אוניברסיטת בר-אילן
המחלקה למתמטיקה ולמדעי המחשב

מבחן במערכות הפעלה

סמסטר ב', מועד ב', תש"ס
י"ג באלול תש"ס, 13-09-2000

הוראות

1. משך הבחינה - שעתיים.
2. יש לענות על כל השאלות.
3. לכל השאלות משקל שווה.
4. מותר להשתמש במחשבון (Calculator) או במילון.
5. המבחן בחומר סגור. אסור להשתמש בכל חומר עזר.
6. יש לכתוב בעט שחור או כחול (אין לכתוב בעפרון או בעט אדום, ירוק וכדומה).
7. בכל שאלה סמנו רק תשובה אחת, המדויקת ביותר.
8. התשובה "כל התשובות המפורטות צריכות להיות מסומנות" אינה כוללת את עצמה, את התשובה "בשאלה אין נתונים מספיקים כדי לבחור בתשובה מפורטת", ואת התשובה "אף תשובה מפורטת אינה צריכה להיות מסומנת".
9. התשובה "אף תשובה מפורטת אינה צריכה להיות מסומנת" אינה כוללת (כמובן) את עצמה.

---

1) מה ההבדל העיקרי בין מערכת מקבילית רב-מעבדים (Multiprocessor parallel system, tightly-coupled) לבין מערכת מבוזרת (Distributed system, loosely coupled)?

א) לראשון יש התקני I/O משותפים, ולשני יש התקני I/O נפרדים.
ב) לראשון יש שעון משותף ובדרך כלל זיכרון משותף, ולשני זיכרון ושעון נפרד לכל יחידה.
ג) בראשון ניתן לבצע ישומים של זמן אמת, בשני לא ניתן לבצע ישומים כאלה.
ד) אף תשובה מפורטת אינה צריכה להיות מסומנת.
ה) כל התשובות המפורטות צריכות להיות מסומנות.

2) מנגנון ה-Spooling אוגר בקשות בשטח מסוים. שטח זה נמצא:

א) בזיכרון הפנימי.
ב) בדיסק.
ג) ב-Cache.
ד) ב-CPU registers.
ה) אף תשובה מפורטת איננה נכונה.

3) מי מהפעולות הבאות מחייבת מעבר למצב משגוח (monitor mode \ kernel mode \ supervise mode)?

א) פניה לזיכרון הפנימי.
ב) פניה לדיסק.
ג) שינוי ה-program counter.
ד) פעולת חיבור בין שני אוגרים וכתיבת התוצאה לאוגר שלישי.
ה) אף תשובה מפורטת איננה נכונה.

4) מתוך הסתכלות על הקוד של תהליך (התוכנית) בלבד ניתן לזהות נקודות בהן יעבור התהליך

א) ממצב רץ (running) למצב מוכן (ready)
ב) ממצב מוכן למצב רץ
ג) ממצב חסום (blocked/waiting) למצב user (user mode)
ד) ממצב חסום (blocked) למצב משגוח (kernel mode)
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת

5) לגבי תהליך המשתמש במספר סמפורים אפשר (אולי) למנות:

א) תהליך זה לעולם לא יהיה חלק מקיפאון (deadlock).
ב) אסור להשתמש ביותר מסמפור אחד בכל תהליך.
ג) הסדר שבו התהליך יבצע signal(Up) לסמפורים השונים הוא בהכרח הסדר ההפוך לסדר שבו בוצע ה-Wait(Down).
ד) ככל שנשתמש ביותר סמפורים נקטין את מספר החטיפות. (preemption).
ה) אף תשובה מפורטת איננה נכונה.

6) שימוש בפקודת ה- TSL (Test and Set) יכול להבטיח מניעה הדדית אך אינו מתגבר על הקשיים הבאים:

א) צורך בתמיכה של חומרה מיוחדת
ב) המתנה פעילה (busy wait)
ג) המתנה חסומה (bounded wait)
ד) כל התשובות המפורטות צריכות להיות מסומנות
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת

7) לפניכם הצעה לפרוטוקול לפתרון בעיית המרוץ הקריטי עבור שני תהליכים Pj ו- Pi כאשר i ו- j יכולים לקבל את הערכים 0 או 1.

(המשתנים flag ו- turn הם משתנים משותפים לשני התהליכים)

```
int flag[2];
int turn;

While(TRUE)
{
        flag[i]=TRUE;
        while flag[j] {
                if (turn==j)
                 {
                        flag[i] =FALSE
                        while (turn==j);  /*  wait  */
                        flag [i] = TRUE;
                }
        ...
        Critical_Section
        ...
        turn =j;
        flag [i]  = FALSE;
        ...
        Non_Critical_Section
        ...
};                          /* Loop Forever */
```

האם הפרוטוקול פותר את בעיית המרוץ הקריטי?

א) כן

ב) לא, כי אין מניעה הדדית (mutual exclusion)

ג) לא. כי אין התקדמות (progress)

ד) לא. כי אין המתנה חסומה (bounded wait)

ה) לא. כי יש המתנה פעילה (busy wait)

8) בזיכרון וירטואלי, נתונה מחרוזת התייחסות לדפים (Page Reference String או trace) הבאה, משמאל לימין:

2,3,1,0,2,4,3,2,0,4,2

בהנחה שגודל הזיכרון המוקצה לתהליך הוא 3 מסגרות, מהו מספר תקלות הדף (Page Fault) הכולל (כולל טעינה למסגרות הריקות מלכתחילה) לכל אחד מאלגוריתמי ההחלפה הבאים:

א) FIFO 6, OPT 6, LRU 8
ב) FIFO 9, OPT 8, LRU 9
ג) FIFO 9, OPT 6, LRU 8
ד) FIFO 8, OPT 9, LRU 8
ה) FIFO 10, OPT 9, LRU 8
ו) FIFO 9, OPT 9, LRU 11

9) בזיכרון בגודל 400K במפת זיכרון הנתונה:

OS	1	X	2	X	X	3	X	4	X	5

30K 60K 80K 140 175 215 280 300 375 400

(הערה: החורים מסומנים ע"י מספר סידורי ושטחים תפוסים ע"י X)

באיזה חור ימוקם קטע בגודל 25K לפי שיטת BEST FIT?

א) 1.
ב) 2.
ג) 3.
ד) 4.
ה) 5.

10) כאשר תהליך גורם לתקלת דף (page fault):

א) ה-kernel נקרא כדי שיביא את הדף שגרם לתקלה מן הדיסק.
ב) התהליך שגרם לתקלה מביא בעצמו את הדף החסר מן הדיסק.
ג) התהליך שגרם לתקלה מופסק ועל המסך מודפסת הודעה מתאימה.
ד) מפסיקים להשתמש בדפדוף (Paging) ועוברים להשתמש בקיטוע (Segmentation).
ה) אף תשובה מפורטת איננה נכונה.

11) מיישמים סיבית לכלוך (Dirty Bit) בדפדוף כדי ש:

א) יהיו פחות כתיבות לדיסק.
ב) כדי שיהיה ניתן ליישם את אלגוריתם FIFO.
ג) ניתן יהיה להשתמש בקיטוע (Segmentation) ולא רק בדפדוף (Paging).
ד) ניתן יהיה לשתף דפים (Page Sharing).
ה) אף תשובה מפורטת איננה נכונה.

12) נתונים Ready queue כדלהלן:

process	1	2	3	4	5
arrival	0	2	4	6	8
bust time	3	4	6	5	2

מהו זמן הסבב (המתנה בתורך Ready + זמן ביצוע) הממוצע, וזמן המתנה ממוצע לפי שיטת Shortest Job First - SJF

	המתנה	סבב
א)	3.60	7.60
ב)	3.20	8.00
ג)	4.00	7.60
ד)	3.60	8.00
ה)	אף תשובה מפורטת אינה צריכה להיות מסומנת.	

13) שיטת Round Robin עם פלח זמן (Time slice, quantum) יחסית קצר נחוצה עבור מערכות שעובדות ב-Time Sharing, המחיר לשיטה זו:

א) גידול בזמן הסבב הממוצע.
ב) מספר גדול של מתוגי הקשר (Context Switches).
ג) גידול בזמן ההמתנה הממוצע.
ד) כל התשובות המפורטות צריכות להיות מסומנות
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

14) בשיטת תזמון לפי עדיפויות (קבועות) (fix) priorities)) עם חטיפות (with preemption), תהליך T בעל עדיפות p יעבור ממצב רץ (running) למצב מוכן (ready) כאשר:

א) תהליך בעל עדיפות גבוהה יותר מ- p עובר ממצב חסום (blocked) למצב מוכן (ready)
ב) נסתיימה פעולת "sleep" של תהליך אחר בעל עדיפות p
ג) מסתיים פלח הזמן (time slice/quantum) של התהליכים מעדיפות p
ד) כאשר T מתחיל פעולת I/O
ה) כל התשובות המפורטות צריכות להיות מסומנות
ו) אף תשובה מפורטת אינה צריכה להיות מסומנת

15) תזמון זרוע דיסק ע"י SSTF (נקרא גם Shortest Seek First (SSF)) עדיף על FCFS (First Come First Served, נקרא גם FIFO) במובן הבא:

א) הוא נותן שרות אמין יותר
ב) הוא נותן שרות הוגן (fair) יותר
ג) הוא נותן ,במקרים רבים, תנועת זרוע (כוללת) קצרה יותר
ד) הוא מונע הרעבה (starvation)
ה) הוא מתאים יותר ליישומי מולטי-מדיה (multi-media streams)

16) בריצה נורמלית תחת מערכת הפעלה בה גודל ה-pipe הוא 4K, תכנית1 בנספח התוכניות:

א) תדפיס 1000 למסך.
ב) תדפיס 4096 למסך.
ג) תעוף בגלל broken pipe.
ד) לא תסיים (תתקע).
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

17) בריצה נורמלית, תכנית2 בנספח התוכניות:
א) תרוץ לנצח.
ב) תדפיס "hello world" למסך.
ג) תותיר תהליך בן במצב יתום.
ד) תתפוס לפחות סיגנל אחד מסוג SIGPIPE.
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

18) בהנחה שה fork מצליח, תוכנית3 בנספח התוכניות:
א) תדפיס "PARENT" ו "CHILD" למסך בסדר כלשהו ולאחר מכן תסיים.
ב) תדפיס "PARENT" ו "CHILD" למסך בסדר כלשהו ולאחר מכן תרוץ לנצח.
ג) תרוץ לנצח מבלי להדפיס דבר.
ד) תדפיס "PARENT" ואח"כ "CHILD" למסך ולאחר מכן תסיים.
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

19) אילו מהבאים הוא פלט אפשרי של תכנית4 בנספח התוכניות:
א) 122
ב) 212
ג) 112
ד) כל התשובות המפורטות צריכות להיות מסומנות.
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

20) (בהנחה כי פקודות ה fork מצליחות) תכנית5 בנספח התוכניות:
א) תותיר תהליך אחד (בלבד) במצב יתום.
ב) תותיר שני תהליכים במצב יתום.
ג) תותיר תהליך אחד (בלבד) במצב zombie.
ד) תותיר שני תהליכים במצב zombie.
ה) אף תשובה מפורטת אינה צריכה להיות מסומנת.

נספח תוכניות

תוכנית 1

```
#define MYNUM 1000

int main()
{
      int p[2];
      char c = 'x';
      int i;

      if(pipe(p) < 0)
            exit(1);

      for(i = 0; i < MYNUM; i++)
            write(p[1], &c, 1);
      close(p[1]);

      i = 0;
      while(read(p[0], &c, 1) > 0)
            i++;
      printf("%d\n", i);
}
```

תוכנית 2

```
int main()
{
      int p[2];
      char c;
      if(pipe(p) < 0)
            exit(1);
      switch(fork())
      {
      case -1:
            exit(1);
      case 0:  /* child */
            while(read(p[0], &c, 1) > 0)
                  putchar(c);
            break;
      default: /* parent */
            close(p[0]);
            write(p[1], "hello world\n", 12);
            close(p[1]);
            exit(0);
      }
      exit(0);
}
```

תוכנית 3

```
#include <signal.h>

void trap(int);
int flag = 1;

int main()
{
	int pid, ppid;

	signal(SIGUSR1, &trap);

	switch(pid = fork())
	{
	case -1:
		exit(1);
	case 0:
		/* child process */
		ppid = getppid();
		kill(ppid, SIGUSR1);
		while(flag)
			;
		printf("CHILD\n");
		break;
	default:
		/* parent process */
		kill(pid, SIGUSR1);
		while(flag)
			;
		printf("PARENT\n");
	}
	exit(0);
}

void trap(int sig)
{
	flag--;
}
```

תוכנית 4

```
int main()
{
	int s;
	int i = 1;

	while(i < 3)
	{
		if(fork() > 0)
		{
			wait(&s);
			printf("%d", i);
		}
		i++;
	}
}
```

```
int
main()
{
      int s;
      if(fork() == 0)
      {
        exit(0);
      }
      if(fork() == 0)
      {
        exit(0);
      }

      wait(&s);
      while(1)
         ;
}
```

נספח תוספות

תוספת א1

```
int flag[2];
int turn;

While(TRUE)
{
        flag[i]=TRUE;
        while flag[j] {
                if (turn==j)
                 {
                        flag[i] =FALSE
                        while (turn==j);  /*  wait  */
                        flag [i] = TRUE;
                }
        ...
        Critical_Section
        ...
        turn =j;
        flag [i]  = FALSE;
        ...
        Non_Critical_Section
        ...
};                          /* Loop Forever */
```

תוספת א2

OS	1	X	2	X	X	3	X	4	X	5
30K	60K	80K	140		175	215	280	300	375	400

תוספת א3

process	1	2	3	4	5
arrival	0	2	4	6	8
bust time	3	4	6	5	2

Moed B
---

1 - 2
2 - 2
3 - 2
4 - 5
5 - 5
6 - 2
7 - 5
8 - 2
9 - 5
10 - 1
11 - 1
12 - 5
13 - 2
14 - 1
15 - 3
16 - 1
17 - 2
18 - 1
19 - 5
20 - 3