(+ в. 23-24)
Mutex –глобальный именованный объект ядра, который одновременно может принадлежать (быть используемым) только одному потоку: если объект Mutex захвачен каким-либо потоком, то остальные потоки не могут захватить его. С помощью объектов Mutex реализуется классический алгоритм критической секции. Доступ к объектам Mutex осуществляется посредством описателей. Общая схема такова: один из потоков порождает объект Mutex и присваивает ему глобальное имя при помощи функции CreateMutex. Остальные потоки могут использовать для получения описателя объекта OpenMutex или CreateMutex с тем же именем. Перед входом в критическую секцию поток вызывает одну из функций ожидания, передавая ей в качестве параметра описатель объекта Mutex. Если объект Mutex уже захвачен, то поток блокируется до освобождения объекта. Если не захвачен, то исполнение потока продолжается, а объект Mutex захватывается. Для освобождения объекта Mutex используется функция ReleaseMutex.
POSIX – mutex_init(), mutex_lock(), mutex_unlock(), mutex_destroy();
Windows – CreateMutex, ReleaseMutex, WaitForSingleObject

С помощью объектов **семафор** реализуется классический механизм семафоров. При этом в качестве примитива P(S) выступает одна из функций ожидания (уменьшает счетчик на единицу, но не меньше нуля, или ждет такой возможности), а в качестве примитива V(S) – функция ReleaseSemaphore (увеличивает счетчик семафора на заданную величину). При создании семафора функцией CreateSemaphore можно определить максимально допустимое значение переменной S.
POSIX – sem_init(), sem_wait(), sem_post(), sem_destroy().
Windows – ReleaseSemaphore, CreateSemaphore, WaitForSingleObject

События – самая примитивная разновидность синхронизирующих объектов. Они порождаются функцией CreateEvetnt и бывают двух типов с “ручным” (manual) и с “автоматическим” сбросом. Объект "событие" может находиться в двух состояниях: “занят” (non-signaled) и “свободен” (signaled). Для перевода объекта событие в свободное состояние используется функция SetEvent, а для перевода в занятое – ResetEvent. С помощью любой из функций ожидания можно перевести вызвавший поток в состояние блокировки до освобождения объекта событие. Если объект событие является объектом “с автоматическим сбросом”, то функция ожидания автоматически переведет его в состояние занятого. Объекты "событие" активно используются при асинхронном файловом вводе/выводе.

Барьер – это механизм синхронизации, который позволяет логически интегрировать результаты работы нескольких потоков, заставляя их ждать на определенной точке до тех пор, пока число заблокированных потоков не достигнет установленного значения. Только после этого все потоки деблокируются и могут продолжить свое исполнение. Когда специфицированный набор потоков достигнет барьера, все потоки деблокируются и продолжают исполнение.
Создание: int pthread_barrier_init(указатель к барьеру, атрибуты барьера, счетчик барьера). Счетчик count определяет число потоков, которые должны вызывать функцию int pthread_barrier_wait(pthread_barrier_t *barrier);
После создания барьера каждый поток будет вызывать эту функцию для указания на завершение этапа вычислений. При вызове данной функции поток блокируется до тех пор, пока число блокированных потоков не достигнет значения счетчика. После этого все потоки будут деблокированы. При создании барьера формируется указатель на объект атрибутов, который далее должен быть инициализирован. Атрибут разделения барьера может иметь 2 значения: PTHREAD_PROCESS_SHARED – разрешает барьеру синхронизировать потоки различных процессов; PTHREAD_PROCESS_PRIVATE (default) – разрешает барьеру синхронизировать потоки в пределах одного процесса.