Operating Systems – 234123

Homework Exercise 2 – Dry

Student	id	email
Amal hihi	213519333	Hihi.amal@campus.technion.ac.il
Sarah hamou	329618169	sarah@campus.technion.ac.il

Teaching Assistant in charge:

Niv Kaminer

Assignment Subjects & Relevant Course material

Modules, Scheduling (Lectures 4--5, Tutorials 4--5)

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 –

   pdf.300200100_123456789
3. The submission should contain the following:
   a. The first page should contain the details about the submitters - Name, ID number and email address.
   b. Your answers to the dry part questions.
4. Submission is done electronically via the course website, in the **HW2 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 **hw2-dry**, put them in the **hw2-dry** 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://forms.office.com/r/28uesBSL1P

חלק 1 - שאלות בנושא התרגיל הרטוב (50 נק')

מומלץ לקרוא את הסעיפים בחלק זה לפני העבודה על התרגיל הרטוב, ולענות עליהם בהדרגה תוך כדי פתרון התרגיל הרטוב.

1. (6 נק') מה עושה פקודת yes בלינוקס? מה הארגומנטים שהיא מקבלת?
היעזרו ב-man page, ולאחר מכן השתמשו בפקודה ב-shell שלכן כדי לבדוק.

- פתרון: *הפקודה Linux yes מאפשרת לך להפוך תגובות לאוטומטיות לסקריפטים ולפקודות,*
- *.אם הפקודה yes מקבל ארגומנט "string" אז היא מדפיסה את המחרוזת אינסוף פעמים עד אשר התהליך מופסק או נעצר. אם היא לא מקבלת ארגומנטים, היא מדפיסה את התו y אינסוף פעמים עד אשר התהליך מופסק או נעצר.*

2. (6 נק') מדוע השתמשנו בפקודת yes עם מחרוזת ריקה במהלך הפקודה הבאה?

```
yes '' | make oldconfig <<
```

נסו להריץ את הפקודה make oldconfig לבדה והסבירו מה הבעיה בכך.

- פתרון : *.הפקודה make oldconfig מצפה לקבל ארגומנטים וע"י השימוש בפקודה yes אנו*
- *שולחים לפקודה זו דרך הpipe קלט של המחרוזת הריקה לארגומנטים שהפקודה*
- *מצפה לקבל.*
- *בהרצת הפקודה לבדה, אנו נקבל אזהרה כי הפקודה מצפה לקבל ארגומנטים שכן אין*
- *לה ערכים דיפולטיביים שהיא מקבלת.*

3. (6 נק') מה משמעות הפרמטר GRUB_TIMEOUT בקובץ ההגדרות של GRUB?

```
GRUB_TIMEOUT=5
```

הסבירו מה היתרונות ומה החסרונות בהגדלת הפרמטר GRUB_TIMEOUT.

פתרון : If this variable is set, it specifies the time in seconds to wait for keyboard input before booting the default menu entry

- הפרמטר TIMEOUT_GRUB מגדיר את מספר השניות שבהן יוצג למשתמש
- ה menuשדרכו ניתן לבחור את מערכת ההפעלה אותה נרצה להפעיל. במידה ובמשך
- הזמן שבו הפרמטר מוגדר, לא קיבלנו קלט מהמשתמש תטען מערכת ההפעלה
- שמוגדרת כברירת מחדל. כאשר ערך הפרמטר 0, מערכת ההפעלה הדיפולטיבית
- תטען מיד. כאשר ערך הפרמטר הוא -1, ה menu יוצג ללא הגבלת זמן עד קבלת קלט
- מהמשתמש. במקרה שלנו כאשר הפרמטר מוגדר להיות 5, ה menu יוצג למשך 5 sec
- חסרונות: זמן המתנה יותר ארוך לעליית מערכת ההפעלה.
- יתרונות: אפשרות נוחה לבחירת מערכת ההפעלה בה נרצה להשתמש ומשך זמן ארוך יותר לעשות זאת .

4. (6 נק') מדוע הפונקציה run_init_process() אשר נמצאת בקובץ init/main.c בקוד הגרעין קוראת לפונקציה do_execve() במקום לקריאת המערכת execve()?

```
static int run_init_process(const char *init_filename)
{
```

```
        argv_init[0] = init_filename;
        return do_execve(getname_kernel(init_filename),
              (const char __user *const __user *)argv_init,
              (const char __user *const __user *)envp_init);
}
```

נסו להחליף את הפונקציות זו בזו ובדקו האם הגרעין מתקמפל.

פתרון :

- הפונקציה process_init_run רצה על קוד הגרעין ומטרתה ליצור את תהליך הinit הראשוני של מערכת ההפעלה. מכיוון שזהו קוד גרעין הפונקציה הנ"ל קוראת ל.do_execve
- קריאת המערכת היא פונקציית מעטפת שנקראת מuserspace ומחליפה את רמת ההרשאה והמחסניות מקוד משתמש לקוד גרעין, ולאחר מכן מבצעת את execve באמצעות הפונקציה()do_execve לכן כשאנחנו בתוך kernelspace כמו בקוד המדובר, משתמשים ב ()do_execve שמבצעת את הפעולה execveבתוך קוד הגרעין, ולא ל ()execve שמחליפה לקוד גרעין ואז מבצעת את הפעולה שכן אנו כבר בקוד הגרעין. ניתן גם לראות שהkernel לא מתקמפל לאחר ההחלפה של ()execve_do ב()execve .

השגיאה שהופיעה בטרמינל :

```
init/main.c: In function 'run_init_process':
init/main.c:953:9: error: implicit declaration of function 'execv' [-Werror=implicit-function-de
claration]
  return execv(getname_kernel(init_filename),
         ^~~~~
cc1: some warnings being treated as errors
scripts/Makefile.build:333: recipe for target 'init/main.o' failed
make[1]: *** [init/main.o] Error 1
Makefile:1091: recipe for target 'init' failed
make: *** [init] Error 2
make: *** Waiting for unfinished jobs....
```

5. (6 נק') מה עושה קריאת המערכת ()syscall? כמה ארגומנטים היא מקבלת ומה תפקידם? באיזו ספריה ממומשת קריאת המערכת ()syscall? היעזרו ב-man page בתשובתכן.

פתרון: is a small library function that invokes the system call whose assembly language interface has the specified *number* with the specified arguments

- syscallהיא פונקציה אשר מקבלת מספר משתנה של ארגומנטים החל מ 1 ומפעילה את הsyscall שהמספר שלה בassembly (המספר שמכניסים לרגיסטר rax לפני הקריאה ל syscall) הינו המס" של first argument של הפונקציה syscall . שאר הארגומנטים שהפונקציה מקבלת נשלחים לפי הסדר לקריאת המערכת אותה היא מפעילה,לפי הארגומנטים של קריאת מערכת זו .
- syscall משמשת גם לקריאה לקריאות מערכת שלא מומשה להן פונקציית מעטפת, בכך מבצעת את המעטפת ואת המעבר מ user mode->kernel mode , ולאחר מכן דואג לביצוע הפעולה עצמה Syscall אשר ממושת בספריית glibc.

6. (10 נק') מה מדפיס הקוד הבא? האם תוכלו לכתוב קוד ברור יותר השקול לקוד הבא?

```
int main() {
    long r = syscall(39);
    printf("sys_hello returned %ld\n", r);
    return 0;
}
```

רמז: התבוננו בקובץ `arch/x86/entry/syscalls/syscall_64.tbl` בקוד הגרעין.

- פתרון :
- הקוד מדפיס את הpid של התהליך הנוכחי מכיוון שלפי טבלת קריאות המערכת, הערך 39 הוא קריאת המערכת sys_getpid() , אשר getpid() נמצאת תחת unisted.h המשומשת ע"י המשתמש לכן זה שקול ל:

39	getpid	sys_getpid	kernel/sys.c

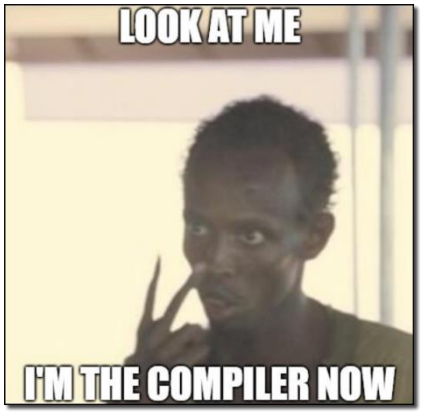
```
int main() {
long pid=(long)getpid();;
printf("sys_hello returned %ld\n", pid);
return 0;
}
```

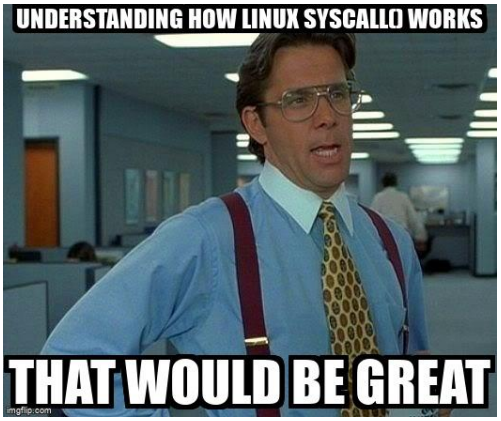
(10 נק') התבוננו בתוכנית הבדיקה `test1.c` שסופקה לכן והסבירו במילים פשוטות מה היא בודקת:

```
int main() {
    int x = get_weight();
    cout << "weight: " << x << endl;
    assert(x == 0);
    x = set_weight(5);
    cout << "set_weight returns: " << x << endl;
    assert(x == 0);
    x = get_weight();
    cout << "new weight: " << x << endl;
    assert(x == 5);
    cout << "===== SUCCESS =====" << endl;
    ;return 0
{
```

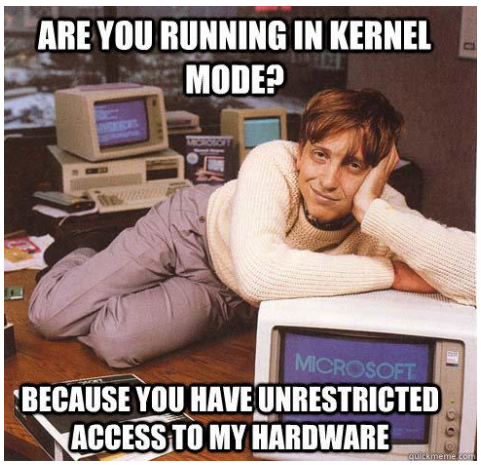
פתרון:

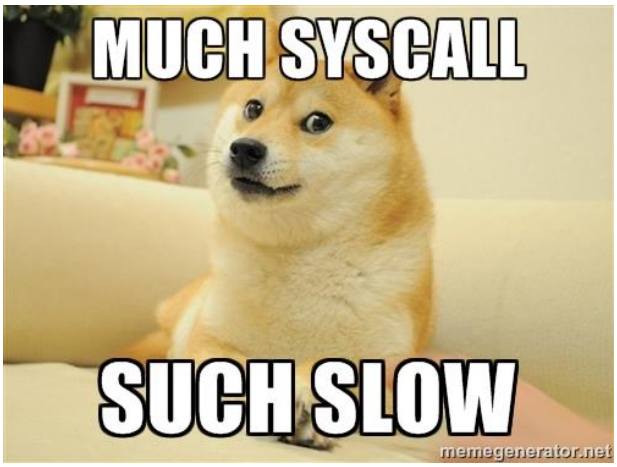
- בהתחלה התוכנית שומרת במשתנה X את משקל התהליך הנוכחי ומוודא שערכו 0 , אחר כך התוכנית משנה את המשקל של התהליך הנוכחי ל 5 , מוודא שערך חזרה זה הינו 0 ומודא שערך זה השתנה על ידי כך שהיא בודקת את המשקל של התהליך הנוכחי לאחר השינוי .


LOOK AT ME
I'M THE COMPILER NOW


UNDERSTANDING HOW LINUX SYSCALLO WORKS
THAT WOULD BE GREAT


HEY I JUST MET YOU AND THIS IS CRAZY
BUT I'M THE KERNEL SYSCALL ME MAYBE?
311 × 457


ARE YOU RUNNING IN KERNEL MODE?
MICROSOFT
BECAUSE YOU HAVE UNRESTRICTED ACCESS TO MY HARDWARE


MUCH SYSCALL
SUCH SLOW
memegenerator.net

חלק 2 - זימון תהליכים (50 נק')

1.

a. (4 נק') אילו משיטות התזמון הבאות עלולות לגרום לאפקט השיירה (convoy effect)? (שימו לב שעשויה להיות יותר מתשובה אחת נכונה)

i. FCFS
ii. RR
iii. SRTF
iv. כל מדיניות זימון עם הפקעה (preemption)
v. כל מדיניות זימון בלי הפקעה (preemption)
vi. אף תשובה אינה נכונה

הסבר: למדנו בהרצאה שכול מדיניות זימון בלי הפקעה עשויה לגרום לאפקט השיירה, בפרט שיטת זימון FCFS שהיא שיטת זימון ללא הפקעה. (כיוון שיכול להיווצר מצב שנריץ תהליך ארוך מאוד לפני תהליך קצר מאוד).

b. (3 נק') כפי שלמדנו, תחת תנאים מסוימים אלגוריתם SJF הינו אופטימלי עבור מדד זמן תגובה ממוצע. מהם שלושת התנאים?

תשובה:

על פי מה שלמדנו בהרצאה, תחת התנאים הבאים, אלגוריתם SJF הינו אופטימלי עבור מדד זמן תגובה ממוצע:

א) כאשר נעשה שימוש במעבד יחיד
ב) כאשר כול התהליכים מגיעים יחדיו
ג) כאשר זמן הריצה של כל תהליך ידוע מראש

2. (7 נק') כזכור, בתרגול ראיתן נוסחה לקצב התקדמות הזמן הווירטואלי ונוסחה לחישוב הקוונטום של כל תהליך. בסעיף זה ננסה לבחון את הקשר בין 2 הנוסחאות. הניחו כי:

a. בסוף כל epoch לכל התהליכים יש את אותו זמן וירטואלי.
b. הזמן הווירטואלי מתקדם ביחס הפוך לעדיפות התהליך: $VR_i += \frac{C}{W_i} dT$ עבור $C > 0$ כלשהו.
c. אורך ה-epoch הוא קבוע $\sum_i \quad Q_i = sched\_latency$.

הראו כי הנחות אלו גוררות את הנוסחה שראינו בתרגול לחישוב הקוונטום של כל תהליך:
$Q_i = (\frac{W_i}{\sum_j \quad W_j}) \cdot sched\_latency$.

<u>הערה:</u> שימו לב כי הנחות אלו הן הנחות מקלות שנועדו להקל עליכן לפתור את התרגיל. הנוסחה שהוצגה בתרגול נכונה כמובן גם ללא הנחות אלו, שאינן נכונות במקרה הכללי.

תשובה:

על פי הנחה a : בסוף כל epoch לכל התהליכים יש את אותו זמן וירטואלי, על כן נוכל להניח כי בסוף ה-epoch , $\frac{VR_i}{\sum_{i=1}^{n} VR_i} = \frac{1}{n}$.

$$\frac{\boldsymbol{c \cdot \Delta T_i \cdot \frac{1}{w_i}}}{\sum_{i=1}^{n} \boldsymbol{c \cdot \Delta T_i \cdot \frac{1}{w_i}}} = \frac{\boldsymbol{c \cdot \Delta T_i \cdot \frac{1}{w_i}}}{c \cdot n * \sum_{i=1}^{n} \boldsymbol{\Delta T_i} * \sum_{i=1}^{n} \boldsymbol{\frac{1}{w_i}}} = \frac{1}{n}$$

הקוונטום של תהליך בסוף ה-EPOCH יהיה זהה לזמן הריצה של התהליך מתחילת ה-epoch ועד סופו ולכן נוכל להגיד כי: $\boldsymbol{Q_i = \Delta T_i}$

ובנוסף לכך,סכום זמני הריצה של כול התהליכים בסוף ה- EPOCH יהי שווה לאורך ה-epoch שעל פי הנחה c שווה ל-sched_latency ולכן נוכל להגדיר כי: $\sum_{i=1}^{n} \Delta \boldsymbol{T_i} = sched\_latency$

נציב ונצמצם:

$$\frac{\boldsymbol{Q_i} \cdot \frac{\mathbf{1}}{\boldsymbol{w_i}}}{\boldsymbol{sched\_latency} \cdot \sum_{i=1}^{n} \frac{\mathbf{1}}{\boldsymbol{w_i}}} = 1$$

נעביר אגפים :

$$\frac{\boldsymbol{Q_i} \cdot \frac{\mathbf{1}}{\boldsymbol{w_i}}}{\frac{1}{\sum_{i=1}^{n} \boldsymbol{w_i}}} = \boldsymbol{sched\_latency}$$

סה"כ:

$$\boldsymbol{Q_i} \cdot \frac{\sum_{i=1}^{n} \boldsymbol{w_i}}{\boldsymbol{w_i}} = \boldsymbol{sched\_latency}$$

$$\boldsymbol{Q_i} = \frac{\boldsymbol{w_i}}{\sum_{i=1}^{n} \boldsymbol{w_i}} * \boldsymbol{sched\_latency}$$

כדרוש.

3. (24 נק') הילה, מומחית עולמית לאלגוריתמי זימון, החליטה לנסות לשפר את אלגוריתם CFS המוכר. הילה הציעה שיפורים שונים לאלגוריתם. בכל אחד מהסעיפים הבאים, מוצע שינוי/שיפור לאלגוריתם CFS. כתבו חיסרון שנקבל כתוצאה מאותו שינוי.

   a. (4 נק') תהליך שחזר מהמתנה יישאר עם אותו זמן וירטואלי שהיה לו כשהוא יצא להמתנה.

החיסרון הוא הרעבה אפשרית של שאר התהלכים:
במידה ותהליך יוצא להמתנה ארוכה, אזי הוא יוצא מהעץ. בינתיים, שאר התהליכים שנמצאים בעץ צוברים מזמן ריצה, כיוון שהם ממשיכים לרוץ. אם נחזיר את התהליך עם הזמן הווירטואלי שהיה לו כאשר יצא להמתנה, התהליך שהחזרנו יהיה בעל זמן ריצה ווירטואלי קטן באופן משמעותי (במידה ויצא להמתנה ארוכה) משאר התהליכים בעץ, ולכן ייווצר מצב בו אלגוריתם ה-CFS ימשיך לבחור בתהליך שחזר מהמתנה עד שיגיע למצב בו צבר זמן ריצה ווירטואלי מספק כדי שהאלגוריתם יבחר בתהליך אחר. הזמן הזה עלול להיות ארוך מאוד ולגרום להרעבת שאר התהליכים בעץ.

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

פגיעה בסיבוכיות:
פעולות חיפוש/הסרה/הכנסה לעץ אדום שחור הן בסיבוכיות O(LOG N) לעומת פעולות חיפוש/הסרה/הכנסה מרשימה יהיו סיבוכיות O(n) , במידה ונרצה לשמור על הרשימה מסודרת לפי זמן ריצה ווירטואלי, כדרוש עבור אלגוריתם CFS.

   c. (4 נק') הסרת המינימום על גודל הקוונטום של תהליך (min_granularity)

פגיעה בביצועים של האלגוריתם:

במידה ונסיר את min_granularity, האלגוריתם ימשיך להקצות קאונטום על פי הנוסחה הבאה: $Q_i = \frac{sched\_latency}{N}$ כאשר N מייצג את מספר התהליכים ו- sched_latenvy=48 ms . ועל כן, ככל שיהיו יותר תהליכים, על פי האלגוריתם, לכול תהליך יהיה זמן ריצה קטן יותר (כי N גדל ו-sched_latency קבוע) ויהיו החלפות הקשר מרובות, מה שיפגע בביצועים של התהליכים (החלפות הקשר מרובות מגדילות את התקורה).

d. (4 נק') הגדרת הקוונטום של כל תהליך, להיות בלתי תלוי במספר התהליכים, ובפרט $Q_i = a \cdot W_i$ (עבור קבוע חיובי aכלשהו)

הרעבה אפשרית של תהליכים ופגיעה בזמן התגובה של תהליכים:

במידה ונקבע את הקוונטום של כל תהליך לפי העדיפות שלו תוך התעלמות מכמות התהליכים, עלול להיווצר מצב בו תהליכים בעלי עדיפות גבוהה יקבלו קוונטום גדול, וירוצו זמן ארוך על המעבד, מה שעלול לגרום להרעבה של התהליכים בעלי עדיפות נמוכה יותר. בנוסף, כיוון שלא נתחשב בכמות התהליכים תיווצר פגיעה משמעותית בזמן התגובה של תהליכים (כי לא נקטין את הקוונטום של כול תהליך יקח לתהליך יותר זמן עד שיזומן לריצה) וזה כמובן פוגע ביעילות האלגוריתם.

e. (4 נק') שינוי קצב התקדמות הזמן הוירטואלי להיות מהצורה $VR_i \propto \frac{1}{\sqrt{W_i}}$

פגיעה בתהליכים בעלי עדיפות גבוהה- פגיעה בהוגנות של התהליך:

במקרה בו נוסיף לזמן הווירטואלי של תהליך את המכפלה של יחס זה לזמן הריצה שלו בפועל (כפי שמוסבר בפיאצה), אזי העדיפות של תהליכים בעלי משקל גבוהה ישפיע פחות על זמן הריצה הווירטואלי שהוא יקבל, כיוון שרכיב העדיפות נמצא תחת פונקציית שורש, וכידוע ככל שהמספר שתחת פונקציית השורש גדול יותר, ההקטנה היחסית של המספר גדלה. סה"כ- יווצר מצב בו עבור תהליכים בעלי עדיפות גבוהה ינתן פחות משקל לעדיפות של התהליך מאשר תהליך בעל עדיפות נמוכה ולכן יפגע זמן הריצה הווירטואלי שלהם ביחס לתהליכים בעלי עדיפות פחותה.

f. (4 נק') תהליכים חדשים נכנסים לתור הריצה עם זמן וירטואלי השווה לזמן הווירטואלי הממוצע של שאר התהליכים הקיימים במערכת.

הרעבה אפשרית של תהליכים בעלי זמן ווירטואלי גדול:

במצב בו תהליכים חדשים נכנסים לתור הריצה עם זמן וירטואלי השווה לזמן הווירטואלי הממוצע של שאר התהליכים הקיימים במערכת, במצב בו ישנה כמות גדולה של תהליכים חדשים כול הזמן, יהיה לכול התהליכים זמן ריצה ווירטואלי קצר מתהליכים ישנים שלהם זמן ווירטואלי ארוך מהממוצע ולכן יכול להיווצר מצב שתהליכים עם זמן ריצה ווירטואלי ארוך יותר יורעבו.

4. (6 נק') שירי, קולגה של הילה, רצתה להפחית את התקורה של חימום מטמונים בעת החלפת הקשר. לכן היא הציעה שתהליכים שמוכנים לריצה וחיים באותו מרחב זיכרון כמו התהליך שרץ אחרון על המעבד, יקבלו עדיפות על המעבד וירוצו לפני כל התהליכים האחרים. עומר, סטודנט למערכות הפעלה, טוען שהפתרון בעייתי כי בצורה זו תהליך יכול לגנוב זמן מעבד ולהרעיב תהליכים אחרים. הסבירו כיצד.

תשובה:

כידוע, חוטים השייכים לאותו תהליך נמצאים באותו מרחב זיכרון, ולכן המידה ונקבל את הצעתה של שירי, במידה וחוטים של אותו תהליך יהיו מוכנים לריצה , הם יקבלו עדיפות וירוצו לפני תהליכים אחרים. המצב הזה בעייתי כיוון שיכול להיווצר מצב שבו יהיה תהליך עם מספר חוטים, שכול פעם אחד מהם מוכן לריצה בזמן שהקודם לו סיים את ריצתן,

נתעדף את החוט שמוכן לריצה ובכך בעצם נאפשר רק לתהליך זה לרוץ ונרעיב תהליכים אחרים ונוסף על כך, יגנוב זמן ריצה כייוון שהתהליך ירוץ כול פעם על ידי חוט אחר שלו, יותר מהזמן הרצוי עבור התהליך.

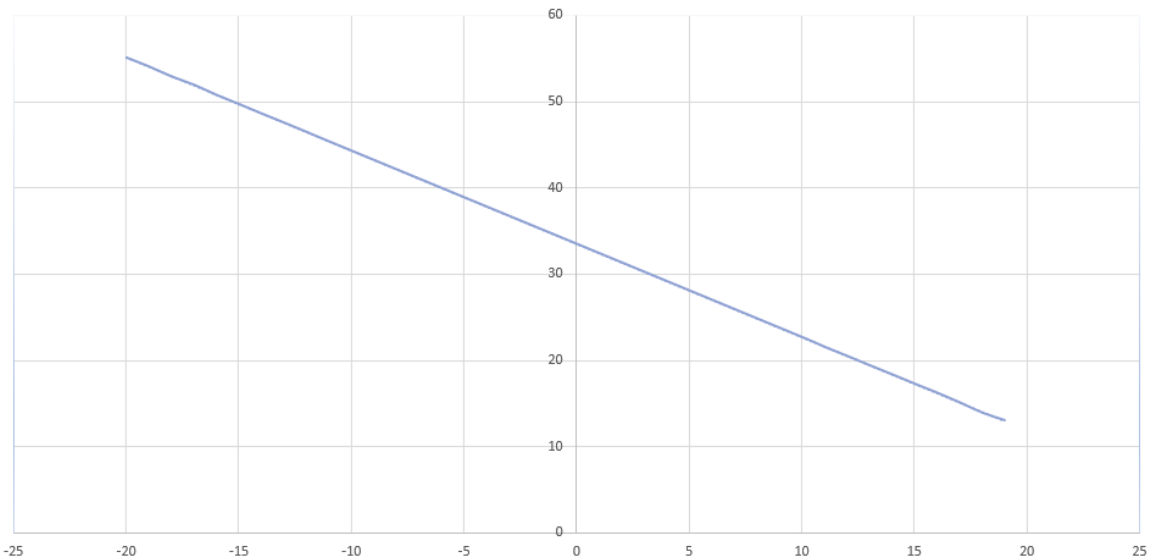
5. (6 נק') נזכיר, שבאלגוריתם CFS, לכל תהליך יש משקל שנקבע לו בהתאם לערך ה-nice שלו ($-20 \leq nice \leq 19$):

```
static const int prio_to_weight[40] = {
 ,36291    ,46273    ,56483    ,71755    ,88761     /* 20- */
,11916    ,14949    ,18705    ,23254    ,29154     /* 15- */
,3906     ,4904     ,6100     ,7620     ,9548      /* 10- */
,1277     ,1586     ,1991     ,2501     ,3121      /* 5-  */
,423      ,526      ,655      ,820      ,1024      /* 0   */
,137      ,172      ,215      ,272      ,335       /* 5   */
,45       ,56       ,70       ,87       ,110       /* 10  */
,15       ,18       ,23       ,29       ,36        /* 15  */
;{
```

כמו כן, ראינו כי גודל הקוונטום של תהליך, נמצא ביחס ישר למשקל התהליך:

$$Qi = \left(\frac{w_i}{\sum_j w_j}\right) \cdot sched\_latency$$

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



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

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

הקשר המתמטי בין ערך הnice למשקל התהליך הוא קשר אקספוננציאלי כיוון שכשהפעלנו פונקציית log על המשקלים, קיבלנו גרף לינארי כלומר קשר לינארי בין לוג המשקלים ל-nice .