

מדבקה

האוניברסיטה העברית בירושלים
ביה"ס להנדסה ומדעי המחשב

מבחן במערכות הפעלה

קורס מס' 67808

תאריך: 26.6.11
זמן: 2.5 שעות

מועד א' תשע"א
המורה: פרופ' דרור פייטלסון

במבחן שני חלקים. בחלק הראשון יש לענות על 11 מתוך 12 שאלות. משקל כל שאלה 5 נקודות. בחלק השני יש לענות על 3 מתוך 4 שאלות, שמשקל כל אחת מהן 15 נקודות. ניתן להשתמש בצד האחורי של הדף לטיוטה. יש לכתוב את התשובות הסופיות בעט בטופס המבחן בצורה נקייה ומסודרת.

בהצלחה!

חלק א' (55 נקודות)

ענה על 11 מתוך 12 השאלות הבאות. בכל שאלה יש להקיף בעיגול את התשובה הנכונה. נא לסמן באופן ברור את השאלה שאין לבדוק על ידי מתיחת קו אלכסוני על כל השאלה.

1. בקרנל של Linux קיימת הפונקציה הבאה:

```
double_rq_lock(struct rq *rq1, struct rq *rq2){  
    if (rq1 < rq2) {  
        spin_lock(&rq1->lock);  
        spin_lock(&rq2->lock);  
    } else {  
        Spin_lock(&rq2->lock);  
        Spin_lock(&rq1->lock);  
    }  
}
```

יש שני מבני נתונים מסוג `rq`, שיש בהם מנעול, ומשתמשים בכתובות שלהם כדי שתמיד הם ינעלו באותו סדר.

מה הסיבה לכתיבת הפונקציה הזו?

- (א) כדי להקטין את זמן ההמתנה הממוצע לנעילה
- (ב) כדי למנוע מצב של היפוך עדיפויות בו תהליך בעדיפות גבוהה מחכה למנעול המוחזק על ידי תהליך בעדיפות נמוכה
- (ג) כדי לאפשר למתזמן (scheduler) לתזמן את התהליכים לפי סדר עדיפויות נכון
- (ד) כדי למנוע deadlock

2. כשממפים דף לטבלת דפים ניתן לסמן אותו כדף לקריאה בלבד, ואז כל ניסיון לכתוב לדף יגרום לפסיקה.

סיבה אפשרית להשתמש במנגנון זה היא

- (א) להגן על דפי קוד מפני שינוי בטעות
- (ב) לממש אופטימיזציה של העתקת זכרון על ידי שכפול המיפוי בלבד, וביצוע ההעתקה בפועל רק אם מנסים לשנות את אחד העותקים
- (ג) לאפשר לתהליך משתמש לקרוא נתונים של מערכת ההפעלה מבלי שיוכל לשנות אותם
- (ד) כל התשובות נכונות

3. ה-inode הוא מבנה נתונים המייצג קובץ במערכת Unix. איזה מפרטי המידע הבאים אינו שדה ב-inode?

- (א) הגודל של הקובץ
- (ב) למי שייך הקובץ
- (ג) השם של הקובץ
- (ד) מצביע לבלוק של מצביעים לבלוקים של הקובץ

4. מה יכולה להיות סיבה לכך שלתהליך יהיו יותר מ-4 סגמנטים של זכרון?

- (א) התהליך פתח קובץ לקריאה והקצה חוצץ (buffer) לשם כך
- (ב) התהליך מכיל מספר חוטים (threads)
- (ג) יש פרגמנטציה
- (ד) לתהליך נגמר הזכרון והוא צריך להקצות זכרון נוסף

לכל אחד יהיה סגמנט מחסנית משלו.

5. חסרון חשוב של עיצוב מערכת הפעלה במבנה של micro-kernel הוא

- (א) חוסר המודולאריות שזה גורר
- (ב) חוסר האפשרות לגמישות במימוש מדיניות ניהול משאבים
- (ג) שזה מקטין את אמינות המערכת
- (ד) התקורה הנובעת מכך שפעולות בסיסיות דורשות טיפול על ידי שרת חיצוני

6. אם משתמשים באלגוריתם first fit להקצאת זיכרון רציף, לא יתכן שלקטע האחרון שהוקצה תהיה התכונה הבאה:

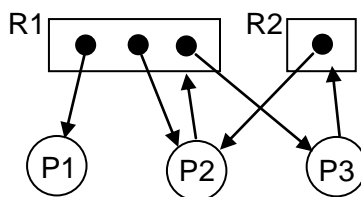
- (א) יש קטע בלתי מוקצה יותר גדול ממנו בכתובת נמוכה יותר
- (ב) יש קטע בלתי מוקצה יותר קטן ממנו בכתובת גבוהה יותר
- (ג) יש קטע בלתי מוקצה צמוד אליו
- (ד) קיים קטע בלתי מוקצה יותר קטן מהסכום של הגודל שלו והגודל של קטע בלתי מוקצה שכן

7. אלגוריתם LRU משמש לבחירת איזה דף לפנות כחלק ממנגנון הניהול של זכרון וירטואלי, אבל לא לפינוי בלוקים מה-buffer cache של מערכת הקבצים. הסיבה היא

- (א) בהקשר של דפדוף לא מעשי לשמור timestamps מדויקים
- (ב) הוא יעיל יותר מאלגוריתם השעון לבחירת דפים לפינוי מהזכרון הוירטואלי
- (ג) בהקשר של buffer cache עדיף להשתמש במנגנון של second chance
- (ד) הטענה בשאלה אינה נכונה, אלא להיפך: משתמשים ב-LRU ב-buffer cache ולא בדפדוף

8. כאשר יש פסיקה,

- (א) החומרה טוענת את הכתובת של הפונקציה הרלוונטית של מערכת ההפעלה לרגיסטר ה-PC, ומדליקה את ביט ה-kernel mode ברגיסטר ה-PSW
- (ב) החומרה מעבירה שליטה למערכת ההפעלה, ומערכת ההפעלה מדליקה את הביט הנ"ל ב-PSW
- (ג) החומרה מחפשת את הכתובת המתאימה ב-interrupt vector, וטוענת אותה ל-PSW
- (ד) מערכת ההפעלה מדליקה את ביט ה-kernel mode ב-PC, ומתחילה לרוץ באופן אטומי



9. בהנתן גרף משאבים והקצאות כמו זה הנתון משמאל,

- ואין אפשרות לבקשות נוספות,
- (א) ניתן לבצע את הבקשה של P2 כי זה יוביל למצב בטוח
- (ב) אי אפשר לבצע את הבקשה של P3 כי זה יוביל למצב לא בטוח
- (ג) המערכת כבר במצב של deadlock והכל אבוד
- (ד) אין כאן deadlock ובינתיים לא צריך לעשות כלום

שני תהליכים בקשו עוד משאבים ולא יכולים לקבל אותם כי כל המשאבים מוקצים, אבל כש-P1 יגמור לרוץ וישחרר את המופע של R1 נוכל

10. בהקשר של הבטחת הכניסה לקטע קריטי
- (א) הדרישה להתקדמות אומרת שכל תהליך בהכרח יכנס בסופו של דבר לקטע הקריטי
 - (ב) הדרישה להתקדמות אומרת שלא ידלגו מעל אף תהליך אינסוף פעמים
 - (ג) הדרישה להתקדמות אומרת שתהליך כלשהו יכנס לקטע הקריטי אחרי מספר סופי של צעדים
 - (ד) הדרישה להתקדמות נובעת מכך שמתקיימת הדרישה להגינות

11. אחת הסיבות לשימוש בהפקעה (preemption) במימוש של תזמון היא
- (א) זה מאפשר לפצות על חוסר ידיעת העתיד
 - (ב) זה מקטין את התקורה של מנגנון התזמון
 - (ג) זה מבטיח את זמן התגובה הממוצע האופטימלי
 - (ד) זה מאפשר לפתור בעיות של deadlock

12. מספר המנות (packets) שפרוטוקול TCP ישרד מבלי לחכות לאישור (ack) תלוי בין היתר בגורם

גודל החלון תלוי בכמה המקום יש (ב') וביכולת של הרשת, אבל מסיקים את זה מאובדן מנות ולא מודדים עומס. למרות זאת החלטנו לקבל גם את א'.

- הבא:
- (א) מידע שהתקבל ממדידות עומס ברשת
 - (ב) כמה מקום יש בחוצצים (buffers) של מערכת ההפעלה על מחשב היעד
 - (ג) הזמן שעבר מאז הפעם האחרונה שלא התקבל אישור על מנה שנשלחה
 - (ד) פרוטוקול TCP בכלל לא מחכה לאישורים אלא שולח מיד כל מנה שהאפליקציה מבקשת לשלוח

חלק ב' (45 נקודות)

ענה על 3 מתוך 4 השאלות הבאות. כל סעיף קטן שווה נקודה, פרט לאלה המסומנים.

1. תזמון (scheduling) עוסק בבחירת איזה תהליך להריץ. שתי שיטות תזמון בסיסיות הן FCFS המריצה כל תכנית עד סיומה (כולל המתנה ל-I/O) לפי הסדר בו הם נתונים, ו-round robin המריץ את כל התכניות בו זמנית תוך שימוש בהפקעה פריודית. מה ניתן לאמר על היחסים ביניהן בתנאים שונים ותחת מדדים שונים? הנח כי הניצולת (באחוזים) וההספק (עבודות לשניה) נמדדים על פרק הזמן מהתחלת התהליך הראשון עד סיום האחרון, שיש הרבה מאוד תהליכים ולכן מקרי קצה לא חשובים, התקורה להחלפת תהליכים קטנה משמעותית מהזמן לביצוע I/O, ובמקרי on-line המערכת לא נכנסת לרוויה. מקרא: סמן FCFS או RR אם אחד מהם טוב יותר, = אם שווים, ו-? אם זה תלוי במקרה

תנאים	מדד	FCFS	RR	=	?
כל התהליכים נתונים מראש (מצב off-line), הם מבצעים ניצולת המעבד	הספק (throughput)			☆	
חישוב טהור, ומתעלמים מהתקורה הנובעת מהחלפת תהליכים	זמן תגובה ממוצע				☆
כל התהליכים נתונים מראש (מצב off-line), הם מבצעים שילוב של חישוב ופעולות I/O, ומתעלמים מהתקורה הנובעת מהחלפת תהליכים	ניצולת המעבד		☆		
	הספק (throughput)		☆		
	זמן תגובה ממוצע				☆
כל התהליכים נתונים מראש (מצב off-line), הם מבצעים חישוב טהור, ויש תקורה הנובעת מהחלפת תהליכים	ניצולת המעבד			☆	
	הספק (throughput)	☆			
	זמן תגובה ממוצע				☆
תהליכים חדשים מגיעים באופן אקראי (מצב on-line) עם מרווחים שונים, הם מבצעים חישוב טהור, ומתעלמים מהתקורה הנובעת מהחלפת תהליכים	ניצולת המעבד			☆	
	הספק (throughput)			☆	
	זמן תגובה ממוצע				☆
תהליכים חדשים מגיעים באופן אקראי (מצב on-line) עם מרווחים שונים, הם מבצעים שילוב של חישוב ופעולות I/O, ויש תקורה הנובעת מהחלפת תהליכים	ניצולת המעבד		☆		
	הספק (throughput)			☆	
	זמן תגובה ממוצע				☆

2. כאשר מספר תהליכים או חוטים (threads) חולקים גישה למבני נתונים משותפים, עלולות לצוץ בעיות.

i. אחד המנגנונים המאפשרים להתגבר על בעיות סינכרון הוא הסמפור. מה זה? (סמן את כל הנכונים)

לא הרחבה כי אי אפשר לבצע עליו פעולות חשבוניות

- (א) טיפוס נתונים (data type) שהוא בעצם אותו דבר כמו integer.
- (ב) הרחבה של integer עם פעולות נוספות, בפרט V-1 P.
- (ג) פתרון יעודי "תפור" במיוחד לבעית הקטע הקריטי ושום דבר אחר.
- (ד) טיפוס נתונים המאפשר לתהליכים או חוטים לבטא תלויות של אחד בשני באופן כללי.

ii. יש גם מנגנונים שמתבססים על פעולות אטומיות בחומרה. פעולה כזו היא compare-and-swap, המוגדרת באופן הבא (הכוונה שהחומרה עושה את כל זה באופן אטומי):

```
compare-and-swap(addr, oldVal, newVal) {
    if (*addr == oldVal) {
        *addr = newVal;
        return TRUE;
    }
    return FALSE
}
```

בהתבסס על פרימיטיב זה, מציעים לממש הגנה על קטע קריטי כך:

```
int mutex=13; /* shared variable */

beginCS( int* mutex ) { /* entry code */
    while (!compare-and-swap( mutex, 13, -1 ) )
        /* do nothing */;
}

endCS( int* mutex ) { /* exit code */
    *mutex = 13;
}
```

המימוש המקורי שגוי וכל מי שמנסה יכנס לקטע הקריטי תוך שני נסיונות. א' התקבל גם כי if יתן את אותה התנהגות שגויה. ניתן לתקן ע"י שלילה ב-while.

מה דעתך על המימוש הזה? (סמן את כל הנכונים)

- (א) אפשר היה להשתמש ב-if במקום while.
- (ב) המימוש הזה לא מבטיח מניעה הדדית בכלל. (2 נק')
- (ג) המימוש הזה עלול לגרום ל-deadlock. (2 נק')
- (ד) המימוש לא נכון, אבל אפשר לתקן אותו על ידי הוספת או מחיקת תו אחד בלבד [אם כן, בצע את התיקון על הקוד!]. (2 נק')

iii. כשרוצים להעביר איבר מרשימה משורשרת אחת לשניה במערכת ההפעלה, תוך שמירה על קונסיסטנטיות של כל מבני הנתונים, מה אפשר לעשות? (סמן את כל האפשרויות נכונות)

- (א) מגדירים סמפור שמגן על הפעולה הזו של העברת איבר ותופסים אותו לפני הביצוע.
- (ב) מגדירים מנעול על כל איבר ונועלים את האיבר שרוצים להעביר.
- (ג) במקום להגדיר קטע קריטי, אפשר לבצע את ההעברה באופן אטומי על ידי שימוש ב-compare-and-swap.
- (ד) לא צריך שום דבר מיוחד – פשוט לכתוב את הקוד בלי באגים.

צריך להגן על שתי הרשימות על ידי נעילה – אפשרות שלא מצויינת כאן. לנעול את האיבר לא מספיק כי משנים גם את אלה שמצביעים עליו. compare-and-swap יכול לעבוד על רשימה אחת אבל לא יכול לבצע העברה באופן אטומי.

3. אחת הפונקציות הבסיסיות של מערכת קבצים היא לתמוך בכתיבה וקריאה של נתונים.

i. למרות שנראה כי פעולות קריאה וכתיבה הן דומות מאוד, יש ביניהן הבדלים. מה מאפיין כל אחת? סמן את כל התשובות הנכונות.

(א) כשפותחים קובץ וניגשים לרצף בתים מסוים, פעולת כתיבה על בתים אלה עשויה לדרוש יותר פעולות I/O מאשר פעולת קריאה.

write חוזר כשהנתונים הועתקו ל-bc, ואילו read צריך לחכות לדיסק

(ב) כשפותחים קובץ וניגשים לרצף בתים מסוים, יתכן שגישה באמצעות קריאת המערכת (system call) write תחזור יותר מהר מגישה באמצעות read.

(ג) יתכן מצב בו פותחים קובץ עם open ואז קוראים עם read, והקריאה נכשלת למרות שיש נתונים בקובץ.

למשל אם פתחנו לכתיבה בלבד

(ד) יתכן מצב בו פותחים קובץ עם open ואז כותבים עם write, והכתיבה נכשלת למרות שיש מקום בדיסק.

למשל אם פתחנו לקריאה בלבד

(ה) יתכן שפעולת קריאה לא תקדם את המצביע למיקום בקובץ בהתאם לכמות הנתונים שביקשנו לקרוא.

למשל אם נגמרו הנתונים בקובץ

(ו) יתכן שפעולת כתיבה לא תקדם את המצביע למיקום בקובץ בהתאם לכמות הנתונים שביקשנו לכתוב.

למשל אם נגמר המקום בדיסק

(ז) במערכת NFS, פעולת write צריכה להיות אידמפוטנטית כדי להבטיח שהנתונים בקובץ יהיו קונסיסטנטיים, אבל פעולת read לא צריכה להיות אידמפוטנטית כי אינה משפיעה על הנתונים.

ii. המימוש של פעולות קריאה וכתיבה משתמש ב-buffer cache. מה מהדברים הבאים הם יתרונות של השימוש ב-buffer cache? סמן את כל התשובות הנכונות.

(א) מאפשר ניצול מקסימלי של הזכרון לצרכי התכניות הרצות במחשב.

(ב) תמיכה בלוקאליות תוך חיסכון בגישות לדיסק.

(ג) מתן אפשרות לביצוע קריאה מקדימה (prefetching).

(ד) תמיכה בקבצים זמניים בלי התקורה של גישה לדיסק.

(ה) תמיכה במיפוי קבצים על ידי mmap.

גם בלוקים שלמים מועתקים דרך ה-bc

(ו) אפשרות לחסוך העתקות בזיכרון אם ניגשים לבלוקים שלמים.
(ז) אפשרות לכיוונון ה-tradeoff בין אמינות שמירת הנתונים לבין התקורה של כתיבות לדיסק.
(ח) שיפור האמינות והבטחה שנתונים שנכתבו לא ילכו לאיבוד.

4. יש מקרים בהם רוצים שתהליכים יוכלו להתקשר זה עם זה ולהעביר נתונים, כאשר הם רצים על אותו מחשב או אפילו אם הם רצים על מחשבים שונים.

i. כאשר מדברים על תהליכים הרצים על אותו מחשב, אילו מהמנגנונים הבאים יכולים לשמש להעברת נתונים בין שני תהליכים שרירותיים במערכת? סמן את כל התשובות הנכונות.

(א) שיתוף סגמנט ה-data של אחד התהליכים עם תהליך אחר.

(ב) יצירת pipe (רגיל) בין התהליכים.

(ג) פתיחת socket על ידי שני התהליכים וקישורם על ידי מספר port משותף.

(ד) שימוש במערכת הקבצים.

סיגנלים לא מאפשרים להעביר נתונים

(ה) שליחת סיגנלים.

ii. DSM הוא מנגנון ליצירת זכרון משותף בין תהליכים הרצים על מחשבים שונים, המנצל מנגנונים קיימים של ניהול זכרון וירטואלי. בצורתו הבסיסית והפשוטה ביותר, מנגנון זה כולל בין היתר את הרכיבים והרעיונות הבאים. סמן את כל התשובות הנכונות.

- (א) ברגע נתון דף משותף יכול להיות ממופה לזכרון של אחד התהליכים אך לא לשניהם.
- (ב) בכל רגע כל דף משותף חייב להיות ממופה לזכרון של אחד התהליכים.
- (ג) הטיפול ב-page fault חוסם את התהליך עד שהדף החסר ממופה לזכרון.
- (ד) משתמשים ב-system call מיוחד כדי להעביר דפים מאחד לשני.
- (ה) כל דף ממופה כך שתהליך אחד יכול לכתוב אליו והשני יכול לקרוא ממנו.

iii. DSM בגרסה הפשוטה יכול להיות מאוד לא יעיל. ניתן לשפר אותו על ידי האופטימיזציות הבאות. סמן את כל התשובות הנכונות.

- (א) לאפשר לדף להיות ממופה לזכרון של שני התהליכים בו זמנית במקרים מסוימים. **למשל אם הוא לקריאה בלבד**
- (ב) לאפשר לדף לא להיות ממופה כלל.
- (ג) כאשר תהליך אחד נחסם בגלל page fault, ניתן להריץ בינתיים את התהליך השני.
- (ד) למפות מבני נתונים שונים לדפים שונים.
- (ה) לבסס את המימוש של ה-DSM על זכרון משותף במקום על sockets.

הסברים לשאלה ב 1:

במצב off-line בלי I/O יש פעילות רציפה של המעבד, לכן הניצולת 100% בלי תלות בשיטה. ההספק תלוי בזמן שזה לוקח, אז כשיש תקורה ל-RR ההספק נמוך יותר.
במצב off-line עם I/O שיטת RR יעילה יותר גם בניצולת וגם בהספק כי היא מנצלת את זמן ה-I/O של תהליך אחד להריץ חישוב של תהליך אחר.
במצב on-line ההספק נקבע לפי כמה עבודות מגיעות לאורך הזמן, ולא תלוי בשיטה (בהנחה שאין רוייה). לגבי הניצולת, אם יש ל-RR תקורה אז המעבד יעבוד יותר והניצולת שלו תעלה (אם כי זו לא עבודה יעילה).
ולבסוף, זמן התגובה הממוצע תלוי במקרה. היבט אחד הוא ההתפלגות של אורכי תהליכים. אם הם שווים, עדיף FCFS (גם עבור כל קבוצה של תהליכים שמגיעים באותו הזמן במקרה ה-on-line). אם רובם קצרים ומיעוטם ארוכים (עם זנב ארוך) עדיף RR. היבט אחר הוא ניצול הזמן של I/O. זה אמנם משפר את המצב של RR, אבל עדין אם זמני הריצה שווים וה-I/O לוקח פחות מחצי מהזמן FCFS ינצח.