Инициализиране на системата  
• Преглед на процеса на първоначално зареждане;  
• Init процес и /etc/inittab конфигурационен файл;  
• Конфигуриране на boot manager ( GRUB);  
• Смяна на runlevel и настройка на runlevel по подразбиране;  
• Systemd

**Systemd init системата в Linux и \*nix**

**Какво представлява**[**init**](https://en.wikipedia.org/wiki/Init)**системата в Linux и Unix?**

Стартирането при всички \*nix системи е многостепенно – системата преминава през няколко етапа преди да бъде стартирана операционната система и нужният ѝ набор от софтуер.

Когато се натисне бутона на машината и всички хардуерни компоненти работят, се стартира BIOS (Basic Input/Output System). Следва проверка на хардуерните компоненти като CPU, RAM памет, периферни устройства. След това идва ред на проверка на MBR (Master Boot Record), от него се зарежда boot loader, зарежда се самото ядро на операционната система, включва се init системата и чак тогава се стартират софтуерните приложения и процеси, които са вградени в системата или са инсталирани от потребители в последствие.

Целта на init системата е **да стартира всичко, както трябва да бъде стартирано,** след като бъде заредено ядрото.

На практика тази система стартира и управлява всички системни и потребителски процеси. Работата ѝ включва проверка и монтиране на файловите системи, стартиране на включени потребителски процеси и превключване в потребителската среда, когато инициализацията завърши.

### Systemd като нова init система

Systemd е абревиатура на **System Management Daemon**, към която е присвоена UNIX конвенция за добавяне на „d“ накрая на всеки daemon, за да бъде по-лесно разпознаваем.

Първоначално systemd е публикувана под GNU General Public License, но следващите версии са под GNU Lesser General Public License. Подобно на init, systemd е отговорна за директното или индиректно стартиране на всички процеси в операционната система.

Първият процес представлява именно init и идентификационния му номер (pid) винаги е 1. systemd може да се отнася за всички пакети, инструменти и библиотеки, свързани с daemon. Системата е проектирана, за да превъзмогне недостатъците на init. Тя представлява процес, стартиран на заден фон, който стартира други процеси паралелно, а не поетапно, както init. По този начин ускорява първоначалното зареждане на операционната система.

При конвенционалната init система започва нов процес, само ако предишният е завършил стартирането си и е зареден в паметта. Това често водеше до по-бавно стартиране на системата.

### Предимствата на systemd:

* Чисто и ефективно стартиране на процеси.
* Прост процес по инициализация на системата.
* Паралелно и едновременно стартиране на процесите.
* По-добро API.
* Опростен UNIX синтаксис.
* Възможност за премахване на опционални компоненти.
* По-ниска консумация на памет.
* Подобрена техника в изразяването на зависимости.
* Инструкциите за инициализация са описани в конфигурационен файл, а не в init shell скрипт.
* Може да използва Unix socket-и.
* Запис на събитята в логове с journald.
* Избор на система за логове (вкл. syslog).
* Логовете се съхраняват в бинарни файлове.
* Състоянието на systemd може да бъде запазено и извикано на по-късен етап.
* Следи процесите като се използва cgroup на ядрото, а не техният PID.
* По-добра имплементация с Gnome.

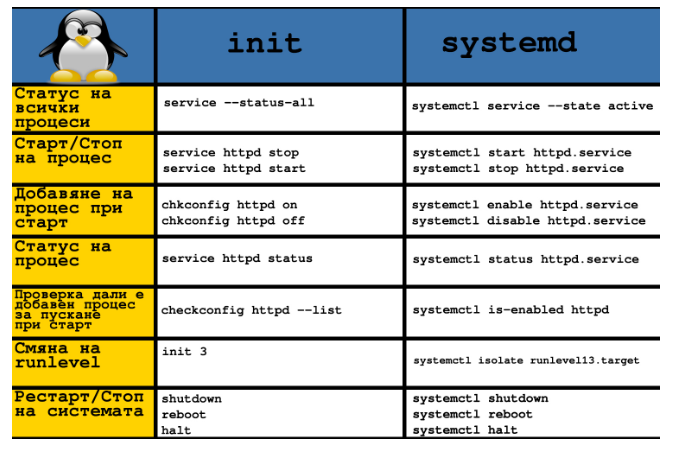
Към този момент systemd се използва във всички големи дистрибуции. Добавянето на systemd към пакетите на някои дистрибуции започна от 2011 г. От средата на 2014 г. пък е зададен като основен демон за мениджмънт на процесите.

Fedora са първата голяма дистрибуция, която реши да използва systemd, след тях и OpenSUSE,Arch, Red Hat (заедно с CentOS), а накрая и Debian, и Ubuntu. Част от Debian обществото обаче изрази силно неодобрение към systemd и това накара група хора да стартират проекта Devuan – Debian, но с традиционния sysvinit (проектът е в ALPHA).

Независимо дали ви харесва, или не, systemd е вече факт и ще остане така за дълго време. Впрочем  има много добри страни, а освен това е една стъпка по напред от стандартния init. Причината за голямото недоволство е не толкова във възможностите и структурата, а по-скоро промяната.

### Полезни команди за init и systemd

Ако имате предишен опит с init, но още не сте свикнали или тепърва ще използвате systemd, можете да опитате този “пищов” с команди, който подготвихме специално за вас:



Init процес и / etc / inittab файл e-reading.club

Както знаете, след включване на компютъра и завършване на хардуерното тестване, BIOS чете кратък буутлоудъра от първия сектор на зареждащия диск. Тази програма стартира основния системен товарач (например lilo), който от своя страна зарежда ядрото на системата в паметта, която обикновено се съхранява във файла vmlinuz-xyz-в директорията / boot. Тук xyz е номерът на версията на ядрото, а вместо знака често има индикация за някои специфични модификации на ядрото. Въпреки това, името на файла на ядрото може да е различно, за bootloader няма значение, само това име трябва да бъде посочено в конфигурационния файл на bootloader. Веднага след зареждането, ядрото монтира кореновата файлова система и стартира init процеса. Процесът init е програма, която е отговорна за продължаване на процедурата за зареждане и прехвърляне на системата от първоначалното състояние, което се случва след като ядрото е заредено в стандартното състояние на обработка на много потребители. Init също извършва много различни операции, необходими за по-нататъшната работа на системата: проверка и монтиране на файлови системи, стартиране на различни услуги (демони), стартиране на процедури за регистриране, опаковане на потребители на различни терминали и др. Точният списък на тези операции зависи от така нареченото ниво на изпълнение. Нивото на изпълнение определя списъка на действията, изпълнявани от процеса на init и състоянието на системата след зареждане, т.е. конфигурацията на изпълняваните процеси. Нивото на изпълнение се идентифицира с един символ. Има 8 основни нива на изпълнение в Linux: • 0 - спиране на системата; • 1 - еднопотребителски режим (за специални случаи на администриране); • 2 - многопотребителски режим без NFS (същия като 3, ако компютърът не работи с мрежата); • 3 - пълен мултиплеър режим; • 4 - употребата не е регулирана; • 5 - обикновено се използва за управление на системата в графичен режим; • 6 - рестартиране на системата; • S (или s) е приблизително същия като режим за един потребител, но S и s се използват главно в скриптове. Както можете да видите, нива 0, 1 и 6 са запазени за специални случаи. Що се отнася до използването на нива от 2 до 5, няма консенсус. Някои системни администратори използват различни нива, за да задават различни работни опции, например те изпълняват графичен режим на едно ниво, работят в друга мрежа и т.н. Можете да решите как да използвате различни нива за създаване на различни опции за зареждане. Но за начало най-лесният начин е да се използва методът за определяне на различните нива, които са зададени по време на инсталацията. На първо място, след стартирането, процесът init чете конфигурационния си файл / etc / inittab. Този файл се състои от отделни редове. Ако редът започва с # или е празен, той се игнорира. Всички останали редове се състоят от 4 полета, разделени с двоеточия: id:runlevels:action:process когато: • id - идентификатор на низ. Това е произволна комбинация от 1 до 4 знака. Файлът inittab не може да има два реда със същия идентификатор; • нива на изпълнение - нива на изпълнение, на които ще се използва тази линия. Нивата се посочват с цифри или букви без разделители, например 345; • процес - процес, който трябва да се изпълнява на определени нива. С други думи, това поле указва името на програмата, която се извиква при преминаване към определените нива на изпълнение; • действие - действие. Полето за действие съдържа ключова дума, която определя допълнителни условия за изпълнение на командата, зададена от полето на процеса. Валидни стойности за полето за действие: • respawn - рестартиране на процеса в случай на неговото завършване; • веднъж - изпълнява процеса само веднъж при преминаване към определено ниво; • изчакайте - процесът ще бъде стартиран веднъж при преминаване към определено ниво и init ще изчака завършването на този процес, преди да продължи да работи; • sysinit - тази ключова дума показва действия, извършени по време на процеса на стартиране на системата, независимо от нивото на изпълнение (полето за нива на изпълнение се игнорира). Процесите, маркирани с тази дума, започват преди процесите, маркирани с думите boot и bootwait; • boot - процесът ще бъде стартиран на етапа на зареждане на системата, независимо от нивото на изпълнение; • bootwait - процесът ще бъде стартиран на стадия на зареждане на системата, независимо от нивото на изпълнение, и init ще изчака неговото завършване; • initdefault - редът, в който тази дума е в полето за действие, определя нивото на изпълнение, към което системата се превключва към подразбиране. Полето на процеса в този ред се игнорира. Ако нивото на изпълнение по подразбиране не е определено, процесът init ще изчака потребителя, който стартира системата, да го въведе от конзолата; • off - игнорира този елемент; • powerwait - позволява на init процеса да спре системата при загуба на мощност. Използването на тази дума предполага, че има непрекъснато захранване (UPS) и софтуер, който следи състоянието на UPS и информира init, че захранването е изключено; • ctrlaltdel - позволява на init да рестартира системата, когато потребителят натисне клавишната комбинация ‹Ctrl› + ‹Alt› + ‹Del› на клавиатурата. Обърнете внимание, че системният администратор може да дефинира действия с помощта на клавишната комбинация ‹Ctrl› + ‹Alt› +‹ Delta, например, игнорирайки натискането на тази комбинация (която е доста разумна в система с много потребители). Този списък не е изчерпателен. За повече информация относно inittab файла вж. Man (8), inittab (5) и getty (8) man страници. Обработката на файла / etc / inittab от init процеса започва в режим на единичен потребител (ниво 1), в който единственият потребител е коренният потребител, изпълняван от конзолата. Първо, init намира ред, който определя кое ниво на изпълнение се стартира по подразбиране: id:3:initdefault: Това ще бъде нивото, на което системата ще стартира и ще работи след зареждането, така че е естествено нива 0 и 6 да не могат да бъдат определени в линията initdefault. След това init изпълнява командите, посочени в низа с ключовата дума sysinit. В стандартната конфигурация тук се изпълнява скриптът rc.sysinit от директорията /etc/rc.d. След това процесът init сканира файла / etc / inittab и изпълнява скриптовете, съответстващи на еднопотребителското ниво (1 във второто поле на реда) до всички нива (линии с празно второ поле) и ниво по подразбиране. В реда, съответстващ на нивото по подразбиране, се нарича rc скрипт от директорията /etc/rc.d. Този скрипт е един и същ за всички нива (т.е. задължително се нарича без значение на какво ниво на изпълнение е заредена системата), само в зависимост от нивото на изпълнение, съответната стойност на параметъра за повикване се предава на него, така че например за ниво 3 Script извикването се извършва от тип string l3:3:wait:/etc/rc.d/rc 3 Функциите, изпълнявани от rc.sysinit и rc скриптовете, ще бъдат подробно разгледани по-долу, в раздел. 8.2.2, а сега ще се върнем към кратък преглед на действията на init процеса. Следващата важна функция, която този процес изпълнява (на нива от 2 до 5), е стартирането на шест виртуални конзоли (getty процес), които позволяват на потребителите да влизат в системата от терминалите. За да направите това, init създава процеси, наречени getty process (от get tty, получаване на терминал), и следи кой процес се отваря на кой терминал. Всеки getty процес задава собствена група процеси, използвайки функцията setpgrp системна функция, отваря отделна терминална линия и обикновено спира, докато се изпълнява отворената функция, докато машината има хардуерна връзка с терминала. Когато отворената функция върне контрола, процесът getty изпълнява програма за вход (login), която изисква от потребителите за да се идентифицират с потребителско име и парола. Ако потребителят се е регистрирал успешно, програмата за влизане най-накрая стартира командния процесор на черупката и потребителят започва да работи. Тази обвивка на обвивката се нарича "shell shell" (входна обвивка, интерпретатор на команда за влизане). Процесът, свързан с обвивката, има същия идентификатор като първоначалния getty процес, така че логинската обвивка е процесът, водещ процеса. Ако потребителят не е успял да се регистрира успешно, програмата за регистрация завършва след определен период от време, затваряйки отворената терминална линия, а процесът на иницииране генерира следващия getty процес за тази линия, като отваря терминала вместо завършения. След като изтеглянето приключи, init продължава да работи във фонов режим, проследявайки промените в състоянието на системата. Например, ако командата telinit е издадена за промяна на нивото на изпълнение, процесът init ще гарантира, че командите, зададени за новото ниво от файла / etc / inittab, се изпълняват. Този файл се препрочита отново в случай на HUP сигнал; Тази функция премахва необходимостта да се рестартира системата, за да се направят промени в първоначалната конфигурация. По този начин, първоначалният процес на зареждане на init се намира в RAM и след получаване на съответните сигнали, изпълнява отново цикъла на четене от файла / etc / inittab на инструкции за това какво да прави и този набор от инструкции е различен за различните нива на изпълнение. Когато суперпотребителят спре системата (с командата shutdown), тя е init, която прекратява всички други изпълняващи се процеси, демонтира всички файлови системи и спира процесора. забележка В горното описание много важни детайли са пропуснати. По-подробно описание може да бъде намерено в man страниците за init (8), inittab (5) и getty (8), както и в документите "Serial HOWTO" на Linux Documentation Project. забележка Ако неправилно промените файла / etc / inittab, системата може да спре зареждането. Така че, преди да направите каквито и да е промени в този файл, поне запазете зареждащата дискета и запишете копие на оригиналния файл в случай на фатални грешки. e-reading.club

# Как да конфигурирате настройките на boot loader на GRUB2

Ubuntu и повечето други дистрибуции на Linux вече използват зареждащия диск GRUB2. Можете да промените настройките му, за да изберете операционната система по подразбиране, да зададете фоново изображение и да изберете колко дълго GRUB ще отброява преди да стартирате автоматично операционната система по подразбиране.

Конфигурирахме GRUB2 на Ubuntu 14.04 тук, но процесът трябва да е подобен за други Linux дистрибуции. Възможно е да сте персонализирали първоначалните настройки на GRUB, като сте редактирали файла menu.lst в миналото, но сега процесът е различен.

## GRUB2 Основни конфигурации

GRUB2 не използва файл menu.lst. Вместо това основният конфигурационен файл е файлът /boot/grub/grub.cfg. Не трябва обаче да редактирате този файл ръчно! Този файл е само за собственото използване на GRUB2. Тя автоматично се създава, като се стартира **актуализация-плюскане** команда като корен - с други думи, като стартирате **sudo update-grub** на Ubuntu.

Вашите собствени настройки на GRUB се съхраняват във файла / etc / default / grub. Редактирайте този файл, за да промените настройките на GRUB2. Скриптовете също се намират в директорията /etc/grub.d/. Например, в Ubuntu, тук има скриптове, които конфигурират темата по подразбиране. Също така има скрипт os-prober, който проверява вътрешните твърди дискове на системата за други инсталирани операционни системи - Windows, други Linux дистрибуции, Mac OS X и т.н. - и автоматично ги добавя към менюто на GRUB2.

Когато стартирате командата update-grub, GRUB автоматично комбинира настройките от / etc / default / grub файла, скриптовете от директорията /etc/grub.d/ и всичко останало, създавайки / boot / grub / grub. cfg файл, който се чете при стартиране.

С други думи, за да персонализирате настройките си за GRUB2, ще трябва да редактирате файла / etc / default / grub и след това да стартирате**sudo update-grub** команда.



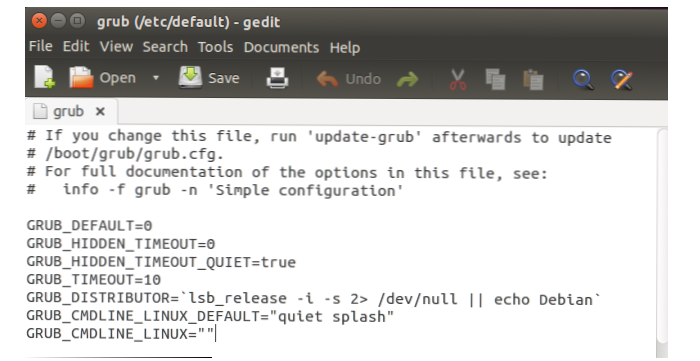
## Редактирайте GRUB конфигурационния файл

Отворете файла / etc / default / grub за редактиране в стандартен текстов редактор. Ако искате да използвате графичен текстов редактор, отворете терминал - или натиснете Alt + F2 - и изпълнете следната команда:

gksu gedit / etc / default / grub

За лесен за използване редактор на базата на терминали - Нано - използвайте следната команда. Можете да използвате всеки текстов редактор, който ви харесва, разбира се - включително стандартния текстов редактор vi.

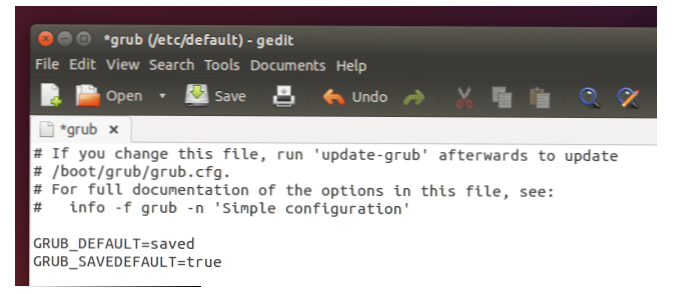
sudo nano / etc / default / grub



Файлът / etc / default / grub е кратък и трябва да бъде лесен за редактиране. Както при всеки друг конфигурационен файл, трябва да промените опциите до желаното състояние и след това да промените файла. Ако някоя от опциите по-долу не се появи във файла, добавете го на нов ред. Ако е така, редактирайте съществуващата линия, вместо да добавите дублирана.

**Изберете стандартната операционна система**: Променете **GRUB\_DEFAULT =**линия. По подразбиране, **GRUB\_DEFAULT = 0** използва първия запис като по подразбиране - променете номера на 1, за да използвате втория запис, 2, за да използвате третия запис или така нататък. Можете също да използвате **GRUB\_DEFAULT = спасени** и GRUB автоматично ще зареди последната операционна система, която сте избрали всеки път, когато стартирате. Можете също да посочите етикет в кавички. Например, ако имате операционна система, наречена Windows 7 (товарач) в списъка си с операционни системи, можете да го използвате **GRUB\_DEFAULT = "Windows 7 (товарач)"**

**Запазване на стандартната операционна система**: Ако изберете **GRUB\_DEFAULT = спасени**, вие също трябва да добавите a **GRUB\_SAVEDEFAULT = вярно** линия - в противен случай няма да работи.

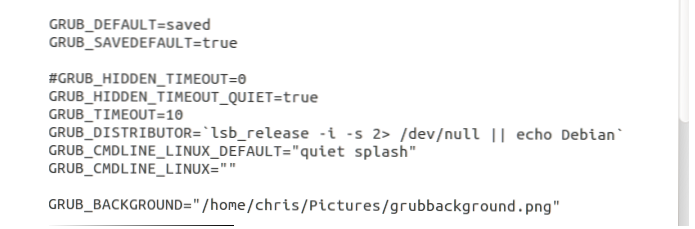


**Изберете дали GRUB е скрит**: При инсталирана само една операционна система, Ubuntu по подразбиране GRUB автоматично стартира операционната система с опцията GRUB\_HIDDEN\_TIMEOUT = 0. Тази опция указва, че GRUB ще бъде скрита и автоматично ще се зареди на стандартната операционна система след 0 секунди - незабавно, с други думи. Все още можете да осъществите достъп до менюто, като задържите Shift, докато компютърът ви зарежда. За да зададете по-голямо време, използвайте нещо подобно **GRUB\_HIDDEN\_TIMEOUT = 5** - ГРУБ ще покаже празен екран или начален екран за пет секунди, през който можете да натиснете който и да е клавиш, за да видите менюто. За да предотвратите автоматично скриване на GRUB, коментирайте реда - просто добавете # пред него, така че да се чете **# GRUB\_HIDDEN\_TIMEOUT = 0** .

**Контролирайте изтичането на менюто на GRUB**: Ако GRUB не се скрива автоматично, ще видите менюто всеки път, когато компютърът ви стартира. GRUB автоматично ще направи бот на операционната система по подразбиране след период от време, обикновено десет секунди. През това време можете да изберете друга операционна система или да я оставите автоматично да се зарежда. За да промените периода на изчакване, редактирайте **GRUB\_TIMEOUT = 10**line и въведете броя на секундите, които харесвате. (Не забравяйте, че това се използва само ако GRUB не е скрит.) За да предотвратите автоматично зареждане на GRUB и винаги да чакате да изберете операционна система, променете линията на **GRUB\_TIMEOUT = -1**

**Изберете фоново изображение**: The **GRUB\_BACKGROUND** линията контролира дали се използва фоново изображение - по подразбиране GRUB използва черно-бял монохромен вид. Можете да добавите линия като **GRUB\_BACKGROUND = "/ Начало / потребител / Снимки / background.png"** за да зададете графичен файл, който GRUB ще използва.

Файлът с изображения трябва да отговаря на различни спецификации. GRUB поддържа JPG / JPEG изображения, но те са ограничени до 256 цвята - затова вероятно няма да искате да използвате JPG изображение. Вместо това, вероятно ще искате да използвате PNG изображение, което може да има произволен брой цветове. Можете също да използвате TGA файл с изображения.



## Направете ефектите си на промени

За да влезете в сила промените, просто запишете текстовия файл - File> Save in Gedit или Ctrl + O и след това Enter, за да запазите файла в Nano - и след това стартирайте **sudo update-grub** команда. Промените ви ще станат част от файла grub.cfg и ще бъдат използвани всеки път, когато стартирате компютъра си.



Това не са всички настройки на GRUB, но те са едни от най-често променяните.Други настройки могат да бъдат персонализирани във файла / etc / default / grub или чрез редактиране на скриптовете в директорията /etc/grub.d.

Ако не искате да редактирате файловете ръчно, може да намерите графични инструменти за персонализиране на GRUB2 в софтуерните хранилища на Linux. Горният метод трябва да работи дори и на Linux дистрибуции, където такива инструменти не са лесно достъпни или просто имате достъп до командния ред и искате да го направите на ръка.

Ако на този ред бъде премахнат коментара операционната система ще се опитва да зарежда модула 8139too при всяко стартиране (този модул отговаря за мрежовите карти с чип Realtek 8139). Чрез втория файл може този модул да се присвои към определено устройство. Например:

*alias eth0 dmfe*

*alias eth1 8139too*

*alias char-major-195 nvidia*

Първият ред присвоява към устройство отговарящо за първата мрежова карта модула dmfe (мрежови карти Davicom), а на втората мрежова карта модула 8139too. Последния ред указва на операционната система, че за char устройство с главен номер 195 отговаря модула nvidia (модула на видео картите с чип на Nvidia).

**5. НАСТРОЙКА НА ПУСКАНЕТО И СПИРАНЕТО НА ОПЕРАЦИОННАТА СИСТЕМА**

Пускането и спирането на операционната система се извършва от шел скрипове намиращи се w директорията /etc/rc.d/. Линукс има седем режима в които може да работи: режим на единичен потребител, многопотребителски режим, режим с графична среда, режим на рестартиране и режим на изключване. Те са обозначени със следните номера – 1, 3, 4, 6, 0 наречени runlevels. Номерата 2 и 5 не се използват. За всеки един от режимите в директорията има по един шел скрипт:

* *rc.0* – това е символична връзка към rc.6. Oтговаря за runlevel 0 (изключване на операционната система)

* *rc.K*– отговаря за runlevel 1 (еднопотребителски режим)

* *rc.M*– отговаря за runlevel 3 (многопотребителски режим)

* *rc.4* – отговаря за runlevel 4 (режим с графична среда)

* *rc.6*– отговаря за runlevel 6 (рестартиране на операционната система)

* *rc.S* – не отговаря за никой от режимите тъй като се стартира преди тях. Използва се за инициализация на операционната система. Тук се прави проверка за коректно изключване на операционната система, стартиране на модулите и т.н.

Всеки един от тези скриптове стартира множество програми необходими за нормалната работа в дадения runlevel. По подразбиране системата влиза в runlevel 3. Това може да се промени чрез промяна на файла */etc/inittab*.

*id:3:initdefault:*

трябва да се промени на

*id:4:initdefault:*

Освен тези скриптове, в тази директория има още няколко важни скрипта:

* *rc.local*– първоначално този скрипт е празен. Тук администратора може да добавя собствени команди, които ще се изпълняват при всяко стартиране на операционната система.

* *rc.inet2* – от тук се стартират скриптовете на сървърите които са инсталирани.

* *rc.inetd* – в последната версия, Slackware добавиха този прост скрипт който позволява да се рестартира inetd. Това в старите версии ставаше с командата *killall -HUP inetd*.

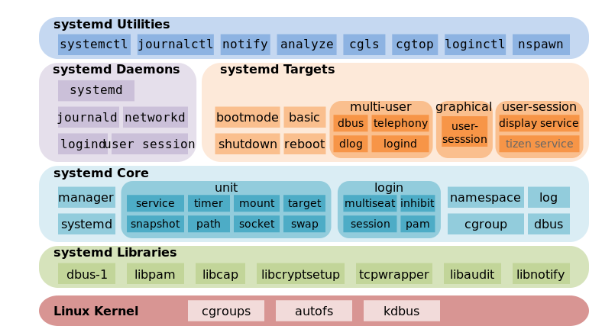
* *rc.sysvinit*– позволява автоматично разпознаване и стартиране на скриптове предназначени за Red Hat. Тъй като Red Hat (Mandrake, Debian, Fedora, SUSE) използва Sys V тип на скриптовете, а Slackware BSD стил, чрез този файл се постига съвместимост.

* *rc.syslog*– управлява системния процес наречен *syslog*. Той следи и записва възниканлите грешки и по-важни събития в определени журнални файлове.

* *rc.serial* – управлява серийните устройства, като чете конфигурацията от файла /etc/serial.conf

Останалите файлове са скриптове за управление на инсталираните сървъри и зависят от направената инсталация.

Разбиране и използване на Systemd



Харесва ли ви или не, systemd е тук, за да остане, така че можем да знаем какво да правим с него.

systemd е спорен по няколко причини: Това е заместител на нещо, което много потребители на Linux не мислят, че трябва да бъдат заменени, а лудориите на системните разработчици не са спечелили сърцата и умовете. Но по-скоро обратното, както е видно от тази известна нишка LKML, където Линус Торвалдс забрани системния dev Kay Sievers от ядрото на Linux.

Изкушаващо е да позволим на личности да попречат. Колкото и да е забавно, тъй като е да се разгласяват и излъчват и излъчват цветни епитети, това е нещо различно. За толкова много години Linux беше доволен от SysVInit и BSD init. След това се появиха допълнителни управленски услуги като командите за услуги и chkconfig. Които би трябвало да улеснят управлението на услугите, но за мен бяха само още неща, които трябваше да научат, които не правеха задачите по-лесни, а по-скоро по-бързи.

След това дойде Upstart и systemd, с всички видове сложен аддон за поддържане на SysVInit съвместимост. Което е хубаво нещо да се направи, но късмет да го разберем. Сега Upstart се оттегля в полза на systemd, вероятно в Ubuntu 14.10, и ще намерите тон на systemd libs и инструменти в 14.04. Само за кикотене, погледнете списъка с файлове в пакета systemd-services в Ubuntu 14.04:



Разгледайте страниците на човека, за да видите какво правят всички тези неща.

Винаги е страшно, когато разработчиците започнат да си играят с ключови подсистеми на Линукс, тъй като ние сме доста затворени с каквото и да ни наложат. Ако не харесваме конкретно софтуерно приложение или десктоп среда или команда, има няколко алтернативи и е лесно да се използва нещо друго. Но съществените подсистеми имат дълбоки куки в ядрото, всякакви управленски скриптове и зависимости от софтуерни пакети, така че подмяната на една не е тривиална задача.

Ето защо моралът се променя, компютрите неизбежно стават все по-сложни и в крайна сметка всичко се развива. Или не, но липсва способността да се оформят събитията според собствените ни предпочитания.

Първи стъпки на системата

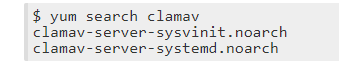
Red Hat е изобретателят и основният бустер на systemd, така че най-добрите дистрибуции за игра с него са Red Hat Enterprise Linux, RHEL клонинги като CentOS и Scientific Linux, и разбира се, добрият Fedora Linux, който винаги се доставя с най-новите, най-големите, и кръвотечение. Моите примери са от CentOS 7.

Опитните потребители на RH все още могат да използват услугата и chkconfig в RH 7, но това е дълго време, за да ги изхвърлите в полза на родните системни помощни програми. systemd ги е изпреварил и услугата и chkconfig не поддържат родните системни услуги.

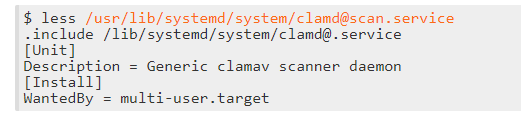
Нашата любима / etc / inittab вече не съществува. Вместо това, имаме / etc / systemd / system / директория с пълни символи за файлове в / usr / lib / systemd / system /. / usr / lib / systemd / system / съдържа скриптове; за да стартирате услуга при зареждане, трябва да бъде свързана с / etc / systemd / system /. Командата systemctl прави това за вас, когато активирате нова услуга, като този пример за ClamAV:



Откъде знаете името на init скрипта и откъде идва? На Centos7 те са разделени в отделни пакети. Много сървъри (например Apache) не са уловили tosystemd и нямат инициализирани системни скриптове. ClamAV предлага системни и SysVInit скриптове, така че можете да инсталирате този, който предпочитате:

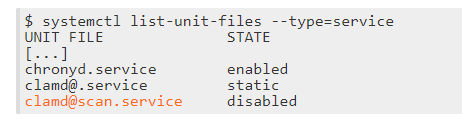


И така, какво е вътре в тези скриптове? Можем сами да видим:



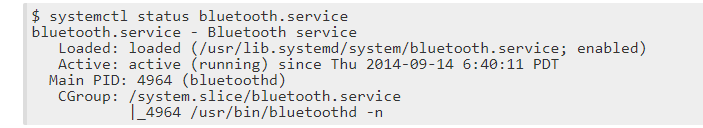
Сега можете да видите как systemctl знае къде да инсталира символната връзка и този скрипт включва също зависимостта от друга услуга, clamd @ .service.

systemctl показва състоянието на всички инсталирани услуги, които имат инициализирани скриптове:



Има три възможни състояния за услуга: активирана или забранена и статична. Активирано означава, че има символична връзка в директория .wants. Инвалидите означава, че не. Статично означава, че услугата липсва секцията [Install] в своя първоначален скрипт, така че не можете да я активирате или деактивирате. Статичните услуги обикновено са зависимости от други услуги и се контролират автоматично. Можете да видите това в ClamAV примера, като clamd @ .service е зависимост от clamd@scan.service и се изпълнява само когато clamd@scan.service се изпълнява.

Нито едно от тези състояния не ви казва, ако се изпълнява услуга. Командата ps ще ви каже или ще използвате systemctl, за да получите по-подробна информация:



systemctl ви казва всичко, което искате да знаете, ако знаете как да питате.