בחן במערכות הפעלה (6 במאי 2005)

202-1-3031

מר רועי זיוון ופרופ' אמנון מייזלס

משך הבחן: **שעתיים**
חומר עזר: **אסור**
פתור את כל השאלות: **סה"כ 100 נקודות.**

1. (35) תלמיד התבקש לכתוב תוכנית שמסמלצת Shell ואמורה לרוץ על Shell של UNIX. מטרת ה - Shell היתה להריץ תהליך אחד בלבד ב forground ללא אפשרות הרצה של תהליכים ב background. התלמיד התבקש לטפל\לתמוך בסיגנל שנוצר מהקשת ^C (שכזכור שולח SIGINT לקבי התהליכים שב- fg). כלומר, אם מקישים ^C, אז שהסיגנל ישלח לקבי התהליכים שרצים ב- fg של ה - Shell של התלמיד , ושה - Shell שלו לא יסתיים כתוצאה מכך.
נתון הפתרון של התלמיד:

```
int main()
{
	int fgPid;
	int status;
	char* command = (char*)malloc(1024*sizeof(char));
	char** params = (char**)malloc(7*sizeof(char*));
	signal(SIGINT,SIG_IGN);
	while(true)
	{
		printf("my_shell>>");
		read_command(command, params);
		if((fgPid=fork())==0)
			execvp(command, params);
		else
			wait(&status);
	}
}
```

א. מה יהיה הפלט אם נריץ את התוכנית הבאה על ה - Shell של התלמיד?

```
int main()
{
	int i=0;
	for(i=0;i<10;i++)
	{
		printf("loop num %d\n",i);
		kill(getpid(),SIGINT);
	}
```

}

נמקי.

ב. מה יקרה אם נקיש ^C מיד לאחר הרצת התוכנית הנ"ל , נמקי. (נניח שהסיגנל יתקבל ע"י התוכנית מיד לאחר ההדפסה הראשונה).

ג. מה יהיה הפלט אם נריץ את התוכנית הבאה על ה - Shell של התלמיד?

```
void cntl_c_handler(int dummy)
{
  printf("caught SIGINT\n");
}

int main()
{
        int i=0;
        signal(SIGINT,cntl_c_handler);
        for(i=0;i<1000;i++)
        {
                printf("loop num %d\n",i);
                kill(getpid(),SIGINT);
        }
}
```

נמקי.

(30) **2**. שלושה תהליכים : p1 ,p2 ,p3 מבקשים לרוץ על פי השלשה הבאה : (ריצה,OI,ריצה) משמאל לימין. השלשה של p1 – (1,4,2), של p2- (1,2,2) ושל p3- (2,1,2). המערכת היא Preemptive. המתזמן יכול להחליף את התהליך הרץ לאחר כל פרוסת זמן. התהליכים מתוזמנים על פי אלגוריתם Guaranteed Scheduling. במקרה של שוויון יש עדיפות לתהליך בעל האינדקס הנמוך. ציירי תרשים Gantt של ריצת התהליכים וחשבי זמן Turn Around ממוצע עבור המקרים א-ג :

א. זמני ההגעה זהים. כל תהליך מבצע i/o מול רכיב שונה.

ב. זמני ההגעה זהים. כל התהליכים מבצעים i/o מול אותו רכיב. אלגוריתם תיזמון ל- i/o – FCFS.

ג. התהליכים מגיעים בסדר - p1 בזמן 0, p2 בזמן 1 ו-p3 בזמן 2 - כל תהליך מבצע i/o מול רכיב שונה.

ד. האם אלגוריתם Round Robin היה נותן את אותן תוצאות עבור שלושת המקרים הנ"ל? נמקי !

(35) **3**. נתון הקוד לפתרון בעיית ה- Readers-Writers בעזרת מוניטורים, כפי שניתן בכתה.

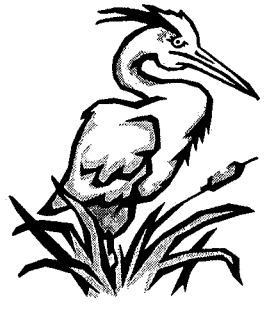
```
Monitor reader_writer{
    int             numberOfReaders = 0;
    boolean  busy = FALSE;
    conditionokToRead, okToWrite;
```

```
public:
    startRead {
       if(busy || (okToRead.queue))  okToRead.wait;
       numberOfReaders = numberOfReaders + 1;
       okToRead.signal
            };
    finishRead {
       numberOfReaders = numberOfReaders - 1;
       if(numberOfReaders == 0) okToWrite.signal
            };

startWrite {
       if((numberOfReaders != 0) || busy) okToWrite.wait;
       busy = TRUE
            };

finishWrite {
    busy = FALSE;
    if(okToWrite.queue)
       okToWrite.signal
    else
       okToRead.signal;
          };
}
```

א. האם יתכן שהתנאי על okToRead.queue (השני בביטוי ה -if בפונקציה startRead) ייבדק וימצא true? הוכיחי.

ב. האם ישנה בעיית הגינות בקוד ? הוכיחי.

ג. הציעי דרך להשגת הגינות על ידי שינוי בשתיים מתוך 4 הפונקציות. בפונקציה אחת שינוי בשורה אחת ובפונקציה אחרת שינוי בשלוש שורות. הוכיחי את נכונות טענתך על הקוד המוצע.



בהצלחה !