**Операции над процессами**

Количество таких состояний совпадает с количеством стрелок.

Все операции можно объединить в 3 пары:

1. Создание и завершение процесса
2. Запуск и приостановка процесса
3. Блокирование и разблокирование процесса

При этом первая пара является одноразовыми операциями (выполняются всего лишь 1 раз в течении его «жизни»)

Все остальные операции являются многоразовыми.

**Процесс ctrl блок (ПЦБ)   
(блок управления процессом) контекст процесса**

Каждый процесс представляется в ОС некоторой структурой данных. В общем случае в этой структуре данных содержится следующее:

1. Состояние, в котором находится процесс
2. Программный счётчик процесса
3. Содержимое регистров процессора
4. Данные, необходимые для планирования использования процессора и управления памятью (приоритет, адресное пространство, размер и т.д.)
5. Учётные данные процесса (идентификационный номер, идентификатор пользователя, запустившего процесс, время обработки и т.д.)
6. Сведения об устройствах ввода-вывода, связанных с процессом
7. Блок управления процессом (ПЦБ)

ПЦБ – некая модель процесса в ОС

Любая операция, которую выполняет ОС над процессом, должна найти своё отражение в ПЦБ.

Всю информацию в ПЦБ можно разделить на 2 части:

1. Содержимое всех регистров процессора (регистровый контекст процесса)
2. Вся остальная информация (системный контекст процесса)

Знания регистрового и системного контекстов процесса достаточно для того, чтобы управлять его работой в ОС, совершая над ним операции. Однако этого недостаточно для того, чтобы полностью охарактеризовать процесс.

ОС не интересует, что делает процесс. В то время как пользователя, который инициирует процесс, наоборот, интересует, что он делает. Поэтому код и данные, находящиеся в адресном пространстве процесса (в оперативной памяти) называются пользовательским контекстом. В любой момент времени процесс полностью характеризуется своим контекстом.

**Операции над процессами и связанные с ними понятия  
Одноразовые операции**

1. Рождение

ОС должна поддерживать возможность создания процессов. В простых системах все процессы могут рождаться сразу после старта.

В более сложных системах процессы могут создаваться динамически по мере необходимости.

Инициатором создания процесса может выступать процесс пользователя (совершая специальный системный вызов), либо сама ОС (представляющая некий процесс).

Процесс, инициировавший создание нового процесса, принято называть «процессом-родителем», а вновь созданный процесс «процессом-ребёнком»

Процессы-дети могут в свою очередь порождать новых детей внутри системы набор генеалогических деревьев процессов – генеалогический лес.

При рождении нового процесса, ОС заводит новый pcb с состояние процесса рождение и начинает его заполнять. Новый процесс получает собственный уникальный идентификационный номер. Поскольку для хранения идентификационного номера в ОС отводится ограниченное количество битов, то для соблюдения уникальности номеров количество одновременно присутствующих в ней процессов должно быть ограничено.

После завершения какого-либо процесса, его идентификационный номер может быть повторно использован для нового процесса.

Для выполнения своей функции, процессу «ребёнку» необходимы некоторые ресурсы (память, файлы, устройства ввода-вывода и т.д.)

Существуют 2 подхода выделения ресурсов:

1. Процесс ребёнок получает часть родительских ресурсов. При этом возможно деление с процессом-родителем
2. Новый процесс получает свои собственные ресурсы ОС. Информация о выделенных ресурсах заносится в pcb.

После этого в адресное пространство нужно занести программный код, данные счётчик.

Тоже существует два подхода:

1. Процесс потомок становится полным дубликатом процесса родителя по регистровому и пользовательскому контексту. Так же должен быть предусмотрен механизм определения кто родитель, а кто потомок.
2. Новый процесс загружается новой программой из какого-либо файла.

В Unix подобных системах возможен только 1 подход (в Linux в том числе)  
В Windows возможны оба подхода.

После наделения процесса необходимой информацией в pcb дописывается оставшаяся часть, после чего процесс переводится в состояние готовности.

1. Завершение работы

После завершения ОС переводит процесс в состояние завершения.

ОС должна освободить все связные с ним ресурсы, делая соответствующие изменения в его ПЦБ. При этом сам ПЦБ не уничтожается, а остаётся в системе ещё некоторое время (пока родитель не запросит информацию, или пока не завершится сам процесс «родитель») После этого вся информация будет удалена

Это связано с тем, что после «смерти» ребёнка процесс родитель может запросить ОС о причинах его смерти или ходе его работы.

В Unix подобных системах принято называть это процессами-зомби.

Если родительский процесс будет уничтожен, то потомки могут тоже завершиться или продолжить работу в зависимости от ОС.

Если потомки продолжают работать, у них меняется идентификатор родителя на специальный системный процесс.

**Многоразовые операции**

1. Запуск процесса

Из числа процессов, находящихся в состоянии готовности, ОС выбирает один процесс для исполнения.

Для этого процесса ОС обеспечивает наличие необходимой информации в оперативной памяти.

Состояние процесса меняется на состояние исполнителя, регистровый контекст из pcb восстанавливается в регистры процессора.

1. Приостановка процесса

При получении прерывания процессор сохраняет счётчик команд и значения некоторых регистров в стеке выполняемых процессов.

После чего управление передаётся по специальному адресу обработки прерывания. Сохраняется системный и регистровый контекст текущего процесса в его pcb.

Процесс переводится в состояние готовности.

После чего процессор переходит к обработке прерывания.

1. Блокирование процесса

Процесс обращается к ОС с помощью специального системного вызова. ОС обрабатывает данный системный вызов, добавляет этот процесс в очередь ожидания и при необходимости, сохранив нужную часть контекста в pcb, переводит его из состояния исполнения в состояние ожидания.

1. Разблокирование процесса

После возникновения в системе какого-либо события ОС должна определить, что за событие произошло.

После этого проверяется, есть ли процесс, ожидающий данное событие и находящийся в состоянии ожидания. Если такой процесс найден, то ОС переводит его в состояние готовности.

**Переключение контекста**

