# Операционные системы семейства Unix

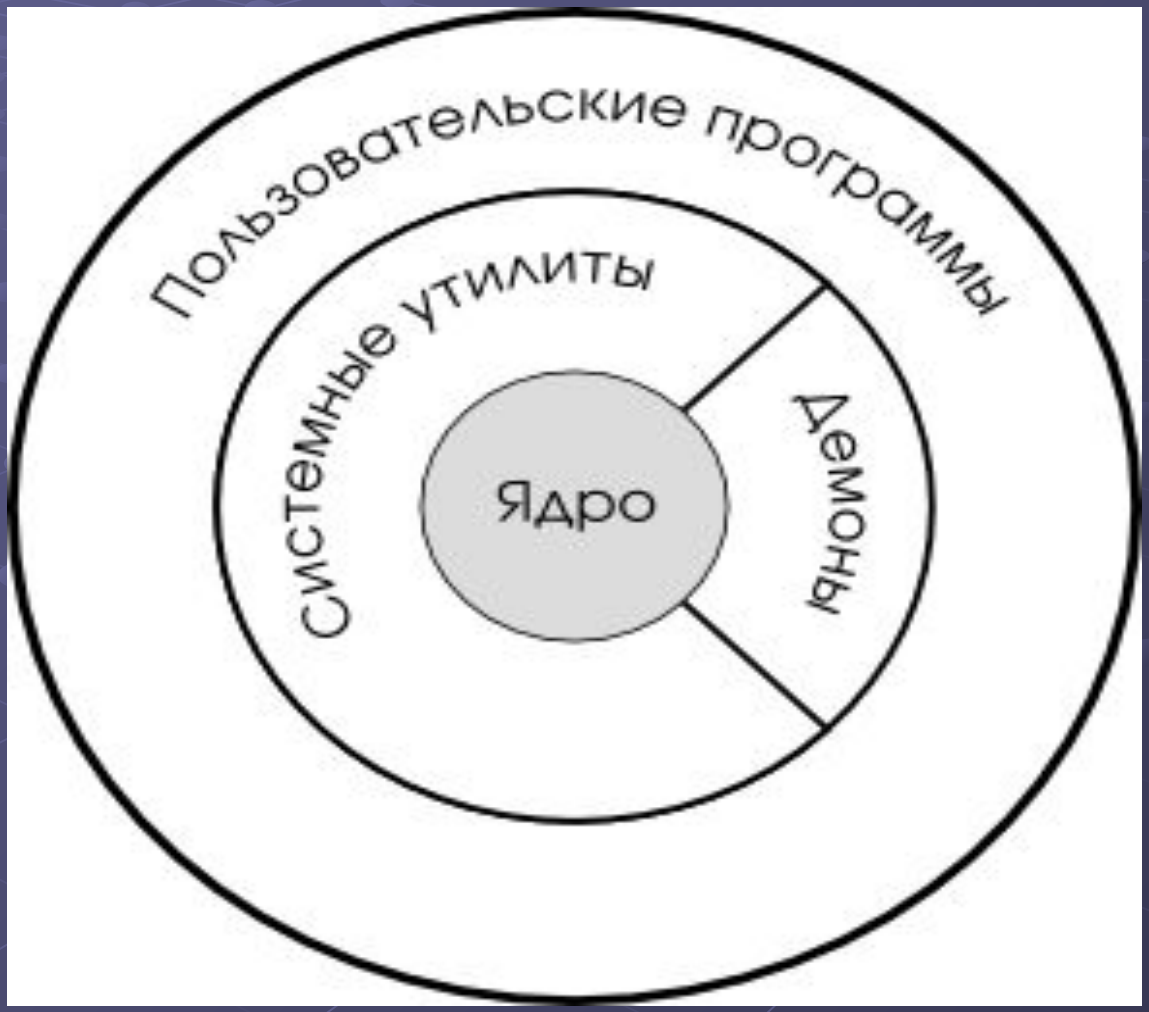
- UNIX— группа переносимых, многозадачных и многопользовательских операционных систем.


UNIX

Первая система UNIX была разработана в 1969 г. в подразделении Bell Labs компании AT&T.

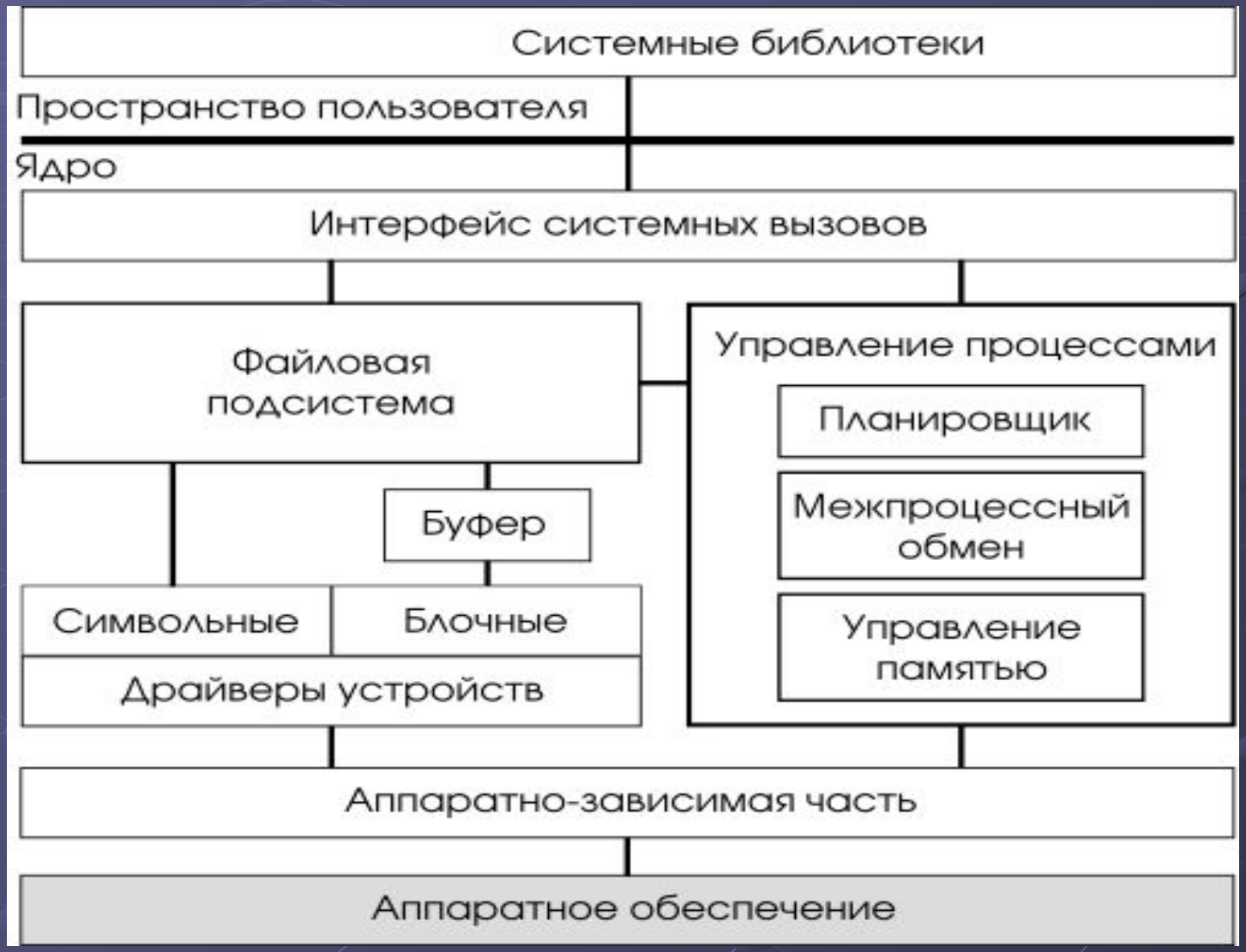
Кен Томпсон и Денис Ритчи — создатели UNIX

# Архитектура операционной системы UNIX



- Самый общий взляд на архитектуру UNIX позволяет увидеть *двухуровневую модель системы*, состоящую из *пользовательской* и *системной части (ядра)* Ядро непосредственно взаимодействует с аппаратной частью компьютера, изолируя прикладные программы (процессы в пользовательской части операционной системы) от особенностей ее архитектуры. Ядро имеет набор услуг, предоставляемых прикладным программам посредством системных вызовов. Таким образом, в системе можно выделить два уровня привилегий: *уровень системы* (привиегии специального пользователя root) и уровень пользователя (привилегии всех остальных пользователей).

# Ядро операционной системы UNIX



# Ядро UNIX

- Операционная система UNIX обладает классическим монолитным ядром, котором можно выделить следующие основные части:

- Несмотря на многообразие версий UNIX, основой всего семейства являются принципиально одинаковая архитектура и ряд стандартных интерфейсов (в UNIX стандартизовано почти всё – от расположения системных папок и файлов, до интерфейса системных вызовов и списка драйверов базовых устройств). Опытный администратор без особого труда сможет обслуживать другую версию, тогда как для пользователей переход на другую систему и вовсе может оказаться незаметным. Для системных же программистов такого рода стандарты позволяют полностью сосредоточиться на программировании, не тратя время на изучение архитектуры и особенностей конкретной реализации системы.

- В системе UNIX может одновременно выполняться множество процессов (задач), причем их число логически не ограничивается, и множество частей одной программы может одновременно находиться в системе. Благодаря специальному механизму управления памятью, каждый процесс развивается в своем защищенном адресном пространстве, что гарантирует безопасность и независимость от других процессов. Различные системные операции позволяют процессам порождать новые процессы, завершают процессы, синхронизируют выполнение этапов процесса и управляют реакцией на наступление различных событий.

- Два кита UNIX: файлы и процессы

- Существует два основных объекта операционной системы UNIX, с которыми приходиться работать пользователю – *файлы* и *процессы*. Эти объекты сильно связаны друг с другом, и в целом организация работы с ними как раз и определяет архитектуру операционной системы.
- Все данные пользователя храняться в *файлах*; доступ к периферийным устройствам осуществляется посредством чтения и записи специальных файлов; во время выполнения программы, операционная система считывает исполняемый код из файла в память и передает ему управление.
- С другой стороны, вся функциональность операционная определяется выполнением соответствующих *процессов*. В частности, обращение к файлам на диске невозможно, если файловая подсистема операционной системы (совокупность процессов, осуществляющих доступ к файлам) не имеет необходимого для этого кода в памяти.

- **Контекст процесса**

- Каждому процессу соответствует *контекст*, в котором он выполняется. Этот контекст включает содержимое пользовательского адресного пространства – пользовательский контекст (т.е. содержимое сегментов программного кода, данных, стека, разделяемых сегментов и сегментов файлов, отображаемых в виртуальную память), содержимое аппаратных регистров – регистровый контекст (регистр счетчика команд, регистр состояния процессора, регистр указателя стека и регистры общего назначения), а также структуры данных ядра (контекст системного уровня), связанные с этим процессом. Контекст процесса системного уровня в ОС UNIX состоит из «статической» и «динамических» частей. Для каждого процесса имеется одна статическая часть контекста системного уровня и переменное число динамических частей.
- Статическая часть контекста процесса системного уровня включает следующее:

- *Идентификатор процесса (PID)*
  - Уникальный номер, идентифицирующий процесс. По сути, это номер строки в таблице процессов – специальной внутренней структуре ядра операционной системы, хранящей информацию о процессах.
  - В любой момент времени номера запущенных в ситеме процессов отличаются, однако после завершения процесса, его номер может быть в дальнейшем использован для идентификации вновь запущенного процесса.

- *Идентификатор родительского процесса (PPID)*
  - В операционнной системе UNIX процессы выстраиваются в *иерархию* – новый процесс может быть создан в рамках текущего, который выступает для него родительским.
  - Таким образом, можно построить дерево из процессов, в вершине которого находится *процесс init*, запускающийся при старте системы и являющийся прародителем для всех системных процессов. Подробнее об этом процессе сказано в разделе

- *Состояние процесса*
  - Каждый процесс может находиться в одном из возможных состояний: инициализация, исполнение, приостановка, ожидание ввода-вывода, завершение и т.п

# Состояния процесса в UNIX

- Большинство этих состояний совпадает с классическим набором состояний процессов в многозадачных операционных системах. Для операционной системы UNIX характерно особое состояние процесса – *зомби*. Это состояние имеет завершившийся процесс, родительский процесс которого еще не закончил работу, и служит для корректного завершния группы процессов, освобождения ресурсов и т.п..

- Особенности UNIX, отличающие данное семейство от других ОС:
- Файловая система древовидная, чувствительная к регистру символов в именах, очень слабые ограничения на длину имён.
- Нет поддержки структурированных файлов ядром ОС, на уровне системных вызовов файл есть поток байт.
- Командная строка находится в адресном пространстве запускаемого процесса, а не извлекается системным вызовом из процесса интерпретатора команд (как это происходит, например, в RSX-11).
- Понятие «переменных окружения».
- Запуск процессов вызовом fork, то есть возможность клонирования текущего процесса со всем состоянием.

- Ввод/вывод только через дескрипторы файлов.
- Традиционно крайне слабая поддержка асинхронного ввода/вывода, по сравнению с VMS и Windows NT
- . Широкое использование текстовых файлов для хранения настроек, в отличие от двоичной базы данных настроек, как, например, в Windows.
- Широкое использование утилит обработки текста для выполнения повседневных задач под управлением скриптов.
- «Раскрутка» ОС после загрузки ядра путём исполнения скриптов стандартным интерпретатором команд.
- Широкое использование конвейеров (pipe).
- Все процессы, кроме init, равны между собой, не бывает «специальных процессов».

- Большое количество разных вариантов системы UNIX привело к необходимости стандартизовать её средства, чтобы упростить переносимость приложений и избавить пользователя от необходимости изучать особенности каждой разновидности UNIX.
- С этой целью ещё в 1980 была создана пользовательская группа /usr/group. Самые первые стандарты были разработаны в 1984—1985 гг.
- Одним из самых первых стандартов стала спецификация System V Interface Definition (SVID), выпущенная UNIX System Laboratories (USL) одновременно с UNIX System V Release 4. Этот документ, однако, не стал официальным.

- Наряду с версиями UNIX System V существовало направление UNIX BSD. Для того, чтобы обеспечить совместимость System V и BSD, были созданы рабочие группы POSIX (Portable Operating System Interface). Существует много стандартов POSIX, однако наиболее известным является стандарт POSIX 1003.1-1988, определяющий программный интерфейс приложений (API, Application Programming Interface). Он используется не только в UNIX, но и в других операционных системах. В 1990 он был принят институтом IEEE как IEEE 1003.1-1990, а позднее — ISO/IEC 9945.
- В настоящее время наиболее важными являются следующие стандарты:
- POSIX 1003.2-1992, определяющий поведение утилит, в том числе командного интерпретатора.
- POSIX 1003.1b-1993, дополняющий POSIX 1003.1-1988. Определяет поддержку систем реального времени.
- POSIX 1003.1c-1995, дополняющий POSIX 1003.1-1988. Определяет нити (threads), известные также как pthreads.
- Все стандарты POSIX объединены в документе IEEE 1003.

- В начале 1990-х годов The Open Group предложила другой, похожий на POSIX стандарт — Common API Specification, или Spec 1170. Стандарт приобрёл большую популярность, чем POSIX, поскольку был доступен бесплатно, в то время как IEEE требовало немалую плату за доступ к своему стандарту.
- В 1998 году были начаты работы по объединению данных стандартов. Благодаря этому в настоящее время данные стандарты почти идентичны. Совместный стандарт называется Single UNIX Specification Version 3 и доступен бесплатно в интернете [3].

- В целях совместимости несколько создателей UNIX-систем предложили использовать ELF-формат систем SVR4 для двоичных и объектных файлов. Единый формат полностью обеспечивает соответствие двоичных файлов в рамках одной компьютерной архитектуры.