Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу

«Операционные системы»

Тема работы

Студент: Рылов Александр Дмитриевич
Группа: М8О-207Б-21
Вариант: 14
Преподаватель: Миронов Евгений Сергеевич

Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/Brokiloene/os/tree/main/lab_2

Постановка задачи

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.

Правило фильтрации: Child1 переводит строки в нижний регистр. Child2 убирает все задвоенные пробелы.

Общие сведения о программе

Программа содержится в трех файлах — main.c, child.c, child2.c

Общий метод и алгоритм решения

Запуск осуществляется при помощи ввода в командную строку unix:

./main <размер строки>

При помощи вызова fork создаются два процесса.

В родительском процессе вновь вызывается fork, теперь активны 3 процесса – два дочерних и родительский.

Родитель считывает строки text, и отсылает их в дочерний процесс child. Child выполняет операцию над строками и отправляет результат на обработку в child2. Child2 отсылает строки в родительский процесс, который выводит результат в терминал.

Исходный код

main.cc

```
//14 вариант Child1 переводит строки в нижний регистр. Child2 убирает все
задвоенные пробелы.

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char const *argv[]) //main.c
{

      int size;
      size = atoi(argv[1]) + 1;
      if (size <= 0) {
            return 0;
      }

      int fd[3][2];
```

```
    for (int i = 0; i < 3; i++) {
        if (pipe(fd[i]) == -1) { return 2; }
    }

    int pid = fork();

    if (pid == -1) { return 3; }
    if (pid == 0) { // 65--90 +32
        close(fd[1][0]);
        close(fd[0][1]);
        close(fd[2][0]);
        close(fd[2][1]);

        dup2(fd[0][0], STDIN_FILENO);
        dup2(fd[1][1], STDOUT_FILENO);

        close(fd[0][0]);
        close(fd[1][1]);

        execlp("./child", "child", (char *)NULL);
    }

    int pid2 = fork();
    if (pid2 == 0) { //child 2
        close(fd[0][0]);
        close(fd[0][1]);
        close(fd[1][1]);
        close(fd[2][0]);

        dup2(fd[1][0], STDIN_FILENO);
        dup2(fd[2][1], STDOUT_FILENO);

        close(fd[1][0]);
        close(fd[2][1]);
        execlp("./child2", "child2", (char *)NULL);


    }
    close(fd[0][0]);
    close(fd[1][0]);
    close(fd[1][1]);
    close(fd[2][1]);


    int size = atoi(argv[1]) + 1;
    char text[size];
    if (fgets(text, size, stdin) == NULL) { return 1; }

    if (write(fd[0][1], &size, sizeof(int)) == -1) { return 4; }
    if (write(fd[0][1], text, size * sizeof(char)) == -1) { return 4; }
```

```
      if (read(fd[2][0], &size, sizeof(int)) == -1) { return 5; }
      if (read(fd[2][0], text, size * sizeof(char)) == -1) { return 5; }
      printf("%s\n", text);

      wait(NULL);
      wait(NULL);

      close(fd[0][1]);
      close(fd[2][0]);

      return 0;
}

 #include <stdio.h>
#include <unistd.h>

int main(int argc, char const *argv[]) //child.c
{
      int size;
      read(0, &size, sizeof(int));
      //printf("%d\n", size);
      char text[size];
      read(0, text, size * sizeof(char));
      for (int i = 0; i < size; i++) {
            int ch = text[i];
            if (ch >= 65 && ch <= 90) {
                  ch += 32;
            }
            text[i] = ch;
      }
      write(1, &size, sizeof(int));
      write(1, text, size * sizeof(char));
      return 0;
}

#include <stdio.h>
#include <unistd.h>

int main(int argc, char const *argv[]) //child2.c
{
      int size;
      if (read(0, &size, sizeof(int)) == -1) {return 5;}
      char text[size];
      if (read(0, text, size * sizeof(char)) == -1) {return 5;}

      int new_size = size;
      for (int i = 0, end = size - 1; i < end; i++) {
            if (text[i] == ' ' && text[i] == text[i + 1]) {
                  for (int j = i, end = size - 1; j < end; j++) {
                        text[j] = text[j + 1];
```

```
                }
                size--;
                i--;
            }
        }

        if (write(1, &new_size, sizeof(int)) == -1) { return 4; }
        if (write(1, text, new_size * sizeof(char)) == -1) { return 4; }
        return 0;
}
```

Демонстрация работы программы

```
user@brokiloene:~/Desktop/all/os/lab_1/lab_1/src$ ./main 25
ThiS  is a TEST  SEntence
this is a test sentence
```

Выводы

Я приобрёл навыки в управлении процессами в ОС Unix и обеспечении обмена данных между процессами при помощи каналов.