Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра теоретической и прикладной информатики

Лабораторная работа №2

по дисциплине «Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМ-92

Вариант: 8

Студенты: Иванов В., Кутузов И.

Преподаватели: Стасышин В.М., Сивак М.А.

Новосибирск

2022

1. Цель работы

Изучить программные средства создания процессов, получить навыки управления и синхронизации процессов, а также простейшие способы обмена данными между процессами. Ознакомиться со средствами динамического запуска программ в рамках порожденного процесса, изучить механизм сигналов ОС UNIX, позволяющий процессам реагировать на различные события, и каналы, как одно из средств обмена информацией между процессами.

2. Задание

Разработать программу, n раз вычисляющую значение функции ln(x) для случайного действительного числа x путем разложения в ряд, и выводящую полученные значения в файл. В это время предварительно подготовленный процесс-потомок читает данные из файла и выводит на экран до тех пор, пока процесс-предок не передаст ему через файл ключевое слово (например, "STOP"), свидетельствующее об окончании работы. Количество вычислений n генерировать случайным образом (в разумных пределах).

3. Описание используемых структур

int fork()
Порождение процесса-потомка, точной копии процесса-предка. Процесс-потомок в качестве возвращаемого значения системного вызова fork() получает 0, а процесс-предок - идентификатор процесса-потомка.

void sleep(unsigned time)
Приостанавливает выполнение процесса на время, задаваемое параметром time. Время задается в секундах.

int strcmp(const char *s1, const char *s2)
Сравнивает две строки: s1 и s2. Она возвращает целое число, которое меньше, больше нуля или равно ему, если s1 соответственно меньше, больше или равно s2.

char *fgets(char *str, int num, FILE *stream)
Считывает до num-1 символов из файла stream и помещает их в массив символов, на который указывает str. Символы считываются до тех пор, пока не встретится символ «новая строка», EOF или до достижения указанного предела. По окончании считывания в массив str сразу после последнего считанного символа помещается нулевой символ. Символ «новая строка» при считывании будет сохранен и станет частью массива str.

void exit(int value)
Выполняет немедленное завершение программы. Аргумент параметра value возвращается принимающей стороной (ОС или другой программой) в родительский процесс. Как правило, возвращается значение 0 или EXIT_SUCCESS указывает на успешное завершение программы, и любое другое значение или значение макроса EXIT_FAILURE используется для указания об аварийном завершении программы.

int rand(void)
Генерирует случайные числа, возвращает псевдослучайное целое число в диапазоне от 0 до RAND_MAX.

void srand(unsigned int seed)
Выполняет инициализацию генератора случайных чисел rand().

4. Спецификация

Программа разработана и протестирована на компьютере с операционной системой Linux. В качестве компилятора используется GCC версии 11.2.0.
Рабочая директория: **/home/lenferdetroud/Repos/university-tasks/Unix Resource Management/Unix Process Control**
Название файла с программой: **unix_process_control.c**
Название файла для вывода: **result**
Компиляция программы: **gcc -o <результат компиляции> unix_process_control.c**
Запуск программы: **./<результат компиляции>**

5. Описание алгоритма

1. Генерируем случайные числа n и x; вызываем функцию, n раз вычисляющую значение натурального логарифма от x
2. Создаем процесс, который начинает считывать и выводить на экран строки из result до тех пор, пока не будет считана строка “STOP”
3. Проверяем, не выходит ли x за границы области определения натурального логарифма
4. Открываем родительским процессом файл result; вычисляем и записываем в файл значение ряда Меркатора n раз:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} x^n$$

5. Записываем строку “STOP” в result родительским процессом

6. Код программы

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

```c
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

void compute_ln(double x, int n) {
    unsigned step_count;
    FILE *result;

    if (fork() == 0) создание процесса
    {
        char str[50];
        char *estr;
        const char *stop = "STOP"; ключевое слово для завершения процесса
        result = fopen("result", "r"); открытие файла для чтения дочерним процессом

        sleep(0.1); приостановка процесса для разрешения ситуации, когда
процесс-потомок запускается первым и считывает пустой файл

        while (1)
        {
            estr = fgets(str, sizeof(str), result); считывание строки из файла
            int v = strcmp(estr, stop); бинарное сравнение строки с ключевым словом

            if (v == 0) выход из цикла в случае считывания ключевого слова
            {
                printf("Exit keyword passed.\n");
                break;
            }
            printf("%s", str);
        }

        fclose(result); завершение чтения файла
        exit(0); завершение процесса
    }

    result = fopen("result", "wa"); открытие файла для записи родительским процессом

    if (x <= 0.0) проверка области определения натурального логарифма
    {
        printf("Error: The argument must be greater than 0!\n");
        exit(0);
    }

    x = x - 1.0;

    for (int i = 0; i < n; ++i) цикл, вычисляющий ряд Меркатора n раз
    {
        double sum = 0.0, m = x, nom = x;
        step_count = 1;

        while (fabs(m) > 1e-11) точность вычислений
        {
            sum += m;
            nom *= - x;
            m = nom / ++step_count;
        };

        fprintf(result, "ln(%.3lg) = %f\n", x + 1.0, sum); вывод результата
    }
```

```
    fprintf(result, "STOP"); запись ключевого слова в конец файла
    fclose(result); завершение работы с файлом
}

int main() {
    srand(time(NULL)); установка текущего времени в качестве базы генератора
псевдослучайных чисел
    int n = 1000 + rand() % 1000; генерация случайного числа от 1000 до 1999
    double x = (double)rand() / RAND_MAX*2.0; генерация случайного числа от 0.0 до 2.0

    compute_ln(x, n);

    return 0;
}
```

7. Тестирование

Результат выполнения команды `gcc -o a.out unix_process_control.c ; ./a.out`:

```
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
ln(0.83) = -0.186315
Exit keyword passed.
 ~/Repos/university-tasks/Unix
```

Каждая строка считывалась и выводилась дочерним процессом сразу после записи в файл родительским процессом.

Содержимое файла `result`:

result | unix_process_control.c

```
1  ln(0.83) = -0.186315
2  ln(0.83) = -0.186315
3  ln(0.83) = -0.186315
4  ln(0.83) = -0.186315
5  ln(0.83) = -0.186315
6  ln(0.83) = -0.186315
7  ln(0.83) = -0.186315
8  ln(0.83) = -0.186315
9  ln(0.83) = -0.186315
10 ln(0.83) = -0.186315
```

```
1756    ln(0.83) = -0.186315
1757    ln(0.83) = -0.186315
1758    ln(0.83) = -0.186315
1759    ln(0.83) = -0.186315
1760    ln(0.83) = -0.186315
1761    ln(0.83) = -0.186315
1762    STOP
```