**תרגיל בית 2 – מערכות הפעלה**

יניב הולדר

207025297

[yaniv.holder@campus.technion.ac.il](mailto:yaniv.holder@campus.technion.ac.il)

אילון קורנבוים

315677880

[eilon.ko@campus.technion.ac.il](mailto:eilon.ko@campus.technion.ac.il)

חלק 1:

1. הפקודה yes מחזירה מחרוזת שוב ושוב עד שהיא נהרגת. הארגומנטים שהפעולה מקבלת הם מחרוזת (אם אין הברירת מחדל עבור מחרוזת היא ‘y’) שתודפס ו Options כמו --help, --version
2. שימוש בפקודה כמו בסעיף גורם לכך שבמקום שנתבקש לבחור בכמה שלבים שונים במהלך ביצוע הפקודה make oldconfig אפשרויות, הפקודה yes תיתן כקלט מחרוזות ריקות לפקודה make oldconfig וכך הפקודה תפעל ברצף וללא בקשות לקלט מהמשתמש.
3. המשתנה GRUB\_TIMEOUT קובע את כמות השניות שיוצג תפריט ה GRUB עד שיתחיל תהליך ה boot עם ברירת המחדל, אם לא נלחץ אף מקש. יתרון בהגדלת הזמן הוא שלמשתמש יהיה יותר זמן להגיב לתפריט, אך החיסרון הוא שאם אין צורך למשתמש בתפריט הפעלת המחשב תתעכב בגלל שצריך לחכות שיתחיל תהליך ה boot.
4. לא ניתן להשתמש בפונקציות רגילות בקוד גרעין (מכיוון שספריות משתמש אינן בגרעין), אלא רק בפונקציות גרעין ולכן נצטרך להשתמש בגרסה של הגרעין לפעולה execve(). אחרת, קוד זה לא יתקמפל מכיוון שהוא לא יזהה את המימוש של execve().
5. קריאת המערכת syscall(long number, …) מבצעת העלאת רמת הרשאה לרמת גרעין, קוראת לקריאת מערכת מסוימת לפי המספר number ומחזירה את רמת ההרשאה למשתמש. כמו כן, פונקציה זו שומרת רגיסטרים לפני קריאת המערכת ומשחזרת אותן לאחר מכן. הפונקציה מחזירה את ערך החזרה של קריאת המערכת. מעבר לארגומנט number, הארגומנטים שמקבלת הפונקציה משתנים לפי קריאת המערכת הספציפית שנקראת. הפקודה ממומשת בספריה glibc
6. קריאת מערכת מס' 39 היא getpid(). לכן, ניתן להחליף את השורה הראשונה אחרי הmain ב -: pid\_t r = getpid();
7. התוכנית בודקת את הפונקציות get\_total\_weight() ו set\_weight(). התוכנית בודקת שערך ההחזרה של הפונקציה get\_total\_weight() הוא 0 בתחילת הריצה, לאחר מכן שהפונקציה set\_weight(5) עובדת (מחזירה 0) ואכן לאחר מכן הפונקציה get\_total\_weight() מחזירה 5 (המשקל שניתן על ידי הפונקציה הקודמת). אם עברנו את כל הבדיקות סימן שהפונקציות עובדות ונחזיר “SUCCESS”.

חלק 2

1. *ההפרש הזמנים הוירטואליים המקסימלי (והמינימלי) בין תהליכים בסוף ה epoch הוא 0.*

*יהי תהליך עם משקל . ע"פ התרגול*

*כמו כן, בסוף ה epoch כאשר ולכן:*

*כלומר, לכל התהליכים ללא קשר למשקל יהיה בסיום ה epoch זמן וירטואלי זהה ולכן ההפרש הוא 0.*

1. *נניח את הנחות התרגיל. ע"פ a,b לכל מתקיים בסוף epoch:*

*ולפי c:*

*ולסיכום,*

1. *a) כפי שראינו בתרגול הצעה זו היא בעייתית שכן היא יכולה לגרום להרעבה של תהליכים בעץ אם תהליך יוצא להמתנה לזמן ארוך (בזמן זה הזמן הווירטואלי שלו אינו משתנה) ובחזרתו הוא ירוץ ברצף עד ש"ידביק" את זמן הריצה הווירטואלי של שאר התהליכים.*

*b) הצעה זו תפגע בסיבוכיות של אלגוריתם הזימון מכיוון שפעולות בעת אדום-שחור קורות ב O(log n) לעומת רשימה שלמשל חיפוש לוקח בה O(n). כלומר זמן ריצת האלגוריתם יפגע.*

*c) כפי שלמדנו בתרגול הסרה של ה min\_granularity עלול לגרום לכך שהקוונטום יהיה קצר מאוד וזמן החלפת ההקשר יהפוך להיות מאוד יקר ביחס לזמני הריצה של התהליכים (תקורה גבוהה) ולכן יפגע בביצועי אלגוריתם הזימון. נעדיף לשמור על איזון בין האינטראקטיביות ליעילות הריצה.*

d) ה- epoch הוא סכום כל הקוונטום של כל התהליכים. לכן עם השינוי המוצע, אם יהיו מספר גודל של תהליכים עם משקל גדול נקבל epoch גדול מאוד דבר שיפגע באינטראקטיביות של המשתמש. אם התהליכים יהיו עם משקלים קטנים מאוד נקבל epoch קטן מאוד דבר שיגרור החלפות הקשר תכופות ויפגע בביצועי הזמן של האלגוריתם (במידה ולא יהיה מינימום ריצה לכל קוונטום).

e) שינוי זה יגרום לשינוי היחס שבין זמני הריצה הווירטואליים בין תהליכים עם משקל גדול וקטן. כל התהליכים יצברו פחות זמן וירטואלי שכן לאחר השינוי:

אך עבור תהליכים עם משקל גדול ההבדל יהיה גדול יותר מאשר תהליכים עם משקל קטן (שכן שורש על מספר גדול מקטין אותו ביחס גדול יותר מאשר מספר קטן) וכך יקרה מצב בו תהליכים עם משקל גדול יעקפו תהליכים אחרים מכיוון שהם יצרכו פחות זמן וירטואלי ולא תהיה הוגנות בין הזמנים.

f) שינוי יכול לגרום להרעבת תהליכים עם זמן וירטואלי גדול (בפרט גדול מהממוצע) שכן אם יש תהליכים שיוצאים להמתנה וחוזרים לעיתים קרובות הם תמיד יעקפו את התהליכים הללו ולכן זה יפגע בהוגנות בין התהליכים.

1. עבור מערכת ההפעלה, תהליכים שחולקים מרחב זיכרון הם חוטים (threads) של אותו תהליך. אם חוטים אלו יקבלו עדיפות נוכל נקבל מצב בו אותו תהליך יקבל עדיפות שוב שוב עבור חוטים שונים שלו וכך תהליך זה יכול "לגנוב זמן מעבד ולהרעיב תהליכים אחרים"
2. a)

b) אנו רואים כי קיים קשר לינארי בין ה nice ללוג משקל התהליך.