|  |  |
| --- | --- |
| **C:\Users\samitz\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\JWHB1H5S\לוגו עבור דפוס -עברית.png** | **המסלול האקדמי המכללה למינהל**  **בית הספר מדעי המחשב** |

**ת.ז הסטודנט:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**מס' חדר:\_\_\_\_\_\_\_\_ מס' נבחן:\_\_\_\_\_\_\_**

**מבחן בקורס: מערכות הפעלה**

**קוד נושא: 655002**

**תאריך הבחינה: 05.07.19 שעה 09:00**

**שנה"ל: תשע"ט סמסטר: ב' מועד: א'**

**שם המרצה: דר' איגור רוכלין**

**שם המתרגלים: מר' חיים שפיר, מר' יורם סגל**

**משך הבחינה: 3 שעות**

**מספר חלקים בבחינה: 2**

**הוראות לנבחנים:**

* **במבחן 2 חלקים:**

**חלק א' –** שאלות פתוחות – 2 שאלות פתוחות (בסה"כ בחלק זה 40 נקודות)

1. משקל כל שאלה מצוין בגוף השאלה
2. יש לענות על כל שאלה **במחברת נפרדת** ולציין בדף שער כל מחברת את כל הפרטים הנדרשים שם
3. יש לנמק בפירוק את תשובותיך. על תשובה לא מפורטת יורדו נקודות.

**חלק ב' –** שאלות ברירה – 12 שאלות ברירה (בסה"כ בחלק זה 60 נקודות)

1. משקל כל שאלה – 5 נקודות
2. יש למלא **בכתב יד ברור** במקומות המיועדים **בחציו הימני של דף הקידוד** את שם ביה"ס, חדר המבחן, מספר הנבחן, שם הקורס, תאריך הבחינה, שם המרצה, מספר תעודת הזהות (מספר בן תשע ספרות, כולל ספרת ביקורת ועם אפס מקדים באם נדרש) ואת מספר השאלון (המופיע בצידו השמאלי העליון של השאלון)
3. **\*\*\* חשוב מאוד**:

**בדף הקידוד יש לרשום ולקדד את מספר השאלון מימין לשמאל (להוסיף אפסים משמאל במידת הצורך).**

1. בכל שאלה יש לבחור את התשובה הנכונה ביותר ולסמנה במקום המיועד בצידו השמאלי של דף הקידוד, **בעט שחור או כחול בלבד ובאופן ברור ומודגש**
2. אין לסמן את התשובות על גבי דף הקידוד במַדְגֵּשׁ (מַרְקֶר) זוהר!
3. רק דף הקידוד ייבדק!

- יש לענות על כל השאלות

- אין שימוש בחומר עזר

- מותר שימוש במחשבון כיס

- נדרש להחזיר את השאלון

**בהצלחה !!!**

**חלק א' – שאלות פתוחות (40 נקודות)**

**שאלה פתוחה 1 (20 נקודות)**

הנח שפעולת Fetch\_and\_add(\*p,inc) היא פונקציה **אטומית** אשר קוראת את הערך שבכתובת p בזיכרון מקדמת אותו ב-inc ומחזירה את הערך שנמצא בכתובת p **לפני השינוי**.

כלומר:

Fetch\_and\_add(\*p,inc){  
 val=\*p;  
 \*p=val+inc  
 return val;   
}

בהינתן הפתרון הבא לבעיית הקטע הקוד הקריטי אשר משתמש במשתנה lock משותף לתהליכים שמאותחל ל-0:

while(1){  
 **[None Critical section]**

while(Fetch\_and\_lock(&lock,1)!=0){

lock=1;

}

**[Critical section]**

Lock = 0;

}

1. (6 נק') האם האלגוריתם מספק את קריטריון Mutual exclusion? נמקו תשובתכם.
2. (6 נק') האם האלגוריתם מספק Starvation Freedom? נמקו תשובתכם.
3. (5 נק') האם האלגוריתם מספק Deadlock Freedom? נמקו תשובתכם.
4. (3 נק') האם האלגוריתם סובל מ-Busy-waiting? נמקו תשובתכם.

**פתרון:**

1. כן**:**

Assume that two threads are in the critical section at the same time, then they both got 0 from Fetch&Add but Fetch&Add increments the value of lock and it is atomic and that's a contradiction to one of these terms.

1. לא:

Assume p1 and p2 are running, p1 enters the critical section and when it leaves the critical section just before it changes the value of lock to zero, we have a context switch to p2 which enters the while loop. Then we have another context switch, p1 sets lock to zero. Then another context switch and then p2 sets lock to 1. From that point on – all processes will be starved.

1. כן:

There is only one resource hence we cannot fulfill the condition of circular wait. Two or more threads cannot wait for each other while each thread is holding a resource that the other one needs. Therefore, the algorithm provides deadlock freedom.

1. כן:

Yes, the implementation of waiting by using a "do nothing" loop like the internal while loop is busy waiting.

**שאלה פתוחה 2 (20 נקודות)**

1. (10נק') כיצד פועל מנגנון File-allocation table (FAT)? מהם היתרונות והחסרונות העיקריים של השיטה?

**תשובה**:

שיפור של השיטה Linked Allocation. בניגוד ל Linked Allocation - מחזיקים את שרשרת המצביעים בנפרד, בטבלה הנקראת File Allocation Table (FAT).

הטבלה מחזיקה צילום של הדיסק. לכל בלוק בדיסק תהייה כניסה בטבלה, והכניסה תכיל את כתובת הבלוק הבא בתור. כניסה עם ערך "1-" יציין בלוק אחרון בקובץ (EOF). הטבלה שמורה בדיסק קשיח (hard disk), אך בזמן עבודה הטבלה ניטענת לזיכרון (RAM) ולכן חיפוש של בלוק בטבלה מתבצע בזיכרון ורק אז מבצעים גישה ישירה אל הבלוק ב hard disk.

יתרונות:

* חיפוש יותר יעיל מ Linked Allocation – מחפשים את הבלוק בטבלה ואז מבצעים גישה ישירה אל הבלוק ב hard disk.
* ללא External Fragmentation.
* קובץ יכול לגדול.

חסרונות:

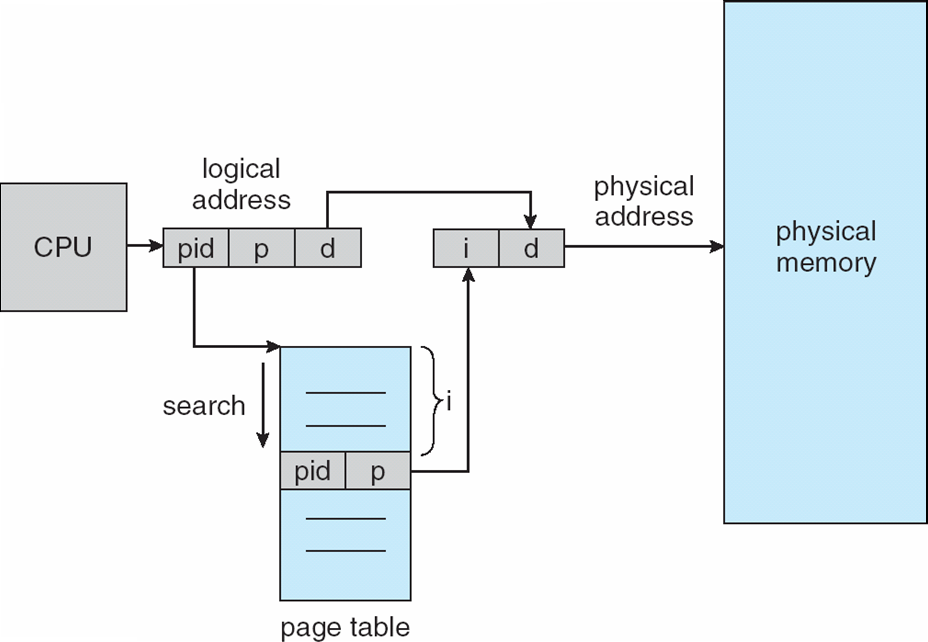
* פגיעה בטבלה גורמת לאובדן של כל המידע.
* בזבוז מקום.
* ניהול קשה יותר – עבור כל עדכון בטבלה צריך לא לשכוח לעדכן גם את הטבלה בדיסק קשיח.

1. (10נק') כיצד פועל מנגנון Inverted Page Tables וכיצד מתורגמת הכתובת הלוגית לפיזית באמצעותו (מומלץ לצרף להסבר סכמה מתאימה)?

**תשובה:**

The simplest form of an inverted page table contains one entry per each frame in a linear array. Since the table is shared, each entry must contain the process ID of the page owner and since physical pages are now mapped to virtual, each entry contains a virtual page number instead of a physical. The physical page number is not stored, since the index in the table corresponds to it.

In order to translate a virtual address, the virtual page number and current process ID are compared against each entry, traversing the array sequentially. When a match is found, the index of the match replaces the virtual page number in the address to obtain a physical address. If no match is found, a page fault occurs. This translation procedure is shown in figure. While the table size is small, the lookup time for a simple inverted page table can be very large.



**חלק ב' – שאלות ברירה (60 נקודות)**

לפניך 12 שאלות ברירה. בכל שאלה עליך לסמן את התשובה הנכונה מבין 4 האפשרויות הרשומות.

1. מהו העיקרון, במערכות הפעלה, על-בסיסו פועל ה- working-set?

תשובה: עיקרון ה- locality

2. מהי internal fragmentation:

תשובה: כאשר הזיכרון המוקצה לתהליך גדול מהנצרך.

3. גידול ב size של FRAME גורם:

תשובה: להגדלת internal fragmentation

4. הקטנה של גודל ה- FRAME גורם:

תשובה: להגדלת table page

5. נתונים ארבעה תהליכים A, B, C, D כאשר זמני הגעתם, בקשות ה- CPU Bursts שלהם וה- priorities שלהם מפורטים בטבלה להלן. ככל שהערך המצוי בטור priority נמוך יותר ה- priority שלו גבוה יותר.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A | תהליך |
| 6 | 4 | 3 | 11 | Burst |
| 2 | 1 | 4 | 3 | Priority |
| 4 | 3 | 1 | 0 | Arrival Time |

הנח כי ה- context switch overhead זניח. מהו ה- waiting time הממוצע כאשר מערכת משתמשת באלגוריתם First come first served?

תשובה: 8.75

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 18-24 | 14-18 | 11-14 | 0-11 | **timeline** |
| D | C | B | A | **Process** |

Waiting time = (0+(11-1)+(14-3)+(18-4))/4=(0+10+11+14)/4=8.75

6. נתונים ארבעה תהליכים A, B, C, D כאשר זמני הגעתם, בקשות ה- CPU Bursts שלהם וה- priorities שלהם מפורטים בטבלה להלן. ככל שהערך המצוי בטור priority נמוך יותר ה- priority שלו גבוה יותר.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A | תהליך |
| 6 | 4 | 3 | 11 | Burst |
| 2 | 1 | 4 | 3 | Priority |
| 4 | 3 | 1 | 0 | Arrival Time |

הנח כי ה- context switch overhead זניח. מהו ה- waiting time הממוצע כאשר מערכת משתמשת באלגוריתם Priority scheduling - עם preemption?

תשובה: 8.25

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 21-24 | 13-21 | 7-13 | 3-7 | 0-3 | **timeline** |
| Brest=0 | Arest=0 | Drest=0 | Crest=0 | Arest=8 | **Process** |

Waiting time = ((21-11-0)+(24-3-1)+(7-4-3)+(13-6-4))/4=(10+20+0+3)/4=8.25

7. נתונים ארבעה תהליכים A, B, C, D כאשר זמני הגעתם, בקשות ה- CPU Bursts שלהם וה- priorities שלהם מפורטים בטבלה להלן. ככל שהערך המצוי בטור priority נמוך יותר ה- priority שלו גבוה יותר.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A | תהליך |
| 6 | 4 | 3 | 11 | Burst |
| 2 | 1 | 4 | 3 | Priority |
| 4 | 3 | 1 | 0 | Arrival Time |

הנח כי ה- context switch overhead זניח. מהו ה- waiting time הממוצע כאשר מערכת משתמשת באלגוריתם Shortest job first - עם preemption?

תשובה: 4.5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 14-24 | 8-14 | 4-8 | 3-4 | 1-3 | 0-1 | **timeline** |
| Arest=0 | Drest=0 | Crest=0 | Brest=1 | Brest=2 | Arest=10 | **Process** |

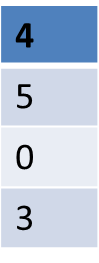
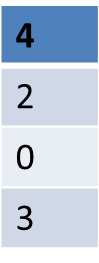
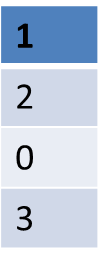
Waiting time = ((24-11-0)+(4-3-1)+(8-4-3)+(14-6-4))/4=(13+0+1+4)/4=4.5

8. נתונה סדרת הקריאות הבאה ל – PAGES של תהליך מסוים (משמאל לימין):

1,1,2,0,1,3,2,3,4,3,5,1,2,5

במקרה שלתהליך זה מוקצים 4 FRAMES, כמה PAGE FAULTS יהיו אם ההחלפות מתבצעות על-פי FIFO (First-In-First-Out)?

תשובה: 8



4PF 5PF 6PF 7PF 8PF

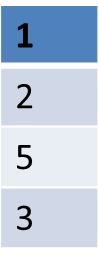
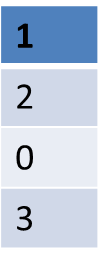
Number of page faults is 8

9. נתונה סדרת הקריאות הבאה ל – PAGES של תהליך מסוים (משמאל לימין):

1,1,2,0,1,3,2,3,4,3,5,1,2,5

במקרה שלתהליך זה מוקצים 4 FRAMES, כמה PAGE FAULTS יהיו אם ההחלפות מתבצעות על-פי LFU ([Least Frequently Used](http://en.wikipedia.org/wiki/Least_frequently_used))?

תשובה: 6



4PF 5PF 6PF

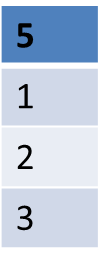
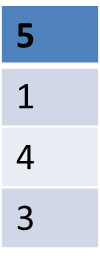
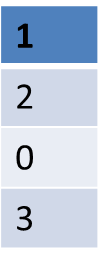
Number of page faults is 6

10. נתונה סדרת הקריאות הבאה ל – PAGES של תהליך מסוים (משמאל לימין):

1,1,2,0,1,3,2,3,4,3,5,1,2,5

במקרה שלתהליך זה מוקצים 4 FRAMES, כמה PAGE FAULTS יהיו אם ההחלפות מתבצעות על-פי LRU (Least Recently Used)?

תשובה: 8



4PF 5PF 6PF 7PF 8PF

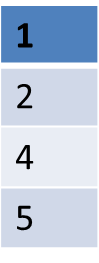
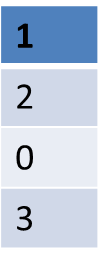
Number of page faults is 8

11. נתונה סדרת הקריאות הבאה ל – PAGES של תהליך מסוים (משמאל לימין):

1,1,2,0,1,3,2,3,4,3,5,1,2,5

במקרה שלתהליך זה מוקצים 4 FRAMES, כמה PAGE FAULTS יהיו אם ההחלפות מתבצעות על-פי Optimal Algorithm?

תשובה: 6



4PF 5PF 6PF

Number of page faults is 6

12. נתון דיסק בו הצילינדרים מאונדקסים מ - 0 ל - 100. הראש הקורא-כותב נמצא בצילינדר 15 ונע כרגע לכיוון cylinder 100. נתונה סדרת הקריאות הבאה (משמאל לימין):

1,5,2,99,25,40,10

מהו המרחק (מספר הצילינדרים) שתצטרך הזרוע לעבור, בהנחה שהאלגוריתם בו עושים שימוש הוא

C-SCAN?

תשובה: 195

Sequence: 15->25->40->99->100->0->1->2->5->10

TIME= 10+15+59+1+100+1+1+3+5=195