Средства синхронизации и приёмы их использования

Семафоры

Значение: $S \geq 0$

Операции:

- P – proberen / проверять / acquire

  P(S):

  пока S == 0 процесс блокируется;

  S = S – 1;

- V – verhogen / увеличивать / release

  V(S):

  S = S + 1;

Задача производитель-потребитель

```
Producer:
    while(1)
    {
        produce_item;
        put_item;
    }
Consumer:
    while(1)
    {
        get_item;
        consume_item;
    }
```

Задача производитель-потребитель

```
Semaphore mutex = 1;
Semaphore empty = N; /* где N – емкость буфера*/
Semaphore full = 0;
```

Производитель:

```
while(1)
{
    produce_item;
    P(empty);
    P(mutex);
    put_item;
    V(mutex);
    V(full);
}
```

Потребитель:

```
while(1)
{
    P(full);
    P(mutex);
    get_item;
    V(mutex);
    V(empty);
    consume_item;
}
```

Условные переменные Condition Variable

- Условная переменная – примитив синхронизации, позволяющая одним потокам дожидаться сигналов о наступлении событий от других потоков.

- Отличия от Event в Windows:
  - не имеет хранимого состояния: пробуждаются только те потоки, которые в момент активации ждали этой переменной;
  - работает в паре с Mutex'ом – мьютекс атомарно освобождается в момент начала ожидания и захватывается в момент его окончания. Мьютекс должен быть захвачен перед началом ожидания.

Условные переменные Condition Variable

Windows	C++ stdlib	Qt
Инициализация		
CRITICAL_SECTION cs; CONDITION_VARIABLE cv; InitializeCriticalSection(&cs); InitializeConditionVariable(&cv);	std::mutex m; std::condition_variable cv;	QMutex m; QWaitCondition cv;
Ожидание		

EnterCriticalSection(&cs);
SleepConditionVariableCS(&cv, &cs, время);
LeaveCriticalSection(&cs);
	std::unique_lock <std::mutex> lock(m);
	cv.wait(lock);
		m.lock();
		cv.wait(&m, время);
		m.unlock();

Разбудить один поток		
WakeConditionVariable(&cv);	cv.notify_one();	cv.wakeOne();
Разбудить всех		
WakeAllConditionVariable(&cv);	cv.notify_all();	cv.wakeAll();

Условные переменные и задача производитель-потребитель

```
data buffer[BufferSize];  /* круговой буфер */  int used = 0;
QWaitCondition bufferNotEmpty, bufferNotFull; QMutex mutex;
```

Производитель

```
for (int i = 0; ; ++i)
{
  mutex.lock();
  if (used == BufferSize)
        bufferNotFull.wait(&mutex);
  mutex.unlock();
  buffer[i % BufferSize] = …
  mutex.lock();
  ++used;
  bufferNotEmpty.wakeAll();
  mutex.unlock();
 }
```

Потербитель

```
for (int i = 0; ; ++i)
{
  mutex.lock();
  if (used == 0)
        bufferNotEmpty.wait(&mutex);
   mutex.unlock();
    … = buffer[i % BufferSize];
   mutex.lock();
   --used;
   bufferNotFull.wakeAll();
   mutex.unlock();
}
```

Условные переменные Ложное пробуждение

- При использовании условных переменных согласно API Windows и POSIX возможно **ложное пробуждение** – выход из ожидания без соответствующего вызова Wake.. из другого потока.

- При использовании условных переменных в таких случаях рекомендуется проверять условие после завершения ожидания:

```
while(!buf->full)
        wait(&buf->cond, &buf->lock);
```

Мониторы

```
monitor monitor_name
{
    описание внутренних
    переменных ;
    void m1(...){...   }
    void m2(...){...   }
    ...
    void mn(...){...   }
    {
    блок инициализации
    внутренних
    переменных;   }
}
```

Условные переменные/
Conditional variables

Операции:

Signal()

Wait()

Мониторы и задача производителя-потребителя

```
 monitor ProducerConsumer
{     condition full, empty;
      int count;
void put()
{
      if(count == N) full.wait;
      put_item;
      count += 1;
      if(count == 1) empty.signal;
}
void get()
{
      if (count == 0) empty.wait;
      get_item();
      count -= 1;
      if(count == N-1) full.signal;
}
{  count = 0;  }
}
```

```
Producer:
      while(1)
      {
            produce_item;
            ProducerConsumer.put();
      }

Consumer:
      while(1)
      {
            ProducerConsumer.get();
            consume_item;
      }
```