# Лабораторная работа №4: "Синхронизация процессов и управление общими ресурсами"

**Тема:** Синхронизация процессов и управление общими ресурсами (События, Мьютексы, Семафоры).

**Цель работы:** На примере реалистичной модели научиться использовать стандартные объекты ядра Windows для синхронизации независимых процессов. Понять, как решаются проблемы ограниченных ресурсов (пропускная способность сети) и конкурентного доступа (запись в лог) в реальных приложениях.

**Задача:** Разработать систему, имитирующую работу менеджера загрузок. Система состоит из двух программ:

- `Browser.exe` — главный процесс браузера, который управляет загрузками.
- `Downloader.exe` — дочерний процесс, имитирующий одну активную задачу по скачиванию и обработке файла.

**Сценарий из реальной жизни:**
Любой браузер ограничивает количество одновременных загрузок (например, не больше 6), чтобы не перегружать сетевое соединение. Остальные загрузки ставятся в очередь. Все события (начало, конец, ошибка) записываются в общий журнал загрузок. Когда пользователь закрывает браузер, все активные и ожидающие загрузки должны быть корректно прерваны.

---

**Роль программы `Browser.exe` (Главный процесс браузера)**

1. При запуске `Browser.exe` запрашивает у пользователя:
   - `N` — максимальное количество **одновременных загрузок** (например, 4).
   - `M` — общее количество **файлов для скачивания**, поставленных в очередь (например, 15). `M` должно быть больше `N`.
2. Создает все необходимые **именованные** объекты синхронизации для управления "дочерними" загрузками:
   - **Семафор** с именем `"DownloadSlots"`. Он моделирует ограниченную пропускную способность сети. Его начальное и максимальное значение равно `N`.
   - **Мьютекс** с именем `"LogAccessMutex"`. Он защищает "Журнал загрузок" (консоль), чтобы сообщения от разных процессов не смешивались в "кашу".
   - **Событие** с ручным сбросом (`manual-reset`) с именем `"BrowserClosingEvent"`. Это сигнал тревоги, который сработает, когда пользователь решит "закрыть браузер". Изначально событие несигнальное.

3. Запускает `M` экземпляров процесса `Downloader.exe`, каждый из которых будет пытаться скачать и обработать свой файл.
4. Выводит сообщение "Browser is running. Press Enter to close..." и ожидает нажатия клавиши.
5. После нажатия Enter:
   a. Выводит сообщение "Browser is closing. Sending termination signal to all downloads..."
   b. Переводит событие `"BrowserClosingEvent"` в сигнальное состояние (`SetEvent`).
6. Дожидается завершения **всех** запущенных процессов `Downloader.exe` (с помощью `WaitForMultipleObjects`).
7. Корректно закрывает все дескрипторы и завершает работу.

---

**Роль программы `Downloader.exe` (Процесс загрузки)**

1. При запуске каждый `Downloader.exe` "подключается" к среде браузера, открывая существующие именованные объекты: семафор `"DownloadSlots"`, мьютекс `"LogAccessMutex"` и событие `"BrowserClosingEvent"`.

2. Каждый процесс входит в **однократный** цикл выполнения (один процесс = одна попытка скачать один файл).

3. **Шаг 1: Ожидание в очереди.** Процесс ожидает, пока не освободится "слот" для загрузки (`WaitForSingleObject` на семафоре). Во время ожидания процесс проверяет, не закрывается ли браузер:

   - Для этого используется `WaitForMultipleObjects` для ожидания либо семафора, либо события закрытия. Если сработало событие закрытия, процесс сразу переходит к завершению (Шаг 5).

4. **Шаг 2: Начало загрузки.** Как только слот получен:
   a. Захватывает мьютекс `"LogAccessMutex"`.

   b. Выводит в консоль (в "Журнал загрузок") сообщение: `"[PID: XXX] Connection established. Starting download of 'file_name.ext'..."`

   c. Освобождает мьютекс `"LogAccessMutex"`.

   d. **Имитирует "обработку" скачанного файла, выполняя задачу согласно индивидуальному варианту.**

   e. "Спит" случайное время (от 1 до 3 секунд), имитируя время загрузки.

   f. Захватывает мьютекс `"LogAccessMutex"`.

g. Выводит в консоль сообщение: `"[PID: XXX] File 'file_name.ext' processed successfully."`

h. Освобождает мьютекс `"LogAccessMutex"`.

5. **Шаг 3: Завершение.**
    - Если задача была выполнена успешно, процесс освобождает свой слот (`ReleaseSemaphore`), чтобы другой ожидающий процесс мог начать загрузку.
    - Если работа была прервана сигналом `"BrowserClosingEvent"`, он **не** освобождает слот (так как он его, возможно, и не занимал), а просто выводит сообщение о прерывании.
6. Корректно закрывает все свои дескрипторы и завершает работу.

---

**Индивидуальные варианты**

**Задача, имитирующая обработку скачанного файла (пункт 4d):**

1. Обратить порядок байт в массиве размером 2048.
2. Посчитать количество слов в длинной строке.
3. Отсортировать массив из 500 случайных символов.
4. Найти стандартное отклонение для массива из 200 чисел.
5. Посчитать количество открывающих и закрывающих скобок в длинной текстовой строке.
6. Найти самую длинную строку в массиве строк.
7. Проверить строку на сбалансированность скобок `()`, `[]`, `{}`.
8. Найти произведение всех элементов матрицы 10x10.
9. Посчитать количество простых чисел от 1 до 10000.