חתל' נבחן

שם הקורס: מערכות הפעלה
קוד הקורס: 10363

הוראות לנבחן:
-חומר עזר שימושי לבחינה: כל חומר כתוב
-אין לכתוב בעפרון / עט מחיק
-אין להשתמש בטלפון סלולארי
-אין להשתמש במחשב אישי או נייד
-אין להשתמש בדיסק און קי ו/או מכשיר מדיה אחר
-אין להפריד את דפי שאלון הבחינה

בחינת סמסטר: קיץ
השנה: 2017
מועד: א

תאריך הבחינה: 16/10/17
שעת הבחינה: 17:00
משך הבחינה: 3 שעות

השאלון לא ייבדק בתום הבחינה ע"י המרצה

מרצה: גדי פסח

מבנה הבחינה והנחיות לפתרון:
בבחינה 6 שאלות. יש לענות על כולם.

משקלי השאלות
שאלה 1: 20 שאלה 2: 20 שאלה 3: 20 שאלה 4: 10 שאלה 5: 15
שאלה 6: 15

שאלה 1
מלכת אנגליה מגיעה לשאת נאום לרגל פתיחת מושב הפרלמנט. המסורת מחייבת שהמלכה תכנס לאולם בית הלורדים ראשונה וראש הממשלה ויושב ראש הפרלמנט (the speaker) יכנסו מאוחר יותר . שני האחרונים יכולים להכנס לאולם בסדר כלשהו.
לאחר סיום המושב, המלכה אמורה לעזוב את האולם **אחרי** ראש הממשלה וה-speaker (לא משנה הסדר בו הם עוזבים).
בקיצור: המלכה נכנסת ראשונה ויוצאת אחרונה. (הערה: תאור המסורת המובא כאן אינו מדויק).

כתבו מוניטור `stateOpeningofParliament` **שמבטיח שהמסורת תישמר.**

המוניטור צריך לכלול את הפרוצדורות הבאות:

```
enterQueen (), exitQueen (),
enterPrimeMinister (), exitPrimeMinister (),
enterSpeaker (), exitSpeaker ().
```

המלכה, ראש הממשלה וה- speaker מיוצגים ע"י שלושה חוטים.
החוט של המלכה קורא לפרוצדורה `enterQueen()` לפני שהיא נכנסת לאולם בית הלורדים וקוראת לפרוצדורה `exitQueen()` לפני שהיא עוזבת את האולם.

ראש הממשלה וה- speaker קוראים לפרוצדורות המתאימות להם לפני הכניסה והיציאה מהאולם.

הגדירו משתנים ומשתני תנאי של המוניטור לפי הצורך. יש לתת ערך התחלתי למשתנים הרגילים.
רמז: אפשר (אין חובה) להשתמש במשתנה queenStatus (מסוג int) שיש לו שלושה ערכים אפשריים:
NOT_ARRIVED_YET, ARRIVED, LEFT
(לא הגיעה עדיין, הגיעה, עזבה). משתנים דומים יוגדרו עבור ראש הממשלה וה- speaker.
הפרוצדורות שעליכם לכתוב כולן קצרות מאוד.

שאלה 2
נניח שב- i-node יש מקום **לעשרים** כתובות של data blocks (direct pointers) ובנוסף לכך כתובות של single indirect block אחד, double indirect block אחד ו- triple indirect block אחד.

בלוק הוא בגודל של 16KB. כתובת של בלוק על הדיסק היא בגודל 8 בתים.
נניח שתהליך רוצה לקרוא את בלוק מספר 6K + 24 בקובץ מסוים.
(הבלוק הראשון מספרו אפס). נניח עוד שמערכת ההפעלה מצאה את ה- i-node של הקובץ והביאה אותו לזכרון הראשי. תארו **בקצרה** מה עושה מערכת ההפעלה עכשיו כדי לאתר את הבלוק המבוקש על הדיסק. התאור צריך לכלול פרוט של מי הם הבלוקים שיובאו לזכרון הראשי ומה בדיוק מחפשים בכל בלוק כזה.

שאלה 3
השאלה עוסקת בזכרון וירטואלי. מרחב הכתובות של תהליך כולל שמונה דפים וירטואליים. ברגע מסוים טבלת הדפים של התהליך נראית כך (לא כל המידע מופיע כאן).

Virtual page #	Valid	Page frame #	Protection	Modified bit
0	1	58	read execute	0
1	0	on disk		
2	1	12	read write	1
3	0	on disk		
4	1	72	read write	0
5	1	82	read write	0
6	1	92	read write	0
7	0	on disk		

נניח שה TLB (Translation Lookaside Buffer) של המעבד כולל באותו רגע את הכניסות הבאות (הכניסה עם valid = 0 אינה בשימוש כרגע):

Valid	Virtual Page#	Modified bit	Protection	Page frame #
1	6	1	read write	92
1	4	0	read write	72
1	5	0	read write	82
0				

יש להניח שכרגע מסגרות פיזיות ... 101, 102, 103 הן פנויות ובמקרה הצורך דפים וירטואליים שיטענו מהדיסק -- יטענו לתוכם . נניח עוד שאם יש צורך לפנות כניסה ב- TLB אז מפנים קודם את הכניסה הראשונה, לאחר מכן את השניה וכן הלאה.
הניחו גם (לצורך השאלה בלבד ...) שכל דף שנטען מהדיסק הוא בעל הרשאות קריאה וכתיבה.

מה יהיו השינויים (אם יהיו) בטבלת הדפים וב- TLB כאשר יתבצעו הגישות הבאות לזכרון (בסדר זה):
סעיף א: כתיבה לדף 0.
סעיף ב: קריאה מדף 6
סעיף ג: קריאה מדף 3

שאלה 4

This question is about safe states (like in the Banker's Algorithm). A system has four processes and four allocatable resources. The current allocation and resources still needed are as follows:

	Allocated	additional resources needed
Process A	0,1,0,1	4,0,0,7
Process B	1,4,0,3	1,2,0,4
Process C	1,0,0,3	2,1,0,0
Process D	1,0,1,1	3,5,0,16

Available Resources = (1, 2, 0, x)
What is the smallest value of x for which this is a safe state ? Give a **short** explanation.

שאלה 5

נניח שדרייבר עבור דיסק פועל לפי אלגוריתם המעלית (לא מעגלי). הזרוע כרגע נמצאת בצילינדר x. מגיעות בקשות לקריאה מצילינדרים a, b, c, d כך שמתקיים $a < b < c < d$
$a < x < d$
רשמו תנאי פשוט ככל האפשר (ואפשר ...) לכך שבמקרה זה יעיל יותר יהיה אם הזרוע תנוע תחילה למעלה (לכיוון צילינדרים עם מספרים גבוהים) מאשר אם תנוע תחילה למטה.

שאלה 6

השאלה עוסקת במימוש מנעולים בעזרת פקודה אטומית שנקראית `assign_if_greater_than`.
מנעול מיוצג כך:

```
typedef struct {
    int flag;
} Lock;
```

עליכם לכתוב את הפונקציות `acquire (Lock *lock)` ו- `release (Lock *lock)`

יש לעשות שימוש בפקודה האטומית assign_if_greater_than המוגדרת כך:

```
boolean assign_if_greater_than (int *address,
                                int lower_bound,
                                int new_value)
{
     int old_value = *address; /* get old value */
     if (old_value > lower_bound) {
          *address = new_value;/* store new value */
          return TRUE;
     }
     return FALSE;
}
```

לפקודה יש שלושה אופרנדים: כתובת בזיכרון, ערך שהוא "חסם תחתון" וערך חדש.

אם הערך המאוחסן בכתובת גדול מהחסם התחתון אז הערך החדש מאוחסן בכתובת.

אחרת אין שינוי בערך המאוחסן בכתובת. במקרה הראשון מוחזר TRUE. במקרה השני מוחזר FALSE.

בהצלחה!

עמוד 4

כל הזכויות שמורות ©. מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקליט, לשדר, לאחסן מאגר מידע, בכל דרך שהיא, בין מכאנית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה

GUIDE

פתרון בחינה במערכות הפעלה סמסטר 2017 קיץ מועד X

שאלה 1

מלכת אנגליה מגיעה לשאת נאום לרגל פתיחת מושב הפרלמנט. המסורת מחייבת שהמלכה תכנס לאולם בית הלורדים ראשונה וראש הממשלה ויושב ראש הפרלמנט (the speaker) יכנסו מאוחר יותר . שני האחרונים יכולים להכנס לאולם בסדר כלשהו.

לאחר סיום המושב, המלכה אמורה לעזוב את האולם **אחרי** ראש הממשלה וה-speaker (לא משנה הסדר בו הם עוזבים).

בקיצור: המלכה נכנסת ראשונה ויוצאת אחרונה. (הערה: תאור המסורת המובא כאן אינו מדויק).

כתבו מוניטור stateOpeningofParliament שמבטיח שהמסורת תישמר.

המוניטור צריך לכלול את הפרוצדורות הבאות:

```
enterQueen (), exitQueen (),
enterPrimeMinister (), exitPrimeMinister (),
enterSpeaker (), exitSpeaker ().
```

המלכה, ראש הממשלה וה- speaker מיוצגים ע"י שלושה חוטים.

החוט של המלכה קורא לפרוצדורה enterQueen() לפני שהיא נכנסת לאולם בית הלורדים וקוראת לפרוצדורה exitQueen() לפני שהיא עוזבת את האולם.

פתרון וועד X
10303

ראש הממשלה וה- speaker קוראים לפרוצדורות המתאימות להם לפני הכניסה והיציאה מהאולם.

הגדירו משתנים ומשתני תנאי של המוניטור לפי הצורך. יש לתת ערך התחלתי למשתנים הרגילים.

רמז: אפשר (אין חובה) להשתמש במשתנה queenStatus (מסוג int) שיש לו שלושה ערכים אפשריים:

NOT_ARRIVED_YET, ARRIVED, LEFT

(לא הגיעה עדיין , הגיעה, עזבה). משתנים דומים יוגדרו עבור ראש הממשלה וה- speaker.

הפרוצדורות שעליכם לכתוב כולן קצרות מאוד.

פתרון

```
monitor  StateOpeningOfParliament {

    int  queenStatus = NOT_ARRIVED_YET;

    int  primeMinisterStatus = NOT_ARRIVED_YET;

    int speakerStatus = NOT_ARRIVED_YET;

    condition ok_to_enter;   /* ok for  speaker and
primeMinister to enter */

    condition ok_to_exit;  /* ok for  queen to exit */


    void enterQueen ()

    {

       queenStatus = ARRIVED;

       broadcast (ok_to_enter);
```

```
}
void exitQueen ()
{
   while (primeMinisterStatus != LEFT  ||
        speakerStatus !=  LEFT)
            wait (ok_to_leave);
   queenStatus = LEFT;
 }

 void enterPrimeMinister ()
 {
   while (queenStatus == NOT_ARRIVED_YET)
       wait (ok_to_enter);
  primeMinisterStatus = ARRIVED;
 }
 void exitPrimeMinister ()
 {
    primeMinisterStatus = LEFT;
    if (speakerStatus == LEFT)
        signal (ok_to_leave);
 }

void enterSpeaker ()
{
    while (queenStatus == NOT_ARRIVED_YET)
```



```
        wait (ok_to_enter);
    speakerStatus = ARRIVED;
}
void exitSpeaker ()
{
    speakerStatus = LEFT;
    if (primeMinisterStatus == LEFT)
        signal (ok_to_leave);
    }
```

שאלה 2

נניח שב- i-node יש מקום **לעשרים** כתובות של data blocks (direct pointers) ובנוסף לכך כתובות של single indirect block אחד, double indirect block אחד ו- triple indirect block אחד.

בלוק הוא בגודל של 16KB. כתובת של בלוק על הדיסק היא בגודל 8 בתים.

נניח שתהליך רוצה לקרוא את בלוק מספר 6K + 24 בקובץ מסוים.

(הבלוק הראשון מספרו אפס). נניח עוד שמערכת ההפעלה מצאה את

ה- i-node של הקובץ והביאה אותו לזכרון הראשי. תארו **בקצרה** מה עושה מערכת ההפעלה עכשיו כדי לאתר את הבלוק המבוקש על הדיסק. התאור צריך לכלול פרוט של מי הם הבלוקים שיובאו לזכרון הראשי ומה בדיוק מחפשים בכל בלוק כזה.

פתרון

את הכתובות של 20 הבלוקים הראשונים ניתן למצוא ב- i-node עצמו.

את הכתובות של 2K הבלוקים הבאים ניתן למצוא ב- single indirect block שכתובתו נמצאת ב- inode. (מאחר והגודל של בלוק הוא 16KB וכתובת היא בגודל 8 בתים אז בכל indirect block יש מקום

ל- 16K/ 8 = 2K כתובות).

כדי להגיע ל- $2K * 2K = 4* K^2$ הבלוקים הבאים יש להשתמש

ב- double indirect block שכתובתו נמצאת ב- inode.

כל כניסה ב- double indirect block זה מצביעה ל- single indirect block המכיל 2K כתובות של data blocks.

הכניסה הראשונה ב- double indirect block מובילה לבלוקים

בעלי מספרים 2K+20 עד 4K + 19.

הכניסה השניה מובילה לבלוקים בעלי מספרים 4K + 20 עד 6K + 19

הכניסה השלישית מובילה לבלוקים 6K + 20 עד 8K + 19.

הבלוק המבוקש (שמספרו 6K + 24) הוא הבלוק החמישי בקבוצה

של הכניסה השלישית.

לכן אלו השלבים באיתור הבלוק המבוקש:

מערכת ההפעלה מביאה את ה-double indirect block (שכתובתו נמצאת ב - i-node) מהדיסק לזכרון הראשי,

היא קוראת את הכתובת השלישית בבלוק הזה ומביאה מהדיסק את ה- single indirect block שזו כתובתו. הכתובת החמישית ב- single indirect block שזה עתה הובא מהדיסק היא כתובתו של הבלוק המבוקש.

שאלה 3

השאלה עוסקת בזכרון וירטואלי. מרחב הכתובות של תהליך כולל שמונה דפים וירטואליים. ברגע מסוים טבלת הדפים של התהליך נראית כך (לא כל המידע מופיע כאן).

Virtual page #	Valid	Page frame #	Protection	Modified bit
0	1	58	read execute	0
1	0	on disk		
2	1	12	read write	1
3	0	on disk		
4	1	72	read write	0
5	1	82	read write	0
6	1	92	read write	0
7	0	on disk		

נניח שה TLB (Translation Lookaside Buffer) של המעבד כולל באותו רגע את הכניסות הבאות (הכניסה עם valid = 0 אינה בשימוש כרגע):

Valid	Virtual Page#	Modified bit	Protection	Page frame #
1	6	1	read write	92
1	4	0	read write	72
1	5	0	read write	82
0				

יש להניח שכרגע מסגרות פיזיות ... 101, 102, 103 הן פנויות ובמקרה הצורך דפים וירטואליים שיטענו מהדיסק -- יטענו לתוכם . נניח עוד שאם יש צורך לפנות כניסה ב- TLB אז מפנים קודם את הכניסה הראשונה, לאחר מכן את השניה וכן הלאה.

הניחו גם (לצורך השאלה בלבד ...) שכל דף שנטען מהדיסק הוא בעל הרשאות קריאה וכתיבה.

מה יהיו השינויים (אם יהיו) בטבלת הדפים וב- TLB כאשר יתבצעו הגישות הבאות לזכרון (בסדר זה):

סעיף א: כתיבה לדף 0.

סעיף ב: קריאה מדף 6

סעיף ג: קריאה מדף 3

פתרון

סעיף א. כתיבה לדף 0. זה TLB miss.

ל- TLB תטען שורה חדשה:

Valid = 1, Virtual page = 0, Modified = 0, Protection = read execute,

Page frame number = 58

סעיף ב. קריאה מדף 6

זה TLB hit. אין שינוי בטבלת הדפים או ב- TLB.

סעיף ג. קריאה מדף 3.

זה יגרום ל- page fault שבעקבותיו יטען הדף לזכרון והכניסה המתאימה בטבלת הדפים תהיה:

Valid = 1, Virtual page = 3, Page frame = 101, Protection = read write, Modified = 0,

הכניסה תטען גם ל- TLB לכניסה הראשונה:

Valid = 1 Virtual page = 3, Modified = 0, Protection = read write, Page Frame = 101.

בכניסה בטבלת הדפים של דף וירטואלי מספר 6 (שיפונה מה- TLB כדי לפנות מקום לדף 3) ה- Modified bit תעודכן ל- 1.

שאלה 4

This question is about safe states (like in the Banker's Algorithm). A system has four processes and four allocatable resources. The current allocation and resources still needed are as follows:

	Allocated	additional resources needed
Process A	0,1,0,1	4,0,0,7
Process B	1,4,0,3	1,2,0,4
Process C	1,0,0,3	2,1,0,0
Process D	1,0,1,1	3,5,0,16

Available Resources = (1, 2, 0, x)

What is the smallest value of x for which this is a

safe state ? Give a **short** explanation.

פתרון

אם $x >= 4$ אז B יוכל לרוץ אחרת נקבל קפאון (deadlock).

אחרי ש- B רץ ומחזיר את המשאבים שברשותו:

Available = (2, 6, 0, (x +3))

עכשיו C רץ ומחזיר את המשאבים שברשותו:

Available = (3, 6, 0, (x+6))

כדי להמנע מקפאון, x צריך להיות לפחות 10 כדי ש- D יוכל לרוץ. אחרת A ו- D בקפאון. אחרי ש- D רץ:

Available = (4, 6, 1, (x+7))

עכשיו A יוכל לרוץ ואין קפאון.

התשובה היא: הערך המינימלי של x עבורו המצב בטוח הוא 10.

שאלה 5

נניח שדרייבר עבור דיסק פועל לפי אלגוריתם המעלית (לא מעגלי). הזרוע כרגע נמצאת בצילינדר x. מגיעות בקשות לקריאה מצילינדרים a, b, c, d

כך שמתקיים $a < b < c < d$

$a < x < d$

רשמו תנאי פשוט ככל האפשר (ואפשר ...) לכך שבמקרה זה יעיל יותר יהיה אם הזרוע תנוע תחילה למעלה (לכיוון צילינדרים עם מספרים גבוהים) מאשר אם תנוע תחילה למטה.

פתרון

אם הזרוע נעה תחילה למעלה עליה לעבור בסך הכל מרחק (בצילינדרים)

$(d - x) + (d - a)$

אם היא נעה תחילה למטה עליה לעבור בסך הכל מרחק

$(x - a) + (d - a)$

התנאי המבוקש הוא אם כן $(d - x) + (d - a) < (x - a) + (d - a)$

זה שקול לתנאי $(d - x) < (x - a)$ כלומר הזרוע קרובה יותר לצילינדר d

מאשר לצילינדר a.

שאלה 6

השאלה עוסקת במימוש מנעולים בעזרת פקודה אטומית שנקראית

`assign_if_greater_than`.

מנעול מיוצג כך:

```
typedef struct {
```

```
    int flag;
} Lock;
```

עליכם לכתוב את הפונקציות acquire (Lock *lock) -ו

release (Lock *lock)

יש לעשות שימוש בפקודה האטומית assign_if_greater_than המוגדרת כך:

```
boolean assign_if_greater_than (int *address,
                                int lower_bound,
                                int new_value)
{
    int old_value = *address; /* get old value
*/
    if (old_value > lower_bound) {
        *address = new_value;/* store new value
*/
        return TRUE;
    }
    return FALSE;
}
```

לפקודה יש שלושה אופרנדים: כתובת בזיכרון, ערך שהוא "חסם תחתון" וערך חדש.

אם הערך המאוחסן בכתובת גדול מהחסם התחתון אז הערך החדש מאוחסן בכתובת.

אחרת אין שינוי בערך המאוחסן בכתובת. במקרה הראשון מוחזר TRUE.

במקרה השני מוחזר FALSE.

<u>פתרון</u>

```
acquire (Lock *lock)
{
    while (assign_if_greater_than (&lock->flag,
            10, 1) == FALSE)
              ; /* spin wait
}
```

```
release (Lock *lock)
{
    lock->flag = 11;
}
```

כאן כשל- flag יש ערך 1 אז המנעול תפוס. כשל- flag יש ערך 11 המנעול פנוי. המספרים האלו נבחרו באופן שרירותי אבל "הערך הפנוי" צריך להיות גדול מ-"הערך התפוס" כי כך נוח להשתמש בפקודה האטומית הנתונה.