**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Отчет по курсовой работе**

**по курсу "Операционные системы"**

Студент Захаров И С

группы М8О-208Б-17

№ по списку \_

Работа выполнена \_\_.\_\_.\_\_\_\_

Преподаватель Миронов

Отчет сдан \_\_.\_\_.\_\_\_\_

Оценка

Москва 201\_

**Цель:**

* Приобретение практических навыков в использовании знаний, полученных в течении курса
* Проведение исследования в выбранной предметной области

**Задание:**

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

*Вариант задания*:

Реализовать текстовый прероцессор с поддержкой регулярных выражений , кэширования и mmap .

**Введение:**

Для реализации текстового препроцессора необходимо было ознакомиться с основными тектовыми редакторами , я выбрал sed .

Он привлекает своим синтаксисом , простотой в использовании . Для реализации я взял три команды : замена слова в тексте , удаление n- ой строки , удаление слова в тексте .

**Решение:**

Для решения задачи я создал два файла : данные и кэш .

Сначала я открывал эти файлы и узнавал их статистику : для файла с кэшэм это надо , так как файл необходимо увеличивать за свет новых команд , поэтому я также использовал функцию ftruncate , которая уличивает размер файла , в результате чего я смог добавлять новые команды в кэш , не получая ошибки сигментирования .

Далее я проверял , есть ли данная команда в кэшэ с помощью поиска подстроки - команды в строке - отображенной в память файла с кэшэм .

В конце программы начитается вся работа программы :

В функции change\_file происодит анализ запрошенной команды : проверка и парсинг . Далее просиходит перенаправление файла и команды в функции для изменения файла .

В функции del\_string происходит удаление n-ой строки с помощью разделения отображенного в память файла на три части : до удаляемой строки , сама строка и после удаляемой строки . Затем происходит копирование 1 и 3 в память .

Аналогично работает функция replace\_word .

Функция del\_word основана на работе функции replace\_word .

Кэширование реализовано следующим способом : В файл сохраняются команды и в функции find\_command команды поочередно предоставляются пользователю .

**Сценарий выполнения работы:**

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 65

ggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg

fggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg

before after fggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 66

fggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg

before after

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 0

before after

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 0

before after

succes

nick@nick-VirtualBox:~/Desktop/игорь$

nick@nick-VirtualBox:~/Desktop/игорь$ cat > file.txt

wdsfsdf

sdfsdf

sdfsdfsdf

sdgsgdsgsdgsdgsdg

^C

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 7

wdsfsdf

sdfsdf

sdfsdfsdf

sdgsgdsgsdgsdgsdg

before after sdfsdf

sdfsdfsdf

sdgsgdsgsdgsdgsdg

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 6

sdfsdf

sdfsdfsdf

sdgsgdsgsdgsdgsdg

before after sdfsdfsdf

sdgsgdsgsdgsdgsdg

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 9

sdfsdfsdf

sdgsgdsgsdgsdgsdg

before after sdgsgdsgsdgsdgsdg

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 17

sdgsgdsgsdgsdgsdg

before after

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-1d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

0 0

before after

succes

nick@nick-VirtualBox:~/Desktop/игорь$ cat file.txt

nick@nick-VirtualBox:~/Desktop/игорь$ cat > file.txt

1)sa;ldkfslkfd;lsdkf

2)sdlfkjlskdfj;lk

3)dz;lkf;sldkf;lskf

4)

slf;slkdf;lsdkf

^C

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-2d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

20 38

1)sa;ldkfslkfd;lsdkf

2)sdlfkjlskdfj;lk

3)dz;lkf;sldkf;lskf

4)

slf;slkdf;lsdkf

before 1)sa;ldkfslkfd;lsdkf

after 3)dz;lkf;sldkf;lskf

4)

slf;slkdf;lsdkf

succes

nick@nick-VirtualBox:~/Desktop/игорь$ cat

file.txt

1)sa;ldkfslkfd;lsdkf

3)dz;lkf;sldkf;lskf

4)

slf;slkdf;lsdkf

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt "-4d"

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

43 59

1)sa;ldkfslkfd;lsdkf

3)dz;lkf;sldkf;lskf

4)

slf;slkdf;lsdkf

before 1)sa;ldkfslkfd;lsdkf

3)dz;lkf;sldkf;lskf

4)

after

succes

nick@nick-VirtualBox:~/Desktop/игорь$ ./a.out file.txt

In this text preprocessor you can use the following commands :

key -r <old>/<new> : for replace old string on new string one time

key -<N>d : for delete this string in file

key -d <word>: for delete this word from file

enter option for change file or enter n for find comand in history

n

enter your choice and press 'y' or press 'n' to enter new command

command from cash : -1d

a

command from cash : -1d

a

command from cash : -1d

a

command from cash : -2d

a

command from cash : yes

a

command from cash : -d kok

a

command from cash : -d kok

a

command from cash : -d kok

a

command from cash : -r kek/sk

a

command from cash : -3d

a

command from cash : -3d

a

command from cash : -4d

a

command from cash : -3d

a

command from cash : -3d

a

command from cash : -4d

a

command from cash : -5d

a

command from cash : -6d

y

end with -6d

no

**Листинг программы** :

**main.c**

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <sys/mman.h>

#include <string.h>

struct stat buff\_2;

char file\_name[128];

typedef struct command{//структура для хранения ключа и поля команды

char \*key;

char todo[256];

}Command;

void replace\_word(char \*);

void menu(){

printf("In this text preprocessor you can use the following commands :\n");

printf("key -r <old>/<new> : for replace old string on new string one time\n");

printf("key -<N>d : for delete this string in file\n");

printf("key -d <word>: for delete this word from file\n");

}

int check\_command(char \*command){

if(strlen(command)<3 //минимальная длина команды

&&command[0]!='-'//начало любой команды

&&command[1]!='r'//замена

&&(command[1]<48||command[1]>57)//номер строки

&&command[1]!='d')//удаление слова

return -1;

if(command[1]=='r')

return 1;

if(command[1]=='d')

return 2;

if(command[1]>=48||command[1]<=57)

return 3;

}

Command\* parse\_command(char \*command,int command\_num){//возвращает разобранную на части команду

char todo[50];

Command \*cmd = malloc(sizeof(Command));//выделяем память для возвращаемого значения

switch(command\_num){

case 1:

cmd->key="-r";

strncpy(todo,command+3,strlen(command)-2);

sprintf(cmd->todo,"%s",todo);

break;

case 2:

cmd->key="-d";

strncpy(todo,command+3,strlen(command)-2);

sprintf(cmd->todo,"%s",todo);

break;

case 3:

cmd->key=command;

break;

}

return cmd;

}

void del\_word(char \*todo){//пользуемся уже написанной функцией replace\_word

char \*todo\_2 = malloc(sizeof(todo)+sizeof(char));

sprintf(todo\_2,"%s/",todo);

replace\_word(todo\_2);

}

void del\_string(char \*key){

int fd = open(file\_name,O\_RDWR);

int \*ptr = mmap(NULL,buff\_2.st\_size,PROT\_READ|PROT\_WRITE,MAP\_SHARED,fd,0);

close(fd);

if(key[strlen(key)-1]!='d')//проверка на некорректную команду

return;

char \*tmp = malloc(sizeof(key));

strncpy(tmp,key+1,strlen(key)-2);

int number\_str = atoi(tmp);

free(tmp);

char \*temp = malloc(sizeof(ptr));

sprintf(temp,"%s",(char\*)ptr);

int counter = 0;

int strings\_number = 0;

while(counter<strlen(temp)){

if(temp[counter]=='\n')

strings\_number++;

if(strings\_number==number\_str)

break;

counter++;

}

if(number\_str>strings\_number)

return;

int a,b;//начало и конец удяляемой строки

counter = 0;

strings\_number = 0;

while(counter<strlen(temp)){//определяем начало

if(temp[counter]=='\n')

strings\_number++;

if(strings\_number==number\_str-1){

counter++;

break;

}

counter++;

}

a = counter;

while(counter<strlen(temp)){//определяем конец

counter++;

if(temp[counter]=='\n')

break;

}

b = counter;

//выделяем память под правые и левые части от удаляемой строки

char \*before = malloc(sizeof(temp));

char \*after = malloc(sizeof(temp));

if(a==1)//если удалямая строка - первая

before="";

else

strncpy(before,(char\*)ptr,a);

if(after==NULL)//если удалямая строка - последняя

after="";

else

strncpy(after,(char\*)ptr+b+1,strlen(temp) - b-1);

memset(ptr,0,strlen((char\*)ptr));//очищаем память

sprintf((char\*)ptr,"%s%s",before,after);//записываем в память новые данные

}

void replace\_word(char \*todo){

int fd = open(file\_name,O\_RDWR);

int \*ptr = mmap(NULL,buff\_2.st\_size,PROT\_READ|PROT\_WRITE,MAP\_SHARED,fd,0);

close(fd);

int counter=0;

while(counter<strlen(todo)){//поиск разделителя

if(todo[counter]=='/')

break;

counter++;

}

if(counter==strlen(todo)-1&&todo[strlen(todo)-1]!='/'){//некорректно задана команда

printf("error in command (enter slash between old and new words)\n");

return;

}

char old[50],new[50];//старая и новая строки

memset(old,0,50);

memset(old,0,50);

strncpy(old,todo,counter);

strncpy(new,todo+counter+1,strlen(todo)-counter);

char \*find = malloc(buff\_2.st\_size);//поиск старой строки в отображенной памяти

find = strstr((char\*)ptr,old);

if(find==NULL)

return;

counter = strlen((char\*)ptr) - strlen(find);//индекс первого символа первого вхождения строки

//выделение памяти под две части памяти без удаляемого слова

char \*before = malloc(sizeof(char)\*counter);

char \*after = malloc(sizeof(char)\*(strlen((char\*)ptr) - counter - strlen(old)));

//копирование памяти в инициализированные выше переменные

strncpy(before,(char\*)ptr,counter);

strncpy(after,(char\*)ptr+counter+strlen(old),strlen((char\*)ptr) - counter - strlen(old));

memset(ptr,0,strlen((char\*)ptr));//очищаем память

sprintf((char\*)ptr,"%s%s%s",before,new,after);//копирование памяти в отображенный в память файл

}

void change\_file(char \*file\_name,char \*command){//выбираем правило обработки файла

Command \*cmd = NULL;

int command\_num = check\_command(command);

if(command\_num>0){

cmd = parse\_command(command,command\_num);

}

if(!strcmp(cmd->key,"-r")){

replace\_word(cmd->todo);

}

if(!strcmp(cmd->key,"-d")){

del\_word(cmd->todo);

}

else{

del\_string(cmd->key);

}

}

char \*find\_command(){//поиск команды в кэше

int fd = open("history.txt",O\_RDWR);

lseek(fd,0,0);//сдвигаем указатель на начало файла

pid\_t pid;

int offset=0,i=0;//отступ от начала файла и счетчик команд

char cash\_commands[50][50];//запросы из кэша

memset (cash\_commands, 0, sizeof(cash\_commands));//обнуляем массив

char \*command = malloc(sizeof(char)\*50),c;

printf("enter your choice and press 'y' or press 'n' to enter new command\n");

while(read(fd,&c,1)>0){//заполняем массив данными из кэша

if(c=='\n'){

i++;

}

else{

strcat(cash\_commands[i],&c);

}

}

offset=0;

printf("command from cash : %s\n",cash\_commands[offset]);

offset++;

scanf("%s",command);

while(strcmp(command,"n")&&strcmp(command,"y")&&offset<i){

pid = fork();

if(pid==0){

execl("/usr/bin/clear","./clear",NULL);//запускаем очитску экрана

exit(0);

}

else{

wait();

printf("command from cash : %s\n",cash\_commands[offset]);

offset++;

scanf("%s",command);

}

}

offset--;

if(!strcmp(command,"y"))

strcpy(command,cash\_commands[offset]);

else{

printf("enter command\n");

scanf("%s",command);

}

printf("end with %s\n",command);

close(fd);

return command;

}

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

menu();

if(argc<2){

printf("enter file and command!\n");

return 0;

}

strcpy(file\_name,argv[1]);

int fd = open(argv[1],O\_RDWR,0666);//открываем файл для изменения

if(fd<0){

perror("open");

exit(-1);

}

int info\_file\_2 = fstat(fd,&buff\_2);//узнаем размер файла

if(info\_file\_2!=0){

perror("fstat file");

exit(-1);

}

close(fd);

struct stat buff;

int history = open("history.txt",O\_RDWR|O\_CREAT,0666);//открываем файл с историей запросов

int info\_file = fstat(history,&buff);//узнаем размер файла с кэшэм

if(info\_file!=0){

perror("fstat cash");

exit(-1);

}

int \*ptr\_history = mmap(0,buff.st\_size,PROT\_WRITE|PROT\_READ,MAP\_SHARED,history,0);

lseek(history,buff.st\_size,0);

char command[50];

if(argc==3){

strcpy(command,argv[2]);

if(strstr((char\*)ptr\_history,command)==NULL){

ftruncate(history,buff.st\_size+strlen(argv[2]));//увеличиваем размер файла с кэшэм

write(history,argv[2],strlen(argv[2]));

write(history,"\n",1);

}

}

else{

printf("enter option for change file or enter n for find comand in history\n");

scanf("%s",command);

if(!strcmp(command,"n"))

strcpy(command,find\_command());

ftruncate(history,buff.st\_size+strlen(command));

if(strstr((char\*)ptr\_history,command)==NULL){

printf("no\n");

write(history,command,strlen(command));

write(history,"\n",1);

}

}

close(history);

//int \*ptr = mmap(0,buff\_2.st\_size,PROT\_READ|PROT\_WRITE,MAP\_SHARED,fd,0);

change\_file(argv[1],command);//меняем сам файл

printf("succes\n");

return 0;

}

**Вывод**:

Текстовый препроцессор - интересная для реализации вещь . В моей курсовой работе я встетил много интересных ошибок , которые потребовали много времени для исправления , некоторые из них я решил нелучшим способом .

Интересно было реализовывать кэширование с помощью fork и mmap .

Так же я много использовал функций по работе со строками , что закрепило мои знания в си .

Много ошибок возникало из-за плохого понимания процесса выделения памяти и арифметики указателей .

Моя программа несовершенна , требует доработок . Но то , что я получил - результат трудной работы .

В курсовой можно было использовать потоки , чтобы ускорить процесс обработки теста , а также можно было реализовать больше команд , аналогичных командам утилиты sed .

Здесь я испольховал знания , полученные в течений курса обучения : fork ,mmap .