

# **Доклад**

**на тему: Системные инициализаторы system V**

Тойчубекова Асель Нурлановна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Система инициализации</b>	<b>6</b>
<b>3</b>	<b>Система инициализации System V</b>	<b>10</b>
3.1	Классическая программа <code>init</code> в сочетании со скриптами <code>rc.d</code> в стиле System V . . . . .	12
3.2	Преимущества и недостатки System V . . . . .	19
<b>4</b>	<b>Заключение</b>	<b>21</b>
	<b>Список литературы</b>	<b>22</b>

## Список иллюстраций

2.1	Процесс загрузки ОС . . . . .	7
3.1	Файл /etc/inittab . . . . .	12
3.2	Файл /etc/inittab . . . . .	13

## **Список таблиц**

# 1 Цель работы

Целью данного доклада является понять роль и функции систем инициализации в операционных системах на базе ядра. Также рассмотреть традиционную систему инициализации System V, включая ее структуру, режимы загрузки и управление службами. Определить преимущества и недостатки System V как системы инициализации.

## 2 Система инициализации

**Система инициализации** является критически важным компонентом, который определяет, как операционная система (ОС) запускается и загружается. Эта система служит первым уровнем, который вступает в действие сразу после запуска ядра Linux, выполняя роль связующего звена между ядром ОС и высокоуровневыми приложениями.

Система инициализации выполняет несколько ключевых задач:

- **Запуск процессов:** она иницирует и управляет различными процессами, необходимыми для работы системы и пользовательских приложений: управление сетевыми соединениями, системными логами, планировщиком задач и другими критически важными сервисами.
- **Управление зависимостями:** система инициализации управляет порядком и условиями запуска служб, обеспечивая правильное разрешение всех необходимых зависимостей и запуск сервисов в нужном порядке.
- **Контроль за процессами:** после начальной загрузки система инициализации продолжает контролировать работающие процессы, перезапуская службы в случае их сбоя или остановки для обеспечения стабильности и надёжности системы.

В операционной системе Linux и других системах семейства Unix после завершения загрузки ядра начинается инициализация Linux системы, сервисов и других компонентов. За это отвечает процесс инициализации, он запускается

ядром сразу после завершения загрузки, имеет PID 1, и будет выполняться пока будет работать система.

Процесс инициализации запускает все другие процессы, которые должны быть запущены, это родительский процесс для всего, что выполняется в системе. Другие процессы могут тоже создавать дочерние процессы, но если родительский процесс завершается, для его дочерних процессов родительским становится процесс инициализации. Процесс загрузки ОС на базе ядра Linux представлен на (рис. 2.1).

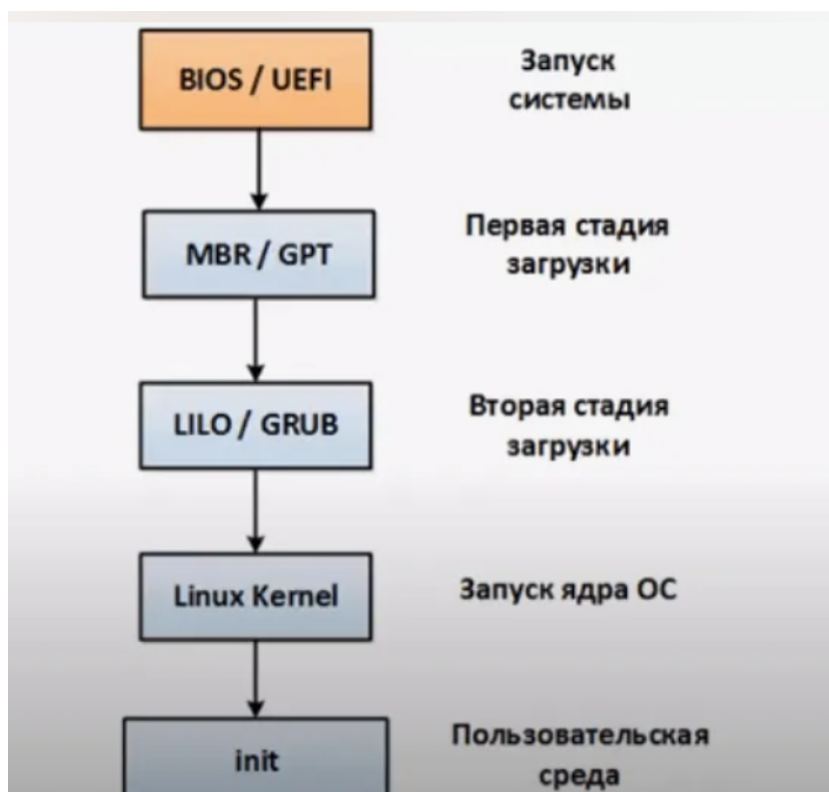


Рис. 2.1: Процесс загрузки ОС

Системы инициализации запускают демоны силами сценариев, причем каждый из сценариев осуществляет запуск одного демона, а каждый следующий сценарий ожидает завершения исполнения предыдущего сценария.

**Демон (daemon)** является процессом, который выполняется в фоновом режиме без связи с графическим интерфейсом или терминалом. Обычно демоны

запускаются при загрузке системы и находятся в рабочем состоянии вплоть до момента завершения работы системы. В современной технической документации демоны чаще всего называются службами (services).

Системный загрузчик передает контроль над системой ядру ОС. После непроизвольного периода времени ядро ОС запускает демон системы инициализации. Этот демон системы инициализации (/sbin/init) является первым демоном, запущенным в рамках системы, поэтому соответствующий процесс получает идентификатор 1 (PID 1). Демон системы инициализации никогда не завершает свою работу.

После того, как исполняется бинарный файл /sbin/init, в первую очередь осуществляется чтение конфигурационного файла /etc/inittab. В данном файле демон будет искать значение переменной initdefault (равное 3 в примере ниже).

С помощью значения переменной initdefault указывается стандартный уровень исполнения (default runlevel). В некоторых дистрибутивах Linux в файле /etc/inittab приводится краткое описание уровней исполнения подобное приведенному ниже переведенному описанию из соответствующего файла дистрибутива Red Hat Enterprise Linux 4.

Уровень исполнения 0 соответствует отключению системы. Уровень исполнения 1 используется для устранения неполадок, так как осуществить вход в систему может исключительно пользователь root, причем для входа в систему может использоваться исключительно консоль. Уровень исполнения 3 типичен для серверов, а уровень исполнения 5 - для настольных компьютеров (на которых вход в систему осуществляется в графическом режиме). За исключением уровней исполнения 0, 1 и 6, различные уровни исполнения могут отличаться в зависимости от дистрибутива. К примеру, в дистрибутиве Debian и производных дистрибутивах Linux на уровнях исполнения 2 и 5 имеется возможность входа в систему с использованием сетевого соединения и графического интерфейса. Исходя из этого, следует всегда сверяться с корректным описанием уровней исполнения вашей системы.



## Типы систем инициализации

Существует несколько различных типов систем инициализации, в том числе:

- **System V init:** Традиционная система инициализации, используемая во многих старых дистрибутивах Unix и Linux.
- **Upstart:** Система инициализации, разработанная для Ubuntu и других дистрибутивов Linux, которая предоставляет более современный и гибкий интерфейс.
- **Systemd:** Современная и широко используемая система инициализации, которая предлагает расширенные возможности, такие как параллельная загрузка и управление зависимостями.

## 3 Система инициализации System V

**System V** или **SysV** - это довольно старая, но до сих пор ещё популярная система инициализации Linux и Unix подобных операционных систем. Она была основой для создания многих других систем инициализации, а также первой коммерческой системой инициализации разработанной для Unix в AT&T. Она была разработана еще в 1983 году.

Почти все дистрибутивы Linux изначально использовали SysV. Исключением была только Gentoo, в которой использовалась собственная система инициализации и Slackware, с инициализацией в стиле BSD.

SysVinit работает последовательно, выполняя shell -скрипты инициализации, расположенные, как правило, в директории /etc/init.d , в соответствии с порядковыми номерами. Данная модель предлагает простую и понятную структуру, где каждый скрипт отвечает за определённый сервис или задачу.

К основным возможностям SysV относятся: 1. Написание файлов запуска служб на bash;

2. Последовательный запуск служб;
3. простота управления сервисами через символические ссылки и скрипты;
4. минимальные зависимости от стороннего программного обеспечения, что делает её правильным выбором систем с ограниченными ресурсами.
5. Сортировка порядка запуска с помощью номеров в именах файлов;
6. Команды для запуска, остановки и проверки состояния служб.

Для того, чтобы понять, как происходит инициализация необходимо понять, что такое режимы загрузки (они же runlevel), разобраться как между ними переключаются, рассмотреть работу со службами.

**Уровни выполнения в SysV** - это механизм для определения различных состояний операционной системы и служб, которые должны быть запущены на каждом уровне. Они позволяют администраторам системы гибко управлять запуском и остановкой служб в зависимости от потребностей системы.

**Уровень 1** (однопользовательский режим): Используется для выполнения задач администрирования системы, требующих эксклюзивного доступа к системе, таких как восстановление файловых систем или сброс пароля root.

**Уровень 2** (многопользовательский режим без сетевого доступа): Полезен для выполнения задач администрирования системы, таких как обновление программного обеспечения или устранение неполадок с оборудованием, без необходимости запуска сетевых служб.

**Уровень 3** (многопользовательский режим с сетевым доступом): Стандартный уровень выполнения, который запускает все необходимые службы для многопользовательской работы, включая сетевые службы.

**Уровень 4** (не используется): не имеет стандартного толкования и практически не используется.

**Уровень 5** (графический режим): Используется для запуска графического интерфейса пользователя (GUI).

**Уровень 6** (перезагрузка системы) — при включении этого режима останавливаются все запущенные программы и производится перезагрузка.

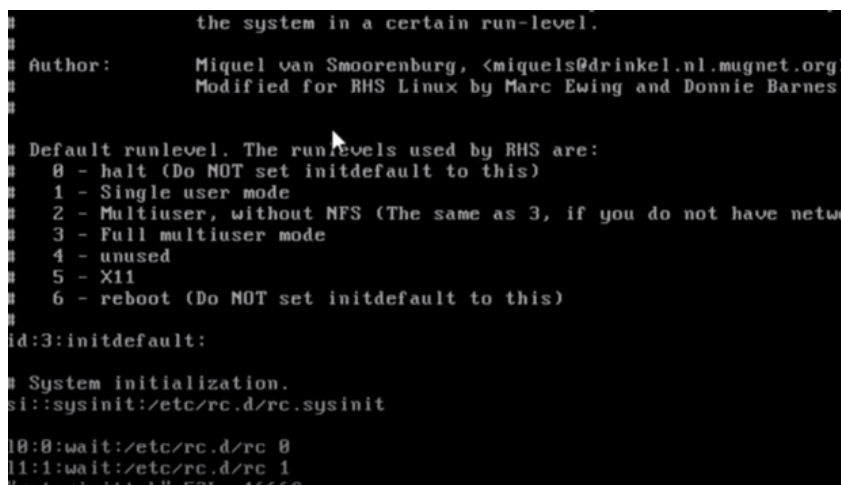
Но существуют операционные системы, где 10 уровней по умолчанию. Конечно, речь идет о самых распространенных ядрах и сборках \*nix образных операционных системах.

## 3.1 Классическая программа `init` в сочетании со скриптами `rc.d` в стиле **System V**

Классический **System V** `init` читает файл `/etc/inittab` и выполняет ряд предписаний, которые прописаны в этом файле. `Inittab` этот текстовый файл каждая строка которого, это, по сути дела, одна команда или какое-то правило поведения.

`Inittab` выглядит так: `id:3: initdefault:`  
`si::sysinit:/etc/rc.d/rc.sysinit`  
`l3:3: wait:/etc/rc.d/rc 3`  
`1:2345: respawn:/sbin/mingetty tty1`  
`ca: ctrlaltdel:/sbin/shutdown -t3 -r now`

Открытый файл `/etc/inittab` с помощью редактора `vi` представлена на (рис. 3.1 и рис. 3.2).

A screenshot of a terminal window showing the contents of the `/etc/inittab` file. The text is as follows:

```
# the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
# Modified for RBS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RBS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have netw
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
# ...
```

Рис. 3.1: Файл `/etc/inittab`

```

# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Trap CTRL-ALT-DELETE
ca:_ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes

```

Рис. 3.2: Файл /etc/inittab

Вначале строки стоит метка. В чем большой смысл этой метки я не очень понимаю. Можно считать, что это простой текст и все. Вторым пунктом стоит либо так называемый уровень загрузки, либо пустое значение. Дальше идет некое действие. Действия бывают следующие: wait, respawn, sysinit, ctrlaltdel. Есть и другие действия, но это самые используемые. Наконец, в конце строки написана некая команда с именем исполняемого файла и аргументов, которые этой команде надо передать.

**Действие sysinit** выполняется однократно при старте системы.

**Действие ctrlaltdel** это на самом деле не совсем действие – это обработчик сочетания клавиш control alt del. Само нажатие перехватывается ядром системы, и информация об этом пересылается в процесс init, который должен выполнить определенную команду. Например, может быть выполнена команда shutdown, которая выполнит выключение компьютера. В принципе сюда можно прописать любую другую программу, например, echo, которая после нажатия control alt del будет выдавать на все терминалы системы какое-нибудь сообщение.

**Действие wait** означает, что необходимо запустить команду, дождаться пока она закончится и только после этого продолжить обработку следующих строк.

**Действие respawn** означает, что надо запустить программу и не дожидаясь ее завершения, перейти в дальнейшим действиям. Если эта программа в после-

дующем завершится, то необходимо ее рестартовать.

Итак, есть однократное выполнение с ожиданием результатов и многократное выполнение в асинхронном режиме – запустились, дождались пока закончить, запустили слова.

Рассмотрим строки реального файла.

### **l3:3: wait:/etc/rc.d/rc 3**

Запускается какая-то программа, которая должна выполнить все необходимые действия, которые ожидаются на третьем уровне. Наверно, на третьем уровне нужно настроить сетевые интерфейсы, запустить драйвер терминалов, стартовать какие-то службы. Только после того, как всё этого завершится мы сможем работать в системе. Поскольку надо дождаться завершения запуска, мы выбираем действие wait.

Программа запуска называется **rc** и запускается с номером уровня в качестве параметра. Сама программа **init** достаточно простая. Она умеет построчно читать свой файл с простым синтаксисом и стартовать новые процессы, запуская какие-то вспомогательные программы. Вся логика уровней загрузки спрятана в скрипте **rc**. Запустив **rc** с параметром 3 мы перейдем на третий уровень, с параметром 5 - на пятый.

**Программа rc** тоже очень простая. Это скрипт который выполняет все файлы в каталогах, соответствующих уровню загрузки, например, **/etc/rc3.d/**. В этих каталогах находятся исполняемые файлы, которые принимают один параметр - либо **start**, либо **stop**. Если файл запущен с параметром **start**, то он стартует службу, если с параметром **stop**, то останавливает её. Например, **network start** будет настраивать сетевые интерфейсы, а **network stop** будет переводить интерфейсы в выключенное состояние. Кроме сетевых интерфейсов есть скрипты подключения/отключения сетевых файловых систем, запуска/остановки сервисов и т.д.

Имена файлов в каталогах построены по определенным правилам. Они начинаются либо с буквы **K** либо с буквы **S**, за которыми идет число и имя службы. Скрипт **rc** просматривает содержимое каталога **rc3** и выбирает оттуда все фай-

лы, которые начинаются с буквы **K (kill)**. Файлы упорядочиваются в порядке возрастания номера и выполняются с параметром `stop`. Потом те же действия выполняются с файлами на букву **S (start)**, которые запускаются с параметром `start`. Вот в общем и вся процедура перехода на определенный уровень.

Можно предположить, что в каталоге `/etc/rc0.d/` лежат только файлы, начинающиеся на букву K, поскольку при выключении надо все остановить, а в каталоге `/etc/rc1.d/` будет один файл на букву S для запуска консоли администратора.

Для простоты программирования есть отдельный каталог `/etc/init.d/`, в котором лежат те же самые файлы только без буквы цифр в начале имени. На самом деле, файлы в каталогах уровней — это просто символические ссылки на основные файлы. Так `/etc/rc3.d/S10apache` это ссылка на файл `/etc/init.d/apache`. Буквы и цифры в названии ссылок нужны для того, чтобы скрипт `rc` вызвал их в нужном порядке и с нужными аргументами.

В системах, которые построены по такому принципу, чтобы стартовать или остановить какую-либо службу в каталоге `/etc/init.d/` надо найти файл который, который ей соответствует, и запустить его с параметром `start` или `stop`. Чем не нравится запускать службы именно таким способом - явно вызывая скрипты. Дело в том, что в командной строке `linux` замечательно работает автодополнение. С его помощью очень быстро можно ввести путь до файла запуска.

Чтобы спрятать от пользователя конкретную реализацию поверх системы скриптов и символических ссылок написаны две вспомогательные программы.

**Программа `chkconfig`** позволяет манипулировать символическими ссылками на соответствующие скрипты. Чтобы посмотреть, что стартует, а что останавливается на каждом из уровней можно воспользоваться командой `ls` и выдать список скриптов в соответствующем каталоге, но проще воспользоваться командой `chkconfig --list`. Программа `chkconfig` пробегает по всем каталогам `rc` и выдает список того что стартует, а что останавливается на каждом уровне. Если мы хотим, чтобы при старте системы у нас что-то автоматически стартовала определенная служба мы выполняем `chkconfig on` и скрипт создает ссылку для

запуска в нужном каталоге и с правильным именем. Запуск `chkconfig off` приводит к удалению ссылки для запуска и созданию ссылки для остановки. Таким образом программа `chkconfig` позволяет управлять списком служб, которые стартуют в момент старта системы.

Ещё одна **программа** - **service** используется для ручного запуска и остановки служб. `Service` это обертка, которая позволяет не обращаться напрямую к скрипту, а указать имя службы и сказать хотим мы ее стартовать или остановить. В `bash`, который я использую, нет автодополнения для команды `service`, поэтому мне проще набрать путь к скриптам.

В стартовых скриптах аргументы `start` и `stop` должны обрабатываться обязательно. Кроме того, можно придумать какие-то свои аргументы, которые будут делать что-то полезное.

В большинстве скриптов реализована **опция status**, которая показывает запущена служба или нет. Когда мы выполняем `start`, то скрипт после успешного запуска службы получает ее идентификатор PID и записывать его в определенный файл. По команде `stop` файл удаляется. Обычно такие файлы создаются в каталоге `/var/run/`. Команда `status` проверяет есть ли такой файл. Его нет, то сообщает, что служба не запущена. Если файл есть, то она извлекает из него идентификатор процесса и проверяет текущий список процессов. Если этот идентификатор присутствует все запущено, если программа по каким-то причинам поломалась, то статус выдаёт, что была сделана попытка запустить эту службу - файл существует, но сама служба не запущена.

**Опция restart** последовательно выполняет внутри скрипта две команды – сначала `stop`, а потом `start`. Это совершенно необязательная команда - просто удобная. Наконец, есть службы, которые позволяют на ходу перечитать какие-то конфигурационные файлы. Для них добавляют команду `reload`, задачей которой является отправка службе сигнала о том, что конфигурация изменилась. Отдельный случай, команды `save` и `load` для сохранения конфигурации брандмауэра.



Ещё несколько строк в `inittab`, это запуск терм

**1:2345: respawn:/sbin/mingetty tty1**

Для того, чтобы обеспечить диалоговую доступ к системе, в `inittab` может присутствовать некоторое количество строчек такого рода. 2345 это уровни, на которых надо запускать команду, `respawn` означает, что программу надо перезапускать в случае завершения. Программа `getty` – это программа управления терминалом. Традиционно терминал в UNIX называется телетайпом, поскольку первыми терминалами были электрические пишущие машинка. Соответственно, `tty` это сокращение от телетайпа. `Mingetty` – программа, которая умеет работать с виртуальными терминалами на персональном компьютере. Она умеет настраивать драйвер терминала, а в качестве параметров получает имя устройства терминала, который надо настроить. В каталоге `/dev/` есть файл устройства `tty1`, который соответствует первому виртуальному терминалу. Если бы у нас был модем и мы хотели бы инициализировать его момент загрузки, то могли бы вызвать `getty` с параметром `ttyS0`, который соответствует порту COM1. При инициализации модема можно было бы задать дополнительные параметры: скорость соединения 19200 бод, 7 или 8 бит в байте, четность, количество стоп-битов.

**S0:2345: respawn:/sbin/getty ttyS0 19200 8 n 1**

Текстовые пользовательские сеансы устроены на таких цепочках: сначала `init` делает свою копию и запускает в ней программу `mingetty`. `Mingetty` инициализирует терминал и клавиатуру, а потом запускает в том же процессе программу `login`. `Login` выводит на экран приглашения на ввод имени и пароля и, если все прошло успешно то назначает себе привилегии пользователя и в том же процессе, затирая самого себя, запускает интерпретатор пользователя, например, `bash`. Когда пользователь набирает команду `exit`, то интерпретатор завершает жизненный путь этого процесса. Когда процесс завершается, `init` получает об этом сигнал. `Init` смотрит, что полагается делать, видит действие `respawn`, снова запускает программу `mingetty`, которая заново инициализирует терминал и все повторяется. Таким образом каждый сеанс находится внутри одного процесса.

Как только мы вышли из сеанса наш процесс закончился и тотчас же запустилась программа, которая почистит за нами терминал и восстановит все настройки по умолчанию.

В файле `inittab` есть ещё одно специальное ключевое слово `initdefault` - уровень по умолчанию. Если через ядро `init` получил параметр `single`, то мы загрузимся на уровень 1. Если через загрузчик ничего не передали, то используется значение по умолчанию. Если после установки графической оболочки оказалось, что наш компьютер слабоват для графики, то можно установить уровень по умолчанию на 3, и после следующей перезагрузки мы попадаем на третий уровень - то есть в текстовый режим. Установили систему без графического режима, потом доустановили все пакеты для `x window`, поменяли уровень по умолчанию на 5 и после следующей перезагрузки попали сразу в графический режим.

#### **Основные команды в командной строке:**

- `Init` управление инициализацией с помощью нее можно перемещаться между `runlevel`.
- `Telinit` управление процессом `init`, в старых дистрибутива использовалась именно эта команда.
- `Wall` вывод сообщения пользователям системы
- `Halt` - выключение компьютера
- `Reboot` перезагрузка компьютер
- `Shutdown` - запланированное выключение

Для того, чтобы перемещаться по уровням загрузки, нам необходимо понять на каком уровне мы находимся сейчас. Набираем `runlevel`. Соответственно, если мы хотим переключиться `telinit 1` отработывают скрипты мы попадаем в однопользовательский режим 1.

Для того, чтобы послать сообщение все пользователям на данной машине необходимо набрать с соблюдением регистра wall “Abrakadabra”. У всех пользователей появится данное сообщение на экране.

Для выключения сейчас компьютера можно использовать shutdown h now.

## 3.2 Преимущества и недостатки System V

**Преимущества System V init по сравнению с другими системами инициализации:**

1. Простота и понятность: Сценарии инициализации написаны в виде простых текстовых файлов, что упрощает их понимание и модификацию.
2. Гибкость: Администраторы системы могут легко добавлять, удалять и изменять сценарии инициализации, чтобы настроить систему в соответствии со своими потребностями.
3. Совместимость: System V init широко используется в дистрибутивах Unix и Linux, что обеспечивает совместимость с большим количеством приложений и сервисов.
4. Низкие требования к ресурсам: System V init является относительно легкой системой инициализации, что делает ее подходящей для систем с ограниченными ресурсами.

**Недостатки System V init по сравнению с другими системами инициализации:**

1. Последовательный запуск: System V init запускает службы и процессы последовательно, что может привести к замедлению загрузки системы.
2. Сложность управления зависимостями: System V init не предоставляет встроенного механизма для управления зависимостями между службами, что может привести к ошибкам при запуске и остановке служб.

3. Ограниченные возможности параллельного запуска: System V init запускает службы и процессы последовательно, что не позволяет использовать возможности параллельного запуска современных многоядерных систем.
4. Устаревший: System V init является устаревшей системой инициализации, которая больше не так активно развивается, как другие системы инициализации.

В целом, System V init остается популярным выбором для систем, требующих простоты и гибкости. Однако в современных системах, требующих высокой производительности и расширенных возможностей управления, более подходящими альтернативами являются Upstart или Systemd.

## 4 Заключение

Система инициализации является жизненно важным компонентом в операционных системах на базе ядра, таком как Linux, которая отвечает за правильную загрузку и инициализацию системы. Она управляет запуском процессов и служб, обеспечивает упорядоченный запуск и остановку, а также контролирует их поведение во время работы системы.

System V init является традиционной системой инициализации, которая использовалась во многих старых дистрибутивах Linux. Она предоставляет простую и понятную структуру для управления службами, но имеет ограничения, такие как последовательный запуск и отсутствие управления зависимостями.

Современные системы инициализации, такие как Systemd, предлагают расширенные возможности и большую гибкость. Они поддерживают параллельный запуск, управление зависимостями и предоставляют более полный набор функций. Выбор между разными системами инициализации зависит от требований конкретной системы и предпочтений администратора. System V init остается популярной в простых и встроенных системах, в то время как Systemd широко используется в современных дистрибутивах Linux для более сложных и требовательных систем.

## Список литературы

1. <https://parallel.uran.ru/book/export/html/487?ysclid=lvqoxg5dcv321111041>
2. <https://moluch.ru/archive/503/110797/?ysclid=lvq8qoj6k7851804848>
3. <https://losst.pro/sistemy-initsializatsii-linux?ysclid=lvqcpskbl6380114430>
4. [https://pikabu.ru/story/sistema\\_initsializatsii\\_5191339?ysclid=lvq8i3pnwl986484468](https://pikabu.ru/story/sistema_initsializatsii_5191339?ysclid=lvq8i3pnwl986484468)
5. <https://wiki.merionet.ru/articles/3-varianta-inicializacii-operacionnoj-sistemy?ysclid=lvqcw2mbfm974602146>