Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 11

з дисципліни «Операційні системи»

Тема: «Процеси. Семафори»

Виконав:

студент 3 курсу

групи КС-32

Безрук Юрій Русланович

Перевірив: Споров О.Є.

Харків – 2020

Целью данной работы продолжение изучения основных средств, позволяющих осуществлять взаимодействие между процессами – IPC. В данной работе будет рассмотрено средство для синхронизации процессов – семафоры System V.

# ХОД РАБОТЫ

**Задание 1**

В ходе выполнения работы требуется модифицировать программу из прошлой лабораторной работы, заменив синхронизацию из сигналов на синхронизацию семафорами.

В работе помимо способа синхронизации изменяется также и внутренняя структура программ. Теперь две программы выглядят таким образом: есть сервер, который выполняет все операции по созданию и очистке общей памяти и массивов семафоров, а так же выполняет вычисления; и есть клиент, который подключается к памяти и семафорам, и задает серверу значения для вычислений, общаясь с пользователем. Два процесса теперь больше не связаны родственными связями и запускаются отдельно и независимо с той лишь оговоркой, что сервер должен запуститься раньше.

Создание сегментов общей памяти сервером полностью аналогично предыдущей лабораторно работе, за исключением того, что ключ для нее используется не IPC\_PRIVATE, а сгенерированный при помощи функции ftok() из имени заранее подготовленного текстового файла. Так как два процесса напрямую никак не взаимодействуют друг с другом, это единственная возможность получить обоим один и тот же идентификатор сегмента.

Параллельно с эти так же создается массив семафоров при помощи функции semget() с так же сгенерированным ключом, но на основе другого файла. Семафоров создается 2 для удобства работы – один будет ограничивать запись в память, а второй – чтение из нее.

Для того, чтобы совершать операции над семафорами используется массив структур sembuf, каждый элемент из которого соответствует одной операции над одним семафором. Полями этой структуры указываются: sem\_num – номер семафора, над которым выполняется операция, sem\_op – операция, которая с ним выполняется. При передаче числа n семафор увеличивает значение на него. При передаче –n блокируется пока не будет возможности отнять это значение (т.е., пока значение семафора s<n, далее s=s-n). При передаче 0 вызов блокируется пока семафор не станет равен 0.

Поле sem\_flg заранее ставим 0. Т.к. атомарно мы будем выполнять только одну операцию над семафорами, массив состоит из одного элемента (можно вообще использовать переменную структуры). Для реализации операций определяются функции sem\_block и sem\_unblock, которые совершают прибавление или попытку вычитания из переданного номера семафора (номера передаются как заранее определенные макросы SEM1 = 0, SEM2 = 1) при помощи вызова функции semop(), которая принимает на вход идентификатор массива семафоров, массив операций над ними и количество семафоров, над которыми совершается операция.

Т.е., вызов sem\_block(SEM1) заблокирует процесс если значение семафора с номером 0 равно нулю. Он выйдет из этой блокировки в случае если другой процесс увеличит значение семафора при помощи вызова sem\_unblock(SEM1).

После проведения всех необходимых приготовлений, сервер начинает свою работу в цикле с попытки блокирования первого семафора. Т.е., он ожидает, когда ему можно будет приступить к рассчетам, а это можно будет сделать только в случае если другой процесс разблокирует семафор (по умолчанию семафор заблокирован, т.е. равен 0).

Когда семафор удается заблокировать, сервер считает сумму чисел в общей памяти, после чего разблокирует второй семафор для второго процесса, что бы он мог прочитать результат. Далее цикл повторяется и сервер снова ждёт семафора, разрешающего вычисление.

Взаимодействие клиента с семафорами реализовано при помощи таких же структур, макросов и функций.

Сам процесс клиента точно так же получает ключи общей памяти и семафоров, но без флага IPC\_EXCL. В основном же цикле клиент сразу начинает взаимодействие с пользователем, т.к. подразумевается, что к этому моменту сервер уже работает.

В самом цикле работы клиента считываются данные от пользователя в общую память так же, как и в прошлой лабораторной работе, После чего клиент разблокирует первый семафор для того, чтобы сервер мог начать вычисления, а сам пытается заблокировать второй семафор, для того что бы начать считывать результат, но он не сможет его заблокировать, пока сервер не закончит вычисления и не разблокирует второй семафор. Как только второй семафор получается заблокировать, клиент выводит результат и процесс повторяется.

Завершение работы системы реализовано таким образом: клиент завершается всё так же с использованием ввода отрицательного числа. При завершении он отсоединяет совместную память. Но, т.к. программы не связаны, завершение одной части не влечет за собой завершение другой: сервер может работать и с другим клиентом. То есть, сервер должен завершаться отдельно и не обязательно сразу.

Для его корректного завершения был использован следующий механизм: во-первых, для процесса устанавливается обработчик завершения, который автоматически отсоединяет память и удаляет сегменты и массивы семафоров при любом ее завершении. Кроме того, для процесса регистрируется обработчик сигнала принудительного завершения SIGINT, посылаемого терминалом комбинацией клавиш Ctrl+C. Этот обработчик завершает процесс функцией exit(), благодаря чему запускается обработчик завершения и очищает данные.

Таким образом, для корректного завершения работы сервера достаточно просто ввести Ctrl+C в терминале, либо отправить ему сигнал SIGINT.

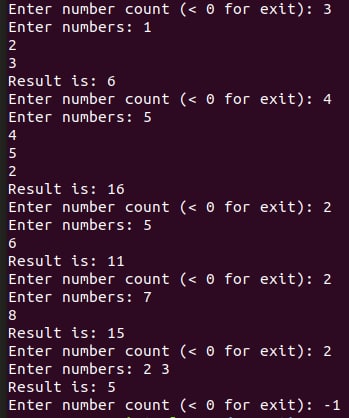


Рисунок 1 – пример работы программы

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены средства синхронизации семафоры System V IPC. Исходные, исполняемые и файлы сборки помещены в архив.