תרגיל בית 2

מועד הגשת התרגיל: עד יום חמישי 22/11/18 בשעה 23:55:00.

מטרת התרגיל

- הבנת מושגי התהליך (Process) והחוט (Thread) במערכות הפעלה בכלל וב-Windows בפרט.
- עבודה עם מספר תהליכים וחוטים במקביל.
- שימוש ב-MSDN Documentation.
- שימוש בפונקציות ה-WINAPI של תהליכים וחוטים.

הגשה

צורת ההגשה מפורטת במסמך "הנחיות להגשת תרגילי בית – תשע"ט" שבאתר המודל. אנה הקפידו למלא אחר ההוראות.

הגישו פרוייקט מלא, כולל קבצי פרויקט (*.sln, *.vcxproj, *.vcxproj.filters) של Visual Studio 2017, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא התראות או שגיאות.

שם הפרויקט צריך להיות ex2. שם ה-executable צריך להיות ex2.exe.

הגישו בנוסף את תיקיית ה-Debug עם קובץ ה-exe.

דגשים

- הקפידו על קוד קריא ומתועד.
- עבדו באיטרציות – בדקו את הקוד שלכם לעתים תכופות בעת הקידוד, ולא לאחר כתיבת התוכנה כולה.
- רשמו לעצמכם את מבנה התוכנה הכללי לפני שאתם מתחילים לקודד.
  - חשבו איזה מודולים ופונקציות אתם צריכים. מתוך הפונקציות, איזה יהיו סטטיות ואיזה פומביות.
  - זכרו כי כל קטע קוד שאתם משתמשים בו יותר מפעם אחת, צריך להיכתב כפונקציה נפרדת. כאשר פונקציה נעשית גדולה ומסובכת, פצלו אותה למספר פונקציות.
  - בפרט, נסו לקבץ במודולים נפרדים פונקציות שקשורות לתהליכים וחוטים (כגון CreateProcessSimple שראינו בתרגול). המודולים האלה ישמשו אתכם גם בתרגילים הבאים.
- זכרו להשתמש בכלי הדיבוג שה-IDE מספק.
- אין דרך אחת נכונה לפתור את התרגיל והתרגיל לא כוון לפתרון ספציפי.
- 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. כל פונקציה שמקבלת מצביע צריכה לבדוק שהוא שונה מ-NULL לפני שהיא עושה dereference (אופרטור *).
- בדקו את ערך החזרה של כל פונקציה, שיכולה להחזיר שגיאה (malloc, WaitForSingleObject וכו'). פעלו בהתאם לערך.
- לפני שאתם משתמשים בפקודת API בפעם הראשונה, רצוי לקרוא את התיעוד שלה ב-MSDN. באופן כללי, רצוי גם לקרוא את הפונקציות שמופיעות ב-MSDN תחת Related Functions.
- הפורום עומד לשירותכם. אנו מעודדים אתכם לנסות תחילה לחפש תשובות באינטרנט, כאשר מדובר בשאלות תכנות כללית.

בהצלחה!

סקירה כללית

עבור תוכנות מורכבות, קשה לדעת בוודאות שהתוכנית נקייה מ"באגים" לפני השחרור לקהל הלקוחות. בתרגיל זה ניצור מערכת לבדיקת תוכנות, באמצעות הרצת התוכניות עם ארגומנטים שונים. המטרה היא לבחון האם התוכניות נתנו את הפלט הצפוי ***בכל אחד*** מהטסטים. הרצת טסט מורכבת מ-2 חלקים: הרצת קובץ ה-executable עם הארגומנטים הרלוונטיים, ובדיקת התוצאה מול הפלט הרצוי.

קלט

התוכנית שלכם תקבל כקלט את שני הארגומנטים הבאים:

<path to test file> <path to result file>

<path to test file> - הינו נתיב לקובץ הטסטים שכל שורה בו מגדירה "טסט" אחר.

כל שורה בקובץ תהיה בפורמט הבא:

<path to exe> <arg1> <arg2> … <arg n> <path to expected results>

כלומר כל שורה בקובץ מכילה נתיב לתוכנה אותה נרצה לבדוק, את הארגומנטים שלה לפי הסדר, ונתיב לקובץ המכיל את הפלט הצפוי לאותה תוכנה. שימו לב שייתכן ולתוכנית הנבדקת אין ארגומנטים, כלומר רק הנתיב לתוכנית הנבדקת והתוצאות הצפויות שלה חייבות להופיע.

<path to result file> - נתיב לקובץ הפלט, כלומר קובץ שהתוכנית הראשית שלנו תיצור עם תוצאות ההרצה.

דרישה – מקביליות

גם אם נפתח מספר תהליכים כדי לבדוק את תוכנית המטרה עם ארגומנטים שונים, השוואת התוצאות עלולה להתבצע באופן טורי. לכן, נפעל באופן הבא: התוכנית הראשית שנכתוב, תפתח מספר חוטים (חוט אחד לכל שורת קלט). כל חוט אחראי ליצירת תהליך בודד, ובדיקת תוצאות ההרצה שלו. נמתין לתום הריצה של כל החוטים, ואז נכתוב את תוצאות ההרצה לתוך קובץ פלט.

פלט

מערכת בדיקת התרגילים מייצרת את קובץ הפלט המתאר את תוצאות הרצת הטסטים לפי הסדר.

הגדרות:

- נאמר שטסט הצליח (Succeeded) אם הפלט הנוצר ע"י התוכנית הנבדקת זהה בדיוק לפלט הרצוי (Case sensitive). הפלט במקרה הזה יווצר באותו נתיב של קובץ ה-exe הנבדק, יהיה בעל אותו שם, אך סיומת txt במקום exe. לדוגמה, קובץ test1.exe יצור פלט באותה תיקייה הנקרא test1.txt .
- נאמר שטסט נכשל (Failed) אם הפלט הנוצר ע"י התוכנית הנבדקת שונה מהפלט הרצוי (Case sensitive). גם במקרה הזה התוכנית הנבדקת יוצרת קובץ פלט כמו בסעיף הקודם.
- נאמר שהטסט קרס (Crashed) אם התוכנית הנבדקת הסתיימה עם ערך המוחזר שונה מ-0. במקרה זה, ***נדפיס לקובץ גם את הערך המוחזר*** (ראו דוגמת פלט). במקרה זה לא מעניין אותנו קובץ הפלט שהיא יצרה.
- נאמר שהתוכנית הנבדקת לא הסתיימה בזמן (Timed Out) אם לקח לה יותר מ-10 שניות להסתיים. במקרה זה אין צורך להרוג את ה-process, אלא רק לזהות זאת. גם במקרה זה לא נתעניין בקובץ הפלט שלה.

פורמט קובץ הפלט הוא כדלהלן:

test #<number> : <result>

דוגמת פלט:

```
test #1 : Succeeded
test #2 : Timed Out
test #3 : Succeeded
test #4 : Crashed -2
test #5 : Crashed -1
test #6 : Succeeded
test #7 : Failed
test #8 : Crashed 1
```

סדר ההדפסות לקובץ הפלט זהה לסדר הטסטים בקלט. שימו לב, המשמעות מבחינת הקוד היא שאת הדפסת התוצאות צריך לבצע מתוך ה-main thread, ולא מתוך החוטים שהוא פותח. זכרו – סדר פתיחת החוטים לא קובע את סדר ההרצה והסיום שלהם.

כאשר התוכנה הראשית מצליחה, אין להדפיס מידע נוסף (ואף לא שורות ריקות נוספות).

כאשר התוכנה הראשית נכשלת, אין להדפיס את תוצאות הבדיקה. במקום, יש להדפיס הודעת שגיאה. ראו את החלק על טיפול בשגיאות.

טיפול בשגיאות

התוכנה תחזיר 0 אם הסתיימה ללא שגיאות.

במקרה של שגיאה כלשהי, כגון כישלון בהקצאה דינאמית או כישלון בפתיחת תהליך, התוכנה תפסיק את פעולתה ותחזיר ערך שאינו 0. התוכנה תדפיס למסך הודעת שגיאה בעלת משמעות (בדומה לדוגמאות מהתרגולים). אתם רשאים לקבוע את הניסוח המדויק. מטרת הדרישה היא לסייע לכם בתהליך הדיבוג.

במקרה של כשלון כזה, יש לשחרר/לסגור את כל המשאבים הרלוונטיים בצורה מסודרת לפני היציאה מהתוכנית.

יש לבדוק את הצלחה של כל פונקציה שעלולה להיכשל (הקצאת זיכרון, פתיחת קבצים, יצירת חוט וכו').

אם השגיאה מתרחשת בחוט שאיננו החוט הראשי:

1. יש ליידע את החוט הראשי באמצעות שימוש בערך היציאה (exit code) של החוט שבו התרחשה השגיאה.
2. ניתן לאפשר לחוטים המשניים לרוץ עד סיום. אין צורך לעצור אותם מיד.

משאבים משותפים / (היעדר) סנכרון

התרגיל הזה מכסה את החומר עד Mutexes, **לא כולל**. עליכם לבנות את הקוד שלכם כך שלא יידרשו מנגנוני סנכרון (כגון Mutex) בין החוטים והתהליכים השונים.

באחריותכם לדאוג שלא יהיו בתוכנית משאבים, שעבורם יש סיכוי שיותר מחוט/תהליך אחד ינסה להשתמש בהם ברגע נתון. משאבים כאלה דורשים שימוש במנגנוני סנכרון שהוא, כאמור, אסור בתרגיל זה.

מותר להשתמש ב-WaitForSingleObject ו-WaitForMultipleObjects.

הערות והנחיות נוספות

- ניתן להניח שאורך שורה בכל קובץ אינו ארוך מ-100 תווים.

- ניתן להניח שהקלט תקין. כלומר כל שורה תהיה ע"פ הפורמט שהוגדר.
- ניתן להניח ששמות הקבצים (והנתיבים אליהם) לא מכילים רווחים.
- אגדו פונקציות בעלות נושאים משותפים למודולים רלוונטיים.
- אם אתם מעוניינים לבחון באופן ידני את הערך המוחזר מתוכנית מסויימת, תוכלו להשתמש בפקודה echo %errorlevel% דרך ה-command line:

```
C:\WINDOWS\system32>echo %errorlevel%
0
```