**Написать демон на Python, который будет уведомлять о новых коммитах в git репозитории.**

Для начала напишем класс для демона.

| *# daemon.py* import ???  class Daemon:  def \_\_init\_\_(self, pidfile): self.pidfile = pidfile   def daemonize(self):  """Deamonize class. UNIX double fork mechanism."""  try:  pid = os.fork()  if pid > 0:  *# exit first parent*  sys.exit(0)  except OSError as err:  sys.stderr.write('fork #1 failed: {0}\n'.format(err))  sys.exit(1)   *# decouple from parent environment*  os.chdir('/')  os.setsid()  os.umask(0)   *# do second fork*  try:  pid = os.fork()  if pid > 0:  *# exit from second parent*  sys.exit(0)  except OSError as err:  sys.stderr.write('fork #2 failed: {0}\n'.format(err))  sys.exit(1)   *# redirect standard file descriptors*  sys.stdout.flush()  sys.stderr.flush()  si = open(os.devnull, 'r')  so = open(os.devnull, 'a+')  se = open(os.devnull, 'a+')   os.dup2(si.fileno(), sys.stdin.fileno())  os.dup2(so.fileno(), sys.stdout.fileno())  os.dup2(se.fileno(), sys.stderr.fileno())   *# write pidfile*  atexit.register(self.delpid)   pid = str(os.getpid())  with open(self.pidfile,'w+') as f:  f.write(pid + '\n')  def delpid(self):  os.remove(self.pidfile)    def start(self):  *# your ode*   def stop(self):  *# your code*   def restart(self):  """Restart the daemon."""  self.stop()  self.start()   def run(self):  """You should override this method when you subclass Daemon.   It will be called after the process has been daemonized by  start() or restart().""" |
| --- |

С помощью **\_\_init\_\_(self, pidfile)** мы передаём демону имя файла, в котором будет записан его pid (process identifier).

**daemonize(self)** предназначен для создания клона процесса, запущенного пользователем. При создании клона необходимо внести в него ряд изменений:

1. Он должен быть независим от родительского процесса. Это осуществляется с помощью UNIX double fork mechanism.
2. Все потоки (stdin. stdout, stderr) перенаправляются в devnull.

**Задание.**

1. **start(self)**
   1. Выполнить проверку на наличие файла pidfile.

Если файл существует и содержит pid, вывести предупреждение о том, что демон уже запущен. Завершить скрипт.

* 1. Запустить **daemonize()**.
  2. Запустить **run()**.

1. **stop(self)**
   1. Получить pid из pidfile.

Если pid отсутствует, вывести предупреждение о том, что демон не запущен.

* 1. Убить демона с помощью **os.kill(pid, signal.SIGTERM)**.

1. **restart(self)** и **run(self)** изменять не требуется.
2. Создать файл **daemon\_example.py**.
3. На основе класса Daemon создать новый класс MyDaemon:

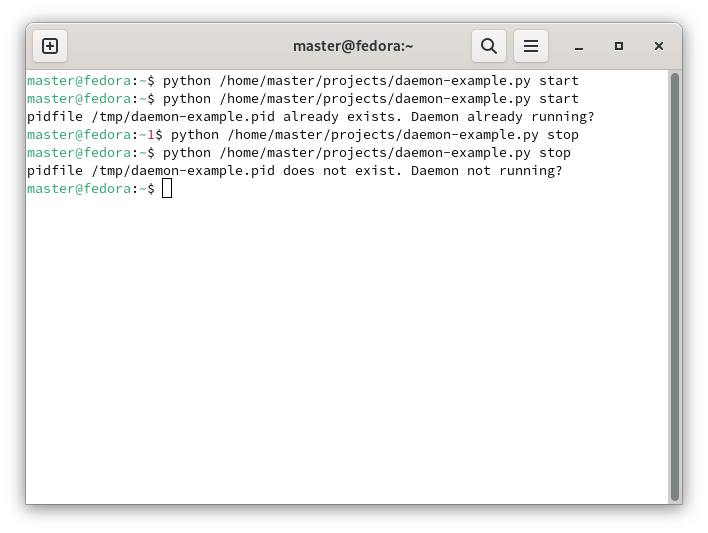
| class MyDaemon(Daemon):  def run(self):  while True:  time.sleep(1) |
| --- |

1. Функция run(self) будет переписана позже.
2. Создаём экземпляр демона:

| daemon = MyDaemon('/tmp/daemon-example.pid') |
| --- |

1. Проверяем аргументы, переданные при запуске скрипта:
   1. “start” - запуск демона.
   2. “stop” - остановка демона.
   3. “restart” - перезапуск демона.

После этого шага должны получить следующий результат:



1. Добавляем пробное уведомление:

| from gi.repository import Notify  Notify.init("MyDaemon") Notify.Notification.new("Test notification!").show() |
| --- |

1. Проверку на наличие новых коммитов реализовать самостоятельно, используя команды:

**subprocess.run()** - для запуска подпроцессов.

**git fetch** - для загрузки информации о новых коммитах в текущей ветке.

**git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)%Creset' --abbrev-commit --date=relative master..origin/master** - для вывода информации о новых коммитах.

1. Соединить всё воедино. Сделать 2 папки с копиями одного репозитория. Выполнить push коммита из первой папки и проверить вывод уведомления относительно второй папки.