Homework 2 Dry

Due date: 08/05/2015 12:30

Teaching assistant in charge:

* Assaf Rosenbaum

**Important:** the Q&A for the exercise will take place at a public forum [Piazza](https://piazza.com/class/i1q3ilmq2gs7kp) only. Please note the forum is a part of the exercise, clarifications/corrections that will be published 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 hw2, put them in the hw2 folder

Please submit your answers printed – not printed works will not be checked.

בכל השאלות לכל הסעיפים משקל זהה אלא אם כן צוין אחרת.

**שאלה 1 (40 נקודות)**

בשאלה זו נדון באלגוריתם הזימון של לינוקס במערכת מרובת מעבדים ובפרט באלגוריתם המשמש את לינוקס לאיזון העומס בין המעבדים, לצורך פתרון השאלה עליכם להיעזר בספר Understanding The Linux Kernel 3rd Edition, פרק 7.

1. האם יש הבדל בין ביצוע השגרה schedule במערכת עם מעבד יחיד לבין ביצועה במערכת מרובת מעבדים? אם יש הבדל הסבר מהו ואם לא הסבר מדוע.  
   **תשובה:**

(תשובה ב 7.5. Runqueue Balancing in Multiprocessor Systems )

ביצוע השגרה schedule נועד למצוא את התהליך העדיף ביותר לריצה מבין התליכים המוכנים לריצה ב runqueue של מעבד מסויים, כאשר לכל מעבד קיים runqueue משלו. על כן במערכת בעלת מספר מעבדים ובמערכת בעלת מעבד יחיד לא יהיה הבדל בשגרת schedule. נרחיב ונאמר כי כאשר קיימים המון תהליכים על runqueue אחד של מעבד מסויים אזי במכונה בעלת מספר מעבדים – מעבד אחד יכול לעבוד , כאשר כל המעבדים האחרים נשארים ללא הרצת תהליכים – על ידי כך שכל התהליכים נמצאים בrunqueue של מעבד יחיד. לכן במקרה כזה קיים אלגוריתם load balancing algorithm שמטרתו לאזן את כמות התליכים בין כל ה runqueues של המעבדים במערכת.

1. כפי שראינו בתרגולים, לכל מעבד ישנו runqueue משלו והתהליך הבא שיבחר לרוץ על המעבד נבחר מתוך אותו runqueue. פרט יתרון אחד וחסרון אחד של שיטה זו.  
   **תשובה:**

יתרון של השיטה – התהליך הבא לריצה יחושב באופן מהיר משום שאין צורך לחפש תהליך בכל ה runqueues במערכת. לא בטוח בזה

חסרון של השיטה – כאשר יש מספר רב של תהליכים מוכנים לריצה ב runqueue מסויים כאשר ה runqueues האחרים ריקים מתהליכים , במקרה זה אנחנו מנצלים את היכולת של מעבד יחיד, מה שגורם להאטת המערכת – כאשר תהליכים רבים מחכים בתור לקבל זמן מעבד.

(תשובות ל- ג – ה נמצאות 7.5. Runqueue Balancing in Multiprocessor Systems )

1. הסבר מהו scheduling domain וכיצד הוא משמש באלגוריתם איזון המשימות.  
   **תשובה:**
2. הסבר כיצד מחליט האלגוריתם האם יש צורך בהפעלת איזון משימות על המעבד.  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. הסבר כיצד מאזן האלגוריתם את המשימות בין המעבדים, האם ישנו איזון מלא או הקלה בעומס בלבד?  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**שאלה 2 (20 נקודות)**

קבע האם הטענה נכונה או לא ונמק, תשובות ללא נימוק לא יתקבלו.

1. בעקבות שימוש מרובה בפקודה wait תהליך יכול להפוך לאינטראקטיבי.  
   **תשובה:**

יש מצב שהטענה נכונה כי אולי ב wait ה sleep avr עולה מה גורם לבונוסים (אבל צריך לבדוק לעומק)

1. בעקבות שימוש מרובה בפקודה sched\_yield תהליך יכול להפוך לאינטראקטיבי.  
   תשובה:

יכולת להיות שלא כי ה sleep avr לא נספר במקרה זה . גם צריך לבדוק לעומק

**שאלה 3 (20 נקודות)**

1. הסבר מדוע לא ניתן להשתמש בקריאה ישירה ל - schedule בפונקציה scheduler\_tick ויש צורך להדליק את הדגל need\_resched כאשר תהליך מסיים את ה - time\_slice שלו.
2. הסבר מדוע במאקרו switch\_to קופצים לפונקציה \_\_switch\_to בעזרת הפקודה jmp ולא בעזרת הפקודה call  
   **תשובה:**

בתחילת המאקרו ביצענו שמירה של רגיסטרים רלוונטיים (לתהליך המחולף). לאחר ביצוע הפונ' \_\_switch\_to אנחנו נצטרך לחזור ולשחזר את אלה של התהליך החדש (כפי שנשמרו בפעם הקודמת שהוא הוחלף). אבל זה רלוונטי רק להחלפות הקשר להתהליכים שכבר רצו. אם ההחלפה התבצעה לתהליך שנוצר עכשיו, הוא צריך להמשיך לרוץ לאחר החלפת ההקשר ממקום אחר לחלוטין (ret\_from\_fork).לכן, הקפיצה לפונקציה תתבצע בעזרת jmp – כלומר ללא ערך חזרה. אם מדובר בתהליך חדש, נמשיך כרגיל, אבל אם מדובר במקרה של שני תהליכים "ותיקים" נחזור ללייבל ששמרנו מראש על ראש המחסנית (הלייבל 1), שמסמל כתובת לשחזור אותם רגיסטרים.

**שאלה 4 (20 נקודות)**

אלגוריתם lottery scheduling הינו אלגוריתם זימון הסתברותי. כל תהליך במערכת מקבל מספר כלשהו של "כרטיסי הגרלה" (לכל כרטיס מספר יחודי); בכל שלב, האלגוריתם מגריל מספר של אחד מכרטיסי ההגרלה שחולקו, והתהליך שמחזיק בכרטיס הגרלה זה מקבל את המעבד לפרק זמן קצוב (תמיד אותו פרק זמן).

1. נניח כי לאלגוריתם יש 20 כרטיסים. כיצד עליו לחלק אותם בין ארבעה תהליכים (A, B, C, D) כך שהתהליכים יקבלו 10%(A), 5%(B), 60%(C), 25%(D) בהתאמה?  
   **תשובה:**

A – 2 כרטיסים, B – כרטיס אחד, C – 12 כרטיסים, D – 5 כרטיסים

1. כיצד ניתן להגיע באמצעות אלגוריתם זה לביצועים הדומים לאלו המושגים על-ידי אלגוריתם round-robin?

**תשובה:**

באלגוריתם RR התהליכים מסתדרים במעגל – כאשר תהליך אחד רץ ומסיים את הקוואנטום שלו הוא

מושהה והתהליך הבא בתור מורץ. כאשר מגיעים לתהליך שממנו התחלנו מסיימים epoch.

על כן על מנת להגיע לביצועים דומים יש לחלק לכל תהליך מספר שווה של כרטיסי הגרלה, באופן הזה

כל תהליך יקבל זמן מעבד שווה – לכן בסה"כ ניתן לדמות את האופן ריצתם של התלהיכים באלגוריתם זה לאלגוריתם RR.

1. איך ניתן להגיע באמצעות אלגוריתם זה לביצועים הדומים לאלו המושגים על-ידי אלגוריתם

Shortest remaining time to completion first?   
(ניתן להניח כי יש preemption, התהליכים לא מגיעים יחד, אבל זמן הביצוע ידוע כאשר התהליך מגיע.)  
**תשובה:**

לפי אלגוריתם Shortest remaining time to completion first , תהליך שנותר לו זמן הכי קצר לביצוע המשימה שלו , יקבל את המעבד ראשון. לפי הנתון זמן הביצוע של כל תהליך ניתן כאשר התהליך מגיע

על כן יש לבצע חישוב של ניתנת מספר כרטיסי הגרלה באופן דינמי לפי זמן הביצוע. צריך להמשיך את התשובה

1. איזה יתרון יש לשיטה הזו על פני shortest remaining time to completion first אמיתי?  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**בהצלחה!**