Архитектура ЭВМ. Операционные системы

Власов Е.Е.

Условные переменные

Условные переменные используются для того, чтобы заблокировать потоки до выполнения определенных условий. Условные переменные применяются в сочетании мьютексам, чтобы несколько потоков могли ожидать момента выполнения одного условия.

Сначала поток блокирует мьютекс, но и сам блокируется с помощью системного вызова ожидания условно переменной до момента выполнения условия. На то время, пока поток заблокирован, установленная им блокировка мьютекса автоматически снимается. Когда другой поток выполняет поставленное условие, он дает условной переменной сигнал (не имеющий отношения к сигналам unix) о разблокировании первого потока. После блокировки потока мьютекс автоматически устанавливается и первый поток повторно проверяет условие. Если оно не выполняется, поток опять блокируется переменной. Если условие выполняется, поток разблокирует мьютекс и выполняется дальше.

Условные переменные

Концептуально, условная переменная — это очередь потоков, ассоциированных с разделяемым объектом данных, которые ожидают выполнения некоторого условия, накладываемого на состояние данных. Таким образом, каждая условная переменная связана с утверждением Pc. Когда поток находится в состоянии ожидания на условной переменной, он не считается владеющим данными и другой поток может изменить разделяемый объект и просигнализировать ожидающим потокам в случае выполнения утверждения Pc.

Системные вызовы для работы условными переменными

Системный вызов	Описание
pthread_cond_init	Инициализирует атрибутный объект условной переменной
pthread_cond_timedwait pthread_cond_wait	Ожидание условия
pthread_cond_signal	Разблокирует поток, заблокированный вызовом pthread_cond_wait()
pthread_cond_broadcast	Разблокирует все потоки, заблокированные вызовом pthread_cond_wait()
pthread_cond_destroy	Уничтожает условную переменную

Системный вызов pthread_cond_init

```
#include <pthread.h>
int pthread_cond_init (pthread_cond_t *cond
const pthread_condattr_t *attr);

pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

Инициализирует атрибутный объект условной переменной, заданный параметром attr, значениями, действующими по умолчанию для всех атрибутов, определенных реализацией.

Системный вызов pthread_cond_wait

```
#include <pthread.h>
int pthread_cond_wait(pthread_cond_t *restrict cond,
pthread_mutex_t *restrict mutex);
```

При вызове `pthread_cond_wait mutex` должен быть захвачен, в противном случае результат не определен. `pthread_cond_wait` освобождает `mutex` и блокирует вызывающий поток до момента вызова другим потоком `pthread_cond_signal` или `pthread_cond_broadcast`. После пробуждения `pthread_cond_wait` пытается захватить `mutex`; если это не получается, он блокируется до того момента, пока `mutex` не освободят.

```c
/* Функция потока потребителя */
void *consumer(void *args)
{
    puts("[CONSUMER] thread started");
    int toConsume = 0;

    while(1)
    {
        pthread_mutex_lock(&mutex);
        /* Если значение общей переменной меньше максимального,
       * то поток входит в состояние ожидания сигнала о достижении
       * максимума */
        while (storage < STORAGE_MAX)
        {
            pthread_cond_wait(&condition, &mutex);
        }
        toConsume = STORAGE_MIN;
        printf("[CONSUMER][%x] storage is maximum, consuming %d\n", \
               (int)pthread_self(),\
               toConsume);
        /* "Потребление" допустимого объема из значения общей
       * переменной */
        storage -= toConsume;
        printf("[CONSUMER][%x] storage = %d\n",  (int)pthread_self(),storage);
        pthread_mutex_unlock(&mutex);
    }

    return NULL;
}
```

```c
/* Функция потока производителя */
void *producer(void *args)
{
    puts("[PRODUCER] thread started");

    while (1)
    {
        usleep(200000);
        pthread_mutex_lock(&mutex);
        /* Производитель постоянно увеличивает значение общей переменной */
        storage +=5;
        printf("[PRODUCER] storage = %d\n", storage);
        /* Если значение общей переменной достигло или превысило
         * максимум, поток потребитель уведомляется об этом */
        if (storage >= STORAGE_MAX)
        {
            puts("[PRODUCER] storage maximum");
            pthread_cond_broadcast(&condition);
        }
        pthread_mutex_unlock(&mutex);
    }
    return NULL;
}
```

Блокировки чтения-записи

Обычно чтение данных выполняется чаще, чем изменение и запись. В таких случаях можно заблокировать данные таким образом, чтобы несколько нитей могли одновременно считывать данные, и только одна нить могла их изменять. Для этого предназначена блокировка типа "несколько читателей, один писатель", или блокировка чтения-записи. Блокировка чтения-записи захватывается для чтения или записи, а затем освобождается. Освободить блокировку чтения-записи может только та нить, которая ее захватила.

Блокировки чтения-записи

Блокировки чтения-записи похожи на мьютексы, за исключением того, чтоони допускают более высокую степень параллелизма. Мьютексы могут иметь всего два состояния, закрытое и открытое, и только один поток может владеть мьютексом в каждый момент времени. Блокировки чтения-записи могут иметь три состояния: режим блокировки для чтения, режим блокировки для записи и отсутствие блокировки. Режим блокировки для записи может установить только один поток, но установка режима блокировки для чтения доступна нескольким потокам одновременно.

Если блокировка чтения-записи установлена в режиме *блокировки для записи*, все потоки, которые будут пытаться захватить эту блокировку, будут приостановлены до тех пор, пока блокировка не будет снята.

Если блокировка чтения-записи установлена в режиме *блокировки для чтения*, все потоки, которые будут пытаться захватить эту блокировку для чтения, получат доступ к ресурсу, но если какой-либо поток попытается установить режим блокировки для записи, он будет приостановлен до тех пор, пока не будет снята последняя блокировка для чтения. Различные реализации блокировок чтения-записи могут значительно различаться, но обычно, если блокировка для чтения уже установлена и имеется поток, который пытается установить блокировку для записи, то остальные потоки, которые пытаются получить блокировку для чтения, будут приостановлены. Это предотвращает возможность блокирования пишущих потоков непрекращающимися запросами на получение блокировки для чтения.

Блокировки чтения-записи еще называют *совместноисключающими* блокировками. Когда блокировка чтения-записи установлена в режиме для чтения, то говорят, что блокировка находится в режиме совместного использования.

Когда блокировка чтения-записи установлена в режиме для записи, то говорят, что блокировка находится в режиме *исключительного использования*.

Системные вызовы для БЧЗ

Системный вызов	Описание
`pthread_rwlock_init`	Создать БЧЗ
`pthread_rwlock_destroy`	Уничтожить БЧЗ

`pthread_rwlock_rdlock`
`pthread_rwlock_tryrdlock`
`pthread_rwlock_timedrdlock`
	Заблокировать БЧЗ на чтение

`pthread_rwlock_wrlock`
`pthread_rwlock_trywrlock`
`pthread_rwlock_timedwrlock`
	Заблокировать БЧЗ на запись

`pthread_rwlock_unlock`	Разблокировать БЧЗ