МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
Кафедра 43

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ
ст.преп _____ М.Д.Поляк

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

СИСТЕМА СЛЕЖЕНИЯ

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛА
СТУДЕНТКА ГР. 4331 _____ Т.А.Белозуб

Санкт-Петербург, 2016 г.

Содержание

1. Цель работы

Цель работы: реализовать демон для веб-камеры под ОС Linux

2. Задание

Реализовать демон для веб-камеры, который делает снимок с камеры каждые 30 секунд и сохраняет его на диске в одном из популярных сжатых графических форматов (JPEG, TIFF, и т.п.). Предусмотреть возможность настройки временного интервала между снимками и пути, по которому сохраняются файлы.

3. Сравнение с аналогами

Есть программа, написанная на си, с похожим функционалом для ОС Linux под названием Motion. Она позволяет каждые N секунд делать снимки, позволяет пользователю указывать путь к сохраняемому файлу. Также имеется возможность выбора формата имени файла, что отсутствует в разработанном демоне. Однако данная программа лишена возможности автозапуска при загрузке системы.

4. Техническая документация

4.1. Установка

Склонировать репозиторий с github при помощи команды:

```
git clone https://github.com/tantanieli/Daemon-for-Webcamera.git
```

Для работы демона необходим установленный python версии 3.5.2 и выше.

4.2. Использование

Разработанный демон поддерживает следующие команды: start, stop, status. Вызов команды осуществляется следующим образом: python main.py <команда>. Команда start запускает демона и создает PID-файл с ID процесса. Команда status сообщает о статусе демона(работает он или нет), а так же ID процесса, если демон запущен. Команда stop останавливает работу демона.

5. Выводы

В процессе выполнения данной курсовой работы мною были получены знания и навыки, необходимые для работы с вебкамерой в ОС семейства Linux, а так же знания и навыки в написании демонов.

6. Приложения

main.py:

```
from kurs_daemon import Daemon
import os
import sys
import subprocess
import time

TAKE_PHOTO_CMD = "avconv -f video4linux2 -i /dev/video0 -vframes 1 {0}/{1}.jpeg"
INTERVAL = 30
PATH = "/tmp"
PIDFILE = "/tmp/kursach.pid"

class MyDaemon(Daemon):
def run(self):
while True:
subprocess.Popen(TAKE_PHOTO_CMD.format(PATH, str(time.time())).split(),
stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
time.sleep(INTERVAL)


if __name__ == "__main__":
if len(sys.argv) != 2 or sys.argv[1] not in ["start", "stop", "status"]:
print("Usage: python main.py start|stop|status")
sys.exit(1)
kursach = MyDaemon(PIDFILE)
if sys.argv[1] == "start":
kursach.start()
elif sys.argv[1] == "stop":
kursach.stop()
elif sys.argv[1] == "status":
if os.path.isfile(PIDFILE):
pidfile = open(PIDFILE, "r")
pid = pidfile.read()
print("Daemon is running on PID {0}".format(pid))
else:
print("Daemon is not running yet")
```

kurs daemon.py

```
import sys, os, time, atexit
from signal import SIGTERM

class Daemon:
"""
A generic daemon class.
Usage: subclass the Daemon class and override the run() method
"""
def __init__(self, pidfile, stdin='/dev/null',
 stdout='/dev/null', stderr='/dev/null'):
```

```
self.stdin = stdin
self.stdout = stdout
self.stderr = stderr
self.pidfile = pidfile

def daemonize(self):
"""
do the UNIX double-fork magic, see Stevens' "Advanced
Programming in the UNIX Environment" for details (ISBN 0201563177)
http://www.erlenstar.demon.co.uk/unix/faq_2.html#SEC16
"""
try:
pid = os.fork()
if pid > 0:
# exit first parent
sys.exit(0)
except OSError as e:
sys.stderr.write("fork #1 failed: %d (%s)\n" % (e.errno, e.strerror))
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 e:
sys.stderr.write("fork #2 failed: %d (%s)\n" % (e.errno, e.strerror))
sys.exit(1)

# redirect standard file descriptors
sys.stdout.flush()
sys.stderr.flush()
si = open(self.stdin, 'r')
so = open(self.stdout, 'a+')
# se = open(self.stderr, 'a+', 0)
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())
open(self.pidfile,'w+').write("%s\n" % pid)

def delpid(self):
os.remove(self.pidfile)
```

```
def start(self):
"""
Start the daemon
"""
# Check for a pidfile to see if the daemon already runs
try:
pf = open(self.pidfile,'r')
pid = int(pf.read().strip())
pf.close()
except IOError:
pid = None

if pid:
message = "pidfile %s already exist. Daemon already running?\n"
sys.stderr.write(message % self.pidfile)
sys.exit(1)

# Start the daemon
self.daemonize()
self.run()

def stop(self):
"""
Stop the daemon
"""
# Get the pid from the pidfile
try:
pf = open(self.pidfile,'r')
pid = int(pf.read().strip())
pf.close()
except IOError:
pid = None

if not pid:
message = "pidfile %s does not exist. Daemon not running?\n"
sys.stderr.write(message % self.pidfile)
return # not an error in a restart

# Try killing the daemon process
try:
while 1:
os.kill(pid, SIGTERM)
time.sleep(0.1)
except OSError as err:
err = str(err)
if err.find("No such process") > 0:
if os.path.exists(self.pidfile):
os.remove(self.pidfile)
else:
print(str(err))
sys.exit(1)
```

```
def restart(self):
"""
Restart the daemon
"""
self.stop()
self.start()

def run(self):
"""

"""
```