# Содержание

[Содержание 1](#_Toc484007196)

[Используемая система и версия ядра 2](#_Toc484007197)

[Инсталлируемая программа в Windows 3](#_Toc484007198)

[Создание демона в Linux 13](#_Toc484007199)

[Источники 22](#_Toc484007200)

# Используемая система и версия ядра

a) Windows

|  |
| --- |
| Процессор: Intel(R) Core(TM) i5-2450 CPU @2.50GHz 2.50GHz  ОЗУ: 5,00 Гб  Тип системы: Windows 10 Pro x64. |

б) Linux

|  |
| --- |
| osboxes@osboxes:~/Documents$ **lsb\_release -a**  No LSB modules are available.  Distributor ID: Ubuntu  Description: Ubuntu 16.04.2 LTS  Release: 16.04  Codename: xenial  osboxes@osboxes:~/Documents$ **cat /proc/version**  Linux version 4.8.0-53-generic (buildd@lgw01-56) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4) ) #56~16.04.1-Ubuntu SMP Tue May 16 01:18:56 UTC 2017 |

# Инсталлируемая программа в Windows

Для примера будет использоваться программа, подробное описание которой приведено в предыдущем отчете. Программа выводит информацию о количестве логических дисков и для каждого жесткого диска: его название, общий объем, объем свободного и занятого места.

Для создания инсталлируемого приложения воспользуемся бесплатной программой **Inno Setup**.

Inno Setup – бесплатный инструментарий для создания инсталляторов на базе сценариев. Среди главных особенностей продукта – работа со всеми версиями ОС Windows, расширенная поддержка 64-битных приложений, настраиваемые типы установки, встроенный препроцессор и мощный язык сценариев на основе Pascal. Inno Setup – проект с открытым исходным кодом. Используемый язык - Borland Delphi.

Документ разделен на секции, и каждая из них отвечает за определенную задачу инсталлятора соответственно. Всего имеется два типа секций – с параметрами и «директива-значение». Inno использует язык [RemObjects Pascal Script](http://www.remobjects.com/ps.aspx). Сценарии отвечают за добавление новых опции, создание интерфейса, вызов файлов или библиотек, настраиваемые действия (custom actions) и т. д. В рабочей папке Inno Setup находятся примеры скриптов (также доступны в меню компилятора). По умолчанию в Inno Setup входит компилятор сценариев Compiler IDE.

Самый простой вариант создания инсталлятора – обратиться к пошаговому мастеру настройки Inno Script Studio. Здесь указывается информация о продукте, координаты программы, файл лицензии и другие данные, языки локализации, опции компилятора. Впоследствии проект доступен для компиляции, отладки либо модификации полученного сценария в окне редактора (Рисунок 1 – Рисунок 6).

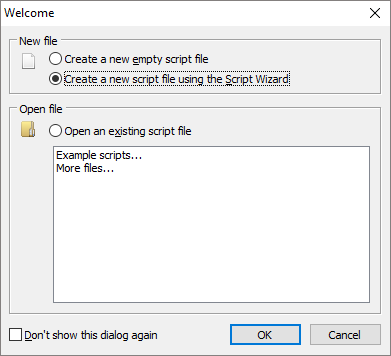


Рисунок 1. Приветственное окно Inno Setup

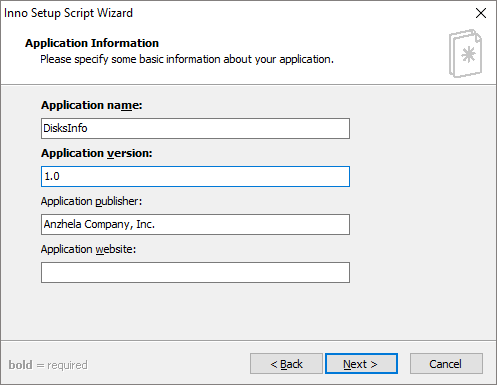


Рисунок 2. Настройка создаваемого приложения

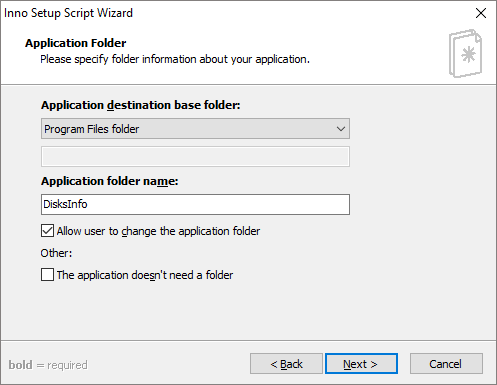


Рисунок 3. Настройка создаваемого приложения

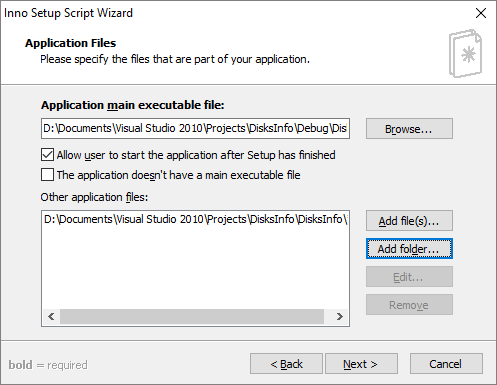


Рисунок 4. Настройка создаваемого приложения

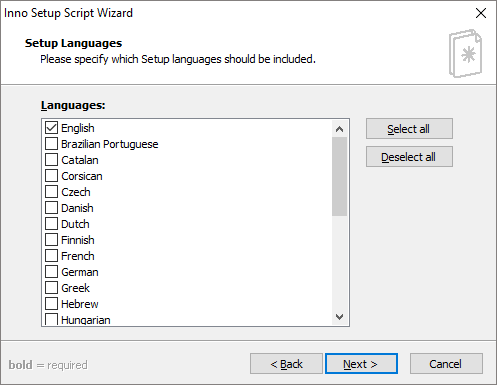


Рисунок 5. Настройка создаваемого приложения

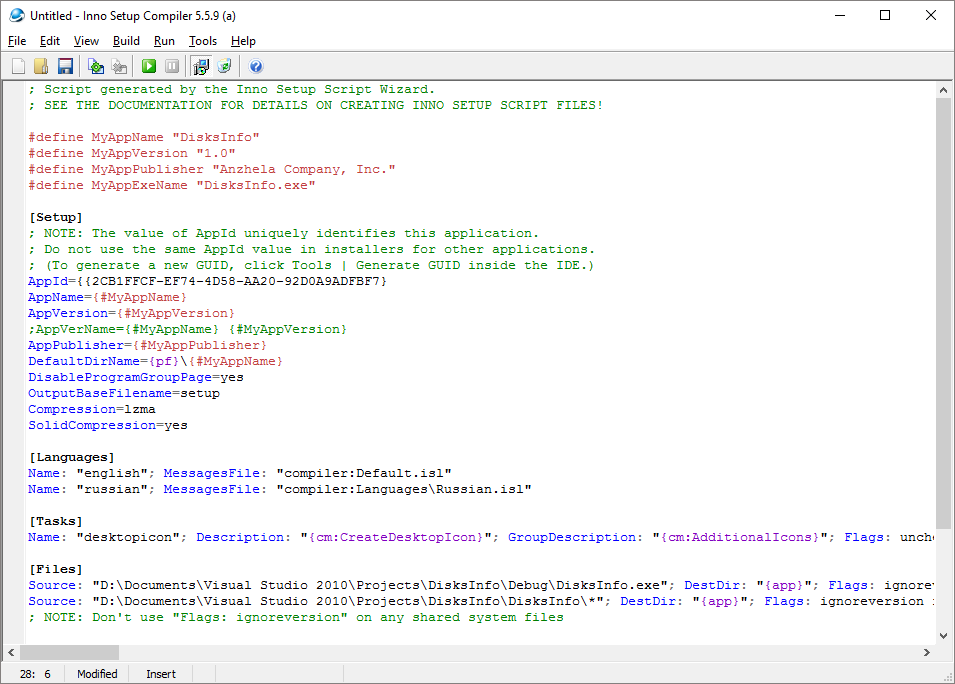


Рисунок 6. Сгенерированный скрипт

Редактор Inno Script Studio доступен на русском языке. Значительно проще настраивать проект последовательно, переходя по секциям, а не создавая сценарий с нуля. При добавлении нового элемента изменения вносятся в код сценария, синтаксис подсвечивается, код соответствующим образом разбит на секции ([Setup], [Types], [Components]…).

Рассмотрим подробно код скрипта и при необходимости внесем изменения.

Первым шагом определим необходимые константы:

|  |
| --- |
| #define MyAppName "DisksInfo"  #define MyAppVersion "1.0"  #define MyAppPublisher "Anzhela Company, Inc."  #define MyAppExeName "DisksInfo.exe" |

Тело скрипта разделяется на секции, каждая из которых несет свое функциональное назначение. Обязательная секция [Setup] задает глобальные параметры работы инсталлятора и деинсталлятора:

|  |
| --- |
| [Setup]  ; NOTE: The value of AppId uniquely identifies this application.  ; Do not use the same AppId value in installers for other applications.  ; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)  ; Уникальный идентификатор приложения, сгенерированный через Tools > Generate GUID  AppId={{2CB1FFCF-EF74-4D58-AA20-92D0A9ADFBF7}  ; Прочая информация, отображаемая при установке  AppName={#MyAppName}  AppVersion={#MyAppVersion}  ;AppVerName={#MyAppName} {#MyAppVersion}  AppPublisher={#MyAppPublisher}  ; Путь установки по-умолчанию  ; {pf} – путь в каталог Program Files  DefaultDirName={pf}\{#MyAppName}  ; Имя группы в меню "Пуск"  DisableProgramGroupPage=yes  DefaultGroupName={#Name}  ; Каталог, куда будет записан собранный setup и имя исполняемого файла  OutputDir=D:\DisksInfo  OutputBaseFilename=DisksInfo  ; Алгоритм сжатия  Compression=lzma  ; Все файлы сжимаются одновременно  SolidCompression=yes |

Для поддержки нескольких языков существует секция [Languages]. При отсутствии данной секции будет использоваться английский язык. Каждая строка в данной секции задает один из используемых при установке языков. Параметр *Name* – «имя» языка, допускаются общепринятые двухбуквенные сокращения («en», «ru», «de» и так далее). Параметр *MessagesFile* сообщает компилятору в каком месте взять шаблон сообщений, выводимых при инсталляции. Эти шаблоны берем в каталоге компилятора Inno Setup, о чем мы сообщаем директивой compiler. Для английского языка годится шаблон *Default.isl*, для русского – *Languages\Russian.isl*

|  |
| --- |
| ; Поддержка языков  [Languages]  Name: "english"; MessagesFile: "compiler:Default.isl"  Name: "russian"; MessagesFile: "compiler:Languages\Russian.isl" |

Обычно установщик предлагает нам, например, определиться, хотим мы или не хотим создать ярлык на рабочем столе. Такие опции установки определяются необязательной секцией [Tasks].

|  |
| --- |
| [Tasks]  Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked |

Здесь *Name* задает имя операции – «desktopicom» – создание иконки на рабочем столе; *Description* – описание чекбокса с опцией, которое увидит пользователь. Конструкция задает стандартный текст сообщения, соответствующий выбранному в начале инсталляции языку. Параметр *GroupDescription* – заголовок группы чекбоксов с опциями. Параметр *Flags* задает определенные действия и состояния элементов управления, в данном случае указывая, что галочка «создать ярлык на рабочем столе» должна быть снята.

Теперь укажем, какие файлы надо включить в дистрибутив и где их надо поместить при установке. Для этого используется обязательная секция [Files].

|  |
| --- |
| [Files]  Source: "D:\Documents\Visual Studio 2010\Projects\DisksInfo\Debug\DisksInfo.exe"; DestDir: "{app}"; Flags: ignoreversion  Source: "D:\Documents\Visual Studio 2010\Projects\DisksInfo\DisksInfo\\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs  ; NOTE: Don't use "Flags: ignoreversion" on any shared system files |

*Source* – путь к файлу-источнику. *DestDir* – каталог установки, переменная {*app*} содержит путь, выбранный пользователем в окне установщика. *Flags* — флаги. В нашем примере для исполняемого файла: игнорирование версии программы при перезаписи исполняемого модуля, если он уже существует в системе (ignorevarsion).

Наконец укажем компилятору, где брать иконки для размещения в меню программ и на рабочем столе.

|  |
| --- |
| [Icons]  Name: "{commonprograms}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"  Name: "{commondesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; Tasks: desktopicon |

Попробуем скомпилировать созданный скрипт инсталлятора и запустить:

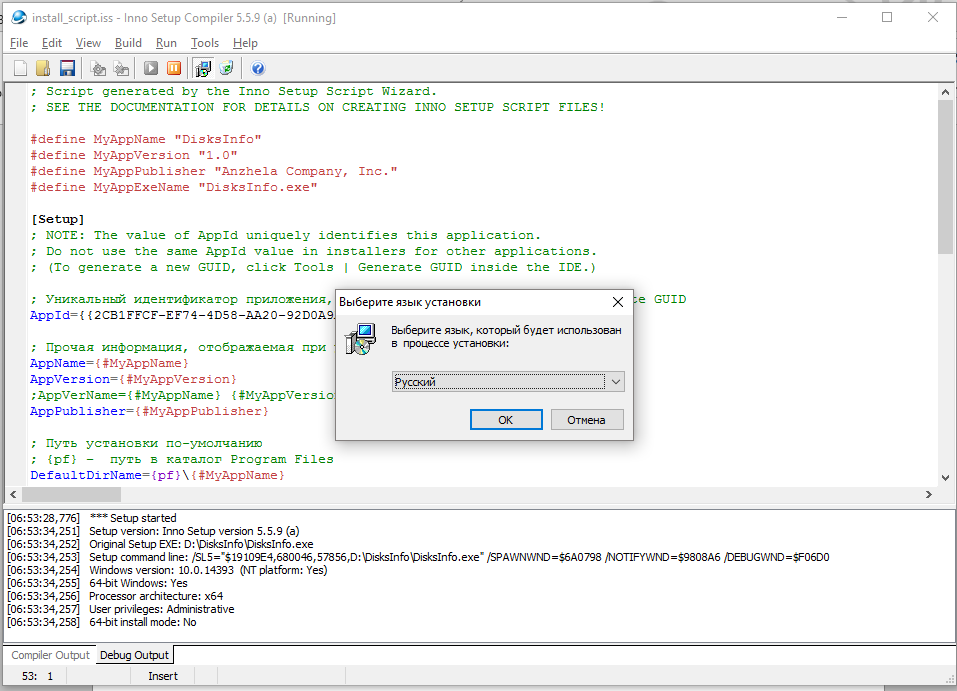


Рисунок 7. Компиляция инсталлируемой программы

Далее наша программа будет инсталлироваться как любое стандартное приложение Windows:

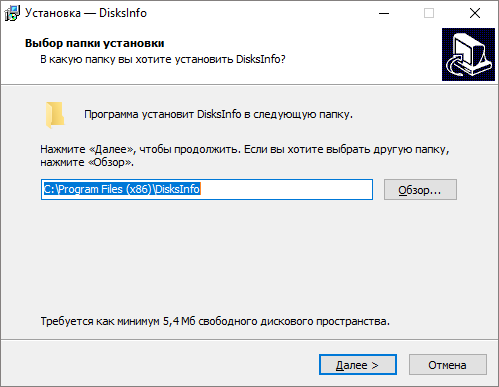


Рисунок 8. Инсталляция моей программы

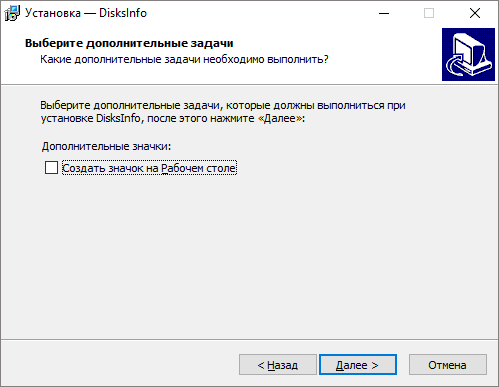


Рисунок 9. Инсталляция программы DisksInfo

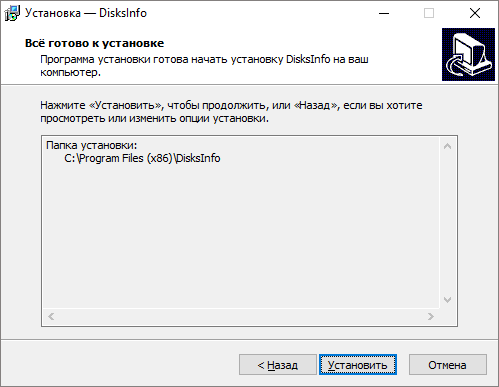


Рисунок 10. Инсталляция программы DisksInfo

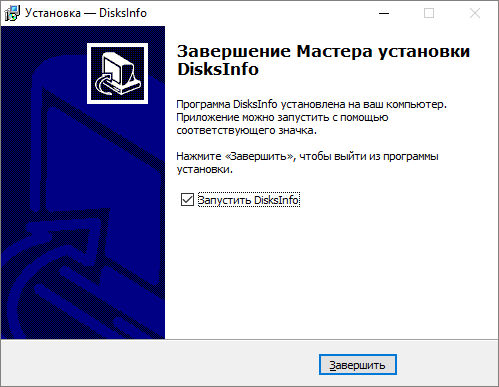


Рисунок 11. Инсталляция программы DisksInfo

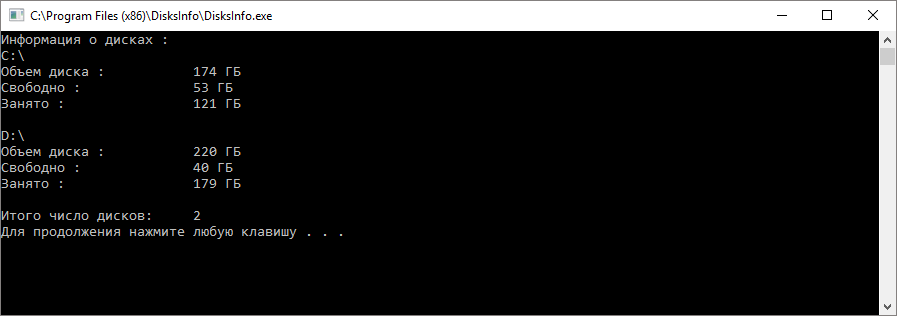
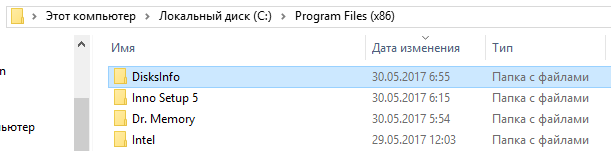


Рисунок 12. Запуск установленной программы

В системной папке ***Program Files (x86)*** появилась директория с нашим приложением:



# Создание демона в Linux

Демонами в мире Unix традиционно называются процессы, которые не взаимодействуют с пользователем напрямую. У процесса-демона нет управляющего терминала и нет, соответственно, пользовательского интерфейса. Для управления демонами приходится использовать другие программы. Само название «демоны» возникло благодаря тому, что многие процессы этого типа большую часть времени проводят в ожидании какого-то события. Когда это событие наступает, демон активизируется (выпрыгивает, как чертик из табакерки), выполняет свою работу и снова засыпает в ожидании события. Следует отметить, что многие демоны, такие как, например, Web-сервер или сервер баз данных, могут отбирать на себя практически все процессорное время и другие ресурсы системы. Такие демоны гораздо больше работают, чем спят.

Модифицируем нашу программу под Linux для реализации в виде демона, который каждые 60 секунд будет записывать в лог информацию о размере дисков:

Листинг 1. DiskInfo.spp

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/stat.h>  #include <sys/types.h>  #include <time.h>  #include <unistd.h>  #include <errno.h>  #include <fcntl.h>  #include <syslog.h>  #include <sys/vfs.h>  #include <iostream>  using namespace std;  char\* getTime() {  time\_t now;  struct tm \*ptr;  static char tbuf[64];  bzero(tbuf,64);  time(&now);  ptr = localtime(&now);  strftime(tbuf,64, "%Y-%m-%e %H:%M:%S", ptr);  return tbuf;  }  char\* getDiscSize ()  {  static char str[256];  bzero(str, 256);  struct statfs fs;  unsigned long long free, size,blocks;  statfs("/home/osboxes/Desktop/fstat/file\_name.txt", &fs);  free = fs.f\_bfree;  size = fs.f\_bsize;  blocks=fs.f\_blocks;  double blocks\_gb=blocks\*size/1000000000.;  double busy\_gb=(blocks-free)\*size/1000000000.;  double free\_gb=free\*size/1000000000.;  sprintf(str, "Total number of bytes -> %.4f \nNumber of free bytes -> %.4f \nNumber of busy bytes -> %.4f \n", blocks\_gb, free\_gb, busy\_gb);  return str;  }  int writeLog(char msg[256]) {  FILE \* pLog;  pLog = fopen("/home/osboxes/Documents/daemon.log", "a");  if(pLog == NULL) {  return 1;  }  char str[312];  bzero(str, 312);  strcpy(str, getTime());  strcat(str, " ==========================\n");  strcat(str, msg);  strcat(str, "\n");  fputs(str, pLog);  fclose(pLog);  return 0;  }  int main(int argc, char\* argv[]) {  char logStart[] = {"Daemon Start"};  writeLog(logStart);  char \*log;  while(1) {  log = getDiscSize();  writeLog(log);  sleep(60); // 1 min  }  return 0;  } |

Выполнили программу:

|  |
| --- |
| osboxes@osboxes:~/Documents$ **g++ -o DisksInfo.bin DisksInfo.cpp** |

Каждый демон имеет **файл конфигурации**, по которому системный менеджер определяет, как запускать, останавливать и вообще управлять данной службой.

Создадим конфигурационный файл демона **DisksInfo.service**.

Листинг 1. DiskInfo.spp

|  |
| --- |
| [Unit]  Description=Information about disks  After=network.target  [Install]  WantedBy=multi-user.target  [Service]  Type=simple  ExecStart="/home/osboxes/Documents/DisksInfo.bin"  RuntimeMaxSec=3600  Restart=always  RestartSec=5  KillMode=process  KillSignal=SIGTERM  SendSIGKILL=yes  TimeoutStopSec=5 |

**Секция [UNIT]**

Содержит общую информацию о модуле, не зависящую от его конкретного типа. Ниже приведена таблица с описанием возможных переменных данной секции.

|  |  |
| --- | --- |
| Description= | описывает информацию о модуле |
| Documentation= | Список источников документации. Элементы разделяются через пробелы и могут быть следующих типов "http://", "https://", "file:", "info:", "man:" |
| Requires= | Зависимости модуля. |
| Requisite= | Аналогично полю Requires=, однако если модуль, указанный в данном поле уже не был запущен, то его запуск производиться не будет, а вся попытка запуска (транзакция) текущего модуля будет считать провальной. |
| Wants= | Аналогично Requisite= (опция с более слабыми требованиями), но при этом если модуль не получится запустить, то прерывание всей операции не наступит |
| PartOf= | Опция, которая указывает, что текущий модуль являет частью чего-то более крупного, другого модуля. Если systemd остановит или перезагрузит данный сервис, то тоже будет выполнено для модуля, указанного в данном поле. Однако связь односторонняя, перезагрузив или остановив модуль, указанный в этой опции, текущий модуль продолжит работу. |
| Conflicts= | Список модулей, несовместимых с данным. При попытке запустить модуль, будет остановлен конфликтующий модуль, и наоборот. |
| Before=, After= | Список модулей, требований, которые должны быть запущены ДО и ПОСЛЕ запуска текущего модуля. Данное поле независимо от поля Requires=, и ему подобных. Если вызвать команду отключается всех модулей, то оно производится задом наперёд. |
| AllowIsolate= | Флаг. Указывает можно ли использовать команду **systemctl isolate** для изменения уровня изоляции модуля. По умолчанию false. |
| SourcePath= | Путь к конфигурационному файлу, который будет создан под данную систему на основе исходного конфига. |

**Секция [INSTALL]**

Секция с информацией об установке модулей.

|  |  |
| --- | --- |
| Alias= | Возможные имена модуля. В момент установки, команда **systemctl enable** создаст символьные ссылки на модули с указанными в поле именами |
| WantedBy=, RequiredBy= | Список с именами связанных модулей, разделенных через запятую. |
| Also= | Список модулей, который будут установлен/удалены автоматически при установке текущего модуля. |

**Секция [SERVICE]**

Основная секция конфигурации.

|  |  |
| --- | --- |
| KillMode = [control-group, process, mixed, none] | ***control-group*** – отключить все процессы из данной контрольной группы процессов (process group), состоящей из его прямого родителя (и других предков), братьев и сестер, а также детей (и прочих потомков). ***process*** – только главный процесс. ***mixed*** – главному процессу посылается сигнал SIGTERM, затем всем оставшимся процессам из данной группы посылается сигнал SIGKILL.  ***None*** – ни один процесс не будет завершен, только команда STOP убьет процесс. Процессы, оставшиеся в данной группе продолжают исполнение. |
| KillSignal= [SIGTERM, SIGKILL или другой сигнал] | По умолчанию посылается сигнал SIGTERM, обычно за ним следует сигнал SIGKILL (если опция SendSIGKILL=1 включена). |
| SendSIGHUP= [yes, no] | Флаг указывает, посылать или нет сигнал SendSIGHUP после сигнала KillSignal=. Если опция включена (yes), то сразу после KillSignal будет послан сигнал SIGHUP. По умолчанию [no]. |
| SendSIGKILL= [yes, no] | Флаг, указывающий следует ли посылать нетерминирующийся сигнал SendSIGKILL, после стандартного сигнала KillSignal. Если опция включена, то сигнал SIGKILL будет послан спустя TimeoutStopSec секунд. По умолчанию опция включена [yes]. |
| Type= [simple, forking, oneshot, dbus, notify or idle] | ***simple*** – по умолчанию, если не определены Type и BusName, но определен параметр ExecStart. ***forking*** – традиционное поведение при создании демона в UNIX. ExecStart = вызывает процесс, который производит необходимые для денома настройки (IPC и др.), вызывает дочерний процесс с помощью fork(), а сам завершается. Созданный таким образом процесс уже является демоном. ***oneshot*** – тип похож на simple, однако, ожидается, что процесс должен выйти, прежде чем systemd запустит последующие модули. ***dbus*** – тип так же похожий на simple. Тип устанавливается по умолчанию, если BusName= задано. Systemd запустит процесс, после того как демону будет присвоено указанное в BusName= имя на Dbus шине. ***notify*** - тип так же похожий на simple. После того как демон завершил свой запуск, он должен послать сообщение через sd\_notify(3) или другим способом. Systemd продолжит загрузку осташихся модулей данного демона, только после получения сообщения. ***idle*** – почти точная копия simple: отличие в том, что бинарник демона будет выполнен в последнюю очередь, после того, как все процессы будут обслужены. Тип эффективен только при считывании вывода консоли, в остальных случаях лучше использовать simple и установленный time-out в N секунд. |
| RemainAfterExit= [yes, no] | Флаг, определяющий нужно ли службу считать активной, даже когда все ее процессы (видимо процессы ее контрольной группы) отключены. По умолчанию [no]. |
| GuessMainPID= [yes, no] | Флаг указывает systemd пытаться ли угадать PID службы, если его не удается определить достоверно. Данный флаг имеет значение, если **Type=**forking и **PIDFile**= не определен, т.к. для всех остальных типов служб PID демона известен достоверно. Алгоритм “угадывания” может прийти к не правильному заключению, если демон состоит из нескольких процессов. Если PID основного процесса не может быть определен, обнаружение отказов и автоматический перезапуск службы будут работать не надежно. По умолчанию yes. |
| PIDFile= | Параметр содержит абсолютный путь к PID файлу демона. |
| BusName= | Параметр актуален для служб с **Type=** dbus. Параметр указывает имя демона, по которому к нему можно обратиться на Dbus-шине |
| ExecStart= | Команда запуска сервиса с заданными аргументами. |
| ExecStartPre=, ExecStartPost= | Синтаксис команд аналогичный ExecStart. Все команды исполняются последовательно: сначала ExecStartPre, затем ExecStart и в завершении ExecStartPost. |
| ExecReload= | Команда для перезагрузки сервиса. Поддерживает множественные вызовы, по той же схеме, что и ExecStart= |
| ExecStop= | Команда используется для остановки сервиса. Возможно множественное объявление. После запуска команды, все процессы, относящиеся к указанному демону, завершаются в соответствии с установленным KillMode=. Если опция не установлена, то демон (все его процессы) будут убиты сигналом KillSignal=. |
| ExecStopPost= | Синтаксис похож на ExecStart: команда будет выполняется в случаях, когда демон запустился неудачно, или приходит повторная команда останова, или команда ExecStop= не помогла или не задана. |
| RestartSec= | Параметр указывает, какое время необходимо подождать до, того как перезапустить демона командой Restart=. |
| TimeoutStartSec= | Параметр отвечает за промежуток времени, который необходимо подождать, перед тем как считать, что сервис не запустился и послать ему сигнал останова. |
| TimeoutStopSec= | Время, ожидаемое после команды останова: сначала сервису дается команда SIGTERM, и если он не ответит в течении заданномго времени, то ему посылает нетерминируемый сигнал SIGKILL. |
| TimeoutSec= | Опция позволяет задать TimeoutStartSec= и TimeoutStopSec= одновременно. |
| RuntimeMaxSec= | Задает максимальное время, которое сервис может работать. Спустя указанное время сервис останавливается. Значение по умолчанию - infinty. Для демонов с типом oneshot данная опция не имеет никакого значения. |

Для запуска демона его необходимо скопировать в файл **etc/systemd/system/**:

|  |
| --- |
| osboxes@osboxes:~/Documents$ **sudo cp ~/Documents/DisksInfo.service /etc/systemd/system/** |

Затем добавить в список демонов командой **systemctl enable**:

|  |
| --- |
| osboxes@osboxes:~/Documents$  **sudo systemctl enable DisksInfo.service**  Created symlink from /etc/systemd/system/multi-user.target.wants/DisksInfo.service to /etc/systemd/system/DisksInfo.service. |

Теперь наш демон может быть запущен командой **systemctl start**:

|  |
| --- |
| osboxes@osboxes:~/Documents$  **sudo systemctl start DisksInfo** |

Проверяем статус командой **systemctl status**:

|  |
| --- |
| osboxes@osboxes:~/Documents$  **sudo systemctl status DisksInfo**  ● DisksInfo.service - Information about disks  Loaded: loaded (/etc/systemd/system/DisksInfo.service; enabled; vendor preset: enabled)  Active: active (running) since Sat 2017-05-27 19:32:02 BST; 6s ago  Main PID: 4877 (DisksInfo.bin)  CGroup: /system.slice/DisksInfo.service  └─4877 /home/osboxes/Documents/DisksInfo.bin  May 27 19:32:02 osboxes systemd[1]: Started Information about disks. |

Теперь система каждую минуту (в соответствии с настройками программы) будет заносить информацию об объеме дисков в файл **daemon.log**:

|  |
| --- |
| 2017-05-27 19:32:02 ==========================  Daemon Start  2017-05-27 19:32:02 ==========================  Total number of bytes -> 102.1224  Number of free bytes -> 97.0810  Number of busy bytes -> 5.0414  2017-05-27 19:33:02 ==========================  Total number of bytes -> 102.1224  Number of free bytes -> 97.0810  Number of busy bytes -> 5.0414  2017-05-27 19:34:02 ==========================  Total number of bytes -> 102.1224  Number of free bytes -> 97.0809  Number of busy bytes -> 5.0414  2017-05-27 19:35:02 ==========================  Total number of bytes -> 102.1224  Number of free bytes -> 97.0810  Number of busy bytes -> 5.0414  2017-05-27 19:36:02 ==========================  Total number of bytes -> 102.1224  Number of free bytes -> 97.0810  Number of busy bytes -> 5.0414 |

При внесении изменений в файл конфигурации демона, его необходимо перезагрузить командой **sudo systemctl daemon-reload.**

**Способы выключения демона**

* **systemctl stop** Простая остановка службы. Оставляет возможность запустить сервис аналогичной stop, командой start (вручную). Сервис может быть перезапущен автоматически при загрузке системы, при поступлении запроса через сокет или системную шину, при срабатывании таймера, при подключении соответствующего оборудования и т.д.

|  |
| --- |
| osboxes@osboxes:~/Documents/Daemon$ **sudo systemctl stop DisksInfoDaemon** |

Отличие между **systemctl kill** и **systemctl stop** состоит в том, что kill просто отправляет сигнал заданному процессу, в то время как stop действует по определенному в параметре ExecStop методу.

* **systemctl disable.** Позволяет отключить службу, то есть отсоединить ее от всех триггеров активации. В результате служба уже не будет автоматически запускаться ни при загрузке системы, ни при обращении к сокету или адресу на шине, ни при подключении оборудования, и т.д. Но при этом сохраняется возможность «ручного» запуска службы (командой systemctl start). Важно отметить, что служба продолжает работу до выключения ОС, либо до получения команды **systemctl stop.**

***Примечание*:** отключение службы является перманентной мерой, и действует вплоть до явной отмены соответствующей командой. Перезагрузка системы не отменяет отключения службы.

|  |
| --- |
| osboxes@osboxes:~/Documents/Daemon$ sudo systemctl disable DisksInfoDaemon  Removed symlink /etc/systemd/system/multi-user.target.wants/DisksInfoDaemon.service. |

* **systemctl start** отменяет действия **systemctl** **stop**
* **systemctl** **enable** отменяет действие **systemctl** **disable**

Теперь, при запросе статуса демона, мы получаем следующую информацию:

|  |
| --- |
| osboxes@osboxes:~/Documents/Daemon$ sudo systemctl status DisksInfoDaemon● DisksInfoDaemon.service - Information about disks  Loaded: loaded (/etc/systemd/system/DisksInfoDaemon.service; disabled; vendor  Active: inactive (dead)  May 30 07:05:15 osboxes systemd[1]: Started Information about disks.  May 30 07:05:15 osboxes systemd[2980]: DisksInfoDaemon.service: Failed at step E  May 30 07:05:15 osboxes systemd[1]: DisksInfoDaemon.service: Main process exited  May 30 07:05:15 osboxes systemd[1]: DisksInfoDaemon.service: Unit entered failed  May 30 07:05:15 osboxes systemd[1]: DisksInfoDaemon.service: Failed with result  May 30 07:05:20 osboxes systemd[1]: DisksInfoDaemon.service: Service hold-off ti  May 30 07:05:20 osboxes systemd[1]: Stopped Information about disks.  May 30 07:05:20 osboxes systemd[1]: Started Information about disks.  May 30 07:07:22 osboxes systemd[1]: Stopping Information about disks...  May 30 07:07:22 osboxes systemd[1]: Stopped Information about disks.  lines 1-14/14 (END) |

# Источники

1. Андрей Боровский, Демоны  
   <http://citforum.ru/programming/unix/daemons/>
2. systemd.service — Service unit configuration  
   <https://www.freedesktop.org/software/systemd/man/systemd.service.html>