**מבוא למערכות הפעלה, תרגיל מספר 3**

**ניהול זיכרון**

כללי :

מועד ההגשה: ה-14 ביוני 2019, 23:59 בלילה.

אופן הגשה: מערכת ההגשה האוטומטית של המחלקה.

שאלות הקשורות לעבודה יש לשאול **אך ורק** בפורום המתאים. העזרו בו ככל הניתן.

רקע:

בתרגיל זה נממש יחידת ניהול זיכרון (memory management unit – MMU) המשתמשת ב-swapping בכדי לאפשר הקצאת זיכרון מעבר לגודל הזיכרון הפיזי. התרגיל עוסק בזיכרון **רציף** ולא בשיטת ה-paging הנהוגה בזיכרון וירטואלי. התרגיל יחולק למספר שלבים: ראשית, נממש אלגוריתם מיון מקבילי המשתמש בזיכרון הרגיל, אחר כך, נוסיף יחידת זיכרון ללא swapping ובתוך כך נממש הקצאות זיכרון. לבסוף, נוסיף את היכולת להקצות זיכרון גדול יותר מהזיכרון הפיזי. למען הפשטות, לא נשתמש בזיכרון היכול להכיל טיפוסי משתנים שונים, אלא במערכים מטיפוס int בלבד. כמו כן, לכל תהליך יהיה מערך יחיד מטיפוס int.

הערות:

1. בתרגיל זה נתון לכם שלד הפרויקט אותו עליכם להשלים. מותר להוסיף שיטות (פרטיות) ומבני נתונים כרצונכם. עם זאת, אין לשנות את חתימות השיטות הפומביות במחלקות השונות. הקוד במחלקה Program הינו לצרכי דיבוג בלבד. מותר לכם לשנות קוד זה כרצונכם – להגדיל או להקטין את גודל הזיכרון והמערכים או לשנות את ערכיהם.
2. על מנת לעבור משלב לשלב בביצוע התרגיל, עליכם לשנות ב-Main רק את טיפוס ה-MemoryManagementUnit.
3. שימו לב כי אנחנו מעוותים מעט בתרגיל זה את הגדרת פרטיות הזיכרון, ומאפשרים לתהליך לגעת בזיכרון המוקצה לתהליך אחר.
4. נשתמש בתרגיל ב-design pattern הידוע בשם singleton, אותו למדתם בקורסים קדמים, המאפשר יצירת אובייקט יחיד בלבד ממחלקה מסוימת (במקרה זה, אובייקטים המממשים את MemoryManagementUnit). ה-Singleton ממומש על ידי יצירת בנאי פרטי, שדה סטטי, ושיטה סטטית בשם getInstance. כמו כן, הוספנו שיטות בשם SetMemoryManagementUnitType הקובעות באיזה מבין המימושים האפשריים להשתמש (דוגמת שימוש ב-Main).
5. השתדלו להשתמש במחלקה Debug ובשיטותיה. לדוגמא, הוספנו שימוש בשיטה Debug.Assert, העוצרת את הקוד כאשר תנאי מסוים אינו משתמש.

**שלבי העבודה:**

1. **מימוש MergeSort באופן מקבילי**:

נתחיל במימוש האלגוריתם MergeSort באופן מקבילי, על ידי שימוש במספר תהליכים (threads). **שימו לב** – בשלב זה וגם בשלבים הבאים (מלבד ב-Main), אין ליצור לעולם מערך מטיפוס Int בדרך הרגילה (מלבד המערך הגלובלי המוכל ב-MMU). עבודה עם מערכים מותרת רק על ידי שימוש ב-IntArray.

* 1. המחלקה MultipleArrayMemoryManagement נתונה לכם. מחלקה זו מממשת את הזיכרון על ידי טבלה (dictionary) הממפה כל תהליך למערך פרטי של ints. התבוננו במחלקה על מנת להבין כיצד היא פועלת.
  2. בשלב זה עליכם להשלים את השיטה Sort במחלקה SortingThread, על פי אלגוריתם ה- MergeSort, כך שתמיין את המערך m\_aArrayToSort. להזכירכם, אלגוריתם MergeSort פועל באופן הבא:
     1. אם המערך בגודל 1, הרי שהוא כבר ממוין.
     2. צרו שני SortingThread חדשים. גודל הזיכרון הוא כמחצית המערך המקורי (±1).
     3. העתיקו את חציו הראשון של המערך אל הזיכרון של ה- thread הראשון, ואת חציו השני אל ה-thread השני.
     4. הריצו את שני ה-SortingThread שייצרתם (על ידי שימוש בשיטה Run), על מנת שימינו את שני החצאים באופן רקורסיבי. המתינו לסיום הריצה על ידי שימוש במתודה Join.
     5. מזגו את שני החלקים מזיכרונות ה-threads אל תוך המערך המקורי. כעת המערך ממוין.
     6. לאחר המיזוג – מחקו את המערכים על ידי קריאה ל-DeleteArray של ה-thread הרלוונטי.
  3. שימו לב – בשלב זה ההתמקדות הינה במימוש מיון מקבילי תקין. בדקו על מספר דוגמאות כי המיון שלכם עובד כראוי. בשלב זה הזיכרון מורכב ממספר מערכים שונים, ואין עבודה מול סימולציית הזיכרון.

1. **מימוש זיכרון בגודל חסום**:

בשלב זה, כאשר אלגוריתם המיון פועל באופן תקין, נעבור למימוש ה-MMU, ראשית על ידי זיכרון בגודל חסום. עליכם לממש את השיטות הרלוונטית במחלקה BoundedMemoryManagementUnit. במחלקה זו הזיכרון הפיזי ממומש על ידי מערך של משתנים מטיפוס int המוגדר בהתחלה. כאשר תהליך מבקש הקצאת זיכרון על ידי השיטה New, יש להקצות לתהליך חלק מהמערך, אם קיים מקום פנוי. אם לא, יש לזרוק שגיאה מטיפוס OutOfMemoryException.

עליכם לשמור מבני נתונים הממפים כל thread אל המקום המוקצה לו במערך הזיכרון הפיזי, כך שתוכלו להפנות את הקריאות SetValue ו-ValueAt המגיעות מהמערכים IntArray למקום המתאים בזיכרון. בשלב זה של העבודה אין "חורים" בזיכרון, כי נצמיד תהליכים אחד אחרי השני ולא נתעסק עם מחיקה ופינוי מקום בזיכרון. כלומר, שימוש בשיטה Delete פשוט יפנה את המקום המתאים במבנה הנתונים ששמרתם הממפה thread לזיכרון שלו אך מערך הזיכרון עצמו לא ישתנה.  
הערה – בשלב זה יתכנו מרוצים, למשל כאשר שני Threads מבקשים הקצאת זיכרון בו זמנית. הגנו על הקטעים הקריטיים באמצעות ה-Mutex הנתון במחלקה MemoryManagementUnit

1. **מימוש זיכרון לא חסום על ידי swapping**:

שלב זה הינו השלב המרכזי בתרגיל. בשלב זה תממשו זיכרון לא חסום, כאשר נעניק זיכרון כולל מעבר לקיבולת הזיכרון הפיזי על ידי העברת הזיכרון של תהליכים לא פעילים אל הדיסק, פעולה המכונה swapping. כמובן, מדובר בזיכרון הכולל של כל התהליכים, כאשר תהליך בודד יכול לקבל לכל היותר זיכרון כגודל הזיכרון הפיזי. ניהול הזיכרון בתחילה דומה לניהול הזיכרון החסום בשלב הקודם. אולם, כאשר תהליך דורש זיכרון ואין מספיק זיכרון פיזי פנוי, נעביר את הזיכרון של חלק מהתהליכים אל הדיסק ובכך נפנה מקום. כאשר תהליכים אלו ינסו לגשת לזיכרון שוב, יש להביא את הזיכרון הרלוונטי מהדיסק חזרה לזיכרון הפיזי. על מנת להקל עליכם, נתנו הגדרה של מספר שיטות פרטיות בהם תוכלו להשתמש. עם זאת, אתם רשאים להגדיר שיטות אחרות פרטיות כרצונכם ולהוסיף קוד לשיטות שניתנו. הערה – גם במחלקה זו יתכנו מרוצים רבים ועליכם להגן על הקטעים הקריטיים על ידי mutex.

* 1. עליכם להחזיק רשימה משורשת של מקטעי הזיכרון המוקצים והחופשיים ממוינת על פי הכתובת בסדר עולה. השתמשו במחלקה MemoryBlock על מנת להגדיר רשימה משורשרת זו. שימו לב – למרות חוסר היעילות, ועל מנת להקל על העבודה, נשתמש ברשימה יחידה עבור כל מקטעי הזיכרון.
  2. ממשו את השיטה SwapOut המעתיקה את הזיכרון של תהליך אל הדיסק. לכל תהליך נקצה קובץ זיכרון משלו, כאשר שמו הוא שם התהליך (Thread.Name) עם הסיומת ".data". אל הקובץ נרשום ראשית את גודל הזיכרון המוקצה לתהליך ואח"כ את התו ",". אחר כך תבוא רשימת הערכים המאוחסנים בזיכרון מופרדת על ידי רווחים.
  3. ממשו את השיטה New:
     1. מצאו "חור" – קטע זיכרון חופשי בגודל מספיק. במידה ואין קטע פנוי בגודל מספיק, העבירו תהליכים לדיסק על ידי השיטה SwapOut. עליכם להעביר תהליכים על פי סדר הגישה אליהם (**least recently used**). כלומר, התהליך שלא ניגשו לזיכרון שלו זמן רב ביותר ייצא לדיסק ראשון. שימו לב – כאשר תהליך עובר לדיסק קטע הזיכרון שהוקצה לו הופך לחור. בשלב זה יתכן כי ישנו חור נוסף לפני או אחרי החור החדש (לעיתים גם לפני וגם אחרי), ועליכם למזג את החורים העוקבים לחור יחיד. יש להמשיך ולהוציא תהליכים לדיסק עד שנוצר חור בגודל מספיק. אם יש מספר חורים פנויים, השתמשו בשיטת ה-**first fit**.
     2. הקצו את המקטע לתהליך על ידי הגדרת ה-Owner כ-thread המבקש את הזיכרון. אם החור גדול מדי, יש לחלק אותו ולהקצות לתהליך רק את מספר ה-ints הנדרש. את שארית התאים יש לשים בחור חדש.
     3. צרו IntArray חדש והחזירו אותו.
  4. ממשו את השיטה SwapIn, המביאה זיכרון מהדיסק. ראשית, עלינו לפתוח את הקובץ הרלוונטי ולקרוא את גודל הזיכרון הנדרש. אחר כך השיטה מקצה זיכרון בגודל הנדרש על ידי קריאה ל-New. שימו לב כי בשלב זה יתכן ואין די זיכרון זמין והשיטה New תאלץ לשלוח תהליכים אחרים אל הדיסק. לאחר הקצאת הזיכרון, השיטה תקרא מהקובץ את ערכי התאים ותעתיק אותם לזיכרון הפיזי.
  5. ממשו את השיטות ValueAt ו-SetValueAt. בשתי השיטות יש לבדוק ראשית אם התהליך נמצא בזיכרון הפיזי. אם לא, יש להשתמש בשיטה SwapIn על מנת להביא את הזיכרון מהדיסק. אחר כך עליכם לפנות לתא הזיכרון הפיזי על מנת לקרוא או לכתוב ממנו, שימו לב – שיטות אלו מהוות גישה לזיכרון של תהליך והדבר צריך להילקח בחשבון עבור שיטת הswapOut.
  6. ממשו את השיטה Delete – כאשר תהליך מסתיים עליו להודיע למערכת על שחרור הזיכרון שלו על ידי שיטה זו. עליכם למחוק גם את הקובץ על הדיסק, אם קיים כזה.

הגינות – אל תעתיקו! אין להעתיק קטעי קוד מתלמידים אחרים, מהאינטרנט, או מכל מקור אחר. להזכירכם, אסור לשלוח קטעי קוד לתלמידים אחרים.

בהצלחה!