מטלת מנחה (ממ"ן) 12

הקורס: "מערכות הפעלה"

חומר הלימוד למטלה: ראו פירוט בסעיף "רקע"

מספר השאלות: 5 **משקל המטלה: 12**

סמסטר: 2019א **מועד אחרון להגשה:** 13.12.2018

הגשת המטלה: שליחה באמצעות מערכת המטלות המקוונת באתר הבית של הקורס.
הסבר מפורט ב"נוהל הגשת מטלות המנחה".

החלק המעשי (80%)

בחלק זה של המטלה נממש מנגנון של job control ב command interpreter עם פונקציונליות מוגבלת.

מטרת התרגיל: תהליכים, תקשורת בין התהליכים, job control.

רקע

1. סיפקנו את הקובץ shell.c אותו אתם אמורים לשנות ולהרחיב ובפרט להוסיף את המנגנון של job control. קמפלו והריצו את התוכנית. עיינו בקובץ shell.pdf להסברים. כל השינוי מסתכם במספר מועט של שורות קוד אך כדי לבצע אותו עליכם להבין מספר נושאים להלן.
2. http://www.gnu.org/software/libc/manual/html_node/Executing-a-File.html הסבר עם פונקציות ממשפחת exec (אפשר להשתמש בפונקציה לבחירתכם). כמו כן סיפקנו את הקובץ exec.c שאפשר לקמפל ולהריץ.
3. פרקים 24.7.2, 24.7.3 מ http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הסבר על פונקציות signal, sigemptyset, sigfillset, sigaddset, sigprocmask, sigsuspend. סיפקנו קובץ suspend.c. תקמפלו תריצו והבינו.
4. פרק http://www.gnu.org/software/libc/manual/html_mono/libc.html#Pipes-and-FIFOs עם הסבר על פונקציית pipe לתקשורת בין התהליכים. ספקנו קובץ pipe.c. תקמפלו תריצו והבינו.

5. פרק 13.12 מ
https://www.gnu.org/software/libc/manual/html_mono/libc.html#Duplicating-Descriptors עם הסבר על פונקצית dup. תקמפלו ותריצו את dup.c שספקנו כדי לראות שימוש ב dup. dup.c מהווה וריאציה של pipe.c מהסעיף הקודם ומדגימה כיצד ניתן ליצור ערוץ תקשורת בין שני תהליכים בצורה שהיא שקופה לתהליכים עצמם.
6. פרק 14.1 מ http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הסבר על הפונקציה chdir.
7. פרקים 27.6.2, 27.7.3 ו 27.7.2 מ
http://www.gnu.org/software/libc/manual/html_mono/libc.html עם הפונקציות getpgrp, setpgrp, tcgetpgrp.
8. כמו כן יש להיזכר בפונקציות wait, fork.

כמו ניתן לקבל מידע על הפונקציות הנ"ל מה man של LINUX.

תיאור המשימה

סיפקנו את הקובץ shell.c אותו אתם אמורים לשנות ולהרחיב. בפרט עליכם לממש מנגנון של job control ב command interpreter בשם smash (small shell). אחרי ההרחבה smash יהיה מסוגל:

1) לאפשר שרשור של לפחות 2 פקודות.
2) לתמוך בפקודות פנימיות exit ו cd.
3) להריץ תוכניות ברקע ובזמן אמת (background ו foreground).
4) לתמוך בפקודות bg, fg, jobs ולהגיב לסיגנלים של job control.

קיבלתם קובץ shell.c המממש פונקציונליות 1, 2, 3. כתבו עבורו Makefile שמייצר קובץ הרצה smash והריצו אותו משורת הפקודה:

```
maman12$ ./smash
```

כמו כן קיבלתם את קובץ smash_SSol המממש גם את 4. הריצו אותו משורת הפקודה:

```
maman12$ ./smash_SSol
```

במטלה הזאת עליכם לכתוב כ 40שורות קוד למימוש בפקודות של job control (סעיף ד). שאר הפונקציונליות כבר ממומשת (סעיפים א,ב,ג). אבל למימוש ה job control עליכם להבין כיצד פועלים שאר הדברים. המיקום של השורות אותן תצטרכו לממש מופיע בקובץ shell.pdf יחד עם ה והפסאודו-קוד. מי שמכיר מהו שרשור הפקודות ב shell ואת הפקודות jobs, fg, bg יכול לעבור ישירות לקריאת ההסברים הקובץ shell.pdf. אחרת, מומלץ קודם לקרוא הסברים מטה.

<u>הרצת תוכניות ברקע ובזמן אמת</u>

הרצת תוכנית(תוכניות) בזמן אמת (foreground) גורמת ל command interpreter להמתין עד סיום התוכנית (תוכניות). למשל

```
# ls
# ps | wc –l
```

הן דוגמאות להרצת תוכניות בזמן אמת.

הרצת תוכנית(תוכניות) ברקע (background) לא גורמת ל command interpreter להמתין עד לסיום התוכנית (התוכניות) שרצות ברקע. הרצת תוכניות ברקע תתבצע ע"י הוספת "&" בסוף שורת הפקודה. למשל:

```
# find / home -name Makefile –print &
# chown –R root: root /tmp&
```

שרשור של פקודות

smash מאפשר שרשור של לפחות שני פקודות בשורת פקודה אחת. השרשור מתבצע ע"י סימן "|" (pipeline) בין הפקודות. משמעות השרשור היא שפלט של הפקודה הראשונה מהווה קלט לפקודה השנייה. כך למשל הרצת

```
# cat /etc/passwd | wc –l
```

גורמת לספירת כמות השורות בקובץ /etc/passwd. הפקודה "cat /etc/passwd" מדפיסה את תוכן הקובץ /etc/passwd ל stdout. באמצעות ה "|" אפשר "לומר" ל smash להפנות את הפלט של cat לתוכנית wc . והתוצאה שהיא כמות השורות בקובץ תודפס על הצג.

תמיכה בפקודות של job control

smash יתמוך בפקודות הבאות:

1) jobs – הפקודה תגרום להדפסה של כל התהליכים **<u>המושהים</u>** ושל כל התהליכים **<u>שרצים</u>** **<u>ברקע</u>** אשר הורצו בעבר מתוך smash. תהליך מושהה הוא תהליך שהיה רץ בזמן אמת ואשר הושהה (למשל באמצעות Ctrl-Z). אם תהליך כלשהו רץ בזמן אמת, הצירוף Ctrl-Z משהה את ריצתו ומחזיר את שורת ה prompt של smash. מכאן שאם רוצים להריץ פקודת jobs ייתכן ויהיה צורך להשהות קודם תהליך שרץ בזמן אמת. לדוגמא:

```
# find / home -name Makefile –print
<Ctrl-Z>
[1] Stopped          find /home -name Makefile –print
# jobs
[1] Stopped          find /home -name Makefile –print
#
```

פקודת jobs נותנת לתהליכים מספר סידורי פנימי (ששונה בד"כ מ pid של תהליך) לפיו ניתן לזהות באופן יחיד כל תהליך שעדיין לא הסתיים ואשר הורץ מתוך smash.

2) fg %N – הפקודה תגרום להרצת תהליך [N] בזמן אמת. כך בדוגמא הקודמת הרצת

```
# fg %1
```

תעביר את find לרוץ בזמן אמת ולא תחזיר את ה prompt של ה smash עד לסיום ה find או עד להשהיתו הבאה.

3) bg %N – הפקודה תעביר את התהליך [N] ממצב מושהה למצב רץ ברקע.

תמיכה בפקודות פנימיות

smash יתמוך בשתי פקודות פנימיות:

א) exit – בעקבות הקשת הפקודה יסיים smashאת פעולתו.

ב) cd - בעקבות הקריאה לפקודה זו ישנה smash את ספרית העבודה הנוכחית שלו.

טיפול בשגיאות

smash צריכה לתת הודעות שגיאה על כשלון של קריאות מערכת או פונקציות שמכילות קירות מערכת. במקרה של שגיאה פטאלית יש לצאת עם סטטוס 1 (ע"י exit(1)).

הגשה

יש להגיש **כל** קבצי הקוד ו Makefile המייצר קובץ הרצה smash. אין להגיש קבצים מקומפלים. את הקבצים המוגשים יש לשים בקובץ ארכיון בשם exYZ.zip (כאשר YZ הנו מספר המטלה). הכנת קובץ ארכיון מתבצעת ע"י הרצת הפקודה הבאה משורת הפקודה של Linux:

zip exYZ.zip <ExYZ files>

הערה חשובה: בכל קובץ קוד שאתם מגישים יש לכלול כותרת הכוללת תיאור הקובץ, שם הסטודנט ומספר ת.ז.

פתרון ביה"ס

קיבלתם את קובץ smash_SSol כפי שמומש על ידינו.

החלק עיוני (20%)

שאלה 1 – (5%)

מצב לא בטוח איננו גורר באופן מיידי את מצב הקיפאון. תנו דוגמא לכך.

שאלה 2 – (5%)
האם דף יכול להיות בו זמנית בשתי קבוצות עבודה (working sets)? נמקו.

שאלה 3 – (5%)
תארו מצב שבו התמיכה בזיכרון וירטואלי מסתברת כרעיון לא מוצלח. מה אפשר לשפר במצב שתיארתם אם אין תמיכה בזיכרון הוירטואלי?

שאלה 4 – (5%)

בעזרת תוכניות העזר size ו file גלו מהו גודלן הממוצע והחציון של קבצי הרצה במערכת הפעלה שסיפקנו לכם (Ubuntu 16.04). הסתכלו רק על קבצי ההרצה (לא על קבצי scripts) בספריות:

- /bin
- /usr/bin

הדגימו את החישוב של הגודל האופטימאלי של דפים במערכת בהתבסס על גודל ה text segment של קבצי ההרצה שמצאתם? הניחו שגודל של entry בטבלת הדפים הוא 4 בתים. הניחו שלכל קבצי ההרצה הסתברות זהה לרוץ ושהם מקבלים יחס זהה מבחינת מערכת ההפעלה. קחו בחשבון את הריסוק הפנימי.

הגשת החלק העיוני

החלק העיוני יוגש כקובץ Word או כקובץ pdf. שם הקובץ צריך להיות exYZ.pdf או exYZ.doc (כאשר YZ הנו מספר המטלה).

בדיקה לאחר ההגשה

לאחר ההגשה יש להוריד את המטלה (חלק מעשי/עיוני) משרת האו"פ למחשב האישי ולבדוק שהקבצים אכן הוגשו באופן תקין ושניתן לקרוא אותם. בנוסף, הבדיקה של החלק המעשי תכלול את הצעדים הבאים:

- פתיחת ארכיון *exXY.zip* בספרייה חדשה (*new folder*).
- וידוא שכל הקבצים הדרושים נוצרו בספריה בה פתחתם את הארכיון.
- הרצת *make* ווידוא שכל ה *targets* נוצרו ללא שגיאות וללא *warnings*
- הרצת בדיקות רלונטיות לווידוא תקינות הריצה של החלק המעשי