Operating Systems – 234123

**Homework Exercise 2 – Dry**

**Name 1: Natan Borisov**

**ID1: 316595305**

**Email 1: natanborisov@campus.technion.ac.il**

**Name 2: Dor Rosh**

**ID2: 312147838**

**Email 2: dor.rosh@campus.technion.ac.il**

חלק 1

1. פקודה בלינוקס מקבלת מחרוזת ומדפיסה באופן רצוף אותה עד שהיא מתה. לחלופין, יכולה לקבל את הדגל --help שתדפיס את מסמך המידע והעזרה של הפקודה ותסיים או לקבל את הדגל --version המציג מידע על גרסת הפקודה ואז הפקודה תסיים.
2. בהרצת הפקודה make oldconfig עלינו לקבוע ערכים לפרמטרים שונים בקובץ ה-config , ובמקרה שלנו נרצה לאשר את כל ערכי ברירת המחדל, וכדי לחסוך את כל הלחיצות על enter במקלדת, הפקודה yes '' בתחילת הPIPE תדמה לנו את אפקט הלחיצה על enter מכיוון והארגומנט הריק שנשלח אליה יגרום לה להדפיס ירידות שורה עד ש-make oldconfig תסתיים והצינור ייסגר ויגרום לעצירת הפקודה yes.
3. משמעות הפרמטר GRUB\_TIMEOUT היא אורך הזמן בו יוצג התפריט לבחירת גרעין מערכת ההפעלה הרצוי לטעינה לזיכרון.

חסרון להגדלת הפרמטר: לאחר שבחרנו גרעין רצוי לטעינה נצטרך לחכות זמן מיותר לסיום ההשהיה.

יתרון להגדלת הפרמטר: יהיה מספיק זמן לבחור גרעין רצוי (נניח במידה והגרעין הדפולטיבי הוא לא זה שנרצה לטעון, לא נספיק לבצע את ההחלפה לגרעין הרצוי)

1. הפונקציה run\_init\_process שייכת לקרנל, לעומתה, execve היא מעטפת לקריאת המערכת עבור user mode, והגרעין אינו מכיר אותה. לכן, run\_init\_process חייבת לקרוא ל-do\_execve משום שהיא שעושה את הפעולה הרצויה בתוך הגרעין.

אם נחליף את הפונקציות זו בזו נקבל שגיאת קומפילציה. (מהסיבה המוזכרת לעיל)

1. קריאת המערכת syscall() מקבלת את number - מספר קריאת המערכת ופרמטרים נוספים כתלות בקריאה שמספרה צוין ב-number. היא קוראת לקריאת המערכת המתאימה עם הפרמטרים שהתקבלו.

היא ממומשת בספריה .

1. הקוד מדפיס syscall returned PID כאשר PID הוא ה-PID של התהליך הקורא (main).  
   קוד שקול וברור יותר:

#include <iostream>

#include <unistd.h>

int main() {

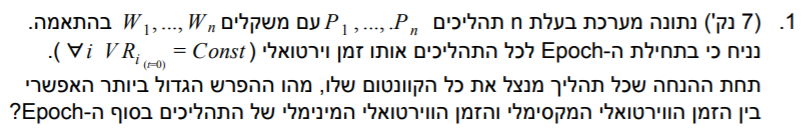
std::cout << "syscall returned " << getpid() << std::endl;

return 0;

}

1. התכנית מנסה לבדוק שהפונקציה get\_vruntime() מתפקדת. מכיוון ומדובר בזמנים מאד קצרים, נבדקת שגיאה יחסית עם tolerance (טווח ערכים) שהוגדר מראש (0.1). היא עושה זאת ע"י הגדרת "סטופר" שמחכה 5 שניות ובסיומו בודק שהתוצאה הצפויה (theoretical) "קרובה מספיק" (ע"פ ההסבר הנ"ל) לתוצאה שנמדדה (measured). במידה וכן מודפסת הודעת SUCCESS ואחרת ה-assert נכשל.
2. הפקודה taskset -c 0 "משייכת" את התכנית שמתקבלת כארגומנט שלה לליבה מספר 0.  
   במקרה שלנו, אם לא נשתמש בה, אנו עלולים להיכשל בתכנית הבדיקה וזאת מכיוון והתכנית עלולה לרוץ על ליבות שונות בהרצות שונות, מה שעלול לגרום לה לא לקבל את הבונוס על ריצה באותו מעבד, וכתוצאה מכך ה-vruntime שלה יהיה קטן יותר, והוא עלול לחרוג מהשגיאה היחסית ביותר מ-tolerance המוגדר.

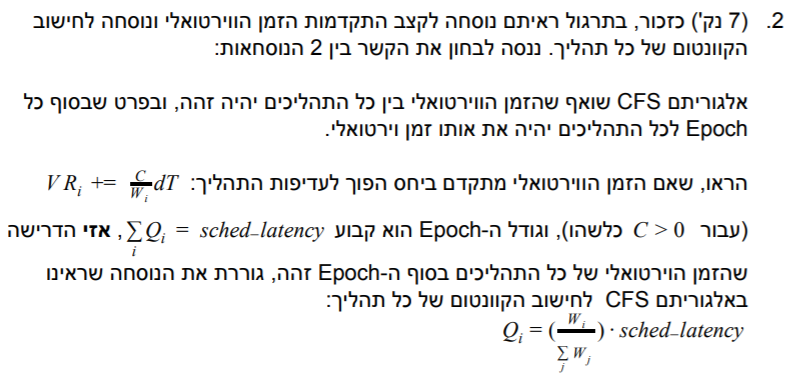
חלק 2



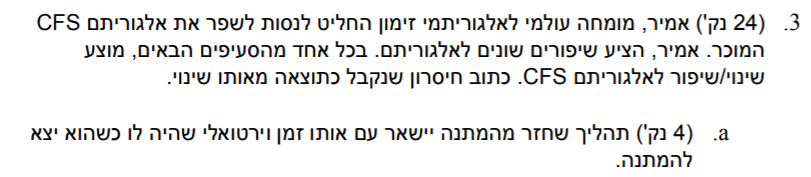
פתרון:

נסמן:

מתקיים:  
ע"פ הנתון, בתחילת ה-Epoch לכל התהליכים אותו זמן וירטואלי ולכן מתקיים:  
לכל שני תהליכים מתקיים:  
  
כלומר, ההפרש המוצע לעיל אכן ההפרש הגדול ביותר האפשרי בין זמני הריצה הווירטואליים.



פתרון:



*פתרון:  
  
סעיף א': הרעבת תהליכים: אם תהליך שחוזר מהמתנה יישאר עם אותו זמן וירטואלי שהיה לו כשהוא יצא להמתנה, ייתכן שהוא יקבל עדיפות על פני תהליכים אחרים ויגרום להרעבתם.  
לדוגמה, התרחיש שצוין בתרגול :*

* שני תהליכים A,B רצים זה לצד זה.
* תהליך B יוצא להמתנה ארוכה של 10 שניות.
* כאשר B מתעורר, הוא נכנס לעץ עם vruntime קטן ב-10 שניות מזה של A.
* לפי CFS, תהליך B יהיה זה שירוץ ב-10 השניות הבאות.
* הרעבה של תהליך A.

**

*סעיף ב': פגיעה בסיבוכיות האלגוריתם:   
אנו משתמשים בעץ אדום-שחור משום שהוא מאפשר פעולות הכנסה, חיפוש ומחיקה בסיבוכיות . לעומת זאת, לפחות אחת מפעולות אלה ברשימה תהיה יקרה יותר מבחינת סיבוכיות שכן תבוצע ב-.  
ככל שמספר התהליכים במערכת גדל, התקורה תעלה משמעותית יותר כתוצאה מהפעולות הלינאריות שנצטרך לעשות בשל השימוש ברשימה וכתוצאה מכך ביצועי המערכת יורדים.*

**

*סעיף ג': הגדלת התקורה  
כידוע, החלפת ההקשר "עולה" מחיר מסוים (ונעדיף לדאוג למצב שבו אחוז התקורה נמוך יחסית). בהסרת המינימום על גודל הקוונטום של תהליך, ייתכן וניצור מצב בו נאפשר לאפשר לתהליכים לרוץ זמן קצר מאד על המעבד ותדירות החלפות ההקשר תהיה גדולה משמעותית, כך שזמן מעבד ילך לאיבוד לצורך החלפות הקשר אלה.  
לדוגמה:  
נניח כי זמן epoch הינו 48ms, זמן ביצוע החלפת הקשר הינו 1ms ומספר התהליכים במערכת הינו 48 ואנו רוצים להריץ את כולם במשך זמן ה-epoch.  
בנוסף, נניח כי משקל כל התהליכים זהה (כלומר, עדיפויות שוות).  
לכן, הקוונטום של כל תהליך הינו 1 ms. במקרה כזה, התקורה היא 50% וזו פגיעה משמעותית בתפקוד המערכת.*

**

*סעיף ד': הרעבת תהליכים:  
הגדרת קוונטום באופן זה אינה מתחשבת במספר התהליכים במערכת וכתוצאה מכך יכולה ליצור הרעבה של תהליכים.  
לדוגמה, תהליך אחד בעדיפות 20- ושאר התהליכים בעדיפות 19.  
כתוצאה מהגדרת הקוונטום המוצעת, התהליך הראשון הינו בעל משקל מקסימלי וכך התהליך יגזול את רוב ה-epoch או יחרוג ממנו (כתלות ב-a), בעוד שאר התהליכים לא ירוצו כלל. (או שירוצו מעט אך עקרון ההוגנות ייפגע)*

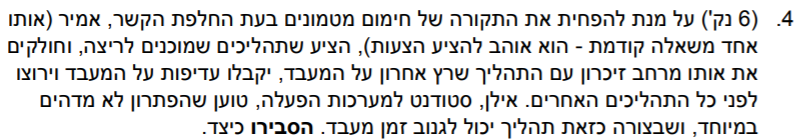
**

*סעיף ה': פגיעה בחלוקת זמן ה-epoch בין התהליכים השונים:  
  
שינוי קצב ההתקדמות לצורה הנ"ל מצמצם באופן משמעותי את ההפרשים בין תהליכים בעדיפויות שונות, כלומר את יחס החלוקה של ה-epoch. כמו כן, המשקולות שנלמדו בתרגול, עברו אנליזה ומיטוב כדי להשיג את החלוקה הרצויה בצורה המדויקת ביותר, וללא עדכון של המשקולות לא בטוח שנשיג את אותו אפקט שאליו כיוונו. (עבור תהליכים בעדיפות גבוהה נקבל זמן epoch קצר יותר ממה שתוכנן עבורם באופן אידיאלי, דבר שיפגע בתפקוד המערכת)*

**

*סעיף ו': הרעבת תהליכים :  
  
המנגנון החדש יכול בתרחישים מסוימים (ראה דוגמה) להרעיב תהליכים שהזמן הווירטואלי שלהם גדול מהממוצע של כל התהליכים במערכת.*

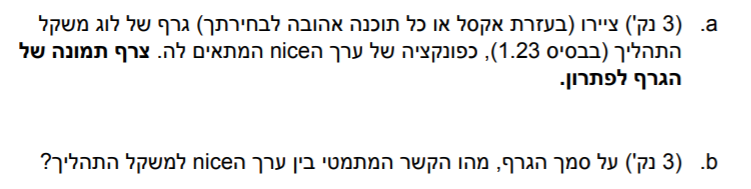
*לדוגמה: במערכת 3 תהליכים, עם זמנים וירטואליים של 1,2,3 ננו-שניות.  
תהליך 1 רץ, לכן בעץ התהליכים המוכנים לריצה יש שני תהליכים שזמן הריצה הווירטואלי שלהם הוא 2,3 ולכן הזמן הממוצע של התהליכים במערכת הוא 2.5, לכן הבנים של התהליך הראשון ייכנסו עם זמן וירטואלי 2.5 ואז הם יקבלו עדיפות על פני התהליך 3.  
אם נמשיך באופן זה, הבנים של תהליך 1 יקבלו עדיפות על פני תהליך 3 שיורעב.  
דבר שמצביע על חוסר הוגנות ותפקוד שגוי של המערכת.*

**

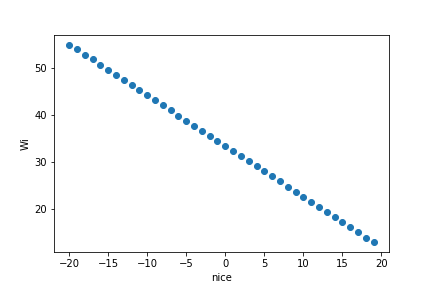
*פתרון: נשים לב שכתוצאה מהצעתו של אמיר נוכל לגרום לכך שתהליך שחולק אותו מרחק זיכרון עם התהליך שרץ אחרון על המעבד ובעל עדיפות נמוכה יותר ירוץ לפני תהליכים אחרים שמוכנים לריצה ללא תלות בעדיפות של התהליך.*

*לדוגמה: חוט בעדיפות 19 שחלק את אותו מרחב זיכרון עם התהליך האחרון שרץ על המעבד, ותהליך נוסף בעל עדיפות 20- שלא חולק את אותו מרחב זיכרון עם התהליך שרץ אחרון על המעבד. ייתכן שכתוצאה מסיטואציה זו, החישוב בפונקציה המחליטה על התהליך הבא שצריך לרוץ תחזיר את התהליך עם עדיפות 20-, בעוד שהצעתו של אמיר תחליט שהתהליך שצריך לרוץ הוא התהליך עם עדיפות 19.  
דוגמה אחרת – תהליך שלו מספר חוטים, שאחד מהם בעל עדיפות נמוכה וחוט אחר היה זה שרץ על המעבד אחרון. בשל ההצעה של אמיר, כל החוטים של התהליך הנ"ל ירוצו לפני תהליכים אחרים שלא חולקים את אותו מרחב זיכרון וייתכן שהיו בעלי עדיפות גבוהה יותר מחוטים אלה. בכך בעצם נגנב זמן מעבד ע"י התהליך עם החוטים.*

*שאלה 5*

**

*פתרון:*



a.

b. *מהגרף ניתן לראות שהקשר בין ערך ה-*nice *למשקל* *התהליך הוא לינארי.*