Лабораторная работа №5

«Расширенные функции синхронизации потоков»

Цели: Детальное изучение возможностей стандартной библиотеки Pthreads, предназначенных для синхронизации потоков в ОС Linux.

Задачи: Получение практических навыков в области разработки специализированных функций с использование стандартной библиотеки Pthreads.

Срок выполнения: 2 недели

Общие сведения

Как было показано в Лабораторной работе №3, работа с мьютексами и условными переменными является блокирующей, т.е. при вызове в потоке одной из функций, например, `pthread_mutex_lock(…)` или `pthread_cond_wait(…)`, вызывающий поток блокируется до наступления одного из событий – освобождения мьютекса, либо сигнализации условной переменной. При этом не всегда удобна ситуация, когда вызываемый поток является заблокированным. В большинстве случаев, при наступлении ситуации взаимоблокировки (deadlock), программа должна быть принудительно завершена и, чаще всего, с потерей данных программы.

Использование не блокирующих вызовов функций работы с объектами синхронизации позволяют создавать более гибкие программы.

Следующие функции библиотеки Pthreads позволяют организовать работу с переменными синхронизации по-иному, чем стандартные функции:

Функция	`int pthread_mutex_trylock(pthread_mutex_t *mutex);`
Описание	Попытка захватить мьютекс. Однако если мьютекс уже захвачен другим потоком, этот вызов не будет блокировать вызывающий поток, а вернет управление немедленно. Эта функция может быть полезна для предотвращения взаимоблокировок.
Функция	`int pthread_cond_timedwait(pthread_cond_t *cond, pthread_mutex_t *mutex, const struct timespec *abstime);`
Описание	Блокировка вызывающего потока до вызова из другого потока функции `pthread_cond_signal` или `pthread_cond_broadcast`. Эта функция должна вызываться после захвата мьютекса; она освобождает мьютекс на все время ожидания. Если время ожидания превышает время, заданное в abstime, то происходит возврат управления вызывающему потоку. Время задается в абсолютном формате, как в функциях `time()` и `gettimeofday()`.

Пример использования функции ограниченного ожидания условной переменной:

```
pthread_mutex_lock(&t.mn);
          t.waiters++;
     clock_gettime(CLOCK_REALTIME, &ts);
     ts.tv_sec += 5;
```

```
        rc = 0;
        while (! mypredicate(&t) && rc == 0)
             rc = pthread_cond_timedwait(&t.cond, &t.mn, &ts);
        t.waiters--;
        if (rc == 0) setmystate(&t);
pthread_mutex_unlock(&t.mn);
```

Задания на лабораторную работу:

1) Напишите функцию, удовлетворяющую следующим условиям:
   a. Прототип функции:
      `int lab_pthread_mutex_trylock(pthread_mutex_t *mutex);`
   b. *Входной параметр*: адрес мьютекса
   c. Функция возвращает следующие значения:
      – **0** в случае, если мьютекс может быть захвачен без блокировки потока,
      – **1** если попытка захвата мьютекса с помощью функции pthread_mutex_lock(…) приведет к блокировке вызывающего ее потока,
      – **-1** в случае любой ошибки.

При реализации функции разрешается использовать стандартные структуры и функции библиотек ОС Linux, за исключением функции `pthread_mutex_trylock(…)`.

2) Напишите функцию, удовлетворяющую следующим условиям:
   a. Прототип функции:
      `int lab_pthread_cond_timedwait(pthread_cond_t *cond, pthread_mutex_t *mutex, unsigned int timetowait);`
   b. *Входные параметры*: адрес условной переменной, адрес мьютекса, количество миллисекунд, устанавливаемое для ожидания условной переменной.
   c. Функция возвращает следующие значения:
      – **0** в случае успешного выполнения функции,
      – **1** в случае, если время ожидания условной переменной превысило заданное в параметре `timetowait` количество миллисекунд,
      – **-1** в случае любой ошибки.

При реализации функции разрешается использовать стандартные структуры и функции библиотек ОС Linux, за исключением функции `pthread_cond_timedwait(…)`.

Содержание отчета по лабораторной работе:

1. Цели и задачи лабораторной работы.
2. Описание реализации заданий 1 и 2 в следующем виде:
   2.1. Текстовое описание работы каждой из функций 1 и 2.
   2.2. Блок-схемы алгоритмов реализации.
   2.3. Текст реализованных функций.
   2.4. Тестовые программы с использованием функций заданий 1 и 2.
3. Выводы.

Рекомендуемые источники:

1. Исходные тексты библиотеки Pthreads. Архив доступен на сайте http://vm.khstu.ru в разделе дисциплины «Операционные системы».