בחינה בעקרונות מערכות הפעלה

קראו בעיון לפני שתתחילו בפתרון הבחינה!

1. הבחינה מורכבת משלושה חלקים.
2. בחלקים א' ו ב' מופיעות שאלות פתוחות. ענו **תשובות מלאות**, **בכתב קריא** ו**בקיצור נמרץ**. אין חובה להשתמש בכל השורות המוקצות לצורך התשובות, אך אין לחרוג מהמקום המוקצה.
3. בחלק ג' (שאלות אמריקאיות). לכל שאלה עליכם לבחור תשובה יחידה מבין התשובות המוצעות ולהקיף בעיגול את אות התשובה שבחרתם.
4. כל חומר עזר אסור בשימוש בזמן הבחינה, פרט למחשבון פשוט
5. משך הבחינה - שלוש שעות.

בהצלחה!

* 1. חלק א

ענו על שלוש השאלות 3-1.

55 נקודות לחלק זה.

* + 1. שאלה 1 (19 נקודות)

בעיית הפילוסופים הסועדים שלמדנו את פתרונה בכיתה מנוסחת עבור n פילוסופים ו-n מזלגות. ננסח בעיה קלה יותר שבה נוסיף מזלג לשולחן. זאת אומרת, מערכת עם n פילוסופים ו- (n+1) מזלגות. המזלג הנוסף מונח במרכז השולחן וכל אחד מהפילוסופים יכול לקחת אותו (אולם, כמובן, רק פילוסוף אחד יכול לקחת אותו בכל זמן). הציעו אלגוריתם המשתמש בסמפורים, אשר מונע קיפאון ומביא למקביליות מירבית.

שני פתרונות אפשריים:

א. פתרון המבוסס על כך שפילוסוף כלשהו תמיד מבקש את המזלג fork[n], למשל

מזלג 1

מזלג 0

מזלג n-1

מזלג n

Pn

P0

ב. פתרון המבוסס על trylock, זאת-אומרת, לוקחים את אחד הסמפורים בין הפילוסופים, אחר כך *מנסים* לקחת את השני ואם לא מצליחים, הולך לתחרות כוללת על המזלג המרכזי.

\_סטודנט נדרש גם לכתוב קוד \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(המשך המקום לתשובה בעמוד הבא)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**המשך הבחינה בעמוד הבא**

* + 1. שאלה 2 (18 נקודות)

אלגוריתם lottery scheduling הינו אלגוריתם תזמון הסתברותי. כל תהליך במערכת מקבל מספר כלשהו של "כרטיסי הגרלה" (לכל כרטיס מספר ייחודי); בכל שלב, האלגוריתם מגריל מספר של אחד מכרטיסי ההגרלה שחולקו, והתהליך שמחזיק בכרטיס הגרלה זה מקבל את המעבד לפרק זמן קצוב**.**

1. נניח כי לאלגוריתם יש 20 כרטיסים. כיצד עליו לחלק אותם בין ארבעה תהליכים (A, B, C, D) כך שהתהליכים יקבלו קבלו 10%(A), 5%(B), 60%(C), 25%(D) בהתאמה?

לתת ל-A 2 כרטיסים, ל-B 1 כרטיסים, ל-C 12 כרטיסים, ול-D 5 כרטיסים.

1. כיצד ניתן להגיע באמצעות אלגוריתם זה לביצועים הדומים לאלו המושגים על-ידי אלגוריתם round-robin?

לחלק לכל התהליכים אותו מספר כרטיסים

1. במה שונה הפתרון מהסעיף הקודם מ- round-robin המקורי?

עקב האקראיות יש סיכוי מסוים (קטן מאוד) להרעבה

1. איך ניתן להגיע באמצעות אלגוריתם זה לביצועים הדומים לאלו המושגים על-ידי אלגוריתם shortest remaining time first?

(ניתן להניח כי יש preemption, התהליכים לא מגיעים יחד, אבל זמן הביצוע ידוע כאשר התהליך מגיע.)

עם הגיעו, התהליך מקבל מספר כרטיסי הגרלה ביחס הפוך לזמן הריצה שלו. אחרי כל יחידת זמן שהוא מבצע, הוא מקבל כרטיסים נוספים.

1. איזה יתרון יש לשיטה הזו על פני shortest remaining time first אמיתי?

בשיטה הזו, לכל תהליך יש סיכוי מסוים להתקדם בכל זאת, בעוד שב srtf תהליך ארוך יכול להיות מורעב לנצח על-ידי רצף תהליכים קצרים שמגיעים כל הזמן.

1. אם המערכת מזהה את קיום בעיית היפוך עדיפויות (priority inversion), כיצד אפשר לפתור את הבעיה?

להעביר חלק מכרטיסי ההגרלה שלו לתהליך אחר.

* + 1. שאלה 3 (18 נקודות)

הבהרות: סעיפי שאלה זו מתייחסים למערכת linux. הניחו שמשתמש יחיד רץ במערכת (כלומר התופעות המתרחשות להלן אינן נובעות מעבודה של מספר משתמשים במערכת הקבצים במקביל).

שימו לב: סעיפי השאלה אינם קשורים זה לזה, אלא אם נאמר אחרת.

1. משתמש הריץ את פקודת ls במדריך (directory) מסוים וקיבל בפלט 2 קבצים: hw.txt, other.txt. לאחר עדכון הקובץ hw.txt כך שגודלו השתנה, המשתמש שם לב שגם גודל הקובץ other.txt השתנה. איך קרה הדבר?

\_\_\_other.txt הוא קישור חזק (hard link) לקובץ hw.txt.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. משתמש הריץ את פקודת ls במדריך מסוים וקיבל בפלט 2 קבצים: hw.txt, other.txt. לאחר עדכון הקובץ hw.txt כך שגודלו השתנה, המשתמש שם לב שגם תוכן הקובץ other.txt השתנה למרות שגודלו נשאר ללא שינוי. איך קרה הדבר?

\_ other.txt הוא קישור סימבולי (symbolic link) ל-hw.txt.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. בהמשך לסעיף **ב**, המשתמש מחק את הקובץ hw.txt, ולאחר מכן הפעיל את הפקודה cat other.txt (תזכורת: cat מדפיסה תוכן של קובץ).

כתוצאה מכך, המערכת פלטה הודעת שגיאה:

cat: other.txt: No such file or directory

המשתמש בדק את תוכן המדריך (בעזרת ls) וראה בפלט את הקובץ other.txt. איך ניתן להסביר תופעה מוזרה זו? (הניחו שההודעה הנ"ל אינה התוכן של הקובץ.)

\_\_\_other.txt מצביע לקובץ שאינו קיים\_(אפשרי עבור symbolic link)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. בהמשך לסעיף **א**, המשתמש מחק את הקובץ hw.txt, ולאחר מכן הפעיל את הפקודהcat other.txt. מה תהיה תגובת המערכת כעת?

\_\_\_ תוכן הקובץ hw.txt (שקול: תוכן הקובץ ther.txt)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_(תוכן הקובץ ימחק רק לאחר שכל ה-hard links שלו ימחקו)\_\_\_\_\_\_\_\_

**המשך הבחינה בעמוד הבא**

1. הפקודה “rm –rf dir” מוחקת את המדריך dir ואת כל תתי המדריכים והקבצים שלו על-ידי הפעלת אלגוריתם DFS על תת-עץ המדריכים ששורשו במדריך dir (כלומר קודם נמחקים הקבצים בתת המדריך העמוק ביותר, אחר-כך במדריך שמעליו וכך עד שורש העץ). לאחר הפעלת פקודה זו על מדריך בדיסק הקשיח, המשתמש בדק את רשימת הקבצים הנמצאים על ה Disk On Key שלו שהיה מחובר למחשב בעת הפעלת rm, ומצא שכל קבציו נעלמו. איך קרה הדבר?

הניחו בסעיף זה שבמחשב אין קבצים מסוג link. (softlink/hardlink)

היה mount ל-Disk On Key בתת מדריך כלשהו של dir.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. תארו מקרה שבו הפקודה “rm –rf dir” יכולה לא להסתיים במערכת קבצים כלשהי.

במדריך dirיש קובץ שהוא symbolic linkל-dir.\_(קישור ציקלי)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**המשך הבחינה בעמוד הבא**

* 1. חלק ב

ענו על השאלות 8-4.

משקל כל שאלה 5 נקודות; 25 נקודות לחלק זה.

* + 1. שאלה 4

מה זה thrashing? האם אפשר לזהות את המצב הזה וכיצד אפשר לצאת ממנו?

תופעה במערכות זיכרון וירטואלי, שבה תהליך גורם לפסיקות דף אחרי ביצוע כל כמה הוראות קוד. במצב זה התהליך כמעט אינו מתקדם והמעבד עסוק רוב זמנו בהעברת דפים מהזיכרון ואליו. כיוון שזמן הטיפול בפסיקת דף גדול בהרבה מזמן הביצוע של הוראה שלא גורמת לפסיקת דף אפשר לזהות את המצב בקלות. אפשרויות לצאת המצב הן: הריגת תהליך או הקצאת מסגרות נוספות בזיכרון.

* + 1. שאלה 5

מהו monitor? מהו condition variable?

מבנה המסופק ע"י שפת תכנות המאפשר:

א. לא ניתן לגשת לנתונים ששייכים למוניטור, אלא רק באמצעות פונקציות ששייכות לאותו המוניטור.

ב. מובטח כי בכל נקודת זמן תהליך אחד לכל היותר יימצא בתוך המוניטור (כלומר, יבצע פונקציה כלשהי ששייכת למוניטור).

Condition variables הם מנגנון לסנכרון בין תהליכים של מבנה פיקוח.

* + 1. שאלה 6

מה ההבדל בין כתובת זיכרון וירטואלית לפיזית?

הראשונה מצביעה על יחידת התייחסות מינימאלית (לדוגמא memory word) במרחב הזיכרון המדומה והשנייה מצביעה ליחידת התייחסות בזיכרון הפיזי.

* + 1. שאלה 7

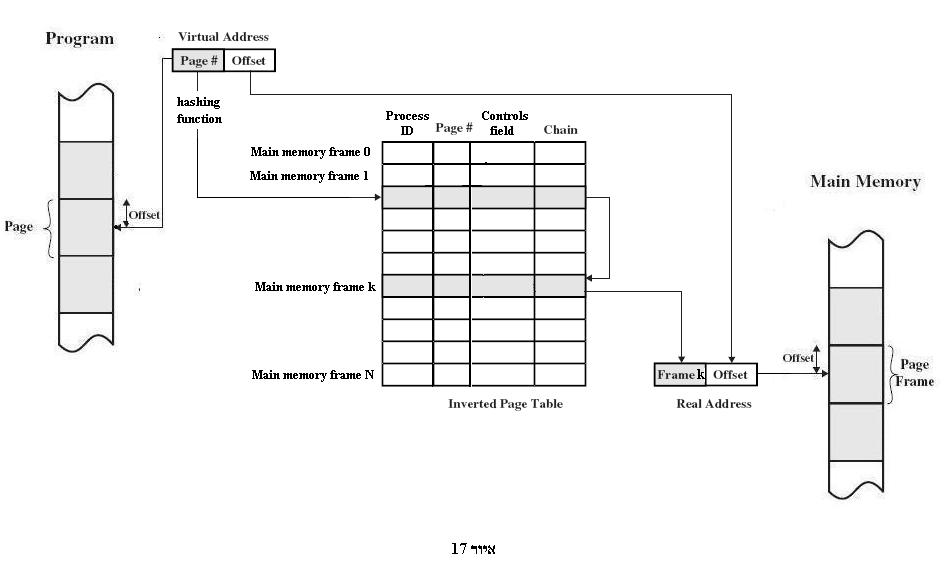
תארו מהי שיטת ה-DMA (direct memory access) ומהם היתרונות שלה על פני interrupt-driven I/O.

רכיב DMA מאפשר העברת נתונים ישירה בין חוצץ (buffer) פנימי של הבקר לבין הזיכרון הראשי, ללא מעורבות המעבד הראשי (וזה היתרון המבוקש). המעבד נותן פקודה לבקר, מציין מהיכן להיכן לבצע את ההעברה ומתפנה לעיסוקיו. בתום העברת כל הנתונים, הבקר שולח למעבד פסיקה ומערכת ההפעלה מטפלת בה ומבצעת תזמון תהליכים לאחר תום הטיפול בפסיקה.

* + 1. שאלה 8

מהו inverted page table? ציירו כיצד מתבצע תרגום כתובת לוגית לכתובת פיזית באמצעות inverted page table.

טבלה הממפה מסגרות זיכרון לדפים וירטואליים במרחבי זיכרון של תהליכים שונים.



* 1. חלק ג

שאלות רב-ברירה (אמריקאיות).

לכל שאלה עליכם לבחור תשובה יחידה מבין התשובות המוצעות ולהקיף בעיגול את אות התשובה שבחרתם.

משקל כל שאלה 5 נקודות; 20 נקודות לחלק זה.

* + 1. שאלה 9

מערכת הקבצים של מערכת הפעלה מסוימת משתמשת בשיטת ה I-node.

* גודל הבלוק במערכת הקבצים הוא 1 Kbyte
* כתובת הבלוק בדיסק היא 4 בתים (bytes)
* 12 שדות של ה- I-node יכולים להחזיק ישירות כתובת הבלוק בדיסק
* שדה נוסף אחד נועד להחזיק כתובת של ה- single indirect block
* עוד שדה נוסף אחד נועד להחזיק כתובת של ה- double indirect block
* ועוד שדה נוסף אחד נועד להחזיק כתובת של ה- triple indirect block

גודלו של קובץ מסוים במערכת 1000 Kbyte. מהי כמות הבלוקים שדרושה להחזקת קובץ זה במערכת הקבצים (לא כולל את הבלוק שמכיל את ה-i-node של הקובץ)?

1. 1000
2. 1005
3. 1010
4. 1011
   * 1. שאלה 10

במערכת UNIX נתונה הורצו 2 התכניות הבאות:

|  |  |
| --- | --- |
| //Program 2  #define AR\_SIZE 2048  main(){  int i,j,a[AR\_SIZE][ AR\_SIZE];  for ( i=0; i<AR\_SIZE; ++i )  for ( j=0; j<AR\_SIZE; ++j)  a[j][i]=0;  } | //Program 1  #define AR\_SIZE 2048  main(){  int i,j,a[AR\_SIZE][ AR\_SIZE];  for ( i=0; i<AR\_SIZE; ++i )  for ( j=0; j<AR\_SIZE; ++j )  a[i][j]=0;  } |

שימו לב, ההבדל היחיד בין התכניות הוא בשורה אחת בלבד. שתי התכניות הורצו במקביל מספר רב של פעמים ובכל פעם נמדד הזמן מתחילת ההרצה. התברר כי בכל פעם התכנית הראשונה מסתיימת בזמן קצר יותר מאשר התכנית השנייה. ידוע שגודל המערך הדו-ממדי a נכנס לקבוצת העבודה. מה הסיבה **הסבירה ביותר** לכך?

הערה: שימו לב שעקב מספרן הרב של ההרצות ניתן לומר שהתכניות נמצאות בתנאים זהים מבחינת הקצאת משאבים על-ידי מערכת הפעלה.

* + 1. הרצת התכנית השנייה גורמת ליותר פעולות החלפת דפים בזיכרון.
    2. הרצת התכנית השנייה גורמת לתחלופה תדירה יותר של תוכן ה-cache במעבד.
    3. הרצת התכניות במקביל עלולה לגרום לקיפאון ולכן מערכת ההפעלה מעדיפה לסיים את הרצת התכנית הראשונה קודם.
    4. התשובות א ו-ב הן הנכונות.
    5. שאלה 11

איזו פעולה מן הפעולות הבאות אפשר לבצע אך ורק במצב ראשוני (kernel mode) במערכת ההפעלה Linux?

1. חסימת פסיקות החומרה (disabling hardware interrupts)
2. החלפת תהליכונים (thread switch) כאשר מדובר בספריית תהליכונים ברמת המשתמש
3. השמת ערך במשתנה גלובאלי
4. את כל שלוש הפעולות הנ"ל יש לאפשר אך ורק במצב ראשוני
   * 1. שאלה 12

לפניכם קטע קוד:

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <signal.h>  #include <stdlib.h>  void handler(int signal){  printf("in signal handler\n");  }  int main( int argc, char \*argv[]){  struct sigaction sa;  sa.sa\_flags = SA\_RESTART;  sigfillset(&sa.sa\_mask);  sa.sa\_handler = handler;    if (sigaction(SIGUSR2, &sa, NULL) < 0)  exit(1);    printf( "before kill()\n");  kill(SIGUSR2, getpid());  printf( "after kill()\n");  return( 0 );  { |

בחרו מה יהיה הפלט בעקבות הרצתה של התכנית:

1. before kill()

in signal handler

after kill()

1. before kill()

after kill()

1. in signal handler

after kill()

1. before kill()

in signal handler

**בהצלחה!**