תרגיל בית 2

מועד הגשת התרגיל: עד יום 11.12.2016 בשעה 16:59:59. לא יהיו דחיות.

מטרות התרגיל:

- הבנת מושגי התהליך (Process) והחוט (Thread) במערכת הפעלה בכלל וב-Windows בפרט.
- עבודה עם מספר תהליכים וחוטים במקביל.
- שימוש ב-MSDN Documentation.
- שימוש בפונקציות ה-API של תהליכים וחוטים.
- קבלת נתונים ריצה מתהליכים.

מבוא:

בתרגיל זה נלמד לעבוד עם תהליכים וחוטים. לכן, תכתבו מערכת להרצת מספר בדיקות במקביל על מספר קבצים. סט הבדיקות ירוץ בו זמנית על כל קובץ. התוכנית תבצע מעקב אחרי התהליכים שעדיין רצים, תדע לתת נתונים על התהליכים (זמנים, IDs, exit codes וכו'), ולסיים את התהליכים ולא שוכחת לסגור את ה-Handles שלהם.
בתרגיל זה עליכם לנהל **רשימה מקושרת** עבור נתוני התהליכים.

מימוש:

על התוכנה שאותה אתם מממשים להריץ את כל הבדיקות המתוארת בסעיף הבדיקות כמעט בו-זמנית. לשם כך, יש לייצר עבור כל בדיקה **חוט** שיהיה אחראי על הרצתו.
בנוסף, על סט הבדיקות הכולל לרוץ כ**תהליך** נפרד עבור כל קובץ.
כאשר יש עדיין תהליכים שרצים התוכנית תדווח על רשימת התהליכים שעדיין רצים ואת רשימת התהליכים שהסתיימו.
התוכנית תסיים כאשר על התהליכים יסיימו לרוץ והתוכנית תדפיס הודעת סיום.
כלומר, עליכם לכתוב שתי תכניות:

1. **תכנית הניהול TestManager.exe** - שתנהל את כלל הבדיקות (**תהליכים**), כאשר כל סט בדיקה רץ כ**תהליך** נפרד – זו התכנית שאנחנו נריץ ונבדוק לכם.
   *התכנית הזו מריצה כתהליכים את התכנית שמופיעה בסעיף הבא.
2. **תכנית הבדיקות TestFile.exe** - שתריץ את הבדיקות כ**חוטים**.

יש להגיש SOLUTION אחד שמכיל את שני הפרויקטים: TestFile ו-TestManager. כאשר קימפול ה-SOLUTION ייצור שני קבצי הרצה עם השמות של הפרויקטים בהתאמה ונוכל להריץ אותם כמתואר בשורות ההרצה לדוגמא שיפורטו בהמשך. לח
יש להשתמש בשמות התכנית שניתנו בתרגיל.

תכנית הבדיקות TestFile.exe:

על תכנית הבדיקות, שתקרא TestFile.exe להריץ את הבדיקות השונות כחוטים נפרדים ותסתיים רק כאשר כל החוטים סיימו את ריצתם.
על החוט הראשי לשלוח את הבדיקות השונות לריצה, להכניס למצב המתנה לסיום ריצת החוטים. על מנת ולהמתין לסיום ריצת החוטים עליכם להשתמש בפונקציית WaitForMultipleObjects.
הבדיקות שיש להריץ על כל קובץ הן:

1. בדיקת סיומת הקובץ (התווים האחרונים אחרי הנקודה הכי ימנית בשם הקובץ).

2. בדיקת גודל הקובץ
3. בדיקת זמני הקובץ – יש לבדוק מתי הקובץ נוצר ומתי הוא שונה (modified) בפעם האחרונה.
4. בדיקת תוכן הקובץ – על התכנית להחזיר את חמשת הבתים (bytes) הראשונים שמופיעים בקובץ.

על כל בדיקה להתחיל בפקודת Sleep(10).

תכנית הניהול TestManager.exe:

תכנית הניהול תקרא TestManager.exe. תכנית הניהול תנהל את הריצות השונות של תכנית הבדיקות. כלומר, התכנית תריץ את TestFile.exe עבור כל קובץ ברשימה כתהליך נפרד. התכנית תסיים כאשר כל התהליכים יסיימו את ריצתם ויכתבו ההודעות הנדרשות כפי שיתואר בהמשך. על התכנית להכניס למצב המתנה לפרק זמן שיוגדר על פי הפרמטר Process Status Check Frequency (שיוגדר בהמשך) ולאחריו יש לכתוב את מצב התהליכים הרצים ואלו שהסתיימו כפי שיופרט בהמשך כל עוד יש תהליכים שלא סיימו לרוץ. כל עוד יש תהליכים שרצים התכנית תכנס שוב למצב המתנה שבסיומו פרק הזמן שלו יודפס סטטוס התהליכים. התכנית תסתיים כאשר כל התהליכים יסיימו לרוץ ויודפס מידע סיום התכנית, כפי שיפורט בהמשך.

שורת הרצה של תכנית הניהול TestManager.exe:

התוכנית תרוץ ע"י שורת פקודה עם הארגומנטים הבאים:

TestManager.exe <Files to Test> <Output Files Directory> <Process Status Check Frequency>

- Files to Test – זהו קובץ המכיל רשימה של שמות קבצים אותם אנו מעוניינים לבדוק.
- Output Files Directory – תיקייה שבה ישמרו כל קבצי ה-output:
  - על התכנית ליצור בתיקייה זו קובץ log בשם runtime_logfile.txt, תוכנו יפורט בחלק הפלט.
  - קבצי הפלט של תכנית הבדיקות יכתבו לתיקייה זו. כאשר לכל סט בדיקות צריך להיות קובץ פלט אחת בלבד (**אין לכתוב קובץ פלט עבור כל בדיקה בנפרד**). שם הקובץ של פלט תכנית הבדיקות צריך להיות בפורמט הבא: <tested_file_name>_log.txt. תוכנו יפורט בחלק הפלט.
  - Process Status Check Frequency - התדירות, במילישניות, שבה יש לבצע בדיקות סטטוס לתהליכים שרצים והסתיימו ולכתוב ללוג (מפורט בהמשך).

יש לבדוק שמספר הארגומנטים של שורת הפקודה חוקי. מעבר לכך, מותר להניח שכל הקלטים תקינים ותואמים להגדרות.

שורת הרצה של תכנית הבדיקות TestFile.exe:

גם לתכנית 2 יש ארגומנטים:

FileTest.exe <File_path> <Output Logfile>

- File_path – מיקום הקובץ שעליו צריך להריץ את סט הבדיקות.
- Output Logfile – מיקום קובץ הפלט של תכנית הבדיקות. על קובץ הפלט להיות במיקום:

<Output Files Directory >\<tested_file_name>_log.txt

כאשר Output Files Directory ו-tested_file_name כפי שתואר בסעיף שורת ההרצה של תכנית הניהול.

קלט תכנית הניהול - מבנה הקובץ "Files to Test":

התכנית תקבל קובץ המכילים את מיקומי הקבצים שעל התוכנית לבדוק.
ניתן להניח, כי הקבצים לבדיקה כולם נמצאים בתיקייה הנוכחית (current directory), כלומר שם הקובץ יופיע ללא **path**, לדוגמא:

HW1.txt

אך, לא ניתן להניח שגודל שמות הקבצים קטן מגודל מקסימיאלי כלשהו.
דוגמא לתוכן קובץ Files to Test:

```
ReadMe.txt
HW1.txt
HW2.txt
```

פלט תכנית הבדיקות – מבנה הקובץ <tested_file_name>_log.txt:

כל קובץ ירכז את תוצאות החוטים השונים.
על כל קובץ להכיל את הפורמט הבא:

<Tested File Name>
The file extension of the tested file is "<file extension>"
The test file size is <file size>
The file was created on <creation date>
The file was last modified on <modification date>
The file's first 5 bytes are: <first 5 bytes of the file>

זמנים צריכים להיות מודפסים על פי הפורמט הבאים:

dd/mm/yyyy, hh:nn:ss

כאשר מספר החזרת של כל אות מייצגת את מספר הספרות המוקדשות לה, ומשמעות האותיות הן:

d – day
m – month
y – year
h – hour
n – minute
s – second

דוגמא לתוכן קובץ <tested_file_name>_log.txt:

```
ReadMe.txt
The file extension of the test file is ".txt"
The test file size is 1kB
The file was created on 01/01/2016 00:00:00
The file was last modified on 31/08/2016 23:59:59
The files first 5 bytes are: hello
```

פלט תכנית הבדיקות – מבנה הקובץ runtime_logfile.txt:

בקובץ הזה ירוכז המעקב אחרי התהליכים השונים. על מנהל התהליכים לבצע דגימה (מעקב) של סטטוס התהליכים שרצים ויסתיימו כל x מילישניות, כאשר x מוגדר לתכנית לפי ארגומנט <Process Status Check Frequency>. כלומר בכל דגימה יש לרשום את כל התהליכים שעדיין

רצים ואת התהליכים שהסתיימו כפי שיופרט בהמשך. במידה ואין תהליכים שרצים או אין תהליכים שהסתיימו בעת הדגימה, אין צורך להוציא כלל את ההודעה הרלוונטית, כולל הכותרת.

על הקובץ לכלול את ההודעות הבאות:

1. אם יצירת התהליך החדש הצליחה התוכנית מדפיסה את השורה:

Successfully created a process with ID <id> to execute <command line>

2. אם יציאת התהליך נכשלת התוכנית תדפיס:

!!! Failed to create new process to run <command line>. Error code: <error code> !!!

במקום ה-<error code> יש להדפיס את הערך ההקסדצימלי של קוד השגיאה.

3. כאשר מדווחים על רשימת התהליכים שרצים יש לציין את השורה הבאה, רק במידה ויש תהליכים שעדיין רצים:

List of running processes:

לאחריה להדפיס את רשימת התהליכים שרצים בסדר עולה של ה-ID, כלומר יש למיין את התהליכים לפי שדה ה-ID. כל תהליך יודפסו לפי בשורה לפי הפורמט הבא:

Process <id> is running command <command line> for <s> seconds and <m> milliseconds.

כאשר <s> ו-<m> מתארים את הזמן שעבר מאז נוצר התהליך ועד הרגע הנוכחי.

4. כאשר מדווחים על רשימת התהליכים שהסתיימו יש לציין את השורה הבאה, רק במידה ויש תהליכים שהסתיימו:

List of finished processes:

לאחריה להדפיס את רשימת התהליכים שהסתיימו בסדר עולה לפני ה-ID. כל תהליך יודפסו לפי בשורה לפי הפורמט הבא:

Process <id> ran command <command line> and exited with exit code <exit code> after <s> seconds and <m> milliseconds.

כאשר <s> ו-<m> מתארים את הזמן שעבר מאז נוצר התהליך ועד סיומו ו-<exit code> הוא ערך הקסדצימלי.

ניתן להניח שמשך הזמן שתהליך ירוץ הינו קטן מדקה.

5. הודעת סיום, כאשר דוגמים את סטטוס התהליכים ואין יותר תהליכים שרצים, יש להדפיס את רשימת כל התהליכים שהסתיימו בפורמט שהוסבר בסעיף הקודם ולאחר מכן להדפיס את השורה הבא:

All the processes have finished. Exiting program.

רצף של פלט לדוגמא:

```
Successfully created a process with ID 1111 to execute ReadMe.txt
Successfully created a process with ID 1112 to execute HW1.txt
Successfully created a process with ID 1113 to execute HW2.txt
List of running processes:
Process 1111 running command ReadMe.txt for 0 second 100 milliseconds
Process 1112 running command HW1.txt for 1 second 1 milliseconds
Process 1113 running command HW2.txt for 2 second 5 milliseconds
List of running processes:
Process 1112 running command HW1.txt for 2 second 25 milliseconds
Process 1113 running command HW2.txt for 3 second 100 milliseconds
List of finished processes:
```

```
Process 1111 ran command ReadMe.txt and exited with exit code 0x0 after 2 seconds and 374 milliseconds
List of finished processes:
Process 1111 ran command ReadMe.txt and exited with exit code 0x0 after 2 seconds and 374 milliseconds
Process 1112 ran command HW1.txt and exited with exit code 0x5 after 5 seconds and 297 milliseconds
Process 1113 ran command HW2.txt and exited with exit code 0x0 after 4 seconds and 763 milliseconds
All the processes have finished running. Exiting program.
```

סנכרון:

בתרגיל זה עליכם לבנות את הקוד שלכם כך שלא יידרשו מנגנוני סנכרון בין החוטים והתהליכים השונים.

משאב יקרא כזה הדורש סנכרון אם מתקיים עבורו שבאותה נקודת זמן שבה אחד החוטים/תהליכים כותב לתוכו, קיים לפחות שחקן נוסף שמבצע קריאה או כתיבה אל אותו המשאב.

עליכם לוודא שאין משאבים מסוג זה בתוכנית שלכם.

התמודדות עם שגיאות:

יש לבדוק את הצלחה של כל פונקציה שעשויה להיכשל (הקצאת, זיכרון, פתיחת קבצים, יצירת חוט וכו׳). במידה שמתרחשת שגיאה, יש לסיים את התוכנית באופן מסודר:

1. יש לשחרר/לסגור בצורה מסודרת את כל המשאבים שהוקצו: זיכרון, חוטים, קבצים וכו׳.
2. יש להדפיס הודעת שגיאה למסך.
3. יש להוסיף הודעת שגיאה לקבציי הפלט, גם ל-runtime log וגם ל-log של סט הבדיקות בציון ערך היציאה (exit code).
4. התמודדות עם שגיאה שמתרחשת בחוט שאיננו החוט הראשי:
   4.1. אין צריך לעצור מיד את שאר החוטים הלא-ראשיים וניתן לאפשר להם לרוץ עד סיום.
   4.2. יש ליידע את החוט הראשי באמצעות שימוש בערך היציאה (exit code) של החוט שבו התרחשה השגיאה.

הנחיות נוספות:

אין דרך אחת נכונה לפתור את התרגיל והתרגיל לא כוון לפתרון ספציפי.

הקפידו על ביצוע design לתוכנית לפני שאתם ניגשים לכתוב אותה. חישבו אילו פונקציות עזר אתם צריכים לפני שאתם ניגשים לממש אותן. זכרו כי כל קטע קוד שאתם משתמשים בו יותר מפעם אחת, צריך להיכתב כפונקציה נפרדת. כאשר פונקציה נעשית גדולה ומסובכת, פצלו אותה למספר פונקציות.

כתבו קוד קריא ותעדו אותו.

השתמשו בזיכרון דינמי לאחסן מידע שגודלו אינו ידוע בזמן הקומפילציה. אינכם רשאים להניח חסם עליון שרירותי לגודל המידע.

השתמשו בקבועים ושימו לב לשחרור זיכרון דינמי.

אתחלו את כל הפוינטרים ל-NULL.

אל תחכו עד שכל התכנית תהיה כתובה כדי לקמפל ולהריץ בדיקות. קמפלו ובדקו כל יחידה בנפרד (יחידה יכולה להיות פונקציה משמעותית, מודול, קבוצת מודולים וכו׳).

לפני שאתם משתמשים בפקודת API בפעם הראשונה, רצוי לבדוק את דף הMSDN שלה ע"מ להבין מה היא עושה ומה הפרמטרים שהיא מקבלת ומוציאה. אתם **חייבים** לבדוק בתוכנית את הערך שהיא מחזירה, ברגע שהיא מסתיימת ולפעול בהתאם. באופן כללי, רצוי גם לקרוא את הפונקציות שמופיעות תחת Related Functions ב-MSDN.

טיפים:

- את הערך של ה-error code של בעת יצירת תהליך ניתן למצוא בעזרת הפונקציה GetLastError().
- הפונקציה CreateProcess מצפה לקבל את שורת הפקודה כמחרוזת אחת מטיפוס LPTSTR. על כן יש ליצור משם התכנית שרצים להריץ ומהארגומנטים שלה מחרוזת אחת.
- כדי לחשב את הזמנים, עליכם להיעזר בפונקציות הרלוונטיות, וייתכן שגם בחלק מהפוקציות הבאות:,SystemTimeToFileTime(), GetSystemTime() GetLocalTime().

בהצלחה!