# Практическая работа 7

## Межпроцессное взаимодействие

## 1 Задание

Для выполнения работы необходимо:

**1)** Повторно ознакомиться с теоретическим материалом из презентации по теме «Межпроцессное взаимодействие».

**Внимание! Все настройки, действия и манипуляции, проводимые для выполнения заданий из пункта 2, необходимо фиксировать в отчёте в виде снимков экрана или листингов вводимых команд.**

**2)** С помощью языка программирования Си или C++ необходимо написать **четыре** программы (по одной для родительского и дочернего процесса для двух ОС) с идентичным функционалом, решающих задачу из варианта, – одну для ОС Windows и одну для ОС Linux. Для выполнения задания может использоваться любая среда разработки. При выполнении необходимо учитывать следующие требования:

а) Задача считается решённой, если необходимое по варианту действие выполняется для всех допустимых входных данных.

б) Для работы с процессами (запуска, ожидания, завершения), а также со средствами IPC, следует использовать только предназначенные для этого функции из API соответствующих ОС. **Внимание! Не используйте `return` для завершения процесса в данной работе.** Однако для файлового и консольного ввода-вывода **можно** воспользоваться любыми доступными средствами.

в) Для непосредственной передачи обрабатываемых данных в этой работе следует использовать только средства межпроцессного взаимодействия. Через аргументы следует передавать только названия объектов IPC или другую вспомогательную информацию. Коды

возврата следует использовать только для сообщения об успешном или ошибочном завершении работы процесса.

г) Программа должна осуществлять параллельное выполнение расчётов, т.е. в определённый момент времени работы программы в системе должны **одновременно** работать несколько процессов, каждый из которых выполняет свою часть задачи.

д) Программа должна корректно завершаться, не вызывая аварийный останов.

е) Программа должна брать входные данные и названия файлов, в которых они располагаются, из аргументов, переданных при запуске в консоли. В случае, если количество переданных аргументов не равно ожидаемому, программа должна вывести подсказку для пользователя, поясняющую правила её (программы) использования.

ж) Возвращаемые значения **всех** вызываемых функций должны проверяться на предмет возникновения ошибок. В случае возникновения ошибки необходимо вывести сообщение, оповещающее пользователя о произошедшем, содержащее в обязательном порядке **код ошибки и её текстовое описание**. В случае, если в результате возникшей ошибки программа должна быть завершена, перед завершением необходимо освободить все занятые ресурсы (очистить выделенную память, закрыть открытые файлы).

з) Программа должна работать с любым количеством данных без необходимости внесения изменений в код и перекомпиляции. Если количество процессов, которые необходимо породить, больше количества входных данных, необходимо запустить кол-во процессов, вдвое меньшее кол-ва входных данных и вывести соответствующее предупреждение.

**3)** Оформить отчёт в соответствии с шаблоном из файла «Шаблон и требования к оформлению отчёта по практической работе». Не забыть прикрепить к отчёту **исходные коды** всех программ **в виде приложения (не**

**приложить отдельными файлами, а именно вставить в сам файл с отчётом под заголовками «Приложение А», «Приложение Б» и т.д.)**. В отчёте необходимо отразить, как минимум:

а) Результаты работы программы в виде снимков экрана, демонстрирующих работу программы для 2-3 вариантов верных и для каждого типа неверных входных данных (чтобы продемонстрировать все предусмотренные сообщения об ошибках).

б) Процесс сборки и запуска программы.

в) Изменения ключевых значений (в переменных) в ходе работы программы.

г) Описание реализованного алгоритма, содержащее листинги соответствующих отрезков кода программы. В описание следует включать только ключевые моменты работы программы (т.е. описывать обработку ошибок и освобождение памяти не требуется).

**4)** Защитить работу на занятии. Во время защиты необходимо:

а) Продемонстрировать работу программы.

б) При необходимости изменить входные данные и продемонстрировать, что программа отрабатывает верно на новых данных.

в) Ответить на вопросы по логике работы программы (почему и зачем была написана та или иная строка, каково её назначение и вклад в решение данной задачи).

г) Ответить на вопросы по теоретической части (**в том числе по лекциям**).

**5)** Дополнительное задание: дополнительно реализовать третью кроссплатформенную пару программ, решающую задачу по варианту (с использованием IPC, как описано в подразделе 2.2) на языке программирования, отличном от C/C++. Даёт два дополнительных балла. **Не обязательно к выполнению!**

# 2 Варианты заданий

## 2.1 Распределение входных данных

Общее требование для всех заданий: если количество входных данных (чисел или символов) не делится нацело на количество дочерних процессов, числа должны распределяться между дочерними процессами следующим образом:

Пусть кол-во входных данных равно M, кол-во дочерних процессов равно N, тогда первым N - 1 процессам должны быть назначены фрагменты входных данных длины:

$$n = \left\lfloor \frac{M}{N} \right\rfloor,$$

а оставшемуся N-ному процессу:

$$n_N = M - (N-1)\left\lfloor \frac{M}{N} \right\rfloor,$$

где $\lfloor x \rfloor$ – округление вещественного числа $x$ в меньшую сторону. Например, если дано 10 чисел и их необходимо распределить между 4 процессами, получим:

$$n = \left\lfloor \frac{10}{4} \right\rfloor = \lfloor 2{,}5 \rfloor = 2,$$

$$n_N = 10 - (4-1)\left\lfloor \frac{10}{4} \right\rfloor = 10 - 3 \cdot 2 = 10 - 6 = 4.$$

Таким образом, первым трём процессам достанется по 2 числа, а последнему – 4 числа.

Однако, если $N > \frac{M}{2}$, следует уменьшить количество дочерних процессов до $\left\lfloor \frac{M}{2} \right\rfloor$ и вывести соответствующее предупреждение для пользователя.

## 2.2 Способ формирования задач

Чтобы выполнить данную практическую работу, следует взять свой вариант задания из практической работы по теме «Процессы». При выполнении следует учесть следующие изменения:

1) Для передачи данных между процессами не следует использовать файлы. Из файла следует брать только исходные данные для родительского процесса.

2) Каждая пара программ (для Windows и для Linux) должна предоставлять возможность передавать данные двумя способами: через каналы (pipes) и через разделяемую память (shared memory). Пользователь должен иметь возможность задать нужный способ при запуске программы в виде аргумента командной строки.

3) При передаче данных через каналы для **каждого** дочернего процесса должна быть создана пара каналов: для передачи данных от родительского процесса к дочернему и обратно.

4) При передаче данных через объект разделяемой памяти необходимо создать:

   а) **Один** (общий для всех процессов) объект разделяемой памяти для передачи входных данных дочерним процессам. При этом дочерние процессы должны иметь доступ к этой памяти только на чтение.

   б) **Один** (общий для всех процессов) объект разделяемой памяти для сохранения результата расчётов **всех** дочерних процессов. При этом доступ к данному объекту должен синхронизироваться (см. п. в)).

   в) На Linux также должен быть создан дополнительный **один** (общий для всех процессов) объект разделяемой памяти, в котором должен быть расположен мьютекс, предназначенный для синхронизации операций записи результирующих значений. В

Windows для этих целей следует просто создать именованный мьютекс в системе.

Таким образом, обе пары программ должны работать по следующему обобщённому алгоритму:

1) Родительский процесс запускается, получая от пользователя название файла с входными данными, количество процессов, которые необходимо создать, а также отметку о средстве IPC, которое следует использовать при этом запуске.

2) Родительский процесс создаёт необходимое средство IPC и помещает в него данные для дочерних процессов.

3) Родительский процесс запускает дочерние.

4) Дочерние процессы выполняют расчёты с данными, полученными через средства IPC, и возвращают родительскому результат через специально предназначенные для этого отдельные объекты IPC, при необходимости синхронизируя свои действия (например, чтобы избежать гонки данных при изменении результирующего участка разделяемой памяти).

5) Родительский процесс считывает результаты расчётов дочерних процессов из специально предназначенных для этого объектов IPC, при необходимости проводит подсчёт результата и выводит результат в консоль пользователю.