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:

```
1  #define PAGE_SIZE (4*1024)

2  int main()
   {
3      int fd = open("./my_file.txt", O_RDWR);

4      char* buffer = malloc(2 * PAGE_SIZE);

5      read(fd, buffer, 2 * PAGE_SIZE);

6      char* array = (char*) mmap(NULL, 4 * PAGE_SIZE, PROT_READ |
       PROT_WRITE, MAP_SHARED, fd, 0);
7      char x = array[12];

8      char z = array[12 + 2 * PAGE_SIZE];

9      array[12 + 2 * PAGE_SIZE] = x;

10     return 0;

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 Cache ולא 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:

<u>תשובה:</u> 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.)

<u>תשובה:</u> 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 $x + 2$, 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:

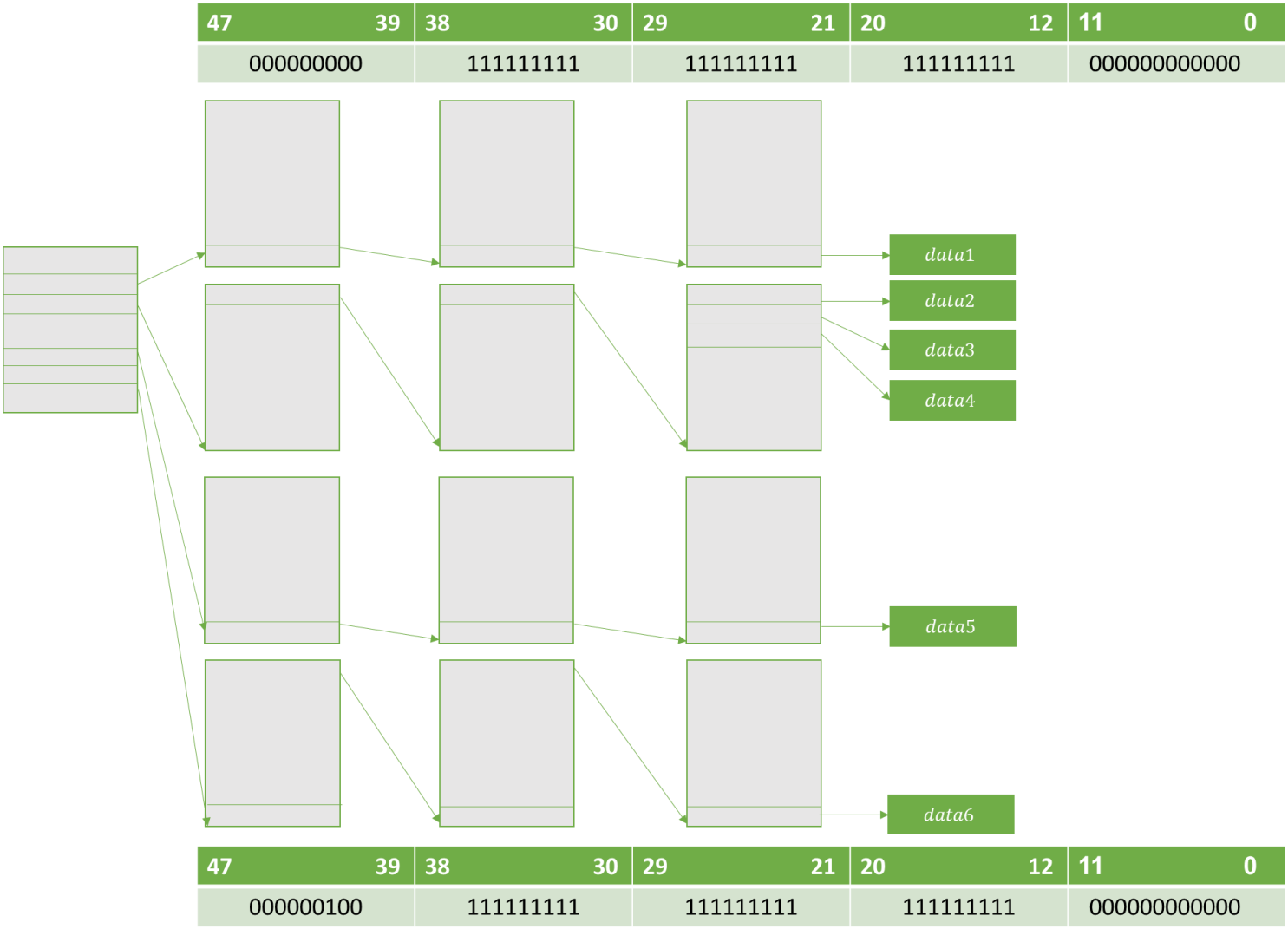
47 39	38 30	29 21	20 12	11 0
000000000	00000**1**00	111111111	111111111	000000000000

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

47 39	38 30	29 21	20 12	11 0
000000000	00000**0**00	111111111	111111111	000000000000

<u>תשובה:</u> המספר המקסימלי של מסגרות פיזיות חדשות שמוקצות בעבור טבלת הדפים של התהליך כאשר מורד הקוד לראשונה הוא: 12.

איור:


47 39
38 30
29 21
20 12
11 0
000000000
111111111
111111111
111111111
000000000000
data1
data2
data3
data4
data5
data6
47 39
38 30
29 21
20 12
11 0
000000100
111111111
111111111
111111111
000000000000

5. הגדירו major page fault.

<u>תשובה:</u> 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.

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

<u>תשובה:</u> 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.

7. כיצד תשתנה תשובתכן לסעיף **6** אם בשורה 6 היה רשום MAP_PRIVATE במקום MAP_SHARED? נמקו

<u>תשובה:</u> 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.

2. 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:

```
student@pc:~/Desktop$ strace ./allocate.out 1024
execve("./allocate.out", ["./allocate.out", "1024"], 0x7ffc5216b7f8 /* 50 vars */) = 0
brk(NULL)                               = 0x5649ddd77000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=102798, ...}) = 0
mmap(NULL, 102798, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fcc71e28000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fcc71e26000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcc71828000
mprotect(0x7fcc71a0f000, 2097152, PROT_NONE) = 0
mmap(0x7fcc71c0f000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fcc71c
0f000
mmap(0x7fcc71c15000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fcc71c15000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fcc71e274c0) = 0
mprotect(0x7fcc71c0f000, 16384, PROT_READ) = 0
mprotect(0x5649dc457000, 4096, PROT_READ) = 0
mprotect(0x7fcc71e42000, 4096, PROT_READ) = 0
munmap(0x7fcc71e28000, 102798)          = 0
brk(NULL)                               = 0x5649ddd77000
brk(0x5649ddd98000)                     = 0x5649ddd98000
exit_group(0)                           = ?
+++ exited with 0 +++
```

3. 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.

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

   Program Code:

```c
#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:

```
student@pc:~/Desktop$ strace ./allocate.out 1024
execve("./allocate.out", ["./allocate.out", "1024"], 0x7ffc403ccb48 /* 50 vars */) = 0
brk(NULL)                               = 0x55c53bbcf000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=102798, ...}) = 0
mmap(NULL, 102798, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f90c3ae4000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f90c3ae2000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f90c34e4000
mprotect(0x7f90c36cb000, 2097152, PROT_NONE) = 0
mmap(0x7f90c38cb000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f90c38
cb000
mmap(0x7f90c38d1000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f90c38d1000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f90c3ae34c0) = 0
mprotect(0x7f90c38cb000, 16384, PROT_READ) = 0
mprotect(0x55c53ba31000, 4096, PROT_READ) = 0
mprotect(0x7f90c3afe000, 4096, PROT_READ) = 0
munmap(0x7f90c3ae4000, 102798)          = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
brk(NULL)                               = 0x55c53bbcf000
brk(0x55c53bbf0000)                     = 0x55c53bbf0000
write(1, "BEGIN MALLOC!\n", 14BEGIN MALLOC!
)       = 14
write(1, "END MALLOC!\n", 12END MALLOC!
)          = 12
exit_group(0)                           = ?
+++ exited with 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() 2) mmap()

Output:

```
student@pc:~/Desktop$ strace ./allocate.out 200000
execve("./allocate.out", ["./allocate.out", "200000"], 0x7ffdfc2037a8 /* 50 vars */) = 0
brk(NULL)                               = 0x5573af678000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=102798, ...}) = 0
mmap(NULL, 102798, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fcccce7000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fccccce5000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcccc6e7000
mprotect(0x7fcccc8ce000, 2097152, PROT_NONE) = 0
mmap(0x7fccccace000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fcccca
ce000
mmap(0x7fccccad4000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fccccad4000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fcccce64c0) = 0
mprotect(0x7fccccace000, 16384, PROT_READ) = 0
mprotect(0x5573af258000, 4096, PROT_READ) = 0
mprotect(0x7fccccd01000, 4096, PROT_READ) = 0
munmap(0x7fcccce7000, 102798)           = 0
brk(NULL)                               = 0x5573af678000
brk(0x5573af699000)                     = 0x5573af699000
mmap(NULL, 200704, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fcccccb4000
exit_group(0)                           = ?
+++ exited with 0 +++
```

2. 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:

```
student@pc:~/Desktop$ strace ./allocate.out 134536
execve("./allocate.out", ["./allocate.out", "134536"], 0x7fff3ae33888 /* 50 vars */) = 0
brk(NULL)                               = 0x55913f538000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=102798, ...}) = 0
mmap(NULL, 102798, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fa2df7d5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa2df7d3000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fa2df1d5000
mprotect(0x7fa2df3bc000, 2097152, PROT_NONE) = 0
mmap(0x7fa2df5bc000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fa2df5
bc000
mmap(0x7fa2df5c2000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fa2df5c2000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fa2df7d44c0) = 0
mprotect(0x7fa2df5bc000, 16384, PROT_READ) = 0
mprotect(0x55913ead7000, 4096, PROT_READ) = 0
mprotect(0x7fa2df7ef000, 4096, PROT_READ) = 0
munmap(0x7fa2df7d5000, 102798)          = 0
brk(NULL)                               = 0x55913f538000
brk(0x55913f559000)                     = 0x55913f559000
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
student@pc:~/Desktop$ strace ./allocate.out 134537
execve("./allocate.out", ["./allocate.out", "134537"], 0x7fff2bdfe928 /* 50 vars */) = 0
brk(NULL)                               = 0x55a109217000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=102798, ...}) = 0
mmap(NULL, 102798, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdbe9d66000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdbe9d64000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdbe9766000
mprotect(0x7fdbe994d000, 2097152, PROT_NONE) = 0
mmap(0x7fdbe9b4d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fdbe9b
4d000
mmap(0x7fdbe9b53000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdbe9b53000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fdbe9d654c0) = 0
mprotect(0x7fdbe9b4d000, 16384, PROT_READ) = 0
mprotect(0x55a107ad0000, 4096, PROT_READ) = 0
mprotect(0x7fdbe9d80000, 4096, PROT_READ) = 0
munmap(0x7fdbe9d66000, 102798)          = 0
brk(NULL)                               = 0x55a109217000
brk(0x55a109238000)                     = 0x55a109238000
mmap(NULL, 135168, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdbe9d43000
exit_group(0)                           = ?
+++ exited with 0 +++
```