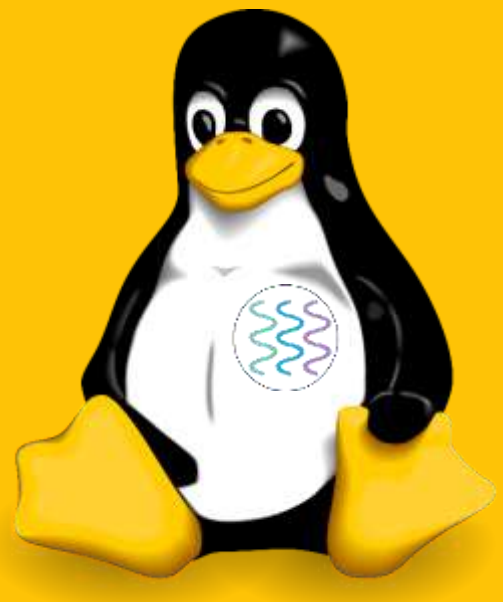
Лабораторная работа #11

Основы многозадачности. Потоки в Linux. POSIX Threads

[thread, POSIX стандарт, библиотека pthread,]



ЛАБОРАТОРНАЯ РАБОТА # 11

Основы многозадачности. Потоки в Linux. POSIX Threads

Цель работы

Изучить фундаментальные концепции, связанные с разработкой многопоточных приложений в ОС Linux.

Требования

1) С использованием стандартной библиотеки POSIX Threads разработать многопоточное приложение под ОС Linux, которое создаёт несколько потоков для выполнения соответствующих действий и предоставляет управление ими (задание можно придумать самостоятельно ☺ или выполнить соответствующее основное задание). Дополнительно предусмотреть вывод информации о всех имеющихся потоках программы.
2) Приложение должно содержать минимум три потока.
3) Для синхронизации потоков использовать мьютексы (***mutex***) или семафоры (***semaphore***).
4) Функции бизнес логики должны быть сгруппированы и вынесены в отдельную библиотеку.
5) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа ***Single Responsibility Principle***.
6) При выполнении задания можно использовать интегрированные средства разработки (*Integrated Development Environment, IDE*).
7) Для компиляции, компоновки и выполнения программы использовать средства автосборки (к примеру, ***Makefiles*** и утилиту **make**).
8) Для повышения производительности программы рекомендуется использовать функции динамического выделения памяти.

Основное задание

Приложение, разработанное при выполнении лабораторной работы № 8 (или других лабораторных работ), разбить на подзадачи, каждую из которых запустить в отдельном потоке в контексте одного процесса.

Пример разделения обязанностей между потоками программы:

- главный поток должен создавать дочерние потоки, устанавливать их приоритеты, запускать их, дожидаться их выполнения, завершать работу программы;
- первый дочерний поток должен осуществлять приём пользовательских данных (к примеру, целевых строк);
- второй дочерний поток вычислять основную бизнес-логику программы;
- третий и четвёртый дочерние потоки выводить соответственно результат выполнения в терминал и в файл.

Примеры самостоятельного задания

1) Создать многопоточное приложение, в котором создаются 4 потока, каждый из которых работает параллельно главному потоку и выполняют следующие действия:
   - первый поток: перемещает строку в горизонтальном направлении от одного края терминала до другого;
   - второй поток: случайным образом меняет содержимое строки;
   - третий поток: случайным образом меняет регистр содержимого сроки;
   - четвёртый поток: случайным образом меняет цвет строки.
2) Создать многопоточное приложение, которое создаёт массив целых значений и сортирует его разными методами (к примеру, Шелла, Хоара, пузырька, на основе бинарного дерева и др., но не менее 3). Для каждого вида сортировки создать отдельный поток выполнения. Приложение определяет эффективность использования сортировок путём определения времени выполнения потока.

Best of LUCK with it, and remember to HAVE FUN while you're learning :)
Victor Ivanchenko

Контрольные вопросы

1. Концепция многопоточности и понятие потока выполнения?
2. Концепция потоков в Linux?
3. Модели многопоточных приложений, их особенности и архитектура?
4. Виды и состояние потоков?
5. Стандарт POSIX?
6. Создание и управление потоками?
7. Использование библиотечных функций *pthread* (*pthread_create* – создание потока, *pthread_exit* – завершение потока, *pthread_join* – ожидание выполнения потока, *pthread_cancel* – отмена потока; *pthread_t* – структурный тип для хранения идентификатора потока)?
8. Зачем нужна поточная функция? Опишите её прототип.
9. Передача параметра в поточную функцию, получение результата?
10. Получение информации о потоках (использование библиотечных функций *pthread_self* и *pthread_equal)?*