Лабораторная работа №4
«Алгоритм взаимного исключения Лэмпорта»

Введение

Взаимное исключение — требование, согласно которому во время выполнения критической области одного процесса ни один другой процесс не должен выполняться в этой же критической области. Различная скорость выполнения процессов и их число, произвольные задержки передачи сообщений и отсутствие полной информации о состоянии всей системы и общей памяти в распределенных системах делают реализацию взаимного исключения одной из фундаментальных проблем в области распределенных вычислений.

Исходные данные

Следует использовать исходные данные из лабораторной работы №1.

Задание

Данная работа выполняется на основе распределенной системы, реализованной в лабораторной №1. При этом понятие «полезной» работы дочернего процесса определяется следующим образом: каждый дочерний процесс должен *N = process_local_id * 5* раз распечатать сообщение, определенное строкой форматирования *log_loop_operation_fmt*, посредством вызова функции *print()*, входящей в состав прилагаемой библиотеки *libruntime.so*. Обратите внимание, что при формировании строки на основе *log_loop_operation_fmt* нумерация итераций должна выполняться, начиная с единицы, а не с ноля.

В *IPC* из л.р. №1 следует внести следующие изменения: вызовы *read()* и *write()* должны быть неблокирующими, как в л.р. №2, а сообщения должны содержать значение скалярных часов отправителя, как в л.р. №3. Это фактически означает, что можно использовать без изменений библиотеку *IPC*, реализованную для л.р. №3.

При запуске программы с параметром командой строки «*--mutexl*» перед каждым вызовом *print()* процесс должен входить в критическую область, а после вызова выходить из нее, т.е. запрещается выполнять несколько вызовов *print()* в пределах одной критической области. При отсутствии параметра «*--mutexl*» программа должна выполняться без использования критической области, обратите внимание на разницу в выводе программы при использовании критической секции и без нее.

Вход в критическую область выполняется с помощью вызова функции *request_cs()*, выход — *release_cs()*. Обе функции необходимо реализовать самостоятельно, используя алгоритм взаимного исключения Лэмпорта. Для этого определены следующие пустые сообщения: *CS_REQUEST*, *CS_REPLY* и *CS_RELEASE*. Идентификатор процесса-отправителя данных сообщений определяется по дескриптору канала, через который сообщение получено. Значение локальных часов отправителя извлекается из заголовка сообщения. Семантика сообщений *CS_REQUEST*, *CS_REPLY* и *CS_RELEASE* подробно описана в лекционных материалах.

Процедура синхронизации процессов при запуске и завершении распределенной системы, как в лабораторной работе №1.

Требования к реализации и среда выполнения

Реализацию необходимо выполнить на языке программирования Си с использованием предоставленных заголовочных файлов и библиотеки из архива *pa2345_starter_code.tar.gz*.

Работа присылается в виде архива с именем *pa4.tar.gz*, содержащим каталог *pa4*. Все файлы с исходным кодом и заголовки должны находиться в корне этого каталога. Среда выполнения — Linux (Ubuntu 14.04, clang-3.5). При автоматической проверке используется следующая команда: *clang -std=c99 -Wall -pedantic *.c –L. -lruntime*. При наличии варнингов работа не принимается. При успешном выполнении запущенные процессы не должны использовать *stderr*, код завершения программы должен быть равен 0.