מס׳ מחברת_____
מס׳ ת.ז._____

האוניברסיטה העברית

ביה״ס להנדסה ומדעי המחשב

מבחן במערכות הפעלה

קורס מס׳ 67808

מועד א׳ תשס״ב — תאריך: 20.2.02

המורה: ד״ר דרור פייטלסון — זמן: 2.5 שעות

במבחן שני חלקים. בחלק הראשון יש לענות על 20 מתוך 22 שאלות נכון/לא נכון. משקל כל שאלה 2 נקודות. בחלק השני יש לענות על 3 מתוך 4 שאלות, שמשקל כל אחת מהן 20 נקודות. מומלץ להשתמש במחברת לטיוטא, ולכתוב את התשובות הסופיות בטופס המבחן *בצורה נקיה ומסודרת.*

בהצלחה!

חלק א׳

ענה על 20 מתוך 22 השאלות הבאות, ע״י סימון התשובה הנכונה בצד שמאל. סמן בבירור את השאלות שאינך רוצה שיבדקו ע״י מחיקתן בקו.

בגרף המצבים של תהליך, המעבר ממצב running למצב blocked הוא תוצאה של פעולת הפקעה (preemption)	נכון	לא☺
הקצאת קטע זכרון רציף בשיטת first-fit תמיד לוקחת לפחות אותו זמן כמו הקצאה בשיטת best-fit	נכון	לא☺
תהליך ביוניקס יכול לאפשר לתהליך אחר גישה לקובץ ע״י כך שישלח לו את ה-file descriptor שקיבל כשפתח את הקובץ	נכון	לא☺
כשמתכננים מערכת הפעלה המשתמשת בדפדוף עבור מחשב מסויים, ניתן לבחור את גודל הדף כרצוננו וללא אילוצים כלשהם	נכון	לא☺
במערכת יוניקס, כאשר תהליך מבצע חילוק באפס מערכת ההפעלה הורגת אותו מיד	נכון	לא☺ (1)
בתהליך עם הרבה חוטים (threads) סגמנט הטקסט משוכפל עבור כל אחד מהם	נכון	לא☺
סיבה חשובה לחלוקת הודעות למנות בעלות גודל מקסימלי ידוע (packets) היא שאז ניתן להבטיח שיהיה להן מקום בחוצץ של מחשב היעד	נכון ☺ (2)	לא
תוצאת לואי של אכסון מידע בדיסק בשיטת RAID1 היא שניתן לקרוא אותו יותר מהר, ע״י בחירת העותק מהדיסק הפחות עמוס	נכון ☺	לא
בשימוש רגיל, כשמחברים מערכות קבצים ע״י פעולת mount רצוי שהספריה שאליה מחברים את המערכת השניה תהיה ריקה	נכון ☺	לא
מערכת ההפעלה יכולה לזהות מצב של thrashing ע״י מעקב אחרי קצב ה-page faults	נכון ☺	לא
מונעים מתכניות משתמש לרוץ ב-kernel mode ע״י כך שהמעבר ל-kernel mode נעשה רק ע״י פקודת trap, וזו פקודה מוגבלת (privileged instruction)	נכון	לא☺
פרוטוקול IP משמש לניתוב הודעות בין הרשתות השונות המרכיבות את רשת האינטרנט	נכון ☺	לא

כיוון שמערכת ההפעלה לא יודעת מראש כמה זמן כל תהליך ירוץ, אין מנוס ממצבים בהם תהליך קצר נתקע אחרי תהליך ארוך וצריך לחכות לסיומו	נכון	לא☺
במערכות הפעלה בכלל לא משתמשים באלגוריתם LRU כי הוא יקר מדי	נכון	לא☺
מבנה ה-inode של יוניקס מתאים לאחסנת קבצים שהגדלים שלהם מאופיינים ע"י התפלגות עם זנב כבד	נכון ☺	לא
במערכת NFS השרת מחזיק פרטי מידע שונים אודות קבצים שנפתחו, אך מותר לו לזרוק מידע זה ואין חובה לשמור אותו לאורך זמן	נכון ☺(3)	לא
במערכת המריצה תכניות מרובות חוטים, התקורה הדרושה למעבר מאחד לשני יכולה להיות תלויה באיזה חוטים מדובר	נכון ☺(4)	לא
תכונת הלוקאליות אומרת שה-resident set של תהליך קטנה יותר מה-working set שלו	נכון	לא☺
כשמשתמשים ברשת תקשורת איטית יחסית (כלומר רשת עם רוחב פס נמוך), צריך דווקא חוצצים יותר גדולים מאשר כשהרשת מהירה	נכון	לא☺ (5)
לשני תהליכים יש סגמנט של זכרון משותף. משתנה בסגמנט זה הנקרא X בתכנית של אחד התהליכים חייב להקרא X גם בתכנית של השני.	נכון	לא☺
ניוד (migration) של תהליך מעכב אותו לזמן רב כיוון שצריך להעתיק את כל מרחב הכתובות שלו לפני שמתחילים אותו מחדש	נכון	לא☺ (6)
הוספת קוד בקרה מתאים כגון CRC לכל מנה (packet) מאפשר להתגבר על כל אבדן מידע ברשת התקשורת	נכון	לא☺

(1) היא שולחת לו סיגנאל

(2) בלי גודל מקסימלי ידוע, השולח עלול להשתמש בגודל גדול מהחוצץ של המקבל, ואז שום תקשורת אינה אפשרית

(3) השרת בעקרון stateless

(4) חוטי משתמש עולים פחות מחוטי מערכת, וחוטים מאותה תכנית פחות מחוטים מתכניות שונות

(5) גודל החוצץ פרופורציוני לרוחב הפס

(6) ניתן להעתיק רק את מה שצריך תוך כדי ריצה ע"י מנגנון הדפדוף

(7) חלק ב׳

ענה על 3 מתוך 4 השאלות הבאות. סמן בברור איזו שאלה אין לבדוק ע״י מחיקתה בקו. ניתן לרשום הסברים בקיצור אחרי כל שאלה, אך אין חובה לעשות זאת אלא במקומות שצויינו במפורש. ההסברים יכולים להעלות ציון (אם התשובה לא נכונה אבל ההסבר כן) או להוריד ציון (אם ההסבר מראה שהתשובה הנכונה נבחרה מסיבה לא נכונה).

1) פקודת fetch-and-add(x,v) מוגדרת כביצוע אטומי של השגרה הבאה:

```
fetch-and-add(x,v) {
        s = x;
        x = x+v;
        return s;
}
```

(כאשר s משתנה זמני הקיים רק בתוך השגרה, ו-x מועבר by reference)
בהסתמך על פקודה זו, מוצע לממש מניעה הדדית בקטע קריטי באופן הבא.
יש משתנה משותף x המאותחל לערך 0.
קוד הכניסה לקטע הקריטי הוא

```
while (fetch-and-add(x,1) > 0) {
        fetch-and-add(x,-1);
}
```

והקוד בסיום הקטע הקריטי הוא

```
fetch-and-add(x,-1);
```

התכונות של מימוש זה הן:

מניעה הדדית	כן☺	לא
באם יש מניעה הדדית, היא תקפה	רק לשני תהליכים	למספר כלשהו של תהליכים☺
סכנת deadlock	כן	לא☺
סכנת livelock	כן☺	לא
באם יש סכנת deadlock או livelock, היא קיימת	לכל מספר של תהליכים מ-2 ומעלה☺	רק כאשר מספר התהליכים גדול ממש מ-2
הגינות	כשקבוצת תהליכים ממתינים, אם תהליך כלשהו נכנס, כל האחרים גם יכנסו בסוף	יתכן שתהליך לא יכנס אף פעם למרות שאחרים נכנסים שוב ושוב☺

הסברים:

יש מניעה הדדית כי ה- fetch-and-add אטומי, ורק תהליך אחד יקבל ערך חזרה 0 ויכנס לקטע הקריטי.

יש livelock לשני תהליכים באופן הבא:

1. תהליך א׳ מבצע fetch-and-add(x,1) ונכנס לקטע הקריטי. כעת x=1.
2. תהליך ב׳ רוצה להכנס ומבצע fetch-and-add(x,1). כעת x=2.
3. תהליך א׳ גומר את הקטע הקריטי, ויוצא. ביציאה הוא מבצע fetch-and-add(x,-1). כעת x=1.
4. תהליך א׳ רוצה להכנס שוב ומבצע fetch-and-add(x,1) . כעת x=2.
5. תהליך ב׳ (בשלב 2) קיבל 1 ולכן נכשל. הוא מבצע fetch-and-add(x,-1) המוריד את x ל-1, ומיד מנסה שוב ומעלה אותו חזרה ל-2.
6. תהליך א׳ (בשלב 4) גם קיבל 1 ולכן נכשל. הוא מבצע fetch-and-add(x,-1) המוריד את x ל-1, ומיד מנסה שוב ומעלה אותו חזרה ל-2.
7. שני השלבים האחרונים חוזרים ללא הגבלה.

2) נתונות חמש תכניות להרצה עם המאפיינים הבאים:

מס' סידורי	זמן הגעה	זמן ריצה
1	0	4
2	0	2
3	0	1
4	0	3
5	0	1

זמן ההגעה מציין מתי ניתן להתחיל להריץ כל תכנית, וזמן הריצה מציין כמה זמן היא דורשת. כל הפרטים הללו ידועים.

	לא קיים ☺	כן קיים	האם קיים תזמון שבו זמן התגובה הממוצע קטן ממש מ-5? הסבר בקיצור בתחתית הדף (למשל אם קיים, הדגם תזמון כזה)
	לא☺	כן	האם בתזמון הטוב ביותר האפשרי (כלומר עם זמן התגובה הממוצע הקטן ביותר) משתמשים בהפקעה (preemption)?
	PS עדיף	אחד אחרי השני עדיף☺	בתנאי השאלה, מה יותר טוב כדי להקטין את זמן התגובה הממוצע: להריץ תכנית אחת בכל רגע נתון, או להשתמש ב-processor sharing (PS) ולהריץ את כולן בו זמנית יותר לאט (בהנחה שניתן לעשות כזה דבר)
לא משנה ☺	עדיף להחליף	עדיף להמשיך	נניח שבהתחלה מריצים את תכנית מס' 2 למשך יחידת זמן אחת. האם זה משנה אם ממשיכים להריץ אותה, או עוברים להריץ את תכנית מס' 3?
	לא	כן☺	האם נכון שהתזמון שמביא לזמן תגובה ממוצע מינימלי גם מביא לגורם האטה (slowdown) ממוצע מינימלי?

הסברים:

התזמון האופטימלי במקרה זה ניתן ע"י אלגוריתם SJF, ומביא לזמן תגובה ממוצע של 5. לכן אין אף תזמון שיביא לזמן קצר יותר.

כיוון שאם מריצים את תכנית 2 למשך יחידת זמן אחת נשאר לה ול-3 אותו דבר, החלפה לא תשנה את זמן התגובה הממוצע.
התקבלה גם תשובה האומרת כי עדיף להמשיך, מלווה בהסבר כי עדיפות זו נובעת מהרצון לחסוך את התקורה של ההחלפה.

גורם ההאטה הוא זמן התגובה חלקי זמן הריצה. ההוכחה כי SJF אופטימלי כי כשיש שתי תוכניות עדיף להריץ את הקצרה קודם פועלת גם כאן.

3) במערכת מחשב יש מעבד ושני דיסקים: האחד משמש למערכת הקבצים, והשני משמש לצורך דפדוף (paging). במערכת נמדדו ערכי הניצולת הבאים:

ניצולת המעבד: 20%

ניצולת דיסק מערכת הקבצים: 6%

ניצולת דיסק הדיפדוף: 97%

איזה מהדברים הבאים צפוי לשפר את ניצולת המעבד? תשובות "אולי" דורשות הסבר קצר בתחתית הדף, המדגים כי גם שיפור וגם הרעה אפשריים.

התקנת מעבד מהיר יותר	כן	לא☺	אולי
התקנת זכרון גדול יותר	כן☺	לא	אולי
התקנת דיסק נוסף שישמש גם הוא לדפדוף	כן☺	לא	אולי
החלפת דיסק הדפדוף בדיסק גדול יותר	כן	לא☺	אולי
החלפת דיסק הדפדוף בדיסק מהיר יותר	כן☺	לא	אולי
החלפה בין שני הדיסקים: זה ששימש לדפדוף ישמש לקבצים, וזה ששימש לקבצים ישמש לדפדוף	כן	לא	אולי☺
הקטנת מספר התהליכים שמריצים בו זמנית ע"י פעולת swapping	כן☺	לא	אולי
הגדלת מספר התהליכים שמריצים בו זמנית ע"י החזרת תהליכים מ-swapping	כן	לא☺	אולי
הגדלת גודל הדף	כן	לא	אולי☺
הקטנת גודל הדף	כן	לא	אולי☺

הסברים:

ערכי הניצולת המצויינים בשאלה מצביעים על כך שצוואר הבקבוק של המערכת הוא הדפדוף (אולי כי המערכת במצב של thrashing). לכן כל שינוי שיביא לדפדוף יעיל יותר (דיסק דפדוף נוסף או מהיר יותר), או יקטין את העומס על הזכרון (יותר זכרון או פחות תהליכים), יאפשר לתהליכים לרוץ יותר, וכך ישפר את ניצולת המעבד.

החלפת הדיסקים יכולה לשפר או לו, בתלות במאפיינים שלהם, שאינם ידועים.

שינוי גודל הדף יכול לשפר או לא, בתלות בלוקאליות ובדגמי השימוש. בפרט, דפים יותר גדולים משמעם פחות דפים ס"ה, ויותר תחרות, אבל מצד שני אם זה גורם לכל הנתונים הנחוצים להגיע לזכרון ביחד זה יחסוך page fault ועשוי למנוע thrashing.

4) נתון כי רוחב הפס של רשת תקשורת הוא B בתים לשניה, והזמן הדרוש לבית מסויים להגיע מהמחשב המשדר למחשב היעד (או להיפך) הוא T שניות.

אם משדרים במנות (packets) בגודל P, ומחכים לאישור קבלה (ack) על כל אחת לפני שידור המנה הבאה, מהו קצב התקשורת (רוחב הפס) האפקטיבי המתקבל? הנח כי P קטן מאוד יחסית ל-BT. (רשום תשובה בלבד. נתן להסביר בקצרה בתחתית הדף)		P ------ 2T
האם שינוי גודל המנה P יביא לשינוי רוחב הפס האפקטיבי?	לא משמעותי אם בכלל	כן ☺ בהחלט
אותו דבר כמו השאלה הראשונה, אך הפעם הנח כי P גדול יותר מ-BT		ראה הסבר למטה
האם שינוי גודל המנה P יביא לשינוי רוחב הפס האפקטיבי במקרה זה?	לא משמעותי אם בכלל	כן בהחלט
בהנחה ש-ack מודיע לשולח שהנתונים הגיעו לחוצץ של המקבל ללא תקלה, המשמעות לשולח היא	שלא צריך לשמור יותר את המנה שנשלחה☺ / שניתן לשלוח את המנה הבאה	שתי התשובות נכונות
מהו גודל החוצץ (buffer) המינימלי הנחוץ במחשב השולח כדי לנצל את רוחב הפס של הרשת במלואו?		2BT (או ליתר דיוק (2BT + P
האם במצב זה (כלומר עם חוצץ בגודל שחישבת בשאלה הקודמת) שינוי גודל המנה P יביא לשינוי רוחב הפס האפקטיבי?	לא משמעותי אם בכלל☺	כן בהחלט
האם שימוש בחוצץ תקשורת גם במחשב המקבל יכול להשפיע על רוחב הפס האפקטיבי? הסבר בקצרה בתחתית הדף	כן☺	לא

הסברים:

הזמן לשלוח P בתים הוא P/B למשלוח + T עד שהאחרון מגיע + T לקבלת ה-ack.
לכן הנוסחה לרוחב הפס היא

$$\frac{P}{P/B + 2T}$$

אם P קטן יחסית ל-BT, הגורם הראשון במכנה זניח.
אם גדול, זה שקול לביטוי

$$\frac{B}{1 + 2BT/P}$$

כלומר רוחב הפס יגדל מ-B/3 ל-B כש-P גדל. השאלה אם זה "משמעותי" היא סובייקטיבית, ולכן סעיף זה בוטל.

ack אומר שהמנה הגיעה, אבל המקבל צריך לשמור אותה בחוצץ עד שיוכל להעביר אותה לאפליקציה שלה היא מיועדת. לכן יתכן שאי אפשר עדיין לשלוח עוד מנות, כי אין עוד מקום. זה גם מסביר למה גודל החוצץ של המקבל יכול להשפיע על רוחב הפס.