Задачи 5

Пань Чжэну. Вариант 18

Задачи

18. ***Задача о болтунах.*** N болтунов имеют телефоны, ждут звонков и звонят друг другу, чтобы побеседовать. Если телефон занят, болтун будет звонить, пока ему кто-нибудь не ответит. Побеседовав, болтун не унимается и или ждет звонка или звонит на другой номер. Создать многопоточное приложение, моделирующее поведение болтунов. Для решения задачи использовать мьютексы.

Модель

Пользую модель **взаимодействующие равные** – модель, в которой исключен не занимающийся непосредственными вычислениями управляющий поток. Распределение работ в таком приложении либо фиксировано заранее, либо динамически определяется во время выполнения. Одним из распространенных способов динамического распределения работ является **«портфель задач»**. Портфель задач, как правило, реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс. Так как все болтунов имеет одинаковые приоритеты.

Работа программа

Программа создает лист с болтунов с рандомный начальный статус: либо ждет звонка либо звонит (не ждет).

Если болтун который не ждет (пытается дозвониться) получил звонка от другого болтунов, он будет перестать звонить и возьмет трубку.

Если болтун продолжает ждать или пытается дозвониться (не ждет, то есть не может найти кому то звонить) в течение рандомный время от 1 до 4 секунд, и при этом больше никто из список болтунов не совершил звонок (call_count не изменился),

тогда болтун понимает что зашел в тупик и меняет свое статус (ждет => не ждет или наоборот).

Все болтунов устает после 5 звонка, если все болтунов устали или больше не с кем говорить (остается один не усталый болтун, ему не с кем звонить), программа останавливается.

Программа создает очередь, туда пушит все потоки (каждый поток является один звонок) Все изменение или получение данный от статическая поля (это Talker: calls_queue, telephone_book_, mutex; Call: call_count) происходит внутри мьютексом, то есть безопасно.

R<int> - rethink, передумает о статуса

C<int> - call, звонок

Std::queue calls_queue	**function in mutex (lock)**	**function**	**function in mutex (lock)**	**…**
C1 ->	Change value ->	Sleep for random 1 - 3 second ->	Change value back ->	…
C2 ->	Change value ->	Sleep for random 1 - 3 second ->	Change value back ->	…
C3 ->	Change value ->	Sleep for random 1 - 3 second ->	Change value back ->	…
C4 ->	Change value ->	Sleep for random 1 - 3 second ->	Change value back ->	…
R1 ->	Save call_count	Sleep for random 1 - 4 second ->	Compare and change waiting status	…
C5 ->	Change value ->	Sleep for random 1 - 3 second ->	Change value back ->	…
…	…	…	…	…

После создание потоки, они сразу же начинает проработать. Каждый на своем потоке.

Существует два вида мьютекса: статический и экземпляры.

- Статический : блокирует изменение статический поле
- Экземплярный : блокирует изменение поле экземпляра

Потом в конец программа нужен делать join для каждого потока чтобы дождаться работа все потоков. Они динамический добавляется в течение работа программы, так что очередь

обеспечивает первый начавший поток join первым. (Join отличается от конец потока, так что программа все еще работает многопоточно).

Способ ввода

Выход через аргумента: сразу после название программа добавляете количество болтунов вы хотите добавить в программе. Другие ввода дает “Incorrect input”

Пример 4 болтунов: `./a.out 4`

function in mutex (lock)