Лабораторная работа №3
«Векторное время»

Введение

При решении задачи подсчета полной суммы денег с помощью скалярных часов предполагалось, что всем процессам известен некоторый момент логического времени *t*, в который процессы будут подсчитывать денежные средства на своих счетах и в каналах между филиалами. Однако сама процедура достижения соглашения между процессами о таком будущем моменте *t* представляет определенные сложности. Действительно, при использовании механизма скалярных часов процесс не может произвольным образом выбрать значение *t* для согласования с другими процессами, т.к. не исключено, что на момент выбора *t* логические часы других процессов ушли далеко вперед, и их текущее логическое время потенциально может превышать любое предлагаемое значение *t*. Кроме того, если *t* окажется слишком большим, то придется долго ожидать наступления этого момента. Использование векторных часов может помочь в решении этой задачи.

Исходные данные

Следует использовать исходные данные из этапа 2 лабораторной работы №1.

Задание

В реализации банковской системы из этапа 2 лабораторной работы №1 необходимо заменить скалярное время Лэмпорта на векторное. Для этого при отправке любого из сообщений, определенных в лабораторной работе №1, процесс должен передавать свое текущее векторное время, а при получении сообщения обрабатывать показания векторных часов процесса-отправителя.

Векторные часы представляют собой массив элементов типа *local_time_t*. Размер вектора равен числу процессов, составляющих распределенную систему. Показания векторных часов добавляются в сообщение при вызове функций *send**() и обрабатываются при получении сообщений в функциях *receive*()*. В канал массив должен передаваться после заголовка сообщения *MessageHeader* и перед телом сообщения *payload*. При этом длина сообщения *s_len* должна быть модифицирована соответствующим образом. Стоит отметить, что это действие абсолютно прозрачно для пользователей данных функций. Поле заголовка *s_local_time* перенесено в *s_reserved[]*. Доступ процесса к своим векторным часам и тип памяти для их хранения не специфицированы.

Для подсчета полной суммы денег добавлены новые типы сообщений: *EMPTY, SNAPSHOT_VTIME, SNAPSHOT_ACK*. Сообщение *SNAPSHOT_VTIME* используется для согласования векторного времени *t*, когда необходимо выполнить подсчет денег, и содержит время следующего, еще не наступившего события процесса-отправителя *P*. Процесс, получивший сообщение *SNAPSHOT_VTIME,* должен отправить в ответ пустое сообщение типа *SNAPSHOT_ACK*. Отправка и получение *SNAPSHOT_VTIME* и *SNAPSHOT_ACK* не должны влиять на показания векторных часов процессов. Сообщение *EMPTY* можно использовать для продвижения времени процесса-получателя. Каждый процесс, получивший *SNAPSHOT_VTIME,* должен передать процессу *P*, инициировавшему подсчет денег, сообщение типа *BALANCE_STATE* после того, как наступил момент времени *t* и известно состояние каналов. При получении информации, необходимой для подсчета полной суммы денег, процесс *P* выводит сумму на экран в виде «*[t_1, t_2, … , t_n] $balance $pending\n*», где в квадратных скобках указывается время *t*.

Процедура согласования векторного времени *t* для подсчета полной суммы денег инициируется родительским процессом посредством вызова функции *total_sum_snapshot()*. Данная функция вызывается родительским процессом из функции *bank_robery()* и должна быть реализована студентами на основе алгоритма, представленного на лекции.

При реализации векторных часов считать, что у процессов отсутствуют внутренние события. События отправки и получения сообщений *SNAPSHOT_VTIME* и *SNAPSHOT_ACK* не участвуют в продвижении векторного времени. При наступлении каждого события показания логических локальных часов процесса увеличиваются на единицу.

Сообщение *BALANCE_HISTORY* в данной лабораторной работе не используется. Никакие другие изменения в банковскую систему вносить не требуется.

Требования к реализации и среда выполнения

Реализацию необходимо выполнить на языке программирования Си с использованием предоставленных заголовочных файлов и библиотеки из архива *pavt_starter_code.tar.gz*.

Среда выполнения — Linux (Ubuntu версии ≥ 12.10). При проверке используется следующая команда: *gcc -std=c99 -Wall -pedantic -L. *.c*. При наличии варнингов работа не принимается. При успешном выполнении запущенные процессы не должны использовать *stderr*, код завершения программы должен быть равен 0.