תרגיל 2 - מיני מעטפת – תרגיל המשך
מערכות הפעלה, סמסטר ב' תשפ"ה
להגשה: 13.05.2025

בתרגיל זה יש להוסיף את האלמנטים הבאים לתרגיל 1

1. **מימוש מנגנון pipe**
2. **תמיכה בפקודה הפנימית my_tee**
3. **מימוש מנגנון resource limit**
4. **תמיכה ב &**
5. **תמיכה ב 2>**
6. **מימוש בצורה מורחבת מנגנון שגיאות**

מבוא ורקע:

1. מימוש מנגנון pipe

הפקודה pipe מאפשרת העברת פלט של פקודה אחת כקלט לפקודה אחרת. מנגנון זה מיושם באמצעות התו | בין שתי פקודות.

דוגמאות:

```
$ ls | grep txt
file1.txt
file2.txt

$ ps aux | grep firefox
```

על המימוש לכלול:

- זיהוי התו | בפקודה. יהיה עם רווח אחד בלבד לפניו ואחריו
- ממשו תמיכה של | בודד. כלומר אין צורך לתמוך ב Cmd1 | cmd 2 | cmd 3

2. תמיכה בפקודה הפנימית my_tee

הפקודה my_tee היא גרסה משלכם לפקודת tee המערכתית. עם בדיוק אותה פונקציונליות! היא מאפשרת לקרוא מהקלט הסטנדרטי (stdin) ולכתוב את אותו מידע הן לפלט הסטנדרטי (stdout) והן לקובץ (או קבצים) במקביל.

חשוב: עליכם לממש את הפקודה my_tee כפקודה פנימית של המעטפת. אין להשתמש בפקודת tee המערכתית הקיימת.

המימוש של `my_tee` צריך לתמוך ב:

- קריאת מידע מ stdin שיגיע דרך ה pipe-במקרה של שימוש עם `|`
- כתיבת המידע ל stdout
- כתיבת המידע לקובץ או קבצים שצוינו כארגומנטים
- תמיכה באופציה `-a` (append) אשר מוסיפה את המידע לסוף הקובץ במקום לדרוס אותו

הסבר:

- `command | my_tee file.txt` יריץ את `command`, ישלח את הפלט שלה למסך וגם יכתוב אותו לקובץ `file.txt`.
- `command | my_tee -a file.txt` כמו הקודם, אבל יוסיף (append) את המידע לסוף הקובץ במקום לדרוס אותו.

דוגמא:

```
$ echo "Hello World" | my_tee output.txt
Hello World
$ cat output.txt
Hello World
$ echo "Second line" | my_tee -a output.txt
Second line
$ cat output.txt
Hello World
Second line
$ echo "Hello" | my_tee file1.txt file2.txt
Hello
$ cat file1.txt
Hello
$ cat file2.txt
Hello
```

3. מימוש מנגנון resource limit

מנגנון resource limits מאפשר למערכת ההפעלה להגביל את כמות המשאבים שתהליך מסוים יכול לצרוך. הגבלות אלו מסייעות למנוע מצבים בהם תהליך אחד משתלט על משאבי המערכת ומשפיע על ביצועי המערכת כולה או על תהליכים אחרים.

במערכות הפעלה, יש שני סוגי הגבלות לכל משאב:

- **הגבלה רכה - (soft limit)** כאשר התהליך מגיע להגבלה זו, הוא מקבל אות (signal) כגון SIGXCPU
- **הגבלה קשה - (hard limit)** הגבלה מקסימלית שלא ניתן לעבור;
-
-

יש לממש פקודה פנימית חדשה בשם `rlimit` שתאפשר להגדיר ולהציג מגבלות משאבים – **<u>ההגבלות יכלו על אותו הבן ספיציפי שנוצר שמריץ את הפקודה:</u>**

- `rlimit set [resource]=value[:hard_value] command` מגדיר הגבלת משאב ומריץ פקודה
- `rlimit show` מציג את ההגבלות הנוכחיות של כל המשאבים

תמיכה בסוגי המשאבים הבאים (חובה לתמוך בכל הארבעה) :

- `cpu` זמן מעבד בשניות
- `mem` גודל הזיכרון במונחי B, KB, MB, GB
- `fsize` גודל מקסימלי של קבצים שנוצרים
- `nofile` מספר מקסימלי של קבצים פתוחים

סינטקס להגדרת הגבלות:

- `rlimit set resource=soft_value[:hard_value] command [args...]`
- אם לא מצוין ערך hard, ההגבלה הקשה תהיה זהה לרכה
- יחידות זמן הן שניות עבור cpu
- יחידות גודל יכולות להיות B, K, KB, M, MB, G, GB עבור mem, fsize
- ניתן להגדיר מספר מגבלות באותה פקודה
- והדעות השגיאה מסומנות לכם בירוק

דוגמאות הפעלה:

```
$ rlimit set cpu=2:3 sleep 10
CPU time limit exceeded!
$ rlimit set mem=50M ./memory_intensive_program
Memory allocation failed!
```

~~$ rlimit show cpu~~

~~CPU time limits: soft=30s, hard=60s~~

```
$ rlimit show
CPU time: soft=30s, hard=60s
Memory: soft=unlimited, hard=unlimited
File size: soft=unlimited, hard=unlimited
Open files: soft=1024, hard=4096
```

~~$ rlimit set nproc=50 ./fork_bomb.sh~~

~~Process creation limit exceeded!~~

```
$ rlimit set fsize=1M dd if=/dev/zero of=output bs=1M count=10
File size limit exceeded!
```

```
$ rlimit set nofile=5 ls
Too many open files!

$ rlimit set cpu=1:2 mem=20M:30M ./compute_intensive
```

4. תמיכה ב &

התו & בסוף פקודה מכניס את הפקודה לרוץ "ברקע" ומאפשר למשתמש להקיש פקודות נוספות. נראה דוגמא:

```
$ sleep 5
[המעטפת " ישנה " למשך 5 שניות]
$ sleep 5 &
$ ls
file1.txt file2.txt
```

בדוגמא לעיל הפקודה הראשונה "sleep 5" מריצה את התוכנית sleep אשר "ישנה" ל-5 שניות ובמשך 5 שניות ה" shell-תקוע" ולא מאפשר להקיש פקודות נוספות. לעומת זאת הפקודה "sleep 5 &" מריצה את התוכנית ברקע וה shell-משתחרר מייד, והמשתמש יכול להקיש פקודות נוספות, בדוגמא שלנו הוא מריץ ls

על המימוש של & דיברנו בהרצאה – לכאורה אינו מסובך -- בגדול כל מה שנדרש לשנות מתרגיל 1, זה לבטל את ה wait- באב -- כאשר מזהים שהמשתמש הקיש & בסוף הפקודה, ולאפשר ללולאה לקלוט עוד פקודות מהמשתמש איפה כן נעשה את ה ?wait??.

5. תמיכה ב 2>

שירשור פלט שגיאות לתוך קובץ

התו 2> משמש להפניית פלט שגיאות (stderr) לקובץ במקום להציג אותו על המסך. זה מאפשר למשתמש לתפוס הודעות שגיאה ולנתח אותן בנפרד מפלט רגיל.

דוגמאות:

- `command 2> error_log` יריץ את הפקודה ויפנה את כל הודעות השגיאה לקובץ.error_log

6. שגיאה בעת הרצת התהליכים

בתרגיל 1 מימשתם בצורה מוגבלת את המנגנון של "כשלון" בהרצת הפקודות. יש להרחיב מימוש זה ולהשתמש בסטטוס של waitpid לבדיקת שגיאות.

מימוש נדרש:

1. **בדיקת קוד חזרה מלא** -יש להשתמש :

- `WIFEXITED(status)` האם התהליך הסתיים באופן נורמלי
- `WEXITSTATUS(status)` קוד החזרה של התהליך -- אם הסתיים נורמלית
- `WIFSIGNALED(status)` האם התהליך הסתיים בגלל סיגנל
- `WTERMSIG(status)` - איזה סיגנל גרם לסיום התהליך

2. **טיפול בסוגי סיום שונים** :
   - עבור תהליכים שהסתיימו בהצלחה (קוד חזרה 0), אין צורך בהודעה מיוחדת
   - עבור תהליכים שהסתיימו עם קוד שגיאה (קוד חזרה שונה מ-0), יש להדפיס הודעת שגיאה מתאימה
   - עבור תהליכים שהסתיימו בגלל סיגנל, יש להדפיס את שם הסיגנל (למשל "Terminated by signal: SIGSEGV")

הערות:

1. יש לתמוך רק ב <2-להפניית ערוץ השגיאות לקובץ ואין צורך לממש redirect אחר כגון <
2. עליכם לשמור על כל הפונקציונליות של תרגיל 1
3. במימוש `my_tee` , יש להשתמש בקריאות מערכת בסיסיות (`read`, `write`) ולא בפונקציות של ספריית stdio
4. במימוש `my_tee` אפשר להניח שהקלט מגיע רק מה pipe
5. במימוש מנגנון `resource limit` יש להשתמש בקריאות המערכת `setrlimit`ו`getrlimit`-
6. זכרו לשחרר זיכרון ולסגור קבצים כנדרש
7. זכרו לבדוק דליפות זיכרון עם valgrind

הגשה

יש להגיש בvpl קובץ אחד בשם ex2.c

בנוסף, יש להגיש במודל קובץ zip עם הקבצים הבאים:

- `ex2.c` קובץ המקור של המעטפת המורחבת
- `README` מסמך המתאר את המימוש והאתגרים

עבודה עצמאית, מצאתי פתרון ברשת או במקום אחר או אצל סטודנט אחר או שנה אחרת – זה לא עבודה עצמאית, וגם לא משהו דומה...אנחנו גם נבדוק את זה וגם נבחן אותכם על זה בסוף השנה. (**בבקשה, בבקשה, בבקשה!** זה רק בשבילכם, לא תעבדו עצמאית לא תלמדו...אין קיצורי דרך)

זיכרו לבדוק דליפות זיכרון עם valgrind