OS Lab 7

Status	approved
checkbox	☑
class	OS
due date	@Mar 17, 2021

Task

7. Таблица поиска строк в текстовом файле 2.

📢 Измените предыдущую программу так, чтобы использовалось отображение файла в память взамен использования `read(2)`, `lseek(2)` и `write(2)`.

Notes

```
#include <sys/types.h>
#include <string.h>
#include <stdbool.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>

#define LINECOUNT 256

void buildFileMap(off_t* offsets, off_t* lengths, const char* mapp
edFile, int fileSize, size_t* lineCount){
    char buffer[BUFSIZ];
    off_t offset = 0;
    off_t lineIdx = 1;

    for (off_t i = 0; i < fileSize; ++i){
        lengths[lineIdx]++;
        offset++;
        if (*(mappedFile + i) == '\n'){
            offsets[lineIdx] = offset - lengths[lineIdx];
            lineIdx++;
        }
    }

    *lineCount = lineIdx;
}

int main(int argc, char* argv[]){
    int fileDescriptor = 0;
    if (argc < 2) {
        perror("Usage: filename as argument\n");
        return 0;
    }

    if((fileDescriptor = open(argv[1], O_RDONLY)) == -1) {
        perror("Input file doesn`t exist\n");
        return 0;
    }

    size_t fileSize = lseek(fileDescriptor, 0, SEEK_END);

    char* mappedFile = mmap(NULL, fileSize, PROT_READ, MAP_PRIVAT
E, fileDescriptor, 0);
    if (mappedFile == MAP_FAILED){
  perror("mmap error:");
  return 1;
    }

    off_t lengths[LINECOUNT] = {0};
```

mmap (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/mman.h>
caddr_t mmap( caddr_t addr,
  size_t len, int prot,
  int flags, int fd, off_t off);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - адрес

неуспех - NULL и errno установлена

```
    off_t offsets[LINECOUNT] = {0};

    size_t lineCount = 0;

    buildFileMap(offsets, lengths, mappedFile, fileSize,  &lineCou
nt);

    fd_set descriptorSet;
    FD_ZERO(&descriptorSet);
    FD_SET(STDIN_FILENO, &descriptorSet);

    struct timeval timeout;
    timeout.tv_sec = 5;
    timeout.tv_usec = 0;

    printf("Enter line number from 1 to %d. Enter 0 to exit. You o
nly got 5 seconds\n", lineCount - 1);

    size_t lineToPrint = 1;
    int selectRetValue;

    while (true){
        char buffer[BUFSIZ] = {0};

        if (lineToPrint == 0){
            break;
        }

        selectRetValue = select(1, &descriptorSet, NULL, NULL, &ti
meout);


        if (selectRetValue == -1){
            perror("select(3C) failed");
            return -1;
        }

        if (selectRetValue == 0){
            for (off_t i = 0; i < fileSize; ++i){
                printf("%c", *(mappedFile + i));
            }
            printf("\n");
            return 0;
        }

        if (scanf("%u", &lineToPrint) != 1){
            fflush(stdin);
            perror("Invalid input");
            continue;
        }

        if (lineToPrint < 0 || lineToPrint > lineCount - 1){
            printf("Line number is an integer number from 1 to %d
\n", lineCount - 1);
            continue;
        }

        for (off_t i = offsets[lineToPrint]; i < offsets[lineToPri
nt+1]; ++i){
            printf("%c", *(mappedFile + i));
        }

    }
    munmap(mappedFile, fileSize);
    close(fileDescriptor);
    return 0;
}
```

💡 **Первый параметр** — адрес на который хотим сделать отображение (очевидно для отображения он должен быть выравнен на начало страницы и этот адрес не должен пересекаться с другмими mmap'нутыми сегментами (если они накладываются то старый маппинг просто затрется и новый вступит в действие)). Если передать NULL — система выберет адрес для маппинга

! Память выделять не нужно

Второй параметр — длина участка который хотим замапать в байтах (! использовать size_t поскольку он отображает размер адресного пространства на данной машине)

Пятый — дескриптор открытого файла который хотим мапать

Шестой — смещение в файле с которого начинаем мапать.

Возвращает адрес начала того сегмента на который оно замапалось

Флаги:

- **MAP_SHARED** — разделяемые изменения. Те измения которые вы вносите в замапанную память, сохраняются в файле и они будет видны другим процессам которые замапали тот же файл

- **MAP_PRIVATE** — начальное состояние мапнутой памяти будет точно таким же как в файле. потом, если вы вносите изменения, они останутся видны только у вашего процесса

- **MAP_ANON** — способ попросить у ядра память заполненную нулями. /dev/zero — псевдофайл бесконечной длины из которого всегда читаются нули

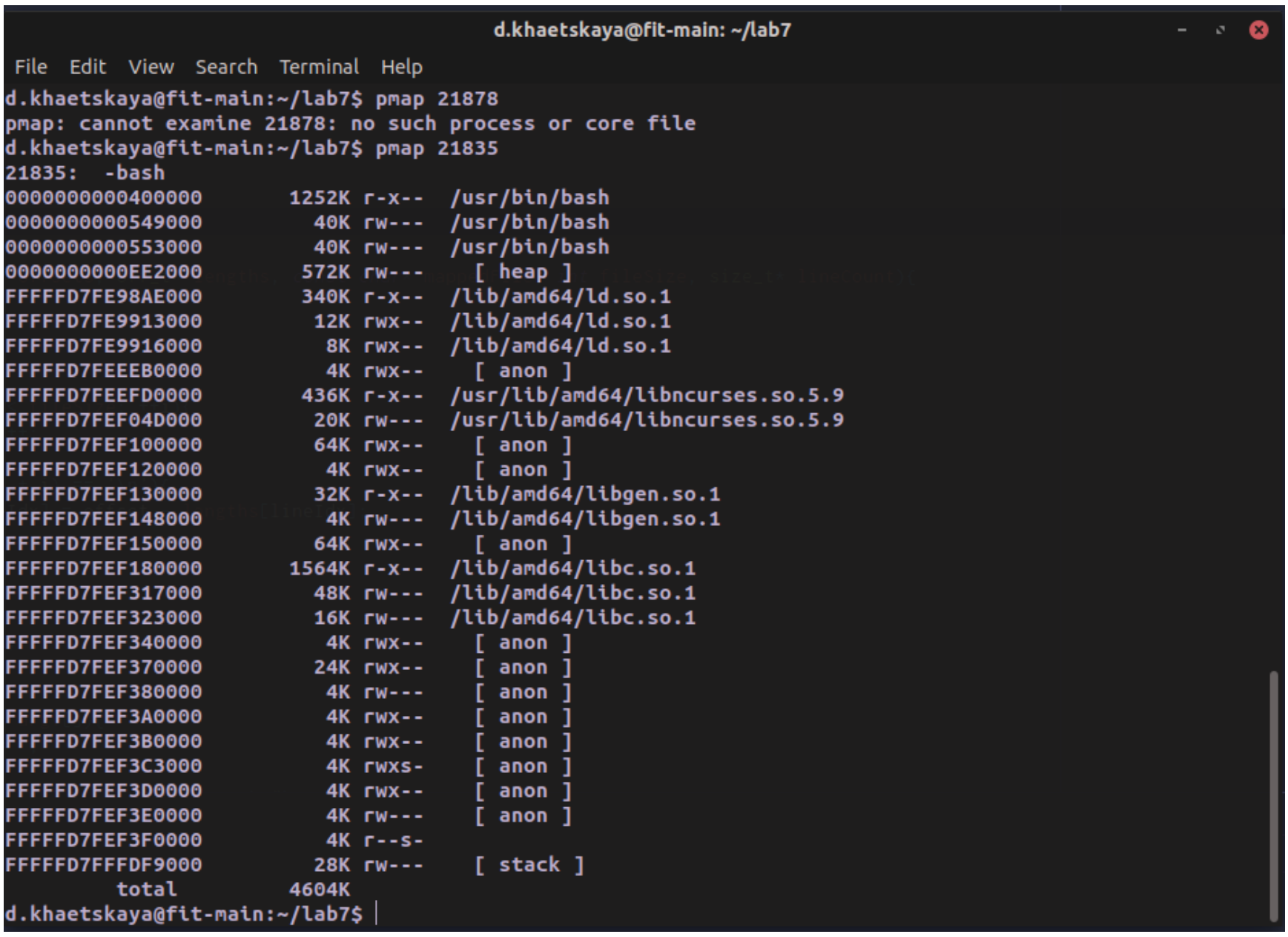
mmap - параметры

prot	PROT_READ можно читать PROT_WRITE можно изменять PROT_EXEC можно исполнять
flags	MAP_SHARED разделяемые изменения MAP_PRIVATE частные изменения MAP_ANON эквивалент mmap /dev/zero

📢 Для пользователя, при вызове mmap всё выглядит так, как будто кусок данных из файла просто появился в памяти. На самом деле, в момент mmap вам отматывают эти странички, ставят на них бит отсутствия, при первом обращении, диспетчер памяти кидает исключение и тогда ядро находит физическую страницу и подкачивают туда данные из файла. (Ленивое чтение, происходит только при первом обращении)

Системный вызов munmap(2) удаляет отображение страниц в диапазоне [addr, addr+len-1]. Последующее использование этих страниц выразится в посылке процессу сигнала SIGSEGV. Границы освобождаемого сегмента не обязаны совпадать с границами ранее отображенного сегмента, но надо иметь в виду, что munmap(2) выравнивает границы освобождаемого сегмента на границы страниц.

Также, неявное удаление отображения для всех сегментов памяти процесса происходит при завершении процесса и при вызове exec(2).



Reading list

- [ ]