Идз №1

Вариант 20

Мушкат Илья Валерьевич БПИ238

Задание

Разработать программы, состоящую из нескольких параллельных процессов, взаимодействующих между собой через каналы. Количество программ в задании и их характеристики определяются в соответствии с требованиями выполнения на соответствующую оценку.

**Вариант 20**

Разработать программу, вычисляющую число вхождений различных цифр в заданной ASCII–строке. В выходном тексте для каждой цифры вывести, сколько раз она встретилась в обработанном тексте.

Общая схема решаемой задачи

**Схема взаимодействия процессов**

Программа состоит из двух независимых процессов, взаимодействующих через **очереди сообщений**:

1. **Процесс 1 (sender):**

Открывает входной файл для чтения.

Читает данные из файла порциями по 128 байт.

Отправляет каждую порцию данных через очередь сообщений процессу 2.

После завершения передачи данных отправляет специальное сообщение, сигнализирующее о завершении.

Ожидает результаты обработки от процесса 2 через другую очередь сообщений (результаты).

Записывает результаты в выходной файл.

1. **Процесс 2 (processor):**

Создает две очереди сообщений: одну для запросов, другую для результатов.

Ожидает данные от процесса 1 через очередь запросов.

Обрабатывает каждую порцию данных, подсчитывая количество вхождений каждой цифры.

После завершения обработки всех данных отправляет результаты процессу 1 через очередь результатов.

Завершает свою работу.

1. **Очереди сообщений:**

Очередь запросов: Используется для передачи данных от процесса 1 к процессу 2.

Очередь результатов: Используется для передачи результатов обработки от процесса 2 к процессу 1.

1. **Файлы:**

Входной файл: Содержит текстовые данные, которые нужно обработать.

Выходной файл: Содержит результаты обработки (количество вхождений каждой цифры).

**[Процесс 1: sender]**

**|**

**| (Очередь сообщений 1: запросы)**

**v**

**[Процесс 2: processor]**

**|**

**| (Очередь сообщений 2: результаты)**

**v**

**[Процесс 1: sender]**

Описание программы

Программа состоит из двух независимых процессов, взаимодействующих через очереди сообщений. Каждый процесс выполняет определенные функции, описанные ниже.

**1. Процесс 1 (sender)**

**Назначение:**

Чтение данных из входного файла.

Передача данных процессу 2 через очередь сообщений.

Получение результатов обработки от процесса 2.

Запись результатов в выходной файл.

**Реализация:**

Чтение данных из файла:

Используется системный вызов open для открытия входного файла.

Данные читаются порциями по 128 байт с помощью системного вызова read.

**Передача данных через очередь сообщений:**

Создается очередь сообщений для запросов с помощью msgget.

Данные передаются процессу 2 через очередь сообщений с помощью msgsnd.

**Получение результатов обработки:**

Создается очередь сообщений для результатов с помощью msgget.

Результаты обработки получаются от процесса 2 с помощью msgrcv.

**Запись результатов в файл:**

Используется системный вызов open для открытия выходного файла.

Результаты записываются в файл с помощью системного вызова write.

**Завершение работы:**

После передачи всех данных процессу 1 отправляет специальное сообщение, сигнализирующее о завершении.

Очереди сообщений удаляются с помощью msgctl.

**2. Процесс 2 (processor)**

**Назначение:**

Получение данных от процесса 1 через очередь сообщений.

Подсчет количества вхождений каждой цифры.

Передача результатов обработки процессу 1 через очередь сообщений.

**Реализация:**

**Создание очередей сообщений:**

Создаются две очереди сообщений: для запросов и для результатов с помощью msgget.

**Получение данных от процесса 1:**

Данные получаются через очередь сообщений с помощью msgrcv.

**Обработка данных:**

Подсчитывается количество вхождений каждой цифры в переданных данных.

**Передача результатов процессу 1:**

Результаты обработки передаются процессу 1 через очередь сообщений с помощью msgsnd.

**Завершение работы:**

После обработки всех данных процесс 2 завершает свою работу.

**Требования, выполненные в процессе 2:**

Использование очередей сообщений для взаимодействия с процессом 1.

Обработка данных, подсчет цифр.

Передача результатов через очередь сообщений.

**3. Очереди сообщений**

**Назначение:**

Обеспечение взаимодействия между процессами.

**Реализация:**

**Очередь запросов:**

Используется для передачи данных от процесса 1 к процессу 2.

Создается с помощью msgget.

Данные передаются с помощью msgsnd и msgrcv.

**Очередь результатов:**

Используется для передачи результатов обработки от процесса 2 к процессу 1.

Создается с помощью msgget.

Результаты передаются с помощью msgsnd и msgrcv.

**Требования, выполненные для очередей сообщений:**

Использование очередей сообщений для передачи данных между процессами.

Обработка данных порциями по 128 байт.

**4. Файлы**

**Назначение:**

Входной файл содержит текстовые данные для обработки.

Выходной файл содержит результаты обработки (количество вхождений каждой цифры).

**Реализация:**

**Входной файл:**

Открывается с помощью open.

Данные читаются с помощью read.

**Выходной файл:**

Открывается с помощью open.

Результаты записываются с помощью write.

**Требования, выполненные для файлов:**

Использование системных вызовов read и write для работы с файлами.

Обработка файлов произвольного размера.

**5. Командная строка**

**Назначение:**

Задание имен входного и выходного файлов.

**Реализация:**

Программа принимает два аргумента командной строки: Имя входного файла и имя выходного файла.

Если аргументы командной строки некорректны, программа выводит сообщение об ошибке и завершает работу.

**Формат командной строки**

Программа sender принимает аргументы командной строки для задания имен входного и выходного файлов. Формат командной строки должен строго соблюдаться, чтобы программа могла корректно обработать данные.

**Компиляция программ:**

Программы следует скомпилировать перед запуском следующими командами:

gcc -o sender sender.c -std=gnu99

gcc -o processor processor.c -std=gnu99

**Запуск программ:**

Программы запускаются с помощью команд**:**

./processor

./sender <input\_file> <output\_file>

Где:

<input\_file> — путь к входному файлу, содержащему текстовые данные для обработки.

<output\_file> — путь к выходному файлу, в который будут записаны результаты обработки.

Если входной файл не существует или недоступен для чтения, а также, если выходной файл не доступен для записи, то программа выводит сообщение об ошибке и завершает работу.

**./sender non\_existent\_file.txt second\_non\_existent\_file.txt**

Вывод:

**Error: Cannot open input file non\_existent\_file.txt**

**Требования к файлам**

Входной файл: Должен существовать и быть доступным для чтения и может содержать текстовые данные произвольного размера.

Выходной файл: Будет создан, если не существует и если файл существует, его содержимое будет перезаписано.

Примеры

**Пример вызова программ**



1. **Простая строка с цифрами, ‘default\_input.txt’**

**Ввод:**

abc123def4567

**Вывод, ‘default\_output.txt’:**

Digit 0: 0  
Digit 1: 1  
Digit 2: 1  
Digit 3: 1  
Digit 4: 1  
Digit 5: 1  
Digit 6: 1  
Digit 7: 1  
Digit 8: 0  
Digit 9: 0

1. **Строка без цифр, ‘no\_digits\_input.txt’**

**Ввод:**

Hello, World!

**Вывод, ‘no\_digits \_ output.txt’:**

Digit 0: 0  
Digit 1: 0  
Digit 2: 0  
Digit 3: 0  
Digit 4: 0  
Digit 5: 0  
Digit 6: 0  
Digit 7: 0  
Digit 8: 0  
Digit 9: 0

1. **Строка с повторяющимися цифрами, ‘repeat\_digits\_input.txt’**

**Ввод:**

1112223331112223331112223331112223331112223339999999999999

**Вывод, ‘repeat\_digits \_ output.txt’:**

Digit 0: 0  
Digit 1: 15  
Digit 2: 15  
Digit 3: 15  
Digit 4: 0  
Digit 5: 0  
Digit 6: 0  
Digit 7: 0  
Digit 8: 0  
Digit 9: 13

1. **Большой файл весом 26Кб, ‘big\_file\_input.txt’**

**Ввод:**

‘Требования к проекту из [файла к заданию’](https://edu.hse.ru/pluginfile.php/4023871/mod_resource/content/1/task01-core.pdf)

**Вывод, ‘big\_file \_ output.txt’:**

Digit 0: 17  
Digit 1: 23  
Digit 2: 10  
Digit 3: 8  
Digit 4: 9  
Digit 5: 14  
Digit 6: 9  
Digit 7: 9  
Digit 8: 11  
Digit 9: 8

1. **Файл с различными символами, ‘symbols\_input.txt’**

**Ввод:**

a1b2c3d4e5f6??!!...fdf—1212

**Вывод, ‘symbols\_ output.txt’:**

Digit 0: 0  
Digit 1: 3  
Digit 2: 3  
Digit 3: 1  
Digit 4: 1  
Digit 5: 1  
Digit 6: 1  
Digit 7: 0  
Digit 8: 0  
Digit 9: 0