Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №4 по курсу**

**«Операционные системы»**

**Работа с файловыми системами в ОС**

Студент: Ефимов Александр Владимирович

Группа: М80 – 201Б-18

Вариант: 6

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2019

1. **Постановка задачи**

На вход программе подается команда интерпретатора команд. Программа должна произвести вывод команды заменяя знаки табуляции и space на знаки «\_\_\_» и «\_» соответственно.

1. **Описание программы**

На глобальном уровне программы объявляются тре переменные: *BuffSize* – для задания буфера команды, *CharSize* – размер одного символа и строка FILE\_NAME[] - случайное название файла.

После запуска создается файл под названием указанным в, *FILE\_NAME[]*, который отображается в виртуальную память. Кроме того, в виртуальной памяти создается переменная *crappyMutex*, которая будет служить синхронизацией между родительским и детским процессом (с терминалом). После чего происходит *fork()* – создание детского процесса. Родительский процесс получает на ввод строку, которую преобразует в формат для команды консоли, в то время как детский процесс ожидает, когда родительский процесс заменит значение *crappyMutex* с 1 на 0 (таким образом синхронизируя процессы), после чего получает команду из виртуальной памяти, копирует файловый дескриптор для вывода в файл с случайным названием (с помощью *dup2*) и заменяет себя на процесс терминала (с помощью execv), который возвращает результат команды. Этот результат читается в родительском процессе из виртуальной памяти, после чего он преобразуется в соответствии с заданием.

1. **Набор testcases**

*ShortnessTest.txt* содержит три пробела и один TAB

*test.txt*

|  |
| --- |
| This is space delimited.  This is tab delimited. |

1. **Результаты выполнения тестов.**

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ make

cc ReformerMapped.c -o ReformerMapped

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ ./ReformerMapped

cat ShortnessTest.txt

\_\_\_

\_\_\_

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ hexdump -bc ShortnessTest.txt

0000000 040 040 040 012 011 012

0000000 \n \t \n

0000006

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ ./ReformerMapped

cat test.txt

This\_is\_space\_delimited.

This\_\_\_is\_\_\_tab\_\_\_delimited.

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ hexdump -bc test.txt

0000000 124 150 151 163 040 151 163 040 163 160 141 143 145 040 144 145

0000000 T h i s i s s p a c e d e

0000010 154 151 155 151 164 145 144 056 012 124 150 151 163 011 151 163

0000010 l i m i t e d . \n T h i s \t i s

0000020 011 164 141 142 011 144 145 154 151 155 151 164 145 144 056 012

0000020 \t t a b \t d e l i m i t e d . \n

0000030

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$ ./ReformerMapped

wc test.txt

\_2\_\_8\_48\_test.txt

rookstar@Refrigerator:~/Git/os-labs/os\_lab\_04/src$

1. **Листинг программы**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/mman.h>

#include <string.h>

#include <fcntl.h>

#ifndef PAGE\_SIZE

#define PAGE\_SIZE 4096

#endif

const size\_t BUFF\_SIZE = 100;

const size\_t CharSize = sizeof(char);

const char FILE\_NAME[] = "c943a334af5f2cc790749b07e3a85733";

int main()

{

int fd;

if ( (fd = open(FILE\_NAME, O\_CREAT | O\_RDWR)) < 0)

{

printf("Failed to create temporary file(\"%s\")\n", FILE\_NAME);

exit(EXIT\_FAILURE);

}

if (posix\_fallocate(fd, 0, PAGE\_SIZE) < 0)

{

printf("File allocation failure(\"%s\")\n", FILE\_NAME);

exit(EXIT\_FAILURE);

}

char \*mappedFile;

if ((mappedFile = mmap(NULL, PAGE\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0)) == MAP\_FAILED)

{

perror("Failed to map file into virtual memory\n");

remove(FILE\_NAME);

exit(EXIT\_FAILURE);

}

//Improvise. Adapt. Overcome.

int \*crappyMutex;

if ((crappyMutex = mmap(NULL, sizeof(int), PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, fd, 0)) == MAP\_FAILED)

{

perror("Failed to create mutex\n");

remove(FILE\_NAME);

exit(EXIT\_FAILURE);

}

\*crappyMutex = 1;

pid\_t pid = fork();

if (pid == -1)

{

printf("Failed to execute\n");

remove(FILE\_NAME);

exit(EXIT\_FAILURE);

}

if (pid == 0)

{

char com[BUFF\_SIZE];

int i = 0;

//Get the command from parrent

while (\*crappyMutex) { ;}

while (i != BUFF\_SIZE)

{

com[i] = mappedFile[i];

mappedFile[i] = '\0';

if (com[i] == '\0')

{

++i;

break;

}

++i;

}

dup2(fd, STDOUT\_FILENO);

dup2(fd, STDERR\_FILENO);

//Use the command and sent result to parent

char \*result[] = {"sh", "-c", com, NULL};

execv("/bin/sh", result);

}

else

{

char ComIn[BUFF\_SIZE];

int i = 0;

do {

scanf("%c", &ComIn[i]);

if (ComIn[i] == '\n')

{

++i;

break;

}

++i;

} while (i != BUFF\_SIZE);

//Write the command to child

for (int j = 0; j < i; ++j)

{ mappedFile[j] = ComIn[j]; }

mappedFile[i + 1] = '\0';

\*crappyMutex = 0;

//Wait for child

waitpid(pid, NULL, 0);

//Parsing process

for (i = 0; mappedFile[i] != '\0'; ++i)

{

if (mappedFile[i] == ' ') printf("\_");

else if (mappedFile[i] == '\t') printf("\_\_\_");

else printf("%c", mappedFile[i]);

}

}

remove(FILE\_NAME);

munmap(mappedFile, PAGE\_SIZE);

munmap(crappyMutex, sizeof(int));

return EXIT\_SUCCESS;

}

1. **Вывод**

Виртульная память является альтернативным способом обмена данными между процессами, а также дает возможность читать файл как строку (массив символов).