Тема 5

Процессы в ОС Unix

Основы управления процессом

Для того, чтобы запустить программу на выполнение, ОС должна создать окружение или среду выполнения задачи, куда относятся ресурсы памяти, возможность доступа к устройствам ввода-вывода, различным системным ресурсам, включая услуги ядра.

Основы управления процессом

Процесс – это окружение (среда выполнения задачи).

Процесс состоит из:

- инструкций, выполняемых процессором,

- данных и информации о выполняемой задаче, такой как размещение в памяти, открытые файлы,

- статуса процесса.

Основы управления процессом

Программа может породить несколько процессов.

Многозадачность ОС говорит о том, что в ОС одновременно могут выполняться несколько процессов.

Выполнение процесса заключается в точном следовании набору инструкций. ***Процесс считывает и записывает информацию в раздел данных и в стек, но ему недоступны данные и стеки других процессов.***

Основы управления процессом

Однако процессы могут взаимодействовать между собой по ***средством средств межпроцессного взаимодействия, таких как разделяемая память, каналы, сигналы, семафоры, сообщения и обменники.***

Основы управления процессом

Типы процессов:

Системные процессы – например init. Системные процессы не имеют соответствующих им программ в виде исполняемых файлов и запускаются особым образом при инициализации ядра системы. Инструкции и данные для системных процессов находятся в ядре ОС.

Основы управления процессом

Демоны – это неинтерактивные процессы, которые запускаются обычным образом – путем загрузки в память соответсвующих им программ и выполняются в фоновом режиме. Обычно запускаются при инициализации системы и обеспечивают работу различных подсистем.

Основы управления процессом

Прикладные процессы. К прикладным процессам относятся все остальные процессы.

Основы управления процессом

Атрибуты процесса позволяют ОС эффективно управлять его работой:

- ***Идентификатор процесса Process ID (PID)*** – уникален, присваивается ядром ОС. Когда процесс завершает работу, ядро освобождает занятый им идентификатор.

- Идентификатор родительского процесса Parent Process ID (PPID) – ID родительского процесса.

- Приоритет процесса (Nice Number) – относительный приоритет процесса. Относительный приоритет не изменяется системой на всем протяжении жизни процесса (но может быть изменен пользователем или администратором).

Основы управления процессом

- ***Терминальная линия (TTY)*** – терминал или псевдотерминал, с которого запущен процесс. Процессы демоны не имеют ассоциированного терминала.

Основы управления процессом

Реальный (RID) и эффективный (EUID) идентификаторы пользователя. Реальным ИП данного процесса является идентификатор пользователя, запустившего процесс. Эффективный идентификатор служит для определения прав доступа процесса к системным ресурсам. Обычно они равны, но существует возможность задать процессу более широкие права, чем права пользователя, путем установки флага SUID, когда эффективному ID присваивается значение идентификатора администратора.

Основы управления процессом

- Реальный (RGID) и эффективный (EGID) идентификаторы группы. (флаг SGID).

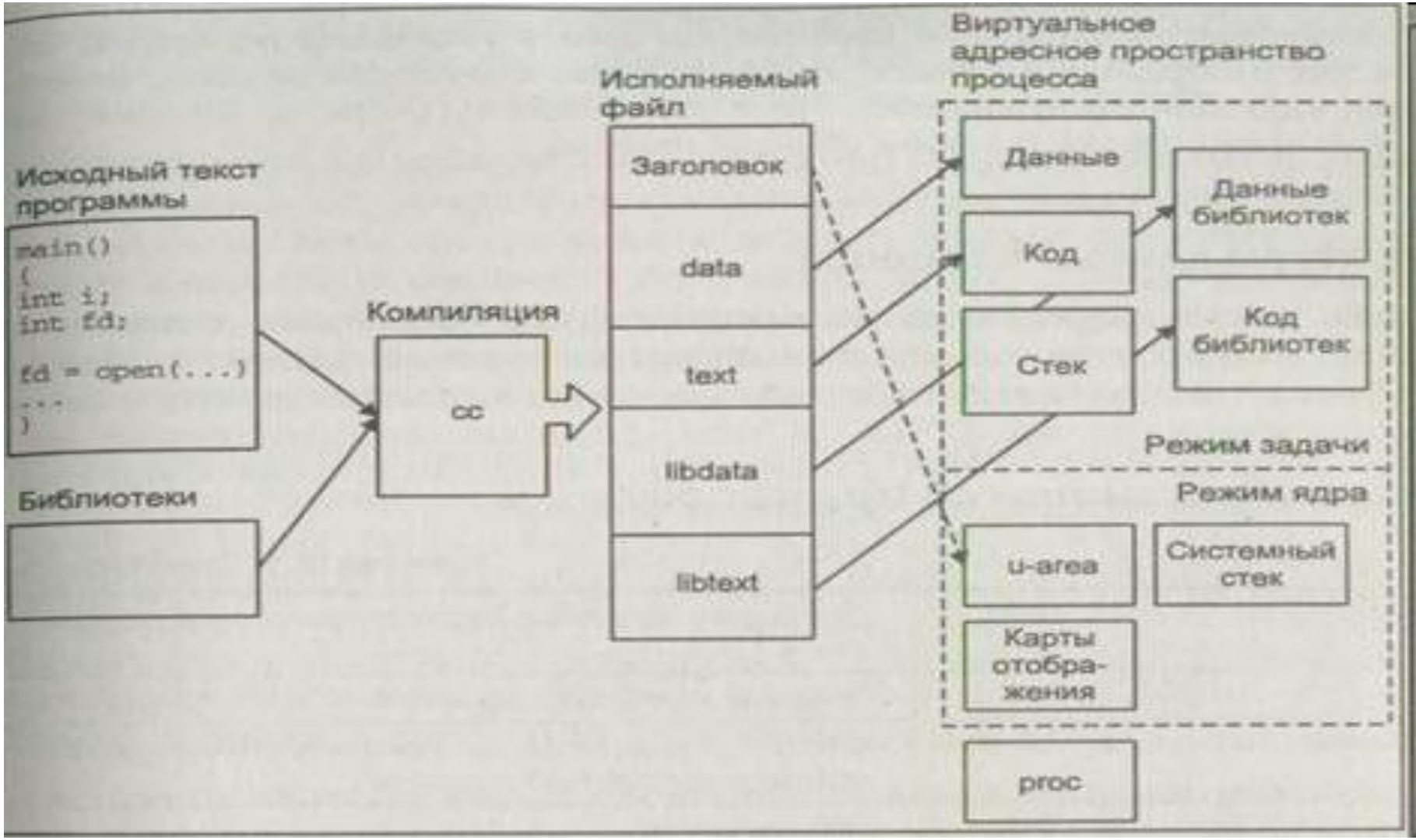
Основы управления процессом

На рис. схематически представлены компоненты, необходимые для создания и выполнения процесса. ***Процесс во время выполнения использует различные системные ресурсы -память, процессор, услуги файловой подсистемы и подсистемы ввода/вывода***.

Основы управления процессом

Операционная система UNIX обеспечивает иллюзию одновременного выполнения нескольких процессов, эффективно распределяя системные ресурсы между активными процессами и не позволяя в то же время ни одному из них монополизировать использование этих ресурсов.

Инфраструктура процесса операционной системы UNIX



Основы управления процессом

Выполнение процесса может происходить в двух режимах — в режиме ядра (kernel mode) или в режиме задачи (user mode).

В режиме задачи процесс выполняет инструкции прикладной программы, допустимые на непривилегированном уровне защиты процессора. При этом процессу недоступны системные структуры данных.

Основы управления процессом

Когда процессу требуется получение каких-либо услуг ядра, он делает системный вызов, который выполняет инструкции ядра, находящиеся на привилегированном уровне. Несмотря на то что выполняются инструкции ядра, это происходит от имени процесса, сделавшего системный вызов. Выполнение процесса при этом переходит в режим ядра.

Основы управления процессом

Таким образом ядро системы защищает собственное адресное пространство от доступа прикладного процесса, который может нарушить Целостность структур данных ядра и привести к разрушению операционной системы.

Часть процессорных инструкций, например, изменение регистров, связанных с управлением памятью, могут быть выполнены только в режиме ядра.

Основы управления процессом

Образ процесса состоит из двух частей:

- ***данных режима ядра***
- ***и режима задачи.***

Основы управления процессом

Образ процесса в режиме задачи состоит из:

- сегмента кода,

- данных,

- стека,

- библиотек

- других структур данных, к которым он может получить непосредственный доступ.

Основы управления процессом

Образ процесса в режиме ядра состоит из структур данных, недоступных процессу в режиме задачи, которые используются ядром для управления процессом.

Основы управления процессом

Сюда относятся:

- данные, диктуемые аппаратным уровнем, например состояния регистров,

- структуры данных, необходимые ядру для обслуживания процесса.

В режиме ядра процесс имеет доступ к любой области памяти.

Структуры данных процесса

Каждый процесс представлен в системе двумя основными структурами данных proc и user, описанными, соответственно, в файлах.

Содержимое и формат этих структур различны для разных версий UNIX.

Структуры данных процесса

В любой момент времени данные структур ргос для всех процессов должны присутствовать в памяти, хотя остальные структуры данных, включая образ процесса, могут быть перемещены во вторичную память, — область свопинга.

Структуры данных процесса

Это позволяет ядру иметь под рукой минимальную информацию, необходимую для определения местонахождения остальных данных, относящихся к процессу, даже если они отсутствуют в памяти.

Структуры данных процесса

Структура pгос является записью системной таблицы процессов, которая, всегда находится в оперативной памяти.

Запись этой таблицы для выполняющегося в настоящий момент времени процесса адресуется системной переменной curproc.

Структуры данных процесса

Каждый раз при переключении контекста, когда ресурсы процессора передаются другому процессу, соответственно изменяется значение переменной curproc, которая теперь указывает на структуру proc активного процесса.

Структуры данных процесса

Вторая структура — user, также называемая u-area или u-block, содержит дополнительные данные о процессе, которые требуются ядру только во время выполнения процесса (т. е. когда процессор выполняет инструкции процесса в режиме ядра или задачи).

Структуры данных процесса

В отличие от структуры proc, адресованной указателем curproc, данные user размещаются (точнее, отображаются) ***в определенном месте виртуальной памяти ядра и адресуются переменной***.

На рис. показаны две основные структуры данных процесса и способы их адресации ядром UNIX.

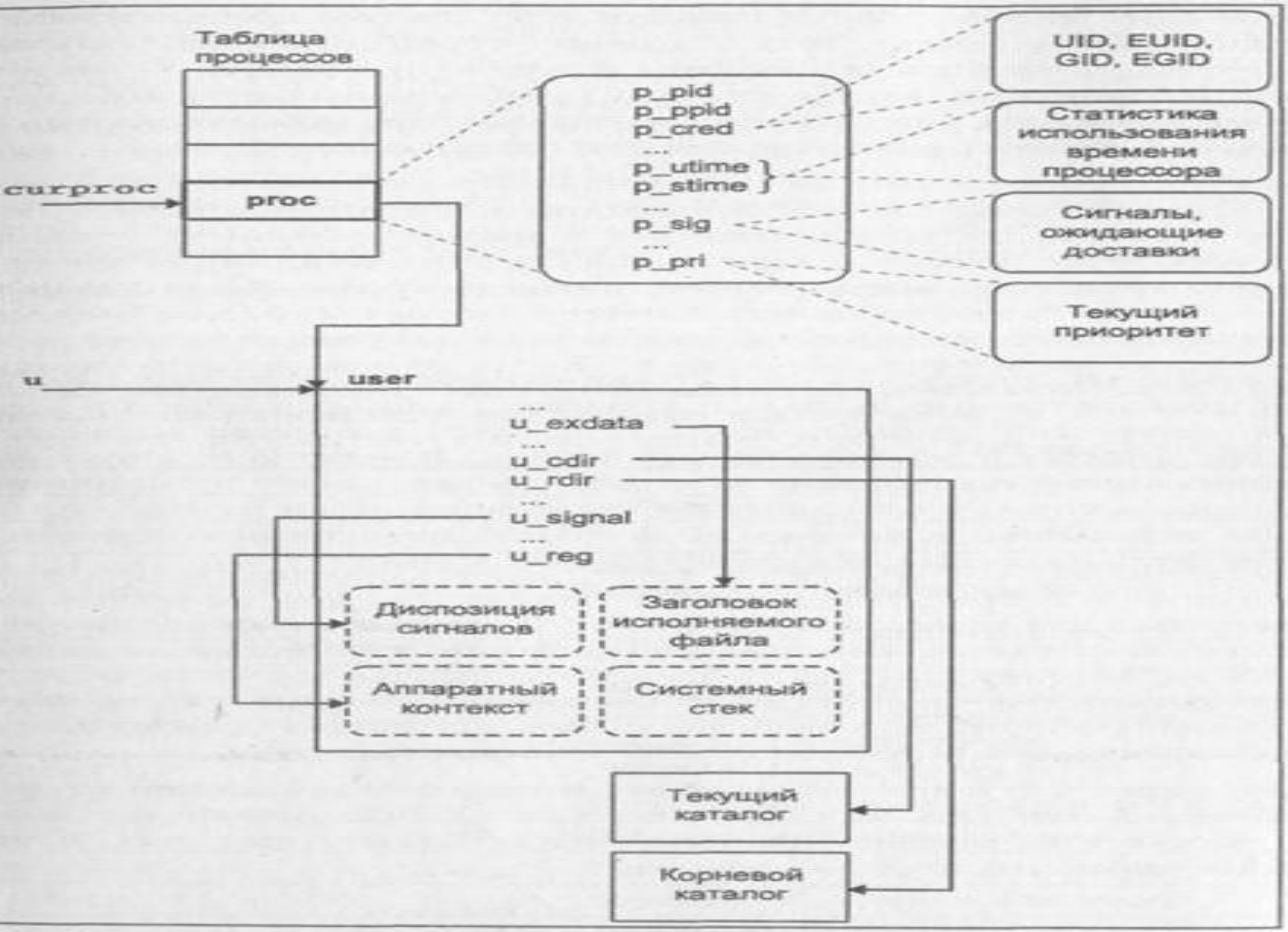


Рис. 3.2. Основные структуры данных процесса

Структуры данных процесса

В u-area хранятся данные, которые используются многими подсистемами ядра и не только для управления процессом. **В частности, там содержится информация об открытых файловых дескрипторах, диспозиция сигналов, статистика выполнения процесса, а также сохраненные значения регистров, когда выполнение процесса приостановлено.** Очевидно, что процесс не должен иметь возможности модифицировать эти данные произвольным образом, поэтому **u-area защищена от доступа в режиме задачи.**

Структуры данных процесса

u-area также содержит стек фиксированного размера, системный стек или стек ядра (kernel stack). ***При выполнении процесса в режиме ядра операционная система использует этот стек ядра, а не обычный стек процесса.***

Состояния процесса

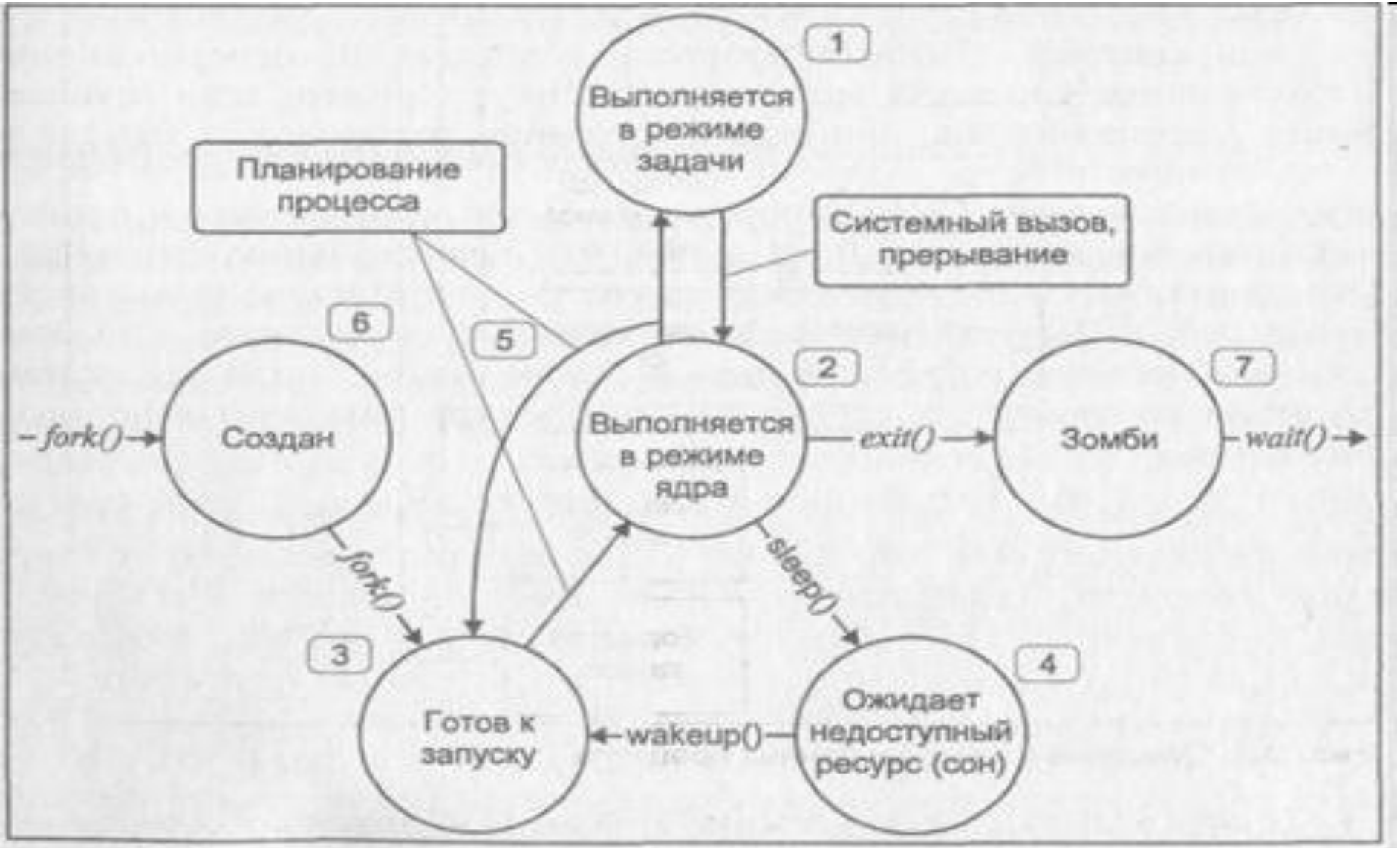


Рис. 3.3. Состояния процесса

Состояния процесса

Жизненный цикл процесса может быть разбит на несколько состояний. Переход процесса из одного состояния в другое происходит в зависимости от наступления тех или иных событий в системе. На рис. показаны состояния, в которых процесс может находиться с момента создания до завершения выполнения.

Состояния процесса

1. Процесс выполняется в режиме задачи. При этом процессором выполняются прикладные инструкции данного процесса.

2. Процесс выполняется в режиме ядра. При этом процессором выполняются системные инструкции ядра операционной системы от имени процесса.

Состояния процесса

3. Процесс не выполняется, но готов к запуску, как только планировщик выберет его (состояние runnable). Процесс находится в очереди на выполнение и обладает всеми необходимыми ему ресурсами, кроме

вычислительных.

4. Процесс находится в состоянии сна (sleep), ожидая недоступного в данный момент ресурса, например завершения операции вво да/вывода.

Состояния процесса

5. Процесс возвращается из режима ядра в режим задачи, но ядро прерывает его и производит переключение контекста для запуска более высокоприоритетного процесса.

6. Процесс только что создан вызовом fork() и находится в переходном состоянии: он существует, но не готов к запуску и не находится в со стоянии сна.

Состояния процесса

7. Процесс выполнил системный вызов exit() и перешел в состояние зомби (zombie, defunct). *Как такового процесса не существует, но остаются записи, содержащие код возврата и временную статистику его выполнения, доступную для родительского процесса*. Это состояние является конечным в жизненном цикле процесса.

Состояния процесса

Необходимо отметить, что не все процессы проходят через все множество состояний, приведенных выше.

Контекст процесса

Контекстом процесса *является его состояние, определяемое текстом, значениями глобальных переменных пользователя и информационными структурами, значениями используемых машинных регистров, значениями, хранимыми в позиции таблицы процессов и в адресном пространстве задачи, а также содержимым стеков задачи и ядра, относящихся к данному процессу.*

Команды ОС Unix для работы с процессами

ps - выдача информации о состоянии процессов.

ps –u

ps –t

ps –aux

ps –a

ps -e

Команды ОС Unix для работы с процессами

Улучшенный текущий контроль процессов: команда top.

Выдает регулярно обновляемую сводку активных процессов и используемых ими ресурсов.

Команды ОС Unix для работы с процессами

Изменение приоритета выполнения: команды nice и renice.

nice - выполнение команды с пониженным приоритетом

renice - изменение приоритета текущего процесса.

Команды ОС Unix для работы с процессами

Прекращение выполнения процессов: команда kill.

kill [-сигнал] pid,

где сигнал – номер или символическое имя посылаемого сигнала, а pid – идентификационный номер процесса-адресата. Команда kill без номера сигнала «не гарантирует», что процесс умрет, потому что сигнал TERM можно перехватить, блокировать и игнорировать. Команда kill –9 pid «гарантирует», что процесс умрет, потому что сигнал 9, KILL, другими процессами не перехватывается.

Команды ОС Unix для работы с процессами

Запуск процессов в фоновом режиме

Оболочка позволяет запустить процесс и, не дожидаясь его завершения, запустить другой. Чтобы это сделать, первый процесс должен быть запущен в фоновом режиме. Для запуска процесса в фоновом режиме используется &, который добавляется в конец командной строки.

Команды ОС Unix для работы с процессами

Когда примитивный интерпретатор команд завершает работу, он посылает во все порожденные им процессы сигнал «отбой». Если процесс выполняется в фоновом режиме, этот сигнал часто уничтожает его, что в некоторых случаях нежелательно. ***Если нужно запустить программу, которая будет работать и после вашего выхода из системы***, ее нужно запускать командой nohup.

nohup команда &