

## פתרון מבחן במערכות הפעלה

חזי ישרון וסיון טולדו, מועד א' סמסטר ב' תש"ע, 7 ביולי 2010

### שאלה 1

- א. תוכנית תקינה, הפונקציות נקראות מחוט אחד ואין בו-זמניות כלל.
- ב. לא תקין. שתי הפונקציות נקראות מחוטים שונים, משנות את אותו מבנה נתונים, אבל נועלות מנעולים שונים.
- ג. תקין. הפונקציה f1 שנקראת על ידי שני חוטים מגינה על מבנה הנתונים הסטטי על ידי המנעול m1, ואילו הפונקציה f3 שמופעלת מחוט אחר פועלת על מבנה נתונים אחר, מקומי (על המחסנית).
- ד. התוכנית תקינה. כל הפעלה של f3 מטפלת במבנה נתונים שונה, על המחסנית, ולכל חוט מחסנית משלו.
- ה. התוכנית לא תקינה. הפונקציה f4 נועלת את m1 אבל לא משחררת אותו, אלא משחררת את m2. זה לא תקין וזה יגרום להפעלה של f1 על ידי חוט אחר להמתין לעד.
- ו. התוכנית תקינה. f1 ו-f5 פועלות על אותו מבנה נתונים, אבל מגינות עליו על ידי m1. f3 פועלת על מבנה אחר. הפונקציות f1 ו-f3 נועלות רק מנעול אחד ולכן לא יכול להיווצר deadlock.

### שאלה 2

הערה כללית: בגלל חוסר בהירות בניסוח השאלה, הוספנו 6 נקודות לכולם, 2 נקודות עבור פתרון מסובך מדי בסעיף א, ו-4 נקודות בסעיף ב' עבור פתרון לא יעיל או פתרון שמניח שהיחידות של 1GB אינן רצופות. הטקסט של השאלה לא היה ברור מספיק כדי להבהיר שאלה פתרונות שלא יתקבלו, ולכן הוספנו נקודות על מנת לפצות על הפחתה של 2 נק' ב-א' ו-4 נק' ב-ב'. ערעורים על ההפחתות הללו לא יתקבלו משום שהנקודות הוספו כבר בחזרה.

- א. בכתיבה כותבים למערך הפטריות בכתובת הנתונה. בקריאה קוראים את המידע ל-RAM של היחידה, כותבים אותו חזרה לפטריות (כי הוא נהרס), ומחזירים מה-RAM את המידע למחשב שביקש אותו. אפשר ליעל את הפתרון על ידי שימוש ב-RAM של היחידה כזכרון מטמון, כך שלא בכל בקשה צריך לגשת לפטריות.
- ב. פתרון לא יעיל אבל נכון: כמו א' אבל ביחידות של 1GB. כאשר קוראים או כותבים בלוק של 1024 בתים, היחידה קוראת 1GB, מאתחלת, וכותבת מחדש (עם בלוק אחד שונה במקרה של כתיבה). פתרון יותר יעיל משתמש במנגנון דומה לשל LFS. שומרים ב-RAM שני מערכים. אחד מסמן עבור כל בלוק של 1024 בתים של פטריות האם הוא מאתחל, מכיל מידע, או הרוס. השני הוא מערך של מצביעים (של 29 ביטים כל אחד) שמצביע על המיקום הנוכחי של כל בלוק שהמחשב החיצוני יכול לבקש; המיקום הוא שרירותי במערך הפטריות. בכתיבה, כותבים לבלוק מאתחל כלשהו ומעדכנים את המצביע. בקריאה, מעתיקים 1024 בתים ל-RAM, כותבים למקום מאתחל, ומעדכנים את המצביע. כאשר נשארות רק 2 יחידות מאתחלות, קוראים יחידה של 1GB, כותבים את הבלוקים שבה לבלוקים מאתחלים, ומאתחלים אותה. תמיד יש יחידה כזו שבה יותר מחצי מהבלוקים כבר הרוסים ולא צריך לכתוב אותם מחדש (כי במוצע כל יחידה מכילה 512K בלוקים הרוסים).
- ג. שני הפתרונות של ב' עובדים כאן. אפשר לתמוך בכמעט 1024GB. אם משתמשים בפתרון השני של ב', הוא כבר לא יעיל, כי בהחלט יתכן מצב שבו נצטרך לקרוא 1GB, לכתוב מחדש, וכל זה כדי לקבל בלוק מאתחל אחד במקום בלוק הרוס אחד. כלומר אין כאן פתרון יעיל.

### שאלה 3

- א. התהליך בצד השני לא יקבל מידע נוסף מהערוץ, ואחר זמן מה לא יוכל יותר לשלוח מידע, כי חוצץ השליחה שלו וחוצץ הקבלה בצד השני יתמלאו.
- ב. התהליך בצד השני יקבל על system call לערוץ ה-TCP קוד שגיאה מכיון שמערכת ההפעלה שלו תקבל מנת reset מהמחשב שהתהליך בו עף. מבחינת המחשב המרוחק הערוץ לא קיים יותר.
- ג. תוכנית שמשתמשת נכון במשתני תנאי חייבת להשתמש גם במנעולים. קיבלנו גם תשובות "לא נכון" שיתכן deadlock בגלל שיש חוסים שמחכים ל-signal אבל אף אחד לא שולח אותו על ידי signal או broadcast.
- ד. לא נכון. NFS לא מבטיחה קונסיסטנטיות ולכן יתכן ששני לקוחות יסמנו בו-זמנית בקובץ שכרטיס טיסה מסויים נרכש, או בעיות דומות.
- ה. נכון. ביוניקס ולינוקס תהליך חדש ממשיך להריץ את התוכנית שמריץ התהליך שיצר אותו (fork), ולכן הוא יכול להמשיך בפעילות דומה בעלות נמוכה.
- ו. נכון. ככל שמספר הדיסקים עולה הסיכוי שאחד מהם יתקלקל עולה, ולכן חשיבות השימוש בקוד לתיקון שגיאות עולה.

### שאלה 4

```
#include "stdafx.h"
#include <windows.h>

#define PORT_NO 8081
#define LISTEN_BACKLOG 5
#define MAX_STR_LEN 256
#define IP_ADDRESS "127.0.0.1"

DWORD WINAPI HandleConnection(LPVOID pParam){

    SOCKET hDataSocket;
    int iPos=0,nSend=0,nLeft=0;
    char buffer[MAX_STR_LEN];

    srand(GetTickCount()^GetCurrentThreadId());

    hDataSocket=(SOCKET)pParam;
    nLeft=1+sprintf(buffer,"your lucky number is %d\r\n",rand());

    while(nLeft){
        nSend=send(hDataSocket,buffer+iPos,nLeft,0);
        nLeft-=nSend; iPos+=nSend;
    }
    closesocket(hDataSocket);
    return 0;
}

int _tmain(int argc, _TCHAR* argv[]){

    WSADATA wsaData;
    WSASStartup(MAKEWORD(2,2),&wsaData);
    SOCKET hSocket;
```

```
SOCKET hDataSocket;
SOCKADDR_IN addr;
HANDLE hThread;

hSocket=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
addr.sin_family=AF_INET;
addr.sin_port=htons(PORT_NO);
addr.sin_addr.s_addr=inet_addr(IP_ADDRESS);

bind(hSocket,(SOCKADDR*)&addr,sizeof(addr));
listen(hSocket,LISTEN_BACKLOG);

while(1){
    hDataSocket=accept(hSocket,NULL,0);
    hThread=CreateThread(NULL,0,HandleConnection,
                        (LPVOID)hDataSocket,0,NULL);

    CloseHandle(hThread);
}
return 0;
}
```