Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 8

з дисципліни «Операційні системи»

Тема: «Потоки. Синхронізація II»

Виконала:

студентка 3 курсу

групи КС-32

Дібцева Анна Миколаївна

Перевірив: Споров О.Є.

Харків – 2020

Цель: ознакомление с еще двумя средствами, предназначенными для синхронизации потоков: семафорами и условными переменными.

# ХОД РАБОТЫ

**Задание 1**

Задание 1 - задача о производителе и потребителе, которую требуется реализовать при помощи семафоров.

Данные, которые производитель предоставляет потребителю глобальный массив-буфер. Благодаря существованию именно массива, производитель и потребитель смогут выполнять свои функции не дожидаясь, пока другая сторона обработает его элемент (до определенного количества, равного размеру буфера). В программе используются три семафора: семафор lock используется по аналогии с мьютексом: запрещает или разрешает доступ к критической секции, семафор empty\_items отображает количество пустых ячеек (изначально - все), а семафор full\_items – количество занятых (в начале - 0).

В основном потоке происходит инициализация семафоров функцией sem\_init(), создание потоков, после чего основной поток начинает выводить статус буфера с периодичностью в 1 секунду заданное количество секунд, отменяет потоки, дожидается их завершения и освобождает память.

Поток-производитель бесконечно генерирует псевдослучайное число заданного диапазона, ожидает семафор пустых элементов (т.е. блокируется, пока пустых элементов не станет хотя бы 1), ожидает семафор критической секции, и помещает ресурс в буфер. После этого он увеличивает на 1 семафор критической секции и заполненных ячеек и засыпает на 1 секунду.

Поток потребитель устроен практически с точностью наоборот: он ожидает появления заполненного элемента, ожидает доступ к крит. секции, и берет значение из буфера по позиции, получаемой аналогичным образом. После этого освобождаются семафоры и крит. секции пустых элементов.

Потоки узнают о том, из какой ячейки нужно брать информацию и в какую нужно ложить благодаря глобальной переменной position, доступ к которой они получают в критической секции, благодаря чему она является достоверной.

Программу можно модифицировать при помощи опций: есть возможность задать размер буфера, диапазон псевдослучайных чисел и время ожидания основного потока.

Пример выполнения программы:

Рисунок 1 - производитель и потребитель, семафоры

**Задание 2**

В задании 2 также задача производителя-потребителя, реализуется с использованием условных переменных.

Вместо массива данных теперь используется одна переменная. Для исправления доступом используется переменная-флаг, показывающая наличие продукта либо его отсутствие, две условных переменных для каждого из потоков и мьютекс для ограничения доступа к ним.

Условные переменные создаются при помощи функции pthread\_cond\_init(), а в конце программы очищаются вызовом pthread\_cond\_destroy()

Поток-производитель производит товар (случайное число), запирает мьютекс для единичного доступа к условной переменной и вызывает ожидание по условной переменной производителя, т.е., ждёт сигнала от потребителя, что он забрал товар вызовом pthread\_cond\_wait() (с проверкой переменной flag, которая призвана доказать, что товар действительно исчез), и помещает товар в общую переменную. После этого производитель сигнализирует об этом потребителя по его условной переменной (pthread\_cond\_signal())и открывает мьютекс.

Поток-потребитель работает таким же образом, с отличием лишь в том, что данные он забирает, а не помещает, и условные переменные для ожидания и отправления сигнала стоят наоборот.

Поток-производитель производит товар с задержкой. Потребитель не ограничен в этом, и больше ожидает товар.

При помощи коротких опций программе можно задать диапазон генерации псевдослучайных чисел и время ожидания основного потока исполнения.

Пример работы программы:

Рисунок 2 - производитель и потребитель, условные переменные

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены возможности синхронизации потоков при помощи семафоров и условных переменных. Соответствующие исходные файлы, исполняемые файлы и скрипты сборки помещены в архив.