Санкт-Петербургский Политехнический университет

Институт Компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Лабораторная работа №1**

**Дисциплина:** Проектирование ОС и компонентов

**Тема:** Разработка компонентов прикладного уровня с заданной функциональностью, реализация в виде: приложения, утилиты, демона, сервиса, службы

Выполнил студент гр. 13541/4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шайтан

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Душутина

(подпись)

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Санкт-Петербург

2017

**Выполнение работы:**

**Windows**

**Создание службы в Windows**

SC CREATE создание службы

Описание команды SC CREATE

Команда SC CREATE создает запись службы в реестре и в базе данных диспетчера служб. А для запуска только что созданной службы из командной строки служит команда SC START.

Синтаксис и параметры команды SC CREATE

sc [имя\_сервера] create [имя\_службы] [type= {own|share|kernel|filesys|rec|adapt|interact type= {own|share}}] [start= {boot|system|auto|demand|disabled}] [error= {normal|severe|critical|ignore}] [binpath= имя\_двоичного\_пути] [group= группа\_порядка\_загрузки] [tag= {yes|no}] [depend= зависимости] [obj= {имя\_учетной\_записи|имя\_объекта}] [displayname= отображаемое\_имя] [password= пароль]

имя\_сервера - Задает имя удаленного сервера, на котором находится служба. В имени следует использовать формат UNC ("\\myserver"). Чтобы запустить SC локально, этот параметр следует пропустить.

имя\_службы - Указывает имя службы, возвращенное операцией getkeyname.

type= {own|share|kernel|filesys|rec|adapt|interact type= {own|share}} - Указывает тип службы. Тип по умолчанию type= own.

start= {boot|system|auto|demand|disabled} - Указывает тип запуска для службы. Тип запуска по умолчанию start= demand.

error= {normal|severe|critical|ignore} - Указывает серьезность ошибки, если служба не запускается при загрузке. Значение параметра по умолчанию error= normal.

binpath= имя\_двоичного\_пути - Указывает путь в двоичном файле службы. Значение по умолчанию для параметра binpath= не задано. Эту строку необходимо указать.

group= группа\_порядка\_загрузки - Указывает имя группы, членом которой является эта служба. Список групп сохраняется в реестре в подразделе HKLM\System\CurrentControlSet\Control\ServiceGroupOrder. Значение по умолчанию является пустым.

tag= {yes|no} - Указывает, следует ли получить код TagID из вызова CreateService. Теги используются только драйверами, запускающимися при загрузке или запуске системы.

depend= зависимости - Указывает имена служб и групп, которые должны быть запущены раньше данной службы. Имена разделяются косой чертой (/).

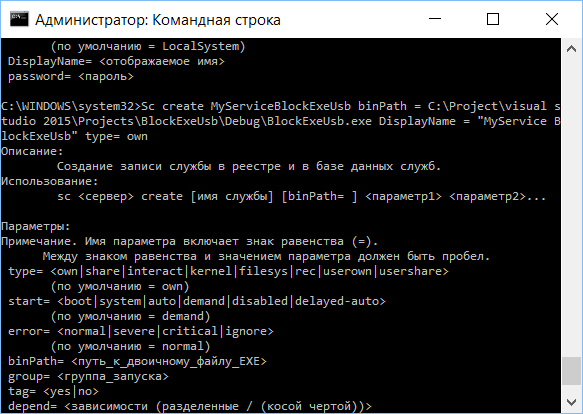
obj= {имя\_учетной\_записи|имя\_объекта} - Указывает имя учетной записи, для которой будет выполняться служба, или имя объекта драйвера Windows, в котором будет запущен драйвер

displayname= отображаемое\_имя - Определяет понятное, точное имя для службы, которое используется в программах пользовательского интерфейса.

password= пароль - Задает пароль. Данный параметр требуется при использовании учетной записи, отличной от учетной записи «Локальная система».

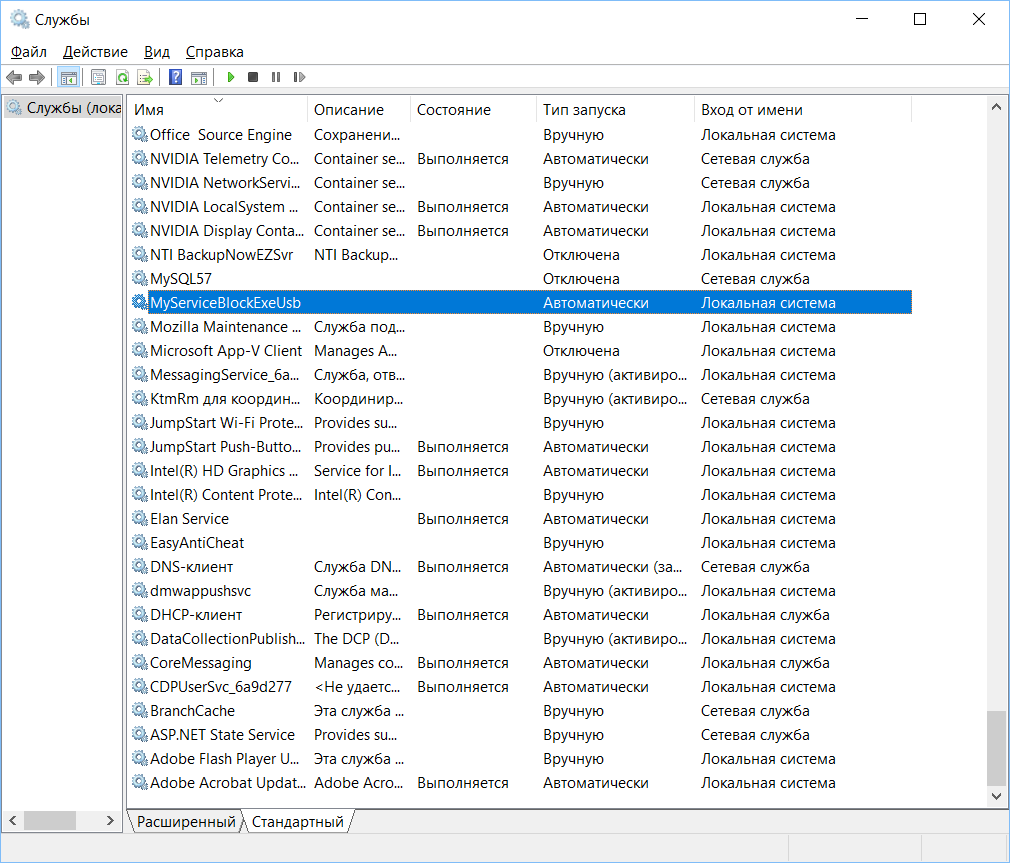
/? - Отображает справку в командной строке.

Есть несколько вариантов, один из них, при помощи утилита sc. Установка производиться следующей командой:



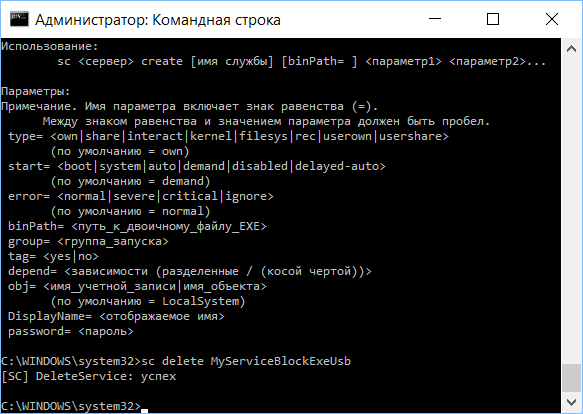
PS C:\WINDOWS\system32> New-Service -Name MyServiceBlockExeUsb -BinaryPathName "C:\Project\visual studio 2015\Projects\BlockExeUsb\Debug\BlockExeUsb.exe"

-DisplayName "MyService BlockExeUsb" -Description "Block Exe Files Service !!!"



Удаление сервиса:

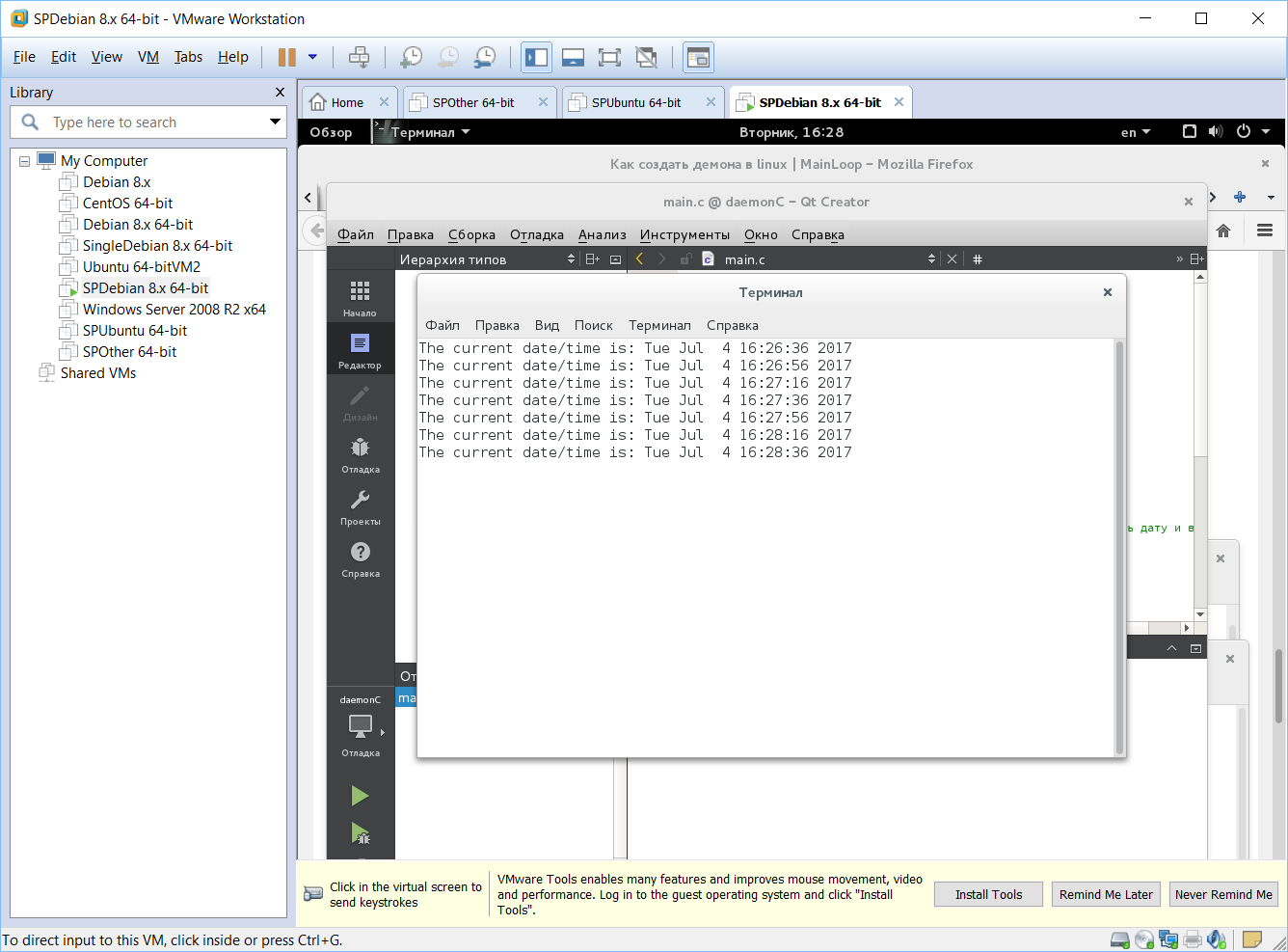
sc delete MyServiceBlockExeUsb



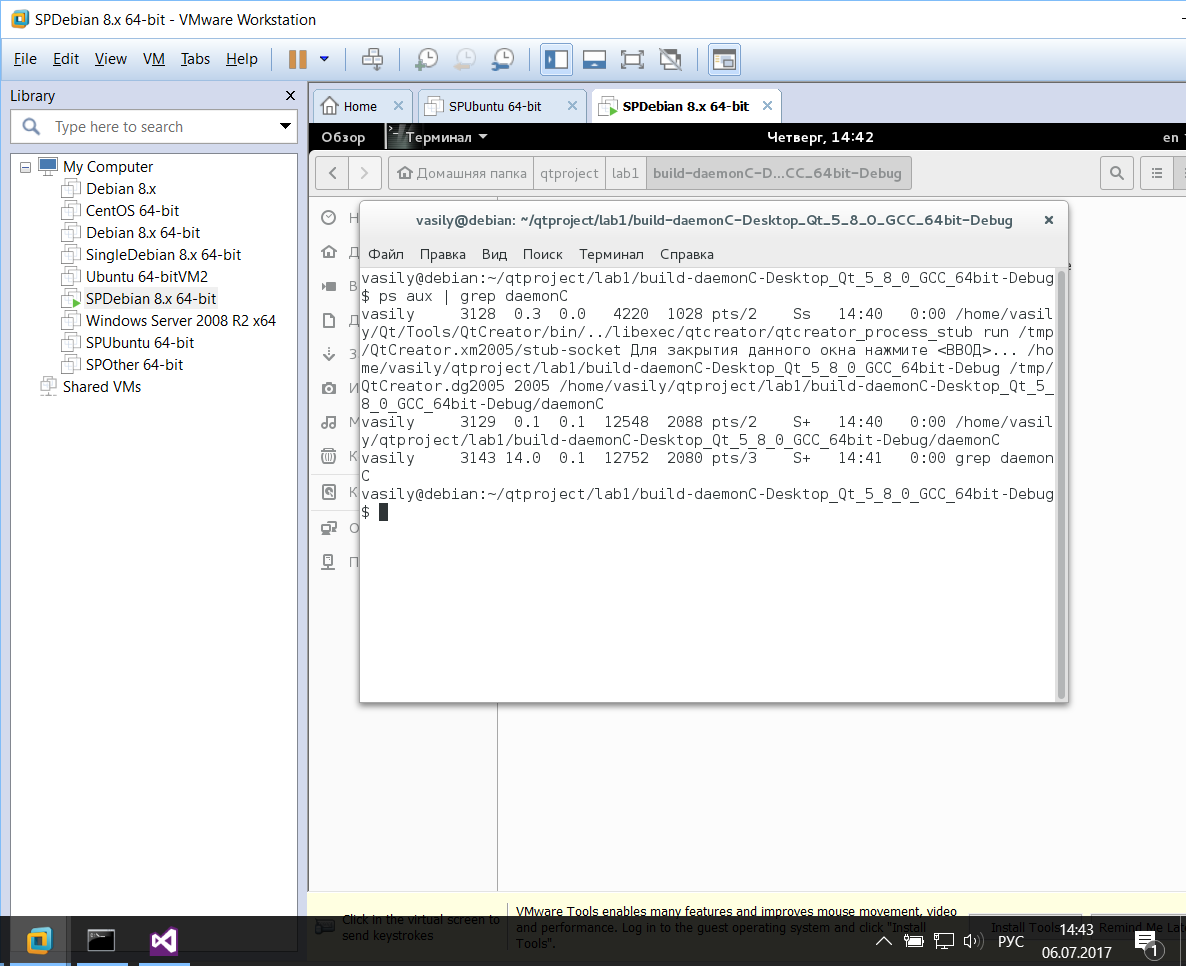
**Linux**

**Создание демона в Linux**

|  |
| --- |
| #include <stdio.h>  #include <time.h>  void daemon(void) {  time\_t rawtime;  while(1)  {  time(&rawtime);  printf("The current date/time is: %s",ctime(&rawtime));  sleep(20);  }  }  int main() {  setsid();  chdir("/");  fclose(stdin);  //оставим вывод в консоль наш демон будет периодически выводить дату и время  //fclose(stdout);  fclose(stderr);  daemon();  // int pid = fork();  // switch( pid ) {  // case 0:  // setsid();  // chdir("/");  // close(stdin);  // //close(stdout);  // close(stderr);  // demon();  // //mainloop();  // exit(0);  // case -1:  // printf("Fail: unable to fork\n");  // break;  // default:  // printf("OK: demon with pid %d is created\n", pid);  // break;  // }  //return 0;  } |



После сборки проекта, запускаем демон также, как и обычную консольную утилиту. И проверим, что демон запустился через ps aux



**Вывод**

В ходе данной лабораторной работы были изучены методы создания и запуска демона на UNIX подобных системах. Также, были изучены методы создания и установки служб в Windows.

Помимо изучения теоретической части, была произведена практическая реализация. То есть, был создан демон, который работает как консольная утилита. В системе Windows была создана служба.

**Список используемых источников:**

1. <http://cmd4win.ru/administrirovanie-computera/systemnoye-administrorovanie/25-sc-create> - SC CREATE создание службы
2. <http://windowsnotes.ru/cmd/sozdanie-sluzhby-v-windows/> - Создание службы в Windows
3. <http://mainloop.ru/c-language/kak-sozdat-demona-v-linux.html> - Как создать демона в linux