

מטלת מנהה (ממ"ז) 01

הקורס : "מערכות הפעלה"

חומר הלימוד למטלה : ראו פירוט בסעיף "רקע"

משקל המטלה : 10

מספר השאלות : 5

מועד אחרון להגשה : 29.11.2025

סמסטר : 2026א

הגשת המטלה : שליחה באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.
הסביר מפורט ב"נהל הגשת מטלות מנהה".

החלק המעשי (70%)

כללי

בתרגיל זה נזכיר את מבנה של מערכת הפעלה בכלל ומערכת הפעלה XV6 בפרט. מערכת הפעלה XV6 היא מערכת משפחת 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. היא לא מושלמת ויש בה באגים!

מטרות:

- הכנת מערכת הפעלה xv6
- הכנת היבטים המעשיים של שימוש קריאות מערכת
- הכנת מבני נתונים שונים של מערכת הפעלה
- הוספת קריית מערכת חדשה
- הוספת פקודות מערכת חדשה זו שمدפיסה את מצב תהליכיים במערכת
- התנסות בבניה והרצה של מערכת הפעלה בצוות הקורובה למציאות(כשלא כל המידע זמין וצריך להבין ולמצוא אותו לבד) !

רקע

א) פרק מוחברת Makefile "Ubuntu 24.04 programming environment, making first steps" (הורידו את החוברת לאתר 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 ubuntu .maman01.zip . התקינו את המכונה הווירטואלית לאתר הקורס, סיסמת המנהל

- ג) פרק 0 (עד PIPEs בעי 13), פרק 1 ופרק 3 (עד protection X86 בעי 40) מותוך
<https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf>
- אין צורך להתייחס לענייני ניהול זיכרון ראשי.
- ד) expect - שפת סкриיפט אינטראקטיבית :
<https://likegeeks.com/expect-command> אינטראקטיבית עם פקודות shell ותוכנות אשר מורות מושרכות הפוקודה.
- ה) במידת הצורך סרטונים על שימוש ודיבוג ב XV6 מאתר הקורס(בחלק ממ"נ). מספרי הממ"נים והדוגמאות בהם לא זהים לתוכן המטלות העכשווי.

תיאור המשימה

הוספה פקודה מערכת חדשה zk שמדפיסה את מצב תהליכיים במערכת.
 בקובץ `maman01.zip` תמצאו ספרייה עם מערכת הפעלה xv6 שאין בה פקודה zk ואין קריאת מערכת
 הדרושה לביצועה, המטרה היא להוסיף אותן.

הסבר מפורט

- הפעילו את מערכת הפעלה xv6 כמתואר בסעיף ב' של "חומר רקע". הריצו את תוכנת zk , תקבלו את הפלט הבא(אולי במצב שונה) :

```
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 zk כלל לא קיימת במערכת.

אחרי הוספה הפקודה תוצאה הרצה צריכה להיות :

```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ps
name      pid      state          ppid      size
init        1      SLEEPING        0       12288
sh         2      SLEEPING        1       16384
ps         3      RUNNING         2       12288
$
```

אפשר להריץ את פתרון ביה"ס לפי ההוראות ולראות את ההדפסה בפועל.

פתרון ביה"ס

להריץ מותוך תיקיית הממ"ן את make clean או אחרי make qemuss

אפשר להריץ בשורת הפוקודה של XV6 פקודה sh ואחריה שב zk ולראות שנוסף עוד תהליך.
 כדאי לעקוב אחרי מספרי ה PID ו PPID בפתרון ביה"ס ולהבין מי בן של מי ולמה. כמו כן מה התפקיד של תהליכיים שונים.

2. הוסיפו את קריאת המערכת הדרישה ואת פקודת המערכת `dc`(אחרי ההוספה היא תופיע בין פקודות המערכת). כדי ש `dc` יוכל לבצע את עводתה, צריך להוסיף קריאת מערכת מותאמת, **ראו בהמשך**. כדי שהמטלה לא תיראה קשה, כדאי להתחיל במדריך שעשו משה דומה(אבל לא מטפל בהדפסת : (PPID

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

אפשר למצוא בתחום [xv6-modified](#) רוב המטלות שעשו (כולל Makefile מותאים).
צריך לבצע שינויים קטנים.
במשך הסברים יותר מפורטים.

הדפסה מתוך הגרעין נעשית בעזרת `cprintf` ולא `printf` הרגילה. זה גם היגיון שהשם של קריאת המערכת במטלה הוא `cps` . "c" - פلت ישיר ל CONSOLE ולא ל STDOUT .
סרטון שסביר את התהליך ההוסף(אנגלית) כולל שינוי ב Makefile :

<https://www.youtube.com/watch?v=21SVYiKhewM>

כדי להשתמש במדריך שלמעלה במידה נcona ולא להפוך שת פתרון המטלת להעתיק הדבק בלבד !

- **לקריית המערכת צריך להיות שם `cps1xx` , כש xx 2 הספרות האחרונות של ת"ז של הסטודנט. לדוגמה, אם ת"ז 313567892 אז שם קריית המערכת צריך להיות `cps192` !!!**
- **מספר קריית המערכת צריך להיות כמו(שווה) למספרות אחרי `cps` , 92 בדוגמה הנ"ל.**
- **לקובץ `c` צריך להיות שם ללא תוספת ספרות(`c.ps` בלבד) !!!**

אופן ביצוע המטלת:

כדי לבצע את המטלת, צריך להכניס שינויים לקבצים:
`defs.h` , `user.h` , `sysproc.c` , `usys.S` , `syscall.c` , `proc.c`
לתיקיה של XV6.

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

אי עמידה בכללים תביא לפיטול חלק המעשי !!!

דרך הפתרון שונה במקצת ממדריכים, שימוש לב שאסור לשנות(ולהגייש) 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) וアイיך "לעקורף" את המגבלה הנ"ל. בנוסף, צריך
לבצע שינוי(הוסף) בקובץ `S.usys` , יש בו דוגמא לשינוי שנעשה כהוספה "ידנית" של קריית מערכת
, `FORK`

תפעלו בצורה דומה בשביל להוסיף קריית מערכת חדשה. בקובץ `syscall.h` יש הערכה לגבי קריית
מערכת `FORK` שمدמה את המכובד במטלה כאשר אין אפשרות לשנות את תוכנו של הקובץ.
חשוב לציין, **שהגבלה נועדה רק לגרום להבנה ולא מהו דרך מקובלת להכניס שינויים לקובד
המערכת.**

בנוסף, שימוש לב שהפעולה עצמה של קריית המערכת (מה שהיא מציגה) נדרש להיות שונה ממה שיש במדריכים.

כדי להציג את שדה PPID ו SIZE (גודל זיכרונוenthalik) שלא מומשים בקישור הניל, צריך למצואו
אותם ב PCB של התהלהיק - מבנה struct proc בקובץ proc.h , שומות השדות שונות , אבל ניתן למצאו
בקלות ע"פ הערות.

שימוש לברוח בין \t ל\t `cprintf("name \t pid \t state \t size\n";`

והפלט עצמו צריך להיות כמו בתמונה ופתרון ביה"ס, **אחרת הבדיקה האוטומטית תיכשל**.

שיםו לב שהשדה PPID המודפס של INIT **חייב להיות 0** למרות שבשדה המחזיק את PPID ב-PCB יש מספר אחר, אנחנו מניחים שלABA של התהיליך הראשון במרחב המשתמש יש $PID = 0$, חייב למש את זה במללה ולמצוא את הדרכן ליזהות את תהליך INIT. אצל השאר בשדה המחזיק את PPID מופיע המספר הנכון.

- צרי להזפיס את כל התהליכיים הקיימים (שנוצרו במערכת), ללא שורות שלא משקפות את התהליכיים הקיימים בטבלת התהליכיים). בשביל פשוטות ואחדירות הבדיקה **כל תהליך שלא נמצא במצב RUNNING אמייתי מוחפס כ SLEEPING** (במשמעות NOT RUNNING).

שלבי הביצוע:

a. תקרוואו את ההסביר על תהליך הוספת קריית מערכת ל XV6 ואת תפקידים של הקבצים הרלוונטיים :

<https://viduniwickramarachchi.medium.com/add-a-new-system-call-in-xv6-5486c2437573> הסבר הכי מתאים לאורכי המטלה ביו מה שראיתי.

ביצוע ממשי של קריית המערכת צריך לשים ב-`c`.
ב-`proc` .

[וומתוך](https://www.ics.uci.edu/~aburtsev/238P/hw/hw5-syscall/hw5-syscall.html)

רַק פִּתְחָה וְהַקְטָעָה . Considerations

שימו לב שהליך עשיית המטלה דומה, אבל שונה במקצת ממדריכים.
המטרה להביו את התהליך ולהזכיר את התפקיד של קבאים 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 החדש(במקרה מתפקידת במצבה. תריצו גם ותראו שהפלט תקין.

d. אחרי סיום המטלה צריכים להיות ברורים המושגים הבאים והבדלים ביניהם :
 מצב גרעין, מצב משתמש, קריית מערכת, ומה בכלל במקרה שלו יש צורך בקריאת מערכת ולא מספיק תוכנית המשתמש, מספר קריית מערכת, ממשק לקריית מערכת, אופן הפעלה

מעשית של קריית מערכת(על פי מספר בעורת INT), למה קריית מערכת מופעלת בעורת INT ולא סתם קרייה לפונקציה, פונקציה(קוד) המבצעת את קריית מערכת, תוכנית המשמש ש愧עה את קריית המערכת.

חשוב לשים לב שב 6X יש 2 אימג'ס(IMAGES) שمدמיים 2 מערכות קבועים, אחת של הגערין והשנייה של מערכת עצמה עם אפשרות לשמור בה את הקבצים של משתמש.

e. אופציונלי, אבל חשוב מאוד : לדגש את עלילית המערכת כמו שמוסבר בסרטון באתר ולעקוב אחרי השלבים של עלילית הגערין ומעבר למרחב המשמש.

f. אופציונלי, אבל חשוב : להזכיר את פעולה ה SHELL של המערכת(קובץ sh.c) ובפרט, זיהוי הפקדה, הבדל באופן ביצוע בין פקודות SHELL (הפקדות הפנימיות) לבין פקודות המערכת (יצירת התהיליך המבוצע בעורת FORK והחלפת תוכנו לפקודה המערכת בעורת EXEC).

בדיקות סופית

- לאחר תיקון הבאג הריצו make clean; make qemu וודאו בפעם נוספת מסוגלים להריץ את qemu ושלט של הפקודה תקין.
- cut המשיכו לבדוק regression שטרתן לוודא כי כל הבדיקות (tests) עוברים בהצלחה. לשם כך כבו את QEMU.
- הריצו משורת הפקודה של מערכת ubuntu 24.04 מתוך התיקייה של 6x פקודה הבאה :

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

אם צריך, תנתנו הרשות הריצה לקובץ runtests.exp .

- ודאו כי תוכנת סקריפט יוצאה עם סטאטוס 0 מיד לאחר סיום(ערך הסיום נכתב גם לקובץ).

```
$ echo $?  
0
```

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

פתרון בית"ס

הරיצו מתוך תיקיית הממ"ן את make qemuss ואחריו make clean ואחריו

הגשה בזיפ אחד בלבד עם החלק העיוני!

יש להגשים אך ורק את הקבצים שהוא צריך לשנות/להוסיף :

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

אין להגשים קבצים נוספים ואו מקומפלים. ראה הוראות הגשה כליליות בחוברת הקורס.
את הקובי/הקבצים המוגשים יש לשים בקובץ ארכיוון בשם exYZ.zip (כאשר YZ הם מספר המטלח). עדיף להכין את הארכיוון בפורמט ZIP ב WINDOWS . אם אין אפשרות, ע"י הרצת הפקודה הבאה משורת הפקודה של Ubuntu : zip exYZ.zip <exYZ files> .
הערה חשובה: בתוך כל קובץ קוד שאתם מגישים יש לכלול כתורת(בהערה) הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.

בדיקה לאחר הגשה

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

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

החלק העיוני (30%)

שאלה 2 (5%)

- א) מהי פעולה TRAP? תארו מתי ובשביל מה היא מתבצעת ומה קורה בעת ביצועה.
ב) הסבירו מה קורה בעת הקריאה לפונקציה write של ה C library . בפרט הסבירו כיצד עוברים הפרמטרים של write למערכת הפעלה Linux ובאיזה המערך 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 קיימים מנגנון איתותיים (סיגנלים) SIGNALS חשוב ושימושו שהשימוש בו בתחום אפליקציה יכול להיות סינכרוני ו/או א-סינכרוני.

כדי להכיר את המנגנון תקרוו את המאמר :
<https://cs341.cs.illinois.edu/coursebook/Signals>

tabino hyitb at habdil bin shymosh sinchroni li a-sinchroni.

בין קבצי הממן יש קבצי קוד שמדגים את השימוש הבסיסי בשתי צורות הסיגנלים.

הקבצים באים רק לעוזר, אין חובה להריץ אותם ולבצע את מה שבהערות, אך זה עוזר בהבנת העניין.
אם הכל ידוע או מובן בily דוגמא, אין צורך אפילו להסתכל על הקודים האלה.

א) ממשו את "סמפור" ב内幕ית (לא שם) על בסיס שימוש סינכרוני בסיגנל ולא שימוש **בשות** מנגנון סינכרון נסף(גם לא בהמתנה פיעילה). ה"סמפור" מיועד לשימוש ע"י מספר תהליכי THREADS של אותו תהליך(לא תהליכי שונים).
המירباتות בಗל/sha"סמפור" המיועד אינו חייב לפול אט מבני הנתונים הפנימיים הלא חיוניים לתפקודו.
משמעותו את הפונקציות הבאות בשפת C :

- void sem_init(int status) – אתחול הסמפור למצב מסומן(פתוח, = 1) או לא מסומן(סגור, = 0).
- void sem_down() להורד(סימון כתפוס, המתנה) של סמפור.
- void sem_up() לשחרור(סימון כدلוק/פינוי) של סמפור.
- הרצה ואתחול של משתנים שחיברים להיות גלובליים עם ציון בצוירה חופשית שהם גלובליים.

שימוש לב שיטות השימוש המדוברת, הסיגנל צריך להיות חסום לפני הפעלת *sigwait* .

כמו כן, שימוש לב שפונקציה *kill* שלוחת סיגナル לתהליך המיועד ולא הורגת(חוץ מסיגנל ספציפי)!

הסיגנל שנשלח לתהליך "מגיע" לכל הTHREAD שלו והוא THREAD ויקבל(יתפס) אותו ראשון, ינקה(ימחק) אותו מרשימת הסיגנלים הממתינים. הדרך האוניברסלית לשיחת הסיגナル ללא משמעות קבועה במערכת לתהליך עצמו היא :
`(kill(getpid(), SIGUSR1);` אפשר להשתמש בכל אפשרות תקינה אחרת.
לצורך התרגיל הבסיסי ובשביל הפשטות אין צורך לחסום בנפרד לכל תהליכי THREAD , אלא לחסום בתוך התהליך שיחול על כל הTHREAD שייצרו 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 והסנהרנו ביניהם בעזרת הפונקציות שמיימות, אך זה מוסיף להבנה וניסיון. בנוסף, כתיבתה והרצתה תאפשר לבדוק את נכונות הפתרון.

ב) האם בדרך דומה(ולא מסובכת ממשוערת יותר) ניתן למש את הסטפורה המיועדת לסyncronization בין מספר תהליכיים ? **תנו נימוק מילולי, אין צורך בכתיבה קוד.**

שאלה 4 (5%)

תקרוו את <https://www.geeksforgeeks.org/difference-between-user-level-thread-and-kernel-level-thread>
kernel threads ו user threads המסביר את הבדלים בין ו
<https://www.tpointtech.com/why-must-user-threads-be-mapped-to-kernel-thread>
הסביר מודלים שונים של מיפוי מנגןן threads .
תענו על השאלות הבאות :

- א. האם M:1 model מאפשר לנצל מספר ליבוט במעבד CPU cores ? נמקו .
- ב. האם ב M:1 model חסימת אחד מ user threads תגרום לחסימת כל התהליכיים ? נמקו .
- ג. מה המשמעות וההשפעה של מושגים kernel thread ו user thread ב 1:1 model ?

שאלה 5 (5%)

א) הוכיחו כי בפתרון של Peterson ל 2 תהליכיים(ע' 52 במדריך הלמידה), תהליכיים אינם ממוחזקים זמן אינסופי על מנת להיכנס לקטע קריטי. בפרט הוכיחו כי תהליך שורצוה להיכנס לקטע קריטי לא ממוחזק יותר מכמה שלקח לתהליכיים אחרים להיכנס ולעוזב את הקטע הקריטי.

ב) האם פתרון יישאר תקין אם יחליפו את סדר ביצוע 2 שורות הקוד(הראשונה תבצע אחריה השנייה) :
interested [#] = TRUE;
turn = #;

הסבירו למה כן או הפריכו ע"י דוגמא נגדית.
דבר כזה אכן יכול לקרוא בעקבות אופטימיזציה(Instruction reordering).

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או pdf עם שם exYZ.pdf או exYZ.docx (כאשר YZ הם מספר המטלח)
בתוך אותו zip עם החלק המעשי. אין להגיש יוזף אחד בסה"כ !