OS - Homework 1 Dry

Submitters:

Name: Adi Reznik ID:308571546
Name: Shahak Ben Kalifa ID:311242440

Date: 16.04.18

שאלה 1 (50 נק')

1. (30 נק') שאלה זו מתייחסת לשלבי תהליך הקריאה לשירות מערכת ההפעלה מתהליך המשתמש .עבור כל שלב בתהליך המתואר ציין אם הוא מבוצע על-ידי קוד משתמש (כידוע קוד עם CPL=3), קוד בגרעין (CPL=0) או על-ידי החומרה (מעבד) - הקף בעיגול את המתאים. עבור כל השלבים, הוסף הסבר מה השלב מבצע. הניקוד של כל שורה הינו 3 נק'.

שלב	מבוצע על-ידי	הסבר
קריאה לפונקציית מעטפת עם פרמטרים במחסנית	קוד גרעין/ **קוד משתמש**/ חומרה	**פונקציית המעטפת אחראית לשליחת הפרמטרים לשירות והפעלת קריאת המערכת עצמה**
העברת הפרמטרים לרגיסטרים	קוד גרעין/ **קוד משתמש**/ חומרה	**לפני המעבר לשגרת הטיפול בפסיקה יוחלפו המחסניות לכן על פונקציית המעטפת להעביר את הפרמטרים דרך הרגיסטרים**
פקודת int 0x80	קוד גרעין/ **קוד משתמש**/ חומרה	**פונקציית המעטפת מבצעת קריאה לשגרת הטיפול בפסיקה 128, בה מטופלות כל קריאות המערכת בלינוקס**
מציאת פונקציית הטיפול בפסיקה	קוד גרעין/ קוד משתמש/ **חומרה**	**המעבד מוצא את פונקציית הטיפול מתוך מערך ה-IDT**
שמירת הרגיסטרים ss,esp,eflags,cs,eip במחסנית החדשה לאחר המעבר ל-kernel mode	קוד גרעין/ קוד משתמש/ **חומרה**	**המעבד שומר את הרגיסטרים בעקבות פסיקה יזומה או פסיקת חומרה על מנת לבצע החלפה של המחסנית ממחסנית המשתמש למחסנית הגרעין.** **- eip ו-cs מכילים את כתובת החזרה** **- ss ו-esp מצביעים למחסנית המשתמש** **- eflags מכיל את מצב המעבד**
שמירת orig_eax במחסנית, אשר מייצג את מספר השירות המבוקש, וקריאה ל-SAVE_ALL	**קוד גרעין**/ קוד משתמש/ חומרה	**מתבצע בשגרת הטיפול בפסיקה על מנת שתקבל את מספר פסיקת השירות המבוקשת ורגיסטרים נוספים לשימוש כללי בעזרת המאקרו SAVE_ALL**
איתור כתובת פונקציית השירות ב syscall_table ובדיקה שמספר השירות המבוקש (מספר קריאת המערכת) נמצא בטווח החוקי של מספרי השירות האפשריים	**קוד גרעין**/ קוד משתמש/ חומרה	**קוד הגרעין בודק את מספר קריאת המערכת כנגד הקבוע NR_syscalls שערכו בלינוקס הוא 256 וקופץ לכתובת syscall_table+4*eax**
קריאה לפונקציית השירות	**קוד גרעין**/ קוד משתמש/ חומרה	**מתבצע על ידי קוד הגרעין אשר מפעיל את פונקציית ביצוע השירות nobadsys ובשימוש בפרמטר eax אשר נשמר על ידי המשתמש וכעת נמצא במחסנית הגרעין**

שלב	מבוצע על-ידי	הסבר
ביצוע שגרת השירות	**קוד גרעין/** קוד משתמש/ חומרה	**שגרת השירות כתובה ב-C מבוצעת ב-CPL=0 ברמת הגרעין**
בהנחה והתרחשה שגיאה בקריאת המערכה, כתיבת קוד השגיאה ל-errno	קוד גרעין/ **קוד משתמש/** חומרה	**פונקציית המעטפת מקבלת את ערך החזרה מפסיקת השירות ומעדכנת את המשתנה הגלובלי errno בהתאם**
החזרת ערך קריאת המערכת למשתמש	קוד גרעין/ **קוד משתמש/** חומרה	**ערך ההחזרה מהקריאה מועבר בפונקציית המעטפת דרך הרגיסטר eax שבו השתמשנו לשמירת מספר הפסיקה**

בסעיפים הבאים (2,3,4) הוצעו שינויים במנגנון הטיפול בפסיקות. בכל הסעיפים הבאים **מערכת ההפעלה והמעבד נותרים ללא שינוי, מלבד השינוי המוצע**.

2. (6 נק') השינוי המוצע: בקבלת פסיקה ישמר **גם רגיסטר ebx**, בנוסף לרגיסטרים שנשמרו במימוש המקורי על המחסנית הגרעין. להלן שרטוט הממחיש את המימוש החדש:

	סדר השמירה המקורי	סדר השמירה החדש
בסיס המחסנית	ss	ss
|	esp	esp
|	eflags	eflags
|	cs	cs
V	eip	eip
ראש המחסנית		**ebx**

בנוסף, כדי להשלים את המימוש, הוצע שפקודת iret תשלוף את **רגיסטר ebx** ולאחר מכן ישלפו שאר הרגיסטרים כפי שהיה במימוש המקורי.
האם המימוש תקין? אם לא, איזו בעיה עלולה להיווצר במימוש?
תשובה:
<u>המימוש תקין, הרגיסטר ebx ישמר פעמיים - פעם אחת בעקבות קבלת הפסיקה והשמירה בסדר החדש ובפעם השנייה בסוף המאקרו SAVE_ALL. ניגשים לרגיסטרים לפי מיקומם ביחס לראש המחסנית אשר לא ישתנה בעקבות השינוי.</u>

3. (7 נק') השינוי המוצע: עם קבלת פסיקה לא **מחליפים מחסניות**, דוחפים את **eflags, cs, eip בלבד** בראש המחסנית הנוכחית, ועוברים לבצע את שגרת הטיפול בפסיקה.
בהתאם, בחזרה מפסיקה שולפים את שלושת הערכים שנדחפו ונשארים במחסנית הנוכחית.
האם המימוש תקין? אם לא, איזו בעיה עלולה להיווצר במימוש?
תשובה:
<u>המימוש אינו תקין, אי החלפת מחסניות מאפשרת למשתמש זדוני להכניס ערך לא תקין ל-esp ובכך לגרום לכך שקריאת המערכת תנסה להכניס ערכים לכתובת לא חוקית בזכרון.</u>

4. (7 נק') השינוי המוצע: בקבלת פסיקה 128 כדי לחסוך בשמירת רגיסטרים, הוחלט לא לשמור ולשחזר את רגיסטר eflags.להלן שרטוט הממחיש את המימוש החדש:

	סדר השמירה המקורי	סדר השמירה החדש
בסיס המחסנית	ss	ss
|	esp	esp
|	eflags	cs
|	cs	eip
V	eip	
ראש המחסנית		

בנוסף, כדי להשלים את המימוש, פקודת iret תשלוף את הרגיסטרים לפי הסדר כך שלא משחזרים את ערך eflags כמקודם, כלומר גם בחלק של שליפת הרגיסטרים נבצע את התיקון הדרוש.
האם המימוש תקין? אם לא, איזו בעיה עלולה להיווצר במימוש?
תשובה:
<u>המימוש אינו תקין, הרגיסטר eflags מכיל את מצב המעבד, תוצאות חישובים אחרונים ודגלי בקרה שיכולים להשתנות במהלך שגרת הטיפול בפסיקה. במקרה זה, לאחר החזרה מהפסיקה הערך לא ישוחזר לערכו הקודם ובמידה והקוד תלוי בו הוא לא ירוץ כמצופה.</u>

<u>שאלה 2 (50 נק')</u>

1. (18 נק')

א. (10 נק') ניר, ששונא לחכות, וגם מאמין במשפט "מה ששנוא עליך אל תעשה לתהליכך", החליט שבתוכניות מחשב 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.
<u>ליאור טועה מכיוון שהתהליכים בתוכנית לא רצים זמן רב, התהליכים שיסיימו לרוץ יהפכו תוך זמן קצר ליתומים מכיוון שתהליך האב שלהם יסתיים גם הוא ויהיו בנים של init אשר ישחרר את הזיכרון שלהם.</u>

ב. (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

תשובה:
נוסיף את פקודת ()wait **לפני** ההדפסה של האב (ב-else)
בכך כאשר תוכנית האב תתחיל לרוץ במקביל לתוכנית הבן, האב יחכה בלולאה עד אשר הבן יסיים את ריצתו ורק לאחר מכן ימשיך לרוץ על הקוד שלו ויבצע את ההדפסה בעצמו.

2. (15 נק') כזכור, מתאר התהליך מאוחסן ביחד עם מחסנית הגרעין שלו בקטע זיכרון בגודל 8KB 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.
הטעות נובעת מצורת הגישה למתאר התהליך. על מנת לגשת לכתובת מתאר התהליך, מערכת ההפעלה מאפסת את 13 הביטים הנמוכים של esp, כמו בשימוש המקרו current, זאת מכיוון שמתאר התהליך הוא החלק התחתון בבלוק המיושר לפי כפולות של 8K. לכן, לאחר השינוי של חברת נינוקס, שיטה זו תביא לגישה למקום לא ידוע בזיכרון ולכן המערכת תקרוס.

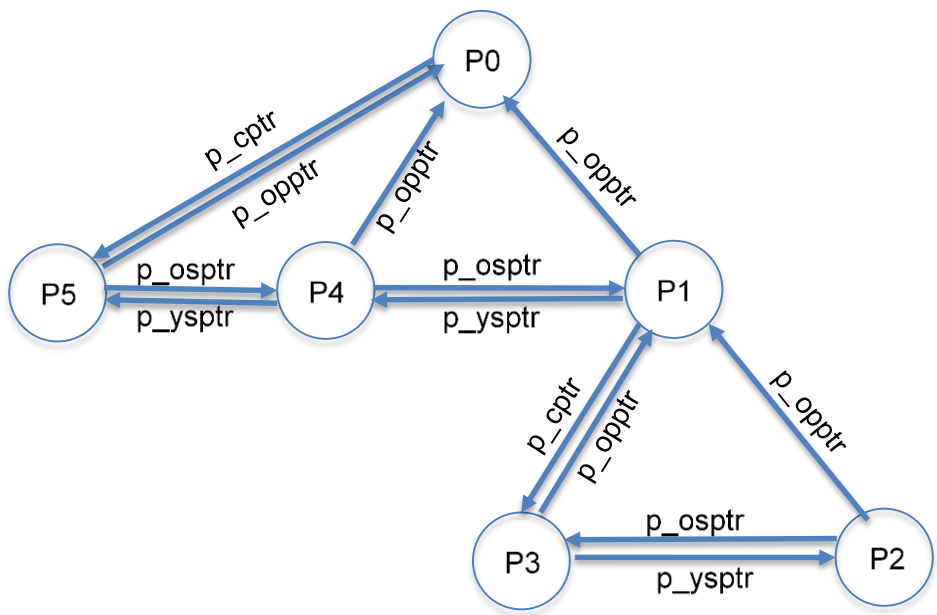
3. (17 נק')
בשאלה זו נדון בקשרי המשפחה כפי שבאים לידי ביטוי בשדות התהליך (p_(o)pptr, p_ysptr,…) ונלמדו בתרגולים:
א. (10 נק') עבור קטע הקוד הבא, ציירו את הגרף המתאר את קשרי המשפחה, כנלמד בתרגולים, כפי שנראה במערכת רגע לפני שתהליך כלשהו מסתיים (ניתן להניח כי כל התהליכים בתכנית נוצרים לפני שתהליך כלשהו 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;
}
```

ציירו כאן את תשובתכם:



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

<u>קריאת המערכת wait(), בקריאה זו תהליך האב ממתין עד שאחד מתהליכיו הבנים מסיים לרוץ.</u>
<u>כדי לבצע קריאה זו, על האב לבדוק בזמן ההמתנה האם אחד מבניו סיים ולכן עליו לעבור על</u>

בניו. בעזרת קשרי המשפחה יוכל תהליך האב לבצע גישה לבן הצעיר שלו בעזרת p_cptr וממנו להמשיך בריצה אל שאר בניו עד הבוגר שבהם בעזרת p_ostr ולאחר מכן לחזור שוב אל הצעיר בעזרת p_ysptr וחוזר חלילה.

שאלת בונוס(5 נק')

כידוע, ניתן להסתכל על ערכי הרגיסטר ebp ששמורים בבסיסי מסגרות הפונקציות, כרשימה מקושרת, כך שהאיבר הראשון של הרשימה נמצא ברגיסטר ebp. מצורף איור, שבו הודגשו בסיסי הפונקציות בכחול, יחד עם המצביעים (שהם למעשה הערכים אשר שמורים במחסנית), זאת כדי שתוכלו לראות בצורה נוחה יותר את הרשימה שנוצרת.

איזה כלי (שכולכם מכירים ממת"מ) מבצע שימוש נפוץ ברשימה זו. בתשובתכם הסבירו כיצד כלי זה משתמש ברשימה.

תשובה:

הכלי הוא ה-debugger אשר מאפשר לראות בכל שלבי ריצת התוכנית את רשימת הקריאות דרכה הגענו לפונקציה הנוכחית, רשימה זו היא רשימת ערכי ה-ebp.