<u>אוניברסיטת בר-אילן, המחלקה למדעי המחשב</u>

<u>מבחן במערכות הפעלה, נוסח א'</u>

מרצה: מר רז לין
מספר קורס: 20-231-89
סמסטר קיץ תשס"ג, מועד א'
תוכנית עלית
זמן הבחינה: שעתיים.
תאריך: אוקטובר 2003

המבחן בחומר סגור. אין להשתמש בספרים או חומר עזר כלשהו.
מותר השימוש במחשבון כיס בלבד.

<u>הנחיות:</u>

1. יש לציין בתחילת מחברת הבחינה את נוסח הבחינה.
2. יש לרשום את כל התשובות, כולל התשובות לשאלות האמריקאיות, במחברות הבחינה בלבד בציון מספר השאלה.
3. יש לרשום בכתב יד ברור בעט כחול או שחור בלבד.
4. בשאלות החישוב יש לרשום את כל הנוסחאות בהם השתמשתם והסבר. תשובה סופית בלבד לא תתקבל.
5. בשאלות האמריקאיות ייתכנו כמה תשובות נכונות או אף תשובה. יש לציין את כל התשובות הנכונות. אין צורך לנמק תשובות אמריקאיות.
6. בשאלות הפתוחות יש לנמק במידת הצורך. בכל מקרה יש לענות בתמציתיות ולעניין.
7. ניתן לקחת את טפסי הבחינה.
8. בהצלחה.

1. מהן פסיקות? אילו סוגי פסיקות קיימות? מה מטרתן?

2. אילו מהפעולות הבאות הן פעולות פריבילגיות?
   א. מעבר למצב monitor.
   ב. טעינת אוגר בסיס.
   ג. איפוס ה-Timer.
   ד. בדיקת ערך ה-Timer.
   ה. אף תשובה אינה נכונה.

3. בין ההבדלים בין תהליך לתוכנית ניתן למנות:
   א. תהליך הוא תוכנית בזמן ביצוע.
   ב. תהליך מקבל משאבים, בעוד לתוכנית אין משאבים.
   ג. תהליך אינו יכול לגשת ישירות למערכת ההפעלה, ואילו בתוכנית ניתן.
   ד. רמת הבטיחות של תוכנית נמוכה מזו של תהליך.
   ה. אף תשובה אינה נכונה.

4. מה ההבדל בין תהליך במצב ריצה (running) לתהליך במצב המתנה (waiting)?

5. א. מהו מיתוג הקשר (context switch)?
   ב. מה מטרת המתזמן לזמן ארוך? מה מטרת המתזמן לזמן קצר? מנו לפחות הבדל משמעותי אחד בין שני המתזמנים.

6. אילו מבין הנתונים הבאים מוכלים בתוך ה-PCB?

א. Program counter.
ב. מידע על פעולות ק/פ של התהליך.
ג. נתונים על אוגרים בשימוש התהליך.
ד. מצב התהליך.
ה. אף תשובה אינה נכונה.

7. מוצע לממש את תור התהליכים המוכנים לריצה כתור LIFO (Last In First Out). האם פתרון זה הוא פתרון טוב? במידה שכן, יש למנות לפחות יתרון אחד על פני השיטה שנלמדה בכיתה. במידה שלא, יש למנות לפחות חסרון אחד על פני השיטה שנלמדה בכיתה.

8. נתונים 4 תהליכים P_1, P_2, P_3, P_4. להלן פרצי העיבוד וזמני הגעה עבור כל תהליך:

תהליך	פרץ עיבוד	זמן הגעה
P_1	5	0.0
P_2	7	0.1
P_3	3	0.1
P_4	4	0.1

הראו טבלת גאנט עבור התזמון של התהליכים לפי SJF עם לקיחה בכוח וחשבו זמן ביצוע (turnaround time) ממוצע.

9. אילו מהפתרונות הבאים אינו פותר מצב של הרעבה של תהליכים:
א. הצמדת עדיפות לתהליך ועדכון העדיפות.
ב. מעבר תהליכים בין תורי התזמון השונים.
ג. Aging.
ד. קביעת פלח זמן (quantum) לביצוע כל תהליך.
ה. אף תשובה אינה נכונה.

10. מוצע הפתרון הבא לבעיית הקטע הקריטי בין 2 תהליכים – P_i ו-P_j. להלן הפתרון המוצע (עבור תהליך P_i):

<u>Process P_i:</u>

<u>Shared variables:</u>

```
var flag[NUM_OF_PROCESSES]: boolean; /* initialized to FALSE */
var num[NUM_OF_PROCESSES]: float[0..1]; /* initialized to 0 */
```

<u>Code:</u>

```
num[i] = random(0,1);
while (num[i] == num[j]) rest;
flag = TRUE;

        Critical Section

flag = FALSE;
num[i] = 0;
```

האם הפתרון לעיל עונה על התנאים לפתרון בעיית הקטע הקריטי (יש לציין עבור כל תנאי האם הוא עונה או לא. במידה שכן - יש להסביר כיצד. במידה שלא – יש לציין דוגמה נגדית).

11. עליכם לתכנן מערכת ליירוט טילים. המערכת מורכבת מכמה תתי-מערכות, כל תת-מערכת הינה תהליכון בפני עצמו (thread):

א. תת-מערכת חישובים.
ב. תת-מערכת בקרה.
ג. תת-מערכת יירוט.

לכל תת-מערכת יש את הקוד שלה (ראו בהמשך).
ידוע שהפונקציה **CalculateSafety** בתת-מערכת חישובים **חייבת** להתבצע (i) לפני שתת-מערכת יירוט מבצעת את הפונקציה **Intercept** ו-(ii) אחרי שתת-מערכת בקרה מבצעת את הפונקציה **GetBorders**.

I. באיזה מנגנון שנלמד בכיתה ניתן להשתמש על-מנת לפתרון בעיה זו? מדוע?
II. השלימו את הפסאדו-קוד בפתרון שהצעתם. יש לציין גם את סוג המשתנים, אתחולם וערכים התחלתיים.

<u>Calculation Subsystem:</u>

```
…
CalculateSafety();
…
```

<u>Control Subsystem:</u>

```
…
GetBorders();
…
```

<u>Interception Subsystem:</u>

```
…
Intercept();
…
```

12. מה ההבדל בין פיצול חיצוני ופיצול פנימי (Internal vs. External fragmentation)? ניתן להסביר על-ידי דוגמה.

13. נתונים רצף הדפים הבא שיש למפות בזיכרון. הזיכרון כולל 3 מסגרות. סדר הגעת הדפים משמאל לימין. השתמשו באלגוריתם LRU למפות את הדפים ולציין כמה תקלות דף (page fault) נגרמו. יש להדגים את כל התהליך ולא לציין תשובה סופית בלבד:
1, 2, 4, 5, 3, 5, 5, 4, 2, 1

14. נתון רצף בקשות הדיסק הבאות (משמאל לימין). בדיסק יש 200 גלילים (0 עד 199). השתמשו בשיטת C-LOOK על-מנת למצוא את מרחק תזוזת הראש. הראש נמצא בתחילה בגליל 23 ונע כלפי מעלה (גליל 199).
24, 2, 124, 56, 99, 153, 34

15. איזו מהטענות הבאות נכונה בנוגע לתוכנית 1 שבנספח התוכניות?

א. הפקודה kill (שורה 24) עשויה להיכשל (להחזיר 1-), שכן יכול להיות שתהליך הבן כבר לא יהיה קיים.

ב. ייתכן מצב בו תהליך הבן לא הספיק לדרוס את פונקצית הטיפול ב-signal מסוג SIGALRM, ולכן יבצע את פונקצית הטיפול הרגילה (ברירת-המחדל) ולא את handler.

ג. ישנה שגיאה בפונקצית הטיפול ב-signal (handler), בכך שאין דריסה מחדש של ה-signal.

ד. תהליך הבן יבצע את פונקצית הטיפול הרגילה בעת קבלת signal מסוג SIGALRM, כיוון שלא ניתן לדרוס את ה-signal של SIGALRM.

ה. אף תשובה אינה נכונה.

16. בהנחה שתוכנית 2 שבנספח התוכניות רצה ללא שגיאות, מה יהיה הפלט שלה?

א. hello

ב. יודפס ...hellohellohello וכן הלאה (אינסוף hello ברצף).

ג. לא יודפס כלום כיוון שתהליך הבן ייתקע לעד על פקודת ה-read (שורה 21).

ד. תהליך הבן יעוף עקב broken pipe לאחר מות תהליך האב.

ה. אף תשובה אינה נכונה.

17. מה לא ניתן להגיד בנוגע לקוד השרת-לקוח שבתוכניות 3 ו-4 שבנספח התוכניות?

א. קוד השרת-לקוח חשוף לבעיית בטיחות (security hazard).

ב. השרת מבצע פקודות המתקבלות מהלקוח.

ג. סוג התקשורת בקוד הוא TCP.

ד. במהלך ביצוע השרת עשויים להיווצר זומבים, אולם השרת אינו מטפל בהם.

ה. הקוד לא יעבור קומפילציה.

ו. אף תשובה אינה נכונה.

18. (ארגון קבצים):

במארז דיסק עם ראשים נעים יש את הנתונים הבאים:

גודל מסילה בבתים	TRK	15,000
מס' סיבובים לדקה	RPM	2,000
מס' מסילות בגליל	ntc	8
מס' גושים במסילה	nbt	10

בהתחשב בנתונים לעיל, מהו זמן גישה ממוצע בדיסק?

נספח נוסחאות:

$B\ (block\ size) = nsb\ ^{*}\ S$

$IBG = nsb\ ^{*}\ ISG$

$TRK\ (track\ size) \approx nst \cdot (S + ISG) \approx nbt \cdot (B + IBG)$

$$r = \frac{1}{2} \cdot \frac{60 \cdot 1000}{RPM}\ (ms)$$

$$a = s + r$$

$$s_l = s_c + \delta \cdot l$$

$$s_1 = s_c + \delta \cdot 1 \Rightarrow s_1 \approx s_c,\quad s_1 \approx \frac{1}{3} \cdot 2r$$

$$s = s_c + \delta \cdot \frac{N}{3}$$

$$b\mathrm{tt} = \frac{\mathrm{B}}{\mathrm{t}} = \frac{\text{size of block}}{\textit{transfer}\ \text{time}}$$

$$T_R = \frac{R + W}{t}$$

$$t = \frac{TRK}{2r}$$

$$T_B = \frac{B + IBG}{t} \approx btt$$

$$T_{TRK} = \frac{TRK}{t} = \frac{nbt \cdot (B + IBG)}{t} = 2r$$

$$T_{cyl} = \frac{ntc \cdot TRK}{t} = \frac{ntc \cdot nrt \cdot (R + W)}{t}$$

$$t'_{cyl} = \frac{ntc \cdot nrt \cdot R}{T_{cyl}} = t \cdot \frac{R}{R + W}$$

$$s' = \frac{2r}{nrt} = \frac{T_{TRK}}{nrt} = \frac{R + W}{t}$$

$$t' = \frac{R}{\frac{R + W}{t} + s'} = \frac{R}{2 \cdot \frac{R + W}{t}} = \frac{1}{2} \cdot t \cdot \frac{R}{R + W} = \frac{1}{2} \cdot t'_{cyl}$$

נספח תוכניות:

תוכנית 1:

```
1       #include <sys/types.h>
2       #include <unistd.h>
3       #include <sys/types.h>
4       #include <sys/wait.h>
5       #include <stdio.h>
6       #include <signal.h>
7
8       void handler(int sig);
9
10      int main()
11      {
12         int pid;
13
14         signal(SIGALRM, handler);
15
16         switch ((pid = fork()) )
17         {
18           case -1:
19                 perror("fork");
20                 exit(1);
21           case 0:
22                  pause();
23           default:
24                  kill(pid, SIGALRM);
25                  printf("1");
26         }
27      }
28
29      void handler(int sig)
30      {
31         printf("caught");
32         exit(2);
33      }
```

תוכנית 2:

```
1	#include <sys/types.h>
2	#include <unistd.h>
3	#include <sys/types.h>
4	#include <sys/wait.h>
5	#include <stdio.h>
6
7	int main()
8	{
9	   int fd[2];
10	   char buf[5];
11
12	   pipe(fd);
13
14	   switch (fork()) {
15	     case -1:
16	             perror("fork");
17	             exit(1);
18	     case 0:
19	             while (1)
20	             {
21	                read(fd[0], buf, 5);
22	                printf("%s", buf);
23	                write(fd[1], buf, 5);
24	             }
25	     default:
26	             write(fd[1], "hello", 5);
27	   }
28	}
```

תוכנית 3:

```
// server.c – server code
   #include <stdio.h>
   #include <stdlib.h>
   #include <unistd.h>
   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netinet/in.h>
   #include <arpa/inet.h>

   #define MYPORT 3490    // the port users will be connecting to
   #define MAXDATASIZE 100 // max number of bytes we can get at once
   #define BACKLOG 10     // how many pending connections queue will hold

   int main(void)
   {
      int sockfd, new_fd, numbytes, sin_size;
      struct sockaddr_in my_addr, their_addr;
      char buf[MAXDATASIZE];

      if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
         perror("socket");
         exit(1);
      }

      my_addr.sin_family = AF_INET;
      my_addr.sin_port = htons(MYPORT);
      my_addr.sin_addr.s_addr = INADDR_ANY;
      memset(&(my_addr.sin_zero), '\0', 8);

      if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1) {
         perror("bind");
         exit(1);
      }

      if (listen(sockfd, BACKLOG) == -1) {
         perror("listen");
         exit(1);
      }

      while(1) {
         sin_size = sizeof(struct sockaddr_in);
         if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == -1) {
            perror("accept");
            continue;
         }
         if (!fork()) {
            close(sockfd);
            if ((numbytes = recv(new_fd, buf, MAXDATASIZE-1, 0)) == -1) {
                              perror("recv");
                              exit(2);
             }
             buf[numbytes] = '\0';
             execlp(buf, buf, NULL);
            close(new_fd);
            exit(0);
         }
         close(new_fd);
      }
      return 0;
   }
```

תוכנית 4:

```
// client.c - client code
   #include <stdio.h>
   #include <stdlib.h>
   #include <unistd.h>
   #include <errno.h>
   #include <string.h>
   #include <netdb.h>
   #include <sys/types.h>
   #include <netinet/in.h>
   #include <sys/socket.h>

   #define PORT 3490
   #define MAXDATASIZE 100

   int main(int argc, char *argv[])
   {
      int sockfd, numbytes;
      char buf[MAXDATASIZE];
      struct hostent *he;
      struct sockaddr_in their_addr;

      if (argc != 2) {
         fprintf(stderr,"usage: client hostname\n");
         exit(1);
      }

      if ((he=gethostbyname(argv[1])) == NULL) {
         perror("gethostbyname");
         exit(1);
      }

      if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
         perror("socket");
         exit(1);
      }

      their_addr.sin_family = AF_INET;
      their_addr.sin_port = htons(PORT);
      their_addr.sin_addr = *((struct in_addr *)he->h_addr);
      memset(&(their_addr.sin_zero), '\0', 8);

      if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1) {
         perror("connect");
         exit(1);
      }
           scanf("%s", buf);
      if ((send(sockfd, buf, sizeof(buf), 0)) == -1) {
         perror("send");
         exit(1);
      }

     close(sockfd);

      return 0;
   }
```