Homework 2 Dry

**Due Date: 11/12/2018 23:30**

Teaching assistant in charge:

* Andre Kassis

**Important:** the Q&A for the exercise will take place at a public forum Piazza only. 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 hw2, put them in the hw2 folder

Only the TA in charge can authorize postponements. In case you need a postponement, please fill out the following form: https://goo.gl/forms/D5nxMxf9Uvgej1SL2

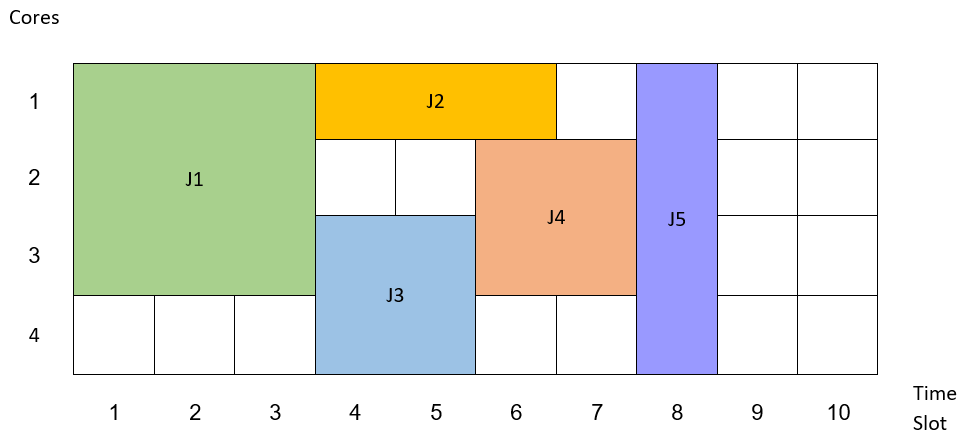
Dry part submission instructions:

1. Please submit the dry part to the electronic submission of the dry part on the course website.
2. The dry part submission must contain a single dry.pdf file containing 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.
3. Only typed submissions will be accepted. Scanned handwritten submissions will not be accepted. Only PDF format will be accepted.
4. You do not need to submit anything in the course cell.
5. When you submit, **retain your confirmation code and a copy of the PDF**, in case of technical failure. It is **the only valid proof** of your submission.

**יש לנמק כל תשובה, תשובות ללא נימוק לא יתקבלו.**

# שאלה 1 - זימון תהליכים (50 נק')

שאלה זו עוסקת בנושא batch-scheduling כפי שנלמד בהרצאות. שימו לב שבכל טבלאות התזמון הבאות המספר המייצג time-slot כלשהו מייצג את הזמן שבו החלון הנתון **מסתיים** (כלומר - אם time slot = 3 אזי החלון **מתחיל** ב-t=2 ומסתיים ב- t=3).



1. (18 נק') בהינתן התזמון הנתון למעלה, חשבו את המדדים הבאים (הראו את דרך החישוב). הניחו שכל התהליכים בנתונים למעלה הגיעו באותו הזמן (t=0):
   1. (6 נק') מדד זמן ההמתנה הממוצע (average wait-time):
   2. (6 נק') מדד זמן התגובה הממוצע (average response-time):
   3. (6 נק') מדד הניצולת (utilization):
2. (8 נק') בהנחה שכל התהליכים למעלה מגיעים בזמן 0. איזו מדיניות זימון יכולה לשפר את הניצולת כפי שחושבה בסעיף הקודם? **נמקו** (ראו מצורפת טבלת זימון ריקה על מנת להמחיש את הזימון החדש לפי האלגוריתם, **עליכם למלא את הטבלה בהתאם**)  
   ע"י שימוש באלגוריתם EASY אשר משלב גם את אלגוריתם FCFS (first come first serve) וגם BACKFILLING נקבל שניתן למלא את הריבועים של המעבדים החל מזמן 0 עד זמן 6 באופן מלא (ראה ציור למטה).  
   לאחר שימוש באלגוריתם הנ"ל נקבל כי המעבד עובד רק עד 7 ובו הניצולת שלו היא 100%



**J1**

**J2**

**J3**

**J4**

**J5**

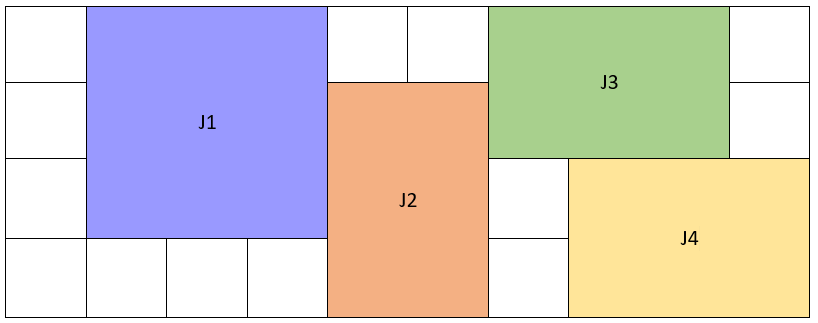
1. (8 נק') מנו 2 יתרונות ו-2 חסרונות של מדיניות זימון מסוג batch-scheduling על פני מדיניות Round-robin. נמקו.  
   **יתרונות:**
   1. בהנחה שמדובר במערכת בעלת מעבד יחיד וכל התהליכים מגיעים ביחד אזי קיים אלגוריתם מסוג batch-scheduling שזמן ההמתנה הממוצע שלו קטן או שווה לזמן ההמתנה הממוצע של אלגוריתם Round-robin ולפיכך זמן התגובה הממוצע של תהליך במדיניות זו הוא קצר יותר או שווה לזמן התגובה במדיניות Round-robin.
   2. Round-robin דורש פסיקות שעון (כל מספר מילי-שניות) ותוכנית יעודית אשר תפקידה לנהל את התהליכים. בנוסף לכך , ב-Round-robin תהליך מופרע באמצע הריצה לעומת זאת ב-batch-scheduling התהליך רץ עד סופו ללא הפרעה

**חסרונות:**

1. יוצר הרעבה (STARVATION) של תהליכים אחרים כיוון שנותן לתהליך לרוץ עד סיומו
2. ב-batch-scheduling אין התייחסות לעדיפות התהליך ולכן יתכן מצב שיתבצע הרעבה של תהליך בעל עדיפות גבוהה.

נתונה מדיניות זימון חדשה (BSAF (Biggest Surface-Area First לפיה בהינתן 2 תהליכים זה שנבחר לרוץ קודם הוא זה **שהשטח שלו הגדול יותר** (שטח = זמן \* מספר-מעבדים). שימו לב שמדיניות זימון זו תומכת ב-backfilling - כלומר שאם התהליך העדיף ביותר לפי BSAF לא יכול להיות מתוזמן בחלון כלשהו, יתוזמן התהליך הטוב ביותר שמתאים לחלון אחריו (זה שמתאים לחלון הפנוי והוא הטוב ביותר לפי BSAF). בהינתן 2 תהליכים בעלי אותו שטח נבחר בזה שיש לו את זמן הריצה הקצר ביותר. בהינתן 2 תהליכים בעלי שטח זהה וזמן ריצה זהה, נבחר בזה עם ה-ID (מספר) הנמוך יותר.

למשל, בהינתן 4 התהליכים הבאים (האיור להמחשה של מימדי התהליכים בלבד):



האלגוריתם BSAF יתעדף אותם בסדר הבא:

J1 (3\*3=9 surface area)

J2 (3\*2=6, shorter runtime than J3 and J4)

J3 (2\*3=6, same as J4, but lower ID)

J4

1. (8 נק') האם BSAF סובל מ-convoy effect? נמקו (עליכם להמחיש בעזרת הטבלה הנתונה).  
   כן , האלגוריתם מתעדף תהליכים עם שטח גדול יותר ולאחר מכן מתעדף אותם לפי זמן ריצה ומספר סידורי.  
   ולכן בהינתן תהליך בעל זמן ריצה ארוך מאוד (בעל שטח מקסימלי מבין כל שאר התהליכים) אשר משתמש במספר מעבדים כך שלא ניתן יהיה לקדם תהליך אחר באמצעות BACKFILLING למעבדים שבהם התהליך הארוף לא משתמש נקבל את אפקט השיירה.



AREA = 4

AREA = 6

1. (8 נק') האם אלגוריתם הזימון BSAF הינו אופטימלי מבחינת מדד הניצולת? הוכיחו או הפריכו (עליכם להמחיש בעזרת הטבלה הנתונה)



# שאלה 2 - החלפת הקשר (20 נקודות)

.1נתון הקטע הבא מתוך קוד הגרעין להחלפת הקשר:

**01.** movl prev, %eax

**02.** movl next, %edx

**03.** pushl %esi

**04.** pushl %edi

**05.** pushl %ebp

**06.** movl %esp, prev->**thread**.esp

**07.** movl next->**thread**.esp, %esp

**08.** movl $**1f**, prev->**thread**.eip

**09.** pushl next->**thread**.eip

**10.** jmp \_\_switch\_to

**11.** **1**:

**12.** popl %ebp

**13.** popl %edi

**14.** popl %esi

א. מה יקרה אם נחליף את שורות 10-8 בשורה **call \_\_switch\_to** ?

הבעיה בפקודת call היא שהיא דוחפת למחסנית את כתובת החזרה של השורה הבאה לביצוע (תווית '1') תמיד, אך אנחנו רוצים לדחוף כתובת חזרה שונה בהתאם למקרה בו אנו נמצאים:

**לראשונה לאחר fork**: כתובת החזרה שנרצה לדחוף היא ret\_from\_fork.

**ובשאר המקרים (המקרה הסטנדרטי)**: כתובת החזרה היא אכן התווית '1'.

ב. נניח כי ביצענו קריאה למאקרו current לפני שורה 1 ,לאחר איזה שורה קריאה נוספת למאקרו תחזיר תשובה שונה?

שורה מספר 10.

ג. מתרגל בקורס רצה לשפר את זמן הביצוע של החלפת ההקשר. מכיוון ששם לב כי המאקרו **switch\_to** אינו עושה כלל שימוש ברגיסטר ecx ומאחר וידוע כי פעולות על רגיסטרים מהירות משמעותית מפקודות המערבות את הזיכרון, הציע להחליף את שורה **05** בשורה **mov %ebp, %ecx** ואת שורה 12 בשורה **mov %ecx, %ebp** האם הקוד יעבוד כראוי לאחר השינוי המוצע?

לא, מכיוון שערך הרגיסטר ecx משותף עבור שני התהליכים next ו prev בתוך הפונקציה switch\_to ולכן השחזור של ebp לאחר ביצוע context\_switch יהיה לערך של ebp של התהליך הקודם ולא החדש.

2. כפי שראינו בתרגול מצביע לבסיס מחסנית הגרעין של התהליך הנוכחי נשמר גם ב **thread->esp0** וגם ב – **tss->esp0**האם ניתן לוותר על אחד מהם? אם לא הסבר מדוע ואם כן פרט על מי וכיצד נשיג את המידע הדרוש.

בלתי אפשרי לוותר עליהם.  
tss->esp0 מקנה לנו מצביע לבסיס מחסנית הגרעין של התהליך הנוכחי אשר לו אנו זקוקים כאשר אנו עוברים מuser\_mode לkernel\_mode.  
thread->esp0 שומר מצביע למחסנית הגרעין של התהליך המוצבע ע"י thread על מנת לעדכן בעת ביצוע context\_switch את המצביע למחסנית של התהליך החדש.

# שאלה 3 – זימון תהליכים (30 נקודות)

שאלה זו עוסקת במדיניות זימון התהליכים של לינוקס כפי שנלמדה בתרגולים.

לנוחיותכם מצורף חלק מהקוד כפי שנלמד בתרגולים:

#define MAX\_PRIO 140

#define MIN\_TIMESLICE (10 \* HZ / 1000) /\* 10 msec \*/

#define MAX\_TIMESLICE (300 \* HZ / 1000) /\* 300 msec \*/

#define TASK\_TIMESLICE(p) \

MIN\_TIMESLICE + (MAX\_TIMESLICE – MIN\_TIMESLICE) \*

(MAX\_PRIO – **1** – (p)->static\_prio)/**39**

#define TASK\_INTERACTIVE(p) \

((p)->prio <= (p)->static\_prio – DELTA(p))

prio = static\_prio – bonus

**if** (prio < MAX\_RT\_PRIO)

prio = MAX\_RT\_PRIO;

**if** (prio > MAX\_PRIO - **1**)

prio = MAX\_PRIO – **1**;

BONUS(p) = **10**\*(SleepAvg/MaxSleepAvg-**1**/**2**)

DELTA(p) = **5**\*TaskNice(p)/**20**+**2**

1. (5 נק') האם ייתכן מצב בו תהליך אינטראקטיבי **A** יהיה באותה העדיפות הדינאמית כמו תהליך חישובי ? **B** אם כן תארו מצב כזה (תארו 2 תהליכים, את עדיפותם הסטטית, ואת עדיפותם הדינאמית - והוכיחו מספרית​ בעזרת חישובי **bonus** ו-**DELTA** שהתהליכים אכן מקיימים את הדרישות), אם לא נמקו מדוע.
2. (2 נק') בתרגול מצוינות **4** סיבות בגינן עשוי תהליך להגיע לפונקציה .schedule בחנו את הקוד של**sys\_sched\_yield** ו- **interruptible\_sleep\_on**. בכל אחת מהפונקציות הנ"ל ישנה קריאה ישירה לפונקציה **schedule** . אם היינו משנים קריאה זו ל - **set\_tsk\_need\_resched(current)** כיצד זה היה משפיע אם השינוי היה מתבצע רק ב- **sys\_sched\_yield**? כיצד זה היה משפיע אם השינוי היה מתבצע רק ב- interruptible\_sleep\_on ? נמקו (בתשובתכם התייחסו לנכונות הביצוע - כלומר האם התהליך אכן מוצא מהקשר? האם הקרנל ימשיך לעבוד בצורה תקינה? וכו')
3. לצורך שאלה זו נזכיר כיצד מחושב **sleep\_avg** של תהליך בגרעין:

#define MAX\_SLEEP\_AVG (2\*HZ)

sleep\_time = jiffies – p->sleep\_timestamp;

p->sleep\_avg += sleep\_time;

**if** (p->sleep\_avg > MAX\_SLEEP\_AVG)

p->sleep\_avg = MAX\_SLEEP\_AVG;

#define EXPIRED\_STARVING(rq) \

((rq)->expired\_timestamp && \

(jiffies – (rq)->expired\_timestamp >= \

STARVATION\_LIMIT \* ((rq)->nr\_running + **1**))

1. נניח שקיים תהליך אינטראקטיבי **A** בעדיפות סטטית 100 ובעל **sleep\_avg=MAX\_SLEEP\_AVG**.

בנקודת זמן t=0התהליך החל לבצע משימה חישובית ארוכה. מהו הזמן המקסימלי (במילישניות) שהתהליך ירוץ לפני שיעבו ר ל **expired** בהנחה שהוא התהליך היחיד ב?runqueue

2. כיצד תשובתכם לסעיף הקודם הייתה משתנה אם נתון שקיים תהליך **B** ב **expired** והחל מהרגע t=0 שתואר קודם, נותרו 1000 מילישניות עד כילוי ה ? **timestamp\_expired**כלומר עד שהמאקרו **EXPIRED\_STARVING** מתחיל להחזיר **TRUE**).

1. מהו פרק הזמן המינימלי שיכול לעבור מהרגע שהודלק הדגל **need\_resched** בתהליך שרץ ועד שהוא מגלה את הצורך בהחלפת הקשר (במערכת מרובת מעבדים)?
2. האם אפשר לתת דוגמה בה תהליך יכול להתחיל לרוץ קצת אחרי פסיקת שעון ולעזוב את ה CPU קצת לפני פסיקת השעון הבאה (פחות מ ?(tickאם הדבר בלתי אפשרי הסבירו למה, אחרת תנו דוגמה מפורטת.