ה א ו נ י ב ר ס י ט ה ה פ ת ו ח ה

20594

מערכות הפעלה

חוברת הקורס סתיו 2024א

כתב: אריה לויטן

אוקטובר 2023 – סמסטר סתיו – תשפ"ד

פנימי – לא להפצה.

תוכן העניינים

אל הסטודנט,

אנו מקדמים את פניך בברכה עם הצטרפותך אל הלומדים בקורס " מערכות הפעלה".

בחוברת זו תמצא את לוח הזמנים, תנאים לקבלת נקודות זכות ומטלות.

לקורס קיים אתר באינטרנט בו תמצאו חומרי למידה נוספים, אותם מפרסם/מת מרכז/ת ההוראה. בנוסף, האתר מהווה עבורכם ערוץ תקשורת עם צוות ההוראה ועם סטודנטים אחרים בקורס. פרטים על למידה מתוקשבת ואתר הקורס, תמצאו באתר שה"ם בכתובת:
http://telem.openu.ac.il
מידע על שירותי ספרייה ומקורות מידע שהאוניברסיטה מעמידה לרשותכם, תמצאו באתר הספריה באינטרנט www.openu.ac.il/Library.

אפשר לפנות אלי בדואר אלקטרוני aryehle@openu.ac.il או בשעות הנחיה הטלפונית המפורסמות באתר הקורס. הפרטים הללו מצויים גם באתר המחלקה למדעי המחשב telem.openu.ac.il/cs. פגישות יש לתאם מראש.

חשוב להדגיש כי התקשוב בקורס ישמש ערוץ רשמי בין צוות ההוראה של הקורס לבין הסטודנט, כלומר חובה על כל סטודנט להתעדכן באופן שוטף על הנעשה בקורס דרך אתר הבית. כל ההודעות - הן בנושאים אקדמיים והן בנושאים מנהליים - יועברו דרך אתר הבית בלבד, ולא יישלחו הודעות בדואר רגיל. סטודנטים אשר אין להם גישה לרשת האינטרנט יוכלו לגשת למרכז הלימוד הקרוב לביתם ולהשתמש במעבדת המחשבים שם. לפרטים מלאים על מרכזי הלימוד ושעות הפתיחה, ניתן להתקשר למוקד הפניות בטלפון: 09-7782222.

לתשומת לב הסטודנטים הלומדים בחו"ל:
למרות הריחוק הפיסי הגדול, נשתדל לשמור אתכם על קשרים הדוקים ולעמוד לרשותכם ככל האפשר.
הפרטים החיוניים על הקורס נכללים בחוברת הקורס וכן באתר הקורס.
מומלץ מאד להשתמש באתר הקורס ובכל אמצעי העזר שבו וכמובן לפנות אלינו במידת הצורך.

- **שאילתא** - לפניות בנושאים אקדמיים שונים כגון מועדי בחינה מעבר לטווח זכאות ועוד, אנא עשו שימוש מסודר במערכת הפניות דרך שאילתא. לחצו על הכפתור פניה חדשה ואחר כך לימודים אקדמיים > משימות אקדמיות, ובשדה פניות סטודנטים: השלמת בחינות בקורס. המערכת תומכת גם בבקשות מנהלה שונות ומגוונות.

בברכת לימוד פורה ומהנה,

אריה לויטן
מרכז ההוראה בקורס

1. לוח זמנים ופעילויות (מס' קורס /א2024)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	08.12.2023-03.12.2023 (ו חנוכה)			
2	15.12.2023-10.12.2023 (א-ו חנוכה)			
3	22.12.2023-17.12.2023			
4	29.12.2023-24.12.2023			הגשת ממ"ן 11: 30.12.2023
5	05.01.2024-31.12.2023			
6	12.01.2024-07.01.2024			
7	19.01.2024-14.01.2024			
8	26.01.2024-21.01.2024			הגשת ממ"ן 12: 27.01.2024
9	02.02.2024-28.01.2024			
10	11.02.2024-04.02.2024			הגשת ממ"ן 13: 12.02.2024 אינו חובה

** התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".*

מועדי בחינות הגמר יפורסמו בנפרד

2. תיאור המטלות

קרא היטב עמודים אלו לפי שתתחיל לענות על השאלות

חוברת זו מכילה מידע על המטלות ואת המטלות עצמן.
פתרון המטלות הוא חלק בלתי נפרד מלימוד הקורס - הבנה מעמיקה של חומר הלימוד דורשת תרגול רב. המטלות יבדקו על-ידי המנחה ו/או בודק מיוחד ויוחזרו לך בצירוף הערות המתייחסות לתשובות.

לכל מטלה נקבע משקל. חובה להגיש את מטלות 11 ו 12, מטלה 13 היא רשות(בונוס).
.

ללא קבלת ציון 60 לפחות בכל אחת ממטלות חובה(11, 12) לא ניתן יהיה לגשת לבחינת הגמר

לתשומת לבכם !
ציון סופי מחושב רק לסטודנטים שעברו את בחינת הגמר בציון 60 ומעלה והגישו את כל המטלות החובה בציון 60 לפחות.

כל סטודנט יכין את הממ"נים לבדו. אין להגיש את הממ"נים בזוגות (או קבוצות) !

3. התנאים לקבלת נקודות זכות

א. הגשת מטלות חובה(11, 12) עם ציון מינימלי של 60 נקודות בכל אחת מהמטלות.
ב. ציון של לפחות **60** נקודות בבחינת הגמר.

4. הדרכה לפתרון תרגילי התכנות

תרגילי התכנות בקורס זה דורשים מאמץ ניכר. התרגילים לכשעצמם אינם קשים באופן מיוחד אולם הם דורשים הכרה והבנה טובה של החומר המוצע כחומר רקע (ראו סעיף "חומר רקע" בגוף כל ממ"ן)

למרות שהקוד הנדרש בסופו של דבר בתרגילי התכנות איננו ארוך, סביר להניח כי תקדישו לתרגילים שעות רבות. תכנות מערכת הפעלה, דורש ניסיון, ולמרבה העצב רכישת הניסיון כרוכה לרוב גם בהקדשת זמן. עם זאת, התרגילים תוכננו כך שיעסקו מעט ככל האפשר בנושאים שמטבעם הם טכניים בלבד.

בפתרון התרגילים אנו מציעים את השלבים הבאים :

א. קראו היטב את דרישות התרגיל והבהירו לעצמכם מה הבעיות שעלולות להתעורר בעת יישומו.

ב. קראו את החומר המוצע כחומר רקע (ראו סעיף "חומר קרע" בגוף כל ממ"ן). לצורך זה מצויים בידכם ארבעה מקורות, עיינו בהם על פי הסדר הבא :

1. ספר הקורס , Modern Operating Systems, המספק את הרקע התיאורטי.
2. המדריך למתכנת המערכת, The GNU C library reference manual, מתאר את פעולת קריאות המערכת ברוב מערכות UNIX הקיימות
3. הפקודה "man *command-name*" ב-UNIX מאפשרת לקבל מידע על פקודות, פונקציות ספריה, וקריאות מערכת, כפי שהן ממומשות במערכת שבידך.
4. מידע נוסף שמכיל דוגמאות קוד והסברים אפשר למצוא באינטרנט, בפרט באתרים שכתובותיהם מצוים בקטגוריה "אתרים ברשת" (ראו את הדף הראשי של אתר הקורס).

ג. בעת כתיבת הקוד, הקפידו על הכללים המקובלים, בהנדסת תוכנה. רוב הדרישות המפורטות כאן מוכרות לכם בודאי מקורסים קודמים אומנם ישנן דרישות ייחודיות לקורס במערכות הפעלה. לקיום הדרישות הללו קיימת השפעה על ציון הממ"ן :

1. מתן שמות משמעותיים למשתנים.
2. הימנעות משימוש במספרים שרירותיים.
3. כתיבת פונקציות קצרות.

4. תיעוד סביר. הכוונה לתיעוד מתומצת של פעולות התוכנית, של פונקציות ושל משתנים. כמו כן, יש לרשום בתחילת כל קובץ קוד שמוגש את הפרטים האישיים (שם מלא ומספר סטודנט) ותיאור קצר של תוכן הקובץ.
5. יש להקפיד על שימוש בשמות המוגדרים במטלה.
6. אין להשתמש ב goto. ליציאה מלולאות ניתן להשתמש במידת הצורך ב continue או break.
7. מבנה מדורג. מודולים ופונקציות קצרות וללא אפקטים משניים.
8. Indentation.
9. משפטי תנאי קצרים.
10. כל יציאה בגלל שגיאה חייבת להיות מתועדת. למשל, באמצעות הפונקציה perror().
11. בכל מקרה יש לבדוק את הערך המוחזר על ידי קריאות מערכת.
12. בכל מקרה יש לבדוק את נכונות הקלט.
13. התוכנית לא תיפול עקב שגיאה/תקלה כלשהי. במידה וקורה אירוע בלתי צפוי, על התוכנית להודיע על כך ולסיים את עבודתה.
14. אין להשתמש בפונקציה system().
15. יש לשחרר את כל המשאבים שאינם בשימוש.
16. הוראות קומפילציה יש לכתוב בשפת ההוראות של תוכנית השירות make ואם נדרש להגישם בקובץ בשם makefile.
17. חובה להשתמש בדגל (flag) "Wall-" בזמן קומפילצית התוכניות.

בונוס

במקרים יוצאי דופן, כאשר מוגשת תוכנית טובה במיוחד או כזו שעושה למעלה ממה שנדרש, תישקל האפשרות להוסיף עד 5 נקודות בונוס. בכל מקרה שהנכם מתכוונים להגיש תוכנית מעין זו, שימו לב כי :

1. כל הדרישות מהתוכנית המקורית יתקיימו.
2. כל תוספת תהיה מתועדת היטב.
3. תוספות המכילות שגיאות עלולות להוריד מהניקוד הסופי גם אם התוספות לא נדרשו במטלה. כוונות טובות אינן מובילות בהכרח לתוצאה הרצויה.

מטלת מנחה (ממ״ן) 11

הקורס : ״מערכות הפעלה״

חומר הלימוד למטלה : ראו פירוט בסעיף ״רקע״

מספר השאלות : 5 | משקל המטלה : 10

סמסטר : 2024א | מועד אחרון להגשה : 30.12.2023

הגשת המטלה : שליחה באמצעות מערכת המטלות המקוונת באתר הבית של הקורס. הסבר מפורט ב״נוהל הגשת מטלות מנחה״.

החלק המעשי (70%)

כללי

בתרגיל זה נכיר את מבנה של מערכת הפעלה בכלל ומערכת הפעלה XV6 בפרט . מערכת הפעלה XV6 היא מערכת ממשפחת **LINUX** שפותחה לצורכי לימוד ע״י **MIT** . היא הרבה יותר פשוטה והרבה פחות נוחה(תרגישו את זה מיד בשימוש בה אפילו ב **CLI** שלא מאפשר שימוש בחצים למשל), אבל מצד שני מאפשרת להבין את קוד מערכת הפעלה ולשנות אותו בקלות יחסית. היא לא מושלמת ויש בה באגים!

מטרות:

- הכרת מערכת הפעלה xv6
- הכרת ההיבטים המעשיים של מימוש קריאות מערכת
- הכרת מבני נתונים שונים של מערכת הפעלה
- הוספת קריאת מערכת חדשה
- הוספת פקודת מערכת חדשה ps שמדפיסה את מצב תהליכים במערכת
- **התנסות בבניה והרצה של מערכת הפעלה בצורה הקרובה למציאות(כשלא כל המידע זמין וצריך להבין ולמצוא אותו לבד) !**

רקע

א) פרק Makefile מחוברת ״Ubuntu 16.04 programming environment, making first steps״ (הורידו את החוברת מאתר הקורס).

ב) ״Running and debugging xv6.pdf״(באנגלית, כולל הוראות דיבוג משורת הפקודה) ו/או ״XV6 Instalation and EclipseConfig.pdf״ (בעברית, מאחד התלמידים, כולל דיבוג ב ECLIPSE) מתוך maman11.zip. התקינו את המכונה הוירטואלית מאתר הקורס ואת QEMU בתוך המכונה הוירטואלית שלכם לפי ההוראות בקובץ הנ״ל,סיסמת המנהל password .

ג) פרק 0 (עד PIPES בע' 13), פרק 1 ופרק 3(עד X86 protection בע' 40)

מתוך https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf .

אין צורך להתייחס לענייני ניהול זיכרון ראשי.

ד) expect - שפת סקריפט אינטראקטיבית : . https://likegeeks.com/expect-command

תתקינו את expect במכונה הוירטואלית שלכם(סיסמת המנהל password) :

```
sudo apt-get update
sudo apt-get install expect
```

ה) במידת הצורך סרטונים על שימוש ודיבוג ב XV6 מאתר הקורס(בחלק ממ"נים). מספרי הממ"נים והדוגמאות בהם לא זהים לתוכן המטלות.

תיאור המשימה

בקובץ maman11.zip תמצאו ספרייה עם מערכת ההפעלה xv6 שאין בה פקודה ps ואין קריאת מערכת הדרושה לביצועה.

הסבר מפורט

1. הפעילו את מערכת ההפעלה xv6 כמתואר בסעיף ב' של "חומר קרע". הריצו את תוכנת ps , תקבלו את התמונה הבאה:

```
cpu1: starting 1
cpu0: starting 0
sb: size 2000 nblocks 1954 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap s
tart 45
init: starting sh
$ ps
exec ps failed
$
```

הסיבה לשגיאה היא שפקודה ps כלל לא קיימת במערכת.

תוצאת ההרצה צריכה להיות:

```
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap st
art 58
init: starting sh
$ ps
name     pid     state          ppid
init     1       SLEEPING       0
 sh      2       SLEEPING       1
 ps      3       RUNNING        2
 $
```

אפשר להריץ את פתרון ביה"ס לפי ההוראות שבתוך ה ZIP עם הפתרון ולראות את ההדפסה בפועל.

אפשר להריץ בשורת הפקודה של XV6 פקודת sh ואחריה שוב ps ולראות שנוסף עוד תהליך.

כדאי לעקוב אחרי מספרי ה PID ו PPID ולהבין מי בן של מי ולמה. כמו כן מה התפקיד של תהליכים שונים.

2. הוסיפו את קריאת המערכת הדרושה ואת פקודת המערכת ps (אחרי ההוספה היא תופיע בין פקודות המערכת). כדי ש ps תוכל לבצע את עבודתה, צריך להוסיף קריאת מערכת מתאימה, ראו בהמשך.

כדי שהמטלה לא תיראה קשה, כדאי להתחיל במדריך שעושה משהו דומה(אבל לא מטפל בהדפסת PPID) :

https://github.com/raj-maurya/xv6-public_modifiedOS

ולמצוא בתוך xv6-modified **רוב המטלה עשויה(כולל** Makefile מתאים**). צריך לבצע שינויים קטנים.**

ההסברים בהמשך יהיו הרבה יותר מובנים אחרי זה.

הדפסה מתוך הגרעין נעשית בעזרת cprintf ולא printf הרגילה. זה גם ההיגיון ששם קריאת מערכת של המטלה cps . C - פלט ישיר ל CONSOLE ולא ל STDOUT .

סרטון שמסביר את תהליך ההוספה(באנגלית) כולל שינוי ב Makefile :

https://www.youtube.com/watch?v=21SVYiKhcwM

כדאי להשתמש במדריך שלמעלה במידה נכונה ולא להפוך שת פתרון המטלה להעתק הדבק בלבד!

- **לקריאת המערכת צריך להיות שם cps1xx** , כש xx הן 2 הספרות האחרונות של ת"ז של הסטודנט. לדוגמא, אם ת"ז 3135678**92** אז שם קריאת המערכת צריך להיות **cps192 !!!**
- **מספר קריאת המערכת צריך להיות כמו(שווה) לספרות אחרי cps , 192 בדוגמא הנ"ל.**
- **לקובץ ps.c צריך להיות שם ללא תוספת ספרות(ps.c בלבד) !!!**

אופן ביצוע המטלה:

כדי לבצע את המטלה, צריך להכניס שינויים לקבצים:

proc.c , syscall.c , usys.S , sysproc.c , user.h , defs.h , ליצור קובץ חדש ps.c ולהוסיף אותו לתיקיה של XV6 .

- **בקובץ syscall.c מותר(וצריך) לשנות 2 שורות.**
- **בקבצים defs.h , user.h מותר(וצריך) לשנות רק שורה/הכרזה/הצהרה/פקודה אחת.**
- **בקבצים usys.S , sysproc.c , ps.c , proc.c מותר(וצריך) לשנות בהתאם לנדרש.**

אי עמידה בכללים תביא לפסילת החלק המעשי !!!

דרך הפתרון שונה במקצת ממדריכים, **שימו לב שאסור לשנות(ולהגיש) את הקובץ syscall.h** .

כדי להצליח במטלה ללא אפשרות לשנותו, צריך להבין את התפקיד ואת המשמעות של המבנה הנתונים הרלוונטי בקובץ syscall.c (שורות 112-134) ואיך "לעקוף" את המגבלה הנ"ל. בנוסף, צריך לבצע שינוי(הוספה) בקובץ **usys.S** , יש בו דוגמא לשינוי שנעשה כהוספה "ידנית" של קריאת מערכת FORK , תפעלו בצורה דומה בשביל להוסיף קריאת מערכת הדשה. בקובץ **syscall.h** יש הערה לגבי קריאת מערכת FORK שמדמה את המצב שבמטלה כשאין אפשרות לשנות את תוכנו של הקובץ.

חשוב לציין, **שהמגבלה** נועדה רק לגרום להבנה ולא מהווה דרך מקובלת להכניס שינויים לקוד המערכת.

בנוסף, שימו לב שהפעולה עצמה של קריאת המערכת(מה שהיא מציגה) צריכה להיות שונה ממה שיש במדריכים.

כדי להדפיס את שדה PPID (לא ממומש בקישור הנ"ל) צריך למצוא אותו ב PCB של התהליך- מבנה struct proc בקובץ proc.h , שם השדה שונה(לא PPID) , ניתן למצוא בקלות ע"פ ההערות.

בשביל אחידות הפלטים בבדיקה, הדפסת שורת הכותרת של הפלט צריך לבצע בעזרת:

שימו לב לרווח בין \t ל \t // cprintf("name \t pid \t state \t \t ppid \n");

ואת הפלט עצמו כמו בתמונה ופתרון ביה"ס.

שימו לב שהשדה PPID המודפס של INIT **צריך להיות 0 למרות** שבשדה המחזיק את PPID ב PCB יש מספר אחר, אנחנו מניחים שלאבא של התהליך ראשון במרחב המשתמש יש PID =0 , צריך לממש את זה במטלה ולמצוא את הדרך לזהות את תהליך INIT . אצל השאר בשדה המחזיק את PPID מופיע מספר נכון.

- צריך להדפיס את כל התהליכים הקיימים. בשביל פשטות ואחידות הבדיקה **כל תהליך שלא נמצא במצב RUNNING אמיתי מודפס כ SLEEPING במשמעות NOT RUNNING .**

a. תקראו את ההסבר על תהליך הוספת קריאת מערכת ל XV6 ואת תפקידים של הקבצים הרלוונטיים:
https://viduniwickramarachchi.medium.com/add-a-new-system-call-in-xv6-5486c2437573 הסבר הכי מתאים לצורכי המטלה בין מה שראיתי.

שימו לב שבד"כ בקובץ **sysproc.c** יש רק את "השלד" של קריאת המערכת שקורא לפונקציה עצמה שעושה את העבודה ונמצאת ב **proc.c** . בקישור למעלה קריאת המערכת קצרה מאוד, כמו שלד עצמו ולכן מיקמו אותה ב **sysproc.c** . בפתרון המטלה את הקוד ביצוע ממשי של קריאת המערכת צריך לשים ב **proc.c** .

ומתוך: https://www.ics.uci.edu/~aburtsev/238P/hw/hw5-syscall/hw5-syscall.html

רק פתיח ו קטע Considerations .

שימו לב שתהליך עשיית המטלה דומה, אבל שונה במקצת ממדריכים.

המטרה להבין את התהליך ולהכיר את התפקיד של קבצים שונים.

b. תשנו את ה Makefile בשביל שיתאים לשינויים. עדיף להכיר את השימוש הבסיסי ב Makefile ולבצע את השינויים הנדרשים. במידת הצורך ניתן למצוא את Makefile מתאים בתוך הקבצים של מערכת xv6-modified (קישור).

c. אחרי ביצוע השינויים תריצו את המערכת מחדש, תבדקו שהמערכת החדשה(במקצת) מתפקדת כמצופה. תריצו ps ותראו שהפלט תקין.

d. אחרי סיום המטלה צריכים להיות ברורים המושגים הבאים והבדלים ביניהם:

מצב גרעין, מצב משתמש, קריאת מערכת, למה בכלל במקרה שלנו יש צורך בקריאת מערכת ולא מספיק תוכנית המשתמש, מספר קריאת מערכת, ממשק לקריאת מערכת, אופן הפעלה מעשית של קריאת מערכת(על פי מספר בעזרת INT), למה קריאת מערכת מופעלת בעזרת INT ולא סתם קריאה לפונקציה, פונקציה(קוד) המבצעת את קריאת מערכת, תוכנית המשתמש שמפעילה את קריאת המערכת. לשים לב שב XV6 יש 2 אימג'ים(IMAGES) שמדמים 2 מערכות קבצים, אחת של הגרעין והשנייה של מערכת עצמה עם אפשרות לשמור בה את הקבצים של משתמש.

e. אופציונאלי, אבל חשוב מאוד: לדבג את עליית המערכת כמו שמוסבר בסרטון באתר ולעקוב אחרי השלבים של עליית הגרעין ומעבר למרחב המשתמש.

f. אופציונאלי, אבל חשוב: להכיר את פעולת ה SHELL של המערכת(קובץ sh.c) ובפרט, זיהוי הפקודה, הבדל באופן ביצוע בין פקודות SHELL (הפקודות הפנימיות) לבין פקודות המערכת (יצירת התהליך המבצע בעזרת FORK והחלפת תוכנו לפקודת המערכת בעזרת EXEC).

בדיקה סופית

1. לאחר תיקון הבאג הריצו make clean; make qemu .
   וודאו בפעם נוספת שאתם מסוגלים להריץ את ps ושפלט של הפקודה תקין
2. כעת המשיכו לבדיקות regression שמטרתן לוודא כי כל הבדיקות (tests) עוברים בהצלחה. לשם כך כבו את QEMU.
3. התקינו expect (אם לא התקנתם לפני)- שפת סקריפט למיכון (automation) אינטראקציה עם פקודות shell ותוכנות אשר מורצות משורת הפקודה. לשם כך הריצו משורת הפקודה

```
sudo apt-get install expect
```

4. הריצו משורת הפקודה של מערכת ubuntu 16.04 מתוך התיקייה של xv6 פקודה הבאה:

```
./runtests.exp my.log
```

5. ודאו כי תוכנת סקריפט יצאה עם סטאטוס 0 מיד לאחר סיומה(ערך הסיום נכתב גם לקובץ).

```
$ echo $?
0
```

6. להכרות כללית עם expect מומלץ(לא חובה) לקרוא את פרק ד' של "חומר רקע".

פתרון ביה"ס

להריץ מתוך תיקיית הממ"ן את make clean ו אחריו make qemuss

הגשה

יש להגיש אך ורק את הקבצים שהיה צורך לשנות/להוסיף :

(ps.c , proc.c , syscall.c , usys.S , sysproc.c , user.h , defs.h ו Makefile) בלבד.

אין להגיש קבצים נוספים ו/או מקומפלים. ראה הוראות הגשה כלליות בחוברת הקורס.

את הקובץ/הקבצים המוגש/ים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). עדיף להכין את הארכיון בפורמט זיפ ZIP ב WINDOWS . אם אין אפשרות, ע"י הרצת הפקודה הבאה משורת הפקודה של Ubuntu : zip exYZ.zip <exYZ files>

<u>הערה חשובה : בתוך כל קובץ קוד שאתם מגישים יש לכלול כותרת(בהערה) הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.</u>

בדיקה לאחר הגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי' לבדוק תקינות של הקבצים המוגשים (לדוגמא, שניתן לקרוא אותם). בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים :

- פתיחת ארכיון *exXY.zip* בספרייה חדשה (*new folder*) .
- יצירת ספריה חדשה עם הקוד המקורי של *xv6*
- העתקת הקובץ המוגש לספרייה עם הקוד המקורי של *xv6*
- הרצת *make qemu* ווידוא שכל ה *target* נוצר ללא שגיאות וללא *warnings*
- הרצת בדיקות רלוונטיות : וידוא תקינות הריצה של החלק המעשי

החלק העיוני (30%)

שאלה 2 (5%)

א) מהי פעולת ה TRAP? תארו מתי היא מתבצעת ומה קורה בעת ביצועה.

ב) הסבירו מה קורה בעת הקריאה לפונקצית write של ה C library . בפרט הסבירו כיצד עוברים הפרמטרים של ה write למערכת הפעלה Linux וכיצד המערכת מטפלת ב write.

ג) מה ההבדל בין write ל printf? תוכלו להיעזר בקבצי מקור של C library מ www.gnu.org/software/libc

שאלה 3 (15%)

במערכת הפעלה LINUX קיים מנגנון איתותים(סיגנלים) SIGNALS חשוב ושימושי **שהשימוש בו בתוך אפליקציה** יכול להיות סינכרוני ו/או א-סינכרוני.

כדי להכיר את המנגנון תקראו את המאמר : https://cs341.cs.illinois.edu/coursebook/Signals

תבינו היטב את ההבדל בין שימוש סינכרוני ל א-סינכרוני.

בין קבצי הממ"ן יש קבצי קוד שמדגימים את השימוש הבסיסי בשתי צורות הסיגנלים.
הקבצים באים רק לעזור, אין חובה להריץ אותם ולבצע את מה שבהערות, אך זה עוזר להבנת העניין.
אם הכל ידוע או מובן בלי דוגמא, אין צורך אפילו להסתכל על הקודים האלה.

א) ממשו את **סמפור בינארי יחיד**(ללא שם) על בסיס שימוש סינכרוני בסיגנל וללא שימוש במנגנוני סנכרון נוספים. הסמפור מיועד לשימוש ע"י **מספר תהליכונים THREADS של אותו תהליך(לא תהליכים שונים)**.

ממשו את הפונקציות הבאות **בשפת C:**

- void sem_init(int status) לאתחול הסמפור למצב מסומן(פתוח, =ל 1) או לא מסומן(סגור, =ל 0). כולל הכרזה ואתחול של משתנים לוקאליים.
- void sem_down() להורדה(סימון כתפוס, המתנה) של סמפור.
- void sem_up() לשחרור(סימון כדלוק/פנוי) של סמפור.
- הכרזה ואתחול של משתנים שחייבים להיות גלובאליים עם ציון בצורה חופשית שהם גלובליים.

שימו לב שבצורת השימוש המדוברת, הסיגנל צריך להיות חסום לפני הפעלת sigwait .
כמו כן, **שימו לב** שפונקציה **kill שולחת סיגנל** לתהליך המיועד **ולא הורגת**(חוץ מסיגנל ספציפי)!
הסיגנל שנשלח לתהליך "מגיע" לכל ה THREADS שלו וה THREAD ויקבל(יתפוס) אותו ראשון, ינקה(ימחק) אותו מרשימת הסיגנלים הממתינים.
הדרך **האוניברסאלית** לשליחת הסיגנל ללא משמעות קבוע במערכת לתהליך עצמו היא:
kill(getpid() , SIGUSR1); . אפשר להשתמש בכל אפשרות תקינה אחרת.
לצורך התרגיל הבסיסי ובשביל הפשטות אין צורך לחסום בנפרד לכל תהליכון THREAD , אלא לחסום בתוך התהליך שיחול על כל ה THREADS שייוצרו בתוכו(לחסום בתוך הפונקציה sem_init(int status)).
אפשר(אך לא חובה) להיעזר בדוגמא:
https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-sigwait-wait-asynchronous-signal
אין חובה לכתוב תוכנית שלמה הכוללת יצירת THREADS והסנכרון ביניהם בעזרת הפונקציות שמימשתם, אך זה מוסיף להבנה וניסיון. בנוסף, כתיבתה והרצתה תאפשר לבדוק את נכונות הפתרון.

ב) האם בדרך דומה(ולא מסובכת משמעותית יותר) ניתן לממש את הסמפור המיועד לסנכרון בין מספר **תהליכים? תנו נימוק מילולי, אין צורך בכתיבת קוד.**

שאלה 4 (5%)

תקראו את מאמר שמסכם את הבדלים בין user threads ו kernel threads ואת מאמר שמסביר מודלים שונים של מימוש מנגנון threads . תענו לשאלות הבאות:

א. האם **M:1 model** מאפשר לנצל מספר ליבות במעבד CPU cores? נמקו.

ב. האם ב **M:1 model** חסימת אחד מ user threads תגרום לחסימת כל התהליך? נמקו.

ג. מה המשמעות של מושגים user thread ו kernel thread ב **1:1 model** ?

שאלה 5 (5%)

הוכיחו כי בפתרון של Peterson ל 2 תהליכים, תהליכים אינם ממתינים זמן אינסופי על מנת להיכנס לקטע קריטי. בפרט הוכיחו כי תהליך שרוצה להיכנס לקטע קריטי לא ממתין יותר ממה שלקח לתהליך אחר להיכנס ולעזוב את הקטע הקריטי.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או pdf עם **שם exYZ.docx או exYZ.pdf** (כאשר YZ הנו מספר המטלה) **בתוך אותו zip עם החלק המעשי.** אין להגיש יותר מזיפ אחד בסה"כ!

מטלת מנחה (ממ״ן) 12

הקורס : ״מערכות הפעלה״

חומר הלימוד למטלה : ראו פירוט בסעיף ״רקע״

מספר השאלות : 5 | משקל המטלה : 10

סמסטר : 2024א | מועד אחרון להגשה : 27.01.2024

הגשת המטלה : שליחה באמצעות מערכת המטלות המקוונת באתר הבית של הקורס. הסבר מפורט ב״נוהל הגשת מטלות מנחה״.

החלק המעשי (70%)

כללי

בתרגיל זה נכיר מרחבי משתמש מבודדים במערכת הפעלה XV6. מופעים של מרחבי משתמש מבודדים הנקראים קונטיינרים (Containers) שמשתייכים לשיטות וירטאליזציה ברמת מערכת הפעלה.

במערכת הפעלה XV6 מימוש הקונטיינרים מתבסס על אפשרויות ניהול של שני מרחבי משאבים (namespaces) הקיימים ב XV6 והם pid namespace ו mount namespace. בתרגיל זה נכיר את מימוש מרחבי משתמש מבודדים במערכת הפעלה 6xv ובפרט למימוש של pid namespace .

מטרות

- הכרת ההיבטים המעשיים של מימוש קונטיינרים
- היכרות עם מימוש של pid namespace במ"ה xv6
- היכרות עם unshare - קריאת מערכת (system call) ליצירת namespace חדש.
- היכרות עם שינויים בקוד של קריאת מערכת fork ליצירת תהליך חדש ושיוך של התהליך שנוצר למרחב משאבים.
- התאמה ושינוי קריאת מערכת למודעות ותמיכה במנגנון ההפרדה.

רקע

א) פרק Makefile מחוברת ״Ubuntu 16.04 programming environment, making first steps״ (הורידו את החוברת מאתר הקורס).

ב) ״Running and debugging xv6.pdf״(באנגלית, כולל הוראות דיבוג משורת הפקודה) ו/או ״XV6 Instalation and EclipseConfig.pdf״ (בעברית, מאחד התלמידים, כולל דיבוג ב ECLIPSE) מתוך maman12.zip.

ג) פרקים 0(עד PIPES בע' 13) ופרק 1 מתוך https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf.
עם דגש על "Process overview" בסוף פרק 1 .

ד) **חוברת** "xv6 containers, namespaces and cgroups" עם דגש על מימוש פונקצית fork בפרק "PID namespaces in xv6" .

ה) expect - שפת סקריפט אינטראקטיבית : https://likegeeks.com/expect-command

ו) במידת הצורך סרטונים על שימוש ודיבוג ב XV6 מאתר הקורס(בחלק ממ"נים). מספרי הממ"נים והדוגמאות בהם לא זהים לתוכן המטלות.

תיאור המשימה

בקובץ maman12.zip תמצאו תיקיה עם מערכת ההפעלה xv6. תיקיה זו שונה מזו שקיבלתם בממ"ן 11 מכיוון שמדובר במערכת שנוספה לה תמיכה בקונטיינרים. יש להשתמש בתיקיה הזאות.

הפעם נרחיב את קריאת המערכת ממ"ן 11 ונתאים אותה לקונטיינרים. הפרטים בהמשך.

ממ"ן זה מתבסס על הידע אשר נצבר במהלך העבודה על הממ"ן הקודם.

הסבר מפורט

1. הפעילו את xv6 (ממ"ן 12) כמתואר בסעיף ב' של "חומר קרע". הריצו משורת הפקודה את:

   make clean; make qemu

   תצרו ותתנסו בשימוש בקונטיינרים כמו שמתואר בסעיף ד' של חומר הרקע.

2. אם תעתיקו את **השינויים מהקבצים שהגשתם** במ"ן 11 **לתוך הקבצים התואמים** בתיקייה של XV6 החדשה(**שימו לב שלא מדובר על העתקת הקבצים עצמם**) ותנסו לבנות(make qemu), הפלט יהיה(שגיאת קומפילציה):

```
proc.c: In function 'cps':
proc.c:826:88: error: 'struct proc' has no member named 'pid'
 printf("%s \t %d  \t SLEEPING \t %d \t \t %d \t %d \n", p->name, p->pid, extp
                                                                          ^
proc.c:827:86: error: 'struct proc' has no member named 'pid'
 cprintf("%s \t %d  \t RUNNING \t %d \t \t %d \t %d \n", p->name, p->pid, extp
                                                                          ^
<builtin>: recipe for target 'proc.o' failed
make: *** [proc.o] Error 1
user@ubuntu:~/xv6-ns$
```

3. סיבת השגיאה כמו שאפשר לראות בפלט היא שמבנה של proc (PCB של תהליך) השתנה בעקבות הוספת התמיכה בקונטיינרים.

4. תבצעו התאמת קריאת המערכת ממ"ן 11 , תוסיפו "מודעות" לקונטיינרים ותרחיבו אותה.

   הפלט התקין אמור להיות:

```
user@ubuntu:~/xv6
init: starting sh
$ ps
main-loop: WARNING: I/O thread spun for 1000 iterations
name     pid     state           extpid          ppid    cputime
init     1       SLEEPING        1               0       55965
sh       2       SLEEPING        2               1       15027
ps       3       RUNNING         3               2       6978
$ pouch start c1
Pouch: c1 starting
$ ps
name     pid     state           extpid          ppid    cputime
init     1       SLEEPING        1               0       55965
sh       2       SLEEPING        2               1       26633
ps       7       RUNNING         7               2       27202
pouch    5       SLEEPING        5               1       12803
sh       1       RUNNING         6               5       3771386
$ pouch connect c1

tty0 connected
ps
name     pid     state           extpid          ppid    cputime
init     1       SLEEPING        1               0       55965
sh       2       RUNNING         2               1       5162115
ps       2       RUNNING         9               1       6639
pouch    5       SLEEPING        5               1       12803
sh       1       SLEEPING        6               5       25363971
$
```

שדה EXTPID הוא PID "חיצוני"- מחוץ לקונטיינר. בשביל זה תבינו היטב את החוברת מסע' ד' של חומר הרקע ובמיוחד פרק **namespaces in xv6** ממנה. המידע הדרוש קיים ב PCB .

את מבנה ה PCB אפשר לראות בחוברת או ב struct proc בקובץ proc.h **. כעת תמיכה בקונטיינרים מקוננים לא מופעלת ב XV6 ואפשר להתבסס על זה בפתרון המטלה(אין קונטיינר בתוך קונטיינר שבתוך קונטיינר) .**

אפשר להתבסס על הפתרון מממ"ן 11 , להעתיק חלק מקבצים שהגשתם בו ולשנות חלק. הפעם הוספת קריאת המערכת צריכה להיות בדרך מקובלת, בעזרת שינוי בקובץ **syscall.h וצריך להגיש גם אותו!**

אפשר להיעזר ב https://github.com/raj-maurya/xv6-public_modifiedOS . לא צריך(והפעם אסור) לבצע את כל "המעקף" בעקבות איסור לשנות את **syscall.h** ולא לבצע שינויים בקבצים רלוונטיים שהיו בעקבות זה.

בקיצור, הפעם מוסיפים קריאת מערכת כמו שמקובל.

בשביל אחידות הפלטים בבדיקה הדפסת שורת הכותרת של הפלט צריך לבצע בעזרת:

שימו לב לרווח בין \t ל \t // cprintf("name \t pid \t state \t \t extpid \t ppid \t cputime \n");

5. לאחר תיקון הבעיה תבנו ותריצו את XV6 מחדש, תפעילו את PS , תצרו קונטיינר, תתחברו אליו, תפעילו את PS בתוך קונטיינר, תבדקו שהפלט תקין ב 2 המקרים. שימו לב לתקינות שדה EXTPID .
   תוודאו שאתם מבינים מתי ובאיזה מצב PID ו EXTPID שווים ומתי לא.

הערה לא חשובה: כתוצאה מצורת מימוש קלט-פלט עם תמיכה בקונטיינרים, ה TTY הלא פעיל גורם ל sh(shell) שמחובר אליו להיות פעיל(רץ) כל הזמן ולכן זמן CPU שלו עולה(סוג של בג).

- צריך להדפיס את כל התהליכים הקיימים. בשביל פשטות ואחידות הבדיקה **כל תהליך שלא נמצא במצב RUNNING אמיתי מודפס כ SLEEPING במשמעות NOT RUNNING .**

הסיבה לזה שבמימוש של קונטיינרים שנוספו ל XV6 חלק מתהליכים שלפי הגיון אמורים להיות SLEEPING , למעשה רוב הזמן לא במצב הזה.

6. שמות הפונקציות והקבצים

- **לקריאת המערכת צריך להיות שם cps1xx** , כש xx הן 2 הספרות האחרונות של ת"ז של הסטודנט. לדוגמא, אם ת"ז 3135678**92** אז שם קריאת המערכת צריך להיות **cps192 !!!**
- **מספר קריאת המערכת צריך להיות כמו(שווה) לספרות אחרי cps , 192 בדוגמא הנ"ל.**
- **לקובץ ps.c צריך להיות שם ללא תוספת ספרות(ps.c בלבד) !!!**

בדיקה סופית

1. לאחר תיקון באגים הריצו make clean; make qemu. וודאו בפעם נוספת שאתם מסוגלים
   - ליצור קונטיינר/ים
   - להתחבר אל הקונטיינר/ים הנוצרים ולהפעיל בהם פקודות
   - להשמיד קונטיינרים אשר נוצרו
2. כעת המשיכו לבדיקות regression שמטרתן לוודא כי כל הבדיקות (tests) עוברים בהצלחה. לשם כך כבו את QEMU.
3. התקינו expect **(אם עדיין לא הותקן)** - שפת סקריפט למיכון (automation) אינטראקציה עם פקודות shell ותוכנות אשר מורצות משורת הפקודה. לשם כך הריצו משורת הפקודה

```
sudo apt-get install expect
```

4. הריצו משורת הפקודה של מערכת ubuntu 16.04 פקודה הבאה:

```
./runtests.exp my.log
```

5. ודאו כי תוכנת סקריפט יצאה עם סטאטוס 0 מיד לאחר סיומה

```
$ echo $?
0
```

6. להכרות כללית עם expect מומלץ לקרוא את פרק ה' של "חומר רקע".

פתרון ביה"ס

להריץ מתוך תיקיית הממ"ן את make clean ו אחריו make qemuss

הגשה

יש להגיש אך ורק את הקבצים שהיה צורך לשנות/להוסיף:

(ps.c , proc.c , syscall.h , syscall.c , usys.S , sysproc.c , user.h , defs.h ו Makefile) בלבד.

אין להגיש קבצים נוספים ו/או מקומפלים. ראה הוראות הגשה כלליות בחוברת הקורס. ראו הוראות הגשה כלליות בחוברת הקורס.

את הקובץ/הקבצים המוגש/ים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). עדיף להכין את הארכיון בפורמט זיפ ZIP ב WINDOWS . אם אין אפשרות, ע"י הרצת הפקודה הבאה משורת הפקודה של Ubuntu: zip exYZ.zip <exYZ files>

<u>הערה חשובה: בתוך כל קובץ קוד שאתם מגישים יש לכלול כותרת(בהערה) הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.</u>

בדיקה לאחר הגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי לבדוק תקינות של הקבצים המוגשים (לדוגמא, שניתן לקרוא אותם). בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון *exXY.zip* בספרייה חדשה (*new folder*).
- יצירת ספריה חדשה עם הקוד המקורי של *xv6*
- העתקת הקובץ המוגש לספרייה עם הקוד המקורי של *xv6*
- הרצת *make qemu* ווידוא שכל ה *target* נוצר ללא שגיאות וללא *warnings*
- הרצת בדיקות רלוונטיות: וידוא תקינות הריצה של החלק המעשי

החלק העיוני (30%)

שאלה 2 – (5%)

האם ניתן וכדאי להשתמש באלגוריתם LRU (בצורתו הטהורה) לצורך פינוי דפים (page eviction)?

שאלה 3 – (5%)

האם דף יכול להיות בו זמנית בשתי קבוצות עבודה (working sets)? נמקו.

שאלה 4 – (10%)

טבלת הדפים של תהליך במערכת עם זיכרון וירטואלי נראית כך. כל המספרים הם דצימליים, מתחילים מאפס, וכל הכתובות הן כתובות של בייט בזיכרון. גודל הדף הוא 1024 בייטים.

Page Number	Valid bit	Frame Number
0	1	4
1	1	3
2	0	1
3	1	4
4	0	1
5	1	0

א. לאילו כתובות פיזיות, אם יש כאלו, ימופו הכתובות הוירטואליות הבאות: 1042, 2211, 5399.

ב. האם יש שגיאות בטבלת הדפים, אם כן מה הן.

שאלה 5 – (10%)

א. תארו את ההליך תרגום כתובת לוגית בעלת 32 סיביות לכתובת פיזית כשבמערכת גודל דפי זיכרון היא 4MB (4 מגה בית).

ב. חשבו את גודל טבלת הדפים בהנחה שאורך שורה בטבלה הוא 4B (4 בית) ושכל תהליך מקבל את מרחב זיכרון וירטואלי מרבי.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או pdf עם **שם exYZ.docx או exYZ.pdf** (כאשר YZ הנו מספר המטלה) **בתוך אותו zip עם החלק המעשי.** אין להגיש יותר מזיפ אחד בסה"כ!

מטלת מנחה (ממ"ן) 13

הקורס: "עקרונות מערכות הפעלה"

חומר הלימוד למטלה: ראו פירוט בסעיף "רקע"

מספר השאלות: 5 | משקל המטלה: 10

סמסטר: 2024א | מועד אחרון להגשה: 12.02.2024

הגשת המטלה: שליחה באמצעות מערכת המטלות המקוונת באתר הבית של הקורס. הסבר מפורט ב"נוהל הגשת מטלות מנחה".

החלק המעשי (70%)

במערכת הפעלה xv6 מימוש הקונטיינרים מתבסס על אפשרויות ניהול של שני מרחבי משאבים (namespaces) הקיימים ב xv6 והם pid namespace ו mount namespace . בתרגיל זה נצלול למימוש מרחבי משתמש מבודדים במערכת הפעלה xv6 ובפרט למימוש של mount namespace .

מטרות

- הכרה של מערכת הפעלה xv6
- הכרת ההיבטים המעשיים של מימוש קונטיינרים
- היכרות עם מימוש של mount namespace במ"ה xv6
- תיקון באג שגורם ל kernel panic בעת השימוש ב mounted filesystem

רקע

א) פרק Makefile מחוברת "Ubuntu 16.04 programming environment, making first steps" (הורידו את החוברת מאתר הקורס).

ב) "Running and debugging xv6.pdf"(באנגלית, כולל הוראות דיבוג משורת הפקודה) ו/או "XV6 Instalation and EclipseConfig.pdf" (בעברית, מאחד התלמידים, כולל דיבוג ב ECLIPSE) מתוך maman13.zip .

ג) פרק 6 מתוך https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf.

ד) חוברת "xv6 containers, namespaces and cgroups" עם דגש על מימוש mount namespaces בפרק Mount namespaces in xv6 .

ה) expect - שפת סקריפט אינטראקטיבית: https://likegeeks.com/expect-command

ו) במידת הצורך סרטונים על שימוש ודיבוג ב XV6 מאתר הקורס(בחלק ממ"נים). מספרי הממ"נים והדוגמאות בהם לא זהים לתוכן המטלות.

תיאור המשימה

בקובץ maman13.zip תמצאו תיקיה עם מערכת ההפעלה xv6. תיקיה זו שונה במקצת מזו שקיבלתם בממ״ן קודם מכוון ששתלנו במערכת באג.

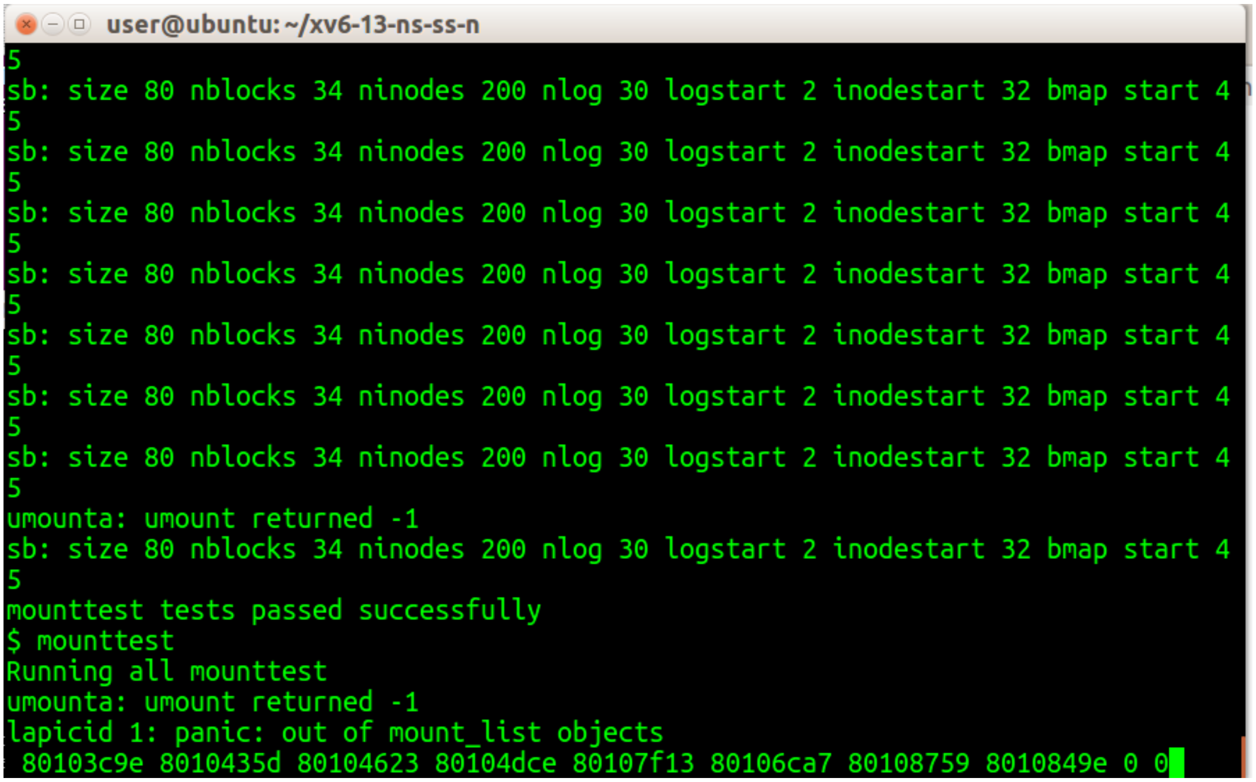
הפעם הבעיה מתגלה בעת ניסיון להריץ סדרת טססטים משורת הפקודה של xv6 המופעלים ע"י קובץ הרצה mounttest.

בהמשך הפרטים למציאה ותיקון הבאג.

שימו לב, ממ"ן זה מתבסס על הידע אשר נצבר במהלך העבודה על הממ"נים קודמים.

הסבר מפורט

1. הפעילו את מערכת ההפעלה xv6 כמתואר בסעיף ב' של "חומר קרע".
2. הריצו משורת הפקודה את התוכנית mounttest **פעמיים!!!** הבעיה תתגלה בהרצה השנייה. היא תתגלה גם בהרצת טסט מלא של EXPECT .
   בפתרון ביה"ס הבעיה לא קיימת לא בהרצה השנייה ולא בהרצות נוספות.



```
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
umounta: umount returned -1
sb: size 80 nblocks 34 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 4
5
mounttest tests passed successfully
$ mounttest
Running all mounttest
umounta: umount returned -1
lapicid 1: panic: out of mount_list objects
 80103c9e 8010435d 80104623 80104dce 80107f13 80106ca7 80108759 8010849e 0 0
```

- ניתן לראות כי טסט גורם לקריסת גרעין ומתקבלת הודעה של kernel panic.
- כדי לסיים ולחזור לשורת הפקודה צריך לסגור את שלון XV6 של QEMU .

3. כמו שאפשר לראות מההודעה, מערכת לא מוצאת mount_list object שפנוי להקצות.
4. הבעיה היא שבעת עזיבת mount namespace (פונקציה namespaceleave מקובץ namespace.c קוראת ל mount_nsleave מקובץ mnt_ns.c) הדברים לא התעדכנו כשורה, מה מדובר על בעיה בפונקציה mount_nsleave מקובץ mnt_ns.c .
5. תקנו את הבעיה. התיקון לא ארוך. בסה"כ צריך להוסיף לפונקציה שכמעט ריקה 3 פעולות שונות(אחת מהן בעזרת קריאה לפונקציה אחרת שכבר קיימת בתוך אותו קובץ עם השם שמראה מה התפקיד שלה). אם תעשו את התיקון בשלבים, יכולות להתגלות בעיות נוספות, צריך לתקן את כולם.

   ברמה לוגית פונקציה mount_nsleave כשהתהליך האחרון עוזב את mount_ns אמורה לרוקן את ה mounts הקיימים ולרשום שאין mounts. אם תהליך שאינו אחרון עוזב את mount_ns צריך לבצע שינוי רישום בהתאם. את המידע הנוסף אפשר למצוא בחוברת "xv6 containers, namespaces and cgroups" ובקובץ mnt_ns.h .

בדיקה סופית

1. לאחר תיקון הבעיה הריצו make clean; make qemu. וודאו בפעם נוספת שאתם מסוגלים לבצע רצף פקודות אשר גרם לקריסתה של המערכת בתחילת הדרך.
2. כעת המשיכו לבדיקות regression שמטרתן לוודא כי כל הבדיקות (tests) עוברים בהצלחה. לשם כך כבו את QEMU.
3. התקינו expect (אם עדיין לא הותקן) - שפת סקריפט למיכון (automation) אינטראקציה עם פקודות shell ותוכנות אשר מורצות משורת הפקודה. לשם כך הריצו משורת הפקודה

```
sudo apt-get install expect
```

4. הריצו משורת הפקודה של מערכת ubuntu 16.04 פקודה הבאה:

```
./runtests.exp my.log
```

5. ודאו כי תוכנת סקריפט יצאה עם סטטוס 0 מיד לאחר סיומה

```
$ echo $?
0
```

6. להכרות כללית עם expect מומלץ לקרוא את פרק ה' של "חומר רקע".

פתרון ביה"ס

להריץ מתוך תיקיית הממ"ן את make clean ו אחריו make qemuss

הגשה

יש להגיש את הקובץ mnt_ns.c בלבד. אין להגיש קבצים נוספים ו/או מקומפלים. ראה הוראות הגשה כלליות בחוברת הקורס.

את הקובץ/הקבצים המוגש/ים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). עדיף להכין את הארכיון בפורמט זיפ ZIP ב WINDOWS . אם אין אפשרות, ע"י הרצת הפקודה הבאה משורת הפקודה של Ubuntu : zip exYZ.zip <exYZ files>

<u>הערה חשובה: בתוך כל קובץ קוד שאתם מגישים יש לכלול כותרת(בהערה) הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.</u>

בדיקה לאחר הגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי לבדוק תקינות של הקבצים המוגשים (לדוגמא, שניתן לקרוא אותם). בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון *exXY.zip* בספרייה חדשה (*new folder*) .
- יצירת ספריה חדשה עם הקוד המקורי של *xv6*
- העתקת הקובץ המוגש לספרייה עם הקוד המקורי של *xv6*
- הרצת *make qemu* ווידוא שכל ה *target* נוצר ללא שגיאות וללא *warnings* .
- הרצת בדיקות רלוונטיות: וידוא תקינות הריצה של החלק המעשי.

החלק העיוני (30%)

שאלה 2 (5%)

לפי מדיניות חדשה של תזמון זרוע הדיסק, בקשות מוחזקות בתור לפי סדר הגעתן. הראשונה המטופלת היא הבקשה אשר הגיע האחרונה. מדיניות זו נקראת last in first out – LIFO.

א) מהו היתרון של המדיניות הזאת?

ב) מהו החיסרון של המדיניות הזאת?

שאלה 3 (10%)

בכל רגע נתון גודלו של קובץ יכול לנוע בין Kb4 ל Mb4. באיזו מבין 3 מדיניות הייתם בוחרים בשביל ש:

א. בזבוז מקום יהיה מינימאלי.

ב. זמני גישות סדרתיות יהיו מינימאליים.

- הקצאה רציפה
- רשימה מקושרת
- i-node

הניחו הנחות סבירות נוספות אם נדרש. הדגימו חישובים עליהם התבססה ההחלטה.

שאלה 4 – (10%)

תקראו את ההסבר של מודל האבטחה וניהול תהליכים במערכת אנדרואיד(פרקים 10.8.10-10.8.12 בספר הקורס,אין צורך להתייחס לפרטים בפרק 10.8.11 , אלא רק למודל עצמו).

תענו על השאלות הבאות:

א) במה המודל המתואר דומה לקונטיינרים הנידונים בחלק המעשי של מטלות הקורס ובמה הוא שונה ממנו?

ב) באיזה סוג בידוד(הפרדה) מדובר?

ג) מה היתרון בשיטת ניהול זיכרון ראשי (Copy On Write) COW שמערכת אנדרואיד משתמשת בה בשביל מימוש מודל תהליכים שלה?

שאלה 5 (5%)

תארו בקצרה את 2 שיטות מניעת ניסיון שינוי ה capability שניתנה למשתמש במערכת ללא תמיכת חומרה.

ציינו בקצרה את היתרון והחיסרון של כל אחת מהן.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או pdf עם **שם exYZ.docx או exYZ.pdf** (כאשר YZ הנו מספר המטלה) **בתוך אותו zip עם החלק המעשי.** אין להגיש יותר מזיפ אחד בסה"כ!