Московский авиационный институт (национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Операционные системы»

Студентка:	А. Довженко
Преподаватель:	Е. С. Миронов
Группа:	08-207
Вариант:	13
Дата:	
Оценка:	
Подпись:	

Москва, 2017

Лабораторная работа №4

Изучить технологию отображения файлов в выбранной ОС и работу с основными функциями файловой системы. Составить программу на языке Си, обрабатывающую текстовые файлы – текстовый процессор. При обработке использовать стандартные средства управления файлами в конкретной ОС. Для сдачи лабораторной работы на отлично необходимо обязательное использование технологии отображения файлов в память. Подгружать файл целиком в оперативную память нельзя. Необходимо предусмотреть следующие ключи текстового процессора:
1. Максимальный размер оперативной памяти, которая используется программой.
2. Варианты использования программы (помощь при работе с программой).

Программа должна уметь работать в 2-ух режимах:
1. Интерактивный режим (в одном процессе программы может быть выполнено несколько действий по обработке файла)
1.1. Необходима функция выхода из программы.
1.2. Функция смены файла, не закрывая приложение текстового процессора.
1.3. Вывод состояния файла (количество строк и символов в файле).
2. В режиме одной команды. Должны быть доступны все те же функции, что и в интерактивном режиме.

Вариант 13.
Вывод на экран: Прочитать конкретную строку из файла.
Поиск в файле: Поиск по подстроке без спецсимволов. Необходимо реализовать возможность выбора режима поиска только по началу слова, по суффиксу слова и по любой части слова.
Редкатирование файла: Замена фрагмента строки на другую часть строки (возможно пустую).
Обеспечение безопасности работы с файлами: Проверка того, что конкретный файл не открыт в нескольких процессах программы.

1 Описание

Организация доступных функций в программе:
1. Прочитать конкретную строку из файла. Проходим по файлу, считая количество знаков переноса строки. Когда это число становится равным числу, запрашиваемому пользователем, то выводим знаки этой строки на стандартный поток вывода. Если счетчик строк за проход по файлу не стал равен числу строк, запрашиваемому пользователем, то выводим сообщение с информацией, сколько строк в файле.
2. Поиск по подстроке без спецсимволов. Необходимо реализовать возможность выбора режима поиска только по началу слова, по суффиксу слова и по любой части слова. Для всех трех случаев использован наивный алгоритм поиска подстроки. В случае поиска части слова просто проходим по файлу от знака к знаку до первого найденного вхождения. Если оно не найдено, выводим сообщение, что вхождений нет. В противном случае – выводим место первого вхождения. В случае поиска префикса заводим переменную, хранящую предыдущий знак из файла (предыдущий относительно совпадения первой буквы подстроки). Если вхождение найдено, проверяем, является ли предыдущий знак разделителем. Если является, то выводим место первого найденного вхождения, если нет, то выводим информацию, что такой префикс не найден. В случае суффикса все то же самое, как и в префиксе, только в переменной храним информацию о следующем знаке в файле.
3. Замена фрагмента строки на другую часть строки (возможно пустую). Проходим по файлу, ищем строку, которую надо заменить. Если такая строка найдена, то получаем байт, с которого она начинается. Если не найдена, выводим сообщение, что вхождения нет. Далее затираем эту строку в файле, затем записываем новую строку.
4. Проверка того, что конкретный файл не открыт в нескольких процессах программы. Доступ к файлу осуществляется посредством flock(), что позволяет синхранизировать работу не только процессов приложения, но и других процессов, которые захотят получить к нему доступ посредством flock().

Системные вызовы:
int flock(int fd, int operation); – блокирует файл по дескриптору fd с флагом operation. Блокировка может быть индивидуальной, групповой. Для разблокировки передается SHUN. Возвращает 0 при успехе.
int ftruncate(int fd, offt lenght); – устанавливает длину файлового дескриптора fd в lenght байт. Если lenght меньше исходного размера, то просто урезается, если больше – то заполняется нулевыми байтами. Возвращает 0 при успехе.
void *mmap(void *addr, sizet lenght, int prot, int flags, int fd, offt offset); – отображает часть файла fd длиной lenght в буффер со сдвигом offset. Так как отобажение происходит по размеру страницы, сдвиг должен быть кратен ей, а если lenght меньше размера страницы, то все, что дальше lenght заполняется нулевыми байтами. В случае

не NULL addr система получает подсказку, где начинается память для отображения. Prot отвечает за типы взаимодействия с файлом (чтение, запись, исполнение). flags отвечает за тип отображения, обязательно должно быть указано MAP SHARED или MAP PRIVATE (в первом случае изменения отобразятся на файле, во втором не отобразятся). В случае неудачи возвращает -1.
int munmap(void *addr, sizet lenght); – снимает отображение addr длиной lenght. Возвращает 0 в случае успеха.
int msync(void *addr, sizet lenght, int flags); – синхронизирует буффер addr длиной lenght с файлом, который был в него отображен с флагом flags. flags может быть установлен в синхронную и асинхронную обработки. Возвращает 0 в случае успеха.

2 Исходный код

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <inttypes.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/io.h>
#include <sys/mman.h>

#include "TextProcessor.h"

int main(int argc, char *argv[])
{
    int32_t size, fd = 0;
    struct stat fileStat;

    if (argc < 2 || argc > 6) {
        handler(0);
    } else if (argc == 2) {
        if (!strcmp(argv[1], "-help")) {
            handler(0);
        } else if (!strcmp(argv[1], "-commands")) {
            handler(2);
        } else {
            int32_t command, numStr;
            char opt[7], strToSearch[256], strToAdd[256], filename[256], oldStr[256],
                newStr[256];
            filename[0] = '\0';

            size = atoi(argv[1]);
            checkLimit(&size);

            do {
                menu(0);
                scanf("%d", &command);

                switch (command) {
                    case 0: {
                        menu(4);
                        if (fd) {
                            if (flock(fd, LOCK_UN)) {
                                handle_error("flock");
                            }
                            close(fd);
                        }
```

```
47
48                      scanf("%255s", filename);
49                      fd = open(filename, O_RDWR);
50                      if (fd == -1) {
51                          handle_error("open");
52                      }
53                      if (fstat(fd, &fileStat) == -1) {
54                          handle_error("fstat");
55                      }
56                      if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
57                          if (errno == EWOULDBLOCK) {
58                              printf("File is in usage\n");
59                              exit(1);
60                          } else {
61                              handle_error("flock");
62                          }
63                      }
64                      break;
65                  }
66
67                  case 1: {
68                      menu(1);
69                      scanf("%255s", strToSearch);
70                      printf("Prefix/suffix/part?\n");
71                      scanf("%7s", opt);
72                      if (!strcmp(opt, "prefix")) {
73                          searchPrefix(fd, fileStat.st_size, size, strToSearch, strlen(
                                strToSearch));
74                      } else if (!strcmp(opt, "suffix")) {
75                          searchSuffix(fd, fileStat.st_size, size, strToSearch, strlen(
                                strToSearch));
76                      } else if (!strcmp(opt, "part")) {
77                          searchPart(fd, fileStat.st_size, size, strToSearch, strlen(
                                strToSearch));
78                      } else {
79                          printf("Unknown command.\n");
80                      }
81                      break;
82                  }
83
84                  case 2: {
85                      menu(2);
86                      scanf("%d", &numStr);
87                      print(fd, fileStat.st_size, size, numStr);
88                      break;
89                  }
90
91                  case 3: {
92                      menu(3);
```

```
93                      printf("Enter substring to replace: ");
94                      scanf("%255s", oldStr);
95                      printf("Enter substring which it will be replaced: ");
96                      scanf("%255s", newStr);
97                      replace(fd, fileStat.st_size, size, oldStr, strlen(oldStr),
                            newStr, strlen(newStr));
98                      if (fstat(fd, &fileStat) == -1) {
99                          handle_error("fstat");
100                     }
101                     break;
102                 }
103
104                 case 4: {
105                     getStats(fd, fileStat.st_size, size);
106                     break;
107                 }
108
109                 case 5:
110                     break;
111
112                 default: {
113                     menu(5);
114                     break;
115                 }
116             }
117         } while (command != 5);
118     }
119 } else if (argc == 4) {
120     if (!strcmp(argv[3], "-stats")) {
121         fd = open(argv[1], O_RDWR);
122         if (fd == -1) {
123             handle_error("open");
124         }
125         if (fstat(fd, &fileStat) == -1) {
126             handle_error("fstat");
127         }
128         if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
129             if (errno == EWOULDBLOCK) {
130                 printf("File is in usage\n");
131                 exit(1);
132             } else {
133                 handle_error("flock");
134             }
135         }
136
137         size = atoi(argv[2]);
138         checkLimit(&size);
139         getStats(fd, fileStat.st_size, size);
140
```

```
141             flock(fd, LOCK_UN);
142         } else {
143             printf("Unknown command.\n");
144         }
145     } else if (argc == 5) {
146         if (!strcmp(argv[3], "-print")) {
147             fd = open(argv[1], O_RDWR);
148             if (fd == -1) {
149                 handle_error("open");
150             }
151             if (fstat(fd, &fileStat) == -1) {
152                 handle_error("fstat");
153             }
154             if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
155                 if (errno == EWOULDBLOCK) {
156                     printf("File is in usage\n");
157                     exit(1);
158                 } else {
159                     handle_error("flock");
160                 }
161             }
162 
163             int32_t numStr = atoi(argv[4]);
164             size = atoi(argv[2]);
165             checkLimit(&size);
166             print(fd, fileStat.st_size, size, numStr);
167             if (flock(fd, LOCK_UN)) {
168                 handle_error("flock");
169             }
170             close(fd);
171         } else if (!strcmp(argv[3], "-replace")) {
172             printf("File name: %s\n", argv[1]);
173             fd = open(argv[1], O_RDWR);
174             if (fd == -1) {
175                 handle_error("open");
176             }
177             if (fstat(fd, &fileStat) == -1) {
178                 handle_error("fstat");
179             }
180             if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
181                 if (errno == EWOULDBLOCK) {
182                     printf("File is in usage\n");
183                     exit(1);
184                 } else {
185                     handle_error("flock");
186                 }
187             }
188 
189             size = atoi(argv[2]);
```

```
190             checkLimit(&size);
191             char empty[0];
192             replace(fd, fileStat.st_size, size, argv[4], strlen(argv[4]), empty, strlen
                    (empty));
193             if (flock(fd, LOCK_UN)) {
194                 handle_error("flock");
195             }
196             close(fd);
197         } else {
198             printf("Unknown command\n");
199         }
200     } else if (argc >= 6) {
201         if (!strcmp(argv[3], "-search")) {
202             fd = open(argv[1], O_RDWR);
203             if (fd == -1) {
204                 handle_error("open");
205             }
206             if (fstat(fd, &fileStat) == -1) {
207                 handle_error("fstat");
208             }
209             if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
210                 if (errno == EWOULDBLOCK) {
211                     printf("File is in usage\n");
212                     exit(1);
213                 } else {
214                     handle_error("flock");
215                 }
216             }
217 
218             printf("File name: %s\n", argv[1]);
219             printf("Search substring: %s\n", argv[4]);
220             size = atoi(argv[2]);
221             checkLimit(&size);
222             if (!strcmp(argv[5], "part")) {
223                 searchPart(fd, fileStat.st_size, size, argv[4], strlen(argv[4]));
224             } else if (!strcmp(argv[5], "prefix")) {
225                 searchPrefix(fd, fileStat.st_size, size, argv[4], strlen(argv[4]));
226             } else if (!strcmp(argv[5], "suffix")) {
227                 searchSuffix(fd, fileStat.st_size, size, argv[4], strlen(argv[4]));
228             } else {
229                 printf("Unknown command.\n");;
230             }
231             if (flock(fd, LOCK_UN)) {
232                 handle_error("flock");
233             }
234             close(fd);
235 
236         } else if (!strcmp(argv[3], "-replace")) {
237             printf("File name: %s\n", argv[1]);
```

```
238             fd = open(argv[1], O_RDWR);
239             if (fd == -1) {
240                 handle_error("open");
241             }
242             if (fstat(fd, &fileStat) == -1) {
243                 handle_error("fstat");
244             }
245             if (flock(fd, LOCK_EX|LOCK_NB) == -1) {
246                 if (errno == EWOULDBLOCK) {
247                     printf("File is in usage\n");
248                     exit(1);
249                 } else {
250                     handle_error("flock");
251                 }
252             }
253 
254             size = atoi(argv[2]);
255             checkLimit(&size);
256             replace(fd, fileStat.st_size, size, argv[4], strlen(argv[4]), argv[5],
                    strlen(argv[5]));
257             if (flock(fd, LOCK_UN)) {
258                 handle_error("flock");
259             }
260             close(fd);
261         } else {
262             printf("Unknown command.\n");
263         }
264 
265     } else {
266         handler(1);
267     }
268 
269     return 0;
270 }
271 
272 
273 TextProcessor.h
274 #ifndef GENERAL_H
275 #define GENERAL_H
276 
277 #include <stdio.h>
278 #include <string.h>
279 #include <stdlib.h>
280 #include <stdbool.h>
281 #include <ctype.h>
282 #include <inttypes.h>
283 #include <fcntl.h>
284 #include <sys/types.h>
285 #include <sys/mman.h>
```

```
286 #include <sys/stat.h>
287 #include <unistd.h>
288 #include <sys/file.h>
289 #include <errno.h>
290
291 #define handle_error(msg) \
292     do { perror(msg); exit(EXIT_FAILURE); } while (0)
293
294 void handler(const int32_t derp);
295 void menu(const int32_t act);
296 void searchPrefix(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit,
        char *subString, const int32_t strSize);
297 void searchSuffix(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit,
        char *subString, const int32_t strSize);
298 void searchPart(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, char
       *subString, const int32_t strSize);
299 void checkLimit(int *userLimit);
300 void print(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, const
        int32_t line);
301 void replace(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, char *
        oldString, const int32_t sizeOldStr, char *newString, const int32_t sizeNewStr);
302 void getStats(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit);
303
304 #endif
305
306 TextProcessor.c
307 #include "TextProcessor.h"
308
309 void handler(const int32_t derp)
310 {
311     if (derp == 0) {
312         printf("use '/PathToFile(if using command) RAMLimitInBytes Command(optional)'\n
              ");
313         printf("use '-help' to show this message\n");
314         printf("use '-commands' to show commands and parameters\n");
315     } else if (derp == 1) {
316         printf("Unknown command.\n");
317         printf("Use:\n");
318         printf("-stats\n");
319         printf("-print NumberString\n");
320         printf("-search SubString prefix/suffix/part\n");
321         printf("-replace OldSubString NewSubString (maybe empty)\n");
322     } else if (derp == 2) {
323         printf("Use:\n");
324         printf("-stats\n");
325         printf("-print NumberString\n");
326         printf("-search SubString prefix/suffix/part\n");
327         printf("-replace OldSubString NewSubString \n");
328     }
```

```
329
330     exit(0);
331 }
332
333 void menu(const int32_t act)
334 {
335     switch (act) {
336         case 0: {
337             printf("Avaliable commands:\n");
338             printf("0) Choose file\n");
339             printf("1) Search substring\n");
340             printf("2) Print string\n");
341             printf("3) Replace substring\n");
342             printf("4) Show stats\n");
343             printf("5) Exit\n");
344             break;
345         }
346         case 1: {
347             printf("Enter substring to search: ");
348             break;
349         }
350         case 2: {
351             printf("Enter number string to show: ");
352             break;
353         }
354         case 3: {
355             break;
356         }
357         case 4: {
358             printf("Enter filename: ");
359         break;
360         }
361         case 5: {
362             printf("Unknown command.\n");
363             break;
364         }
365     }
366 }
367
368 void searchPart(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, char
        *subString, const int32_t strSize)
369 {
370     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
371     int32_t offset = 0, index = 0, bytepos = 1;
372     char *addr;
373
374     while (offset < fileSize) {
375         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
376         if (addr == MAP_FAILED) {
```

```
377             handle_error("mmap");
378         }
379 
380         for (int i = 0; i < size; ++i, ++bytepos) {
381             if (addr[i] == subString[index]) {
382                 ++index;
383                 for (int32_t j = i + 1; (index < strSize) && (j < size); ++index, ++j) {
384                     if (subString[index] != addr[j] || isspace(addr[j])) {
385                         index = 0;
386                         break;
387                     }
388                 }
389 
390                 if (index == strSize) {
391                     printf("First entry was found at %d byte\n", bytepos);
392                     return;
393                 }
394             }
395         }
396 
397         if (munmap(addr, size) == -1) {
398             handle_error("munmap");
399         }
400         offset += size;
401     }
402     printf("There is no entries in file\n");
403 }
404 
405 void searchPrefix(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit,
        char *subString, const int32_t strSize)
406 {
407     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
408     int32_t offset = 0, index = 0, bytepos = 1;
409     char *addr;
410 
411     while (offset < fileSize) {
412         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
413         if (addr == MAP_FAILED) {
414             handle_error("mmap");
415         }
416 
417         int prev = '\n';
418         for (int i = 0; i < size; ++i, ++bytepos) {
419             if (addr[i] == subString[index] && isspace(prev)) {
420                 ++index;
421                 for (int32_t j = i + 1; (index < strSize) && (j < size); ++index, ++j) {
422                     if (subString[index] != addr[j] || isspace(addr[j])) {
423                         index = 0;
424                         break;
```

```
425                     }
426                 }
427 
428                 if (index == strSize) {
429                     printf("First entry prefix was found at %d byte\n", bytepos);
430                     return;
431                 }
432             }
433             prev = addr[i];
434         }
435 
436         if (munmap(addr, size) == -1) {
437             handle_error("munmap");
438         }
439         offset += size;
440     }
441     printf("There is no entries in file\n");
442 }
443 
444 
445 void searchSuffix(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit,
        char *subString, const int32_t strSize)
446 {
447     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
448     int32_t offset = 0, index = 0, bytepos = 1;
449     char *addr;
450     bool found = false;
451 
452     while (offset < fileSize) {
453         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
454         if (addr == MAP_FAILED) {
455             handle_error("mmap");
456         }
457 
458         char next = '\n';
459 
460         for (int32_t i = 0; i < size; ++i, ++bytepos) {
461             index = 0;
462             if (addr[i] == subString[index]) {
463                 ++index;
464                 for (int32_t j = i + 1; (index < strSize) && (j < size); ++index, ++j) {
465                     if (j + 1 < size) {
466                         next = addr[j + 1];
467                     } else {
468                         next = '\n';
469                     }
470                     if (subString[index] != addr[j] || isspace(addr[j])) {
471                         index = 0;
472                         break;
```

```
473                     }
474                 }
475
476                 if (index == strSize && isspace(next)) {
477                     printf("First entry suffix was found at %d byte\n", bytepos);
478                     found = true;
479                     break;
480                 }
481             }
482         }
483
484         if (munmap(addr, size) == -1) {
485             handle_error("munmap");
486         }
487         offset += size;
488     }
489     if (!found) {
490         printf("There is no entries in file\n");
491     }
492 }
493
494 void checkLimit(int *userLimit)
495 {
496     int32_t pageSize = sysconf(_SC_PAGESIZE);
497
498     if (*userLimit < pageSize) {
499         printf("Your limit is less than page size\n");
500         printf("Mmap limit will be set to page size - %d\n", pageSize);
501         *userLimit = pageSize;
502     } else if ((*userLimit % pageSize) != 0) {
503         int32_t count = 0;
504         while (*userLimit > pageSize) {
505             *userLimit -= pageSize;
506             ++count;
507         }
508
509         *userLimit = pageSize * count;
510     }
511 }
512
513 void print(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, const
        int32_t line)
514 {
515     if (line < 0) {
516         printf("Number string must be > 0.\n");
517         return;
518     }
519
520     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
```

```
521     int32_t offset = 0, lineCnt = 1;
522     char *addr;
523     bool printed = false;
524 
525     while (offset < fileSize) {
526         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
527         if (addr == MAP_FAILED) {
528             handle_error("mmap");
529         }
530 
531         for (int32_t i = 0; i < size; ++i) {
532             if (line == lineCnt) {
533                 putchar(addr[i]);
534                 printed = true;
535             }
536             if (addr[i] == '\n') {
537                 ++lineCnt;
538             }
539         }
540 
541         if (munmap(addr, size) == -1) {
542             handle_error("munmap");
543         }
544 
545         if (printed) {
546             return;
547         }
548 
549         offset += size;
550     }
551     if (!printed) {
552         --lineCnt;
553         printf("Wrong number string. Number of strings: %d\n", lineCnt);
554         return;
555     }
556 }
557 
558 
559 void getStats(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit)
560 {
561     char *addr;
562     int32_t offset = 0, bytepos = 1, lines = 1;
563     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
564 
565     while(offset < fileSize) {
566         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
567         if (addr == MAP_FAILED) {
568             handle_error("mmap");
569         }
```

```
570
571         for (int32_t i = 0; i < size; ++i, ++bytepos) {
572             if (addr[i] == '\n') {
573                 ++lines;
574             }
575         }
576
577         if (munmap(addr, RAMLimit) == -1) {
578             handle_error("munmap");
579         }
580
581         offset += RAMLimit;
582     }
583
584     if (lines > 1) {
585         --lines;
586     }
587
588     printf("Amount of symbols - %d\nAmount of lines - %d\n", fileSize, lines);
589 }
590
591 int32_t removeOldString(const int32_t fd, const int32_t fileSize, const int32_t
        RAMLimit, const int32_t from, const int32_t to)
592 {
593     if ((from < 0) || (to < 0) || (from > to) || (to - 1 > fileSize)) {
594         printf("Incorrect bounds.\n");
595         return 0;
596     }
597     int32_t diff = (fileSize == to - 1) ? 1 : fileSize - to;
598     char *old = (char *) malloc(sizeof(char) * diff);
599     if (old == NULL) {
600         handle_error("malloc-");
601     }
602     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
603     int32_t offset = 0, bytepos = 1, k = 0;
604     char *addr;
605
606     while (offset < fileSize) {
607         addr = (char *) mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
608         if (addr == MAP_FAILED) {
609             handle_error("mmap");
610         }
611
612         for (int32_t i = 0; i < size; ++i, ++bytepos) {
613             if (bytepos >= to) {
614                 old[k] = addr[i];
615                 ++k;
616             }
617         }
```

```
618
619         if (munmap(addr, size) == -1) {
620             handle_error("munmap");
621         }
622
623         offset += size;
624     }
625
626     int32_t newFileSize = fileSize - to + from;
627     if (ftruncate(fd, newFileSize)) {
628         handle_error("ftruncate-");
629     }
630
631     bytepos = 1, offset = 0, k = 0;
632     size = (RAMLimit > newFileSize) ? newFileSize : RAMLimit;
633
634     while(offset < newFileSize) {
635         addr = (char *)mmap(NULL, (size_t)size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
                offset);
636         if (addr == MAP_FAILED) {
637             handle_error("mmap");
638         }
639
640         for (int32_t i = 0; i < size; ++i, ++bytepos) {
641             if (bytepos >= from) {
642                 addr[i] = old[k];
643                 ++k;
644             }
645         }
646
647         if (msync(addr, size, MS_SYNC) == -1) {
648             handle_error("msync");
649         }
650
651         if (munmap(addr, size) == -1) {
652             handle_error("munmap");
653         }
654
655         offset += size;
656     }
657
658     free(old);
659     return newFileSize;
660 }
661
662 void insertNewString(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit,
        char *string, const int32_t from, const int32_t to)
663 {
664     if ((from < 0) || (to < 0) || (from > to)) {
```

```
665         printf("Incorrect bounds.\n");
666         return;
667     }
668 
669     char *old = (char*) malloc(sizeof(char) * (fileSize - from));
670     if (old == NULL) {
671         handle_error("malloc+");
672     }
673     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
674     int32_t offset = 0, bytepos = 1, k = 0;
675     char *addr;
676 
677     while (offset < fileSize) {
678         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
679         if (addr == MAP_FAILED) {
680             handle_error("mmap");
681         }
682 
683         for (int32_t i = 0; i < size; ++i, ++bytepos) {
684             if (bytepos >= from) {
685                 old[k] = addr[i];
686                 ++k;
687             }
688         }
689 
690         if (munmap(addr, size) == -1) {
691             handle_error("munmap");
692         }
693 
694         offset += size;
695     }
696 
697     int32_t newFileSize = fileSize + to - from;
698     if (ftruncate(fd, newFileSize)) {
699         handle_error("ftruncate+");
700     }
701 
702     int32_t l = 0;
703     bytepos = 1, offset = 0, k = 0;
704     size = (RAMLimit > newFileSize) ? newFileSize : RAMLimit;
705 
706     while (offset < newFileSize) {
707         addr = (char *)mmap(NULL, (size_t)size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
                offset);
708         if (addr == MAP_FAILED) {
709             handle_error("mmap");
710         }
711 
712         for (int32_t i = 0; i < size; ++i, ++bytepos) {
```

```
713             if ((bytepos >= from) && (bytepos < to)) {
714                 addr[i] = string[l];
715                 ++l;
716             }
717 
718             if (bytepos >= to) {
719                 addr[i] = old[k];
720                 ++k;
721             }
722         }
723 
724         if (msync(addr, size, MS_SYNC) == -1) {
725             handle_error("msync");
726         }
727 
728         if (munmap(addr, size) == -1) {
729             handle_error("munmap");
730         }
731 
732         offset += size;
733     }
734 
735     free(old);
736 }
737 
738 
739 
740 void replace(const int32_t fd, const int32_t fileSize, const int32_t RAMLimit, char *
        oldString,
741              const int32_t sizeOldStr, char *newString, const int32_t sizeNewStr)
742 {
743     int32_t size = (RAMLimit > fileSize) ? fileSize : RAMLimit;
744     int32_t offset = 0, index = 0, bytepos = 1, from = 0, to = 0;
745     char *addr;
746     bool found = false;
747 
748 
749     while (offset < fileSize) {
750         addr = (char *)mmap(NULL, (size_t)size, PROT_READ, MAP_PRIVATE, fd, offset);
751         if (addr == MAP_FAILED) {
752             handle_error("mmap");
753         }
754 
755         for (int32_t i = 0; i < size; ++i, ++bytepos) {
756             if (addr[i] == oldString[index]) {
757                 ++index;
758                 for (int32_t j = i + 1; (index < sizeOldStr) && (j < size); ++index, ++j
                        ) {
759                     if (oldString[index] != addr[j] || addr[j] == '\n') {
```

```
760                         index = 0;
761                         break;
762                     }
763                 }
764 
765                 if (index == sizeOldStr) {
766                     from = bytepos;
767                     found = true;
768                     index = 0;
769                     break;
770                 }
771             }
772         }
773         if (munmap(addr, size) == -1) {
774             handle_error("munmap");
775         }
776         offset += size;
777     }
778 
779 
780     if (!found) {
781         printf("This substring no entries in file.\n");
782         return;
783     }
784 
785     int32_t newFileSize = removeOldString(fd, fileSize, size, from, from + sizeOldStr);
786     insertNewString(fd, newFileSize, RAMLimit, newString, from, from + sizeNewStr);
787 }
```

3 Тестирование

- Тестирование в интерактивном режиме:

  Запуск без ключей

```
karma@karma:~/mai_study/OS/lab4$ ./run
use '/PathToFile(if using command) RAMLimitInBytes Command(optional)'
use '-help'to show this message
use '-commands'to show commands and parameters
```

  Вывод помощи для работы с программой

```
karma@karma:~/mai_study/OS/lab4$ ./run -help
use '/PathToFile(if using command) RAMLimitInBytes Command(optional)'
use '-help'to show this message
use '-commands'to show commands and parameters
```

  Просмотр списка допустимых команд

```
karma@karma:~/mai_study/OS/lab4$ ./run -commands
Use:
-stats
-print NumberString
-search SubString prefix/suffix/part
-replace OldSubString NewSubString
```

  Некорректный флаг

```
karma@karma:~/mai_study/OS/lab4$ ./run -chtodelat
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
5
```

  Попытка открыть несуществующий файл

```
karma@karma:~/mai_study/OS/lab4$ ./run -chtodelat
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: kek
open: No such file or directory
```

Открывем существующий файл и выполняем действия

```
karma@karma:~/mai_study/OS/lab4$ ./run go
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: test
```

Тестирование печати определенной строки. Некорректные данные.

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
Enter number string to show: 0
Wrong number string. Number of strings: 8
```

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
Enter number string to show: 9
Wrong number string. Number of strings: 8
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
Enter number string to show: -10
Number string must be >0.
```

Корректные данные

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
Enter number string to show: 1
easyeasyeasy
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
```

```
Enter number string to show: 5
antihype666
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
2
Enter number string to show: 8
karma police
```

Тестирование поиска.
1. Поиск суффикса.
Поиск несуществующего суффикса

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: kek
Prefix/suffix/part?
suffix
There is no entries in file
```

Поиск суффикса в начале файла, в строке с повторяющимися частями

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: easy
```

```
Prefix/suffix/part?
suffix
First entry suffix was found at 9 byte
```

Поиск суффикса в конце файла

```
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: ice
Prefix/suffix/part?
suffix
First entry suffix was found at 82 byte
```

2. Поиск префикса.
Поиск несуществующего префикса

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: kek
Prefix/suffix/part?
prefix
There is no entries in file
```

Поиск префикса в начале файла.

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
```

```
4) Show stats
5) Exit
1
Enter substring to search: easy
Prefix/suffix/part?
prefix
First entry prefix was found at 1 byte
```

Поиск префикса в конце файла

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: pol
Prefix/suffix/part?
prefix
First entry prefix was found at 79 byte
```

Поиск префикса в середине файла.

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: anti
Prefix/suffix/part?
prefix
First entry prefix was found at 45 byte
```

3. Поиск части строки Поиск несуществующей части строки

```
Avaliable commands:
```

```
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: kek
Prefix/suffix/part?
part
There is no entries in file
```

Поиск части строки в начале файла (часть слова)

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: easyea
Prefix/suffix/part?
part
First entry was found at 1 byte
```

Поиск части строки в конце файла (часть слова)

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: olice
Prefix/suffix/part?
part
First entry was found at 80 byte
```

Поиск части строки в середине файла (полное слово).

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
1
Enter substring to search: real
Prefix/suffix/part?
part
First entry was found at 35 byte
```

Тестирование замены подстроки.
Замена несуществующей подстроки.

```
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
3
Enter substring to replace: kek
Enter substring which it will be replaced: stop
This substring no entries in file.
```

Замена подстроки на пустую строку

```
karma@karma:~/mai_study/OS/lab4$ cat test
easyeasyeasy
eee boy
think about
real talk
antihype666
123
hello world
karma police
```

```
karma@karma:~/mai_study/OS/lab4$ ./run go
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: test
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
3
Enter substring to replace: anti
Enter substring which it will be replaced:
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
5
karma@karma:~/mai_study/OS/lab4$ cat test
easyeasyeasy
eee boy
think about
real talk
hype666
123
hello world
karma police
```

Замена большей строки на меньшую

```
karma@karma:~/mai_study/OS/lab4$ ./run go
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: test
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
3
Enter substring to replace: easyeasyeasy
Enter substring which it will be replaced: easy
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
5
karma@karma:~/mai_study/OS/lab4$ cat test
easy
eee boy
think about
real talk
hype666
123
hello world
karma police
```

Замена меньшей строки на большую

```
karma@karma:~/mai_study/OS/lab4$ ./run go
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: test
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
3
Enter substring to replace: karma
Enter substring which it will be replaced: karmaSuperStar
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
5
karma@karma:~/mai_study/OS/lab4$ cat test
easy
eee boy
think about
real talk
hype666
123
hello world
karmaSuperStar police
```

- Тестирование в режиме одной команды
  Это тестирование просто показывает, что программа работает в таком режиме. Вариации использования функций представлены в тестировании выше.

  Статистика файла

```
karma@karma:~/mai_study/OS/lab4$ ./run test 250 -stats
Your limit is less than page size
Mmap limit will be set to page size -4096
Amount of symbols -82
Amount of lines -8
```

  Малое количество оперативной памяти

```
karma@karma:~/mai_study/OS/lab4$ ./run test 1 -print 1
Your limit is less than page size
Mmap limit will be set to page size -4096
easy
```

  Большое количество оперативной памяти

```
karma@karma:~/mai_study/OS/lab4$ ./run test 1000000 -search alk suffix
File name: test
Search substring: alk
First entry suffix was found at 33 byte
```

  Проверка открытия файла двумя программами Первый процесс:

```
karma@karma:~/mai_study/OS/lab4$ ./run 123
Your limit is less than page size
Mmap limit will be set to page size -4096
Avaliable commands:
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
0
Enter filename: test
Avaliable commands:
```

```
0) Choose file
1) Search substring
2) Print string
3) Replace substring
4) Show stats
5) Exit
```

Второй процесс:

```
karma@karma:~/mai_study/OS/lab4$ ./run test 1000 -replace real float
File name: test
File is in usage
```

4 Выводы

Признаться честно, лабораторная была не из легких. Но я справилась. Мой текстовый процессор выполняет все заявленные фукнций. Конечно, он несоврешенен. В первую очередь, это связано с моей методологией разработки. Передо мной стояла цель как можно раньше предоставить преподавателю (который в условиях учебного процесса является заказчиком) рабочую версию своей программы, в то же время не потеряв в качестве.

Из явных недочетов можно отметить, например, осуществление поиска подстроки в моей программе. Я использую наивный алгоритм поиска подстроки, его сложность неоптимальна и составляет $O(n^2)$. Гораздо грамотнее было бы использовать более эффективные классические алгоритмы поиска, такие как алгоритм Кнута-Моррриса-Пратта или алгоритм Бойера-Мура. Также минусом поиска является то, что поиск осуществляется только до первого найденного вхождения. Все остальные функций выполняются за линейное время (один проход по файлу), что не может не радовать. Хотя используя отображение файла на дерево, как это делается в текстовом процессоре Emacs, можно было бы добиться большей производительности.

Особо хочется отметить системный вызов mmap, который позволяет эффективно работать с большими файлами. Он отображает кэш в адресное пространство программы, избегая поблочного копирования из кэша в память, что позволяет оптимально расходовать память. Системные вызовы для работы с файловой системой предоставляют разработчику удоный инструмент для ее контроля.

Единственной непонятной вещью для меня остается требование реализации 2-х режимов. Из-за того что их два, усложнилась логика взаимодействия с пользователем. На мой взгляд, работа с реализованной программой перестала быть интуитивно понятной. Возможно, стоило более явно обозначить пользователю возможность использования именно двух режимов.