Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

Лабораторная работа №1 по курсу “Операционные системы”

Студент: Иванов Андрей Кириллович

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 13

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1 Репозиторий

https://github.com/kumaroid/osLabs

2 Цель работы

Приобретение практических навыков в:

- Управлении процессами в ОС
- Обеспечении обмена данных между процессами посредством каналов

3 Задание

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.

4 Описание работы программы

Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода.

В ходе выполнения лабораторной работы я использовал следующие системные вызовы:

- fork() - создание нового процесса
- pipe() - создание канала
- dup2() - создание копии файлового дескриптора, используя для нового дескриптора самый маленький свободный номер файлового дескриптора.
- execlp() - запуск файла на исполнение

5 Исходный код

parent.hpp

```
#pragma once

#include <iostream>
#include <fstream>
#include <string>
#include <sys/wait.h>
#include <unistd.h>

int ParentWork();
void CreatePipe(int *fd);
int CreateFork();
void DoClose(int fd);
```

parent.cpp

```
#include "parent.hpp"

void CreatePipe(int *fd) {
    int status;
    status = pipe(fd);
    if (status == -1) {
        std::cerr << "Failed pipe()\n";
        exit(-1);
    }
}

int CreateFork() {
    int pid;
    pid = fork();
    if (pid == -1) {
        std::cerr << "Failed fork()\n";
        exit(-2);
    }
    return pid;
}

void DoClose(int fd) {
    int status;
    status = close(fd);
    if (status == -1) {
        std::cerr << "Failed close()\n";
        exit(-3);
    }
}

void DoDup(int a, int b) {
    int status;
    status = dup2(a, b);
    if (status == -1) {
        std::cerr << "Failed dup2()\n";
        exit(-4);
    }
}

int ParentWork() {
    pid_t pid;
    int status;
```

```
43      int capacity = 0;
44      int fd1[2];
45      int fd2[2];
46      int connect[2];
47      char *child1;
48      char *child2;
49      std::string line;
50
51      child1 = getenv("PATH_CHILD1");
52      child2 = getenv("PATH_CHILD2");
53
54      CreatePipe(fd1);
55      CreatePipe(fd2);
56      CreatePipe(connect);
57
58      pid = CreateFork();
59      if (pid == 0) { // child 1
60          DoClose(connect[0]);
61          DoClose(fd1[1]);
62          DoDup(connect[1], STDOUT_FILENO);
63          if (execlp(child1, "lower.out", std::to_string(fd1[0]).
    c_str(), nullptr) == -1) {
64              std::cerr << "Failed execlp()\n";
65              exit(-5);
66          }
67      }
68      pid = CreateFork();
69      if (pid == 0) { // child 2
70          DoClose(connect[1]);
71          DoClose(fd2[0]);
72          DoDup(connect[0], STDIN_FILENO);
73          if (execlp(child2, "underscore.out", std::to_string(fd2
    [1]).c_str(), nullptr) == -1) {
74              std::cerr << "Failed execlp()\n";
75              exit(-5);
76          }
77      }
78
79      if (pid != 0) {
80          DoClose(fd1[0]);
81          DoClose(fd2[1]);
82          char chWrite;
83          char end = '\0';
84          char toOut[256];
85          int readCount;
86          while (std::getline(std::cin, line)) {
87              write(fd1[1], line.c_str(), line.size());
88
89              chWrite = '\n';
90              write(fd1[1], &chWrite, sizeof(chWrite));
91
92              capacity += line.size() + 1;
93          }
94          write(fd1[1], &end, sizeof(end));
95
96          while(capacity > 0) {
97              readCount = read(fd2[0], toOut, sizeof(toOut));
98              if (readCount != -1) {
99                  capacity -= readCount;
```

```
            }
            //std::cout << toOut << std::flush;
            for (int i = 0; i < readCount; ++i) {
                std::cout << toOut[i] << std::flush;
            }
        }
    }

    waitpid(-1, &status, 0);
    waitpid(-1, &status, 0);
    return 0;
}
```

lower.cpp

```
#include <iostream>
#include <string>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>

int main (int argc, char** argv) {
    if (argc != 2) {
        std::cerr << "Wrong argc in lower.out\n";
        exit(-6);
    }
    char chLow;
    int fd = std::atoi(argv[1]);
    while(true) {
        read(fd, &chLow, sizeof(chLow));
        if(chLow == '\0') {
            break;
        }
        chLow = (char)tolower(chLow);
        std::cout << chLow << std::flush;
    }
    std::cout << '\0';
}
```

underscore.cpp

```
#include <iostream>
#include <string>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>

int main (int argc, char** argv) {
    if (argc != 2) {
        std::cerr << "Wrong argc in underscore.out\n";
        exit(-6);
    }
    char ch;
    int fd = std::atoi(argv[1]);
    (void)argc;
    while(true) {
        std::cin.get(ch);
        if (ch == ' ') {
            ch = '_';
```

```
        } else if (ch == '\0') {
            break;
        }
        write(fd, &ch, sizeof(ch));
    }
}
```

parent.cpp

```
#include "parent.hpp"

void parentProcess(const char *pathToChild) {
    std::string fileName;
    getline(std::cin, fileName);

    int fd1[2], fd2[2];
    createPipe(fd1);
    createPipe(fd2);

    int pid = createChildProcess();
    if (pid != 0) { // Parent process
        close(fd1[0]);
        close(fd2[1]);

        std::string str;
        while (getline(std::cin, str)) {
            str += "\n";
            write(fd1[1], str.c_str(), str.length()); // from str to fd1[1]
        }
        close(fd1[1]);

        std::stringstream output = readFromPipe(fd2[0]);
        while(std::getline(output, str)) {
            std::cout << str << std::endl;
        }
        close(fd2[0]);
    } else { // Child process
        close(fd1[1]);
        close(fd2[0]);

        if (dup2(fd1[0], STDIN_FILENO) == -1 || dup2(fd2[1], STDOUT_FILENO) == -1) {
            perror("Error with dup2");
            exit(EXIT_FAILURE);
        }

        if (execlp(pathToChild, pathToChild, fileName.c_str(), nullptr) == -1) { // to child.cpp
            perror("Error with execlp");
            exit(EXIT_FAILURE);
        }
    }
}
```

main.cpp

```
#include "parent.hpp"

int main() {
    ParentWork();
```

```
5       return 0;
6   }
```

6 Тесты

```
#include <iostream>
#include <gtest/gtest.h>

#include "parent.hpp"

void TestParent(std::string &src, std::string &res) {
    std::istringstream srcStream(src);

    std::streambuf* buf = std::cin.rdbuf(srcStream.rdbuf());

    testing::internal::CaptureStdout();
    ParentWork();
    ASSERT_EQ(testing::internal::GetCapturedStdout(), res + '\n');

    std::cin.rdbuf(buf);
}


TEST(cin_test, ONE) {
    std::string src = "       ";
    std::string res = "_______";
    TestParent(src, res);
}

TEST(cin_test, TWO) {
    std::string src = "AHAHAHAHAH";
    std::string res = "ahahahahah";
    TestParent(src, res);
}

TEST(cin_test, THREE) {
    std::string src = "   HELLO wOrLd 12 3";
    std::string res = "___hello_world_12_3";
    TestParent(src, res);
}
```

7 Запуск тестов

```
andrew@DESKTOP-K3DH39N:~/MAI/osLabs/build/tests/lab1_test$ ./lab1_test
Running main() from /home/andrew/MAI/osLabs/build/_deps/googletest-src/googletest/s
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from cin_test
[ RUN      ] cin_test.ONE
[       OK ] cin_test.ONE (1 ms)
[ RUN      ] cin_test.TWO
[       OK ] cin_test.TWO (1 ms)
[ RUN      ] cin_test.THREE
[       OK ] cin_test.THREE (1 ms)
[----------] 3 tests from cin_test (3 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (3 ms total)
[  PASSED  ] 3 tests.
```

8 Выводы

В результате выполнения данной лабораторной работы была написана программа на языке C++, осуществляющая работу с процессами и взаимодействие между ними. Были приобретены практические навыки в управлении процессами в ОС и обеспечении обмена данных между процессами посредством каналов.