

האוניברסיטה הפתוחה

20594

מערכות הפעלה

חוברת הקורס – סתיו 2019א

כתב: דוד שריאל

אוקטובר 2018 – סמסטר סתיו- תשע"ט

פנימי – לא להפצה.

© כל הזכויות שמורות לאוניברסיטה הפתוחה.

תוכן העניינים

א	אל הסטודנט
ב	1. לוח זמנים ופעילויות
ד	2. תיאור המטלות
ד	3. התנאים לקבלת נקודות זכות
ה	4. הדרכה לפתרון מטלות התכנות
1	ממ"ן 11
7	ממ"ן 12
13	ממ"ן 13

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"ס בכתובת:

<http://telem.openu.ac.il>

מידע על שירותי ספרייה ומקורות מידע שהאוניברסיטה מעמידה לרשותכם, תמצאו באתר הספרייה באינטרנט www.openu.ac.il/Library.

אפשר לפנות אלי בדואר אלקטרוני davidsa@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 מאוד להשתמש באתר הקורס ובכל אמצעי העזר שבו וכמובן לפנות אלינו במידת הצורך.

אל אתר הבית של הקורס ניתן לגשת מדף הבית של החטיבה למדעי המחשב:

<http://telem.openu.ac.il/cs>

בברכת לימוד פורה ומהנה,

דוד שריאל

מרכז ההוראה בקורס

1. לוח זמנים ופעילויות (20594/ א'2019)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	19.10.2018-14.10.2018	ראו חלוקה שבועית באתר הקורס		
2	26.10.2018-21.10.2018	ראו חלוקה שבועית באתר הקורס		
3	2.11.2018-28.10.2018	ראו חלוקה שבועית באתר הקורס		
4	9.11.2018-4.11.2018	ראו חלוקה שבועית באתר הקורס		
5	16.11.2018-11.11.2018	ראו חלוקה שבועית באתר הקורס		ממ"ן 11 15.11.2018
6	23.11.2018-18.11.2018	ראו חלוקה שבועית באתר הקורס		
7	30.11.2018-25.11.2018	ראו חלוקה שבועית באתר הקורס		
8	7.12.2018-2.12.2018 (ב-ו חנוכה)	ראו חלוקה שבועית באתר הקורס		

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

לוח זמנים ופעילויות - המשך

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
9	14.12.2018-9.12.2018 (א-ב חנוכה)	ראו חלוקה שבועית באתר הקורס		ממ"ן 12 13.12.2018
10	21.12.2018-16.12.2018	ראו חלוקה שבועית באתר הקורס		
11	28.12.2018-23.12.2018	ראו חלוקה שבועית באתר הקורס		
12	4.1.2019-30.12.2018	ראו חלוקה שבועית באתר הקורס		
13	11.1.2019-6.1.2019	ראו חלוקה שבועית באתר הקורס		ממ"ן 13 10.1.2019
14	18.1.2019-13.1.2019	ראו חלוקה שבועית באתר הקורס		

מועדי בחינות הגמר יפורסמו בנפרד

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

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 ויוחזרו לך בצירוף הערות המתייחסות לתשובות.

לכל מטלה נקבע משקל. יש לצבור 36 נקודות. חובה להגיש את כל המטלות.

ללא צבירת 36 נקודות בהגשת מטלות
לא ניתן יהיה לגשת לבחינת הגמר

לתשומת לבכם!

ציון סופי מחושב רק לסטודנטים שעברו את בחינת הגמר בציון 60 ומעלה והגישו את כל המטלות 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 במשקל כולל של 36 נקודות לפחות עם ציון מינימלי של 60 נקודות בכל אחת מהמטלות שהוגשו.

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 קרע" 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 מצויים 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 ואומנם 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 ולהגישם בקובץ בשם makefile.
17. חובה להשתמש בדגל (flag) "-Wall" בזמן קומפילציה התוכנית.

בנוס

במקרים יוצאי דופן, כאשר מוגשת תוכנית טובה במיוחד או כזו שעושה למעלה ממה שנדרש, תישקל האפשרות להוסיף עד 5 נקודות בנוס. 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"

משקל המטלה: 12

מספר השאלות: 6

מועד אחרון להגשה: 15.11.2018

סמסטר: 2019א

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

החלק המעשי (70%)

כללי

בממ"ן זה עליכם לממש שתי ספריות לעבודה עם תהליכונים (threads) ברמת המשתמש (user-level). אחת הספריות תממש סמפורים בינאריים לעבודה עם קטעים קריטיים וספריה שנייה תממש מספר פונקציות המאפשרות יצירה והרצה של תהליכונים ברמת המשתמש ומדידת זמן הריצה ל profiling של תוכניות המשתמשות בספרייה זו.

מטרה

- הכרת ההיבטים המעשיים של מימוש תהליכונים ברמת המשתמש
- שימוש בסיגנלים
- שימוש ב-non-local branching
- timers
- profiling
- קטעים קריטיים

רקע

א) פרקים 2.3.5, 2.5.1, 2.2.1, 2.2.2, 2.2.3 של Tanenbaum, "Modern operating systems".
ב) [פרק 24.3](#) של The GNU C library
ג) [פרק 23.4](#) של The GNU C library
ד) פרק "Libraries" מחוברת "Ubuntu 12.04 programming environment, making first steps"
ה) man pages של Linux - מידע על קריאות מערכת ופונקציות הבאות: alarm, sigfillset, sigaction, swapcontext, getcontext, makecontext, steitimer, kill, getpid

תיאור המשימה

בממ"ן זה עליכם לממש שתי ספריות סטטיות:

(1) libut.a - ספרייה פשוטה לעבודה עם תהליכונים ברמת המשתמש, שה-API שלה מוגדר בקובץ ut.h. קובץ זה מכיל תיאור מפורט לגבי תפקידה של כל פונקציה שעליכם לממש (אין לשנות קובץ זה, אך כמובן שבמידת הצורך ניתן להגדיר פונקציות עזר בקובץ C). הספרייה תתמוך רק בפעולות הבסיסיות ביותר, שהן יצירת התהליכונים, הרצתן ותזמון. על מנת שלא להפוך את המשימה למסובכת מדי, הספרייה תממש רק מודל פשוט של שימוש בתהליכונים המבוסס על ההנחות הבאות:

1. כל תהליכון מריץ פונקציה אינסופית שמקבלת פרמטר יחיד מטיפוס int ומחזירה void.
לא נטפל בסיום תהליכונים ובבדיקת סטטוס היציאה.
2. אין הוספה דינאמית של תהליכונים. המשתמש קודם יצור את כל התהליכונים, וא"כ יקרא ל-ut_start כדי להריץ את כל התהליכונים.
3. כל התהליכונים הם בעלי אותה עדיפות. תזמון התהליכונים יהיה בשיטת round-robin, כאשר גודל ה-quantum הוא שנייה אחת.
4. שימו לב שלא הגדרנו מצב blocked לתהליכונים. זאת מפני שבמודל שלנו ההנחה היא שתהליכונים לא מבצעים פעולות הגורמות לחסימה (blocking calls). לאחר הביצוע של ut_start, כל תהליכון יכול להיות באחד משני המצבים - רץ או מוכן לריצה. וודאו שאתם מבינים כי בהנחה כזאת כלל לא נצטרך לשמור את מצב התהליכונים מכוון שמנגנון התזמון שלנו תמיד יבחר את התהליכון הבא בתור ויריץ אותו.

בשלב ראשון של הכנת הממ"ן קראו את הסעיפים א, ב, ג) מחומר רקע והריצו והבינו את התוכניות demo1.c, demo2.c, demo3.c שסיפקנו לכם. התוכנית הראשונה מדגימה כיצד מתאפשר לשים "שעון מעורר" לתהליך ב-Linux. התוכנית השנייה מרחיבה את הראשונה ומדגימה כיצד אפשר ליצור 2 ניבים של ריצה בתוכנית באמצעות המנגנון המכונה non-local jumping. התוכנית השלישית מדגימה כיצד אפשר לבצע רישום של זמן ריצה של תוכנית לצורך profiling.

בשלב שני עליכם לממש את הממשק המוגדר הקובץ ut.h. הממשק מגדיר פונקציות לאתחול הספרייה, ליצירת תהליכון חדש ולהרצת התהליכונים שנוצרו. ut.h מממשת את מודל התהליכונים הפשוט שתיארנו לעיל. שימו לב ש demo2.c מדגימה כיצד ליצור 2 תהליכונים. אתם מתבקשים להכליל את הפתרון למספר תהליכונים. לכן, לאחר שהשלמתם את שני השלבים הקודמים כל שנותר לעשות הוא להעביר חלקים של הקוד מ demo2.c ל ut.c עם שינויים מינוריים.

ב `ut.h` עליכם לממש את `ut_get_vtime` המשמשת למדידת זמן הריצה של תהליכון. השתמשו בקוד של `demo3.c` שמשמשת בשעון מסוג `ITIMER_VIRTUAL` שישלח סיגנל `SIGVTALRM` כל 10 (100msec פעמים לשנייה). בכל פעם שהסיגנל מתקבל, יש להוסיף 100msec לשדה הזמן הווירטואלי של התהליכון האקטיבי בזמן קבלת הסיגנל.

(2) `libbinsem.a` - ספרייה של סמפורים בינאריים שנועדו לשימוש ע"י התהליכונים מהסעיף הראשון. הקובץ `binsem.h` מגדיר את הטיפוס של סמפור בינארי ומתאר את הפונקציות הרלוונטיות (אין לשנות קובץ זה). עליכם לממש את הפונקציות שמוצהרות בקובץ זה, תוך כדי שימוש במקרו `(xchg)` המוגדר בקובץ `atomic.h`. כמו כן, תסתמכו על העובדה שהחלפת התהליכונים מתבצעת כתוצאה מקבלת הסיגנל `SIGALRM` כדי לממש את ההמתנה ב-`binsem_down()` (כפי שפורט בסעיף הקודם, לתהליכונים שעליכם לממש לא מוגדר מצב `blocked`. יש לדמות את המצב ע"י כך שתהליכון ה"מתין" בסמפור מייד לאחר קבלת ה-CPU ישלח סיגנל `SIGALRM` שיגרום להפעלת המתזמן ומעבר לתהליכון הבא).

לצורך הבדיקה של שתי הספריות סיפקנו לכם פתרון של בעיית הפילוסופים הסועדים בקובץ `ph.c`. בעיית הפילוסופים הסועדים מתוארת בפרק 2.5.1 בספר של Tanenbaum. כל פילוסוף רץ כתהליכון נפרד (לצורך זה משתמשים בספריית התהליכונים שהממשק שלה הוגדר ב `ut.h`. התהליכונים משתמשים בסמפורים שהוגדרו ב `binsem.h`). התוכנית תופעל ע"י הפקודה "`<ph>N`", כאשר N (בטווח מ-2 עד 32) הוא מספר התהליכונים (פילוסופים). התוכנית תופסק ע"י הקשת "Ctrl-C", לפני היציאה יודפסו זמני השימוש ב-CPU של כ"א מהתהליכונים.

כדי לקמפל את תוכנית הפילוסופים עם הספריות שתכתבו, תשתמשו ב `Makefile` שסיפקנו. שימו לב שעליכם לשנות את ה `Makefile` לפני ההגשה (ראו סעיף "הגשה" בהמשך).

טיפול בשגיאות

יש תמיד לבדוק את ערכי החזרה של קריאות מערכת ופונקציות סטנדרטיות של C. במקרה של כשלון, יש לפעול כפי שמוגדר בקבצים `ut.h` ו-`binsem.h`. בנוסף, במקרה של כשלון המערכת תוך כדי ביצוע של `signal handler` בספריית התהליכונים, יש להודיע על השגיאה באמצעות `perror` ולהפסיק את הביצוע ע"י `exit(1)`.

הגשה

יש להגיש **כל** קבצי הקוד Makefile המייצר שתי ספריות סטטיות: libut.a ו-libbinsem.a. אין להגיש קבצים מקומפלים. ראה הוראות הגשה כלליות בחוברת הקורס.

את הקבצים המוגשים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). הכנת קובץ ארכיון מתבצעת ע"י הרצת הפקודה הבאה משורת הפקודה של Ubuntu:
zip exYZ.zip <ExYZ files>

הערה חשובה: בכל קובץ קוד שאתם מגישים יש לכלול כותרת הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.

פתרון ביה"ס

קיבלתם את שתי הספריות, libut.a ו-libbinsem.a, כפי שמומשו על ידינו. תוכלו להיעזר בהן בהכנת הממ"ן. למשל, לקמפל את תוכנית הבדיקה ph עם ספרייה אחת משלכם (שאותה אתם רוצים לבדוק) וספרייה השנייה של פתרון ביה"ס.

הערה: תוך כדי העבודה על הממ"ן תצטרכו להכיר ולהבין מספר נושאים שאינם פשוטים - זהו הקושי של ממ"ן זה. יחד עם זאת, הממ"ן לא ידרוש מכם הרבה עבודת תכנות. ניתן לממש את שתי הספריות בכ-100 שורות קוד בסה"כ.

החלק העיוני (30%)

שאלה 2 (10%)

א) מהי פעולת ה TRAP (TRAP instruction). תארו מתי היא מתבצעת ומה קורא בעת ביצועה.
ב) הסבירו מה קורה בעת הקריאה לפונקציית write של ה C library. בפרט הסבירו כיצד עוברים הפרמטרים של ה write למערכת הפעלה Linux וכיצד המערכת מטפלת ב write. יש התייחס הן למקרה של [legacy system calls](http://www.gnu.org/software/libc) והן ל [fast system calls](http://www.gnu.org/software/libc).
ג) מה ההבדל בין write ל printf? תוכלו להעזר בקבצי מקור של C library מ <http://www.gnu.org/software/libc>

שאלה 3 (5%)

הסבר את מדוע פתרון התור (strict alternation), איננו מהווה פתרון סביר. איזה תנאים הוא מפר.

שאלה 4 (10%)

תקראו פרק 3 של [המאמר](#) שדן בנושא הוספת תהליכונים כספריה לשפה שלא תמכה בהם מלכתחילה והסברו מדוע תקן של Pthreads אינו מתאר באופן פורמאלי את מודל הזיכרון ואת הסמנטיקה של המקביליות הממומשות ב Pthreads. כיצד מפתחי התקן מסבירים מהו מודל הזיכרון בכל זאת?

שאלה 6 (5%)

הוכיחו כי בפתרון של Peterson תהליכים אינם ממתנינים זמן אינסופי על מנת להיכנס לקטע קריטי. בפרט הוכיחו כי תהליך שרוצה להיכנס לקטע קריטי לא ממתין יותר ממה שלוקח מתהליך אחר להיכנס ולעזוב את הקטע הקריטי.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או כקובץ pdf. שם הקובץ צריך להיות exYZ.pdf או exYZ.doc (כאשר YZ הנו מספר המטלה).

בדיקה לאחר ההגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי ולבדוק שהקבצים אכן הוגשו באופן תקין ושניתן לקרוא אותם. בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון exXY.zip בספרייה חדשה (new folder).
- וידוא שכל הקבצים הדרושים נוצרו בספרייה בה פתחתם את הארכיון.
- הרצת make ווידוא שכל ה targets נוצרו ללא שגיאות וללא warnings
- הרצת בדיקות רלוונטיות לוודא תקינות הריצה של החלק המעשי

תרגיל העשרה

מטרה

עליכם להוסיף קריאת מערכת חדשה למערכת הפעלה לימודית xv6.

הנחיות

א. קראו פרק 0 מתוך <https://pdos.csail.mit.edu/6.828/2017/xv6/book-rev10.pdf>

ב. תורידו את הקוד של xv6 מ <https://github.com/mit-pdos/xv6-public>

ג. קראו ופעלו על פי ההנחיות הבאות כדי להפעיל/להשתמש/לנפות שגיאות ב xv6:

<https://github.com/ranl/xv6-public/wiki/Compile,-Run-and-Play-Around>

ד. ראו כיצד מוסיפים תוכנית משתמש ל xv6 לפי ההנחיות שב

<https://github.com/ranl/xv6-public/wiki/User-Program>

ה. למדו כיצד מוסיפים קריאת מערכת חדשה ל xv6: [https://github.com/ranl/xv6-](https://github.com/ranl/xv6-0public/wiki/sys_hello_world)

[0public/wiki/sys_hello_world](https://github.com/ranl/xv6-0public/wiki/sys_hello_world)

ו. כעת, כשאתם מכירים איך מוסיפים קריאת מערכת חדשה, הוסיפו קריאת מערכת שבם

`halt` אשר שולחת ל `QEMU` מחרוזת `poweroff` מיוחדת הנתמכת על ידי `QEMU` (אך

לא ע"י חומרה אמיתית)

```
/* This is a special power-off sequence supported by
Bochs and
   QEMU, but not by physical hardware. */

char *p = "Shutdown";
for( ; *p; p++)
    outb(0x8900, *p);
```

ז. לבסוף, הוסיפו ל xv6 את תונית המשתמש אשר קוראת לקריאת המערכת החדשה:

```
/* halt.c */

#include "types.h"
#include "stat.h"
#include "user.h"

int
main(int argc, char *argv[])
{
    halt();
    return 0;
}
```

ח. אין צורך להגיש את הקוד של התרגיל הזה.

מטלת מנחה (ממ"ן) 12

הקורס: "מערכות הפעלה"

חומר הלימוד למטלה: ראו פירוט בסעיף "רקע"

משקל המטלה: 12

מספר השאלות: 5

מועד אחרון להגשה: 13.12.2018

סמסטר: 2019א

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

החלק המעשי (80%)

בחלק זה של המטלה נממש מנגנון של job control ב command interpreter פונקציונליות מוגבלת.

מטרת התרגיל: תהליכים, תקשורת בין התהליכים, job control.

רקע

1. סיפקנו את הקובץ shell.c אותו אתם אמורים לשנות ולהרחיב ובפרט להוסיף את המנגנון של job control. קמפלו והריצו את התוכנית. עיינו בקובץ shell.pdf להסברים. כל השינוי מסתכם במספר מועט של שורות קוד אך כדי לבצע אותו עליכם להבין מספר נושאים להלן.
2. http://www.gnu.org/software/libc/manual/html_node/Executing-a-File.html הסבר עם פונקציות ממשפחת exec (אפשר להשתמש בפונקציה לבחירתכם). כמו כן סיפקנו את הקובץ exec.c שאפשר לקמפל ולהריץ.
3. פרקים 24.7.2, 24.7.3 מ http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הסבר על פונקציות signal, sigemptyset, sigfillset, sigaddset, sigprocmask, sigsuspend. סיפקנו קובץ suspend.c. תקמפלו תריצו והבינו.
4. פרק http://www.gnu.org/software/libc/manual/html_mono/libc.html#Pipes-and-FIFOs עם הסבר על פונקציות pipe לתקשורת בין התהליכים. סיפקנו קובץ pipe.c. תקמפלו תריצו והבינו.

5. פרק 13.12 מ

[https://www.gnu.org/software/libc/manual/html_mono/libc.html#Duplicating-](https://www.gnu.org/software/libc/manual/html_mono/libc.html#Duplicating-Descriptors)

[Descriptors](https://www.gnu.org/software/libc/manual/html_mono/libc.html#Duplicating-Descriptors) עם הסבר על פונקציות dup. תקמפלו ותריצו את dup.c שספקנו כדי לראות שימוש ב dup.c. מהווה וריאציה של pipe.c מהסעיף הקודם ומדגימה כיצד ניתן ליצור ערוץ תקשורת בין שני תהליכים בצורה שהיא שקופה לתהליכים עצמם.

6. פרק 14.1 מ http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הסבר על הפונקציה chdir.

7. פרקים 27.6.2, 27.7.3 ו 27.7.2 מ

http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הפונקציות getpgrp, setpgrp, tcgetpgrp.

8. כמו כן יש להיזכר בפונקציות wait, fork.

כמו ניתן לקבל מידע על הפונקציות הנ"ל מה man של LINUX.

תיאור המשימה

סיפקנו את הקובץ shell.c אותו אתם אמורים לשנות ולהרחיב. בפרט עליכם לממש מנגנון של job control ב command interpreter בשם smash (small shell). אחרי ההרחבה smash יהיה מסוגל:

1) לאפשר שרשור של לפחות 2 פקודות.

2) לתמוך בפקודות פנימיות exit ו cd.

3) להריץ תוכניות ברקע ובזמן אמת (background ו foreground).

4) לתמוך בפקודות bg, fg, jobs ולהגיב לסיגנלים של job control.

קיבלתם קובץ shell.c המממש פונקציונליות 1, 2, 3. כתבו עבורו Makefile שמייצר קובץ הרצה smash והריצו אותו משורת הפקודה:

```
maman12$ ./smash
```

כמו כן קיבלתם את קובץ smash_SSol המממש גם את 4. הריצו אותו משורת הפקודה:

```
maman12$ ./smash_SSol
```

במטלה הזאת עליכם לכתוב כ 40 שורות קוד למימוש בפקודות של job control (סעיף ד). שאר הפונקציונליות כבר ממומשת (סעיפים א, ב, ג). אבל למימוש ה job control עליכם להבין כיצד פועלים שאר הדברים. המיקום של השורות אותן תצטרכו לממש מופיע בקובץ shell.pdf עם ה והפסאודו-קוד. מי שמכיר מהו שרשור הפקודות ב shell ואת הפקודות fg, bg, jobs יכול לעבור ישירות לקריאת ההסברים הקובץ shell.pdf. אחרת, מומלץ קודם לקרוא הסברים מטה.

הרצת תוכניות ברקע ובזמן אמת

הרצת תוכנית(תוכניות) בזמן אמת (foreground) גורמת ל command interpreter להמתין עד סיום התוכנית (תוכניות). למשל

```
# ls
# ps | wc -l
```

הן דוגמאות להרצת תוכניות בזמן אמת.

הרצת תוכנית(תוכניות) ברקע (background) לא גורמת ל command interpreter להמתין עד לסיום התוכנית (התוכניות) שרצות ברקע. הרצת תוכניות ברקע תבצע ע"י הוספת "&" בסוף שורת הפקודה. למשל:

```
# find / home -name Makefile -print &
# chown -R root:root /tmp&
```

שרשור של פקודות

smash מאפשר שרשור של לפחות שני פקודות בשורת פקודה אחת. השרשור מתבצע ע"י סימן "|" (pipeline) בין הפקודות. משמעות השרשור היא שפלט של הפקודה הראשונה מהווה קלט לפקודה השנייה. כך למשל הרצת

```
# cat /etc/passwd | wc -l
```

גורמת לספירת כמות השורות בקובץ /etc/passwd. הפקודה "cat /etc/passwd" מדפיסה את תוכן הקובץ /etc/passwd ל stdout. באמצעות ה "|" אפשר "לומר" ל smash להפנות את הפלט של cat לתוכנית wc. והתוצאה שהיא כמות השורות בקובץ תודפס על הצג.

תמיכה בפקודות של job control

smash יתמוך בפקודות הבאות:

1) jobs – הפקודה תגרום להדפסה של כל התהליכים המושהים ושל כל התהליכים שרצים ברקע אשר הורצו בעבר מתוך smash. תהליך מושהה הוא תהליך שהיה רץ בזמן אמת ואשר הושהה (למשל באמצעות Ctrl-Z). אם תהליך כלשהו רץ בזמן אמת, הצירוף Ctrl-Z משהה את ריצתו ומחזיר את שורת ה prompt של smash. מכאן שאם רוצים להריץ פקודת jobs ייתכן ויהיה צורך להשהות קודם תהליך שרץ בזמן אמת. לדוגמא:

```
# find / home -name Makefile -print
<Ctrl-Z>
[1] Stopped          find /home -name Makefile -print
# jobs
[1] Stopped          find /home -name Makefile -print
#
```

פקודת jobs נותנת לתהליכים מספר סידורי פנימי (ששונה בד"כ מ pid של תהליך) לפיו ניתן לזהות באופן יחיד כל תהליך שעדיין לא הסתיים ואשר הורץ מתוך smash.

(2) fg %N – הפקודה תגרום להרצת תהליך [N] בזמן אמת. כך בדוגמא הקודמת הרצת
fg %1
תעביר את find לרוץ בזמן אמת ולא תחזיר את ה prompt של ה smash עד לסיום ה find
או עד להשהיתו הבאה.

(3) bg %N – הפקודה תעביר את התהליך [N] ממצב מושהה למצב רץ ברקע.

תמיכה בפקודות פנימיות

smash יתמוך בשתי פקודות פנימיות :

- (א) exit – בעקבות הקשת הפקודה יסיים smash את פעולתו.
- (ב) cd - בעקבות הקריאה לפקודה זו ישנה smash את ספרית העבודה הנוכחית שלו.

טיפול בשגיאות

smash צריכה לתת הודעות שגיאה על כשלון של קריאות מערכת או פונקציות שמכילות קירות מערכת. במקרה של שגיאה פטאלית יש לצאת עם סטטוס 1 (ע"י exit(1)).

הגשה

יש להגיש כל קבצי הקוד ו Makefile המייצר קובץ הרצה smash. אין להגיש קבצים מקומפלים. את הקבצים המוגשים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). הכנת קובץ ארכיון מתבצעת ע"י הרצת הפקודה הבאה משורת הפקודה של Linux:
zip exYZ.zip <ExYZ files>

הערה חשובה: בכל קובץ קוד שאתם מגישים יש לכלול כותרת הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.

פתרון ביה"ס

קיבלתם את קובץ smash_SSol כפי שמומש על ידינו.

החלק עיוני (20%)

שאלה 1 – (5%)

מצב לא בטוח איננו גורר באופן מיידי את מצב הקיפאון. תנו דוגמא לכך.

שאלה 2 – (5%)

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

שאלה 3 – (5%)

תארו מצב שבו התמיכה בזיכרון וירטואלי מסתברת כרעיון לא מוצלח. מה אפשר לשפר במצב שתיארתם אם אין תמיכה בזיכרון הוירטואלי?

שאלה 4 – (5%)

בעזרת תוכניות העזר `file` ו `size` גלו מהו גודלן הממוצע והחציון של קבצי הרצה במערכת הפעלה שסיפקנו לכם (Ubuntu 16.04). הסתכלו רק על קבצי ההרצה (לא על קבצי scripts) בספריות:

- `/bin`
- `/usr/bin`

הדגימו את החישוב של הגודל האופטימאלי של דפים במערכת בהתבסס על גודל ה `text segment` של קבצי ההרצה שמצאתם? הניחו שגודל של `entry` בטבלת הדפים הוא 4 בתים. הניחו שלכל קבצי ההרצה הסתברות זהה לרוץ ושהם מקבלים יחס זהה מבחינת מערכת ההפעלה. קחו בחשבון את הריסוק הפנימי.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או כקובץ `pdf`. שם הקובץ צריך להיות `exYZ.pdf` או `exYZ.doc` (כאשר YZ הנו מספר המטלה).

בדיקה לאחר ההגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי ולבדוק שהקבצים אכן הוגשו באופן תקין ושניתן לקרוא אותם. בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון `exXY.zip` בספרייה חדשה (*new folder*).
- וידוא שכל הקבצים הדרושים נוצרו בספרייה בה פתחתם את הארכיון.
- הרצת `make` ווידוא שכל ה `targets` נוצרו ללא שגיאות וללא `warnings`
- הרצת בדיקות רלוונטיות לוודא תקינות הריצה של החלק המעשי

בנוס

מימוש חלק זה מקנה עד 2 נקודות מגן בבחינה הסופית. למען הסר ספק, ניתן לקבל את המגן רק בתנאי שציון המטלה הוא לפחות 90.

מטרה

עליכם לממש קריאת מערכת *ptrace* להתקדמות צעד אחד צעד בתוכנית של מערכת הפעלה לימודית xv6.

הנחיות

א. קראו פרק 1 ו *Appendix B* מתוך <https://pdos.csail.mit.edu/6.828/2017/xv6/book-rev10.pdf>

ב. תורידו את הקוד של xv6 מ <https://github.com/mit-pdos/xv6-public>

ג. קראו ופעלו על פי ההנחיות הבאות כדי להפעיל/להשתמש/לנפות שגיאות ב xv6:

<https://github.com/rml/xv6-public/wiki/Compile,-Run-and-Play-Around>

ד. ממשו את התרגילים המפורטים ב:

<https://docs.google.com/document/d/1DPWIRJMIDAuJ2U46uc9NJ9xt9ye3t9mCbIguCjmd4TE/edit?usp=sharing>

ה. את הקבצים ששיניתם בסעיף הקודם (כולל ה *Makefile*) יש להגיש בקובץ *bonus.tar* דרך מערכת המטלות יחד עם הקבצים של ממ"ן 12 ולכתוב *email* ל davidsa@openu.ac.il על עצם הגשת הבונוס.

מטלת מנחה (ממ"ן) 13

הקורס: "עקרונות מערכות הפעלה"

חומר הלימוד למטלה: ראו פירוט בסעיף "רקע"

משקל המטלה: 12

מספר השאלות: 5

מועד אחרון להגשה: 10.1.2019

סמסטר: 2019א

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

החלק המעשי (80%)

כללי

בחלק המעשי תוכנית לפרמוט (formatting) של מערכת קבצים fat-12 על דיסקט.

מטרה

הכרת מבנה מערכת קבצים fat-12

רקע

א) פרק 13.2 ב [Glibc manual](#) המתייחס לפונקציות lseek, open, close, read, write.

ב) קובץ fat12.pdf.

ג) קובץ fat_paper.pfd (זהו חומר העשרה, אם רוצים לדעת יותר על FAT)

תיאור המשימה

עליכם לכתוב תוכנית:

`my_format<floppy_img_name>`

תוכנית my_format

1) בעקבות הרצה של `my_format` התוכנית תיצור את הקובץ `floppy_img_name` במידה ולא קיים ותייצר על ה `image` מערכת הקבצים `fat12`. במידה והקובץ `floppy_img_name` קיים, התוכנית תשכתב אותו עם מערכת קבצים `fat12` חדשה.

2) `my_format` תחזור עם `status 0` במקרה של הצלחה. במקרה של כישלון של קריאת מערכת כלשהי התוכנית תחזור עם סטטוס 1.

3) במקרה של הצלחה, התוכנית `my_format` תדפיס נתונים על ה `image` של דיסקט כמפורט מטה בתמונת המסך הממחישה את ריצת התוכנית.

הערות

- (1) סיפקנו לכם קובץ fat12.h ו my_format.c שאותם יש להשלים לכדי פתרון.
- (2) קראו בעיון את הקובץ fat12.pdf . fat12.pdf מסביר את מהו layout של מערכת הקבצים fat12. במידת הצורך תוכלו לעיין fat_paper.pdf המכיל פירוט של כל הקבועים, המבנים והאלגוריתמים של fat12.

הגשה

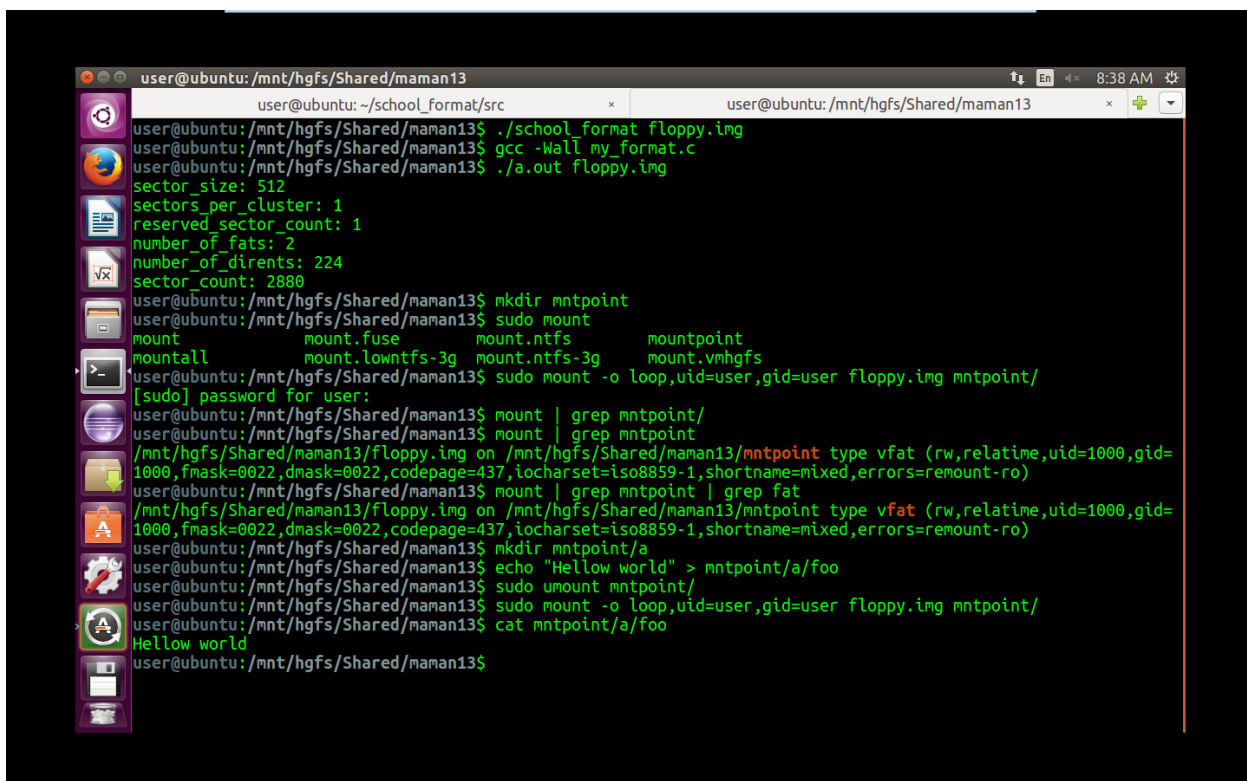
יש להגיש קבצי קוד וקובץ Makefile שמייצר קובץ הרצה בשם my_format. אין להגיש קבצים מקומפלים. את הקבצים המוגשים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). הכנת קובץ ארכיון מתבצעת ע"י הרצת הפקודה הבאה משורת הפקודה של Linux:

```
zip exYZ.zip <ExYZ files>
```

הערה חשובה: בכל קובץ קוד שאתם מגישים יש לכלול כותרת הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.

פתרון ביה"ס

קיבלתם קובץ הרצה school_format. בתמונת צילום מסך מטע תוכלו לעיין ברצף פקודות המפעיל את פתרון בה"ס. לנוחיותכם, תוכלו למצוא את הפקודות הללו בקובץ simpleTest.sh.



```
user@ubuntu: /mnt/hgfs/Shared/maman13
user@ubuntu: ~/school_format/src
user@ubuntu: /mnt/hgfs/Shared/maman13$ ./school_format floppy.img
user@ubuntu: /mnt/hgfs/Shared/maman13$ gcc -Wall my_format.c
user@ubuntu: /mnt/hgfs/Shared/maman13$ ./a.out floppy.img
sector_size: 512
sectors_per_cluster: 1
reserved_sector_count: 1
number_of_fats: 2
number_of_dirents: 224
sector_count: 2880
user@ubuntu: /mnt/hgfs/Shared/maman13$ mkdir mntpoint
user@ubuntu: /mnt/hgfs/Shared/maman13$ sudo mount
mount mount.fuse mount.ntfs mountpoint
mountall mount.lowntfs-3g mount.ntfs-3g mount.vmhgfs
user@ubuntu: /mnt/hgfs/Shared/maman13$ sudo mount -o loop,uid=user,gid=user floppy.img mntpoint/
[sudo] password for user:
user@ubuntu: /mnt/hgfs/Shared/maman13$ mount | grep mntpoint/
user@ubuntu: /mnt/hgfs/Shared/maman13$ mount | grep mntpoint
/mnt/hgfs/Shared/maman13/floppy.img on /mnt/hgfs/Shared/maman13/mntpoint type vfat (rw,relatime,uid=1000,gid=1000,mask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
user@ubuntu: /mnt/hgfs/Shared/maman13$ mount | grep mntpoint | grep fat
/mnt/hgfs/Shared/maman13/floppy.img on /mnt/hgfs/Shared/maman13/mntpoint type vfat (rw,relatime,uid=1000,gid=1000,mask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
user@ubuntu: /mnt/hgfs/Shared/maman13$ mkdir mntpoint/a
user@ubuntu: /mnt/hgfs/Shared/maman13$ echo "Hellow world" > mntpoint/a/foo
user@ubuntu: /mnt/hgfs/Shared/maman13$ sudo umount mntpoint/
user@ubuntu: /mnt/hgfs/Shared/maman13$ sudo mount -o loop,uid=user,gid=user floppy.img mntpoint/
user@ubuntu: /mnt/hgfs/Shared/maman13$ cat mntpoint/a/foo
Hellow world
user@ubuntu: /mnt/hgfs/Shared/maman13$
```

בבדיקה של המטלה ייבדקו פעולות נוספות המופעלות על הדיסקט הפורמט. אם ברצונכם לכתוב תוכניות בדיקה מורכבות, אפשר לתאם עם חברי צוות הקורס את מסוג הבדיקות ולקבל בונוס קטן.

כדי שתוכלו להריץ את הפתרון, יש לוודא שהרשאת x במחרוזת ההרשאות של קבצי הרצה של פתרון בה"ס נמצאות במצב "דלוק". כדי "להדליק" אותה במידה והיא "כבויה" יש להריץ משורת הפקודה של UNIX את הפקודה:

`chmod +x my_format`

החלק העיוני (20%)

שאלה 1 (5%)

לפי מדיניות חדשה של תזמון זרוע הדיסק, הבקשות מוחזקות בתור לפי סדר הגעתן והראשונה שמטופלת היא הבקשה שהגיע אחרונה. מדיניות זו נקראת LIFO (last in first out).

(א) מהו היתרון של המדיניות הזאת?

(ב) מהו החיסרון של המדיניות הזאת?

שאלה 2 (5%)

מערכי דיסקים RAID level 2 ו RAID level 3 מסוגלים להמשיך לעבוד כאשר אחד מהדיסקים במערך מתקלקל. יחד עם זאת, Level 2 דורש מספר רב יותר של דיסקים עודפים. אז מדוע יש בכלל עניין כלשהו בשיטה הזאת?

תזכורת - קוד המינג:

בהנתן מילה בת 4 סיביות:

סיבית b1	סיבית b2	סיבית b3	סיבית b4
----------	----------	----------	----------

קוד המינג שלה הוא:

P1	P2	B1	P3	B2	B3	B4
----	----	----	----	----	----	----

כאשר

$P1 = \text{Even Parity of } b1, b2, b4$

$P2 = \text{Even Parity of } b1, b3, b4$

$P3 = \text{Even Parity of } b2, b3, b4$

לדוגמא: המינג קוד של מילה בת 4 סיביות (משמאל לימין – מ most significant bit ל least significant bit) 1101 יהיה 1100110.

שאלה 3 (5%)

גודלו של קובץ כלשהו יכול להיות בין 4Kb ל 4Mb בכל רגע נתון בחייו. איזו מבין 3 מדיניות הייתם בוחרים:

- הקצאה רציפה

FAT -

I-Node -

הניחו הנחות סבירות נוספות שדרושות. הדגימו את החישובים עליהם תבססו את ההחלטה.

שאלה 4 (5%)

תארו שיטות להגנה על ה capabilities.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או כקובץ pdf. שם הקובץ צריך להיות exYZ.pdf או exYZ.doc (כאשר YZ הנו מספר המטלה).

בדיקה לאחר ההגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי ולבדוק שהקבצים אכן הוגשו באופן תקין ושניתן לקרוא אותם. בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון *exXY.zip* בספרייה חדשה (*new folder*).
- וידוא שכל הקבצים הדרושים נוצרו בספרייה בה פתחתם את הארכיון.
- הרצת *make* ווידוא שכל ה *targets* נוצרו ללא שגיאות וללא *warnings*
- הרצת בדיקות רלוונטיות לוודא תקינות הריצה של החלק המעשי

בונוס

מימוש חלק זה מקנה עד 2 נקודות מגן בבחינה הסופית. למען הסר ספק, ניתן לקבל את המגן רק בתנאי שציון המטלה הוא לפחות 90.

מטרה

עליכם לממש *double indirection* במערכת הקבצים של מערכת הפעלה לימודית *xv6*.

הנחיות

- א. קראו פרק 6 מתוך <https://pdos.csail.mit.edu/6.828/2017/xv6/book-rev10.pdf>
- ב. תורידו את הקוד של *xv6* מ <https://github.com/mit-pdos/xv6-public>
- ג. קראו ופעלו על פי ההנחיות הבאות כדי להפעיל/להשתמש/לנפות שגיאות ב *xv6* :
<https://github.com/rml/xv6-public/wiki/Compile,-Run-and-Play-Around>
- ד. כפי שניתן לראות מ https://github.com/mit-pdos/xv6-public/search?q=NINDIRECT&unscoped_q=NINDIRECT
ה *inode* במערכת הקבצים של *xv6* תומכת ב 12 בלוקים ישירים וקיימת תמיכה ב *single indirection* בלבד. עליכם להבין את הקוד של פונקציית *bmap* במערכת בקבצים ולהרחיב את התמיכה ל *double indirection*. כמו כן, יש לשנות את הפונקציות *main* ו *iappend* של תוכנת עזר *mkfs* המשמשת לפירמוט מערכת בקבצים על מנת לתמוך ב *double indirection*.
- ה. עליכם לכתוב תוכנית המייצרת קובץ בגודל מירבי (התחשב למתיכה שהוספתם), לכתוב בו טקסט, קרוא, ולוודא שנקרא היטב.
- ו. את הקבצים ששיניתם (כולל ה *Makefile*) יש להגיש בקובץ *bonus.tar* דרך מערכת המטלות יחד עם הקבצים של ממ"ן 13 ולכתוב *email* ל davidsa@openu.ac.il על עצם הגשת הבונוס.