כאשר ידוע שכל הדפים שדרושים בדרך נמצאים בזיכרון?

ה-PGD יעביר לטבלת הדפים אשר מכילה את 2 דגלי ה-present. לפי דגלים אלו נדע היכן ובאיזה זיכרון נמצא הדף . התרגום הנותר נשאר זהה.

ד. האם צריך לערוך שינוי במנגנון COW? אם כן, הסבירו מה השינוי, ואם לא, הסבירו מדוע.

אין צורך לערוך שינוי במנגנון ה-COW מכיוון שאין שינוי במנגנון הרשאות ש-COW משתמשת בו.

ה. האם צריך לערוך שינויים בTLB עבור המבנה החדש? אם כן, הסבר מה השינוי ואם לא הסבירו מדוע.

כן, צריך להוסיף מידע עבור ה-present הנוסף (ניתן לשמור אותו בביט או במקום אחר) כך שנדע לאיזה זיכרון שייכת הכתובת המתורגמת.

ו. האם נדרש שינוי במבנה של טבלת המסגרות עבור המבנה החדש? אם כן, הסבירו מה השינוי ואם לא הסבירו מדוע.

כן, נחזיק טבלת מסגרות נוספת לזיכרון הראשי הנוסף.

שאלה 3 - מודולים

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

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

לא נכון. הפקודה kill הורגת תהליך שיש לו pid אך מודול אינו תהליך ואין לו pid לכן פקודה זו לא תעבוד במקרה זה.

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

לא נכון – release יתבצע רק כאשר יתבצע close ע"י תהליך האחרון.

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

לא נכון – מודול עובד בגרעין ולכן אינו יכול להפעיל system call על מנת להשתמש בפקודת getpid.

ד. 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.

לא נכון - register_chrdev מקשר בין ה-major לבין מנהל ההתקן ואין שימוש ב-minor ב-init_module. ה-minor נקבע לאחר יצירת מנהל התקן בעת הוספת התקן ומשתמשים במספר זה על מנת להבדיל בין התקנים השונים החוברים למנהל ההתקן.

Explanation:

We implemented two circle buffers for each device: encoder and decoder. We solved the synchronization problem with help of semaphores and its wait queue. For each device we have two counters for number of readers and number of writers. And also two binary semaphores: the writers semaphore initialized as unlocked and the readers semaphore initialized as locked (because in this case the reader should wait for the first writer that should unlock the semaphore for him). In each write() the writers semaphore we lock the writers semaphore at the beginning, then we do all writing operations and then the semaphore is raised. At the end we do the edge case checks. If the free size in buffer is 0 now then we lower the semaphore in order to put into the wait queue the next writer that do not have any space to write. When this writer will arrive, he will check is there any readers are active and then he will enter the wait queue. Otherwise he will return with EOF. Also writer should raise the semaphore for readers in case that there was no any data on start in the buffer and there were some data written in current call of the write(). In read function the use of semaphores is almost the same. The difference is that reader lower its semaphore in case of no data in the buffer and it raises the writers semaphore in case of no free space on start and some data was read from the buffer.