Для добавления заголовка щёлкните мышью

Планирование процессов в ОС UNIX

Классы приоритетов

В системе UNIX System V Release 4 реализована вытесняющая многозадачность, основанная на использовании приоритетов и квантования.
Все процессы разбиты на несколько групп, называемых классами приоритетов. Каждая группа имеет свои характеристики планирования процессов.
Созданный процесс наследует характеристики планирования процесса-родителя, которые включают класс приоритета и величину приоритета в этом классе. Процесс остается в данном классе до тех пор, пока не будет выполнен системный вызов, изменяющий его класс.

Приоритетный класс	**Выбор планировщика**	**Глобальное значение приоритета**
Реальное время (real time)	первый	159 . . . 100
Системные процессы (system)	. . . .	99 60
Процессы разделения времени (time-shared)	. . . . последний	59 0
Возможно добавление новых классов		

Состояния процесса в UNIX

Типы процессов

Системные процессы. Системные процессы являются частью ядра и всегда расположены в оперативной памяти. Системные процессы не имеют соответствующих им программ в виде исполняемых файлов и запускаются особым образом при инициализации ядра системы. Выполняемые инструкции и данные этих процессов находятся в ядре системы, таким образом они могут вызывать функции и обращаться к данным, недоступным для остальных процессов. Системными процессами являются: shed (диспетчер свопинга), vhand (диспетчер страничного замещения), bdfflush (диспетчер буферного кэша) и kmadaemon (диспетчер памяти ядра). К системным процессам следует отнести init, являющийся прародителем всех остальных процессов в UNIX. Хотя init не является частью ядра, и его запуск происходит из исполняемого файла (/etc/init), его работа жизненно важна для функционирования всей системы в целом.

Типы процессов

Демоны. Демоны — это неинтерактивные процессы, которые запускаются обычным образом — путем загрузки в память соответствующих им программ (исполняемых файлов), и выполняются в фоновом режиме. Обычно демоны запускаются при инициализации системы (но после инициализации ядра,) и обеспечивают работу различных подсистем UNIX: системы терминального доступа, системы печати, системы сетевого доступа и сетевых услуг и т. п. Демоны не связаны ни с одним пользовательским сеансом работы и не могут непосредственно управляться пользователем. Большую часть времени демоны ожидают пока тот или иной процесс запросит определенную услугу, например, доступ к файловому архиву или печать документа.

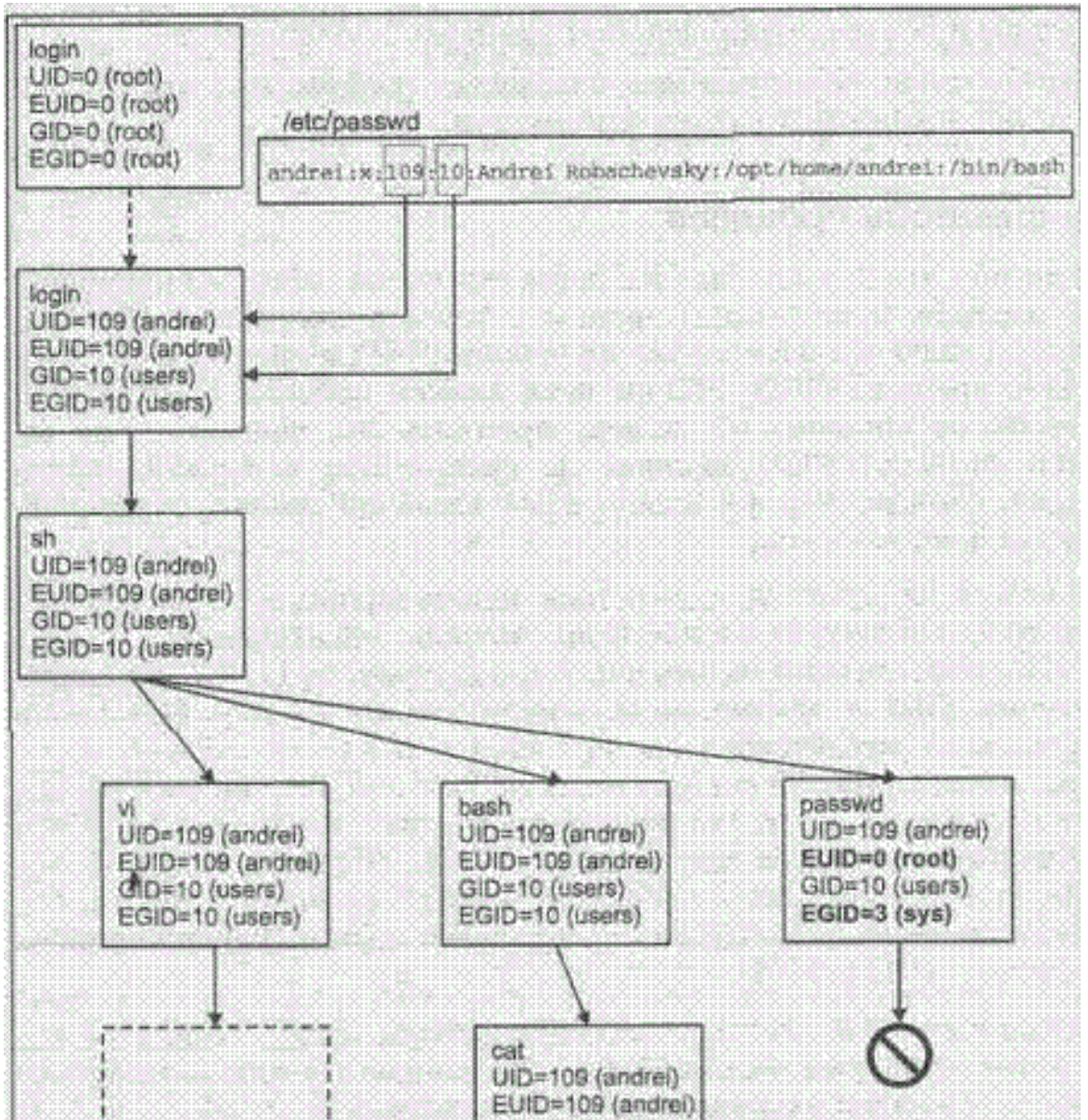
Типы процессов

Прикладные процессы. К прикладным процессам относятся все остальные процессы, выполняющиеся в системе. Как правило, это процессы, порожденные в рамках пользовательского сеанса работы. С такими процессами вы будете сталкиваться чаще всего. Например, запуск команды ls(l) породит соответствующий процесс этого типа. Важнейшим пользовательским процессом является основной командный интерпретатор (login shell), который обеспечивает вашу работу в UNIX. Он запускается сразу же после вашей регистрации в системе, а завершение работы login shell приводит к отключению от системы.

Атрибуты процесса

- Идентификатор процесса Process ID (PID).
- Идентификатор родительского процесса Parent Process ID (PPID).
- Приоритет процесса (Nice Number).
- Терминальная линия (TTY).
- Реальный (RGID) и эффективный (EGID) идентификаторы группы.

Процесс наследования процессов



Получение значений идентификаторов процесса

```
#include <sys/types.h>
#include <unistd.h>
uid t getuid(void);

uid_t geteuid(void);

gid t getgid(void);

gid_t getegid(void);
```

Изменение значения идентификаторов

```
#include <sys/types.h>

#include <unistd.h>
int setuid(uid_t uid);

int setegid(gid_t egid);

int seteuid(uid_t euid) ;

int setgid(gid_t gid);
```

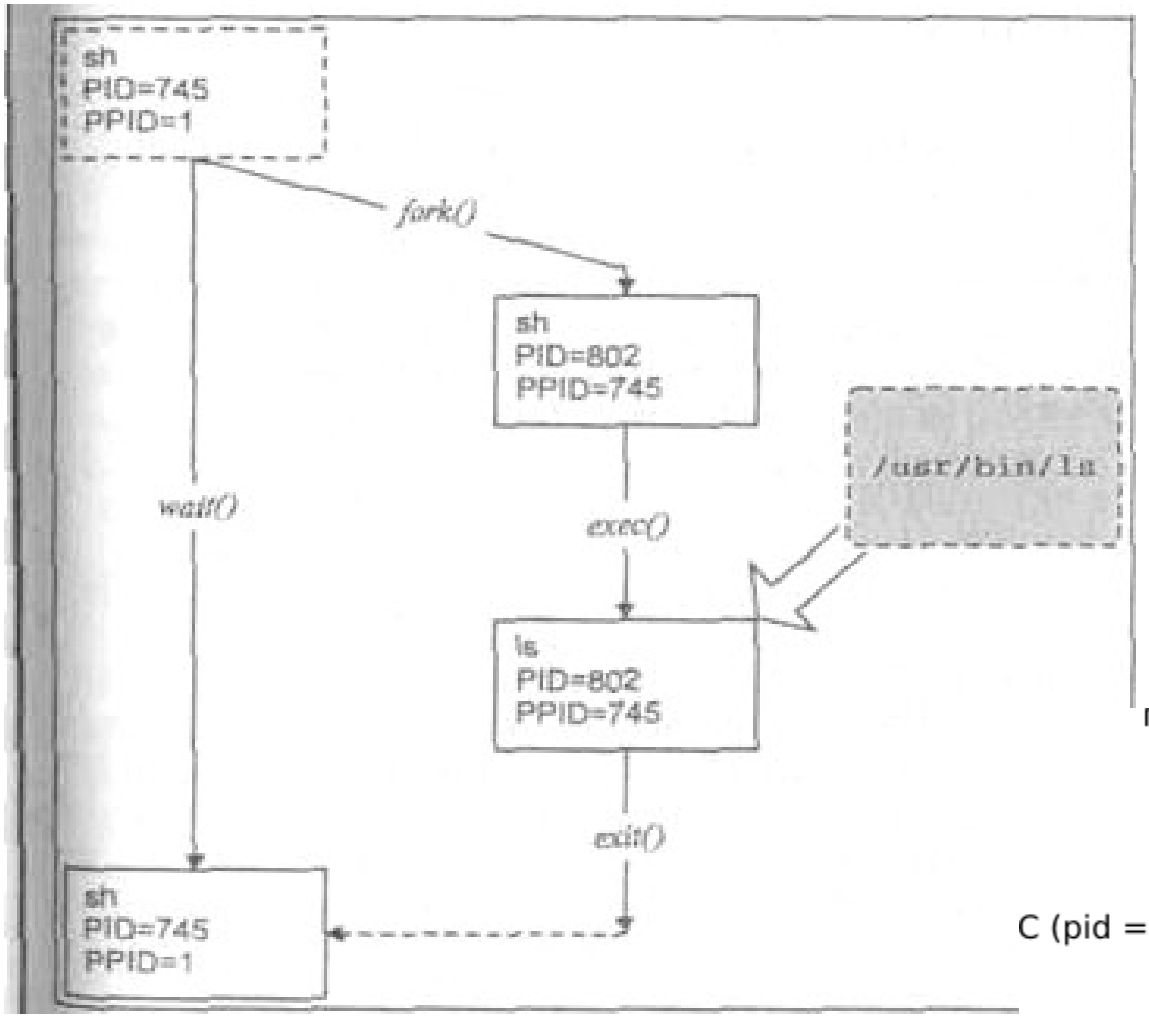
Жизненный путь процесса

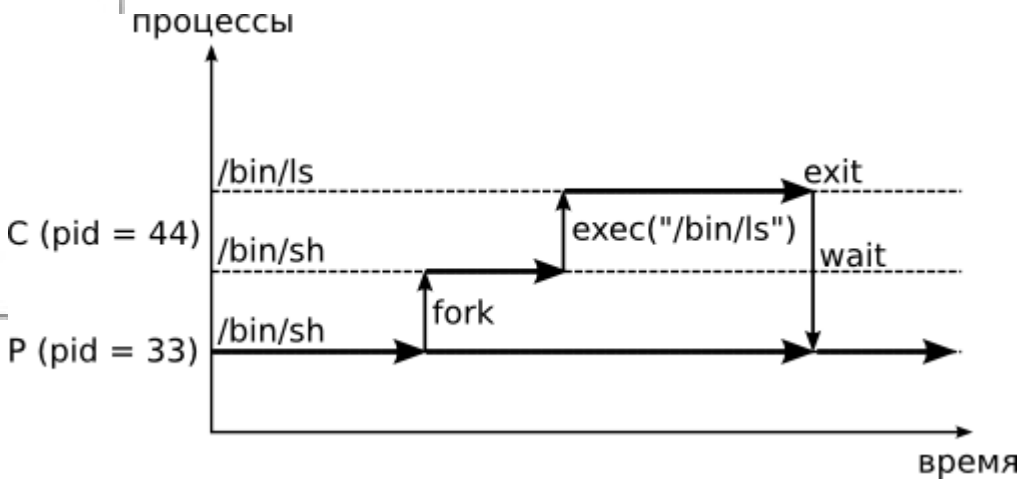
Новый процесс в UNIX порождается с помощью системного вызова fork(2):

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork (void);
```

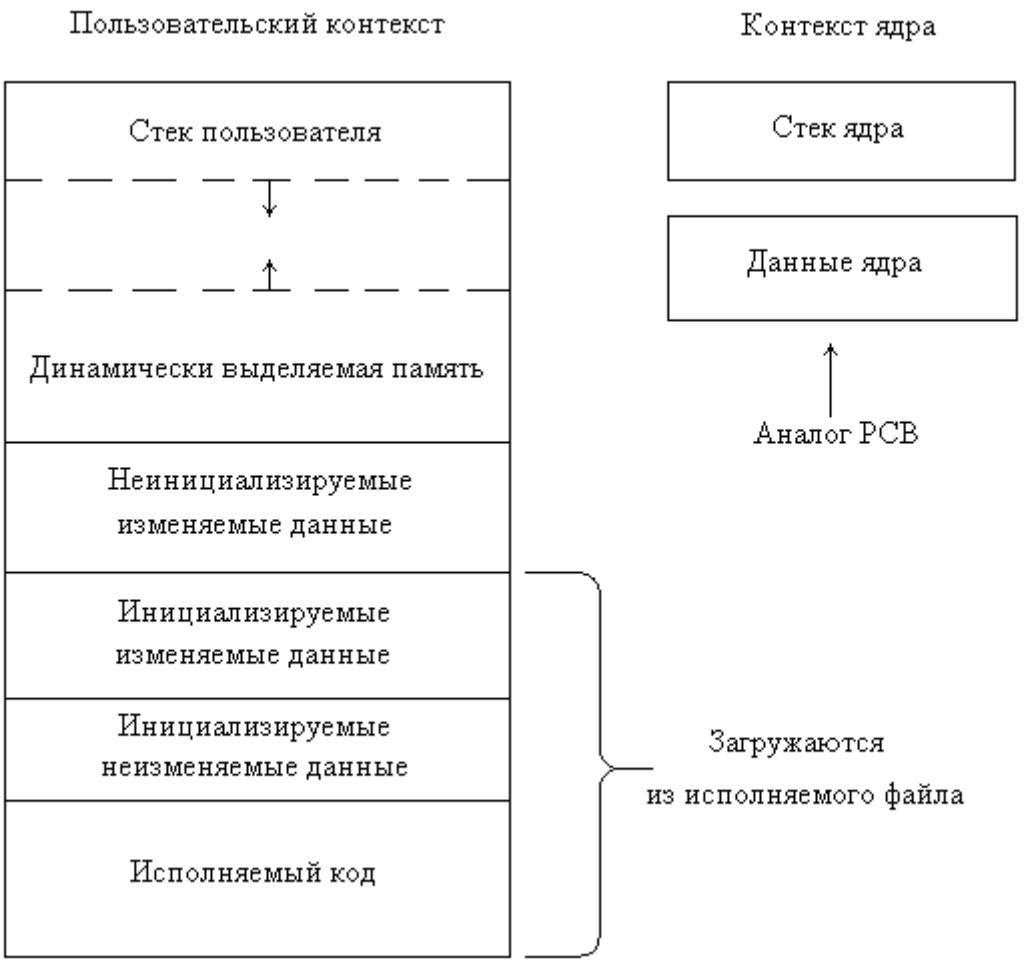
Для запуска задачи, т. е. для загрузки новой программы, процесс должен выполнить системный вызов *exec(2).* При этом новый процесс не порождается, а исполняемый код процесса полностью замещается кодом запускаемой программы.

Жизненный путь процесса



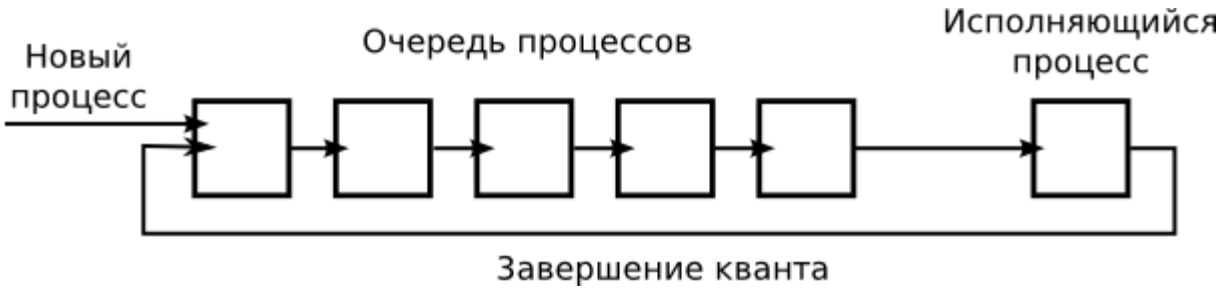


Контекст процесса в Unix



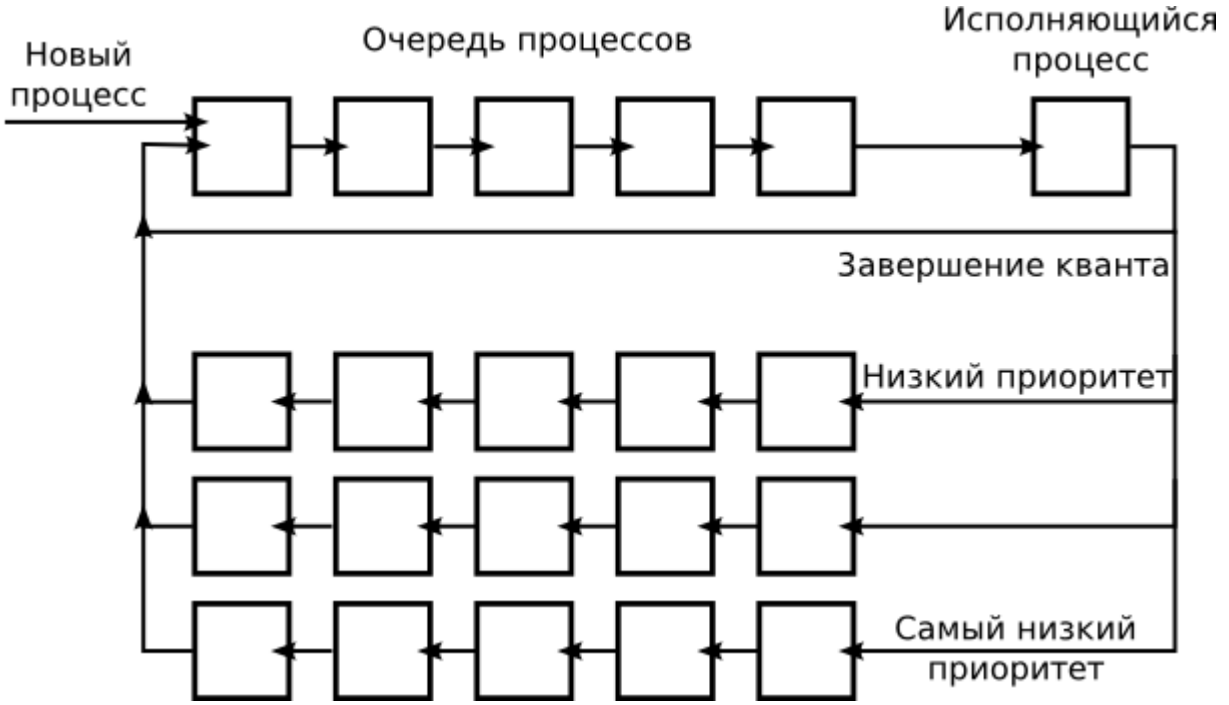
Планирование процессов

Схема планирования с кольцевой очередью



Планирование процессов

Схема планирования с кольцевой очередью и приоритетами



Межпроцессное взаимодействие

Для решения задачи межпроцессного взаимодействия в операционной системе UNIX существует набор специальных средств: потоки ввода-вывода, переменные окружения, каналы и сокеты, разделяемая память и сигналы

способы взаимодействия	**тип связи**	**способ применения**	**системные вызовы**
разделяемая память	общий доступ	высокопроизводит. обмен данными	shmem, mmap
переменные окружения	односторонняя, при запуске	режим работы программы	setenv
сигналы	односторонняя	уведомления	signal
каналы	односторонняя	базовый ввод-вывод	pipe
сокеты	двусторонняя	сетевой обмен	socket, ...

Разделяемая память