Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работ №2 по курсу
«Операционные системы»

Управление процессами в ОС

Студент: Вахрамян Кирилл Олегович
Группа: М8О-206Б-18
Вариант: 25
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Содержание

Постановка задачи.

Родительский процесс отвечает за ввод и вывод. Родительский процесс отвечает за поиск образца в строке.

Общие сведения о программе

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

1. **read** – для чтения данных из pipe.
2. **write** – для записи данных в pipe.
3. **pipe** – для создания однонаправленного канала, через который из дочернего процесса данные передаются родительскому. Возвращает два дескриптора файлов: для чтения и для записи.
4. **fork** – для создания дочернего процесса.
5. **close** – для закрытия pipe после окончания считывания результата выполнения команды.

Общий метод и алгоритм решения.

Системный вызов pipe() создает 3 канала fd1, fd2, fd3 для передачи данных между процессами. С помощью fork() создается дочерний процесс. В родительском процессе считывается образец и строка при помощи системного вызова read(). По каналам fd1[1] и fd2[1] write() передает образец и строку в дочерний процесс. Дочерний процесс читает данные из родительского при помощи системного вызова read() по каналам fd1[0] и fd2[0] соответственно. Далее в дочернем процессе происходит вызов функции search(), в которой реализован простой алгоритм поиска подстроки в строке, работающий за O((n — m) * m), где n — длина строки, m — длина образца. Дочерний процесс возвращает в родительский процесс по каналу fd3[]1] при помощи системного вызова write() строку «Yes» , если образец найден, и «No» в противном случае. Родительский процесс читает строку по каналу fd3[0] и выводит ее на стандартный поток вывода.

Код программы.

main.c

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

// поиск образца в строке
bool search(char* str1, char* str2, int a, int b)
{
	int flag = 0, count = 0;
	if (a > b) {
		return false;
	}
	for (int i = 0; i < b - a + 1; i++) {
		if (str1[0] == str2[i]) {
			count++;
			for (int j = 0; j < a - 1; j++) {
				if (str1[j] == str2[i + j]) {
					flag = 1;
				} else {
					flag = 0;
					break;
				}
			} if (flag == 1) {
				break;
			}
		}
	}
	if (flag == 1 && count > 0) {
		return true;
	}
	return false;
}

int main()
{
	int fd1[2], fd2[2], fd3[2];
	char ans[10];
```

```
      if (pipe(fd1)==-1)
{
   perror("pipe error\n");
   exit(1);
}
if (pipe(fd2)==-1)
{
   perror("pipe error\n");
   exit(1);
}
   if (pipe(fd3)==-1)
{
  perror("pipe error\n");
  exit(1);
}


pid_t p = fork();
if (p < 0) {
      perror("fork error\n");
      exit(1);

} else if(p > 0) {
      char str1[100], str2[100];
      int size1, size2;
      // считывание образца
      size1 = read(0, str1, 100);
      // считывание строки
      size2 = read(0, str2, 100);
      close(fd1[0]);
      close(fd2[0]);
      // передача образца в дочерний процесс
      write(fd1[1], str1, size1);
      close(fd1[1]);
      // передача строки в дочерний процесс
      write(fd2[1], str2, size2);
      close(fd1[1]);
      wait(NULL); // ожидание дочернего процесса



      int size_ans;
      // считывание результата из дочернего процесса
      size_ans = read(fd3[0], ans, 10);
```

```
        write(1, ans, size_ans);

    } else {
        char child_str1[100], child_str2[100];
        int c_size1, c_size2;
        close(fd1[1]);
        close(fd2[1]);
        // считывание образаца из родительского процесса
        c_size1 = read(fd1[0], child_str1, 100);
        close(fd1[0]);
        // считывание строки из родительского процесса
        c_size2 = read(fd2[0], child_str2, 100);
        close(fd2[0]);

        // проверка наличия образаца в строке
        if (search(child_str1, child_str2, c_size1, c_size2)) {
                strcpy(ans, "Yes\n");
                write(fd3[1], ans, 4);
                close(fd3[1]);
        } else {
                strcpy(ans, "No\n");
                write(fd3[1], ans, 3);
                close(fd3[1]);
        }
        exit(0);
    }
}
```

Демонстрация работы программы.

```
courage@courage-X550LC:~/OS$ gcc main.c
courage@courage-X550LC:~/OS$ ./a.out
qwe
asdqweasd
Yes
```

Вывод

Я научился создавать процессы, используя системный вызов fork(), обрел навыки взаимодействия между процессами с помощью pipe(), получил новые знания о файловых дескрипторах. Также впервые реализовал поиск образца в строке.