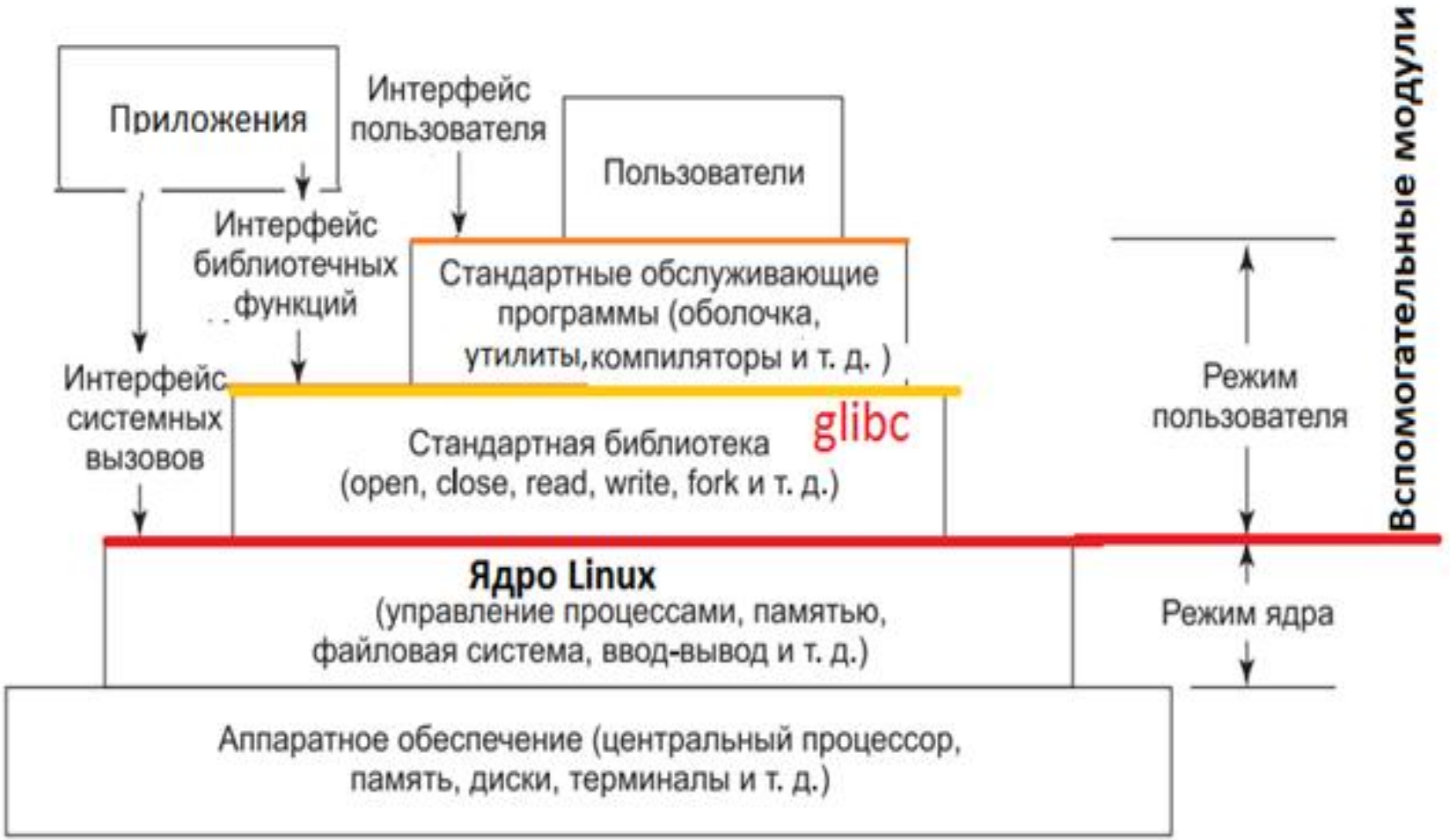
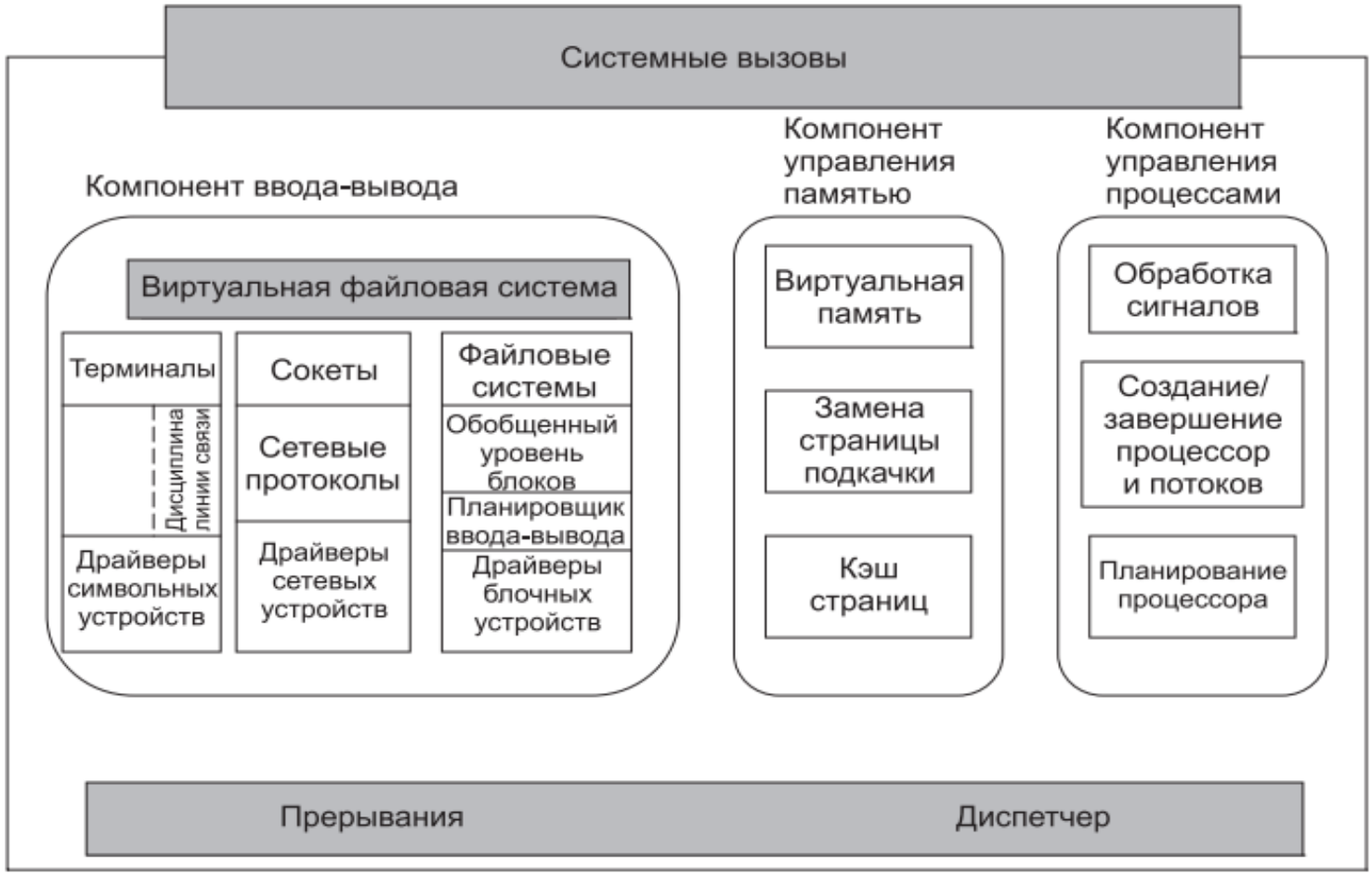
Архитектура ОС

Операционная система

Ядро и вспомогательные модули



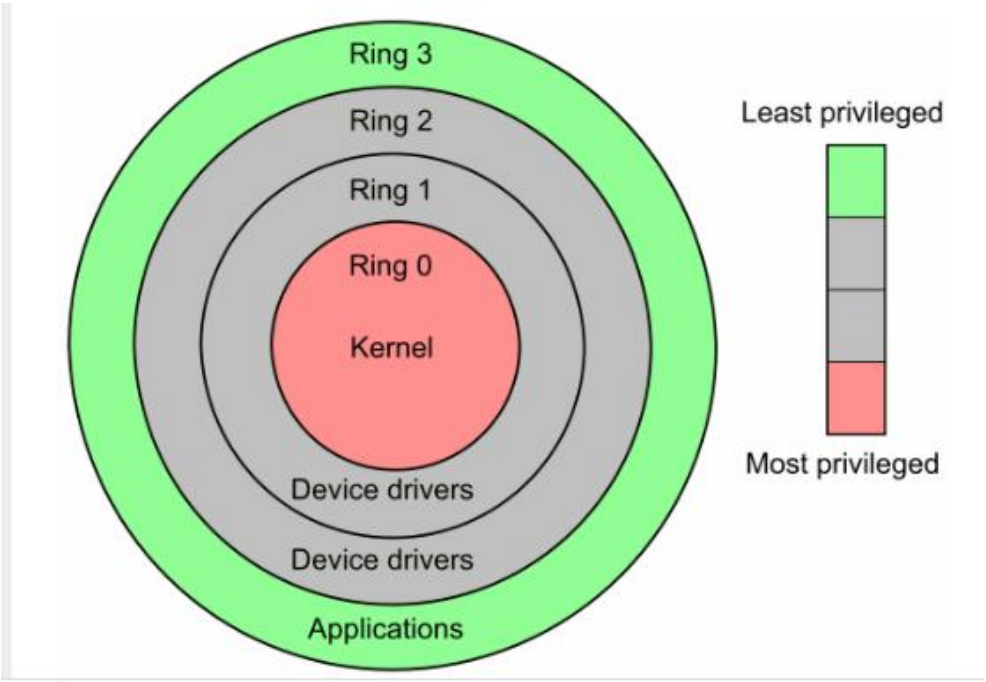
Ядро Linux


Системные вызовы
Компонент ввода-вывода
Виртуальная файловая система
Терминалы
Дисциплина линии связи
Драйверы символьных устройств
Сокеты
Сетевые протоколы
Драйверы сетевых устройств
Файловые системы
Обобщенный уровень блоков
Планировщик ввода-вывода
Драйверы блочных устройств
Компонент управления памятью
Виртуальная память
Замена страницы подкачки
Кэш страниц
Компонент управления процессами
Обработка сигналов
Создание/ завершение процессор и потоков
Планирование процессора
Прерывания
Диспетчер

Состав ОС

- Ядро. Основные функции ядра:
  - Управление процессами;
  - Управление памятью;
  - Управление файлами;
  - Управление вводом /выводом;
  - Предоставление программного интерфейса для доступа к ядру (API)
- Оболочка (командный процессор -для приема команд пользователя)
- Дополнительные утилиты (для выполнения вспомогательных функций : тестирование шифрование, дефрагментация, форматирование носителей и др.)
- Системные библиотеки
- Драйверы (включается в ОС, но не являются её составной частью)

Режимы выполнения программ



- Задаются содержимым системных регистров : для x80_64 EFLAGS, CR0, CR3, CR4, GDTR, LDTR, MSR и др.
- Большинство ОС используют только два уровня (**Privilege Level)** PL0 – режим ядра
- PL3 – режим пользователя

Режим пользователя и режим ядра

- Отличаются
  - Количеством выполняемых инструкций
  - Диапазоном адресуемой памяти

- В режиме пользователя :
  - Ограниченный набор инструкций (запрещены инструкции: меняющие состояние процессора, разрешения прерываний, ввода-вывода (in/out)
  - Ограниченный диапазон адресуемой памяти

В **режиме ядра** доступны **все** инструкции и **все** поле памяти

Правило перехода

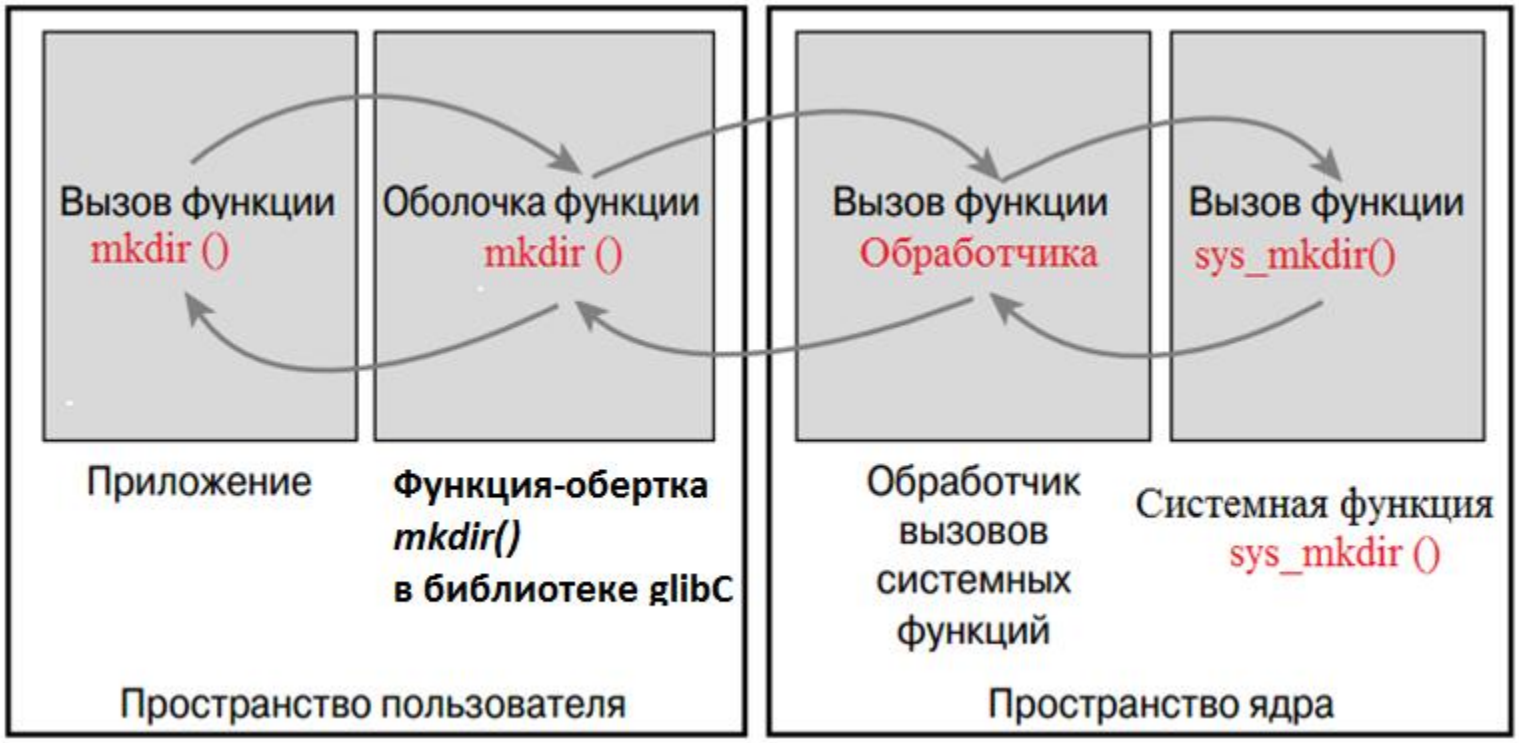
- Процессор переводится из режима ядра в режим пользователя путем установки управляющих бит в системных регистрах.

- Обратный переход из режима пользователя в режим ядра через системный вызов.

Реализации вызова системных функций в Linux

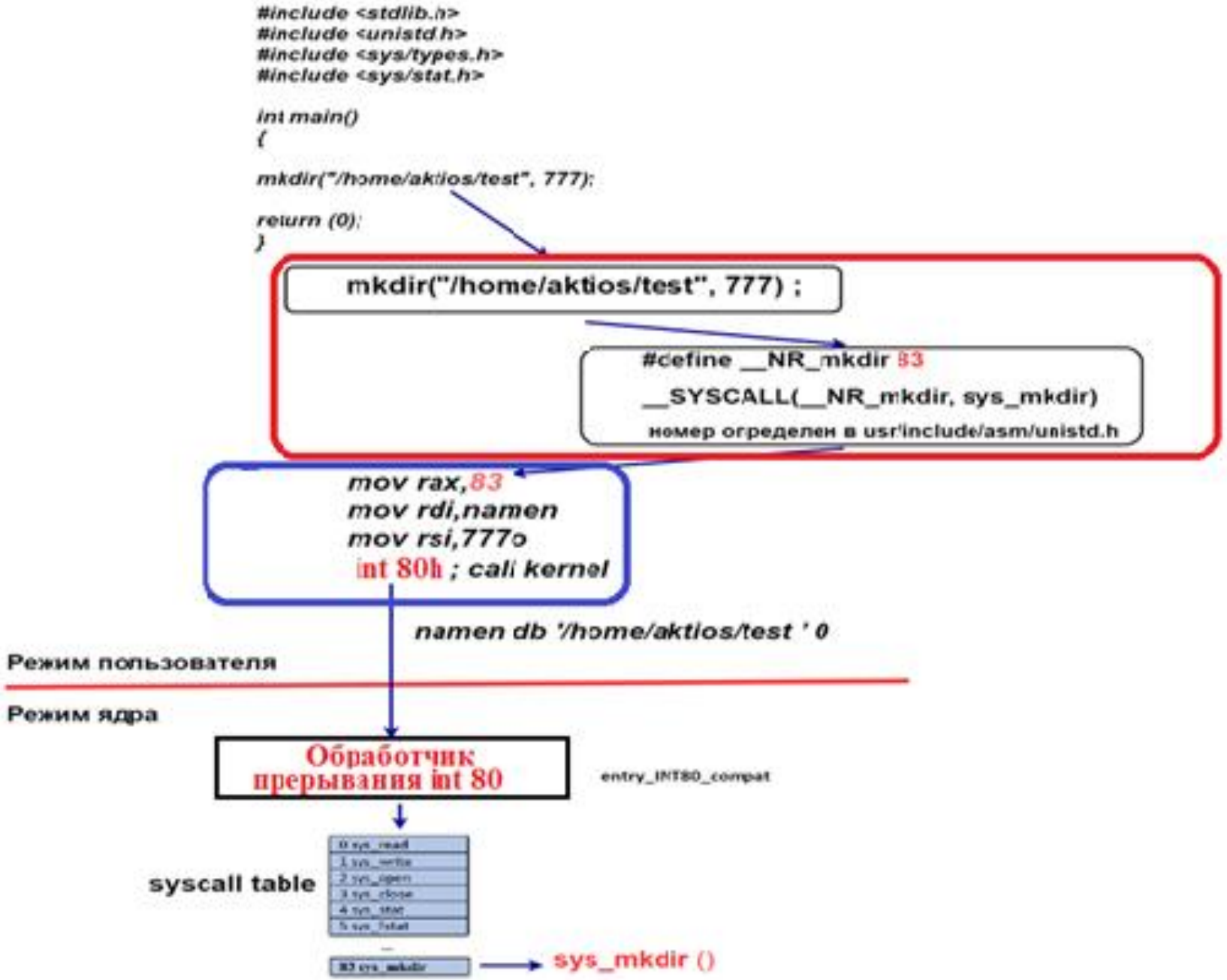
- int 80h
- sysenter/sysexit (Intel) (32)
- syscall/sysret (AMD) (64)

Системный вызов

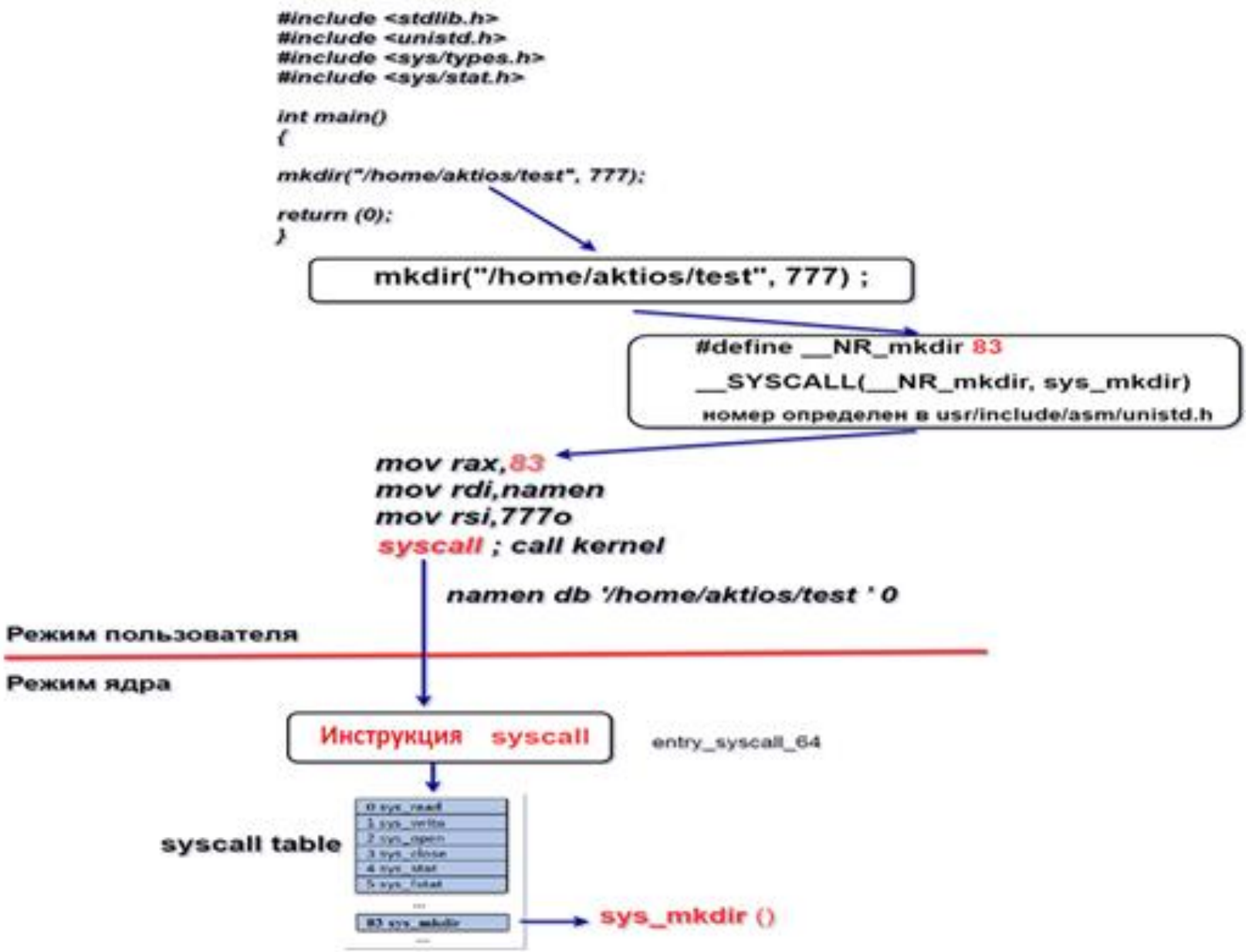


Каждому системному вызову соответствует библиотечная функция обертка в glibc
Функция обертка вызывает низкоуровневую функцию, которая помещает номер и параметры системного вызова в регистры процессора,
Обработчик системного вызова использует информацию в этих регистрах для вызова требуемой системной функции.

Int 80h


#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
int main()
{
mkdir("/home/aktios/test", 777);
return (0);
}
mkdir("/home/aktios/test", 777) ;
#define __NR_mkdir 83
__SYSCALL(__NR_mkdir, sys_mkdir)
номер определен в usr\include/asm/unistd.h
mov rax,83
mov rdi,namen
mov rsi,777o
int 80h ; cali kernel
namen db '/home/aktios/test ' 0
Режим пользователя
Режим ядра
Обработчик прерывания int 80
entry_INT80_compat
syscall table
sys_mkdir ()

Машинная инструкция процессора syscall

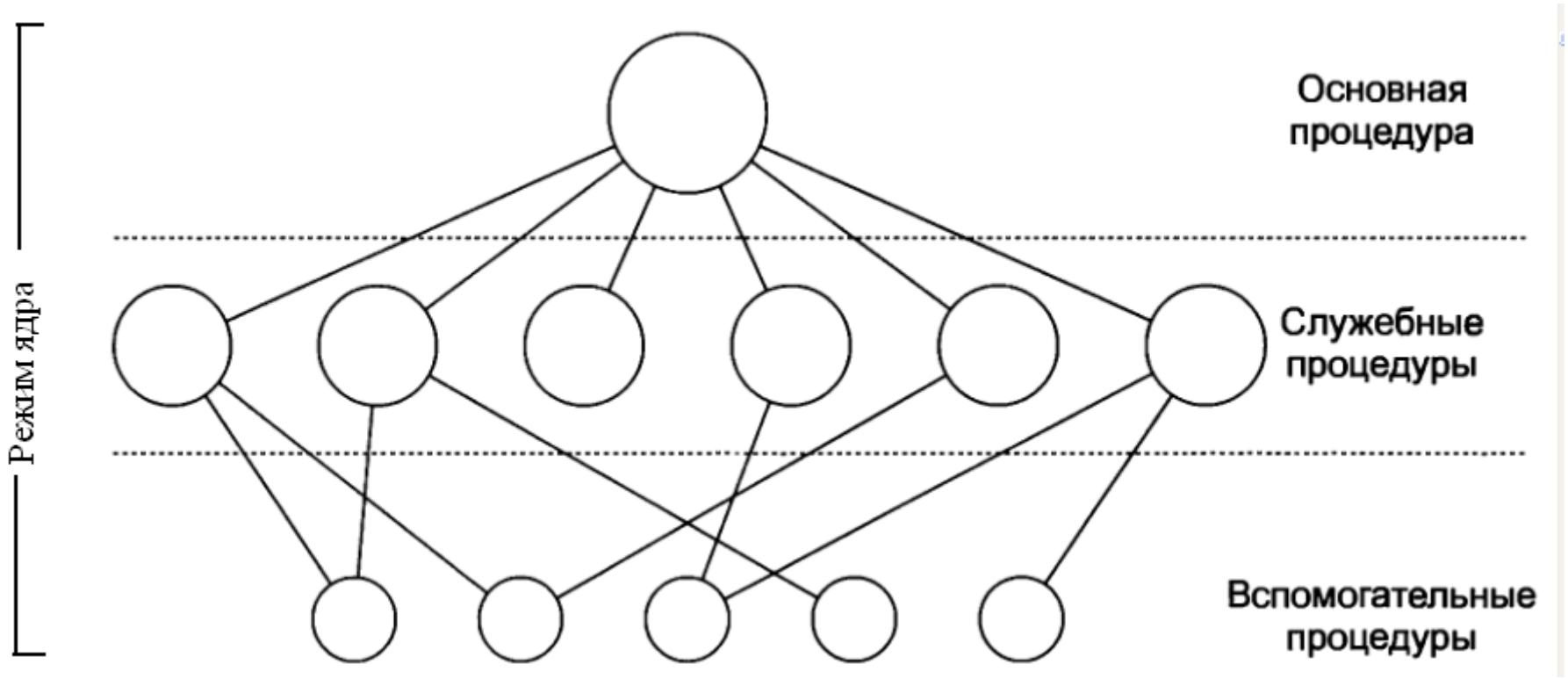


Относительная оценка времени работы

- int 80h — 500 нс,
- sysenter — 340 нс,
- syscall — 280 нс,
- вызов обычной С-функции — единицы нс.

- syscall()
  - Специальная функция-обертка (определена в #include <sys/syscall.h>)
  - Позволяет сделать сделать системный вызов без использования (отсутствия) библиотечных функций-обертки.

Архитектура ОС
Монолитная архитектура



Одна большая программа все части которой работают в режиме ядра (постоянно находятся в памяти) и могут вызывать друг друга **Монолитное ядро** состоит из единственного загрузочного модуля, в котором реализованы все основные функции ОС

Монолитное ядро

- Монолитное ядро
  - работает в привилегированном режиме (PL=0)
  - в едином адресном пространстве
  - не вытесняется из памяти пользовательскими процессами

- Достоинства :
  - Простое и быстрое взаимодействие между компонентами ядра (на уровне вызова функций) повышает производительность

- Недостатки
  - Выход из строя одного компонента ОС может привести к краху системы
  - Большой объем кода, постоянно находящегося в памяти требует больших размеров ОЗУ и негативно влияет на время реакции ОС
  - Любые изменения (аппаратные), добавления требуют перекомпиляции ядра

Монолитное ядро

- Плотность программных ошибок зависит от размера модуля. На 1000 строк кода приходится в среднем 10 ошибок.

- Монолитная операционная система, состоящая из 5 000 000 строк кода, содержит около 50 000 ошибок ядра.
- Примеры : ранние ядра Unix/Linux

- Для справки:
  - Linux версии 0.01 (10 239 строк кода)
  - Linux 5.0 (2019г, более 26 млн строк кода)

Монолитное ядро

- Ограничения монолитной архитектуры
  - Расширяемость
  - Переносимость (на разные аппаратные платформы)
  - Совместимость (возможность выполнять программ ориентированных на более ранние версии или сторонние ОС)

Монолитно - модульные ОС

- Содержит (статическую) монолитную и динамическую часть (подгружаемые модули).
- Монолитная часть всегда находится в памяти
- Динамические модули загружаются или выгружаются из памяти ядра при необходимости
- Возможна автоматическая и ручная загрузка/ выгрузка модулей.
- При добавлении модулей ядро не надо перекомпилировать.
- При изменениях в модуле перекомпилируется только сам модуль

Монолитнно-модульная архитектура

- Достоинства :
  - Быстродействие от монолитной
  - Модульность
  - Меньше места в памяти

- Монолитная часть ядра Linux в файле /boot/vmlinuz
  - vm – поддержка виртуальной памяти
  - z – запакован самораспаковывающим *gzip* – архивом

- Динамические модули ядра в папке - /lib/modules

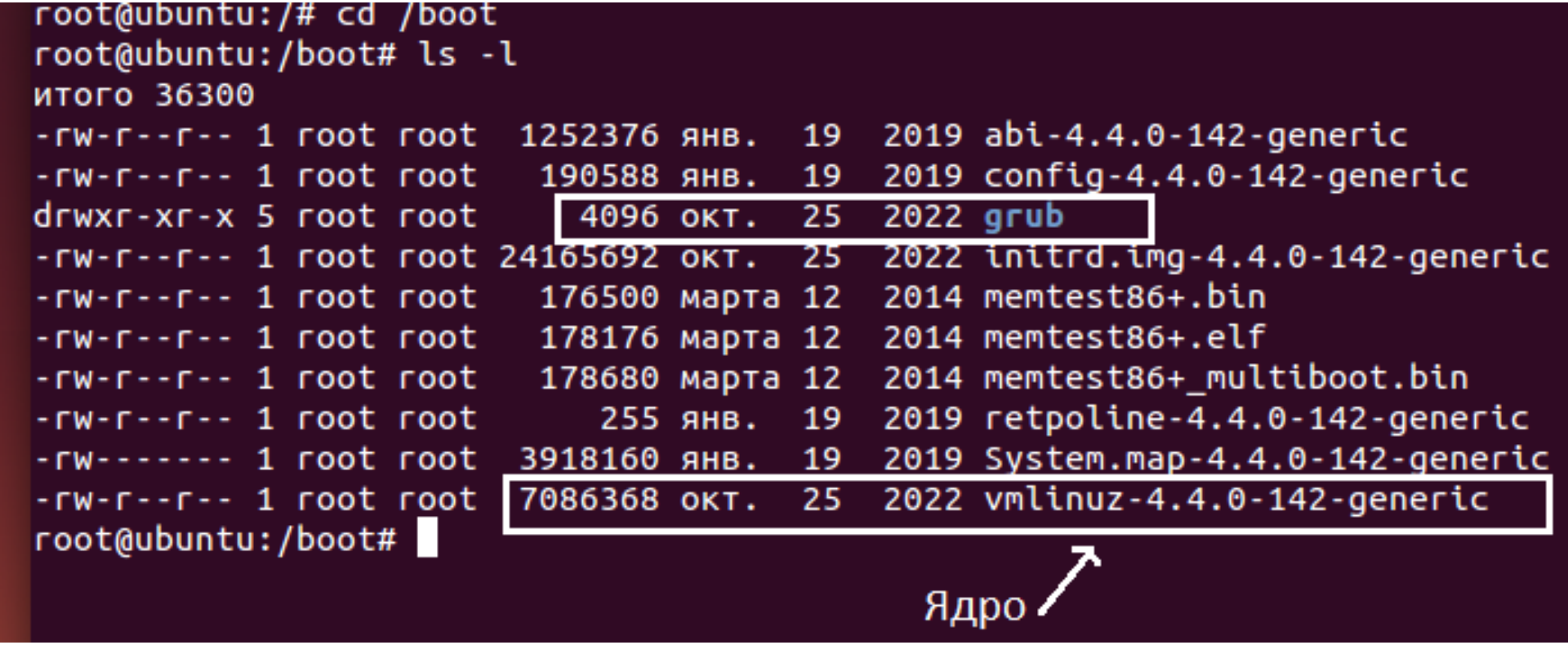
  *find /lib/modules/$(uname -r) -name *.ko*

  просмотр всех модулей текущей версии ядра

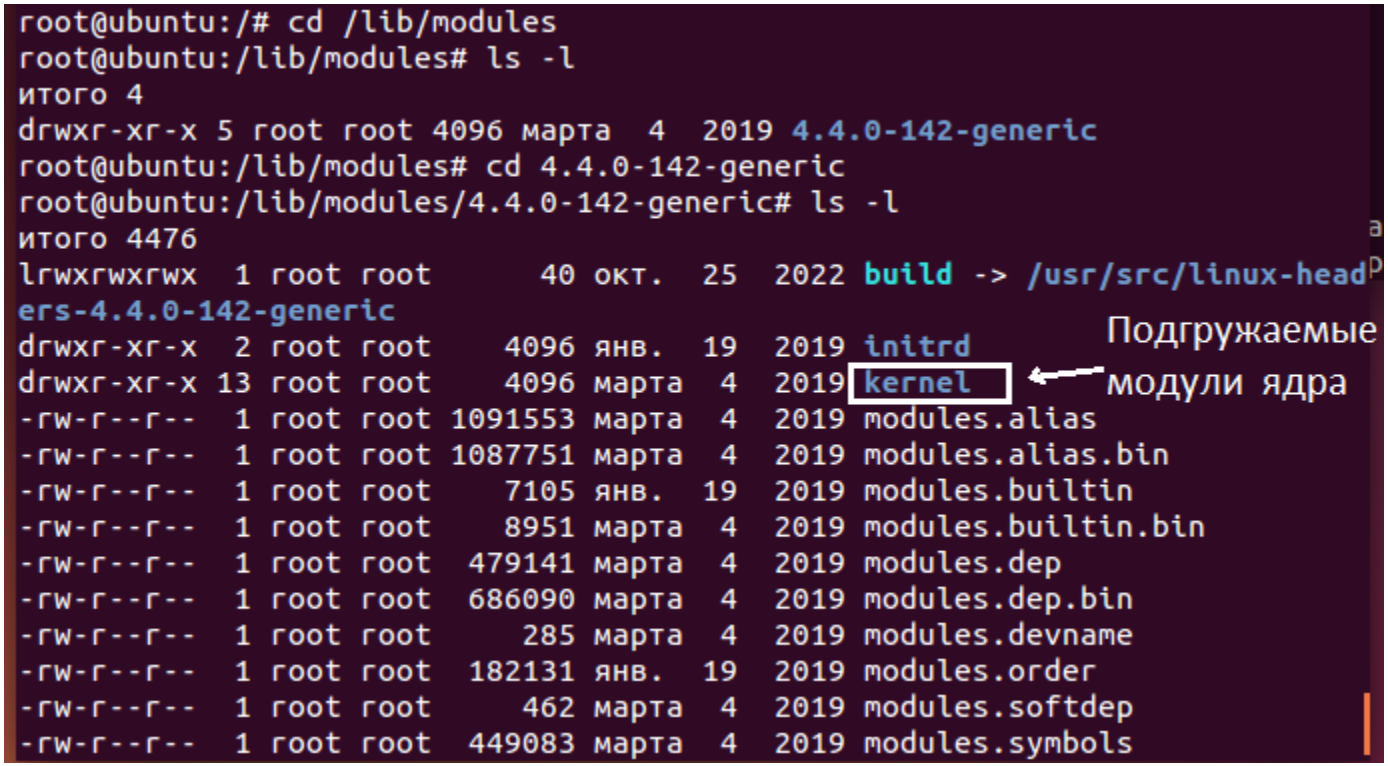
  - Команда просмотра загруженных модулей – lsmod
  - Утилита загрузки модуля – modprobe
  - Утилита выгрузки модуля – rmmod

Монолитная часть

- Ядро и загрузчик



Подгружаемые модули ядра



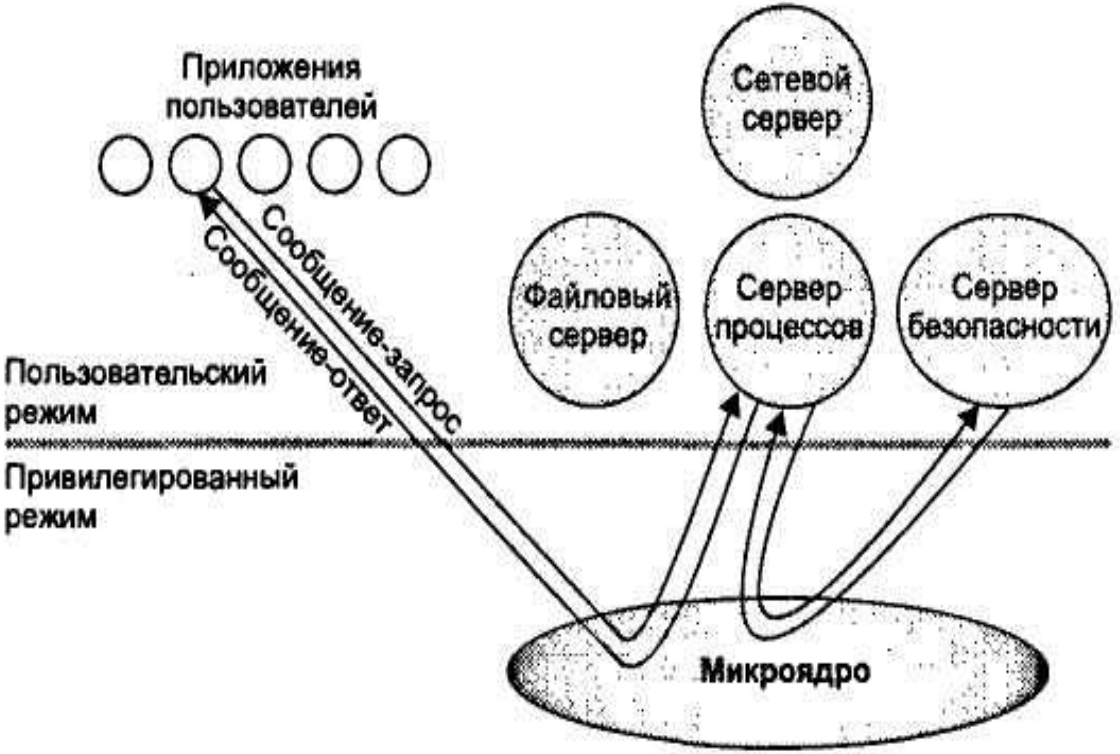
Подгружаемые модули ядра

```
root@ubuntu:/lib/modules/4.4.0-142-generic# cd kernel
root@ubuntu:/lib/modules/4.4.0-142-generic/kernel# ls -l
итого 44
drwxr-xr-x  3 root root 4096 марта  4  2019 arch
drwxr-xr-x  4 root root 4096 марта  4  2019 crypto
drwxr-xr-x 91 root root 4096 марта  4  2019 drivers
drwxr-xr-x 55 root root 4096 марта  4  2019 fs
drwxr-xr-x  4 root root 4096 марта  4  2019 kernel
drwxr-xr-x  7 root root 4096 марта  4  2019 lib
drwxr-xr-x  2 root root 4096 марта  4  2019 mm
drwxr-xr-x 52 root root 4096 марта  4  2019 net
drwxr-xr-x 13 root root 4096 марта  4  2019 sound
drwxr-xr-x 10 root root 4096 марта  4  2019 ubuntu
drwxr-xr-x  3 root root 4096 марта  4  2019 virt
```

Архитектура типа клиент-сервер на основе микроядра

- Идея архитектуры : все компоненты операционной системы разделяются на:
  - программы серверы – поставщики услуг (выполняющие определенные действия по запросам других программ)
  - программы клиенты – потребители услуг (программы клиенты, обращающиеся к серверам для выполнения определенных действий).
  - Основная часть модулей работает в режиме пользователя.
- Клиенты и серверы не общаются напрямую, а через микроядро.

Архитектура типа клиент-сервер на основе микроядра



Типичный состав микроядра

- Модули, выполняющие базовые функции ядра:
  - управление процессами (только код для переключения процессора с процесса на процесс);
  - обработка прерываний (перехват аппаратных прерываний);
  - управление виртуальной памятью;
  - передачи сообщений

Архитектура типа клиент-сервер на основе микроядра

- **Достоинства – надежность**
  - В микроядерной ОС можно, не прерывая её работы, загружать и выгружать поврежденные модули ОС, находящиеся в пространстве пользователя

- **Недостаток** – снижение производительности за счет частого переключения из режима ядра в режим пользователя.
- В монолитных 2 раза, в микроядерных 4

- Примеры:
- ОСРВ QNX (*разные версии этой ОС имеют различные объемы ядер — от 8 до 46 Кбайт*),
- FreeRTOS
- GNU Hurd,
- некоторые варианты Cisco IOS

Надежность микроядра

Монолитное ядро

Микроядро
(QNX Neutrino)

Гибридные ядра

- **Гибридное ядро** не является официальным термином, однако служит для обозначения ядер, сочетающих элементы микроядерной и монолитной архитектуры .
- В таких ОС основная часть модулей работает в режиме ядра, однако некоторые модули реализованы в виде отдельных программ, исполняемых в своих собственных адресных пространствах в режиме пользователя имогут взаимодействовать друг с другом путем передачи сообщений.
- Гибридное ядро ОС кроме функций посредника (микроядра)выполняет следующие дополнительные функции как в монолитных ОС
  - *низкоуровневое управление памятью;*
  - *взаимодействие между процессами(программами),;*
  - *управление вводом-выводом ;*
  - *управление прерываеиями.*

Гибридная архитектура Windows