Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»
Дисциплина «Операционные системы»

Лабораторная работа №2

Тема: Управление процессами в ОС

Студент: Федоров А. С.
Группа: М8О-207Б-19
Преподаватель: Миронов Е. С.
Дата:
Оценка:

Москва, 2020

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 каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Группа вариантов 1

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

Вариант 2

Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип float.

Алгоритм решения задачи

Родительский процесс создает pipe и считывает имя файла для записи. Затем создает дочерний процесс, переопределяет для него стандартный поток ввода и запускает программу child с заменой образа памяти с помошью execv. Все введенные слагаемые для суммирования передаются в пайп из родительского процесса.

Дочерний процесс считывает слагаемые из дескриптора pipe для чтения и записывает сумму в файл, имя которого было передано как аргумент argv[0]. После подсчета суммы программа закрывает файл и дискриптор для чтения.

Родительский процесс после завершения чтения закрывает дискриптор для записи и завершается.

Листниг программы

main.c

```
#include <stdio.h>
#include <unistd.h>
```

```c
int main(){
    printf("enter file name:");
    char filename[256];
    scanf("%s", filename);
    int fd[2];
    if (pipe(fd) == -1){
        printf("pipe error\n");
        return -1;
    }
    int id = fork();
    if (id == -1){
        printf("fork error\n");
        return -1;
    }
    else if (id == 0){
        close(fd[1]);
        if (dup2(fd[0], fileno(stdin)) == -1){
            printf("dup2 error\n");
            return -1;
        }
        char* argv[3] = {"child", filename, (char *)NULL};
        if (execv("child", argv) == -1){
            printf("execl error\n");
        }
    }
    else{
        close(fd[0]);
        printf("insert sum terms:");
        float currentTerm;
        while(scanf("%f", &currentTerm) > 0){
            if (write(fd[1], &currentTerm, sizeof(float)) == 0){
                printf("write error\n");
                return -1;
            }
        }
        close(fd[1]);
    }
    return 0;
}
```

child.c

```c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char* argv[]){
    FILE *file = fopen(argv[1], "w");
    if (file == NULL){
        printf("fopen error\n");
        return -1;
    }
    float result = 0;
    float currentTerm;
    while (read(fileno(stdin), &currentTerm, sizeof(float)) > 0){
        result += currentTerm;
    }
    fprintf(file, "%f", result);
    fclose(file);
    return 0;
}
```

Список литературы

1. Таненбаум Э., Бос Х. *Современные операционные системы. — 4-е изд.* — СПб.: Издательский дом «Питер», 2018. — С. 111 – 123.

Тесты и протокол исполнения

Тест №1

```
protaxy@protaxY:~/CLionProjects/OS_lab2$ ./main
enter file name:a.txt
insert sum terms:1.0 2.0 3.0
protaxy@protaxY:~/CLionProjects/OS_lab2$ cat a.txt
6.000000
```

Тест №2

```
protaxy@protaxY:~/CLionProjects/OS_lab2$ cat a.txt
6.000000protaxy@protaxY:~/CLionProjects/OS_lab2$ ./main
enter file name:a.txt
insert sum terms:
protaxy@protaxY:~/CLionProjects/OS_lab2$ cat a.txt
0.000000
```

Тест №3

```
protaxy@protaxY:~/CLionProjects/OS_lab2$ ./main
enter file name:a.txt
insert sum terms:2.2434 1.213 3
protaxy@protaxY:~/CLionProjects/OS_lab2$ cat a.txt
6.456400
```

Вывод

В ходе выполнения лабораторной работы я приобрел навыки работы с каналом (pipe), научился копировать дескрипторы с помощью системного вызова dup2. Также изучил основы управления процессами в ОС Unix. Изучил работу системных вызовов fork и execv для обработки процессов.

Список литературы

1. Таненбаум Э., Бос Х. *Современные операционные системы. — 4-е изд.* — СПб.: Издательский дом «Питер», 2018. — С. 111 – 123.