

100
2
98

מבחן במערכות הפעלה

זועד א' סמסטר ב' תשס"ט, 15.7.2009
פרופ' חזי ישורון

הוראות

משך הבחינה שלוש שעות. יש לענות על כל השאלות. יש לנמק תשובות כן/לא.
יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד
מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת
ללא טופס עזר תפסל. תשובות במחברת הבחינה לא תבדקנה.
יש למלא מספר סידורי (מספר מחברת) ומספר ת"ז על כל דף של טופס הבחינה.
אסור השימוש בחומר עזר כלשהו, פרט למחשבוניו.

בהצלחה!

1. (30 נקודות)

א במערכת דיסק בגודל 1MB, בלוקים של 1KB, CACHE של בלוק אחד.
קובץ נמצא בבלוקים (משמאל לימין) 10,900, 20,500. הנח שהדיסק הוא
לינארי, כלומר הוא רצף מבלוק 0 לבלוק האחרון. ה-DIRECTORY נמצא
בבלוק 50, ו-FAT בתחילת הדיסק (בלוק 0) וכל כניסה בו היא של שני בתים.
ראש הדיסק נמצא כרגע בבלוק 51. כמה בלוקים יעבור הראש כדי לכתוב מידע
נוסף בסוף הקובץ בבלוק 600 אם המערכת משתמשת ב-LOOK ?

הנחה: הדיסק הוא לינארי, כלומר הוא רצף מבלוק 0 לבלוק האחרון. ה-DIRECTORY נמצא בבלוק 50, ו-FAT בתחילת הדיסק (בלוק 0) וכל כניסה בו היא של שני בתים. ראש הדיסק נמצא כרגע בבלוק 51. כמה בלוקים יעבור הראש כדי לכתוב מידע נוסף בסוף הקובץ בבלוק 600 אם המערכת משתמשת ב-LOOK ?

ב האם יתכנו מקרים בהם כדאי להשתמש בתוך מערכת ההפעלה בהמתנה
בלולאה לאירוע קלט פלט (ולא למשל בפסיקה) ?

הנחה: הדיסק הוא לינארי, כלומר הוא רצף מבלוק 0 לבלוק האחרון. ה-DIRECTORY נמצא בבלוק 50, ו-FAT בתחילת הדיסק (בלוק 0) וכל כניסה בו היא של שני בתים. ראש הדיסק נמצא כרגע בבלוק 51. כמה בלוקים יעבור הראש כדי לכתוב מידע נוסף בסוף הקובץ בבלוק 600 אם המערכת משתמשת ב-LOOK ?

2. (20 נקודות)

אני מניח שביכולתכם לכתוב מערכת המנהלת חוטים בתוך התהליך שלכם
ללא שימוש בקריאות למערכת ההפעלה. בהנחה שרמת הכתיבה זהה וקיימות
אותן פונקציות, האם מערכת כזו תהיה יעילה יותר או פחות ממערכת הניהול
של מערכת ההפעלה?

30

4. (30 נקודות) עליך לממש במערכת לרישום קבצי יומן פונקציונאליות לתמיכה ב- UNICODE STRINGS לקובץ טקסט מכל מקום בתכנית שלך. משתמש יכול לפתוח ולקרוא את קובץ היומן תוך שימוש ב- WINDOWS NOTEPAD, בזמן שהתכנית שלך עדיין רצה. עליך לממש את שלשת הפונקציות הבאות:

Start Logging

- **void StartLogFile(LPCTSTR logFilePath)** should initialize logging mechanism and prepare log file for writing. If log file with this name already exist it should be deleted. No other calls will be made before **StartLogFile** returns.
- **void StopLogFile()** should finish logging and cleanup all program structures (but keep the file). No logging calls will be made after this function is invoked.
- **void LogStringToFile(LPCTSTR log_string)** will be called several times possibly from different threads of the program.

a) Fill the gaps in the code below. **Don't add code lines or additional functions parameters.** You may assume that no system errors or timeouts will occur. Review all of the code before filling any gaps

```
#define UNICODE_SIGNATURE WORD(oxFEFF)
#define SHARING1 FILE_SHARE_READ
#define SHARING2 FILE_SHARE_WRITE
#define FILE_ACC1 GENERIC_READ
#define FILE_ACC2 GENERIC_WRITE
#define FILE_DISP1 CREATE_NEW
#define FILE_DISP2 OPEN_ALWAYS
#define FILE_DISP3 CREATE_ALWAYS
#define FILE_DISP4 OPEN_EXISTING
#define FILE_FLAG1 FILE_FLAG_RANDOM_ACCESS
#define FILE_FLAG2 FILE_FLAG_SEQUENTIAL_SCAN

// ... SECTION guard;
// ... hFile;

void StartLogging(LPCTSTR fileName){
    DWORD dwWritten;
    WORD buffer= UNICODE_SIGNATURE;
    InitializeCriticalSection(&critSec);
    hFile = CreateFile(fileName,
        FILE_ACC_2,
        SHARING_1,
        NULL,
        FILE_DISP_4,
        FILE_FLAG_2,
```

15

```

        NULL);

WriteFile(hFile,
        &buffer,
        sizeof(buffer),
        NULL,
        NULL);
}

void StopLogging()
{
    LeaveCriticalSection(&guard);
}

void LogString(LPCTSTR log_string)
{
    DWORD dwWritten;
    DWORD dwLength;
    DWORD dwBytesToWrite;
    dwLength = _tcslen(log_string);
    dwBytesToWrite = 2 * dwLength;
    EnterCriticalSection(&guard);
    WriteFile(hFile,
        log_string,
        dwBytesToWrite,
        &dwWritten,
        NULL);
    LeaveCriticalSection(&guard);
}

```

b) Now you want to use a single log file for **several processes** that might try to write into it simultaneously. You have decided to pack the log file functionality into the DLL. Complete the modified code for 3 functions. You may assume that no system errors or timeouts will occur. Review all of the code before filling any gaps. **Don't add code lines or additional functions parameters.** Assume that log file already exists so you only need to add new entries into it.

עמוד 5 מתוך 6 מספר סידורי: 88 מספר ת"ז:

```
#define UNICODE_SIGNATURE WORD(0xFFEF)
#define SHARING_FLAG1 FILE_SHARE_READ
#define SHARING_FLAG2 FILE_SHARE_WRITE
#define FILE_ACC1 GENERIC_READ
#define FILE_ACC2 GENERIC_WRITE
#define FILE_DISP1 CREATE_NEW
#define FILE_DISP2 OPEN_ALWAYS
#define FILE_DISP3 CREATE_ALWAYS
#define FILE_DISP4 OPEN_EXISTING
#define FILE_FLAG1 FILE_FLAG_RANDOM_ACCESS
#define FILE_FLAG2 FILE_FLAG_SEQUENTIAL_SCAN

typedef struct
{
    TCHAR file_name[256];
    TCHAR guard_name[256];
    HANDLE hGuard;

    DWORD dwTimeout;
} MY_LOG_FILE;

extern "C" {
    __declspec(dllexport) void StartLogging(MY_LOG_FILE* pLF);
    __declspec(dllexport) void StopLogging(MY_LOG_FILE* pLF);
    __declspec(dllexport) void LogString(MY_LOG_FILE* pLF, LPCTSTR log_string);
}

void StartLogging(MY_LOG_FILE* pLF){
    pLF->hGuard = CreateMutex(NULL,
                               FALSE,
                               pLF->guard_name);
}

void StopLogging(MY_LOG_FILE* pLF){
    CloseHandle(pLF->hGuard);
}

void LogString(MY_LOG_FILE* pLF, LPCTSTR log_string){
    DWORD dwWritten;
    DWORD dwLength;
    DWORD dwBytesToWrite;

    HANDLE hFile;

    dwLength = _tcslen(log_string);
    dwBytesToWrite = dwLength;
    WaitForSingleObject(pLF->hGuard, pLF->dwTimeout);
```

