Operating Systems – 234123

**Homework Exercise 3 – Dry**

**מגישים:**

**גלעד ניר – 311153746 – giladnir2@gmail.com**

**מיכאל רוזנבאום – 332662501 - michael-roz@campus.technion.ac.il**

Teaching Assistant in charge:

Michael Ezra

Assignment Subjects & Relevant Course material

**Modules, Scheduling, Context Switch**

**Recitations 6-8, Lectures 6-8**

# Submission Format

1. Only **typed** submissions in **PDF** format will be accepted. Scanned handwritten submissions will not be graded.
2. The dry part submission must contain a single PDF file named with your student IDs –

**DHW3\_123456789\_300200100.pdf**

1. The submission should contain the following:
   1. The first page should contain the details about the submitters - Name, ID number and email address.
   2. Your answers to the dry part questions.
2. Submission is done electronically via the course website, in the **HW3 – Dry** submission box.

# Grading

1. **All** question answers must be supplied with a **full explanation**. Most of the weight of your grade sits on your **explanation** and **evident effort**, and not on the absolute correctness of your answer.
2. Remember – your goal is to communicate. Full credit will be given only to correct solutions which are **clearly** described. Convoluted and obtuse descriptions will receive low marks.

# Questions & Answers

* The Q&A for the exercise will take place at a public forum Piazza **only**. Please **DO NOT** send questions to the private email addresses of the TAs.
* Critical updates about the HW will be published in **pinned** notes in the piazza forum. These notes are mandatory and it is your responsibility to be updated.

A number of guidelines to use the forum:

* Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
* Be polite, remember that course staff does this as a service for the students
* You’re not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
* When posting questions regarding **hw3**, put them in the **hw3** folder.

# Late Days

* Please **DO NOT** send postponement requests to the TA responsible for this assignment. Only the **TA in charge** can authorize postponements. In case you need a postponement, please fill out the attached form: <https://goo.gl/forms/eW76r9cRNPTw9vAW2>

# חלק 1 - החלפת הקשר (50 נק')

חלק מהשאלות הן שאלות רב-ברירה (שאלות אמריקאיות) כאשר לכל שאלה יש **בדיוק תשובה אחת נכונה**. בשאלות אלו יש להקיף את התשובה הנכונה ביותר לדעתכם ולנמק. **תשובות לא מנומקות לא יתקבלו**.

תזכורת: בלינוקס יש שני סוגים של החלפות הקשר:

(א) החלפת הקשר יזומה - כאשר התהליך מוותר על המעבד.

(ב) החלפת הקשר כפויה - כאשר מערכת ההפעלה מפקיעה את המעבד מהתהליך.

1. (8 נק') איזה אירוע מבין הבאים יוביל בהכרח להחלפת הקשר כפויה (כלומר הפקעה)?
   1. פסיקת חומרה אשר מעירה תהליך עדיף יותר מהתהליך הנוכחי.
   2. חריגת דף בעקבות גישה לא חוקית של התהליך לזיכרון.
   3. קריאת מערכת ()wait.
   4. קריאת מערכת ()read.
   5. קריאת מערכת ()exit.

**נימוק: התשובה היא a כי זהו ארוע שלא תלוי בקוד המשתמש, וזהו ארוע אסינכרוני. כאשר יתעורר תהליך הוא יקבל את הזמן הוירטואלי המינימלי של כולם, ולכן הוא זה שירוץ על המעבד.**

תזכורת: גרעין לינוקס עד גרסה 2.6 אינו ניתן להפקעה --- non-preemptible kernel.

1. (8 נק') מה משמעות המושג "גרעין שאינו ניתן להפקעה"?

**תשובה: המשמעות היא שכאשר קיימת דרישה להחלפת הקשר כפויה, אז אם כרגע הקוד הוא קוד גרעין שרץ במעבד, לא תהיה החלפת הקשר מיד, אלא רק בסוף ריצת קוד הגרעין ולפני החזרת המעבד לקוד המשתמש.**

לגרעין שאינו ניתן להפקעה יש מספר חסרונות, לדוגמה פגיעה בביצועים של תהליכים מסוימים. פרדי, חובב מוזיקה ומשתמש לינוקס גרסה 2.4, סובל מבעיה כזו כאשר הוא מאזין למוזיקה בזמן שהוא מקמפל קוד של תרגילי בית אבל שומע "קפיצות" במוזיקה. לצורך המשך השאלה, נסביר בקצרה איך עובד נגן מוזיקה:

קובץ אודיו ("שיר") מורכב מהרבה דגימות (samples) שנשלחות להתקן חומרה מיוחד -- כרטיס קול -- אשר ממיר את המידע הדיגיטלי (הדגימות) לאות אנלוגי ומשמיע אותו ברמקולים. כרטיס הקול מכיל חוצץ (buffer) קטן יחסית, למשל בגודל 64KB, ולכן נגן המוזיקה לא יכול להעביר את הקובץ כולו בבת אחת. נגן המוזיקה ממלא את החוצץ (במשך X מילי-שניות) ואז מוותר על המעבד ועובר להמתין בזמן שכרטיס הקול קורא את המידע מהחוצץ (במשך Y מילי-שניות). מעט לפני שהחוצץ מתרוקן, כרטיס הקול שולח פסיקה למעבד, אשר מעירה את נגן המוזיקה כדי שימלא שוב את החוצץ, וחוזר חלילה.

נתון כי:

* החוצץ של כרטיס הקול הוא בגודל 64KB.
* תדירות הדגימות ברצועת השמע היא 44.1KHz.
* כל דגימה מכילה מידע על שני ערוצים (stereo) ברוחב 16 ביט כל אחד.
* זמן הקריאה מהדיסק הוא הדומיננטי ביותר בפעולת נגן המוזיקה.
* רוחב פס של דיסק הוא 10MB/s.

1. (6 נק') מהו משך הזמן X?

**תשובה: 6.25ms**

**נימוק: מהירות הקריאה מהדיסק היא 10 MB/s וצריך למלא 64KB ולכן החישוב הוא:**

**64KB/(10\*2^20)**

(6 נק') מה משך הזמן Y? כלומר, מהו משך הזמן שבו יכולה מערכת ההפעלה להריץ תהליכים אחרים לפני שיהיה עליה לחזור ולמלא את החוצץ כך שהמשתמש לא ישמע "קפיצות"?

**תשובה: 371ms**

**נימוק: בכל דגימה ממירים 2\*16 ביט כלומר 4B ותדירות הדגימות היא 44100 דגימות בשניה ורוצים להמיר סה"כ 64KB ולכן נחשב: 44100\*4\*y = 2^16**

**בשאלות הבאות הניחו כי:**

* במחשב של פרדי רצים רק שני תהליכים: נגן המוזיקה + קומפילציה של קוד.
* הגרעין אינו ניתן להפקעה.
* תהליך הקומפיילר משתמש הרבה בשירותי מערכת ההפעלה.
* במחשב של פרדי יש מעבד יחיד.

1. )6 נק') מה אופיו של תהליך נגן המוזיקה, חישובי או אינטראקטיבי? מה לגבי תהליך הקומפיילר האם הוא חישובי או אינטראקטיבי?

נתון כי:

* שני התהליכים הם בעלי ערך (ערך ברירת המחדל).
* תהליך הקומפילציה רץ הרבה על המעבד ויוצא מעט להמתנה.

**תשובה:**

**נגן המוזיקה הוא אינטרקטיבי כי התהליך הזה מעוניין בזמן המתנה נמוך (נגן המוזיקה צריך להתעורר כל 371 מילי שניות ולרוץ 6 מילי-שניות בלבד על המעבד. כלומר, התהליך מוותר על המעבד רוב הזמן ורק מחכה כאשר הוא רוצה להמתין הכי מעט זמן שאפשר מרגע קבלת הנתונים, ולכן הוא תהליך אינטראקטיבי)**

**הקומפיילור מעוניין בזמן תגובה (הזמן שבו יסתיים התהליך) נמוך אבל לא חשוב לו זמן המתנה נמוך ולכן הוא תהליך חישובי.**

1. (8 נק') מדוע נגן המוזיקה סובל מגרעין שאינו ניתן להפקעה?

**כי בגרסאות הישנות של לינוקס רצו להשאיר את הגרעין ללא הפקעה כדי שלא יאלצו להכניס אמצעי סינכרון (שפוגעים בביצועים) בגישה למבני נתונים בגרעין (מחשש שיבוא תהליך אחר ויפקיע את המעבד מהתהליך לפני שהכל שמור- בדיוק כמו הבעיות הקלאסיות של אי סינכרון).**

**כעת מה שקורה לנגן זה שבשלב מסוים למרות שהוא צריך להעביר עוד נתונים לכרטיס הקול, הוא לא מבצע זאת מיד אלא ממתין לקוד הגרעין של התהליך השני, ולכן הנתונים לא יגיעו מספיק מהר לכרטיס הקול ואז השומע מקבל את המוזיקה לא ברצף.**

החברים של פרדי ניסו לעזור לו לפתור את הבעיה.

בריאן הציע לשפר את העדיפות של נגן המוזיקה ע"י הקטנת הערך nice.

רוג'ר הציע להפוך את נגן המוזיקה לתהליך זמן-אמת במדיניות SCHED\_RR.

ג'ון הציע להפוך את נגן המוזיקה לתהליך זמן-אמת במדיניות SCHED\_FIFO.

1. (8 נק') מי מהחברים של פרדי הציע דרך שתצמצם את הבעיה?
   1. אף אחד מבין החברים.
   2. רק בריאן.
   3. רק רוג'ר.
   4. רק ג'ון.
   5. רק רוג'ר וג'ון.
   6. כל שלושת החברים.

**תשובה: אף אחד מבין החברים. כל שלושת החברים מציעים עדיפות לנגן על פני הקומפיילר (ההבדל ביניהם הוא אם העדיפות הזו מוחלטת או לא), אך כל זה לא עוזר כי ממילא כל פעם אחרי 6.25 מילי שניות (זמן הכתיבה לbuffer) התהליך שוב ירצה לכתוב אבל הוא יחסם, ולכן נצטרך לעבור לתהליך השני, ואין הבדל לעניין זה בין המימוש המקורי להצעות החדשות האלו, שכאשר תגיע פסיקה לכך שהסתיים התרגום בכרטיס הקול, ממילא נעבור לנגן אם ורק אם התהליך אינו בקוד הגרעין.**

# חלק 2 - זימון תהליכים (50 נק')

בתרגיל זה ננסה לבחון לעומק את אלגוריתם התזמון של לינוקס לתהליכים רגילים שלמדתם בתרגול - CFS .

1. (7 נק') נתונה מערכת בעלת n תהליכים עם משקלים בהתאמה.   
   נניח כי בתחילת ה-Epoch לכל התהליכים אותו זמן וירטואלי ((.

תחת ההנחה שכל תהליך מנצל את כל הקוונטום שלו, מהו ההפרש הגדול ביותר האפשרי בין הזמן הווירטואלי המקסימלי והזמן הווירטואלי המינימלי של התהליכים בסוף ה-Epoch?

**פתרון**:

1. (7 נק') כזכור, בתרגול ראיתם נוסחה לקצב התקדמות הזמן הווירטואלי ונוסחה לחישוב הקוונטום של כל תהליך. ננסה לבחון את הקשר בין 2 הנוסחאות:  
     
   אלגוריתם CFS שואף שהזמן הווירטואלי בין כל התהליכים יהיה זהה, ובפרט שבסוף כל Epoch לכל התהליכים יהיה את אותו זמן וירטואלי.   
     
   הראו, שאם הזמן הווירטואלי מתקדם ביחס הפוך לעדיפות התהליך:   
   (עבור כלשהו), וגודל ה-Epoch הוא קבוע , **אזי** הדרישה שהזמן הוירטואלי של כל התהליכים בסוף ה-Epoch זהה, גוררת את הנוסחה שראינו באלגוריתם CFS לחישוב הקוונטום של כל תהליך.

**לכל i, j מתקיים: ולכן  *=> => .***

**כעת,  *.***

***לכן ומכאן מש"ל.***

1. (24 נק') אמיר, מומחה עולמי לאלגוריתמי תזמון החליט לנסות לשפר את אלגוריתם CFS המוכר. אמיר, הציע שיפורים שונים לאלגוריתם. בכל אחד מהסעיפים הבאים, מוצע שינוי/שיפור לאלגוריתם CFS. כתוב חיסרון שנקבל כתוצאה מאותו שינוי.  
   1. (4 נק') תהליך שחזר מהמתנה, ישאר עם אותו זמן וירטואלי שהיה לו כשהוא יצא להמתנה.

**כל שאר התהליכים יתקדמו בזמן וירטואלי כשהתהליך הנ״ל בהמתנה. כשהוא יחזור מהמתנה, יהיה לו בפער הזמן הוירטואלי הכי נמוך בין כל התהליכים ולכן ירוץ הכי הרבה בזמן הקרוב, ולכן חלוקת הזמן לא תהיה הוגנת ולא תהיה תואמת לתעדוף שהמשתמש קבע.**

* 1. (4 נק') שימוש במבנה נתונים של רשימה במקום עץ אדום-שחור.

**הסיבוכיות של פעולות הכנסה והוצאה תהיה מאשר עבור עץ מאוזן.**

* 1. (4 נק') הסרת המינימום על גודל הקוונטוםשל תהליך (min\_granularity)

**זה יגרום לתקורה גבוהה כי יהיה יותר מדי החלפת הקשר ואז הרבה מהזמן של ריצת המעבד יתבזבז בהחלפות הקשר.**

* 1. (4 נק') הגדרת הקוונטום של כל תהליך, להיות בלתי תלוי במספר התהליכים, ובפרט (עבור קבוע חיובי כלשהו)

**אם יהיו הרבה תהליכים אז כל אפוק יהיה גדול יותר מהרצוי ונקבל אינטרקטיביות לא טובה, ואם מעט תהליכים, אז האפוק יהיה קטן וחבל על כל החלפות ההקשר.**

* 1. (4 נק') שינוי קצב התקדמות הזמן הוירטואלי להיות מהצורה

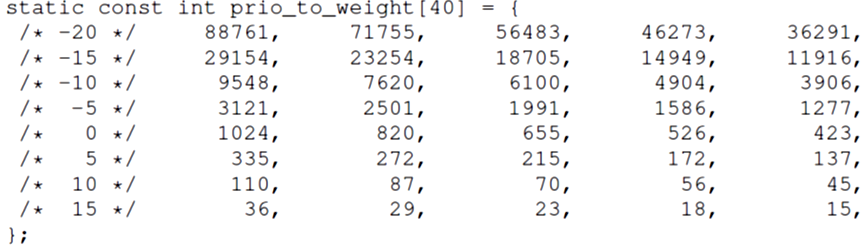
**צריך יהיה להגדיל מאוד (בריבוע) את המשקלים של התהליכים עם העדיפות הגבוהה כדי לקבל את אותו אפקט על ארגון הזמן הוירטואלי, וגם ככה ב CSF ההבדלים בין המשקלים הם גבוהים. בצורה העכשיוית, אפשר עם מספרים קטנים יותר מאשר בהצעה לתת משקל חזק יותר לבעלי העדיפות הגבוהה, כלומר ספקטרום אפקט הוירטואליזציה גדול יותר בצורה העכשיוית.**

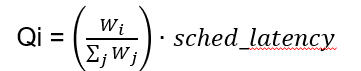
* 1. (4 נק') תהליכים חדשים מגיעים למערכת עם זמן וירטואלי השווה לזמן הווירטואלי הממוצע של כל התהליכים במערכת.

**לא נשמור על ההוגנות מבחינת סדר ההגעה ויתכן מצב שבו יוצרו הרבה תהליכים חדשים כדי לגנוב זמן ריצה.**

1. (6 נק') על מנת להפחית את התקורה בעת החלפת הקשר כתוצאה מהחלפת אזורי הזיכרון וניקוי ה-TLB, אמיר (אותו אחד משאלה קודמת - הוא אוהב להציע הצעות), הציע שתהליכים שמוכנים לריצה, וחולקים את אותו מרחב זיכרון עם התהליך שרץ אחרון על המעבד, יקבלו עדיפות על המעבד וירוצו לפני כל התהליכים האחרים. אילן, סטודנט למערכות הפעלה, טוען שהפתרון לא מדהים במיוחד, ושבצורה כזאת תהליך יכול לגנוב זמן מעבד. **הסבירו** כיצד.

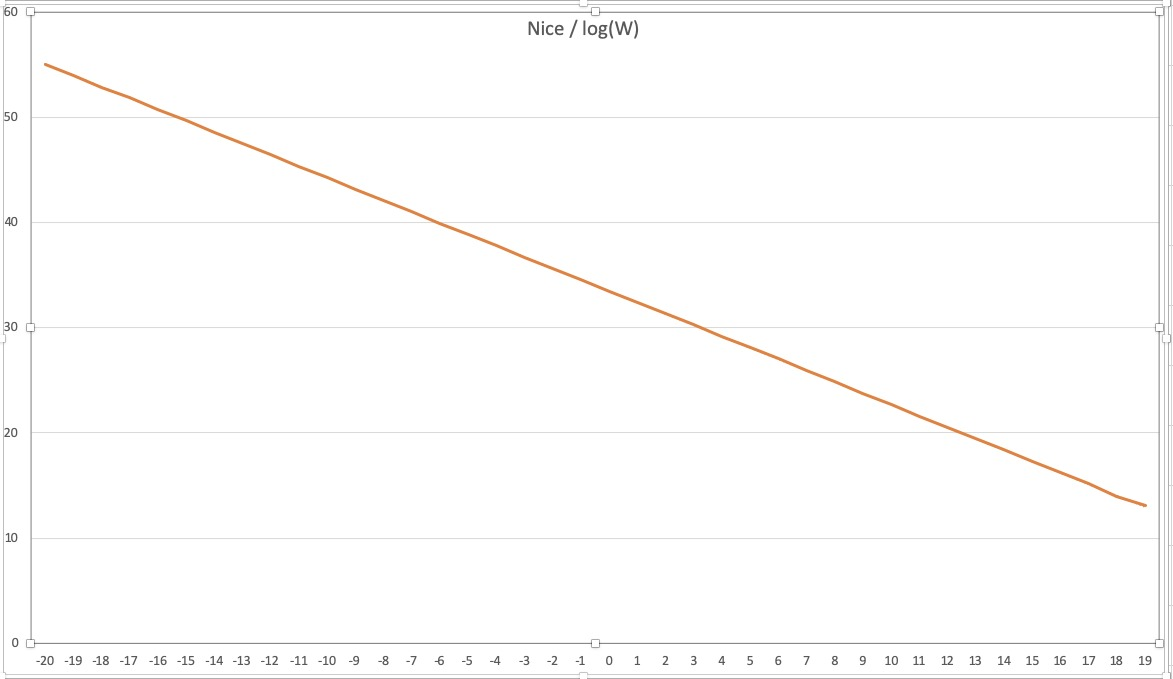
**תהליך שרץ יכול לגנוב זמן מעבד ע״י יצירת חוטים שחולקים איתו את אותו מרחב זיכרון והמתנה להם. הם יקבלו עוד קוואנטום משל עצמם, וכך בעצם התהליך "גנב" עוד קואנטום.**

1. (6 נק') נזכיר, שבאלגוריתם CFS, לכל תהליך יש משקל שנקבע לו בהתאם לערך ה-nice שלו ():  
     
     
   כמו כן, ראינו כי גודל הקוונטום של תהליך, נמצא ביחס ישר למשקל התהליך:



לכן, לתהליך בעל ערך nice **נמוך**, יש משקל **גבוה**, וכתוצאה מכך קוונטום **ארוך** יותר. בסעיף זה, נרצה לבחון את הרציונל מאחורי בחירת ערכי המשקלים.

* 1. (3 נק') ציירו (בעזרת אקסל או כל תוכנה אהובה לבחירתך) גרף של לוג משקל התהליך (בבסיס 1.23), כפונקציה של ערך הnice המתאים לה. **צרף תמונה של הגרף לפתרון.**



* 1. (3 נק') על סמך הגרף, מהו הקשר המתמטי בין ערך הnice למשקל התהליך?