**Московский авиационный институт**
**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

# Лабораторная работа № 4

## Тема: Технология «File mapping»

Студент: Туманов Георгий
Группа: 80-201
Преподаватель: Соколов А.А.
Оценка:
Подпись:

Москва, 2019

1. **Постановка задачи**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 16: На вход программе подается команда интерпретатора команд и имя файла. Программа должна перенаправить стандартный ввод команды с этого файла и вывести результат команды в стандартный выходной поток. Использование операций write и printf запрещено.

2. **Описание программы**

Программа qwuk является вспомогательной и нужна для демонстрации основной программы. Она меняет регистр букв, а если встречает разделитель, выводит его ASCII-код. Остальные символы выводятся без изменений. Компилируется строкой: gcc qwuk.c -o qwuk

Основная программа читает две строки: команду интерпретатора и имя файла. Если вторая строка пустая, программа выполняет команду интерпретатора без изменений. Иначе, на стандартный ввод программе подаётся файл, имя которого написано во второй строке.

Процессы обмениваются через файл “pipe.txt”. Он создаётся при запуске программы и удаляется при завершении. Данный файл отображается в оперативную память и уже через эту область памяти процессы и связываются.

## 3. Набор testcases

| № | Описание | Ввод |
|---|---|---|
| 1 | Тест с двумя строками | ./qwuk<br>lines.txt |
| 2 | Тест с одной строкой | date |
| 3 | Тест с /dev/null | ./qwuk<br>/dev/null |

## 4. Результаты выполнения тестов.

*test 1*:
Write the name of programm:
./qwuk
Write the name of file:
lines.txt
./qwuk
lines.txt
WELCOME!
big letters SMALL LETTERS
aaBBccDD
oooo OOOOO PPPPpppp

*test 2*:
Write the name of programm:
date
Write the name of file:

date

Сб дек 14 09:19:59 MSK 2019

*test 3*:
Write the name of programm:
./qwuk
Write the name of file:
/dev/null

./qwuk
/dev/null
WELCOME!

## 5. Листинг программы

*main.c:*

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

const char FNAME[] = "pipe.txt";
const size_t LEN = 80;

int main()
{
	//open temp file
	int fd = open(FNAME, O_RDWR | O_CREAT | O_TRUNC);
	if (fd < 0) //error
	{
	printf("ERROR: cannot create file %s\n", FNAME);
	remove(FNAME); return -1;
	}
	//set size
	if (lseek(fd, LEN * 2 - 1, SEEK_SET) == -1)
	{
	printf("ERROR: cannot set size file %s\n", FNAME);
	close(fd); remove(FNAME); return -2;
	}
	if (write(fd, "", 1) != 1)
	{
	printf("ERROR: cannot set size file %s\n", FNAME);
	close(fd); remove(FNAME); return -3;
	}
	//map file
	void *addr;
	if ((addr = mmap(NULL, LEN * 2, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0)) ==
MAP_FAILED)
	{
	printf("ERROR: cannot map file %s\n", FNAME);
	close(fd); remove(FNAME); return -4;
	}
	//UI
	printf("Write the name of programm:\n");
	read(STDIN_FILENO, addr, LEN);

	printf("Write the name of file:\n");
```

```
read(STDIN_FILENO, addr + LEN, LEN);

//fork
pid_t pid = fork();
if (pid < 0)
{
printf("ERROR: cannot fork\n");
close(fd); remove(FNAME); return -5;
}
else if (pid > 0) //parent
{
int s;
waitpid(pid, &s, 0); //wait untill child is end
close(fd); remove(FNAME); //close everything
}
else //child
{
char prog_name[LEN], input_name[LEN];
//read program name
for (int i = 0; i < LEN; i++)
{
prog_name[i] = ((char*)addr)[i];
if (prog_name[i] == '\n') { prog_name[i] = '\0'; break; }
}
//read file name
for (int i = 0; i < LEN; i++)
{
input_name[i] = ((char*)addr)[i + LEN];
if (input_name[i] == '\n') { input_name[i] = '\0'; break; }
}
printf("%s\n%s\n", prog_name, input_name);

if (input_name[0] != '\0') //if has second argument, read from file
{
//open file
int inp = open(input_name, O_RDONLY);
if (inp < 0)
{
printf("ERROR: cannot open file %s\n", input_name);
return -6;
}
//redirect input
if (dup2(inp, STDIN_FILENO) == -1)
if (inp < 0)
{
printf("ERROR: cannot open file %s\n", input_name);
close(inp); return -6;
}
} //else read from console

//execute program
```

```
        execlp(prog_name, prog_name, NULL);
        }

        return 0;
}
```

*qwuk.c:*

```
#include <stdio.h>

char swap(char c)
{
        if (c >= 'a' && c <= 'z') putchar(c + 'A' - 'a');
        else if (c >= 'A' && c <= 'Z') putchar(c + 'a' - 'A');
        else if (c != ' ' && c != '\n' && c != '\t') printf("\nchar:%i\n", (int)c);
        else putchar(c);
}

int main(int argc, char *argv[])
{
        printf("WELCOME!\n");
        char c;
        while ((c = getchar()) != EOF) swap(c);
        putchar('\n');
        return 0;
}
```

## 6. Выводы:

Освоил технологию “File mapping” для реализации обмена данных в многопроцессорных программах.