

מבחן במערכות הפעלה

חזי ישורון וסיון טולדו, מועד ב' סמסטר ב' תש"ע 1 בספטמבר 2010

כל הזכויות שמורות

אין להעלות את הקובץ או

כל נגזרת ממנו לאתרי אינטרנט

או להפיצו בכל דרך אחרת ללא

רשות המחברים.

הוראות

משך הבחינה שלוש שעות. לא תינתן הארכה.

יש לענות על כל השאלות.

בשאלות אמריקאיות, יש לסמן את התשובה הנכונה בעיגול על טופס הבחינה ולנמק כשנדרש נימוק. בשאלות שבהן יש צורך לנמק, תשובה ללא נימוק לא תזכה באף נקודה.

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, כולל מחשבוניו או כל מכשיר אחר פרט לעט.

בהצלחה!

שאלה 1 (24 נקודות)

א. ב-TCP, חוצץ שליחה גדול עוזר לנצל היטב ערוץ תקשורת עם רוחב פס גדול והשהיה (latency) גדולה. נכון/לא נכון

ב. ב-TCP, חוצץ קבלה גדול עוזר לנצל היטב ערוץ תקשורת עם רוחב פס גדול והשהיה (latency) גדולה. נכון/לא נכון

ג. אפשר לפתח פרוטוקול חלופי ל-TCP שמי שישתמש בו יקבל בדרך כלל רוחב פס יותר גדול ממי שימשיך להשתמש ב-TCP. נכון/לא נכון

ד. תוקף (hacker) יכול ליצור ערוץ TCP ולגרום לשרת המותקף להתחיל לשלוח נתונים גם אם בכל המנות (packets) שהוא שולח, כתובת ה-IP של השולח אינה כתובת IP של המחשב שממנו נשלחות המנות (כדי שיהיה קשה לאתר את התוקף). השרת מממש את TCP ואין בו באגים. נכון/לא נכון

ה. TLB גדול מפחית את העלות של טיפול בחריג דף. נכון/לא נכון

ו. כאשר שני תהליכים משתפים מסגרת זיכרון פיזית אחת, היא תמיד ממופה בשני התהליכים לאותו טווח כתובות וירטואליות. נכון/לא נכון

ז. ב-NQNF, כאשר אין חוזים על קובץ, לא ניתן להעניק חוזה כתיבה עם הטמנה. נכון/לא נכון

ח. יש תהליכים שאין להם מפת זיכרון. נכון/לא נכון

שאלה 2 (24 נקודות)

חברת הסטרט-אפ GRQ פיתחה מערכת קבצים מתקדמת. המערכת מחלקת את הדיסק המגנטי לאיזורים בגודל 1,000,000 בתים כל אחד. הקבצים מחולקים לבלוקים גדולים בגודל 1,000,000, פרט לסוף הקובץ שמחולק לבלוקים קטנים בגודל 1000 בתים. כל איזור בדיסק מכיל בלוק גדול של קובץ (אחד) או הרבה בלוקים קטנים, לא בהכרח כולם מאותו קובץ. המערכת החליפה מערכת קבצים יותר פשוטה שבה קבצים היו מחולקים לבלוקים בגודל 1000 בתים והדיסק היה מחולק לאיזורים קטנים שגם הם בגודל 1000 בתים.

בשאלה זו אפשר להניח שגודל קובץ הוא תמיד כפולה של 1000 בתים ושגודל הדיסק הוא 100Gbytes.

א. ציין/ציני שתי יתרונות משמעותיים שעשויים להיות למערכת החדשה לעומת המערכת הישנה.

ב. גרסה אחת של מערכת הקבצים פעלה באופן הבא. כאשר נכתב לסוף קובץ בלוק שהגדיל את הקובץ לכפולה של 1,000,000 בתים, המערכת פינתה איזור שלם בדיסק מבלוקים קטנים (העתיקה אותם לאיזורים אחרים) והעתיקה את הבלוקים הקטנים שבסוף הקובץ לאיזור הזה, תוך יצירת בלוק גדול אחד. מהו בערך מספר הגישות המקסימלי לדיסק (הזזות הראש) שפעולה כזו דורשת? מותר להניח שמערכת הקבצים שומרת ב-RAM מפה שמציינת איזה בלוקים בדיסק בשימוש.

ג. המפתחת לוגית טענה שאפשר לשפר את ביצועי המערכת אם היא תשמור תמיד על התכונה הבאה: אם יש במערכת N בלוקים קטנים, אזי יש לפחות $2N/1,000,000$ איזורים בדיסק שמוקצים לבלוקים קטנים (כלומר האיזורים שמוקצים לבלוקים קטנים יכולים להכיל כפליים ממספר הבלוקים הקטנים שבאמת יש). הסבר/הסבירי איך התכונה הזו תשפר את הביצועים של מיזוג הבלוקים הקטנים שבסוף קובץ לבלוק גדול אחד. אין צורך להתייחס לשאלה איך המערכת תשמור על התכונה.

שאלה 3 (22 נקודות)

עליך לפתח ייצוג וקוד עבור מבנה נתונים בשם BoundedUpDown שמייצג מספר שלם בין 0 לחסם עליון נתון. על מבנה הנתונים הזה יש שלוש פעולות: initialize, שמאתחלת אותו לערך 0 וקובעת את החסם העליון, up שמחזירה את ערכו הנוכחי ומגדילה אותו ב-1 אם אפשר, ו-down שמחזירה את ערכו הנוכחי ומקטינה אותו ב-1 אם אפשר. חוט שקורא ל-up או down כאשר אי אפשר לבצע את הפעולה מחכה עד שאפשר יהיה לבצע את הפעולה. המימוש צריך להשתמש ב-mutex ו-condition variable (אחד או יותר מכל סוג); התחביר של הפעולות על mutex ו-condition variable לא חייב להיות מדוייק (כלומר זהה לזה של לינוקס או חלונות). אין צורך לאתחל mutex ו-condition variables. המימוש צריך להיות פשוט ככל האפשר. יש לענות על השאלה הבאה ולמלא את הקוד.

האם המימוש שלך הוגן? נמק'!

ועכשיו לקוד:

```
typedef struct {

} BoundedUpDown;

BoundedUpDown* init(int bound) {
```

```
}  
  
int up(BoundedUpDown* x) {
```

```
}  
  
int down(BoundedUpDown* x)
```

```
}
```

שאלה 4 (30 נקודות)

השלם את הקוד הבא. הקוד מממש את המשחק "אבן-נייר ומספריים" לשני שחקנים שמיוצגים על ידי שני תהליכים שרצים על אותו מחשב. התהליכים מתקשרים ביניהם בעזרת זיכרון משותף ועצמי סינכרון. בכל תור, כל שחקן בוחר מבין שלוש אפשרויות: 0 לאבן, 1 לנייר, 2 למספריים. אם שני המשתתפים בחרו את אותה בחירה, התוצאה היא תיקו (tie). אחרת אבן מנצחת מספריים ומפסידה לנייר ומספריים מנצחים נייר. אחרי כל תור, הקשה על מקש במקלדת מעבירה את המשחק לתור הבא. אין צורך לטפל בשגיאות. יש להשתמש ב-INFINITE כחסם זמן בכל פונקציה שמחכה.

```

struct GAME{
    int iNumPlayers;
    int iMoves[2]; };

LPCTSTR results[]={_T("won\n"),_T("lost\n"),_T("tie\n")};

HANDLE hReady2Play;
HANDLE hGameOver;
HANDLE hGuard;
HANDLE hFileMap;
GAME* pGame;

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

    Initialize();
    while(1){
        _tprintf(_T("eneter your move?"));_tscanf(_T("%d"),&myMove);
        myPlayerNum=MakeMoveWhenReady(myMove);
        PrintResultWhenReady(myPlayerNum);
        LeaveGame();
        _tprintf(_T("press any key to play more\n"));_getch();
    }
    return 0;}

void Initialize(){
    hFileMap=_____ (NULL,NULL,PAGE_READWRITE,
        0,_____,_T("EX.MMF"));

    bool bLast=(_____==ERROR_ALREADY_EXISTS);
    pGame=_____ (hFileMap,FILE_MAP_READ|FILE_MAP_WRITE,
        0,0,_____);

    hReady2Play=_____ (NULL,TRUE,_____,
        _T("EX.R2P"));

    hGameOver=_____ (NULL,_____,_____,
        _T("EX.GO"));

    hGuard=_____ (NULL,_____,_____,
        _T("EX.GUARD"));

    if(bLast){
        _____ = _____
        _____ (hReady2Play);}}

int MakeMoveWhenReady(int myMove){
    int myPlayerNum;

    _____ (hReady2Play,_____);
    _____ (hGuard,_____);

    myPlayerNum=pGame->iNumPlayers++;
    pGame->iMoves[myPlayerNum]=myMove;
    if(myPlayerNum){
        _____ (hReady2Play);
        _____ (hGameOver);}
    _____ (hGuard);

    return myPlayerNum;}

```

עמוד 6 מתוך 6 מספר סידורי: _____ מספר ת"ז: _____

```
void PrintResultWhenReady(int myPlayerNum) {  
    _____ (hGameOver, _____);  
    _____ (hGuard, _____);  
    _tprintf (result [(pGame->iMoves [myPlayerNum]-pGame->  
        iMoves [1-myPlayerNum]+2) %3]);  
    _____ (hGuard);}
```

```
void LeaveGame () {  
    _____ (hGuard, _____);  
    if (0==--pGame->iNumPlayers) {  
        _____ (hGameOver);  
        _____ (hReady2Play);}  
    _____ (hGuard);}
```

מבחן במערכות הפעלה

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

הוראות

משך הבחינה שלוש שעות. לא תינתן הארכה.
יש לענות על כל השאלות.

בשאלות אמריקאיות, יש לסמן את התשובה הנכונה בעיגול על טופס הבחינה ולנמק כשנדרש נימוק. בשאלות שבהן יש צורך לנמק, תשובה ללא נימוק לא תזכה באף נקודה. 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, כולל מחשבוני או כל מכשיר אחר פרט לעט.

בהצלחה!

שאלה 1 (24 נקודות)

נתון קטע התוכנית הבא:

```
typedef struct {...} mystruct;
static mystruct sms;
mutex m1, m2;

void f1() {
    lock( m1 );
    f(&sms);
    unlock( m1 );
}

void f2() {
    lock( m2 );
    f(&sms);
    unlock( m2 );
}

void f3() {
    mystruct lms;
    lock( m2 );
    f(&lms);
    unlock( m2 );
}

void f4() {
    lock( m1 );
    f(&sms);
    unlock( m2 );
}

void f5() {
    lock( m1 );
    lock( m2 );
    f(&sms);
}
```

```
unlock( m1 );
unlock( m2 );
}
```

```
void f(mystuct* ms) {
    ... // manipulate the fields of the struct pointed to by ms; must
    be atomic
}
```

הפונקציה f משנה את השדות של מבנה נתונים שמצביע אליו מועבר אליה; השינויים הללו צריכים להיות אטומיים, אחרת מבנה הנתונים עלול להשתבש. הקריאות היחידות לפונקציה f הן אלה המופיעות כאן, והפונקציה f היא היחידה שניגשת למבנים מטיפוס `mystuct`. עבור כל אחד מהמקרים הבאים, ציין האם התוכנית תקינה או לא והסבר. פונקציות שמופיעות בקוד אבל לא בסעיף לא נקראות בתוכנית כלל (למשל $f3$ לא מופיעה בסעיף 1). תשובה ללא הסברים לא תקבל ניקוד כלל.

א. $f1$ ו- $f2$ נקראות מאותו חוט בתוכנית.

ב. $f1$ ו- $f2$ נקראות משני חוטים שונים בתוכנית.

ג. הפונקציה $f1$ נקראת על ידי חוטים $t1, t2$, והפונקציה $f3$ נקראת על ידי חוט $t3$.

ד. הפונקציה $f3$ נקראת על ידי חוטים $t1, t2$, ו- $t3$.

ה. הפונקציות $f1$ ו- $f4$ נקראות על ידי שני חוטים שונים בתוכנית.

ו. הפונקציות $f1, f3$, ו- $f5$ נקראות על ידי שלושה חוטים שונים בתוכנית.

שאלה 2 (24 נקודות)

חברת MoonCircle פיתחה טכנולוגיית אחסון מתקדמת שמבוססת על שימוש בפטריות מיקרוסקופיות. מבחינת התוכנית, לטכנולוגיה הזו יש את המאפיינים המיוחדים הבאים:

- הפטריות מסודרות במערך לינארי (ארוך מאוד, זו טכנולוגיה צפופה ביותר).
- כל פטרייה שומרת סיבית (ביט).

- הפטריות כל כך זעירות, שקריאת הנתון שכתוב עליהן משבש אותו באופן חמור. כלומר, אפשר לקרוא סיבית שנכתבה על פטריה רק פעם אחת; הקריאה עצמה הורסת את המידע המאוחסן.

המוצר של החברה הוא יחידת זיכרון שמחליפה את ה-RAM. החומרה בתוך היחידה הזו כוללת 1024Gbyte של פטריות אחסון, 6GByte של RAM רגיל, ומעבד קטן (וזיכרון לא נדיף ששומר את התוכנה של המעבד הזה). מחשב שמכיל יחידה כזו יכול לכתוב אליה ולקרוא ממנה בלוקים של 1024 בתים, כאשר כתובת ההתחלה של בלוק כזה היא כפולה של 1024 (כלומר הבלוקים זרים).

- א. הצע/הציעי מנגנון שאכן יאפשר להחליף יחידת זכרון RAM רגיל בגודל 512GByte ביחידת החומרה הזו.

- ב. הדור הבא של הפטריות היה מהיר יותר וזול יותר, אבל דרושה פעולה נוספת שנקראת אתחול. על פטריה שאותחלה אפשר לכתוב סיבית, ואפשר לקרוא אותה פעם יחידה בעתיד. על פטריה שכבר קראנו ממנה את הסיבית, אי אפשר לכתוב שוב; צריך לאתחל אותה, ורק אז אפשר לכתוב. אפשר לאתחל פטריה מספר בלתי מוגבל של פעמים. ניתן לאתחל רק ביחידות של 1GBytes של פטריות. אי אפשר לאתחל פטריה אחת (או קבוצה קטנה). הצע מנגנון שיאפשר ליחידת חומרה דומה (עם 6GB של RAM ומעבד) להחליף 512GBytes של RAM.

- ג. האם היחידה החדשה תומכת בהחלפה של 1024GBytes או קרוב לכך?

שאלה 3 (21 נקודות)

- א. תהליך מרובה חוטים שיש לו ערוץ TCP פתוח למחשב אחר נתקע ב-deadlock וכל החוטים שלו תקועים. איך יחווה את הבעיה התהליך בצד השני בערוץ ה-TCP?

- ב. תהליך שיש לו ערוץ TCP פתוח למחשב אחר עף בגלל גישה לזיכרון שלא הוקצה. איך יחווה את הבעיה התהליך בצד השני של ערוץ ה-TCP?

ג. תוכנית מרובת חוטים שמשתמשת במשתני תנאי (condition variables) אבל לא במנעולים לא יכולה להיתקע ב-deadlock (כלומר מצב שבו כל החוטים מחכים ולא תהיה התקדמות לעולם). נכון/לא נכון

ד. NFS מתאימה במיוחד ליישומים מקביליים כגון מערכת הזמנת כרטיסי טיסה. נכון/לא נכון

ה. יצירת תהליך חדש למימוש מטרה דומה למטרה של התהליך היוצר יעילה יותר ביוניקס ולינוקס מאשר בחלונות. נכון/לא נכון

ו. חשיבות השימוש בקוד לתיקון שגיאות עולה ככל שמספר הדיסקים במערכת אחסון מידע עולה. נכון/לא נכון

שאלה 4 (31 נקודות)

השלים את הקוד הבא. הקוד ממש שרת שמספק ללקוחות מספרי מזל דרך ערוץ TCP. השרת מקבל בקשת התחברות מלקוח ומייד עונה במחרוזת מהצורה "your lucky number is k\r\n" (במקום k צריך להופיע מספר שהשרת בוחר באופן אקראי). השרת שולח את המחרוזת, כולל בית 0 בסופו ומתנתק. אין צורך לטפל בשגיאות ואפשר להניח שהשרת רץ לעד. יש להשתדל למלא קוד שמתאים לתבנית (מספר ארגומנטים וכו).

```
#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=_____

    nLeft=1+sprintf(buffer,"your lucky number is %d\r\n",rand());
    while(_____) {
        _____ = _____
    }
}
```

```

_____ ; _____
}

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=_____ (PORT_NO);
    addr.sin_addr.s_addr=inet_addr(IP_ADDRESS);

    _____

    while(1){
        _____ = _____
        _____ = _____ (NULL, 0, _____,
        _____, 0, NULL);

    }
    return 0;}

```