אוניברסיטת בן-גוריון בנגב, המחלקה למדעי המחשב

מועד א' במערכות הפעלה (26 ביוני 2011)

מרצים: דני הנדלר ואמנון מייזלס; מתרגלים: אלון גרובשטיין, אמיר גרשמן, אמיר מנצ'ל, ודולב פומרנץ

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

1. סינכרוניזציה – סמפורים ובעיית המניעה ההדדית **(30 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).

א. **(6 נק')** להלן אלגוריתם למניעה הדדית המשתמש בסמפורים מונים (counting semaphores) בלתי-הוגנים:

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

R.down( )
S.down( )
T.down( )

Critical section

T.up( )
S.up( )
R.up( )
```

מהו מספר התהליכים המקסימלי k כך שהאלגוריתם לעיל מספק <u>מניעה הדדית</u> <u>וחוסר הרעבה</u> (starvation-free mutual exclusion) עבור k תהליכים? נמקו במדוייק מדוע מתקיימים מניעה הדדית וחופש מהרעבה עבור k תהליכים (עבור אותו k המופיע בתשובתכם). כמו כן, תנו תסריט מדוייק המוכיח כי לא מתקיים starvation-freedom mutual exclusion עבור k+1 תהליכים.

ב. **(7 נק')** ענו בדיוק על אותה שאלה מסעיף א' עבור האלגוריתם אחרי שנעשה בו השינוי הבא: כל שלושה הסמפורים שבאלגוריתם מסעיף א <u>מאותחלים לערך 2 במקום לערך 1</u>.

ג. **(7 נק')** ענו בדיוק על אותה שאלה מסעיף א' עבור האלגוריתם אחרי שנעשה בו השינוי הבא: <u>הסמפור R מאותחל לערך 2 והסמפורים S, T מאותחלים לערך 1</u>.

ד. **(10 נק')** האם ישנם ערכים v1, v2, v3 כך שאם נאתחל את הסמפורים לערכים אלו בהתאמה (כלומר, R=v1 ,S=v2 ,T=v3) האלגוריתם לעיל יספק מניעה הדדית וחוסר הרעבה עבור ארבעה תהליכים? נמקו תשובתכם במדויק.

2. ניהול זכרון **(25 נק')**

א. **(5 נק')** בכיתה הגדרנו אלגוריתם להחלפת דפים כ*<u>אלגוריתם מחסנית</u>* אם הוא מקיים את היחס הבא עבור כל r ועבור כל m:

$$M(m, r) \subseteq M(m + 1, r)$$

הסבירו מהו M, מהו m, ומהו r , ומה משמעות יחס ההכלה לעיל.

ב. **(5 נק')** מהי אנומליית בלאדי (Belady's anomaly)? הסבירו מדוע אלגוריתם מחסנית (עפ"י ההגדרה לעיל) אינו סובל מאנומליית בלאדי.

ג. **(5 נק')** האם LRU הינו אלגוריתם מחסנית ? נמקו את תשובתכם.

ד. **(10 נק')** להלן סדרת גישות לדפים (reference string) – משמאל לימין:

S=1,2,3,4,5,6,5,6,7,8,1,5,4,1

נניח כי גודלו של הזיכרון הפיזי הוא ארבעה page frames.

1. לכמה page faults תגרום S אם אלגוריתם החלפת הדפים הינו FIFO? כתבו במדויק את החישוב עליו מתבססת תשובתכם.

2. לכמה page faults תגרום S אם אלגוריתם החלפת הדפים הינו אלגוריתם החלפת הדפים האופטימלי? כתבו במדויק את החישוב עליו מתבססת תשובתכם.

3. האם ניתן להקטין את מספר ה-page faults שתגרום S תחת אלגוריתם החלפת הדפים האופטימלי על ידי הגדלת הזיכרון הפיזי? אם כן, בכמה page frames צריך להגדילו? אם לא, מדוע אין הדבר ניתן? נמקו את תשובתכם.

3. מערכת הקבצים **(25 נק')**

א. **(5 נק')** בהרצאות תיארנו את מנגנון ה-buffer cache ב-Unix. תארו שני תסריטים שונים בהם תהליך המחפש בלוק ב buffer cache נחסם וחייב להמתין, ופרטו מה עושה התהליך כאשר הוא מתעורר מן ההמתנה.

ב. במערכת הפעלה כלשהי קיים מנגנון buffer cache. נסמן בלוק בדיסק על ידי מספרו הסידורי j. הניחו כי גודלו של כל directory במערכת אינו עולה על בלוק. הניחו גם כי בתחילת הריצה ה-buffer cache ריק.

1. **(5 נק')** בסעיף זה הניחו כי ה- buffer cache יכול <u>להכיל 8 בלוקים</u> לכל היותר. <u>מדיניות ניהול ה-buffer cache היא LRU</u>. הניחו גם כי ניתן לייצג את המידע של כל קובץ טקסט על ידי בלוק יחיד. חשבו כמה גישות לדיסק תתבצענה בסך הכל כאשר תוכנית מסוימת פותחת כל אחד מן הקבצים הבאים וקוראת את תוכנו:

/a/b/c.txt
/a/b/d.txt
/e/f.txt
/a/b/c.txt

כתבו במדויק את החישוב ממנו נגזרת תשובתכם.

2. **(7 נק')** בסעיף זה הניחו כי ה- buffer cache יכול להכיל 3 בלוקים לכל היותר. מדיניות ניהול ה-buffer cache היא FIFO.

a. תארו את תוכנו של ה- buffer cache לאורך סדרת הבקשות הבאה לקריאת בלוקים (משמאל לימין) וחשבו כמה גישות לדיסק תתבצענה בסך הכל.

0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4

b. האם הגדלת מספר הבלוקים ב- buffer cache בבלוק נוסף (משלושה לארבעה) תקטין את מספר הגישות לדיסק עבור סדרת גישות זו? נמקו את תשובתכם.

ג. **(8 נק')** נתונות שתי מערכות הפעלה: האחת משתמשת ב-NTFS כאשר גודלה של כל רשומת MFT הוא 4K, והשניה משתמשת במערכת קבצים של Unix, מבוססת i-nodes. בשתי מערכות הקבצים, גודלו של כל בלוק הוא 1K.

בשאלה זו נניח לשם פשטות כי מערכות הקבצים אינן משתמשות ב-buffer cache או במנגנון caching אחר: כל גישה למבנה נתונים של מערכת הקבצים או לתוכנו של בלוק תגרום לגישה לדיסק.

הפונקציה Printfile מקבלת כפרמטר שם של קובץ ומדפיסה את תוכנו למסך. מריצים את הקוד הבא בשתי מערכת הפעלה אלו:

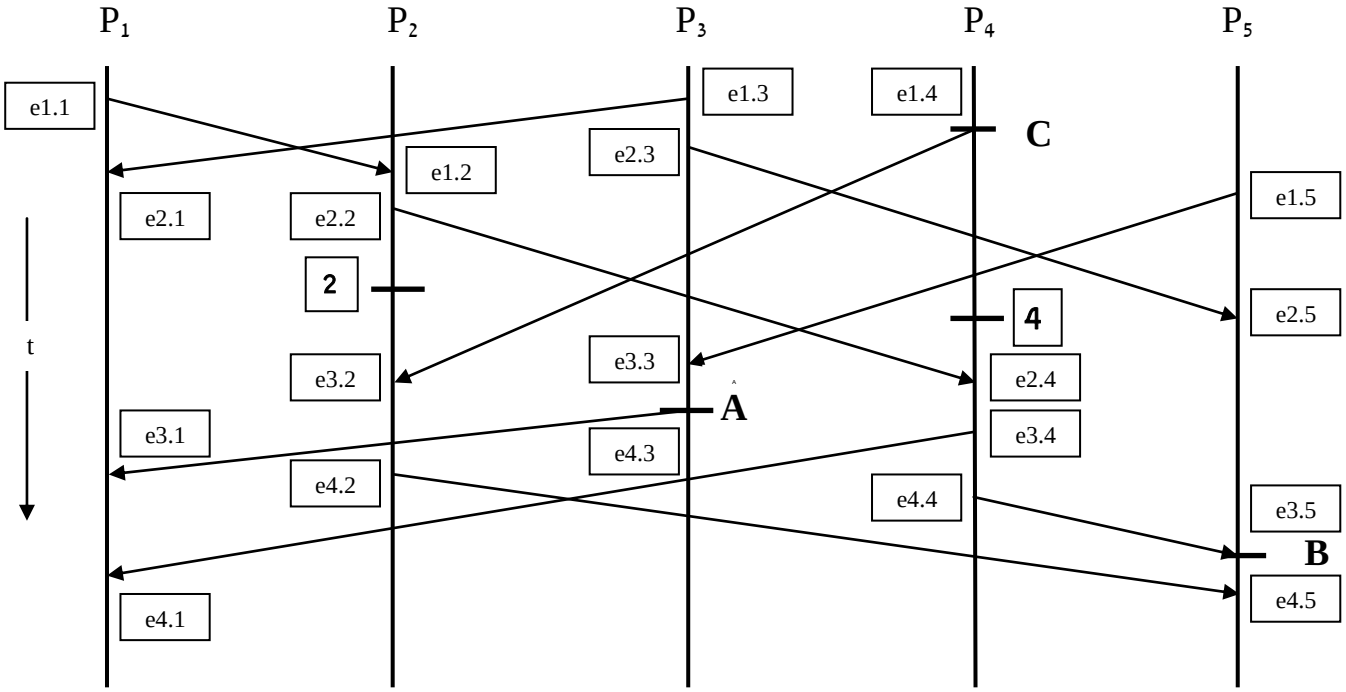
```
Printfile("\a.txt");
Printfile("\tmp\b.txt");
Printfile("\c.txt");
```

גודלם של הקבצים a.txt ו-b.txt הוא 500 בתים וגודלו של קובץ c.txt הוא 4500 בתים. נתון גם שגודל ה-root directory הוא 100 בתים וגודל ספרית \tmp הוא 20 בתים.

לכמה גישות לדיסק יגרמו שלוש הפקודות לעיל בכל אחת ממערכות הקבצים? כתבו במדויק את החישוב ממנו נגזרת תשובתכם. (יש להתעלם מן הגישות לדיסק הנגרמות מקריאת התוכנית Printfile עצמה.)

4. סינכרוניזציה מבוזרת ותיאום שעונים **(20 נק')**

נתונה הדיאגרמה הבאה של צירי הזמן של 5 סוכנים (Processors) $P_1...P_5$ ומשלוח ההודעות ביניהם. כל מאורע מסומן בסימון ייחודי (..e2.1, e3.4, etc)



א. מהם יחסי הזמן happens_before (אחרי, לפני, או בו-זמני) בין כל זוג מתוך שלוש הנקודות המסומנות **A, B, C**? נמקי מדוע היחסים הם כאלה (**8 נק'**).

ב. בנקודות הזמן **4** ו- **2** (המוקפות בריבוע) נקראים מצביהם של P_2 ו-P_4 כחלק מחישוב של snapshot. רשמי בעזרת סימוני המאורעות על קווי הזמן של הסוכנים P_1, P_3, P_5 את נקודות הזמן בהן ניתן לקרוא את מצב הסוכנים כך שיתקבל snapshot חוקי (כלומר consistent cut או consistent state). הרישום אומר, למשל, שנקודה 4 המסומנת בדיאגרמה נמצאת בין המאורעות e1.4 ו- e2.4. נמקי בפירוט (**12 נק'**).

בהצלחה!

פתרון

Q1 - Solution

a. K=2. Mutual exclusion is guaranteed because at most a single process can pass R. Starvation -freedom is guaranteed since at most a single process can wait or R at any given time, and so this process will become ready when R.up is done.

   For 3 processes, one may create a scenario in which 2 processes p1, p2 are waiting on R, when R.up is done by p3, p2 gets the signal and then, when p2 does R.up, p3 already waits on R and gets the signal, and so on, so p1 starves.

b. K=1 – mutual exclusion is violated.

c. K=3: At most a single process waits on R or S (on one ever waits on T). If k=4, then, once again, we may have 2 processes waiting on R and the scenario of a. shows that we may have starvation.

d. K=4. Initialize R=3, S=2, T=1, so there is mutual exlusion (since T=1) and there is no starvation since there is no semphore on which more than a single process may wait at any specific time.

Q2 – Solution:

a. The inclusion relationship $M(m, r) \subseteq M(m + l, r)$ is used to describe the set of pages in memory M which contains m frames when one accesses the r^{th} entry in the reference sting.
   The inclusion relationship states that all pages contained in the memory when accessing the r^{th} entry in the reference sting and using m frames are also present when using m+1 frames.
b. Belady's anomaly describes a situation where an increase in the number of frames m results in more page faults.
   Obviously, by the definition of a stack algorithm it is easy to see that when using a stack algorithm any page present in M(m,r) is also included in M(m+1,r). This means that for stack algorithms more frames will not cause more page faults.

c. LRU is a stack algorithm: at any point in time, the last m pages used are in memory and these are a subset of the last m+1 pages used. Hence the inclusion relation is trivially validated.
d. Gannt tables are used to answer this question:
    1. 11 page faults
    2. 8 pages
    3. Since the memory is empty at the beginning and since there are 8 different pages the minimal number of PF is 8 (which is OPT). Hence, can not be further improved.

Q3 - Solution

a. Presentaion – Files – slide 73.
b. 1. Calculate # of disk accesses for:
/a/b/c.txt
/a/b/d.txt
/e/f.txt
/a/b/c.txt
Cache has 8 cells and the policy is LRU.

- Each file/directory read requires 2 disk accesses, one for the i-node and another for the first data block.
- Root i-node and data block is **not** in memory.

/a/b/c.txt – Total of **8 disk accesses**.

/a/b/d.txt – All required data for /a/b/ is in the buffer cache. We need **2 disk accesses** in order to read d.txt into the buffer cache, one for its i-node and another for its first data block. I-node and data block of d will replace i-node and data blocks of c (according to LRU policy).

/e/f.txt – Root i-node and data block is in the buffer cache. We need **4 disk accesses** for e/f.txt which will replace i-nodes and data blocks of a and b.

/a/b/c.txt – Root i-node and data block is in the buffer cache. We need **6 more disk accesses** for a/b/c.txt.

We get total of 20 disk accesses.

2. a. Follow cache state after each read and calculate # of disk accesses for:

<2,0>, <2,1>, <2,2>, <2,3>, <2,0>, <2,1>, <2,4>,<2,0>, <2,1>, <2,2>, <2,3>, <2,4>

Cache has 3 cells and the policy is FIFO.

2,0	2,1	2,2	2,3	2,0	2,1	2,4	2,4	2,4	2,2	2,3	2,3
	2,0	2,1	2,2	2,3	2,0	2,1	2,1	2,1	2,4	2,4	2,4
		2,0	2,1	2,2	2,3	2,0	2,0	2,0	2,1	2,1	2,1
☺	☺	☺	☺	☺	☺	☺			☺	☺	

Total of **9 disk accesses**.

2.b. Do you suggest of adding 1 cell to the physical memory in order to save disk accesses? Explain.

2,0	2,1	2,2	2,3	2,3	2,3	2,4	2,0	2,1	2,2	2,3	2,4
	2,0	2,1	2,2	2,2	2,2	2,3	2,4	2,0	2,1	2,2	2,3
		2,0	2,1	2,1	2,1	2,2	2,3	2,4	2,0	2,1	2,2
			2,0	2,0	2,0	2,1	2,2	2,3	2,4	2,0	2,1
☺	☺	☺	☺			☺	☺	☺	☺	☺	☺

Total of **10 disk accesses. Belady's anomaly**. We won't benefit from increasing the cache size in 1 cell.

c. Calculate # of disk accesses caused by the following operations:

Printfile("\a.txt");
Printfile("\tmp\b.txt");
Printfile("\c.txt");

There is no buffer cache for this question.

UNIX (i-nodes) – Each read operation for file/directory requires 2 disk accesses, one for its i-node and one for its first data block. Therefore we will get total of 18 disk accesses, assuming root i-node and data is not in memory.

Printfile("\a.txt") – 4 disk accesses.

Printfile("\tmp\b.txt") – 6 disk accesses.
Printfile("\c.txt") – 2 disk accesses for root, i-node for c and 5 data blocks for c.

Total of 18 disk accesses.

WINDOWS (NTFS): Small files and directories get store its data inside its MFT record. We assume that root MFT is not in memory.

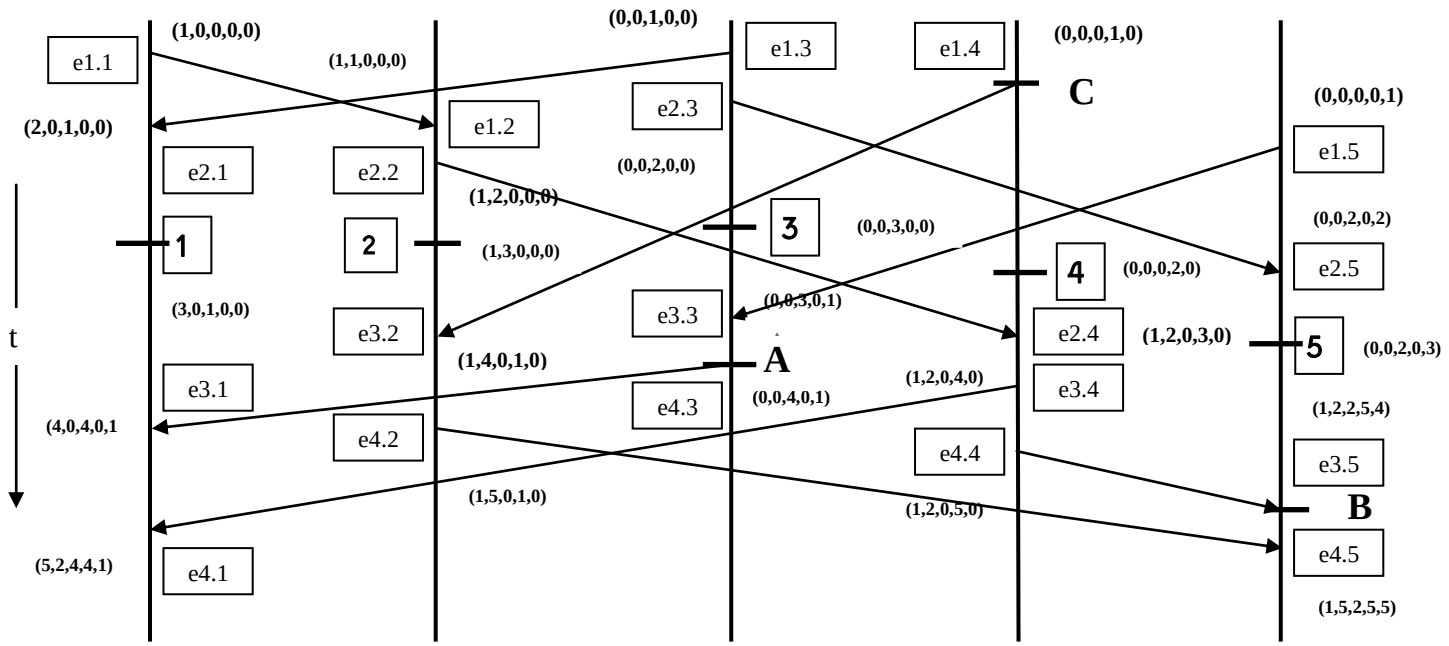
Printfile("\a.txt") – 2 disk accesses, one for root MFT and one for a.txt MFT record.
Printfile("\tmp\b.txt") – 3 disk accesses, one for root MFT and one for /tmp MFT record and
one for b.txt MFT record.
Printfile("\c.txt") – 7 disk accesses. One for root MFT, one for c MFT. C's MFT cannot contain its data (4500 bytes) and therefore contains its runs. Theses runs require reading 5 extra data blocks from disk.

Total of 12 disk accesses.

Q4 - Solution

4. לשני הסעיפים ניתן להשיב בעזרת חישוב של ה- Vector time-stamp של כל המאורעות שבדיאגרמה. על הסעיף הראשון אפשר גם לענות בקלות בעזרת בדיקה של קיום מסלולים (או אי-קיומם) בין המאורעות A, B, C.

א. A ו- B הם בו-זמניים, כיוון שאין מסלול המחבר ביניהם. קל לראות זאת כי A הוא המאורע האחרון בדיאגרמה על ציר הזמן של סוכן 3 השולח הודעה לסוכן 1 ומסוכן 1 אין הודעות אחר כך לסוכן 5. באופן דומה גם המאורעות C ו- A הם בו-זמניים. לעומת זאת, המאורע C קודם (happens_before) למאורע B. זאת כיוון שיש מסלול המוליך מ- C לאורך ציר הזמן של סוכן 4 עד למאורע e4.4 שהוא משלוח הודעה לסוכן 5 והגעתה הוא המאורע B (כלומר e3.5).

ב. נסתכל בתרשים הבא שבו סומנו כל ה- Vector time-stamps :

קודם כל ברור שהנקודות 2 ו- 4 הן בו-זמניות כיוון שהחותמות הוקטוריות הן (1,3,0,0,0) ו- (0,0,0,2,0) נסתכל עכשיו בצירי הזמן של יתר הסוכנים ונבחר נק' 1 לאחר e2.1 וזמנה (3,0,1,0,0), נק' 3 לאחר e2.3 שזמנה (0,0,3,0,0) ונק' 5 לאחר e2.5 שזמנה (0,0,2,0,3). אלה 5 מאורעות שחותמות הזמן הווקטוריות שלהן בו-זמניות ולכן מהוות מצב קונסיסטנטי.