Homework 1 Dry

Due Date: 20/11/2018 23:30

Teaching assistant in charge:

- Shalev Kuba

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 hw1, put them in the hw1 folder

Only the TA in charge can authorize postponements. In case you need a postponement, please fill out the following form: https://docs.google.com/forms/d/e/1FAIpQLSftN-3vAFuM3pmzHjoWb-QokiROxWZG4q75osCe6ItVnwXLig/viewform

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:
    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.
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 (50 נק')

- את שאלה זו יש לפתור לאחר שפתרתם את החלק הרטוב.

1. 10) נק') מהו סדר הפונקציות/רוטינות בביצוע קריאת המערכת open?

a. sys_open → system_call → libc wrapper
b. sys_open →libc wrapper → system_call
c. system_call → sys_open → libc wrapper
d. system_call → libc wrapper→ sys_open
e. libc wrapper→ system_call → sys_open
f. libc wrapper→ sys_open → system_call

פרטו את צורת התקשורת בכל קריאה לפונקציה/רוטינה (העברת פרמטרים והחזרת תוצאות חישוב):

תשובה:

פונקציה קוראת	פונקציה נקראת	צורת העברת הפרמטרים לנקראת	צורת החזרת הפרמטרים לקוראת
Libc wrapper	System_call	העברת הפרמטרים היא ברגיסטרים eax,ebx,… כיוון שמתבצעת החלפת מחסניות.	החזרת תוצאות חישוב היא ברגיסטר eax כיוון שמתבצעת החלפת מחסניות.
System_call	Sys_open	העברת פרמטרים היא על גבי מחסנית הגרעין	החזרת פרמטרים היא על גבי מחסנית הגרעין

תזכורת: בחלק הרטוב הגדרנו רמות הרשאה לתהליכים וכן ספי הרשאה למספר קריאות מערכת "מאובטחות": fork, wait, waitpid, sched_yield. במערכת של תרגיל הבית, תהליך בעל רמת הרשאה x יוכל לבצע קריאת מערכת בעלת סף הרשאה y אם ורק אם $x >= y$.

נרצה להרחיב את מנגנון האבטחה כך שניתן יהיה לעדכן את רשימת קריאות המערכת המאובטחות ואת ספי ההרשאה שלהן ללא צורך בהידור מחדש של הגרעין. לשם כך, ניצור קובץ קונפיגורציה במיקום קבוע וידוע למערכת ההפעלה אשר מכיל רשימה של קריאות מערכת וסף ההרשאה המתאים לכל אחת מהן. המשתמש יכול לערוך את קובץ הקונפיגורציה באופן דינמי וכך לעדכן את מנגנון האבטחה ללא צורך בקידוד והידור מחדש של הגרעין.
שימו לב: מנגנון האבטחה נדרש להיות מעודכן לכל הפחות לזמן הפעלת המערכת האחרונה. כלומר, עריכת הקובץ בזמן ריצת המערכת לא בהכרח משפיעה מיד על מנגנון האבטחה.

לשם מטרה נעלה זו התגייסו כוכבי "בית הנייר" והציעו מימושים שונים להרחבה.

<u>ההצעה של טוקיו:</u> רמת ההרשאה של תהליך תשמר במשתנה גלובלי בגרעין מטיפוס integer. כמו כן, בתחילת כל פונקצית שירות (sys_X) תמקם קריאה לפונקציית גרעין בשם checkSec עם מספר קריאת המערכת. checkSec תקרא את קובץ הקונפיגורציה ותבדוק האם התהליך בעל הרשאה לבצע את קריאת המערכת.

2. 8) נק') האם המימוש של טוקיו תקין? אם לא, מה הבעיה במימוש זה?

<u>תשובה:</u>

המימוש של טוקיו לא תקין.
למשתנה הגלובלי הנ"ל כל אחד מהפונקציות בגרעין יכולה לפנות ולשנותו, וכמו כן, בהינתן שני תהליכים בעלי רמות הרשאה שונות אזי כאשר הם יגיעו לגרעין ותתבצע הקריאה ל-checkSec לשניהם יהיה אותה רמת הרשאה.
כמו כן, בהינתן תהליך בן שנוצר מתהליך אב , אם תהליך האב ירוץ במקביל לתהליך הבן (ללא wait) ייווצר מצב שבו יתכן והמשתנה הנ"ל לא יתעדכן כראוי ולכן לאב או לבן תהיה רמת הרשאה שאינה תואמת לדרוש עבורו באותו זמן.

<u>ההצעה של ריו:</u> רמת הרשאה של תהליך תשמר כשדה במתאר התהליך (PCB). כמו כן, ריו ימקם קריאה בתחילת system_call לפונקציית גרעין בשם checkSec עם מספר קריאת המערכת. הפונקציה checkSec תקרא את קובץ הקונפיגורציה ותבדוק האם התהליך בעל הרשאה לבצע את קריאת המערכת.

3. (8 נק') האם המימוש של ריו תקין? אם לא, מה הבעיה במימוש זה?

<u>תשובה:</u>

המימוש של ריו תקין.
ריו מתגבר על השגיאה של טוקיו כיוון שכאשר ירוץ תהליך מסוים תטען מחסנית הגרעין שלו ששם בכתובות הנמוכות נמצא ה-PCB, מחסנית זו היא יחודית לכל תהליך ולכן המשתנים בה יחודיים לתהליך זה. כמו כן, מערכת הפעלה תוכל להשתמש במשתנים אלה רק אם התהליך הנ"ל יטען (כלומר יהיה התהליך המוצבע ע"י המצביע current).

<u>ההצעה של הפרופסור:</u> רמת הרשאה של תהליך תשמר כשדה במתאר התהליך (PCB). כמו כן, הפרופסור יתחזק רשימה גלובלית בגרעין אשר מכילה מספרי קריאות מערכת וספי הרשאה המתאימים להן. את רשימה זו יאתחל רק בעלייה של המערכת על ידי קריאת קובץ הקונפיגורציה. בנוסף, ימקם קריאה בתחילת system_call לפונקציית גרעין בשם checkSec עם מספר קריאת המערכת. הפונקציה checkSec תבדוק האם התהליך בעל הרשאה לבצע את קריאת המערכת על ידי מעבר על הרשימה הגלובלית.

4. (8 נק') האם המימוש של הפרופסור תקין? אם לא, מה הבעיה במימוש זה?

<u>תשובה:</u>

גם המימוש של הפרופסור תקין.
פה, כמו במימוש של ריו, הפרופסור שומר את רמת ההרשאה של התהליך ב-PCB ולכן אין בעיה בצורת שמירת רמת ההרשאה לכל תהליך. השוני כעת הוא בכך שהפרופסור קורא את

קובץ הקונפיגורציה אך ורק בעליית מערכת ההפעלה והמידע בו נשמר ברשימה גלובלית.
לכן המימוש עדיין נכון מפני שרמות ההרשאה יהיו מעודכנות כפי הנדרש: "לכל הפחות לזמן הפעלת המערכת האחרונה".

5. 8) נק') מה היתרון של ההצעה למימוש של הפרופסור על פני המימוש של ריו?

תשובה:

היתרון הוא שהפרופסור מבצע קריאה יחידה מקובץ הקונפיגורציה רק בעת עליית המערכת כדי ליצור את הרשימה ולאחר מכן כל קריאה ל-checkSec תבצע בדיקה ברשימה במקום בקובץ.
לעומת ריו אשר מבצע קריאה מקובץ הקונפיגורציה בכל קריאה ל-checkSec.
היתרון נובע מכך שכל גישה לקריאה מקובץ דורשת הרבה יותר זמן ומשאבים מאשר גישה למשתנה גלובלי.

(8 נק') מה היתרון של ההצעה למימוש של ריו על פני המימוש של הפרופסור?

תשובה:

היתרון הוא שבגלל שריו קורא ישירות מקובץ הקונפיגורציה, רמות ההרשאה תמיד מעודכנות ברגע הקריאה, מה שלא קורה אצל הפרופסור מכיוון שהוא קורא את הקובץ אך ורק בעליית המערכת.

שאלה 2 (50 נק')

1. (18 נק')

A. (10 נק') ניר, ששונא לחכות, וגם מאמין במשפט "מה ששנוא עליך אל תעשה לתהליכך", החליט שבכל תוכניות המחשב שהוא כותב, הוא לעולם לא ישתמש בקריאת המערכת wait(). ליאור העיר לניר שאם לא ישתמש בקריאת המערכת הנ"ל ייאגר לו מידע בזיכרון על תהליכיו אשר סיימו להתבצע אך לא בוצע להם wait ("זומבים"), האם ליאור צודק? הסבר את טענתך.
הערה: ניתן להניח כי ניר לא כותב תכניות בהן קיים תהליך שרץ זמן רב.

תשובה:

ליאור טועה.
על מנת שהאב יקבל סטטוס על הבן שלו שהסתיים הבן עובר למצב של TASK_ZOMBIE (zombie) שבו הוא מתנהג כמו רשומת נתונים בלבד ללא ביצוע שום משימה , כלומר , הוא עדיין תופס מקום בזיכרון הגרעין.
במידה ותהליך האב הסתיים ולא ביצע wait אז כל תהליכי הבן שלו יעברו להיות בנים של init (שקיים כל עוד המערכת רצה). ו-init דואגת לביצוע wait על הבנים הזומבים שלה כל פרק זמן סביר ותשחרר את הזכרון אשר הם תופסים (הנ"ל בהנחה שניר לא מריץ תהליכים ארוכים).

(8 נק') שקד, שלמד על קריאת המערכת fork(), רצה להתנסות בבית בשימוש בה, ולכן כתב את קטע הקוד הבא:

```
int main(){
        int forkId=fork();
```

```
        if(forkId==0){//son
                printf("hey father, I am your son\n");
        }else{//father
                printf("hey son, I am your father\n");
        }
        return 0;
}
```

למרבה הצער, על המסך הודפס הפלט הבא (בהרצה מסוימת):

hey son, I am your father
hey father, I am your son

שקד התבאס מאוד שכן רצה שקודם הבן ידפיס למסך את ההודעה ורק לאחר מכן האב ידפיס את ההודעה שלו. עזרו לשקד, ע"י הוספת שורת קוד אחת בלבד, לגרום לתוכנית להדפיס בכל הרצה את הפלט:

hey father, I am your son
hey son, I am your father

<u>תשובה:</u>

בכניסה לתנאי של האב (התנאי מודגש בצהוב) יש להוסיף את השורה wait(NULL) מה שיבטיח שהאב ימתין שהבן יסיים לפני שהוא ימשיך לרוץ.

2. (15 נק') כזכור, מתאר התהליך מאוחסן ביחד עם מחסנית הגרעין שלו בקטע זיכרון בגודל KB8 המתחיל בכתובת מיושרת.
חברת נינוקס, החליטה לפתח מערכת הפעלה מודרנית יותר מהמערכת הנלמדת בתרגולים. בפרט, החברה טענה שלא יתכן שגודל כל מתאר תהליך יהיה מוגבל בגודלו, ולכן הפרידה את מתאר התהליך ממחסנית הגרעין (מתאר התהליך ומחסנית הגרעין כבר אינם צמודים כפי שנלמד בתרגולים) כך שגודל מתאר התהליך אינו מוגבל במערכת החדשה. נינוקס, שהעתיקה מלינוקס את קוד הקרנל הנלמד בתרגולים בלי לשנות דבר מלבד הפרדת מתאר התהליך ממחסנית הגרעין, הופתעה לגלות, יום לפני ההפצה של המערכת, שכאשר מבצעים קריאת מערכת שדורשת גישה למתאר התהליך, המערכת קורסת. עזרו לנינוקס להבין היכן הטעות שלה. על תשובתכם להיות מפורטת.

<u>תשובה:</u>

בלינוקס מתאר התהליך נמצא בתחילת מחסנית הגרעין של התהליך.
כל מחסניות הגרעין של התהליכים מיושרות לפי כפולה של 8KB ועל מנת להגיע להתחלת מתאר התהליך לינוקס מאפסת את 13 הביטים הנמוכים של esp כך היא מבטיחה שהיא מגיעה לכתובת שהיא גם כפולה של 8KB הקרובה ביותר מלמטה לכתובת שה-esp היה בה שהיא בעצם כתובת למתאר (PCB) של התהליך הנ"ל.
כאשר מפרידים ומזיזים את המתאר למקום אחר בזכרון החישוב הנ"ל אינו תקף יותר ולכן יתכן ופונים לכתובת לא מוגדרת או כתובת עם ערכים שאינם רלוונטים.

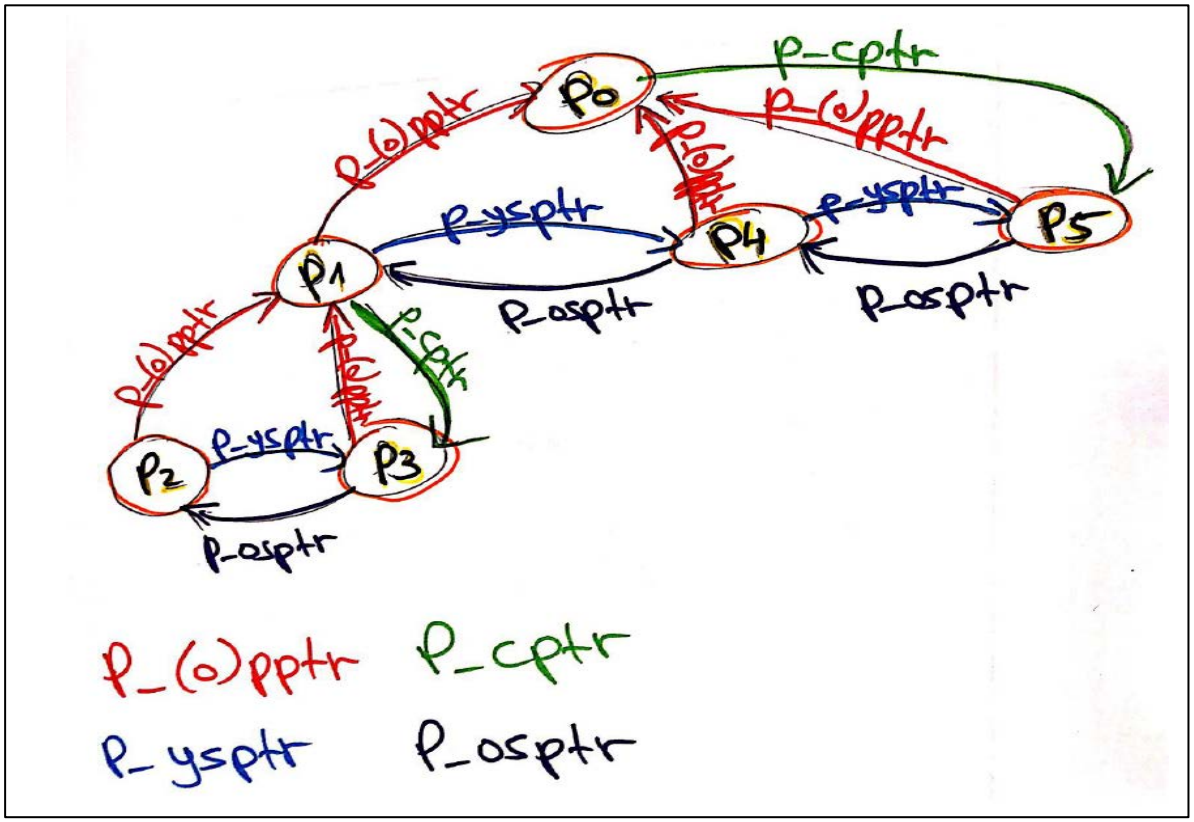
(17 נק')

בשאלה זו נדון בקשרי המשפחה כפי שבאים לידי ביטוי בשדות התהליך (p_(o)pptr, p_ysptr,…) ונלמדו בתרגולים:

A. (10 נק') עבור קטע הקוד הבא, ציירו את הגרף המתאר את קשרי המשפחה, כנלמד בתרגולים, כפי שנראה במערכת רגע לפני שתהליך כלשהו מסתיים (ניתן להניח כי כל התהליכים בתכנית נוצרים לפני שתהליך כלשהו נגמר). הקפידו לרשום על כל חץ את שם השדה ובתוך הצומת רשמו את המחרוזת שאותה התהליך מדפיס:

```
int main(){//father
        printf("P0");
        int forkId=fork();
        if(forkId==0){
                printf("P1");
                forkId=fork();
                if(forkId==0){
                        printf("P2");
                        return 0;
                }
                forkId=fork();
                if(forkId==0){
                        printf("P3");
                }
                return 0;
        }//more code on the right
```

```
///(continue)
.
.
for(int i=4;i<6;i++){
        forkId=fork();
        if(forkId==0){
                printf("P%d",i);
                return 0;
        }
}
return 0;
}
```



B. (7 נק') תנו דוגמה לקריאת מערכת שנלמדה בתרגול, שבה משתמשים בקשרי המשפחה כדי לבצעה? הסבירו איך בא לידי ביטוי השימוש בקשרי המשפחה בה:

קריאת המערכת: getppid()
מחזירה את המזהה של תהליך האבא של התהליך הנוכחי.