Operating Systems – 234123

Homework Exercise 2 – Dry

מגיש: יקיר קרמר ורון וליצקי

ת.ז : 207678665 325892453

Mail: yakir.kremer@campus.technion.ac.il

ron.velitsky@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 –

   123456789_300200100.pdf
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/Fpw9SfbtLY?origin=lprLink

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?
2. משמשת להדפסה אוטומטית של קלט ומדפיסה ללא הפסקה
מקבלת מחרוזת כארגומנט ואותה מדפיסה ברצף
היעזרו ב-man page, ולאחר מכן השתמשו בפקודה ב-shell שלכן כדי לבדוק.

3. (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.
כשמריצים בלי מקבלים אזהרה כי היא מצפה לארגומנטים ואין לה ערך דיפולטיבי
על ידי yes אנחנו שולחים כארגומנט את המחרוזת הריקה וכך MAKE יודעת לבצע את הפעולה על פי ברירת המחדל והיא תמשיך בכך ללא הפסקה כי תקבל את המחרוזת ריקה ללא הפסקה.

4. (6 נק') מה משמעות הפרמטר GRUB_TIMEOUT בקובץ ההגדרות של GRUB?

```
GRUB_TIMEOUT=5
```

הסבירו מה היתרונות ומה החסרונות בהגדלת הפרמטר GRUB_TIMEOUT.
קובע למשך כמה זמן יופיע התפריט בחירת מערכת הפעלה שתטען
כאשר שווה ל0 תטען ברירת המחדל
כאשר שווה ל-1 יחכה עד שתבחר איזה לטעון
היתרון בלהגדיל זה שניתן יותר זמן להחליט לפני שעובר לדיפולטיבי
החסרון הוא שזה מגדיל את זמן העלייה של המערכת

5. (6 נק') מדוע הפונקציה run_init_process() אשר נמצאת בקובץ init/main.c בקוד הגרעין קוראת לפונקציה do_execve() במקום לקריאת המערכת execve()?
Run_init_process רצה במצב גרעין ולכן יכול לקרוא ישירות לפונקציות בגרעין ללא שימוש בקריאות מערכת
נחליף נראה כי הקוד לא מתקמפל מאחר והגרעין נטען לפני הספריות הסטנדרטיות של c ואינו מכיר אותן.

```
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.

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

Syscall משמשת לקריאה לקריאת מערכת וגם גורמת שיתבצע מעבר לקוד גרעין
מקבל כארגומנט את מספר הקריאה ולפי זה קוראת לפונקציה המתאימה
בנוסף אם יש צורך בארגומנטים נוספים לקריאת המערכת היא מקבלת ומעבירה אותם
ממומשת בספרייה glibc

7. (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.

מתבצעת קריאת מערכת מס 39 שזה המספר של sys_getpid() בגרעין והקוד ידפיס :
sys_hello returned <pid>
ניתן להחליף בקוד את הקריאה בgetpid() על מנת שהקוד יהיה קריא יותר, כי זו פונקצית המעטפת לקריאה הזו

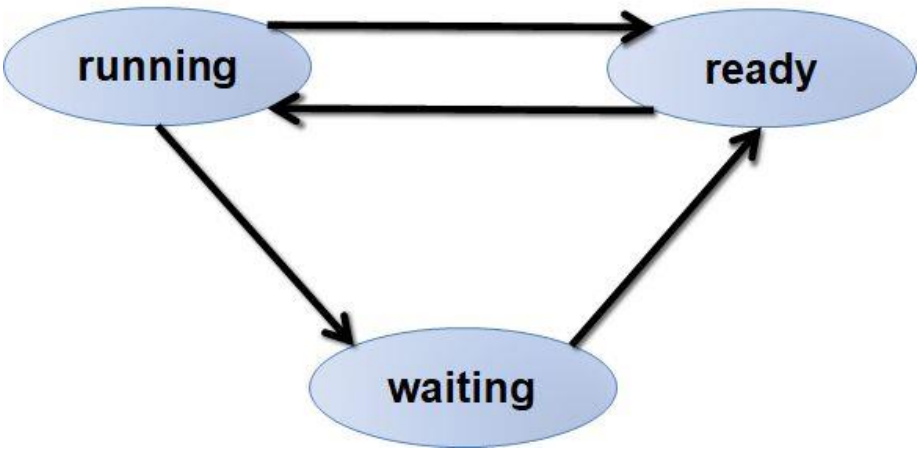
8. (10 נק') התבוננו בתוכנית הבדיקה test1.c שסופקה לכן והסבירו במילים פשוטות מה היא בודקת:

```
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;
}
```

התכנית בודקת שקריאת המערכת get_weight עובדת כמו שצריך ומחזירה 0 שהוא המשקל של אתחול ראשוני לתהליך שהגדרנו
לאחר מכן בודקים ש set_weight מחזירה 0 כמו שאמורה בהצלחה ומשנה כמו שצריך ל5
ובודקים שוב בעזרת get_weight שהמשקל השתנה באמת ל5

חלק 2 - זימון תהליכים (30 נק')

1. (6 נק') נתון התרשים המופשט של מצבי התהליכים:



עבור כל אחד מהמעברים תנו תרחיש המוביל למעבר:

a) waiting->ready
תהליך סיים לקרוא מקובץ מהדיסק

b) ready->running
הזמן קרא לתהליך הבא שאמור לרוץ

c) running->waiting
תהליך יוצא להמתנה כי הוא צריך לקרוא קובץ מהדיסק

2. (4 נק') אילו משיטות התזמון הבאות עלולות לגרום לאפקט השיירה (convoy effect)?

a) RR

b) FCFS

c) SRTF

d) אף תשובה אינה נכונה

3. (6 נק') כפי שלמדנו, תחת תנאים מסוימים אלגוריתם SJF הינו אופטימלי עבור מדד זמן תגובה ממוצע. מהם שלושת התנאים?
כל התהליכים מגיעים באותו זמן
זמן הריצה של כל תהליך ידוע מראש
אף תהליך לא יוצא להמתנה (אין פעולות I/O)

4. (4 נק') עבור מערכת זימון CFS, איזה בעיה פותרת ה min_granularity?
כאשר יש תהליכים רבין במערכת יכול להווצר מצב של quantom קטן מאוד שיגרום להרבה החלפות הקשר => תקורה שלהן ולכן נקבע min_gradualarity כך שיהי סף מינימאלי גם כאשר יש הרבה תהליכים ולפי החישוב צריך לתת קואנטום קטן הוא לא ירד מהסף הזה

5. (10 נק') נתונת מערכת זימון של לינוקס כפי שנלמד בתרגול עם אורך קוונטום 2ms, הרצה על מחשב עם מעבד יחיד. בהנחה שלמערכת מגיעים התהליכים הבאים:

process	Schedualing policy	Priority level	Expected runtime (ms)	Arrival time (ms)

A	SCHED_RR	30	4	0
B	SCHED_FIFO	30	3	1
C	SCHED_RR	31	3	3

איזה תהליך ירוץ בכל אחד מהזמנים הנתונים:

processes	A	A	B	B	B	A	A	C	C	C
Timestamp (ms)	0	1	2	3	4	5	6	7	8	9

הסבר: בזמן 0 רק A במערכת ולכן ירוץ לפי RR בקוואנטום של 2 ואז יופקע בזמן זה B כבר יהיה במערכת ולכן ירוץ עד הסוף כי הוא בFIFO ותהליך RR לא יכול להפקיע ממנו את המעבד לאחר שיסיים A וC יהיו במערכת A בעדיפות גבוהה יותר ולכן ירוץ עד הסוף ולאחר מכן רק C יישאר וירוץ עד הסוף

חלק 3 - מודולים (20 נק')

ענו עבור השאלות הבאות

1. (4 נק') ענו עבור השאלות הבאות האם הטענה ונכונה או לא, ונקמו.

a) מודולים מאפשרים להוסיף לגרעין לינוקס, **בזמן ריצה**, קטעי קוד חדשים שרצים **בהרשאות גרעין**. נכון זה מה שלמדנו בתרגול מודולים מאפשרים להוסיף קוד שרץ במצב גרעין ללא צורך לקמפל את הגרעין מחדש

b) לכל התקן במערכת, תמיד קיים מודול המתאים למספר הראשי של ההתקן, אשר יממש את פונקציות ה file_operations שלו. נכון ההתקן קורא לקריאות מערכת שעוברות דרך המודול שממפה את הקריאות דרך ה file_operations של המודול

2. (6 נק') מנו 2 הבדלים בין התקן תויים להתקן בלוקים.
תווים – אפשר לגשת כרצף תווים בודדים. בלוקים – אפשר לגשת רק בכפולות של בלוק.
תווים – משמשים לרוב להעברת מידע. בלוקים – משמשים לרוב לאחסון מידע.
תווים - לרוב מאפשר רק גישה סדרתית למידע. בלוקים - מאפשר גישה אקראית למידע.

3. (4 נק') נתונות הפקודות הבאות:

a) cat /dev/null
כן, dev/null מחזיר EOF

b) echo "hi" > /dev/full
לא. כי
echo
תקרא לקריאת מערכת
WRITE
שתנסה לכתוב שם אבל
dev/full
מחזירה
no space left on device

c) cat /dev/zero

כן
dev/zero
מחזירה רצף אפסים שיקרא על ידי
cat

d) echo "hi" > /dev/random

כן מה שיקרה זה שייכתב לרצף בתים אקראי

עבור כל פקודה ציינו האם אם תסתיים בהצלחה (תקרא ל exit עם ערך 0), ואם לא, מדוע.

4. (6 נק') נתון הקוד הבא:

```
#define MOD_NAME "MY_MODULE"
int my_major = 0;
struct file_operations my_fops = {
.open = my_open,
.release = my_release,
.read = my_read,
.write = NULL,
.llseek = NULL,
.owner = OWNER,
};
my_major = register_chrdev(my_major, MOD_NAME, &my_fops);
```

בהנחה שקקים מודול בשם MY_MODULE ושכל הפונקציות של my_ops ממומשות כראוי, עבור כל טענהת רשמו האם היא נכונה או לא, ונמקו.

a) בסיום השורה האחרונה יירשם דרייבר עם מספר ראשי ששווה ל 0.

לא נכון שליחת 0 כפרמטר של מספר ראשי גורם להקצאה דינמית כלומר המספר הראשון הפנוי לרישום דרייבר ולכן לא בהכרח (וכנראה שזה לא) יהיה 0.

b) ביצוע קריאת מערכת **seek** על התקן עם מספר ראשי המתאים למודול תמיד יסתיים בהצלחה.

לא כי הפונקציה אינה מוגדרת בfile_operations

c) ביצוע קריאת מערכת **write** על התקן עם מספר ראשי המתאים למודול תמיד יסתיים בהצלחה.

כן ההשמה של NULL גורמת לכך שיקרא מימוש ברירת המחדל