

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

**Лабораторная работа №2 по курсу
«Операционные системы»**

Студент: Дегтярев Денис Андреевич
Группа: М8О-207Б-21
Вариант: 20
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

<https://github.com/CHISH08/OCI/tree/main/2lab>

Постановка задачи

Цель работы

Научиться создавать процессы и взаимодействовать с ними через pipe

Задание

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы инвертируют строки.

Общие сведения о программе

Программа компилируется из файла 2lab.cpp. Также подключаются файлы child1.cpp, child2.cpp через exelr в качестве отдельной программы. В родительский процесс подаем файлы file1.txt, file2.txt. В программе используются следующие системные вызовы:

1. pipe() – создает связь между памятью процессов
2. fork() – создает второй процесс
3. dup2() – копирует old_file_descriptor в new_file_descriptor.

Общий метод и алгоритм решения

Сначала создаем два pipe.

Затем создаем два процесса: 1 занимается обработкой pipe1, второй – pipe2; Родительский процесс занимается заполнением pipe1 и pipe2 перед их обработкой дочерними процессами.

В файлы pipe1.txt и pipe2.txt заносятся выходные данные.

Исходный код

2lab.cpp:

```
#include <bits/stdc++.h>
```

```
#include <sys/wait.h>
```

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

```
#include <fcntl.h>
```

```
#include <signal.h>
```

```
using namespace std;
```

```
void reverseStr(string &str)
```

```
{
```

```
    int n = str.length();
```

```
    for (int i = 0; i < n / 2; i++)
```

```
    {
```

```
        swap(str[i], str[n - i - 1]);
```

```
    }
```

```
}
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    setlocale(LC_ALL, "Russian");
```

```
    int fd1[2];
```

```
    int fd2[2];
```

```
    if (pipe(fd1) == -1)
```

```
    {
```

```
        cout << "An error occurred with opening the pipe1\n";
```

```
        return 1;
```

```
    }
```

```
    if (pipe(fd2) == -1)
```

```
    {
```

```
        cout << "An error occurred with opening the pipe2\n";
```

```
        return 2;
```

```
    }
```

```

int id = fork(), id2;
int flag = 0;
if (id > 0)
{
    id2 = fork();
    flag = 1;
}
if (id == -1 || id2 == -1)
{
    cout << "An error occurred with fork\n";
    return 5;
}
else if (id2 != 0 && id != 0)
{
    char *name_file1;
    char *name_file2;
    string line;
    // cin >> name_file1 >> name_file2;
    int file1 = open("./file1.txt", O_RDONLY);
    dup2(file1, STDIN_FILENO);
    if (file1 != -1)
    {
        while (getline(cin, line))
        {
            line = line + '\n';
            int lineSize = line.length();
            if (lineSize > 10)
            {
                if (write(fd2[1], line.c_str(), lineSize * sizeof(char)) == -1)
                {
                    cout << "An error occurred with writing to the pipe2\n";
                    return 3;
                }
            }
        }
        else
        {
            if (write(fd1[1], line.c_str(), lineSize * sizeof(char)) == -1)
            {

```

```

        cout << "An error occurred with writing to the pipe1\n";
        return 4;
    }
}
}
close(fd1[1]);
close(fd2[1]);
}
int file2 = open("./file2.txt", O_RDONLY);
dup2(file2, STDIN_FILENO);
if (file2 != -1)
{
    close(fd1[0]);
    close(fd2[0]);
    while (getline(cin, line))
    {
        int lineSize = line.length();
        if (lineSize > 10)
        {
            if (write(fd2[1], line.c_str(), lineSize * sizeof(char)) == -1)
            {
                cout << "An error occurred with writing to the pipe2\n";
                return 3;
            }
        }
        else
        {
            if (write(fd1[1], line.c_str(), lineSize * sizeof(char)) == -1)
            {
                cout << "An error occurred with writing to the pipe1\n";
                return 4;
            }
        }
    }
    close(fd1[1]);
    close(fd2[1]);
}
}

```

```

else if (flag)
{
    close(fd1[1]);
    close(fd1[1]);
    close(fd2[0]);
    dup2(fd1[0], STDIN_FILENO);
    execlp("./child1", "child1", NULL);
    close(fd1[0]);
}
else {
    close(fd2[1]);
    close(fd1[0]);
    close(fd1[1]);
    dup2(fd2[0], STDIN_FILENO);
    execlp("./child2", "child2", NULL);
    close(fd2[0]);
}
return 0;
}

child1:
#include<bits/stdc++.h>
#include<sys/wait.h>
#include<unistd.h>
using namespace std;

void reverseStr(string& str)
{
    int n = str.length();

    for (int i = 0; i < n / 2; i++) {
        swap(str[i], str[n - i - 1]);
    }
}

int main(int argc, char *argv[])
{
    string s;
    ofstream fout("./pipe1.txt", ios_base::out | ios_base::trunc);

```

```

while (getline(cin, s)) {
    reverseStr(s);
    fout << s + "\r\n";
}
fout.close();
return 0;
}

child2.cpp:
#include <bits/stdc++.h>
#include <sys/wait.h>
#include <unistd.h>
using namespace std;

void reverseStr(string &str)
{
    int n = str.length();

    for (int i = 0; i < n / 2; i++)
    {
        swap(str[i], str[n - i - 1]);
    }
}

int main(int argc, char *argv[])
{
    string s;
    ofstream fout("./pipe2.txt", ios_base::out | ios_base::trunc);
    while (getline(cin, s))
    {
        reverseStr(s);
        fout << s << endl;
    }
    fout.close();
    return 0;
}

```

Демонстрация работы программы


INPUT:

```
file1.txt x
1 g.jkfbfvshudfhjdbhfsj
2 gsjkalkdf
3 23434tgfg
4 432uhhdfjkgfgdgfsfdf
5 mksfmk;sgfkgfsggfskpgksksks
6 ksgs
7 dslmsml
8 awlfmwkjfjnjd
9 jnsdnkkn322krlfe;lql
10 dfpkmsvsdvkvs
11 w

file2.txt x
1 jsfhbdjhfdhdsvfbfkj.g
2 fdfsfgdgfgkjfdhhu234
3 sgskskgpksggsfgkfgs;kmfskm
4 djnjfjkwmflwa
5 lql;eflrk223nkkndsnj
6 svkvdsvsmpkd
7 gsfgkfgsfdklakjsg
8 mkfvmkmklf
9 erw'vpmb'grwpgb
10 bmgrebgrgbkgrwpgb
11 ]mrtobomrmrmbgror[rmwb
```

OUTPUT:

```
pipe1.txt
1 fdklakjsg
2 gfgt43432
3 sgsk
4 lmsmlsd
5 w
6 wf
7 rg
8 g
9 ewfb
10 brwkewfb
11 bgrk
```

```
Открыть ▾  pipe2.txt
~/home/OCI/2lab/src
1 jsfhbdjhfduhdsvfbfkj.g
2 fdfsfgdgfgkjfdhhu234
3 sgskskgpksfggsfgkfgs;kmfskm
4 djnjfjkwmflwa
5 lql;eflrk223nknksnj
6 svkvdsvsmkpfd
7 gsfgkfgsfdklakjsg
8 mkfvmkmklf
9 erw'vpmb'grwpgb
10 bmgrebgrgbkgrwpgb
11 ]mrtobomrmmbgror[rmwb
```

Выводы

С помощью `c` и `c++` можно создавать процессы, которые значительно ускоряют работу программы. Связь между ними можно осуществить с помощью `pipe` (так называемой трубки), что очень круто!