מערכות הפעלה 31261

פתרון מבחן סופי, מועד א', סמסטר ב' תשע"ד, 30/06/2014

<u>הוראות לנבחן:</u> משך הבחינה שעתיים. חומרי העזר המותרים הם השקפים של הקורס שנמסרו באתר הקורס בלבד. שימוש במחשבונים ומכשירים אלקטרוניים כלשהם אסור בהחלט. רשום את תשובותיך במחברת במצורפת לבחינה. ציין בבירור את מספר השאלה במחברת והשתדל להיות **<u>קצר וענייני</u>** (תשובות ארוכות מדי או נסיונות לתת מספר תשובות אפשריות <u>ייפסלו</u> את תשובתך!). הקפד על כתב יד ברור ומסודר, ומחק את כל חומר הטיוטא. השימוש בשפה האנגלית מותר. השאלון מכיל 6 שאלות בשווי של 100 נקודות, ונפרש על פני 2 עמודים.

בהצלחה !

<u>שאלה1</u> [15%]

רשום שלוש <u>סיבות מרכזיות</u> לנחיצותה של מערכת הפעלה, או במילים אחרות: לשם מה אנו צריכים מערכת הפעלה במערכת מחשב? (יש לרשום שלושה דברים בדיוק! לא יותר!)

<u>פיתרון:</u>

מערכת הפעלה נחוצה בכדי:

א. לתווך בין המשתמש הלא מקצועי לחומרה המורכבת של מערכת המחשב ולאפשר ניצול מירבי של החומרה
ב. לספק למשתמש הרגיל סביבה בה יוכל להפעיל יישומים רבים במקביל באופן נוח, מוגן, ובטוח
ג. לחלק את משאבי המערכת בין המשתמשים והתהליכים השונים באופן הוגן, שקוף, יעיל ונוח

<u>שאלה 2</u> [15%]

תאר שלושה תפקידים **<u>מרכזיים</u>** של מערכת ההפעלה הקשורים לניהול מערכת הקבצים במערכת מחשב (יש לתאר שלושה תפקידים **<u>מרכזיים</u>** בלבד! לא יותר, לא פחות!)

<u>פיתרון:</u> (כל צרוף של שלושה מבין חמשת הדברים הבאים ייתקבל כתשובה מלאה)

א. <u>אירגון לוגי</u> פשוט של כל המידע המאוחסן במערכת התקני קלט ופלט - המאפשר למשתמש הרגיל שימוש נוח, מהיר, ויעיל בהתקנים אלה ללא צורך בהכרה או הבנה של המורכבות החומרה של התקנים אלה (שעשויה להיות מאוד מסובכת)
ב. ניהול שטחי דיסק פנויים באופן יעיל והוגן
ג. מדיניות ברורה והוגנת להקצאת שטחי דיסק בין משתמשים ותהליכים במערכת
ד. תיזמון יעיל של תורי בקשות פעולות קלט/פלט (של תהליכים שצורכים או מספקים מידע להתקנים)
ה. הגנה (עם אפשרות גיבוי), בטיחות ושמירת פרטיות באמצעות <u>מערכת הרשאות פשוטה</u> ונוחה לשימוש

<u>שאלה 3</u> [20%]

א. מנה את כל **<u>סוגי</u>** קריאות המערכת (system calls) שקשורות למינשק שבין מערכת ההפעלה והתקני החומרה שבסביבתה. תן בדיוק שתי דוגמאות ספציפיות של קריאות שרות כאלה עבור כל סוג.
ב. מנה את כל סוגי קריאות המערכת (system calls) שיש במערכת ההפעלה הדרושות לניהול תהליכים. תן דוגמא מתאימה עבור כל סוג קריאה (תוכל להביא דוגמאות מ-Linux או Windows לבחירתך).

<u>פיתרון:</u>

א. קריאות מערכת שקשורות להתקני חומרה מתחלקים לסוגים הבאים:
- יצירה, פתיחה, וסגירה של קבצים: creat(), open(), close(), ioctl()
- כתיבה וקריאה: read(), write()
- ניווט בתוך קובץ או מערכת תיקיות: lseek(), dirent(), opendir(), readdir(), chdir()
- ניהול הרשאות ובקרת גישה: chmod(), stat(), access(), chown()

ב. קריאות מערכת הקשורות <u>לניהול תהליכים</u> מתחלקים לסוגים הבאים:
- יצירה של תהליכים חדשים: fork(), execl(), execv(), execlp(), execvp()
- קבלת מידע: getpid(), getppid(), getuid(), getgid()
- ניהול אירועים: wait(), sleep(), signal(), waitpid()
- פעולות סיום תהליכים: exit(), kill()
- תיקשורת בין תהליכים: pipe(), socket(), fifo()

שאלה 4 [12%]

א. תמצת בקצרה את המטרה העיקרית של רכיב ה-DMA במערכת מחשב מודרנית.

ב. למה לדעתך יש כיום מחשבים בעלי 4 או 8 רכיבי DMA? נסה להסביר בקצרה מהם הייתרונות והחסרונות שיכולים להיות לריבוי רכיבי DMA?

<u>פיתרון:</u>

א. מטרת רכיב ה-DMA היא לבצע בקשות קלט/פלט בין הזיכרון והתקני איחסון איטיים (כגון דיסק קשיח רגיל וכדומה). בכך, רכיב ה-DMA משמש כעוזר משנה של יחידת העיבוד המרכזית, ומטרתו לשחרר אותה מפעולות מאוד איטיות ובכך לאפשר ניצול יותר טוב של יחידת העיבוד עבור פעולות חישוב מהירות יותר.

ב. ריבוי רכיבי DMA מאפשר ביצוע מקבילי של פעולות קלט/פלט, ובכך מגדיל את נצילות המערכת ומשחרר את יחידת העיבוד עבור משימות חישוב בלבד. החסרון הוא במורכבות הגדולה יותר שנדרשת לשם ניהול מספר רכיבי DMA הפועלים במקביל. חסרון נוסף הוא בריבוי אפשרי של פסיקות שעשוי לנבוע מחלוקה של פעולת קלט/פלט בין כמה רכיבי DMA שונים שאינם בהכרח מתואמים ביניהם.

שאלה 5 [24%]

רשום תוכנית קצרה בשפת C או בשפת Python שמשתמשת בקריאות מערכת (כמו: fork, waitpid, exit, pipe write), ומבצעת את הדברים הבאים:

א. תהליך אב parent מוליד שני תהליכים בנים: child1, child2

ב. תהליך הבן child1 שולח את ההודעה "Hello from child 1" לתהליך parent

ג. תהליך הבן child2 שולח את ההודעה "Hello from child 2" לתהליך parent

ד. התהליך parent מדפיס את שתי ההודעות בסדר הנ"ל (באותו סדר!) לפלט הסטנדרטי

ה. וודא כי התוכנית שלך יוצרת <u>בדיוק שלושה תהליכים</u> שונים: אב ושני בנים

ו. וודא כי שתי ההודעות הנ"ל מגיעות לתהליך האב בסדר הנ"ל

ז. וודא כי סדר הדפסת ההודעות על ידי האב אינו אקראי!

<u>פיתרון:</u>

יש לשים לב כי הילד הראשון נוצר בשורה 17, ואילו הילד השני נוצר בשורה 13, וזהו גם הסדר הזמני שבו מתבצעות שורות אלה.
הסיבה לכך היא שבשורה 7, תהליך האב ממתין לסיומו של תהליך בן 1. לכן הילד השני לא יתחיל לפעול עד שהילד הראשון מסיים. בשלב זה, תהליך האב קלט את הודעת תהליך בן 1 והדפיס אותה לפלט הסטנדרטי.

```
1  import os, sys
2
3  rside, wside = os.pipe()
4
5  pid1 = os.fork()
6  if pid1: # parent
7      os.waitpid(pid1,0)
8      print os.read(rside, 100)
9      pid2 = os.fork()
10     if pid2: # parent
11         os.waitpid(pid2,0)
12         print os.read(rside, 100)
13     else: # child 2
14         message = "Hello from Child 2"
15         os.write(wside, message)
16         sys.exit(0)
17 else: # child 1
18     message = "Hello from Child 1"
19     os.write(wside, message)
20     sys.exit(0)
```

שאלה 6 [14%]

משתנה הסביבה PATH במערכת ההפעלה Linux מכיל רשימה של כל התיקיות המכילות את קובצי ההפעלה שניתן להפעיל משורת הפקודות BASH. שים לב כי רשימה זו מופרדת על ידי התו ":".

```
Find all directories of exectables I can run
samyz@brdlinux:~> echo $PATH
/home/samyz/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/games
```

בשפת התיכנות Python ניתן לגשת לרשימה זו באמצעות המשתנה os.environ['PATH'].
רשום פונקציה קצרה בשם which המקבלת שם של פקודת Linux (כגון: ls, wc, או chmod) ומחזירה את המסלול הראשון שמכיל את הפקודה.
דוגמאות שימוש:

```
which('ls') = '/usr/bin/ls'
which('copyfile') = '/home/samyz/bin/copyfile'
```

פיתרון:

```
1   import os
2
3   def which(cmd):
4       Paths = os.environ['PATH'].split(':')
5       for path in Paths:
6           cmdpath = path + '/' + cmd
7           if os.path.exists(cmdpath):
8               return cmdpath
9
10  if __name__ == '__main__':
11      print which('ls')
12      print which('wc')
13      print which('pwd')
```