תרגיל בית 2

מועד הגשת התרגיל: עד יום ראשון 29/11/20 בשעה 23:55. לא תהינה דחיות

בודק אחראי: אורי ברכה uribracha@mail.tau.ac.il

מטרת התרגיל

- הבנה ושימוש בתהליכים ו-Thread-ים ב-Windows
- הכרת פונקציות ה-WinAPI של תהליכים ו-Thread-ים.
- התנסות בכתיבת קוד מקבילי וסנכרון פשוט

הגשה

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

הגישו פרויקט מלא, כולל קבצי פרויקט (*.sln, *.vcxproj, *.vcxproj.filters) של Visual Studio 2019, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא **אזהרות** או **שגיאות**.

הגישו בנוסף את תיקיית ה-Debug עם ה-Exe-ים.

אין צורך להגיש את תיקיית ה-vs.

דגשים

הקפידו על קוד קריא ומתועד.

עבדו באיטרציות. בדקו את הקוד שכתבתם לפחות בסוף כל סעיף.

זכרו להשתמש בכלי הדיבוג שה-IDE מספק.

הפורום עומד לשירותכם. אנו מעודדים אתכם לנסות תחילה לחפש תשובות באינטרנט, כאשר מדובר בשאלות תכנות כלליות.

בהצלחה!

הנחות והנחיות

- הימנעו משימוש במספרי קסם ומחרוזות מפורשות בקוד עצמו – רכזו את כל זה בקובץ ה-HardCodedData.h שיכיל את כל ה-#define-ים שלכם.

- זכרו לשחרר משאבים כגון: זכרון דינאמי (malloc, HeapAlloc וכו'), Kernel Objects (HANDLE-ים של קבצים, Event-ים וכו') וכל משאב אחר שהתוכנית שלכם תופסת לפני שהיא מסיימת את ריצתה ובפרט במקרה של שגיאה – גם במקרה של שגיאה יש לשחרר את כלל המשאבים בצורה best effort-ית, כלומר אם CloseHandle נכשלת פשוט נמשיך הלאה לנסות לשחרר את שאר המשאבים ולא נסיים בפתאומיות.

- בכל מקרה שלא מוגדרת ההתנהגות הנדרשת מכם אתם רשאים לבחור בכל התנהגות **סבירה**. התנהגות סבירה כוללת בתוכה סיום אלגנטי של התוכנית (לא לקרוס) תוך הדפסת הודעת שגיאה מתאימה ושחרור משאבים.

- יש לחלק את הקוד שלכם למודולים ואת המודולים לחלק לפונקציות בצורה הגיונית.

- **חובה** לקרוא על כל אחת מהפונקציות שמתוארות בהמשך באמצעות הקישורים/חיפוש בגוגל לצורך הבנה מיטבית שלהן לתרגיל זה ולטובת התרגילים בהמשך הקורס. שימוש בפונקציה בצורה לא נכונה/לא מוגדרת תגרור הורדת ניקוד גם אם התוכנית כן מתקמפלת ועובדת כראוי.

- פעולות ההצפנה והפענוח הן על אותיות אנגליות קטנות וגדולות ומספרים בלבד. סימני פיסוק, רווחים, ירידות שורה, ו-emoji-ים לא מושפעים מפעולת התוכנית ונשארים כמו שהם.

- ניתן להניח שמספר ה-thread-ים שתתבקשו ליצור לא יעלה על המספר המקסימלי של אובייקטים שניתן להמתין עליהם עם WaitForMultipleObjects.

- אינכם צריכים לבדוק אם מספר התהליכים או ה-thread-ים שנתבקשתם ליצור עובר את המספר המותר על ידי מערכת ההפעלה וניתן להניח שלא תקבלו קלט שעובר גבול זה.

- יש להגיש את התרגיל לאחר השלמת המדרגה האחרונה.

- על הקוד להתקמפל out-of-the-box, כלומר ללא שום התערבות מצד הבודק למעט פתיחת הפרויקט וביצוע Build.

- על הפרויקט להתקמפל ללא שגיאות או אזהרות למיניהן ב-Debug x86.

- יש לתעד ב-Header-ים מעל כל פונקציה מה היא מקבלת, מה היא מחזירה ומה היא עושה וכן לחלק את ה-Header לאזורים (עבור include-ים, עבור #define-ים, עבור פונקציות וכו'). בקבצי ה-Source יש לתעד בגוף הפונקציות על מנת שיהיה קל להבין מה הקוד עושה.

- בפונקציות מיוצאות **יש** לבדוק את תקינות הקלט (לוודא שמצביעים תקינים, גדלים חיוביים וכו') בתחילת הפונקציה. **בפונקציות פנימיות** אין צורך לבצע בדיקות אלו אך מומלץ להשתמש ב-assert-ים (מוגדרים ב-assert.h) על מנת להקל את תהליך הפיתוח והבדיקה.

טיפ של אלופים: מומלץ **מאוד** לעבוד עם שירות Version Control כלשהו למשל כמו GitHub על מנת לשמור את ההתקדמות שלכם וזאת משום שהתרגיל הוא איטרטיבי. ביצוע commit לאחר השלמת

מדרגה בתרגיל יעזור לכם לעקוב אחר ההתקדמות שלכם, לזהות בעיות ויאפשר לכם לחזור לנקודה האחרונה שבה הקוד שלכם עבד במקרה שהסתבכתם ואתם לא מצליחים למצוא את הבעיה. למי מכם שמעולם לא השתמש ב-Git ו-GitHub עדיף מאוחר מאשר אף פעם.

רקע כללי

בתרגיל זה תכתבו תכנית שיודעת לפענח קבצים בצורה מקבילית. נשתמש בהצפנה פשוטה הידועה בשם "צופן קיסר". ה"מפתח" של ההצפנה הוא מספר שלם כלשהו. ההצפנה פועלת על ידי הזזה של כל אות בצורה מעגלית על פני הא"ב האנגלי. נפענח מספרים באותה הצורה רק שהמעגל הוא 0-9, כלומר מודולו 10 במקום 26.

כלומר, עבור אות קטנה ('a', 'b',...,'z') שנסמן ב-letter ומפתח key הפענוח יהיה:

'a' + (letter - 'a' **-** key) % 26

וההצפנה תהיה:

'a' + (letter - 'a' **+** key) % 26

למשל, עבור האות 'y' והמפתח 3 נקבל:

'a' + ('y' - 'a' - 3) % 26 = 'a' (24 - 3) % 26= 'a' + 21 % 26 = 'a' + 21 = 'v'

בצורה דומה עבור אות גדולה ('A', 'B',...,'Z') שנסמן ב-letter ומפתח key הפענוח יהיה:

'A' + (letter - 'A' **-** key) % 26

וההצפנה תהיה:

'A' + (letter - 'A' **+** key) % 26

למשל, עבור האות 'Y' והמפתח 5 נקבל:

'A' + ('Y' - 'A' **-** 5) % 26 = 'A' + (24 **-** 5) % 26 = 'A' + 19 % 26 = 'A' + 19 = 'T'

עובר תו שהוא ספרה ('0', '1', '2',..,'9'), אותה נסמן ב-digit, ומפתח key הפענוח יהיה:

'0' + (digit - '0' **-** key) % 10

וההצפנה תהיה:

'0' + (digit - '0' **+** key) % 10

למשל, עבור הספרה '6' והמפתח 2 הפענוח יהיה:

'0' + ('6' - '0' **-** 2) % 10 = '0' + (6 - 2) % 10 = '0' + 4 % 10 = '0' + 4 = '4'

מומלץ לעבוד לפי סדר המדרגות.

מדרגות

מדרגה 1

פתחו פרויקט חדש ב-Visual Studio בשם Caesar. הפרויקט יהיה executable כלומר, ביצוע build ייצור קובץ exe הניתן להרצה.

כתבו תוכנית המקבלת נתיב לקובץ ומספר. הנתיב יהיה לקובץ לפענוח והמספר יהיה המפתח.

על התוכנית ליצור קובץ חדש באותו הנתיב ששמו יהיה "decrypted.txt".

לדוגמה: אם יש לנו קובץ בנתיב C:\temp\top_secret_file.txt שתוכנו הוא: Uijt jt uif Xbz
וידוע שהוא הוצפן בצופן קיסר עם המפתח 1 אז הפקודה ב-Command Line של Windows שתפענח אותו תהיה:

```
Caesar.exe C:\temp\top_secret_file.txt 1
```

התוכנית תיצור קובץ בנתיב C:\temp\decrypted.txt ותוכנו יהיה "This is the Way"

הערות

1. הקלט יכול להכיל אותיות אנגליות קטנות/גדולות, ספרות, סימני פיסוק וסימנים מיוחדים. יש לטפל רק בתווים שהוגדרו (אותיות אנגליות קטנות/גדולות וספרות).
2. אם קיים כבר קובץ בנתיב זה יש לדרוס אותו לפני תחילת הפענוח.

מדרגה 2

כעת נהפוך את התכנית שלנו למקבילית. נוסיף לארגומנטים שהתוכנית מקבלת בשורת הפקודה את מספר ה-thread-ים שאנחנו רוצים שהיא תשתמש בהם. לדוגמה:

```
Caesar.exe "C:\temp\top_secret_file.txt" 1 4
```

בדוגמה הזאת התוכנית תיצור 4 thread-ים לביצוע הפיענוח.

התוכנית תבדוק את מספר השורות בקובץ ותחלק את השורות בצורה שווה (עד כדי ± 1) למספר חלקים ששווה למספר ה-thread-ים שהיא תפתח.

כל thread יקבל את הארגומנטים הבאים:

1. ה-path המלא של קובץ הקלט
2. ה-path המלא של הקובץ הפלט
3. המיקום בקובץ ממנו החלק שלו מתחיל בקובץ הקלט.
4. המיקום בקובץ שמכיל את סוף החלק שלו בקובץ הקלט.
5. מפתח הפיענוח.

כל thread יקרא מקובץ הקלט את החלק שלו בלבד, יפענח אותו ויכתוב אותו לקובץ הפלט במקום המיועד לו. שימו לב שלא אמורה להיות שום חפיפה בין ה-thread-ים לכל אורך התהליך ולכן אין צורך בסנכרון פעולות הקריאה והכתיבה. שימו לב לפתוח כל אחד מהקבצים בצורה המתאימה כדי למנוע שגיאות ודריסות. בפרט יש לפתוח את הקבצים בצורה שתאפשר גישה במקביל, למשל ב-CreateFile זה בא לידי ביטוי בדגל dwShareMode שבו ניתן לבחור אם לאפשר שיתוף של קריאה/כתיבה/מחיקה.

ה-thread הראשי ימתין לסיום פעולת **כל** ה-thread-ים **אך** לא יותר ממספר קבוע וסופי של שניות, כלומר אין להשתמש ב-INFINITE. כלומר יש להשתמש ב-WaitForMultipleObjects כדי להבטיח שה-thread הראשי נותן ל-thread-ים המשניים זמן לעבוד. אם אחרי הזמן הזה יש thread-ים שטרם סיימו הדבר נחשב לשגיאה.

לדוגמה, אם יש לנו קובץ בנתיב כמו מקודם ותוכנו הוא:

```
Efbs Ns. Kpoft,\n

Xf xpvme mjlf up dpohsbuvmbuf zpv po zpvs qspnpujpo.\n

Cftu sfhbset,\n

Uif Nbobhfnfou
```

שימו לב שיש תו של ירידת שורה בסוף כל שורה (\n) והוא מופיע כאן כדי להזכיר שירידת שורה היא גם תו. כאשר נפתח את הקובץ ב-notepad לא נראה את התו "\n" אלא נראה שיש ירידת שורה.

ונריץ את התוכנית שלנו עם שורת הפקודה הבאה:

Caesar.exe "C:\temp\top_secret_file.txt" 1 2

התוכנית תיצור 2 thread-ים.

ה-thread הראשון יקבל 0 כמיקום של תחילת החלק שלו, 68 כמיקום של סוף החלק ו-1 כמפתח. ה-thread יקרא מקובץ הפלט את שתי השורות הראשונות (69 התווים הראשונים שבקובץ) ויפענח את התוכן שלהן ויכתוב אותו לקובץ הפלט בין המיקומים 0 ל-68:

Dear Mr. Jones,\n

We would like to congratulate you on your promotion.\n

ה-thread השני יקבל 69 כמיקום של תחילת החלק שלו בקובץ הקלט, 96 כמיקום של סוף החלק שלו ו-1 כמפתח. בזמן שה-thread הראשון יקרא את שתי השורות הראשונות ה-thread השני יקרא את שתי השורות האחרונות, יפענח אותן ויכתוב אותן בין המיקומים 69 ל-96 בקובץ הפלט.

Best regards,\n

The Management

סה"כ קובץ הפלט יכיל את התוכן הבא:

Dear Mr. Jones,\n

We would like to congratulate you on your promotion.\n

Best regards,\n

The Management

כאשר התו "\n" מופיע כאן מפורשות על מנת להדגיש שירידת שורה היא גם תו.

הערות

1. שימו לב שכדי למקבל את העבודה בין ה-thread-ים לכל אחד צריך להיות HANDLE משלו לקובץ ולכן כל thread צריך לקרוא ל-CreateFile בכל אחד מה-thread-ים.
2. על מנת להימנע מדריסות מומלץ מאוד לפתוח קובץ פלט ריק בגודל המתאים לפני יצירת ה-thread-ים על מנת להקל את העבודה. ניתן להשתמש ב-SetFilePointer ו-SetEndOfFile כדי להשיג מטרה זו.
3. יש להשתמש ב-SetFilePointer על מנת לשלוט לאן WriteFile מבצעת את הכתיבה שלה.
4. זמני ההמתנה בתרגיל צריכים להיות בסדר גודל של מספר שניות בודדות ולכל היותר -10 20 שניות. אם הקוד שלכם עובד רק כאשר הזמנים שלכם ארוכים מאוד כנראה שיש לכם בעיות יעילות חמורות.

מדרגה 3

הגיע העת להוסיף תמיכה בהצפנה. הוסיפו לתוכנית שלכם עוד ארגומנט בשורת הפקודה שמציין האם יש להצפין או לפענח. פעולת ההצפנה היא הפעולה ההפוכה מפעולת הפיענוח ועל כן לכל קלט התוכנה שלכם חייבת לקיים שפענוח על קובץ שהוצפן עם אותו המפתח יהיה זהה לקובץ המקורי. הדגל להצפנה/פענוח יהיה -d/-e ותמיד יופיע בסוף שורת הפקודה.

כאשר התוכנית שלכם תקבל את דגל ההצפנה e- היא תבצע הצפנה על הקובץ באופן מקבילי ותכתוב את התוצאה לקובץ פלט בשם "encrypted.txt" וכאשר התכנית שלכם תקבל את דגל הפענוח עליה לפעול כמו בשלב במדרגה הקודמת.

לדוגמה: נתון קובץ טקסט שנתיבו המלא הוא C:\temp\plaintext.txt ותוכנו הוא:

Morning!

Nice day for fishing ain't it

Hu ha!

אזי רצף הפקודות הבאות ב-Command Line של Windows:

```
Caesar.exe "C:\temp\plaintext.txt" 1 3 -e

Caesar.exe "C:\temp\encrypted.txt" 1 3 -d
```

הפלט של הפקודה הראשונה יהיה הקובץ C:\temp\encrypted.txt והפלט של הפקודה השנייה יהיה הקובץ C:\temp\decrypted.txt ותוכנו יהיה זהה לתוכן של C:\temp\plaintext.txt.

הערות

1. אחד מהדגלים ורק אחד מהם חייב להופיע בשורת הפקודה, אם שני הדגלים מופיעים או אף אחד מהם לא מופיע יש להתייחס לכך כמו אל כל שגיאה אחרת - יש להדפיס הודעת שגיאה למסך ולסיים.
2. השימוש ב-"e-" ו-"d-" כדגלים הינו חובה - אין לתת לדגלים שמות אחרים.
3. הגדלים תמיד יגיעו בסוף שורת הפקודה - הארגומנט הרביעי והאחרון. אם זה לא המצב - זו שגיאה ויש לפעול בהתאם.
4. כדאי קודם לממש את ההצפנה ללא מקביליות (כמו המימוש של הפענוח במדרגה 1), לבדוק את הקוד שלכם מול הפענוח שכבר כתבתם ורק אחרי שאתם רואים שאין בעיות להוסיף את המקבול.
5. אם קיים כבר קובץ בנתיב זה יש לדרוס אותו לפני תחילת ההצפנה.

מדרגת בונוס (10 נקודות)

שנו את הקוד שלכם ככה שכל ה-thread-ים יתחילו לעבוד באותו הזמן. מכיוון שה-thread-ים לא נוצרים בדיוק באותו הזמן זה אומר שעל כל אחד מה-thread-ים להמתין לאיזשהו סימן שאומר שכל ה-thread-ים נוצרו ורק אז להתחיל לעבוד. מומלץ לקרוא על CreateSemaphore\CreateEvent לצורך הסנכרון אך אתם רשאים לממש את המנגנון כרצונכם כל עוד הוא לא פוגע באופן שבו משתמשים בתוכנית.