ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)"

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Курсовой проект

по курсу «Операционные системы»

Группа: М8о-207Б-18

Студент:

Токарев Никита Станиславович

Преподаватель:

Миронов Евгений Сергеевич

Оценка:

Дата:

Москва,2019

Оглавление

1.Постановка задачи

Нужна полноценная библиотека, которую можно было бы присоединять к любому проекту и далее достаточно было бы обращаться к серверу сообщений указав только его адрес

- Нужно несколько основных функций для библиотеки:

1. Отправка сообщений (Send(«MyQueue»,message, priority:3, expirationTime:30sec)
2. Создание очереди по имени (Create(«MyQueue», persistance:false))
3. Ожидание приёма сообщения (Recieve<MyType>())
4. Удаление очереди

- Дополнительные функции у самой очереди:

1. У каждого сообщения должен быть приоритет
2. У каждого сообщения должно быть время жизни
3. Очередь должна создаваться «персистентной» или нет. Если остановить сервер очередей сообещений, то персистентные очереди должны сохранить свои сообщения после повторного запуска

2.Структура программы

В данной работе в моя программа состоит из трех файлов:

1. server.py

2. queuemodule.py

3. client.py

3.Описание программы

В данной работе была подключен модуль с персистентной очередью: **persistqueue.** Также работая с данной очередью, использовалась база данных Sqlite. Связь между сервером и клиентом осуществляется через сокеты, также хотелось бы отметить, что используется библиотека через которую

принимаются и отправляются данные. Отправка сообщений, прием и все необходимые функции реализованы в нашем подключаемом модуле. При запуске client.py нужно ввести имя, и после данного действия с помощью библиотеки посылается сообщение на сервер о подключении клиента. Затем же у клиента появляются возможности отправки сообщения и остальных функций.

1) Отправка сообщения

Клиенту напечатав сообщение, предлагается указать: время жизни в секундах, приоритет(от 1 до 3), а также указать персистентное сообщение или нет. Далее же необходимые данные добавляются в очередь сообщений, а само сообщение после кодировки с помощью сокетов отправляется на сервер.

2) Прием сообщений

Вначале очередь проверяется на наличие сообщений с просроченным временем жизни и сортируются по приоритету. После данных манипуляций клиенту печатаются сообщения, находящиеся в очереди.

3) Удаление очереди

Все сообщения из очереди удаляются(get).

Время жизни сообщения: В очередь добавляется сообщение с параметром текущего времени, а также времени жизни. При обработке очереди проверяется сколько времени прошло после добавления и сравнивается с временем жизни.

Приоритет: Существуют три приоритета: **"Priority:3(low),2(middle),1(high): "**

При обработке очереди сообщения сортируется в зависимости от их приоритета.

Персистентные сообщения: В очереди хранится переменная, отвечающая за статус сообщения: персистное или нет. При выходе из сервера осуществляется проверка очереди на наличие неперсистентных сообщений.

4.Листинг программы

//client.py

```
import socket, threading, time, string,persistqueue, quemodule #библиотеки
shutdown = False
join = False
alias = input("Name: ")
while shutdown == False:
	if join == False:
		join = quemodule.Connect(alias)
	else:
		try:
			message = input()
			if message == "/recieve":
				quemodule.Recieve(alias) # считываем сообщение
			elif message == "/quit":
				quemodule.Unconnect(alias)
				shutdown = True
			elif message == "/clear":
				quemodule.DeleteAllQueue()
			else:
				extime = input("ExpirationTime(sec): ")
				priority = input("Priority:3(low),2(middle),1(high): ")
				pers = input("Pers Yes or No: ")
```

```
                quemodule.Send(alias,message,extime,priority,pers)
            time.sleep(0.2)
        except Exception as err:
            print(err)
            quemodule.Unconnect(name)
            shutdown = True
```

//server.py

```
import socket, time, persistqueue, os, shutil, collections, threading, math, quemodule
quit = False
host = socket.gethostbyname(socket.gethostname()) # в клиенте
print(host)
port = 9090
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((host,port))
print("[ Server Started ]")
while not quit:
    try:
        data, addr= s.recvfrom(1024)
        itsatime = time.strftime("%Y-%m-%d-%H.%M.%S", time.localtime()) # берем время для того чтобы на сервере отображалось время отправки сообщения
        # тут добавление в клиенты
        print("["+addr[0]+"]=["+str(addr[1])+"]=["+itsatime+"]/",end="") # принтим сообщение и время серверу
        print(data.decode("utf-8"))
    except KeyboardInterrupt:
        quemodule.check()
```

```
                quit = True
                print("\n[ Server Stopped ]")
s.close()
//queuemodule.py
import socket, time, persistqueue,  os, shutil, math, operator
path = os.path.join(os.path.abspath(os.path.dirname(__file__)), 'myqueue')
q = persistqueue.SQLiteQueue(path, auto_commit=True)
# connect
host = ""
server = (host,9090)
host = socket.gethostbyname(socket.gethostname()) # это наш айпи
port = 0
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((host,port)) # передаем ему айпи
s.setblocking(0) # не уверен, что это нужно, но она предотвращает возможные ошибки
# connect
def SortThird(val):
        return val[2]
def check ():
        global q
        q = persistqueue.SQLiteQueue(path, auto_commit=True)
        i = q.qsize()
        while (i > 0):
                dat, dr, pr, vrem, exptime, pers = q.get()
                if pers == "Yes":
```

```
                q.put((dat,dr,pr,vrem,exptime,pers))
        i = i - 1
def delete (q): # время жизни сообщений(высчитываю не совсем корректно но это бетка)
    try:
        i = q.qsize()
        size = i
        while (i > 0):
            clock = time.strftime("%d%M%S", time.localtime()) # время сейчас
            clock = int(clock)
            dat, dr, pr, vrem, exptime, pers = q.get()
            a = int(math.fabs(int(clock) - int(vrem))) # разница времени сообщения и время сейчас
            if (a < int(exptime)): # если больше n секунд, сам тут в ифе выбираешь, то сообщение удаляется из персистентноц очереди
                q.put((dat, dr, pr, vrem, exptime,pers))
            i = i - 1
        return 0
    except:
        return 0
def sort (q):
    try:
        i = q.size
        size = i
        nlist = []
        while(i > 0):
            dat, dr, pr, vrem, exptime, pers = q.get()
```

```
                nlist.append((dat, dr, pr, vrem, exptime,pers))
                i = i - 1
            nlist.sort(key = SortThird)
            nlist.reverse()
            while(size > 0):
                dat, dr, pr, vrem, exptime, pers = nlist.pop()
                q.put((dat, dr, pr, vrem, exptime, pers))
                size = size - 1
            return 0
        except:
            return 0
def Connect(name):
    message = '/connect:' + name
    s.sendto((message).encode("utf-8"),server)
    return True
def Unconnect(name):
    message = '/unconnect:' + name
    s.sendto((message).encode("utf-8"),server)
# main functions
def Send(name, message, extime, priority, pers):
    global q
    clock = time.strftime("%d%M%S", time.localtime()) # для времени жизни сообщений
    q.put((message, name, priority, clock, extime, pers))
    message = '/send:' + str(message) + ',name:' + str(name)
    s.sendto((message).encode("utf-8"),server)
```

```
def Recieve(name):
    global q
    q = persistqueue.SQLiteQueue(path, auto_commit=True)
    try:
        sort(q)
        delete(q)
        i = q.qsize()
        while(i > 0):
            dat, dr, pr, vrem, exptime, pers = q.get()
            if(name != dr):
                forprint = str(dat) + ' ,/name:' + dr + ' ,/priority:' + pr + ' ,/time:' +
exptime
                print(forprint)
                q.put((dat,dr,pr,vrem,exptime,pers))
            elif (name == dr):
                q.put((dat,dr,pr,vrem,exptime,pers))
            i = i - 1
    except Exception as err:
        print(err)
        print("Error in recieve")
def DeleteQueue():
    global q
    i = q.qsize()
    while (i > 0): # очищаем нашу очередь
        dat, dr, pr, vrem, exptime, pers = q.get()
        if(name != dr):
```

```
                q.put((dat,dr,pr,vrem,exptime,pers))
            i = i - 1
def DeleteAllQueue():
    global q
    DeleteQueue()
# main functions
```

5.Результат работы

///client.py

nikita@nikita-HP:~/oc/kp/KPos$ python3 client.py

Name: def

gdkjgdljg

ExpirationTime(sec): 20

Priority:3(low),2(middle),1(high): 1

Pers Yes or No: Yes

fdghgkdhfgkhdgkhd

ExpirationTime(sec): 100

Priority:3(low),2(middle),1(high): 2

Pers Yes or No: No

ghgdfhgjdghdjgh

ExpirationTime(sec): 1000

Priority:3(low),2(middle),1(high): 1

Pers Yes or No: No

fksdfkshgfskhgdfkhgd

ExpirationTime(sec): 1000

Priority:3(low),2(middle),1(high): 1

Pers Yes or No: No

//client.py

nikita@nikita-HP:~/oc/kp/KPos$ python3 client.py

Name: reddd

/recieve

fdghgkdhfgkhdgkhd ,/name:def ,/priority:2 ,/time:100

/recieve

ghgdfhgjdghdjgh ,/name:def ,/priority:1 ,/time:1000

fksdfkshgfskhgdfkhgd ,/name:def ,/priority:1 ,/time:1000

fdghgkdhfgkhdgkhd ,/name:def ,/priority:2 ,/time:100

/recieve

// Данное действие осуществлялось после закрытия сервера и все неперсистентные сообщения были удалены.

//server.py

nikita@nikita-HP:~/oc/kp/KPos$ python3 server.py

127.0.1.1

[Server Started]

[127.0.1.1]=[54775]=[2020-03-10-17.20.03]//connect:def

[127.0.1.1]=[54775]=[2020-03-10-17.20.10]//send:gdkjgdljg,name:def

[127.0.1.1]=[54775]=[2020-03-10-17.20.19]//send:fdghgkdhfgkhdgkhd,name:def

[127.0.1.1]=[56035]=[2020-03-10-17.20.37]//connect:reddd

[127.0.1.1]=[54775]=[2020-03-10-17.21.04]//send:ghgdfhgjdghdjgh,name:def

[127.0.1.1]=[54775]=[2020-03-10-17.21.12]//send:fksdfkshgfskhgdfkhgd,name:def

^C

[Server Stopped]

nikita@nikita-HP:~/oc/kp/KPos$ python3 server.py

127.0.1.1

[Server Started]

6.Вывод

При выполнении курсового проекта были изучены основы языка программирования Python. В ходе работы также познакомились с такими технологиями: сокеты, очередь сообщений, персистентные очереди и модули на языке Python. Для каждого сообщения реализован приоритет, время жизни и персистентность. Был получен ценный опыт работы в команде, который наверняка пригодится в будущем. Из минусов нашей работы хотелось бы отметить то, что была подключена персистентная очередь из открытого доступа, однако в ходе работы были изучены все свойства персистентной очереди и получены необходимые знания о ней.