קורס: מערכות הפעלה 31261

פתרון: מבחן אמצע, סמסטר ב' תשע"ג, 01/05/2013

<u>הוראות לנבחן:</u> משך הבחינה 90 דקות. אין להשתמש בשום חומרי עזר (כולל מכשירים אלקטרוניים מכל סוג). רשום את תשובותיך בגוף הבחינה במקום המתאים לכל שאלה ושאלה. הקפד על כתב יד ברור ומסודר. השימוש בשפה האנגלית מותר. בכדי שתוכל לערוך ולתקן את תשובותיך מומלץ מאוד להשתמש <u>בעיפרון ומחק</u>. השאלון מכיל 10 שאלות ונפרש על פני 5 עמודים.

[10%] **<u>שאלה1</u>**

תאר בקיצור מהם המאפיינים העיקריים של מערכת הפעלה? (כלומר: מהי מערכת הפעלה?)

במשפט אחד: מערכת הפעלה היא תוכנת מחשב שמטרתה לאפשר למשתמש האנושי לנצל את החומרה והציוד ההקפי באופן נוח ויעיל לשימוש.
מאפיינים עיקריים: פשטות ונוחות של הפעלת יישומים ושימוש מועיל בחומרה ובציוד ההיקפי. מערכת ההפעלה יוצרת סביבה נוחה ופשוטה המאפשרת ניצול אופטימלי של המשאבים העומדים לרשות המשתמש (מעבד, זיכרון, שטחי איחסון, מדפסות, וכדומה) במינימום מאמץ והכרה של מבנה החומרה (המורכבת מאוד) שעומדת לרשותו.

[10%] **<u>שאלה 2</u>**

מנה בקצרה בדיוק ארבעה סוגי שירותים כלליים שעל מערכת הפעלה מודרנית לספק למשתמשים.

1. הוגנות: מנגנון לשם בקרה ופיקוח לשם חלוקה הוגנת של המשאבים בין המשתמשים והתהליכים השונים הרצים במערכת: הקצאת זמן עיבוד שווה לתהליכים השונים בהתאם לדחיפות ולמעמד.
2. בטיחות: הגנה על משאבים מפני נזק אפשרי (בשגגה או בזדון) מצד משתמשים או תהליכי מערכת. הגנה ופרטיות של שטחי איחסון (קבצים, שטחי זיכרון, גישה להתקנים).
3. ניהול תהליכם וניהול זיכרון
4. תיקשורת בין מחשבים

בנוסף לכך התקבלו גם התשובות הבאות:

5. ניהול מערכות קלט ופלט
6. ניהול מערכות קבצים
7. אבטחת מידע

[10%] **<u>שאלה 3</u>**

רשום את שמות שלושת המינשקים הנפוצים ביותר העומדים לרשות המשתמשים השונים להפעלת קריאות שירות (system calls) של מערכת ההפעלה.

1. מינשק פקודות שורה (Command Line Interface – CLI) – חלון DOS, חלון xterm, וכדומה
2. מינשק קובצי אצווה (Batch File Interface) - סקריפטים של Batch, או BASH (ב-Unix)
3. מינשקי משתמש גרפיים (GUI – Graphic User Interfaces)

[10%] **שאלה 4**

תאר בקצרה את המנגנון שבאמצעותו מערכת הפעלה מודרנית (Time-Sharing Operating System) מונעת מתהליך של משתמש (User process) להשתלט על יחידת העיבוד לזמן בלתי מוגבל.

בכל פעם שתהליך חדש נולד או חוזר למצב ריצה, מערכת ההפעלה מפעילה Timer (מונה) למספר מוגבל מאוד של מילי-שניות (בדרך כלל בין 15 ל-60 מילי-שניות). כאשר המונה מגיע ל-אפס הוא מבצע פסיקת חומרה, והשליטה על המעבד חוזרת למערכת ההפעלה. זה מאפשר למערכת ההפעלה לבדוק אם הכל תקין ולא התבצעו שום חריגות בכל מחזור קצר של זמן.
ההתקן השכיח נקרא PIT (Programable Interval Timer) מתוכנן לייצר פסיקות חומרה לאחר כל סיום ספירה לאחור של מונה.

[8%] **שאלה 5**

לפניך מספר שיגרות של ספריות מערכת ההפעלה Linux. סמן ✓ בתיבה המתאימה של כל פונקציה שהיא קריאת מערכת (System call) של מערכת ההפעלה Linux. תשובה נכונה תזכה בנקודה, ושגויה תוריד נקודה.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
```

☐ `FILE *fopen(const char *filename, const char *mode);`
✓ ☐ `int close(int fd) ;`
✓ ☐ `int creat(const char *pathname, mode_t mode) ;`
☐ `int printf(const char *fmt, ...);`
✓ ☐ `ssize_t read(int fd, void* buf, size_t noct) ;`
✓ ☐ `ssize_t write(int fd, const void* buf, size_t noct) ;`
✓ ☐ `int open(const char *pathname, int flags, mode_t mode) ;`
✓ ☐ `off_t lseek(int fd, off_t offset, int ref);`
✓ ☐ `pid_t wait(int* pstatus);`
☐ `char *fgets(char *s, int n, FILE *stream);`
✓ ☐ `pid_t getpid();`
☐ `int abs(int);`

כזכור מקורסי C: הפונקציות fopen(), fclose(), printf() הן פונקציות השייכות לספריות הסטנדרטיות stdio.h ואינן קריאות מערכת אמיתיות. למשל, ב-Unix, הפונקציה fopen() ממומשת באמצעות הקריאת מערכת open(), וב-Windows, היא ממומשת על ידי קריאת המערכת CreateFile.
הפונקציה abs() שייכת לספריה המתימטית math.h ואינה מבצעת או קשורה כלל לקריאות מערכת.

[8%] **שאלה 6**

לפניך רשימה של טענות שונות. לגבי כל טענה, סמן ✓ בתיבה המתאימה אם ורק אם הטענה נכונה. תשובה נכונה מזכה ב-4 נקודות. תשובה שגויה תוריד 2 נקודות. (בכל מקרה הסיכום הכללי לא יהיה שלילי).

- ✓ ☐ במערכת מחשב בעלת מעבד מרובה ליבות ניתן להריץ יותר מתהליך אחד בו-זמנית
  [במעבד בעל ארבעה ליבות, יכולים לרוץ ארבעה תהליכים במקביל בו-זמנית!]
- ☐ מערכת שיתוף זמן (Time-Sharing system) מאפשרת לפעמים לתהליכי משתמש רגיל להשתלט על יחידת העיבוד לפרקי זמן ארוכים העולים על דקה.
- ☐ בכדי לפתוח קובץ בדיסק, תהליך של משתמש רגיל צריך בשלב הראשון לפנות ליחידת ה-DMA.
  [רק מערכת ההפעלה רשאית לפנות ליחידת ה-DMA!]
- ☐ כל תוכנית מחשב שנמצאת בדיסק יכולה להפעיל לכל היותר תהליך (Process) אחד ויחיד.
  [שקר וודאי – תוכנת IExplorer או Firefox מסוגלת להפעיל 10 או 20 תהליכים במקביל]
- ☐ ניתן לכתוב קריאות מערכת (System Calls) גם בשפות סקריפט כמו BASH או Windows Batch
  [קריאות מערכת הן פונקציות C נמוכות הנמצאות בספריות C של המערכת. לכן לא ניתן להשתמש בהן (באופן ישיר) מתוך סקריפטים (קובצי אצווה) של Batch או BASH שהם בשפה עילית מאוד.]
- ☐ בכל פעם בו מתבצעת פסיקה (Interrupt), מערכת ההפעלה מפנה את התהליך שהופסק מהזיכרון הראשי לשטח איחסון זמני על דיסק, ולאחר פסק זמן מעלה אותו שוב מהדיסק לזיכרון
  [זה כמובן בניגוד גמור לכל מה שלמדנו בקורס ובנוסף לכך גם לא נשמע אמין מאחר ופעולת ההורדה לדיסק לאחר כל פסיקה, וטעינה לדיסק לאחר מכן, היא עניין מאוד יקר ולא מתקבל על הדעת! הפסיקה כוללת רק שמירה על הרגיסטרים, מונה התוכנית (Program Counter). התוכנית עצמה נשארת בזיכרון (כולל ה-PCB וטבלת הקבצים הפתוחים), כך שחידוש הריצה של התוכנית ימשיך מיד מהנקודה בה נעצר לפני הפסיקה ללא שהיות רבות. למרות זאת, נציין כי במקרים קיצוניים בהם הזיכרון אזל, מערכת ההפעלה עשויה להוריד חלק מהתהליכים לדיסק. אך זהו המקרה הנדיר ואינו קורה "בכל פעם" כמו שכתוב בטענה למעלה.]
- ✓ ☐ בכל פעם בו מתבצעת גישה לדיסק (קריאה או כתיבת קובץ) סיבית ה-kernel mode במעבד דלוקה.
  [רק מערכת ההפעלה רשאית לגשת לדיסק למטרות כתיבה או קריאה, ולכן סיבית ה-kernel צריכה להיות דלוקה במצבים האלה תמיד]
- ✓ ☐ תהליך של משתמש רגיל אינו רשאי לגשת ישירות למערכת הקבצים (לשם קריאה או כתיבת קובץ)

[12%] **שאלה 7**

להלן תוכנית C להעתקת קבצים (כפי שהוצגה בקורס) במערכת ההפעלה Linux. תוכנית זו הופעלה על קובצי משתמש רגיל. רשום בתיבה הבאה את מספרי השורות הקוד שבהם תתבצע פסיקה (Interrupt) במידה והתוכנית תגיע אליהן (צא מהנחה שכל שורות הקוד יופעלו בתהליך שלנו).

תשובה:

שורות 16-23 כולן!

```
01   #include <stdio.h>
02   #include <fcntl.h>
03   #include <syscall.h>
04   #define PERMS 0666 /* rwx for owner, group, others */
05   #define BUFSIZE 4096
06
07   void error(char *, ...) ;
08
09   /* cp: copy f1 to f2 */
10   main(int argc, char* argv[])
11   {
12      int f1, f2, n;
13      char buf[BUFSIZE];
14
15      if (argc != 3)
16          error("Usage: cp file1 file2");
17      if ((f1 = open(argv[1], O_RDONLY, 0)) == -1)
18          error("cp: can't open %s", argv[1]);
19      if ((f2 = creat(argv[2], PERMS)) == -1)
20          error("cp: can't create %s, mode %03o", argv[2], PERMS);
21      while ((n = read(f1, buf, BUFSIZE)) > 0)
22          if (write(f2, buf, n) != n)
23              error("cp: write error on file %s", argv[2]) ;
24      return 0;
25   }
```

שורה 16: הפונקציה error מפעילה (write לחלון terminal או לקובץ ולכן יוצרת פסיקה

שורה 17: הפונקציה (open היא קריאת שירות יסודית למערכת ההפעלה, וכל קריאת שירות יוצרת פסיקה!

שורה 18: שוב הפונקציה (error יוצרת פסיקה כמו בשורה 16

שורה 19: הפונקציה (create ליצירת קובץ חדש (Unix) היא בוודאי קריאת שירות יסודית שיוצרת פסיקה

שורה 20: ראה שורה 16

שורה 21: הפונקציה (read היא קריאת שירות בסיסית של Unix ולכן יוצרת פסיקה (גישה לקובץ)

שורה 22: כתיבה לקובץ (write יוצרת פסיקה – זוהי שוב קריאת מערכת יסודית ב-Unix.

שורה 23: ראה שורה 16

שאלה 8

הסבר במשפט אחד מה בדיוק מתבצע על ידי תוכנית Python הבאה:

```
import os

dir = "C:/BRAUDE/OS"
s = 0
for file in os.listdir(dir):
    path = dir + os.sep + file
    s += os.path.getsize(path)
print s
```

תשובה:

מתבצע סכום הגדלים של הקבצים בספריה "C:/BRAUDE/OS". הסכום הכולל מודפס למסך.

[8%] **שאלה 9**

הפלט של פקודת ls במערכת Linux נראה כך:

```
linux> ls -l /usr/bin
   -rwxr-xr-x 2 root root       31706 Jan 25 12:34 aclocal
   -rwxr-xr-x 1 root root       18144 Feb  4 20:13 aconnect
   -rwxr-xr-x 1 root root       34676 Jan 27 10:31 base64
   -rwxr-xr-x 1 root root       93160 Jan 26 18:26 bitmap
   -rwxr-xr-x 1 root root      112784 Jan 27 10:32 cp
   -r-xr-xr-x 1 root root         794 Jan 26 15:16 crc32
   -rwxr-xr-x 1 root root       46988 Jan 27 10:32 cut
   -rwxr-xr-x 1 root root       26476 Jan 27 10:32 env
```

הסבר מה בדיוק מתבצע בתוכנית Python הבאה:

```
p = subprocess.Popen("ls –l /usr/bin", shell=True)
for line in p.stdout:
    perms, nlinks, user, group, size, month, day, time, command = line.split()
    if size > 1024*1024:
       print command
p.kill()
```

תשובה:

מדפיס את שמות תוכניות המערכת בספריה /usr/bin אשר גודלן עולה על 1MB.

[18%] **שאלה 10**

הפלט של הפקודה ps של Linux (בתוספת האופציות aux) נראה כך:

```
Linux> ps aux
        USER    PID %CPU %MEM    VSZ   RSS TTY    STAT START   TIME COMMAND
        root      1  0.0  0.4   5648  2540 ?      Ss   Apr20   0:01 /sbin/init showopts
        avahi   360  0.0  0.2   3248  1072 ?      Ss   Apr20   0:00 avahi-daemon
        root    363  0.0  0.1   5044   540 ?      Ss   Apr20   0:13 /usr/sbin/haveged -w 1024 -v 0 -F
        root   2617  0.0  0.1   3884   564 ?      Ss   Apr20   0:00 /usr/bin/kdm
        root   2700  0.0  0.2   5084  1180 ?      Ss   Apr20   0:03 /usr/lib/postfix/master
        dina   3513  0.0  0.6   9196  3428 ?      S    Apr20   0:00 xterm
        dany   2728  0.0  0.2   4528  1308 ?      Ss   Apr20   0:00 /bin/bash
        samy   2794  0.0  0.2   7060  1088 ?      Sl   Apr20   0:00 /usr/bin/VBoxClient --clipboard
        samy   3018  0.0  0.1   2496   740 ?      S    Apr20   0:00 python
        samy   3491  0.0  0.5   5656  2792 pts/2  Ss   Apr20   0:01 bash
        samy   9399 50.0  0.2   4476  1028 pts/2  R+   07:29   0:00 ps aux
```

על בסיס צורת הפלט הזו, רשום תוכנית Python, המדפיסה את שם המשתמש ושם התוכנית בעלת אחוז הזיכרון הגבוה ביותר.

```
import subprocess

p = subprocess.Popen("ps aux", shell=True)
max_mem = 0
max_user = ""
for line in p.stdout:
    user, pid, cpu, mem, vsz, rss, tty, stat, start, time, command = line.split()
    if mem>max_mem:
       max_mem  = mem
       max_user = user

print "Max memory =", max_mem
print "Max User =", max_user
```