Operating Systems – 234123

Homework Exercise 2 – Dry

Teaching Assistant in charge:

Sami Zriek

Assignment Subjects & Relevant Course material

Virtual Memory

Lectures 4-5, Recitations 4-5

Submission Format

1. Only **typed** submissions in **PDF** format will be accepted. Scanned handwritten submissions will not be graded.
2. The dry part submission must contain a single PDF file named with your student IDs –

   DHW2_123456789_300200100.pdf
3. The submission should contain the following:
   a. The first page should contain the details about the submitters - Name, ID number and email address.
   b. Your answers to the dry part questions.
4. Submission is done electronically via the course website, in the **HW2 - Dry** submission box.

Grading

1. **All** question answers must be supplied with a **full explanation**. Most of the weight of your grade sits on your **explanation** and **evident effort**, and not on the absolute correctness of your answer.
2. Remember – your goal is to communicate. Full credit will be given only to correct solutions which are **clearly** described. Convoluted and obtuse descriptions will receive low marks.

Questions & Answers

- The Q&A for the exercise will take place at a public forum Piazza **only**. Please **DO NOT** send questions to the private email addresses of the TAs.
- Critical updates about the HW will be published in **pinned** notes in the piazza forum. These notes are mandatory and it is your responsibility to be updated.

A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
- Be polite, remember that course staff does this as a service for the students
- You're not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
- When posting questions regarding hw2, put them in the hw2 folder

Late Days

- Please **DO NOT** send postponement requests to the TA responsible for this assignment. Only the **TA in charge** can authorize postponements. In case you need a postponement, please fill out the attached form :https://forms.gle/EJ9qVM246e2fvDua7

שאלה 1

עדן, זמרת פופולרית, סבלה ממחסור בזיכרון פיזי במחשב שלה (בעל מעבד אינטל 32-ביט וזיכרון פיזי בנפח 4GB) ולכן הציעה תכן חדש של מעבד המרחיב את מרחב הזיכרון הפיזי מ-32 ל-40 ביט. כתוצאה מכך, במימוש של עדן יש שלוש רמות תרגום בטבלת הדפים.
שאר נתוני המעבד של עדן זהים לאלו של מעבד IA-32, כלומר נתוני המערכת החדשה הם:

רוחב כתובת וירטואלית	32bit
רוחב כתובת פיזית	40bit
גודל דף/מסגרת/מגירה	4KB
גודל מסגרת של טבלת דפים (בכל הרמות)	4KB
מספר ביטים לדגלים והרשאות בכל כניסה בטבלת הדפים	12bit

1. (5 נק') בהנחה שגודל כניסה בטבלת הדפים **מעוגל למעלה לחזקה שלמה של 2**, מהו אופן חלוקת הכתובת הוירטואלית לשדות בתהליך תרגום כתובות (page walk)?

	offset	index1	index2	index3
א.	12	9	9	2
ב.	10	10	10	2
ג.	12	10	9	1
ד.	10	9	9	4
ה.	12	10	10	2
ו.	10	9	9	2

<u>נימוק:</u>

מאחר והגדלנו את מרחב הזיכרון הפיזי ב8 ביטים,ביטים אלו יתווספו לביטים העליונים במסגרת של כתובת פיזית,(12 הביטים התחתונים נשארים עבור מיפוי המידע שאנחנו צריכים בדף בגודל 4K),ומאחר שגודל כניסה בטבלת הדפים מעוגל למעלה לחזקה שלמה של 2 , נקבל מבנה חדש עבור כניסה בטבלת הדפים בגודל 64 ביט:

offset	frame number	spare (not used)
12 bits	28 bits	24 bits

גודל מסגרת פיזית הוא 4KB ולכן נצטרך 4KB/8B=512 כניסות בכל מסגרת בטבלת הדפים.

עבור 512 כניסות נזדקק ל9 ביטים כדי למפות כל כניסה בטבלה הדפים ברמה התחתונה, ובאותו אופן (מכיוון שגודל מסגרת טבלת דפים היא גם כן 4KB) גם עבור האינדקס מעליו נזדד ל9 ביטים כדי למפות כל כניסה, ובאינדקס השלישי נשתמש בביטים שנותרו, ונקבל את המבנה הבא עבור כתובת וירטואלית:

index #3	index #2	index #1	offset
2 bits	9 bits	9 bits	12 bits

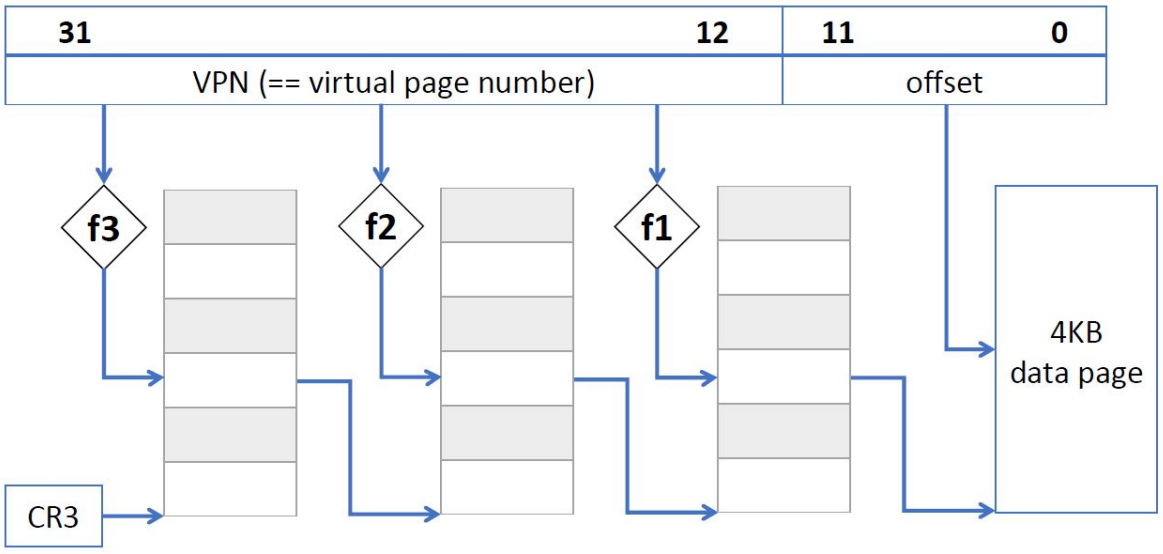
לבעלה של עדן, שוקי, אין שום תואר מהטכניון, ולמרות זאת הוא הבחין כי המימוש של עדן בזבזני בגלל שגודל הכניסות בטבלת הדפים מעוגל למעלה לחזקה שלמה של 2.

2. (5 נק') מהו הגודל המינימלי האפשרי של כניסה בטבלת הדפים אם **לא מעגלים אותו למעלה**?

a. 3 בתים
b. 4 בתים
c. 5 בתים
d. 6 בתים
e. 7 בתים
f. אף תשובה אינה נכונה

<u>נימוק:</u>
מצאנו בסעיף הקודם כי עבור כל כניסה בטבלת הדפים נזדקק ל40 ביטים, 12 עבור הOFFSET ו28 עבור מספר המסגרת הפיזית. ולכן 40BIT/8=5 BYTES.

בהמשך לסעיף הקודם, שוקי (בעלה של עדן) הציע מימוש חדש לטבלת הדפים שבו כל כניסה בטבלת הדפים (בכל הרמות) היא בגודל המינימלי מהסעיף הקודם. במימוש של שוקי, כמו במימוש המקורי של טבלת הדפים במעבדי אינטל, דפים סמוכים בזיכרון הוירטואלי נשמרים בכניסות סמוכות בטבלת הדפים.
שוקי הבחין שבמימוש החדש **הכתובת הוירטואלית אינה מתפרקת לשדות של אינדקסים** ויש צורך בחישובים מורכבים על מנת למצוא את האינדקס המתאים בכל טבלה (כלומר בכל רמה בעץ). להלן שרטוט הממחיש את אופן התרגום:



בשרטוט רואים שלוש פונקציות (f1, f2, f3) המקבלות את מספר הדף הוירטואלי VPN ומחזירות, בהתאמה, שלושה אינדקסים לשלושת הרמות בטבלת הדפים.
בכל הסעיפים הבאים, הפעולות חלוקה / ומודולו % הן **פעולות בשלמים**. למשל:
1=1024/819
205=1024%819

3. (5 נק') מהי הפונקציה f1?

a. $f1(vpn) = vpn/819$
b. $f1(vpn) = vpn\%819$
c. $f1(vpn) = (vpn/819)\%819$
d. $f1(vpn) = (vpn/819)/819$
e. $f1(vpn) = (vpn\%819)/819$
f. $f1(vpn) = ((vpn/819)\%819)/819$

<u>נימוק:</u>
מאחר וכל כניסה בטבלת הדפים היא בגודל 5 בתים, וגודל מסגרת הוא 4KB,ישנם 4KB/5B=819 כניסות בטבלת הדפים.
בארכיטקטורת IA-32 כל כניסה בטבלת הדפים היא בגודל 4 בתים, ולכן ישנם 1024 כניסות.
10 הביטים התחתונים בארכיטקטורת IA-32 , מייצגים את מספר הדף ברמה התחתונה, כלומר VPN%1024 מייצג את מספר הדף **בארכיטקטורת IA-32**. מאחר וגודל כניסה בטבלת הדפים אצלנו אינה חזקה של 2, לא ניתן להתייחס למספר ביטים תחתונים אלה לבצע את פעולת המודולו, לכן נסיק מכך ש $f1(vpn) = vpn\%819$.

4. (3 נק') מהי הפונקציה f2?

a. $f2(vpn) = vpn/819$
b. $f2(vpn) = vpn\%819$
c. $f2(vpn) = (vpn/819)\%819$
d. $f2(vpn) = (vpn/819)/819$
e. $f2(vpn) = (vpn\%819)/819$
f. $f2(vpn) = ((vpn/819)\%819)/819$

<u>נימוק:</u>
בדומה לתשובה הקודמת, בארכיטקטורת IA-32, אינדקס הרמה העליונה הוא 10 הביטים העליונים, שאותם ניתן להסיק ע"י vpn/1024.
מאחר ובמקרה שלנו גודל כניסה בטבלת הדפים(5 בתים) אינה חזקה שלמה של 2,ומאחר וישנם 3 רמות לא ניתן לשלוף את אינדקס הרמה השנייה ע"י חלוקה במספר הכניסות, אחרת נקבל שרשור של ייצוג הרמה העליונה בנוסף לרמה זו.
על כן נבצע פעולת חלוקה ולאחר מכן מודולו ובכך נבטיח שנקבל את ייצוג הרמה האמצעית, כלומר:
$f2(vpn) = (vpn/819)\%819$.

5. (2 נק') מהי הפונקציה f3?

a. $f3(vpn) = vpn/819$
b. $f3(vpn) = vpn\%819$
c. $f3(vpn) = (vpn/819)\%819$
d. $f3(vpn) = (vpn/819)/819$
e. $f3(vpn) = (vpn\%819)/819$
f. $f3(vpn) = ((vpn/819)\%819)/819$

<u>נימוק:</u>
בדומה לסעיפים הקודמים , נרצה לשלוף את הייצוג של הרמה העליונה ולכן נבצע חלוקה ב-819 פעמיים. אין צורך לבצע פעולת מודולו מכיוון שידוע כי יש 3 רמות ולכן מה שנשאר לאחר החלוקה זהו הייצוג של הרמה העליונה.
$f3(vpn) = (vpn/819)/819$.

6. (5 נק') מה היתרון של המערכת שהציע שוקי על פני המערכת שהציעה עדן?

a. מיפוי של מרחב זיכרון וירטואלי גדול יותר.
b. מיפוי של מרחב זיכרון פיזי גדול יותר.
c. ה-TLB אפקטיבי יותר בגלל שהוא מכסה יותר זיכרון.
d. טבלאות הדפים של תהליכי משתמש תופסות נפח קטן יותר בזיכרון.
e. פחות פרגמנטציה חיצונית, כלומר יותר זיכרון רציף.
f. אף תשובה אינה נכונה.

נימוק:
טבלת הדפים אצל שוקי קטנה יותר, מכיוון שבכל מסגרת בטבלת הדפים יש 819 כניסות במקום 512.
לכן במידה ומשתמשים בכל מרחב הזיכרון, מספר המסגרות ברמה התחתונה הוא 1281 לעומת 2048.

שאלה 2

שאלה זו כתובה באנגלית מאחר ושהיא מכילה לא מעט מושגים ושמות אשר קל יותר לבטאם באנגלית. פתרו סעיף זה באיזה שפה שתרצו.

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

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 it does.

strace serves as a light weight debugger, it allows a programmer to see how a program is interacting with the OS(kernel), and does this by monitoring system calls and signals.

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.

```
#include <stdlib.h>

int main( int argc, char **argv ) {
   void* ptr=malloc(atoi(argv[1])* sizeof(char));
return 0;
}
```

```
student@ubuntu18:~/CLionProjects/HW2_dry/cmake-build-debug$ strace ./HW2_dry 5
execve("./HW2_dry", ["./HW2_dry", "5"], 0x7ffdf06cc108 /* 51 vars */) = 0
brk(NULL)                               = 0x5634b9bb5000
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=95548, ...}) = 0
mmap(NULL, 95548, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7d7867a000
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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7d78678000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7d7807a000
mprotect(0x7f7d78261000, 2097152, PROT_NONE) = 0
mmap(0x7f7d78461000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f7d78461000
mmap(0x7f7d78467000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7d78467000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f7d786794c0) = 0
mprotect(0x7f7d78461000, 16384, PROT_READ) = 0
mprotect(0x5634b9807000, 4096, PROT_READ) = 0
mprotect(0x7f7d78692000, 4096, PROT_READ) = 0
munmap(0x7f7d7867a000, 95548)           = 0
brk(NULL)                               = 0x5634b9bb5000
brk(0x5634b9bd6000)                     = 0x5634b9bd6000
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.

Finally, **attach the code of the updated program here**:

```
#include <stdlib.h>
#include <unistd.h>

int main( int argc, char **argv ) {
   usleep(1000000);
   void* ptr=malloc(atoi(argv[1])* sizeof(char));
return 0;
}
```

4. 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 related functions/system calls the **original** malloc() function that is included in stdlib uses.
   Hint: Use the program and the tools from the last sections to help you out!
   Which two system calls does the stdlib standard malloc() use in its implementation?
   Attach **screenshots** that prove your answer.

<u>Answer</u>: malloc uses both mmap and sbrk system calls.

With a 5 bytes input(uses only brk):

```
student@ubuntu18:~/CLionProjects/HW2_dry/cmake-build-debug$ strace ./HW2_dry 5
execve("./HW2_dry", ["./HW2_dry", "5"], 0x7ffcd11d4668 /* 50 vars */) = 0
brk(NULL)                               = 0x5606d3c57000
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=95548, ...}) = 0
mmap(NULL, 95548, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f34ff8fe000
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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f34ff8fc000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f34ff2fe000
mprotect(0x7f34ff4e5000, 2097152, PROT_NONE) = 0
mmap(0x7f34ff6e5000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f34ff6e5000
mmap(0x7f34ff6eb000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f34ff6eb000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f34ff8fd4c0) = 0
mprotect(0x7f34ff6e5000, 16384, PROT_READ) = 0
mprotect(0x5606d271b000, 4096, PROT_READ) = 0
mprotect(0x7f34ff916000, 4096, PROT_READ) = 0
munmap(0x7f34ff8fe000, 95548)           = 0
nanosleep({tv_sec=1, tv_nsec=0}, NULL)  = 0
brk(NULL)                               = 0x5606d3c57000
brk(0x5606d3c78000)                     = 0x5606d3c78000
exit_group(0)                           = ?
+++ exited with 0 +++
```

With a 50000000000000000 bytes input(uses mmap):

```
student@ubuntu18:~/CLionProjects/HW2_dry/cmake-build-debug$ strace ./HW2_dry 50000000000000000
execve("./HW2_dry", ["./HW2_dry", "50000000000000000"], 0x7ffc661a00d8 /* 50 vars */) = 0
brk(NULL)                               = 0x5568a669d000
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=95548, ...}) = 0
mmap(NULL, 95548, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f1ea5889000
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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f1ea5887000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f1ea5289000
mprotect(0x7f1ea5470000, 2097152, PROT_NONE) = 0
mmap(0x7f1ea5670000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f1ea5670000
mmap(0x7f1ea5676000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1ea5676000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f1ea58884c0) = 0
mprotect(0x7f1ea5670000, 16384, PROT_READ) = 0
mprotect(0x5568a54cc000, 4096, PROT_READ) = 0
mprotect(0x7f1ea58a1000, 4096, PROT_READ) = 0
munmap(0x7f1ea5889000, 95548)           = 0
nanosleep({tv_sec=1, tv_nsec=0}, NULL)  = 0
brk(NULL)                               = 0x5568a669d000
brk(0x5568a66be000)                     = 0x5568a66be000
mmap(NULL, 784666624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f1e76638000
exit_group(0)                           = ?
+++ exited with 0 +++
```

5. **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.

<u>Answer:</u> the threshold is **134536**.
when we try to allocate 134536 bytes we use sbrk syscall:

```
student@ubuntu18:~/CLionProjects/HW2_dry/cmake-build-debug$ strace ./HW2_dry 134536
execve("./HW2_dry", ["./HW2_dry", "134536"], 0x7ffc35685678 /* 50 vars */) = 0
brk(NULL)                               = 0x55805d923000
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=95548, ...}) = 0
mmap(NULL, 95548, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f809e123000
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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f809e121000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f809db23000
mprotect(0x7f809dd0a000, 2097152, PROT_NONE) = 0
mmap(0x7f809df0a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f809df0a000
mmap(0x7f809df10000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f809df10000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f809e1224c0) = 0
mprotect(0x7f809df0a000, 16384, PROT_READ) = 0
mprotect(0x55805bccc000, 4096, PROT_READ) = 0
mprotect(0x7f809e13b000, 4096, PROT_READ) = 0
munmap(0x7f809e123000, 95548)           = 0
nanosleep({tv_sec=1, tv_nsec=0}, NULL)  = 0
brk(NULL)                               = 0x55805d923000
brk(0x55805d944000)                     = 0x55805d944000
exit_group(0)                           = ?
+++ exited with 0 +++
```

And when we try to allocate 134537 we use mmap syscall:

```
student@ubuntu18:~/CLionProjects/HW2_dry/cmake-build-debug$ strace ./HW2_dry 134537
execve("./HW2_dry", ["./HW2_dry", "134537"], 0x7ffc2d7444b8 /* 50 vars */) = 0
brk(NULL)                               = 0x55ecb48f6000
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=95548, ...}) = 0
mmap(NULL, 95548, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd37bb96000
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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd37bb94000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd37b596000
mprotect(0x7fd37b77d000, 2097152, PROT_NONE) = 0
mmap(0x7fd37b97d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fd37b97d000
mmap(0x7fd37b983000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd37b983000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fd37bb954c0) = 0
mprotect(0x7fd37b97d000, 16384, PROT_READ) = 0
mprotect(0x55ecb3e2c000, 4096, PROT_READ) = 0
mprotect(0x7fd37bbae000, 4096, PROT_READ) = 0
munmap(0x7fd37bb96000, 95548)           = 0
nanosleep({tv_sec=1, tv_nsec=0}, NULL)  = 0
brk(NULL)                               = 0x55ecb48f6000
brk(0x55ecb4917000)                     = 0x55ecb4917000
mmap(NULL, 135168, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd37bb73000
exit_group(0)                           = ?
+++ exited with 0 +++
```