Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-210Б-23

Студент: Абдыкалыков Н. А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 24.10.24

Москва, 2024

Постановка задачи

Вариант 1:

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 int. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

pipe(int pipefd[2]); — создает неименованный канал для межпроцессного взаимодействия, возвращает два файловых дескриптора для чтения и записи.
fork(); — создает дочерний процесс, который является копией родительского, возвращает PID дочернего процесса в родительском и 0 в дочернем процессе.
execlp(const char *file, const char *arg, ...); — замещает текущий процесс программой, указанной в аргументе file.
dup2(int oldfd, int newfd); — дублирует файловый дескриптор oldfd в newfd, перенаправляя потоки.
read(int fd, void *buf, size_t count); — считывает до count байт данных из файлового дескриптора fd в буфер buf.
write(int fd, const void *buf, size_t count); — записывает до count байт данных из буфера buf в файловый дескриптор fd.
wait(int *status); — заставляет родительский процесс ожидать завершения дочернего, возвращая его статус завершения.
open(const char *pathname, int flags, mode_t mode); — открывает файл по пути pathname с указанными флагами и правами доступа.
close(int fd); — закрывает файловый дескриптор fd, освобождая его ресурсы.
snprintf(char *str, size_t size, const char *format, ...); — записывает отформатированную строку в буфер str размером size.

Реализация: идея и подход

Идея реализации программы заключается в создании двух процессов, где родительский процесс отвечает за ввод данных и их передачу, а дочерний процесс выполняет обработку и вывод результатов. Такой подход позволяет продемонстрировать основные концепции работы с процессами и межпроцессным взаимодействием в Unix-подобных операционных системах.

Код программы

Файл parent.c

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

#define BUFFER_SIZE 2048

int main() {
    int pipe1[2];
    pid_t pid;
    char buffer[BUFFER_SIZE];
    char filename[100];
    ssize_t bytes_read, bytes_written;

    // Создание pipe1
    if (pipe(pipe1) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    // Сообщение пользователю для ввода имени файла
    const char *msg = "Введите имя файла: ";
    bytes_written = write(STDOUT_FILENO, msg, strlen(msg));
    if (bytes_written == -1) {
        perror("write");
        exit(EXIT_FAILURE);
    }

    // Чтение имени файла с консоли
    bytes_read = read(STDIN_FILENO, filename, sizeof(filename) - 1);
    if (bytes_read == -1) {
        perror("read");
        exit(EXIT_FAILURE);
    }

    // Удаление символа новой строки, если он был введен
    if (bytes_read > 0 && filename[bytes_read - 1] == '\n') {
        filename[bytes_read - 1] = '\0';
    }

    // Создание дочернего процесса
    pid = fork();

    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }
```

```
    if (pid == 0) {
        // Дочерний процесс
        close(pipe1[1]);  // Закрываем запись в pipe для дочернего
процесса

        // Перенаправление стандартного ввода на pipe
        dup2(pipe1[0], STDIN_FILENO);
        close(pipe1[0]);

        // Запуск дочерней программы
        execlp("./child", "child", filename, NULL);
        perror("execlp");
        exit(EXIT_FAILURE);
    } else {
        // Родительский процесс
        close(pipe1[0]);  // Закрываем чтение из pipe для
родительского процесса

        // Сообщение для ввода чисел
        const char *msg2 = "Введите три числа через пробел: ";
        write(STDOUT_FILENO, msg2, strlen(msg2));

        // Чтение чисел с консоли
        bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer));
        if (bytes_read == -1) {
            perror("read");
            exit(EXIT_FAILURE);
        }

        // Отправляем данные в pipe
        write(pipe1[1], buffer, bytes_read);
        close(pipe1[1]);  // Закрываем запись после отправки данных

        // Ожидание завершения дочернего процесса
        wait(NULL);
    }

    return 0;
}
```

Файл child.c

```
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>

#define BUFFER_SIZE 2048
```

```c
void convertStringToIntArray(const char *str, int intArray[], int
*size) {
    int i = 0, num = 0;
    *size = 0;
    while (str[i] != '\n' && str[i] != '\0') {
        // Пропускаем пробелы
        while (isspace(str[i])) {
            i++;
        }

        // Считываем число
        if (isdigit(str[i])) {
            num = 0;
            while (isdigit(str[i])) {
                num = num * 10 + (str[i] - '0');
                i++;
            }
            intArray[*size] = num;
            (*size)++;
        }
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        const char msg[] = "Usage: <program> <filename>\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        exit(EXIT_FAILURE);
    }

    // Открываем файл для записи результата
    char *filename = argv[1];
    int file = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (file == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    char buffer[BUFFER_SIZE];
    ssize_t bytes_read;

    // Чтение данных из родительского процесса через pipe
    bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer) - 1);
    if (bytes_read <= 0) {
        const char msg[] = "error: failed to read from pipe\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        close(file);
        exit(EXIT_FAILURE);
    }

    buffer[bytes_read] = '\0';  // Завершаем строку

    int array[BUFFER_SIZE];
```

```
        int size = 0;
        int sum = 1;

        // Конвертируем строку в массив чисел
        convertStringToIntArray(buffer, array, &size);

        // Вычисляем сумму
        for (int i = 0; i < size; ++i) {
            sum *= array[i];
        }

        // Записываем результат в файл
        char output[128];
        int len = snprintf(output, sizeof(output), "Сумма: %d\n", sum);
        write(file, output, len);

        close(file);
        return 0;
    }
```

Протокол работы программы

Работа программы и её вывод:

```
naabdykalykov@DESKTOP-ETRASM1:/mnt/c/Users/User/Desktop/osLabs$ gcc parent.c -o parent
naabdykalykov@DESKTOP-ETRASM1:/mnt/c/Users/User/Desktop/osLabs$ gcc child.c -o child
naabdykalykov@DESKTOP-ETRASM1:/mnt/c/Users/User/Desktop/osLabs$ ./parent
Введите имя файла: result.txt
Введите числа через пробел: 3 8 1 44 3
naabdykalykov@DESKTOP-ETRASM1:/mnt/c/Users/User/Desktop/osLabs$ cat result.txt
Сумма: 59
```

Strace:

execve("./parent", ["./parent"], 0x7ffff1de61e0 /* 15 vars */) = 0

brk(NULL) = 0x7ffff2f0a000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd8f0310000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=16626, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 16626, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd8f035a000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1922136, ...}, AT_EMPTY_PATH) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 1970000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd8f0120000

mmap(0x7fd8f0146000, 1396736, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7fd8f0146000

mmap(0x7fd8f029b000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17b000) = 0x7fd8f029b000

mmap(0x7fd8f02ee000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ce000) = 0x7fd8f02ee000

mmap(0x7fd8f02f4000, 53072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd8f02f4000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd8f0110000

arch_prctl(ARCH_SET_FS, 0x7fd8f0110740) = 0

set_tid_address(0x7fd8f0110a10) = 870

set_robust_list(0x7fd8f0110a20, 24) = 0

rseq(0x7fd8f0111060, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)

mprotect(0x7fd8f02ee000, 16384, PROT_READ) = 0

mprotect(0x7fd8f0362000, 4096, PROT_READ) = 0

mprotect(0x7fd8f0350000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0

munmap(0x7fd8f035a000, 16626) = 0

pipe2([3, 4], 0) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 34) = 34

read(0, "result.txt\n", 99) = 11

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fd8f0110a10) = 871

close(3) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\260 \321\207\320\265\321\200"..., 51) = 51

read(0, "3 8 1 44 3\n", 2048) = 11

write(4, "3 8 1 44 3\n", 11) = 11

close(4) = 0

wait4(-1, NULL, 0, NULL) = 871

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=871, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

exit_group(0)   = ?

+++ exited with 0 +++

Вывод

В ходе выполнения лабораторной работы была разработана программа, состоящая из двух процессов: родительского и дочернего. Родительский процесс отвечает за взаимодействие с пользователем, получая от него имя файла и числа, которые необходимо обработать. Дочерний процесс выполняет вычисления, а именно, подсчитывает сумму переданных чисел и записывает результат в указанный файл.