# Лабораторные работы №2

## Цель работы

Приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данных между процессами посредством каналов

## 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.

## Варианты задания

1. Рекурсивное вычисление факториала, где каждый отдельный уровень рекурсии вычисляется в отдельном процессе.
2. Рекурсивное вычисление чисел Фибоначчи, где каждый отдельный уровень рекурсии вычисляется в отдельном процессе.
3. Рекурсивное вычисление суммы последовательности от 1 до n, где каждый отдельный уровень рекурсии вычисляется в отдельном процессе.
4. Рекурсивная проверка простоты числа, где каждый отдельный уровень рекурсии вычисляется в отдельном процессе.
5. На вход программе подается команда интерпретатора команд. Программа должна произвести вывод команды в верхнем регистре.
6. На вход программе подается команда интерпретатора команд. Программа должна произвести вывод команды заменяя знаки табуляции и space на знаки «\_\_\_» и «\_» соответственно.
7. На вход программе подается название 2-ух файлов. Необходимо отсортировать оба файла (каждый в отдельном процессе) произвольной сортировкой (на усмотрение студента). Содержимое обоих файлов вывести в стандартный поток вывода родительским процессом.
8. На вход программе подается название 2-ух файлов. Необходимо отсортировать оба файла (каждый в отдельном процессе) произвольной сортировкой (на усмотрение студента). Родительским процессом вывести отсортированные файлы чередованием строк первого и второго файлов.
9. На вход программе подается название 2-ух неотсортированных файлов. Необходимо отсортировать оба файла (каждый в отдельном процессе) произвольной сортировкой (на усмотрение студента). Из двух отсортированных файлов необходимо получить на выходе один "слитый" отсортированный файл.
10. Программа позволяющая перенаправлять стандартный вывод. На вход программе подается имя выходного файла. Далее программа должна принимать различные команды интерпретатора команд и весь их вывод перенаправлять в файл.
11. Написание собственного простого целочисленного калькулятора с операциями "+","-". В дочернем процессе должны происходить вычисления выражений. В родительском процессе ввод/вывод.
12. Написание собственного простого целочисленного калькулятора с операцией "\*", " /". В дочернем процессе должны происходить вычисления выражений. В родительском процессе должны происходить вывод/ввод и передача их дочернему процессу.
13. Родительский процесс считывает матрицы из стандартного ввода, дочерний процесс вычисляет среднеарифметическое матриц и передает их родительского процессу. Родительский процесс печатает результат в стандартный выход.
14. Родительский процесс считывает числа со стандартного входного ввода. Дочерний процесс вычисляет квадратный корень этих чисел и передает результаты на печать родительскому процессу.
15. Родительский процесс считывает стандартной входной поток, отдает его дочернему процессу, который удаляет "задвоенные" пробелы и выводит его в файл (имя файла также передается от родительского процесса).
16. **На вход программе подается команда интерпретатора команд и имя файла. Программа должна перенаправить стандартный ввод команды с этого файла и вывести результат команды в стандартный выходной поток. Использование операций write и printf запрещено.**
17. Дочерний процесс представляет собой сервер по работе с очередями и принимает команды со стороны родительского процесса.
18. Дочерний процесс представляет собой сервер по работе со стеками и принимает команды со стороны родительского процесса.
19. Дочерний процесс представляет собой сервер по работе с бинарным деревом и принимает команды со стороны родительского процесса.
20. Дочерний процесс представляет собой сервер по работе с деревом общего вида и принимает команды со стороны родительского процесса.
21. Родительский процесс представляет собой сервер по работе со списками и принимает команды со стороны дочернего процесса.
22. Родительский процесс представляет собой сервер по работе с массивами и принимает команды со стороны дочернего процесса.
23. Родительский процесс считывает две координаты передает их через канал дочернему процессу. Дочерний процесс определяет к какой четверти относится точка, а далее передает результат родительскому процессу.
24. Необходимо, чтобы дочерний процесс "слушал" стандартный ввод. И если там появляется число (экспоненциальную форму числа также необходимо предусмотреть), то передавал бы это число родительскому процессу для вывода в стандартный вывод.
25. Родительский процесс отвечает за ввод и вывод. Дочерний процесс осуществляет поиск образца в строке.
26. Дочерний процесс при создании принимает имя файла. При работе дочерний процесс получает числа от родительского процесса и пишет их в файл. Родительский процесс создает n дочерних процессов и передает им поочередно числа из последовательности от 1...m.
27. Дочерний процесс при создании принимает имя файла. При работе дочерний процесс получает символы от родительского процесса и пишет их в файл. Родительский процесс создает n дочерних процессов и передает им поочередно символы из входной строки.

Варианты задания выдаются преподавателем.

## Системные вызовы

Для ОС Linux

* pid\_t fork() - создание дочернего процесса
* int execve(const char \*filename, char \*const argv[], char \*const envp[]) (и другие вариации exec) - замена образа памяти процесса
* pid\_t waitpid(pid\_t pid, int \*status, int options) - Ожидание завершения дочернего процесса
* void exit(int status) - завершения выполнения процесса и возвращение статуса
* int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
* int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
* int open(const char \*pathname, int flags, mode\_t mode) - открытие\создание файла
* int close(int fd) - закрыть файл
* int mkfifo(const char \*pathname, mode\_t mode) - создание именованного канала

Для ОС Windows

* BOOL WINAPI CreateProcess(…) - создание нового процесса
* WaitForSingleObject(…) - ожидание завершения процесса
* ExitProcess(…) - завершение выполнения процесса
* CreateFile/SetNamedPipeHandleState - создание именованного канала и установления режима его использования
* Int \_dup2(int fd1, int fd2) - переназначение файлового дескриптора
* OpenFile(…) - открытие нового файла
* CreatePipe(…) - создание безымянного канала
* CreateFile(…) - создание нового файла
* CloseHandle(…) - закрытие объекта ОС по "заголовку". Подходит для закрытия файлов.