אוניברסיטת בן-גוריון בנגב, המחלקה למדעי המחשב

מועד ב' במערכות הפעלה (18 ביולי 2010)

מרצים: דני הנדלר ואמנון מייזלס; מתרגלים: ליאנה דיזנדרוק, דניאל גורדון, אלון גרובשטיין ואמיר גרשמן

משך המבחן: שלוש שעות
חומר עזר: אסור
פתרי את כל השאלות: סה"כ 100 נקודות.

1. ניהול זיכרון (35 נקודות). בשאלה זו סעיפים מרובים, השיבו בקצרה ובמדויק על כל הסעיפים.

א. נתון כי במערכת רצים שני תהליכים A ו-B וכי המערכת משתמשת ב- Page tables. הניחו כי התוכניות במערכת אינן קוראות לפונקציית ()exec.

i. נניח כי B נוצר עקב הפעלת ()fork על ידי A, האם המערכת יכולה להשתמש באותה page table עבור שני התהליכים? נמקו בקצרה אך במדויק. [2 נק']

ii. נניח כי A ו- B שניהם תוצאה של הרצת אותה התוכנית c.exe. האם במקרה זה יכולה המערכת להשתמש באותה page table עבור שני התהליכים? נמקו בקצרה אך במדויק. [2 נק']

iii. נניח כעת כי המערכת משתמשת ב- segmentation with paging ו- B נוצר עקב הפעלת ()fork על ידי A. האם ניתן להשתמש באותה page table עבור שני התהליכים בלפחות סגמנט אחד? נמקו בקצרה אך במדויק. [2 נק']

ב. נתון כי המערכת תמיד מקצה כמות שווה של זכרון פיזי לכל תהליך. האם המערכת מממשת אלגוריתם החלפת דפים גלובלי או לוקלי? הסבירו בקצרה [4 נק']

ג. ענו על הסעיפים הבאים.

i. Swapping הינה שיטה לניהול זכרון. הסבירו בקצרה מהי השיטה, מנו שתי בעיות בשיטה זו. (4 נק')

ii. Virtual memory הינה שיטה נוספת לניהול זכרון. הסבירו בקצרה מהי השיטה וכיצד היא פותרת את הבעיות הקיימות בשיטת ה-swapping. (4 נק')

ד. במערכת המנהלת את הזכרון בעזרת page-tables, אילו נתונים נדרשים על מנת לתמוך בהוצאת דף מהזכרון (page swap out) כך שהמערכת תשמור על נכונותה. נמקו. (6 נק')

ה. נתון כי הזמן הנדרש לקריאת מידע הנמצא ב-TLB הוא 10 nsecs וגישה דומה ל-page-table לוקחת 100 nsecs. ב- TLB יש את הכניסות הבאות:
valid bit, modified bit, protection code, virtual address, frame number

i. מה צריך להיות ה-TLB hit-rate על מנת שהזמן הממוצע למציאת כתובת וירטואלית יהיה 30 nsecs? הסבירו את החישוב. (5 נק')

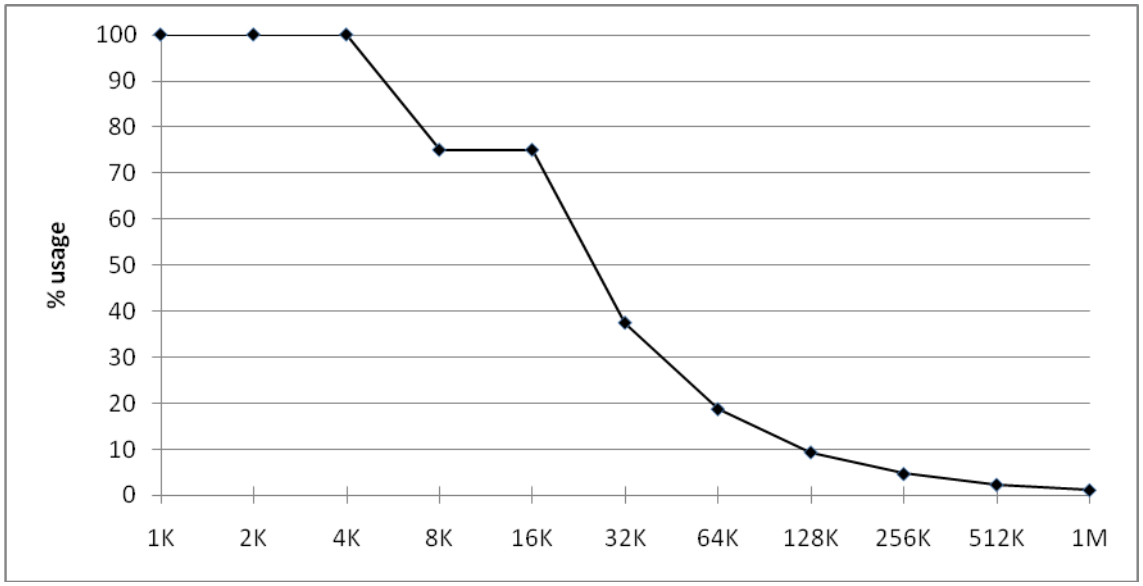
ii. בהנחה שבמערכת יש מעבד יחיד התומך ב-multi-processing האם ניתן להשתמש ב-TLB יחיד כזה (ללא שינוי כלשהו במבנה שלו)? אם כן, פרטו איך אפשר להשתמש

בו. אם לא, הסבירו מדוע. (3 נק')

iii. בהנחה כי במערכת ישנם מספר מעבדים, האם ניתן להשתמש ב- TLB יחיד כזה (ללא שינוי כלשהו במבנה שלו)? אם כן, פרטו איך אפשר להשתמש בו. אם לא, הסבירו מדוע. (3 נק')

2. מערכות קבצים (25 נקודות)

א. נתונה מערכת קבצים ובה לכל הקבצים אותו הגודל בדיוק. הגרף הבא מתאר את היחס בין גודל הבלוקים במערכת לבין ניצולת הדיסק:



i. מהו שמה של הבעייה הגורמת לירידת ניצולת הדיסק? **(2 נק')**

ii. מהו גודל הקבצים במערכת? **(4 נק')**

ב. נתון כי הקובץ \tmp\a.txt מכיל תו אחד בלבד. נתון כי המידע ומבני הנתונים של כל הספריות נמצאים בזיכרון הראשי אבל שום מידע של קבצים אחרים לא נמצא כרגע בזיכרון הראשי. במערכת זו, גודלו של כל בלוק הינו 8 KB.

i. במערכת UNIX, כמה גישות לדיסק דרושות על מנת להביא את תוכן הקובץ לזיכרון? נמקו בקצרה אך במדוייק. **(4 נק')**

ii. בכמה ,לכל הפחות, צריך לגדול תוכן הקובץ a.txt על מנת שתידרש גישה אחת נוספת מאשר בסעיף i ? נמקו בקצרה אך במדויק. **(4 נק')**

iii. בכמה, לכל הפחות, צריך לגדול תוכן הקובץ a.txt על מנת שיידרשו 13 גישות נוספות יותר מאשר בסעיף i כדי להביא את כל תוכן הקובץ לזיכרון? נמקו בקצרה אך במדויק. **(4 נק')**

iv. במערכת NTFS, כמה גישות לדיסק דרושות על מנת להביא את תוכן הקובץ לזיכרון? נמקו בקצרה אך במדויק. **(4 נק')**

v. בכמה, לכל הפחות, צריך לגדול תוכן הקובץ a.txt על מנת שתידרש גישה אחת נוספת מאשר בסעיף iv? נמקו בקצרה אך במדויק. **(3 נק')**

3. סינכרוניזציה – בעיית המניעה ההדדית (mutual exclusion) (25 נקודות)

להלן אלגוריתם למניעה הדדית עבור שני תהליכים תהליך 1 ותהליך 2.

```
shared int x, y both initially 0
Code for process i ∈ {1,2}

1.  start: x:=i
2.         if (y != 0)  {
3.           await y=0
4.           goto start
             }
5.         y:=1
6.         if (x != i) {
7.           y:=0
8.           await x=0
9.           goto start
10.        }
11.        CRITICAL SECTION
12.        y:=0
13.        x:=0
```

i. האם הקוד לעיל מקיים mutual exclusion? הוכיחו במדויק או הפריכו ע"י מתן תסריט מדוייק **(8 נק')**.

ii. האם הקוד לעיל מקיים deadlock-freedom? הוכיחו במדויק או הפריכו ע"י מתן תסריט מדוייק **(8 נק')**.

iii. נניח כעת כי הקוד לעיל משמש **שלושה** תהליכים, בעלי מזהים 1, 2 ו-3. האם הקוד עדיין מקיים mutual exclusion? אם כן, הוכיחו במדויק, אם לא, תנו מסריט מדויק המוכיח טענתכם **(7 נק')**.

4. סינכרוניזציה – סמפורים ובעיית המניעה ההדדית (15 נקודות)

סמפור בלתי הוגן (unfair semaphore) הוא סמפור אשר אינו מבטיח כי הסדר בו תהליכים מתעוררים על הסמפור הוא הסדר בו הם הלכו לישון עליו. כל אשר סמפור כזה מבטיח הוא, כי אם ישנם תהליכים הישנים על הסמפור בעת שמבוצעת עליו פעולת up אזי אחד מהם מתעורר (אך, כאמור, אין זה בהכרח התהליך מבין הממתינים אשר ביצע ראשון פעולת down על הסמפור).

בשאלה זו, עליכם לממש אלגוריתם חופשי מהרעבה למניעה הדדית (starvation free mutual exclusion) עבור שלושה תהליכים משלושה סמפורים בלתי הוגנים (unfair counting semaphores). **אין להשתמש במשתנים אחרים מלבד שלושה סמפורים אלו**. המבנה של הקוד אשר עליכם להשלים **במחברת** הוא כלהלן:

shared unfair-semaphore R, S, T all initially 1.

p1 ENTRY SECTION Critical Section p1 EXIT SECTION	p2 ENTRY SECTION Critical Section p2 EXIT SECTION	p3 ENTRY SECTION Critical Section p3 EXIT SECTION

עליכם להשלים את קטעי ה-entry section וה-exit section של כל אחד מן התהליכים. הסבירו בקצרה אך במדויק מדוע המימוש שלכם מקיים מניעה הדדית וחופש מהרעבה.

בהצלחה!