עבודת מסכמת: מערכות הפעלה 2020, סמסטר ב' (הגשה עד 13/08/2020).

גרסא : 2
תאריך עדכון: 15/07/2020

הוראות חשובות לגבי כל תרגילי התכנות במטלה זו:

1. שגיאות קומפילציה יגררו ציון אפס על השאלה המתאימה. הסבירו במדויק בקובץ README איך לקמפל ולהריץ את הקוד שלכם.
2. כל הקבצים שאתם מגישים צריכים להיות בתוך ספריה בשם< fwork_<your_id. כל שאלה ופתרונה יהיו בספריה נפרדת בשם < q_<number
3. מידע נוסף לגבי הבחינה "הפרונטאלית"(zoom) ינתן בהמשך.
4. נא להקפיד על כל הוראות ההגשה המפורטות במסמך. על מנת שניתן יהיה לבדוק את העבודות ביסודיות ובזריזות נדרש שיתוף הפעולה מצדכם.
5. אנו ממליצים לבצע את המטלה ב**סביבה הווירטואלית** של Ubuntu.
6. **<u>ביצוע המטלה אישי בלבד!</u>**

שאלה 1 (25 נק'): מבנה הזכרון של תוכנית בשפת C.
בשאלה זו תתנסו במבנה הזכרון של תוכנית בשפת C הכולל את הקטעים **heap, stack**, **text** וכו'. נתונה התוכנית program. עליכם לענות על כל השאלות בהערות (comments). למטרה זו באפשרותכם להשתמש בכלים הבאים : **size**, **nm**, **objdump** (למדו איך להשתמש בהם).

מה עליכם לבצע:

0) למדו איך להשתמש ב- **size**, **nm**, **objdump.** השתמשו ב- **man** או מקורות אחרים לפי בחירתכם.

עליכם להכיר את הכלים האלה לפחות ברמה שתאפשר לכם לבצע את המטלה.

1) החליפו כל הערה (comment) שיש בה שאלה בתשובה, בתשובה של שורה אחת בתוך התכנית המקורית.כל השאלות ממוספרות. שמרו על אותו מספור בתשובות שלכם.
2) יש ליצור קובץ pdf נפרד, שבו תסבירו כל אחת מתשובותיכם. כמו כן הוסיפו פלט של הכלים(הנ"ל) שהשתמשתם בהם, שמאשר את התשובה שלכם. יש להשתמש באותו מספור. לצורך נוחות הבדיקה, אנא העתיקו לפני כל תשובה את השאלה המקורית + שורת הקוד המתאימה מהתוכנית המקורית.

מה עליכם להגיש:

1. תכנית C עם תשובות של שורה אחת מסעיף 1 שנקראת **q1_<your_id>.c**
2. קובץ pdf מסעיף 2 ששמו **q1_<your_id>.pdf**

חשוב:
אם אחד הקבצים חסר, הציון על כל השאלה יהיה אפס. הסבר חסר על שאלה מסויימת בקובץ המקור או בקובץ ה- pdf הנוסף - הציון יהיה אפס. היזהרו, לא יתקבל כל חומר משלים לאחר מועד ההגשה !

מה צריך לדעת לקראת הבחינה "הפרונטאלית" (zoom):
יש להבין את מבנה הזכרון (memory layout), אתם עשויים להשאל שאלות שונות על הנושא של המטלה. כמו כן, יתכן שתתבקשו להריץ ולהשתמש בכלים (הנ"ל) שהשתמשתם לצורך המטלה.
לדוגמא - איך המחסנית ממומשת, מתי עדיף להשתמש במחסנית עבור הקצאת זכרון ומתי בערימה, וכד'.

שאלה 2 (25 נק'): signals

המשימה שלכם היא לבדוק האם קיים תהליך (process) עם pid מסויים. דרך אחת היא להשתמש בשליחת **kill(<pid>, 0)**, כאשר סיגנל 0 (zero signal) נתון ואפשר להשתמש בו. אם שליחת סיגנל 0 נכשלת עם הודעת שגיאה ESRCH, אנחנו יודעים שהתהליך אינו קיים. אם הקריאה נכשלה עם הודעת שגיאה EPERM (התהליך קיים אבל אין לנו הרשאה לשלוח לו סיגנל כזה) או מצליחה (אם יש לנו הרשאה לשלוח סיגנל כזה), אז אנחנו יודעים שהתהליך קיים.
צריך להוסיף include errno.h כדי לקבל הודעות שגיאה כאלה.

מה עליכם לבצע:
0) כתבו תכנית בשם **check_pid.c** המקבלת פרמטר יחיד pid והפלט של התכנית הוא :
If EPERM, Process <pid> exists but we have no permission.
If ESRCH, Process <pid> does not exist.
If kill is successful, Process <pid> exists.

Running example: check_pid 2003
Process 2003 exists.

1) הציעו 2 שיטות נוספות לבדיקה הנ"ל. פרטו יתרונות וחסרונות של 3 השיטות הנ"ל. כתבו את תשובתכם בקובץ pdf נפרד בשם **q21_<your_id>.pdf**

מה עליכם להגיש:
check_pid.c, **makefile** to compile, **README** how to run, and **q21_<your_id>.pdf**

חשוב:
אם אחד הקבצים חסר, הציון יהיה אפס על כל השאלה.

2) בחלק זה תראו שמספר הסיגנלים המתקבלים יכול להיות קטן ממספר הסיגנלים שנשלחים. במילים אחרות, סיגנלים אינם נכנסים לתור. על מנת להראות זאת, כתבו 2 תכניות **server.c, client.c**. ה-client יכול לשלוח 2 סוגים של סיגנלים: SIGINT,SIGUSR1.
בתוך ה-server ה- handler של SIGINT סופר את מספר הסיגנלים SIGINT שמתקבלים.
ה-handler של SIGUSR1 מדפיס את מספר הסיגנלים SIGINT שהגיעו ל-server.
ה- client מקבל 3 פרמטרים בסדר הבא: **pid** של ה-server, מספר סיגנל SIGUSR1(10) או SIGINT(2)
ומספר הסיגנלים שישלחו. לדוגמא:

1) client <server pid> 2 1000
2) client <server pid> 10 1

בדוגמה הראשונה client ישלח 1000 סיגנלים SIGINT לשרת עם <server pid>.
בדוגמה השניה השרת עם <server pid> ידפיס את מספר ה- SIGINT שהתקבלו.

מה עליכם לבצע:

0) כתבו את התכניות client.c ואת server.c המבצעות את מה שהוסבר.
1) עליכם להראות שלא כל הסיגנלים מתקבלים.
2) למדו על real time signals שנצברים לתור (**sigqueue**, **sigaction**, etc). הסבירו את היתרונות והחסרונות של שתי השיטות. הוסיפו את ההסברים לקובץ **q22_<your_id>.pdf**.

מה עליכם להגיש:

client.c, **server.c**, **makefile** how to compile, **README** how to run, and **q22_<your_id>.pdf**.

חשוב:

אם אחד הקבצים חסר, הציון יהיה אפס על כל השאלה.

מה צריך לדעת לקראת הבחינה "הפרונטאלית" (zoom):

עליכם לדעת טוב את סוגי שיטות ה- IPC שלמדתם בקורס כמו pipe, mkfifo, shared memory.
יש לדעת איך הם ממומשים. אינכם צריכים לזכור את כל ה- API, אבל אתם צריכים לדעת לחפש אותם ב- **man** ולהסביר איך הם פועלים.

שאלה 3 (25 נק'): CPU scheduling

בשאלה זו תתנסו בתזמון של CPU ב- Linux.

מה עליכם לבצע:

1) עליכם ללמוד על הפקודות **chrt, renice, taskset** ב- Linux. לדוגמא, **chrt** נועד לניהול פרמטרי זמן-אמת של התהליך. פקודה זו מחזירה או משנה את ערכי פרמטרי זמן-אמת של תהליך pid, או מריצה פקודה עם פרמטרים מסויימים. לצורך הלימוד תוכלו להשתמש ב- **man** או כל מקור אחר (למשל כאן).
למדו איך אפשר לשנות את מדיניות התזמון והעדיפויות של תהליך. מה ההבדל בין **chrt** ו- **renice** בתזמון CPU ? מה עושה **taskset** ? כמו כן עליכם להבין לפחות את מנגנוני התזמון הבאים:
SCHED_DEADLINE, SCHED_FIFO, SCHED_IDLE, SCHED_RR, SCHED_OTHER.

2) לאחר שהבנתם את הנדרש בסעיף 1 עליכם לכתוב תכנית בשם **set_policy.c** שמגדירה מדיניות תזמון וקדימויות של תהליך. system call **sched_setscheduler()** משנה גם את מדיניות התזמון ואת הקדימויות של תהליך pid. אם pid=0, הפרמטרים של התהליך הקורא משתנים.
Set_policy מקבל 2 פרמטרים: מספר המייצג את אחת מהמדינויות (למשל **SCHED_DEADLINE, SCHED_FIFO, SCHED_IDLE, SCHED_RR, SCHED_OTHER**) ומספר integer המייצג עדיפות.

מה עליכם להגיש:

set_policy.c, **makefile** to compile, **README** how to run, and **q32_<your_id>.pdf** containing output of commands confirming changed values.

חשוב:

אם אחד הקבצים חסר, הציון יהיה אפס על כל השאלה.

מה צריך לדעת לקראת הבחינה "הפרונטאלית" (zoom):
עליכם להבין איך תזמון ה- CPU ב- Linux עובד. עליכם להבין לפחות את כל מה שהוזכר בסעיף 1

שאלה 4 (25 נק'): files in Linux
בשאלה זו תעסקו בקבצים וה- metadata שלהם ב- Linux.

1) עליכם להבין את המושגים השונים בהקשר של מערכת הקבצים כגון:
file descriptors, inods, directories,soft and hard links in Linux.
כדי לבדוק metadata של קובץ אפשר להשתמש ב- **stat**.
כמו כן, עליכם להבין בכל הנושאים שעסקתם בהם במטלה 3.

2) בחלק זה תשתמשו בפונקציה **nftw**(), המאפשרת לסרוק ספריה שלמה באופן רקורסיבי תוך ביצוע מספר פעולות (כגון קריאה לפונקציות המוגדרות ע"י המשתמש) לכל קובץ בעץ הקבצים. עליכם לכתוב תוכנית בשם **dir_traversal.c** שעוברת באופן רקורסיבי על ספריה נתונה. על כל קובץ או ספריה התוכנית מדפיסה : סוג (לפי הדוגמא שבהמשך), מספר inode ושם. אם הספריה מכילה soft link, יש להתעלם ממנו.

Example:

```
$ mkdir dir
$ touch dir/a dir/b
$ ln -s a dir/sl                    ←- later ignored from the output
$ mkdir dir/sub
$ touch dir/sub/x

$ ./dir_traversal dir
D 2327983 dir
F 2327984 a
F 2327985 b
D 2327988 sub
F 2327989 x
```

מה עליכם להגיש:
dir_traversal.c, **makefile** to compile, **README** how to run, and **q41_<your_id>.pdf** containing the running output for the specified example.

חשוב:
אם אחד הקבצים חסר, הציון יהיה אפס על כל השאלה.

מה צריך לדעת לקראת הבחינה "הפרונטאלית" (zoom):
עליכם להבין את מנגנון הקבצים ב- Linux ואיך מתפעלים אותו. במיוחד עליכם להבין את כל מה שעסקתם בו ב- סעיף 1 של שאלה זו.

בהצלחה רבה ובריאות איתנה לכולם