Operating Systems – 234123

**Homework Exercise 4 – Dry**

**מגישים:**

**נועה פריאנטה 206200305**

[pariente@campus.technion.ac.il](mailto:pariente@campus.technion.ac.il)

**ששון שמואל למעי 325172351**

sason-shmuel@campus.technion.ac.il

**שאלה 1 – ניהול זיכרון:**

נתון שהמחשב זה עתה עלה (מיד אחרי reboot), שהארכיטקטורה של המחשב היא x86/64bit, שמשתמשת יחידה בשם אליס משתמשת בו כרגע, שכל מה שאליס עשתה עד עתה זה להריץ shell על המחשב (כחלק מתהליך ה-login), שקיים קובץ בשם my\_file.txt בתיקיית העבודה הנוכחית של אליס בגודל 32KB, ושהתהליך הראשון שאליס מריצה ב-shell מבצע את קטע הקוד הבא:

|  |  |
| --- | --- |
| #define PAGE\_SIZE (4\*1024) | 1 |
| int main()  { | 2 |
| int fd = open("./my\_file.txt", O\_RDWR); | 3 |
| char\* buffer = malloc(2 \* PAGE\_SIZE); | 4 |
| read(fd, buffer, 2 \* PAGE\_SIZE); | 5 |
| char\* array = (char\*) mmap(NULL, 4 \* PAGE\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0); | 6 |
| char x = array[12]; | 7 |
| char z = array[12 + 2 \* PAGE\_SIZE]; | 8 |
| array[12 + 2 \* PAGE\_SIZE] = x; | 9 |
| return 0; | 10 |
| } | 11 |

1. מה המספר **המינימלי** של מסגרות פיזיות חדשות שמוקצות (בעבור data בלבד, מבלי להתחשב ב-page table) בכל אחת מהשורות הבאות?

|  |  |  |
| --- | --- | --- |
| **שורה** | **מספר מסגרות** | **הסבר קצר** |
| 3 | 0 | בשורה זו מבצעים קריאת מערכת open. קריאה זו יוצר FD ב-FDT בלבד, בלי להקצות מסגרת לקובץ. |
| 4 | 0 | מכיוון שנשאלנו על מספר מינימלי של מסגרות, אנו פועלים בגישה העצלנית: לא מקצים מסגרות עד שלא מתבצעת כתיבה. |
| 5 | 4 | מקצים שני דפים למטמון ושני דפים ל-buffer |
| 6 | 0 | שוב אנו פועלים בגישה העצלנית ולכן לא מקצים דפים לפני ביצוע פעולת כתיבה. |
| 7 | 0 | נצביע לדפים שהקצנו במטמון (שורה 5) |
| 8 | 1 | אנחנו מבצעים חריגה מכמות המסגרות שהקצנו, ולכן במסגרת ה-page fault נקצה עוד מסגרת אחת (זה מה שנחוץ במקרה זה) ונוסיף למטמון. |
| 9 | 0 | בקריאה mmap העברנו את הדגל MAP\_SHARED ולכן לא נבצע COW ונקצה עוד מסגרות. |

2. כיצד תשתנה תשובתכן לסעיף 1 אם בשורה 6 היה רשום MAP\_PRIVATE במקום MAP\_SHARED? ציינו באיזה שורות תשובתכן הייתה **משתנה** והסבירו:

**תשובה:** שורה 9 הייתה משתנה מ-0 ל-1 כי כעת מנגנון COW יופעל ונעתיק את המסגרת למרחב משתמש. נדגיש שרק את המסגרת הזאת כי אנחנו פועלים באופן עצל.

3. כיצד תשתנה תשובתכן לסעיף 1 אם מיד כאשר התהליך הנ"ל מסתיים הוא מורץ שוב? נמקו (השאלה מתייחסת להרצה השנייה.)

**תשובה:** שורה 5 תשתנה. מכיוון שמודבר בהרצה שנייה אז המסגרות כבר קיימות במטמון הדפים ולכן נקצה רק ל-buffer. באופן דומה בשורה 8 נקצה 0 (כי בהרצה ראשונה הקצנו רק 1 למטמון)

4. כאשר מורץ הקוד לראשונה, מה הוא המספר **המקסימלי** של מסגרות פיזיות חדשות שמוקצות בעבור **טבלת הדפים** של התהליך? **יש לשרטט את המסגרות של טבלאות הדפים בשביל נימוק התשובה**.

הנחות: (1) התעלמו ממסגרות המוקצות עבור המחסנית (stack).

(2) הניחו שהכתובת של buffer (שורה 4) היא:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 0 | 20 12 | 29 21 | 38 30 | 47 39 |
| 000000000000 | 111111111 | 111111111 | 000000**1**00 | 000000000 |

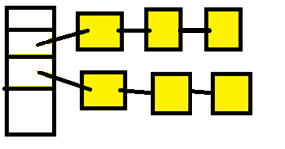
(3) הניחו שהכתובת של array (שורה 6) היא: (שימו לב להבדל בכתובות)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 0 | 20 12 | 29 21 | 38 30 | 47 39 |
| 000000000000 | 111111111 | 111111111 | 000000**0**00 | 000000000 |

**תשובה:** התשובה הינה 9

כפי שציינו אנו מקצים 4 מסגרות חדשות**.** נסים לב שברמה 30-38 יש הבדל בביט השלישי ולכן מההיררכיה כפי שראינו בהרצאה הו יוקצו ב2 טבלאות שונות. כעת מכיוון ולכן נקצה 2 מסגרות עבור זה ולכן התשובה היא 6 מסגרות. את הטבלה הראשית לא סופרים כי היא תמיד מוקצית, לכן תשובה סופית 6.

צרייך לתקןןןןן



1. הגדירו major page fault.

**תשובה:** חריגת דף שמתקבלת כאשר הדף לא בזיכרון ויש להביא אותו מהדיסק.

1. מה מספרי השורות בהן מתרחש major page fault? נמקו

**תשובה:** השורות בהן מקצים מסגרות, שזה 5 ו-8.

1. כיצד תשתנה תשובתכן לסעיף **6** אם בשורה 6 היה רשום MAP\_PRIVATE במקום MAP\_SHARED? נמקו

**תשובה:** התשובה לא הייתה משתנה. בשינוי של הדגל גורם לכך שנעתיק את הדף למרחב המשתמש (בעת שימשו במנגון COW) הדף כן נמצא בזיכרון משורות לפני בקוד איך אינו מסומן ביחידת ניהול הזיכרון שנטען בזיכרון.

**שאלה 2 – ניהול זיכרון:**

שאלה זו כתובה באנגלית מאחר ושהיא מכילה לא מעט מושגים ושמות אשר קל יותר לבטאם באנגלית. נא לפתור אותה באיזה שפה שתרצנה\ו.

In the wet part of this homework, you implemented an interface that manages dynamic memory in for a process.

In this part of the homework, you will analyze the existing malloc() (from <cstdlib>) while learning about some new Linux tools.

**NOTE:** Do NOT submit code you write in this homework with your wet submission. Simply copy your code to your dry submission file, wherever requested.

## Section 1:

1. Look up the “strace” utility online, read a little bit, and try to use it yourself by running `strace ls` in your OS terminal. Finally, explain here in a few words what the it does.

Strace הינה פקודה אשר עוזרת לדיבוג. היא מחזירה רשימה של קריאות המערכת שהתוכנית שהיא קיבלה בקלט (למשל על ידי pid או עלי ידי ריצת פקודה כארגומנט אליה וכו'...) ואת הפרמטרים וערכי החזרה שלהם. כמו כן עבור signals. היא מאפשרת לפלטר עליהם בנוחות וזהו עוזר להבין את התנהגות המערכת ולדבג.

פעולת 'strace ls' הציגה את קריאות המערכת שהפקודה ls ביצעה למשל

A screen shot of a computer program

Description automatically generated

ניתן לראות שיצרה שני thread בהתחלה.

1. Write a simple program in C that receives a number “x” from the command line and allocates (using malloc()) a block of memory that is “x” bytes long. You can assume there’s always one input it will always be a positive integer. Run strace with your compiled program.  
   Finally, attach the code of the program and a screenshot of the output of running strace with your compiled program below

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[]) {

int size = atoi(argv[1]);

char\* buffer = (char\*)malloc(size);

if (buffer == NULL) {

printf("Memory allocation failed\n");

return 1;

}

A computer screen shot of a program

Description automatically generated

free(buffer);

return 0;

}

## The output you received from running strace on your program was probably very messy. There’s no way to tell which system call was used during the execution of malloc. Suggest a simple addition to your C code, such that you will be able to spot the system call used during the execution of malloc anyway. You’re not allowed to add flags to strace. Your change must be made in the C code.

## נעטוף את הפונקציה שקריאת מערכת getpid ובכך נוכל לזהות את הקריאה לMALLOC או במה שהיא משתמשת. כעת ניתן לזהות שהיא משתמשת בBRK ( בארבע השורות האחרונות בצילום המסך). הקוד המעודכן:

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[]) {

int size = atoi(argv[1]);

getpid();

char\* buffer = (char\*)malloc(size);

getpid();

if (buffer == NULL) {

printf("Memory allocation failed\n");

return 1;

A computer screen shot of a computer program

Description automatically generated }

free(buffer);

return 0;

}

## Section 2:

In the wet part of this homework, you wrote/will write a malloc() alternative that uses both sbrk() and mmap(). Your job in this section is to determine which memory functions the malloc() function that is included in your stdlib uses.

Hint: Use the program and the tools from the last section to help you out!

1. Which **two** system calls does the stdlib standard malloc() use in its implementation? Attach screenshots that prove your answer.

1) brk(NULL) 2) brk(0x55b44855000)

ניתן לראות זאת בצילום מסך שצורף בדף הקודם.

1. Find the **threshold** that malloc uses to transition from using one function to the other. In other words, what is the number of bytes, after which calling malloc with that number, would result in using one system call instead of the other? Attach screenshots that prove your answer.

\_\_\_\_\_\_\_\_134537\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ניתן לראות שבמעבר מ 134536 ל134537, יש שימוש בMMAP.

A computer screen shot of a computer code

Description automatically generatedA computer screen shot of a computer program

Description automatically generated