

מערכות הפעלה

פתרון מבחן מועד א', 2.02.03

מרצה: ד"ר אנה מוס
מתרגלת: גב' ז'קי עיון

הנחיות:

1. משך הבחינה 3 שעות.
2. בטופס המבחן 9 דפים כולל דף זה. וודא כי כולם נמצאים בידך.
3. יש לכתוב את התשובות בטופס המבחן.
4. יש לענות על כל השאלות.
5. מותר להשתמש בחומר עזר לא אלקטרוני.

בהצלחה!

הישג	ערך	שאלה
	20	1
	35	2
	45	3
	100	סכום

שאלה 1 (20 נקודות)

בשאלה זו נרצה לישנות את שיטת תזמון התהליכים ה-Xinu כדי למנוע הרעבת ותהליכים בעדיפות נמוכה. לשם כך, הפונקציה `resched()` לפני בחירת תהליך חוזש לריצה, תקרא לפונקציה חדשה `update_ready_prio()`. על הפונקציה החדשה הזו להגדיל ב-1 את העדיפות של כל אחד מהתהליכים בתור `ready` למעט תהליך ה-`NULL`.

כתבו את הפונקציה `update_ready_prio()` המבצעת את הנדרש. זיכרו לעדכן את כל מבני המערכת הקשורים.

```
int update_ready_prio()
{
    int pid;

    pid = q[rdyhead].qnext;          /* the first process in the queue */
    if (pid >= NPROC) return OK;    /* empty queue */
    if (pid == 0) pid = q[pid].qnext; /* skip over the NULL process */
    while (pid < NPROC) {
        q[pid].qkey ++;
        proctab[pid].pprio ++;
        pid = q[pid].qnext;
    }
    return OK;
}
```

שאלה 2 (35 נקודות)

בשאלה זו נרצה לממש מנגנון חדש של שימוש בסמפורים ב-Xinu. המנגנון החדש נועד להגנה על קטעים קריטיים של תהליכים לפי הכללים הבאים:

- תהליכים המשתתפים במנגנון החדש מתחלקים לשתי קבוצות יריבות, קבוצה 1 וקבוצה 2.
- כאשר תהליך נכנס לקטע קריטי, כל תהליך נוסף מאותה קבוצה המבקש להיכנס לקטע קריטי יוכל להיכנס ללא המתנה ואילו כל תהליך מקבוצה אחרת יצטרך להמתין עד אשר יצאו מהקטעים הקריטיים שלהם כל התהליכים מהקבוצה היריבה.
- ברגע שתהליך אחרון מקבוצתו עוזב את הקטע הקריטי ויש תהליכים מהקבוצה היריבה המחכים בכניסה, יוכלו להיכנס כל הממתנים מהקבוצה היריבה בבת אחד.

לצורך מימוש המנגנון החדש נוסף למבנה סמפור struct sentry שני שדות נוספים, inside_count, המסמן כמה תהליכים (מאותה קבוצה) נמצאים כעת בקטע קריטי, ו-group היכול לקבל ערכים 1 או 2 בלבד ומסמן איזו קבוצה נמצאת כעת בקטע קריטי (אם inside_count הוא 0, אז לערך של group אין משמעות). לשדה sement לא יהיה שימוש במנגנון החדש.

המנגנון החדש ימומש ע"י שלוש קריאות מערכת הפעלה הבאות:

- group_mutex_create() - יצירת סמפור חדש עבור המנגנון המתואר לעיל והחזרת המזהה שלו (בדומה ל-screate)
- group_mutex_wait(int sem, int group_no) - המתנה לסמפור sem ע"י תהליך מקבוצה group_no (1 או 2)
- group_mutex_signal(int sem) - עזיבת קטע קריטי ע"י תהליך

להלן מימוש קריאת מערכת הפעלה group_mutex_wait():

```
SYSCALL group_mutex_wait(int sem, int group_no)
{
    int ps;
    register struct sentry *sptr;
    register struct pentry *pptr;

    disable(ps);
    if (isbadsem(sem) || (sptr = &semaph[sem])->sstate == SFREE) {
        restore(ps);
        return(SYSERR);
    }
    if (sptr->inside_count == 0 || sptr->group == group_no) {
        (sptr->inside_count) ++;
        sptr->group = group_no;
        restore(ps);
        return(OK);
    }
    (pptr = &proctab[currpid])->pstate = PRWAIT;
    pptr->psem = sem;
    enqueue(currpid, sptr->stail);
    sched();
}
```

א. (15 נקודות)

ממשו קריאת מערכת הפעלה (`group_mutex_create()`)
הדרגה: מומלץ להתבונן בקוד קריאת המערכת `screate()`.

```
SYSCALL group_mutex_create()  
{  
    int ps, sem;  
  
    disable(ps);  
    if((sem = newsem()) == SYSERR) {  
        restore(ps);  
        return(SYSERR);  
    }  
    semaph[sem].inside_count = 0;  
    restore(ps);  
    return(sem);  
}
```

ב. (20 נקודות)

ממשו קריאת מערכת הפעלה (`group_mutex_signal()`)
הנרמית: מומלץ להתבונן בקוד קריאת המערכת (`sreset()`)

```
SYSCALL group_mutex_signal(int sem )
{
    int ps, pid, slist;
    struct sentry *sptr;

    disable(ps);
    if (isbadsem(sem) || semaph[sem].sstate == SFREE) {
        restore(ps);
        return(SYSERR);
    }
    if (--semaph[sem].inside_count == 0) { /* the last process left */
        if (semaph[sem].group == 1) semaph[sem].group = 2; /* change group */
        else semaph[sem].group = 1;
        sptr = &semaph[sem];
        slist = sptr->qlhead;
        while((pid = getfirst(slist)) != EMPTY) { /* release waiting processes */
            ready(pid);
            semaph[sem].inside_count++;
        }
        resched();
    }
    restore(ps);
    return OK;
}
```

שאלה 3 (45 נקודות)

ושאלה זו 9 סעיפים, ניקוד כל סעיף 5 נקודות.

בכל אחד מהסעיפים הבאים יש להקיף בעיגול את התשובה הנכונה ביותר. אין לסמן יותר מתשובה אחת. סימון של יותר מתשובה אחת או סימון לא ברור יביא לניקוד 0 לסעיף!

1. נתונה מערכת המנהלת את הזיכרון הראשי בשיטת הזיכרון הווירטואלי הממומשת ע"י Demand Paging. הזיכרון הראשי מחולק ל-4 מסגרות אשר במצב ההתחלתי פנויות. תהליך מייצר סידרת פניות הבאה לדפים (הסדר הוא משמאל לימין)
0, 2, 7, 0, 1, 3, 1, 2, 0, 2, 4, 1
מהו מספר החטאות דף עבור הסידרה הנ"ל לפי האלגוריתמים FIFO ו-LRU ?

- א. 8 עבור FIFO ו-6 עבור LRU
- ב. 8 עבור FIFO ו-7 עבור LRU
- ג. 9 עבור FIFO ו-7 עבור LRU
- ד. 9 עבור FIFO ו-6 עבור LRU

2. מה יקרה אם בקריאת מערכת הפעלה () של Xinu נחליף שורה
pptr->phasmg++; בשורה ? pptr->phasmg = 1;

- א. השינוי לא ישפיע על פונקציות המערכת אך עלול לפגוע בתהליכי משתמש
- ב. ההודעה החדשה תדרוס את ההודעה הקודמת במידה והיתה
- ג. השינוי ישבש את הפעולה הנכונה של קריאת מערכת הפעלה () receive()
- ד. אף תשובה אינה נכונה

3. נתון תהליך משתמש הבא ב-Xinu:

```
void proc ( )  
{  
    int sem;  
    sem = screate(1);  
    wait(sem);  
    signal(sem);  
    printf("%d", sem);  
}
```

בהנחה שזהו תהליך משתמש יחיד בעת תחילת ריצתו, מה יהיה הפלט של התהליך ?

- א. 0
- ב. 1
- ג. לא יודפס כלום
- ד. אף תשובה אינה נכונה

4. לבקר הדיסק מגיעות פניות לצילינדרים בסדר הבא (משמאל לימין)
10, 3, 20, 22, 6, 40, 5

ההתחלה זרועת הדיסק נמצאת מעל הצילינדר 18.
מהו המרחק הכולל (בצילינדרים) של תזוונות הזרועה עד לסיום שרות הבקשות הנ"ל לפי האלגוריתם SSTF?

$29 - 18 = 4$
 $29 - 6 = 16$
 $10 - 5 = 5$
 $40 - 5 = 35$
60

א. 53
 ב. 52
 ג. 59
 ד. 60

5. מה יקרה אם בפונקציה $ctxsw()$ נוסף פקודה cli מייד אחרי הפקודה $mov sp, [bx]$?

- א. השינוי לא ישפיע על פעולת המערכת
 ב. התהליך החדש ירוץ עם דגל הפסיקות כבוי
 ג. המערכת עלולה לכרום
 ד. אף תשובה אינה נכונה

6. במערכת המנהלת את הזיכרון הראשי בשיטת הדפדוף, תהליך מסויים מייצר כתובות לוגיות בין 0 ל-255. ידוע שבמערכת זו גודל דף הוא 32 בתים והזיכרון הראשי מחולק ל-16 מסגרות. להלן טבלת הדפים של התהליך:

0	3
1	7
2	0
3	10
4	5
5	12
6	4
7	8

מהי הכתובת הפיזית המתאימה לכתובת הלוגית 113 (01110001 בבינארי)?

$113 = 64 + 49$
 $113 = 64 + 32 + 17$
 $113 = 64 + 32 + 16 + 1$

- א. 49 (000110001 בבינארי)
 ב. 337 (101010001 בבינארי)
 ג. 129 (010000001 בבינארי)
 ד. 241 (011110001 בבינארי)

7. נמונה מערכת המנהלת את הזיכרון. אחת מתהליכי ה-Swapping. ברגע מסוים מצג הזיכרון הפנוי הינו כפי שמצוייר להלן:

0	תפוס
1K	פנוי
5K	תפוס
6K	פנוי
8K	תפוס
10K	פנוי
15K	תפוס

כעת מגיעות בקשות להקצאת זיכרון בסדר הבא: הבקשה הראשונה ל-2K, הבקשה השניה ל-3K והבקשה השלישית ל-4K. בהנחה ש-First Fit תמיד מתחיל חיפוש מהחור הפנוי בכתובת הנמוכה ביותר, עבור איזה אלגוריתמים להקצאת זיכרון תצליח המערכת להענות בהצלחה לשלוש הבקשות?

- א. רק First Fit
- ב. רק Best Fit
- ג. רק Worst Fit
- ד. First Fit וגם Best Fit
- ה. First Fit וגם Worst Fit
- ו. Best Fit וגם Worst Fit

8. מה יקרה אם בקריאת מערכת הפעלה `create()` של Xinu נחליף שורה `(int) INITRET; *(-sp) =` בשורה `sp; --`?

- א. המערכת תעבוד אך בסיום שיגרתם הראשית תהליכים לא ייהרגו
- ב. המערכת עלולה לכרוס לפני שלראשונה תהליך כלשהו מסתיים
- ג. המערכת עלולה לכרוס כאשר לראשונה תהליך כלשהו מסתיים
- ד. תשובות ב' ו-ג' נכונות
- ה. אף תשובה אינה נכונה

9. נתונה תכנית משתמש הבאה ב-Xinu:

```
#include<conf.h>
#include<kernel.h>

void xmain( )
{
    int pid1,pid2;
    int func1(), func2();
    pid1 = create(func1, INITSTK, INITPRIO+1, "proc1", 0);
    pid2 = create(func2, INITSTK, INITPRIO+2, "proc2", 1, pid1);
    resume(pid1);
    resume(pid2);
}

void func1( )
{
    int msg;
    printf("A");
    msg = receive( );
    printf("%c", msg);
}

void func2( int pid )
{
    printf("B");
    sendf(pid, 'C');
    sendf(pid, 'D');
}
```

מה יהיה הפלט של התכנית ?

A. BAC

ב. BAD

ABC

ABD