Лабораторная работа 1

ДОПОЛНИТЕЛЬНЫЕ ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Примечание: ***поток*** *и* ***процесс*** в данной работе являются синонимами, обозначающими активную сущность, называемую в коде pintos как **thread**.

1	Реализуйте тест с именем "max-threads" в файле tests/threads/max-threads.c, измеряющий максимальное количество потоков, которое может быть в системе. Объясните полученный результат в отчете. Ответьте на вопрос: как количество памяти, доступной куче ядра, связано с максимальным количеством потоков в системе и почему?
2	Реализуйте тест "max-rec-calls" в файле tests/threads/max-rec-calls.c, измеряющий максимальное количество рекурсивных вызывов функции с 2 переменными для потока ядра. Объясните полученный результат в отчете. Ответьте на вопрос: как изменится и почему количество максимально возможных вызовов, если в struct thread изменить name[16] на name[128]. *Примечание:* Выполнение ядра, стек одного из потоков которого переполнился, будет нестабильным. Вероятнее всего, ядро будет завершено аварийно (Kernel panic), поэтому результат теста должен быть получен до останова. Для этого можно распечатывать количество достигнутых рекурсивных вызовов при каждом вложенном вызове. Последняя корректно распечатанная строка будет содержать полученный результат.
3	Реализуйте тесты "max-mem-malloc", "max-mem-calloc", "max-mem-palloc" в файле tests/threads/max-mem.c, измеряющий количество памяти, которая доступна в куче ядра. Тесты должны подсчитывать максимально возможное количество вызовов подряд, без освобождения памяти: - функции malloc, за каждый вызов выделяется 256 байт (тест "max-mem-malloc"); - функции calloc, за каждый вызов выделяется память для хранения массива int из 128 элементов (тест "max-mem-calloc"); - функции palloc, за каждый вызов запрашивается 1 страница ("max-mem-palloc"). Объясните полученные результаты в отчете.

4	Реализуйте в src/threads.c подсчет количества тиков процессора, потраченного каждым потоком, т.е. сколько тиков каждый поток был в состоянии RUNNING. Для этого можно ввести дополнительное поле в struct thread. Реализуйте тест "ticks-stats" в файле tests/threads/ticks-stats.c. Тест должен создавать 10 дополнительных потоков. Каждый из них имеет уникальное имя: "thread_%d", где %d - его номер от 1 до 10. Каждый поток исполняет пустой бесконечный цикл. Основная функция теста выполнят 10 итераций. Каждая итерация заключается в совершении паузы в 100 тиков, а после нее — распечатывания статистики. Статистика содержит сколько тиков отработал каждый из созданных им 10 потоков, а также сколько тиков отработал поток с основная функцией теста (initial thread). Укажите полученные результаты в отчете и объясните их.
5	Реализуйте систему аудита запуска и завершения потоков. Система аудита должна добавлять в журнал запись о запущенном потоке в момент его создания и о завершившемся потоке в момент его завершения. В записи хранятся: время создания/завершения потока (измеряется в тиках таймера от старта ОС), его имя и идентификатор tid_t. Журнал событий должен храниться в памяти в виде динамического списка. Добавление записей в список может быть реализовано в src/threads.c Реализуйте тест threads-audit в файле tests/threads/threads-audit.c. Тест создает 10 потоков. Каждый из потоков выполняет какую-либо вычислительную задачу или выполняет активное ожидание в течение небольшого интервала времени (1-3 сек). Разные потоки должны работать разное по продолжительности время. Главная функция теста создает эти потоки, ожидает 5 секунд, далее — распечатывает все зарегистрированные подсистемой аудита события. Укажите полученные результаты в отчете и объясните их.
6	Реализуйте функцию *void thread_terminate(tid t)* в файле threads/thread.c. Функция должна немедленно останавливать выполнение указанного по tid потока и удалять запись о нем из планировщика. Реализуйте тест threads-term в файле tests/threads/threads-term.c. Основная функция теста должна создавать 5 потоков. Каждый из потоков имеет уникальное имя "thread_%d", где %d - его номер от 1 до 5, поток выполняется в бесконечном цикле и периодически (раз в секунду) распечатывает на экран свое имя. Основная функция теста ожидает 4 секунды, затем останавливает с помощью функции thread_terminate потоки 1 и 3. Затем ожидает еще 4 секунды и завершается. Объясните в отчете полученный результат.
7	Реализуйте функции thread_pause(tid_t t) — безусловная приостановка потока (перевод в состояние BLOCKED) и thread_resume(tid_t t) — возобновление выполнения потока, ранее приостановленного с помощью thread_pause. Реализовать функции необходимо в файле threads/thread.c. Реализуйте тест threads-pause-resume в файле tests/threads/threads-pause-resume.c, демонстрирующий использование этих функций: Основная функция теста должна создать 2 потока с уникальными именами. Каждый из потоков исполняется в бесконечном цикле, периодически (раз в секунду) распечатывает в консоль свое имя. Основная функция теста после создания потоков ожидает 4 секунды, затем приостанавливает поток 1, ожидает еще 4 секунды, возобновляет его выполнение, ожидает еще 4 секунды и завершается. Укажите полученные результаты в отчете и объясните их.