Operating Systems – 234123

**Homework Exercise 4 – Dry**

**{ Name: Roni Roitbord , ID: 313575599, email: roniro@campus.technion.ac.il  
Name: Amit Gabay , ID: 206040768, email: amitg@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 | שורה זו משתמשת במנגנון הפתיחה העצלה של קובץ, אין צורך לקרוא מידע מהדיסק. |
| 4 | 0 | שורה זו משתמש במנגנון הקצאה עצלה של זיכרון, אף דף פיזי לא באמת יוקצה עד שיכתבו אליו/יקראו ממנו. |
| 5 | 4 | בשורה זו, לעומת שורה 4, כן יוקצו 2 דפים עבור ה-buffer, מפני שאנו רוצים לבצע כתיבה אליו. כמו כן, יוקצו 2 דפים עבור ה-Page Cache. |
| 6 | 0 | שורה זו משתמש במנגנון הקצאה עצלה של זיכרון, אף דף פיזי לא באמת יוקצה עד שיכתבו אליו/יקראו ממנו. |
| 7 | 0 | בשורה זו "יוקצה" זיכרון וירטואלי בלבד. כלומר, יוקצה דף וירטואלי שיצביע למסגרת המתאימה ב- Page Cache |
| 8 | 1 | בניגוד לשורה 7, בה אנחנו ניגשים לדף, נניח דף x, בשורה זו אנו ניגשים לדף x+2. עד כה, נמצאים במטמון הדפים, כפי שהגדרנו ב-read, 2 דפים בלבד, דף x ודף x+1. כלומר- אנו ניגשים לדף x+2 בפעם הראשונה ורוצים לבצע קריאה ממנו, ולכן תוקצה לו מסגרת ב-Page Cache. |
| 9 | 0 | בשורה 6 הפעלנו את mmap עם MAP\_SHAERD, ולכן השינוי שאנו מבצעים ייכתב ישירות ל- Page Cacheולא יוקצו מסגרות פיזיות חדשות. |

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

**תשובה:** במקרה זה, תשובתנו תשתנה עבור:

* שורה 9: הערך כבר לא היה יכול להיכתב ישירות למטמון הדפים (בגלל השימוש ב- MAP\_PRIVATE), ולכן כן היינו צריכים להקצות מסגרת אחת בשביל האפליקציה.

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

**תשובה:** במקרה זה, תשובתנו תשתנה עבור:

* שורה 5: מכיוון שבהרצה הראשונה כבר הוקצו 2 דפים עבור ה-Page Cache, בהרצה השנייה יוקצו בשורה זו רק 2 דפים עבור ה-buffer (ולא 4).
* שורה 8: בריצה הראשונה אכן הקצינו מסגרת אחת, מפני שזו הפעם הראשונה שיש גישה לדף , אך בריצה השנייה זו כבר לא הפעם הראשונה שניגשים אליו, וכבר יש לו מסגרת ב-Page Cache, ולכן נקצה 0 מסגרות (ולא 1).

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:

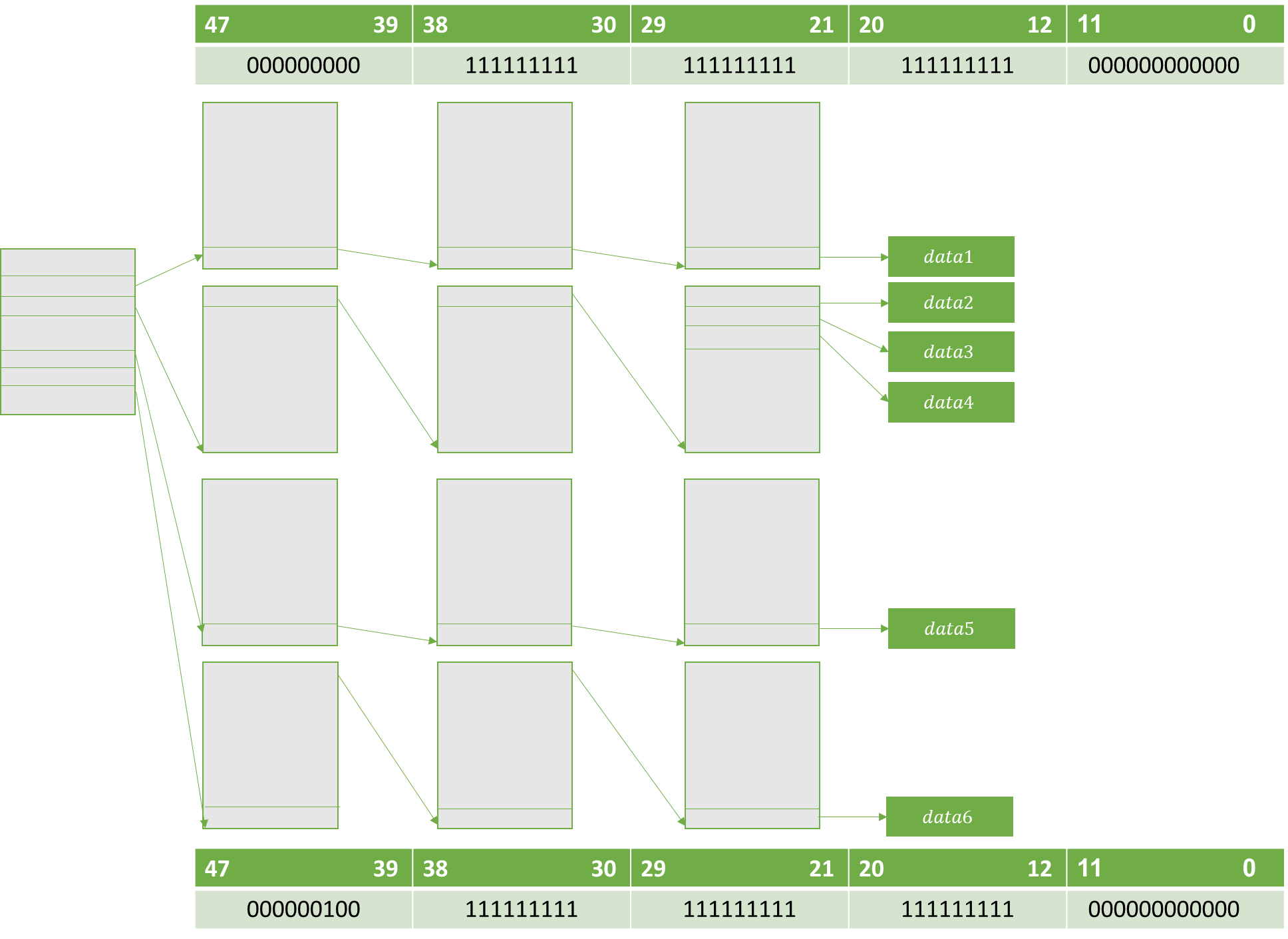
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 0 | 20 12 | 29 21 | 38 30 | 47 39 |
| 000000000000 | 111111111 | 111111111 | 000000**1**00 | 000000000 |

(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:** המספר המקסימלי של מסגרות פיזיות חדשות שמוקצות בעבור טבלת הדפים של התהליך כאשר מורד הקוד לראשונה הוא: 12.

איור:



1. הגדירו major page fault.

**תשובה: Major Page Fault** היא חריגה של Page Fault שנוצרת בהרצה של קוד משתמש שמנסה לגשת לזיכרון, ומסיבה כלשהי גישה זו דורשת שמערכת ההפעלה תפנה לדיסק, כלומר - דורשת יציאה להמתנה.

במקרה כזה, המשתמש מריץ קוד ומצפה לגשת למידע מסוים, והמידע הזה דורש גישה לדיסק, מה שגורם למשתמש לחכות עד שהתהליך יחזור מהמתנה.

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

**תשובה: Major Page Fault** קורה בשורות בהן בסעיף 1 נדרשת הקצאה של מסגרות, שורות 5,8, מפני שבשורות אלה יש צורך בגישה לדיבר לשם שליפת מידע חדש שלא ראינו בעבר.

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

**תשובה:** במקרה זה, בשורה 9 הערך כבר לא היה יכול להיכתב ישירות למטמון הדפים והייתה מוקצית מסגרת נוספת בשביל האפליקציה, ולכן ה-Page fault שיווצר לא ידרוש גישה לדיסק (Page Fault לא חוסם).  
כלומר, עם השינוי הנ"ל יווצר minor page fault ולא major page fault, ולכן התשובה לסעיף 6 תישאר זהה.

**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 allows system calls and signals monitoring.   
By attaching strace to a running process/ executing a program with strace, it records the process’s system calls information (name, arguments, return value), as well as signals received/sent by the process.  
upon running ‘strace ls’ in the terminal, strace would attach itself to the ls command and display a detailed log of system calls made by the ls process and signals received or sent.

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

**Program Code:**

#include <stdio.h>

#include <stdlib.h>

void\* allocate\_memory(size\_t size) {

return malloc(size);

}

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

if (argc != 2) {

return 1;

}

int x = atoi(argv[1]);

void\* memory\_block = allocate\_memory(x);

return 0;

}

**Output:**

****

## 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.

## We Can Add printing before using malloc an after using malloc so we can find those custom prints later in the output:

**Program Code:**

#include <stdio.h>

#include <stdlib.h>

void\* allocate\_memory(size\_t size) {

return malloc(size);

}

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

if (argc != 2) {

return 1;

}

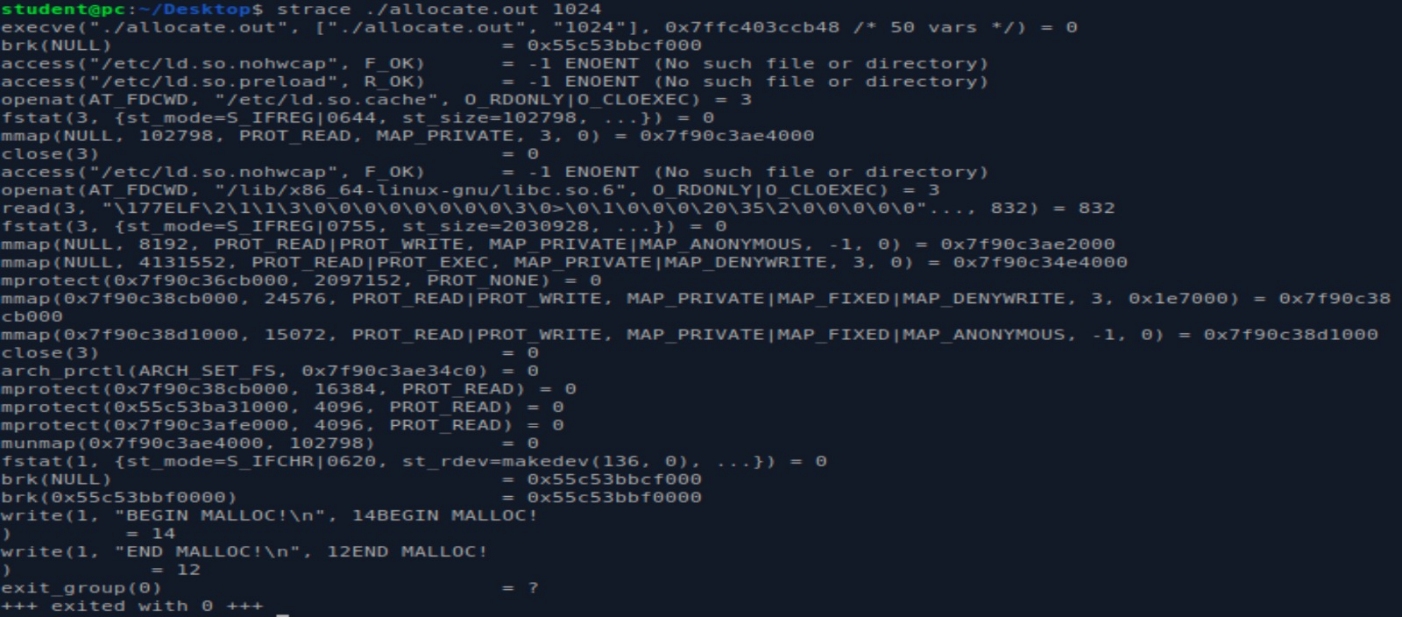
int x = atoi(argv[1]);  
 printf("BEGIN MALLOC!\n");

void\* memory\_block = allocate\_memory(x);  
 printf("END MALLOC!\n");

return 0;

}

**Output:**



## Section 2:

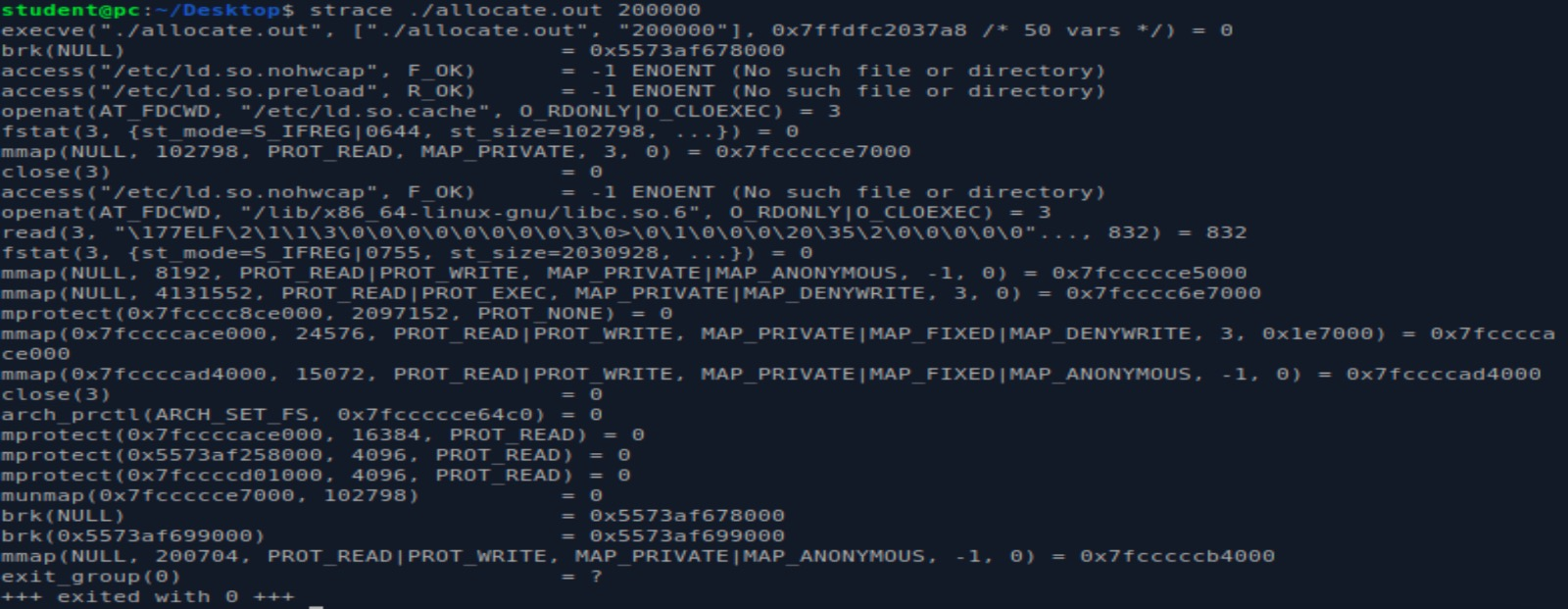
In the wet part of this homework, you wrote/will write a malloc() alternative that uses both sbrk() and mmap(). Your job in this section is to determine which memory functions the malloc() function that is included in your stdlib uses.

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() 2) mmap()

**Output:**

****

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.

answer: 134537.  
No use of mmap when allocating 134536 bytes – usage of brk only.  
Once we allocated one more byte, the mmap() is being used for allocation, instead of brk.

**Output:**

