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: Инютин М. А.
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.

Вариант 6.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

В файле записаны команды вида: «число число число <endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int.

Алгоритм решения задачи.

В основном процессе до создания дочернего считаем имя файла и попробуем переопределить поток вывода. В случае успеха создадим канал для связи родительского и дочернего процесса. Так как в задании требуется представить дочерний процесс в отдельном файле, то используем execl для его запуска. В самом процессе будем считывать числа из файла, а писать результат вычислений в канал. Родительский процесс будет читать из канала и выводить на экран значение суммы.

Листинг программы

Файл с основным процессом - main.c, файл с дочерним - child.c

main.c

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

/*
 * main.c
 */

// Ubuntu has 255 symbol filename limit
const unsigned long long FILENAME_LIMIT = 255;

signed main() {
	char* s = NULL;
	s = malloc(sizeof(char) * (FILENAME_LIMIT + 1));
	if (s == NULL) {
		printf("Error allocating memory!");
		return 1;
	}
	for (int i = 0; i < FILENAME_LIMIT + 1; i++) {
		s[i] = 0;
	}
	scanf("%s", s);
	FILE * input = NULL;
	input = freopen(s, "r", stdin);
	if (input == NULL) {
		printf("Error opening file!\n");
		return 2;
	}
	int fd[2];
	if (pipe(fd) == -1) {
		printf("Error creating pipe!");
		return 3;
	}
	int id = fork();
	if (id == -1) {
		printf("Error creating process!");
		return 4;
	} else if (id == 0) {
		close(fd[0]);
		if (dup2(fd[1], STDOUT_FILENO) == -1) {
			printf("Error changing stdout!\n");
			return 5;
		}
		char * argv = NULL;
		if (execl("child.out", argv) == -1) {
			printf("Error executing child process!\n");
```

```
                return 6;
            }
        } else {
            close(fd[1]);
            int res;
            while (read(fd[0], &res, sizeof(int)) > 0) {
                printf("%d\n", res);
            }
            close(fd[0]);
        }
        return 0;
}
```

child.c

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

/*
 * main.c
 */

signed main() {
        int num = 0, sum = 0, minus = 0;
        char c;
        while (scanf("%c", &c) > 0) {
            if (c == ' ' || c == '\t') {
                sum = minus ? sum - num : sum + num;
                num = 0;
                minus = 0;
            } else if (c == '-') {
                minus = 1;
            } else if (c == '\n') {
                sum = minus ? sum - num : sum + num;
                num = 0;
                minus = 0;
                write(STDOUT_FILENO, &sum, sizeof(int));
                sum = 0;
            } else if ('0' <= c && c <= '9') {
                num = num * 10 + c - '0';
            }
        }
        return 0;
}
```

Тесты и протокол исполнения

Заранее подготовим файл input.txt, из которого будет читать дочерний процесс.

input.txt

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
-1 -2 -3 -4 -5 1 2 3 4 5
1
2147483647
-2147483648
1 2 3 4 5 6 7 8 9 10
```

Вывод программы

```
./main.out
input.txt
210
0
1
2147483647
-2147483648
55
```

Выводы

Я изучил работу процессов и каналов в ОС Linux (fork и pipe), составил и отладил программу на языке С. Во время выполнения работы я столкнулся с проблемой взаимодействия процессов, с чем мне помогли системный вызовы freopen и dup2, которые перенаправляют потоки ввода и вывода для программ, даже если они были запущены командой execv. Некоторые задачи (например, множественные ответы на запросы) можно разделить на ввод, вычисление и вывод. Так разделение программы на несколько процессов уменьшит время выполнения в целом.

Список литературы

1. Изучаем процессы в Linux / Хабр — Habr
   URL: https://habr.com/ru/post/423049/ (дата обращения: 21.09.2020)
2. Таненбаум Э., Бос Х. *Современные операционные системы. — 4-е изд.* — СПб.: Издательский дом «Питер», 2018. — С. 111 - 123