Процесс (задача) - абстракция, описывающая выполняющуюся программу. Для ОС процесс представляет собой единицу работы, заявку на потребление системных ресурсов. Подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Состояния процессов
ВЫПОЛНЕНИЕ - активное сост, процесс непосредственно выполняется процессором;
ОЖИДАНИЕ - пассивное сост, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса;
ГОТОВНОСТЬ - пассивное сост, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.
В ходе жизненного цикла каждый процесс переходит из одного состояния в другое в соответствии с алгоритмом планирования процессов, реализуемым в ОС.

Контекст и дескриптор процесса
Создать процесс - это значит:
1. создать информационные структуры, описывающие данный процесс, то есть его дескриптор и контекст;
2. включить дескриптор нового процесса в очередь готовых процессов;
3. загрузить кодовый сегмент процесса в оперативную память или в область свопинга.

Алгоритмы планирования процессов
Планирование процессов включает в себя решение следующих задач:
1. определение момента времени для смены выполняемого процесса;
2. выбор процесса на выполнение из очереди готовых процессов;
3. переключение контекстов "старого" и "нового" процессов.
1, 2 - решаются программными средствами, 3 - аппаратно

В соответствии с алгоритмами, основанными на **квантовании**, смена активного процесса происходит, если:
• процесс завершился и покинул систему,
• произошла ошибка,
• процесс перешел в состояние ОЖИДАНИЕ,
• исчерпан квант процессорного времени, отведенный данному процессу.

Другая группа алгоритмов использует понятие **приоритет процесса.**

Существует две разновидности приоритетных алгоритмов: алгоритмы, использующие относительные приоритеты, и алгоритмы, использующие абсолютные приоритеты.

Вытесняющие и невытесняющие алгоритмы планирования
Существует два основных типа процедур планирования процессов - **вытесняющие** (preemptive) и **невытесняющие** (non-preemptive).
Non-preemptive multitasking - невытесняющая многозадачность - активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление планировщику ОС для того, чтобы тот выбрал из очереди другой, готовый к выполнению процесс.

Preemptive multitasking - вытесняющая многозадачность - решение о переключении процессора с выполнения одного процесса на выполнение другого процесса принимается планировщиком ОС, а не самой активной задачей.

Примеры классов:
Real time class - 24
High class – 13 - системные процессы, реагирующие на соответствующие события;
Above normal class – 10
Normal class – 8 - большинство процессов в системе, все процессы пользователя;
Below normal class – 6
Idle class – 4 –процесс активизируется только при простое других процессов;

Приоритет каждого потока складывается из приоритета его процесса и относительного приоритета самого потока.

Есть 7 относительных приоритетов потока:

1. Normal – такой же, как у процессов
2. Above Normal – «+1» к приоритету процесса
3. Bellow normal – «-1» к приоритету процесса
4. Highest – «+2»
5. Lowest – «-2»
6. Time Critical – устанавливает базовый приоритет потока для Real-Time классов – 31, для остальных – 15
7. Idle – для Real-Time классов – 16, для остальных – 1

Процессорное время выделяется потокам в соответствии с их уровнем приоритета. Уровни приоритетов варьируются в диапазоне от 0 (низший) до 31 (высший).

Распределение времени между потоками (управление приоритетами)
– Real time (реального времени) – некоторые системные процессы в "особых случаях".
Внутри классов приоритетов процессов определены уровни приоритетов потоков:
– низший, пониженный, нормальный, повышенный, высший, простаивающий, реального времени

Для управления приоритетами выполнения процессов и потоков служат следующие функции.
GetPriorityClass() – получение текущего класса приоритета для процесса.
SetPriorityClass() – установка класса приоритета для процесса.
GetThreadPriority() – получение текущего приоритета выполнения потока.
SetThreadPriority() – установка приоритета выполнения потока.