Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работ №2 по курсу

«Операционные системы»

Группа: М80 – 207Б-18
Студент: Тояков Артем Олегович
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____

Москва, 2019.

Содержание

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

Постановка задачи.

Родительский процесс считывает стандартной входной поток, отдает его дочернему процессу, который удаляет "задвоенные" пробелы и выводит его в файл (имя файла также передается от родительского процесса).

Общие сведения о программе

Программа компилируется из одного файла main.c. В программе используются следующие системные вызовы:

1. **pipe** – для создания однонаправленного канала, через который могут общаться два процесса. Системный вызов возвращает два дескриптора файлов. Один для чтения из канала, другой для записи в канал.
2. **fork** – для создания дочернего процесса.
3. **close** – для закрытия файла.
4. **wait** – для ожидания завершения дочернего процесса.
5. **dup** -вызов использующийся для возврата нового дескриптора файла.
6. **dup2** – вызов для замены дескриптора нового и старого файла.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Используя системный вызов pipe создать 2 каналов, по которым будут обмениваться данными процессы.

2. Используя системный вызов fork создать дочерний процесс.

3. В родительском процессе перенаправить входной поток в дочерний процесс с помощью dup2 и оставить его ожидать завершения дочернего процесса.

4. В дочернем процессе необходимо получать данные из пайпа, затем удалять все задвоенные пробелы и записывать результат в файл.

Основные файлы программы.

Файл main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>

const int min_capacity = 2;
const int k = 2;

typedef struct{
    int *array;
    int length;
    int capacity;
} TVector;

TVector * Create_Vector(){
        TVector * v = (TVector*) malloc(sizeof(TVector));
        v->array = (int*) malloc(sizeof(int) * min_capacity);
        v->length = 0;
        v->capacity = min_capacity;
        return v;
}

int Get_Vector(TVector * v, int index){
        if (index < v->length && index >= 0)
            return v->array[index];
        else
            return 0;
}
```

```
void Resize_Vector(TVector * v, int new_capacity){
        v->array = (int*)realloc(v->array, sizeof(int) * new_capacity);
        v->capacity = new_capacity;
}

void Push_back_Vector(TVector * v, int value){
        if (v->length == v->capacity){
            Resize_Vector(v, v->capacity * k);
        }
        v->array[v->length] = value;
        v->length++;
}

void Remove_Vector(TVector * v){
        v->length--;
        if (v->length <= v->capacity / 2){
            Resize_Vector(v, v->capacity / k);
        }
}

void Destroy_Vector(TVector * v){
        free(v);
}

void Print_Vector(TVector * v){
        for (int i = 0; i < v->length; i++) {
                printf("%d ",v->array[i]);
        }
        printf("\n");
}

int main() {
    TVector * str = Create_Vector();
    char* filename = (char*) malloc(sizeof(char) * 2048);
    int status;
    pid_t p;
    int fd1[2];
    int fd2[2];
    if (pipe(fd1)<0) {
        printf("Невозможно создать pipe \n");
        exit(2);
    }
    if (pipe(fd2)<0) {
        printf("Невозможно создать pipe \n");
        exit(2);
    }
    p = fork();
    if (p < 0) {
        printf("Невозможно создать дочерний процесс \n");
        exit(3);
    } else if (p == 0) {
        close(fd1[0]);
        close(fd2[0]);
        char c;
        c = getchar();
        while (c != '\n') {
            strcat(filename, &c);
```

```
            c = getchar();
        }
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP|  S_IROTH | S_IWOTH; /* 0666 */
        int newfd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, mode);
        close(1);
        fd2[1] = newfd;
        if(dup(fd2[1]) == -1) {
            printf("Can't do dup");
            exit(1);
        }
        c = getchar();
        while (c != EOF) {
            Push_back_Vector(str, c);
            c = getchar();
        }
        bool lspace = false;
        for (size_t i = 0; Get_Vector(str,i) != '\0'; i++) {
            if (Get_Vector(str, i) == ' ' && lspace) {
                continue;
            }
            printf("%c", Get_Vector(str,i));
            if (Get_Vector(str,i) == ' ') {
                lspace = true;
            } else {
                lspace = false;
            }
        }
        close(fd1[1]);
        close(fd2[1]);
        //printf("Выход из дочернего процесса \n");
        exit(0);
    } else {
        close(fd1[1]);
        close(fd2[1]);
        if(dup2(fd1[0], STDIN_FILENO) == -1) {
            printf("Can't do dup2");
            exit(5);
        }
        close(0);
        wait(&status);
        close(fd1[0]);
        close(fd2[0]);
        Destroy_Vector(str);
        free(filename);
        //printf("Выход из родительского процесса \n");
    }
}
```

Демонстрация работы программы.

```
artoy@artoy:~/Desktop/Labs/OS/Lab2$ gcc main.c
artoy@artoy:~/Desktop/Labs/OS/Lab2$ ./a.out
file.txt
i  love informatics
and  its     very cool
```

File1.txt:

```
i love informatics
and its very cool
```

Вывод

Межпроцессорное взаимодействие можно осуществлять с помощью канала. В СИ канал создается с помощью системного вызова pipe. На мой взгляд, такой способ общения процессов очень удобен, так как при данном подходе не приходится сталкиваться с гонками, так как при использовании блокирующих системных вызовов read и write процессы блокируются, если

им нечего считывать или буфер для записи полный. Так же одним из плюсов такого подхода к межпроцессорному взаимодействию является то, что каналом могут пользоваться только родственные процессы, так как канал находится в пределах ядра.