



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Приложение мониторинга сетевой активности

Студент ИУ7-74Б
(Группа)

Власова Е.В.
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта

Рогозин Н.О.
(Подпись, дата) (И.О.Фамилия)

Консультант

(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение.....	3
1. Аналитический раздел	4
1.1 Формализация задачи.....	4
1.2 Сокеты.....	4
1.3 Протоколы TCP и UDP	7
1.4 Ipv 4 и Ipv6	8
1.5 Сравнение с аналогами	9
1.5.1 Nagios.....	9
1.5.2 Observium	9
1.5.3 PRTG Network Monitor	10
1.6 Вывод.....	11
2. Конструкторский раздел.....	12
2.1 Проектирование Use-Case диаграммы	12
2.2 Функциональная модель.....	12
2.3 Клиентская часть.....	13
2.4 Вывод.....	13
3. Технологический раздел.....	14
3.1 Выбор инструментов разработки	14
3.1.1 Выбор языка программирования	14
3.1.2 Выбор библиотек.....	14
3.2 Реализация доступа к данным.....	15
3.3 Интерфейс пользователя	19
3.4 Вывод.....	21
Заключение	22
Список использованной литературы.....	23

Введение

Работа и досуг всё больше уходят в «онлайн». Ноутбуки и компьютеры являются самыми удобными и популярными средствами для выполнения рабочих задач и отдыха в свободное время. В связи с этим, людям требуется устойчивое интернет-соединение, а также возможность быстро находить причину нарушения его производительности.

Внезапные изменения в скорости интернет-соединения могут быть результатом влияния нескольких различных факторов. Сюда входят такие факторы, как неисправности на стороне интернет-провайдера, неправильная работа модема, роутера и другого сетевого оборудования, а также повышенная сетевая активность программ.

Для людей, пропускная способность интернета которых не так велика, проверка отслеживание работы интернета является привычной рутиной. Тем не менее, такие действия не менее важны и для пользователей, использующих безлимитное подключение.

Цель данного курсового проекта – создание приложения для мониторинга сетевой активности.

Для достижения цели курсового проекта необходимо решить следующие задачи:

- Формализовать задачу, задав необходимый функционал;
- Выбрать технологический стек;
- Реализовать логику работы приложения;
- Реализовать десктопное приложение.

1. Аналитический раздел

1.1 Формализация задачи

Разработать приложение, позволяющее пользователю просматривать список активных подключений. Запуск программы производится с ПК с установленной системой Ubuntu. Пользователь также должен иметь возможность просматривать информацию о процессах, активных пользователях, памяти ПК.

1.2 Сокеты

Сокет – это абстракция конечной точки взаимодействия. Абстракция сокетов была введена в 4.2 BSD (Berkley Software Distribution) UNIX и были созданы для организации взаимодействия процессов, причем безразлично, где эти процессы выполняются: на одной машине или на нескольких машинах. [1] Другими словами, сокеты являются универсальным средством межпроцессного взаимодействия в том смысле, что они могут использоваться как для взаимодействия процессов на отдельно стоящей машине, так и для взаимодействия процессов в сети (рисунок 1).

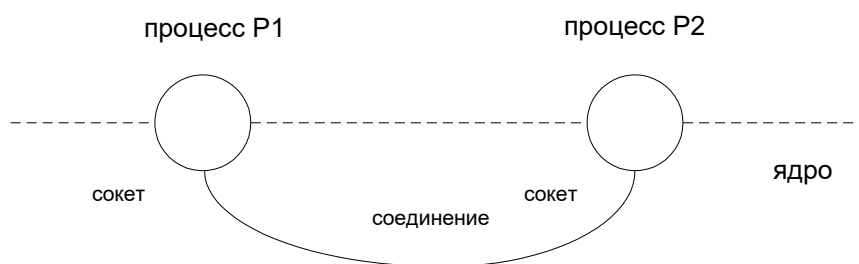


Рисунок 1. Взаимодействие сокета и процесса

Приложение может использовать единообразный интерфейс сокетов для отправки и получения сообщений как на локальной машине, так и по сети, используя разные протоколы.

Протоколы для взаимодействия с использованием сокетов выбираются на основе трех параметров:

- семейство или домен (family);
- тип сокета (type);
- протокол (protocol)

socket() создает конечную точку для связи и возвращает файловый дескриптор, который относится к этой конечной точке. Дескриптор файла при успешном вызове возвращается файловый дескриптор с наименьшим номером дескриптора файла, который в данный момент еще не открыт для процесса. [2]

Параметр domain указывает домен связи. Домен определяет семейство протоколов, которое будет использоваться для связи. Эти семейства определены в <sys / socket.h>. Ядро Linux в настоящее время включает следующие форматы:

№	Название	Назначение
1	AF_UNIX	Локальная коммуникация
2	AF_LOCAL	То же, что и AF_UNIX
3	AF_AX25	IPv4 интернет протокол
4	AF_IPX IPX	AX.25 протокол
5	AF_APPLETALK	AppleTalk
6	AF_X25	ITU-T X.25 / ISO-8208 протокол
7	AF_INET6	IPv6 интернет протокол
8	AF_DECnet	DECet protocol sockets
9	AF_KEY	Ключевой протокол управления, изначально создавался для работы с IPsec
10	AF_NETLINK	Интерфейс ядра

11	AF_PACKET	Интерфейс нижнего уровня
12	AF_RDS	Надежные сокет для датаграмм (RDS) протокол
13	AF_PPPOX	Общий транспортный уровень PPP, для настройки вверх по тоннелям L2 (L2TP и PPPoE)
14	AF_LLC	Управление логической связью AF_LLC (IEEE 802.2 LLC) протокол

AF – это сокращение Address Family.

Наиболее часто используются:

AF_UNIX

AF_INET для сетевого протокола IPv4

PF_INET6 для IPv6.

PF_UNIX для локальных сокетов (используя файл).

Тип (type) определяет семантику соединения:

SOCK_STREAM - обеспечивает последовательное, надежное, двустороннее соединение (надёжная потокоориентированная служба (TCP) (сервис) или потоковый сокет)

SOCK_DGRAM - поддерживает датаграммы (UDP) или датаграммные сокет (без установления соединения, ненадежная передача сообщений; сообщения фиксированной максимальной длины).

`SOCK_SEQPACKET` - обеспечивает последовательное, надежное двустороннее соединение для дейтаграмм фиксированной длины (максимальной); потребитель должен прочитать весь пакет с каждым входным системным вызовом.

1.3 Протоколы TCP и UDP

Существуют два механизма, предназначенных для сетевого взаимодействия программ, - это сокет датаграмм, которые используют пользовательский датаграммный протокол (User Datagram Protocol) (UDP) без установления соединения, и сокет, использующие Протокол управления передачей / Межсетевой протокол (Transmission Control Protocol/Internet Protocol) (TCP/IP), устанавливающий соединение. [3]

Датаграмма - пакет данных, отправленный по сети, прибытие которого, время прибытия и содержание не гарантировано. Не гарантируется также и порядок доставки пакетов. При передаче пакета UDP по какому-либо адресу нет никакой гарантии того, что он будет принят, а также, что по этому адресу вообще существует потребитель пакетов. Аналогично, когда приходит датаграмма, у нет никаких гарантий, что она не была повреждена в пути следования или что отправитель ожидает подтверждения получения датаграммы. Использование UDP может привести к потере или к дублированию пакетов, что приводит к дополнительным проблемам, связанным с проверкой ошибок и обеспечением надежности передачи данных. Если необходимо добиться оптимальной производительности, и возможно сократить затраты на проверку целостности информации, пакеты UDP могут оказаться весьма полезными.

TCP/IP является потоковым протоколом на основе установления двунаправленных соединений точка-точка между узлами Интернет, и взаимодействие между компьютерами по этому протоколу предназначено для реализации надежной передачи данных. Все данные, отправленные по каналу передачи, получаются в том же порядке, в котором они передавались. В отличие

от датаграммных сокетов, сокет TCP/IP реализует высоконадежные устойчивые соединения между клиентом и сервером.

1.4 IPv4 и IPv6

В протоколе IP используются IP-адреса, которые идентифицируют рабочие станции, сервера и маршрутизаторы. IP-адрес должен быть уникальным. Соответствие между именем и IP-адресом узла сети осуществляется с помощью специальных баз данных, хранящих пару имя-адрес. IP-адрес представлен четырьмя октетами. Например, 191.200.182.101. IP протокол — это протокол сетевого уровня, обеспечивающий маршрутизацию данных в Интернете. IPv4 является 4-й версией IP. IPv4 — это протокол без установления соединения, который реализуется в сетях с использованием коммутации пакетов. Он работает на основе лучшей модели доставки данных, что означает, что они получают неопределенную переменную пропускную способность и время доставки, в зависимости от текущей нагрузки трафика. Это не гарантирует саму доставку, не обеспечивает адекватной последовательности и не позволяет избежать повторной доставки. IPv6 является последней версией IP и представляет собой постепенное обновление протокола IPv4. По сути, IPv6 обеспечивает полную передачу данных по нескольким IP-сетям, придерживаясь принципов проектирования, разработанных в предыдущей версии протокола.

IPv4 использует только 32 бита для своих интернет-адресов. В основном это означает, что IPv4 может обрабатывать до 2^{32} IP-адресов, что составляет 4 294 967 296 (4,29 миллиарда). Хотя это число кажется большим, оценочное число устройств, подключенных к интернету, превышает 20 миллиардов, и это число растет день ото дня. Следовательно, IP-адрес любого устройства должен быть конкретным и уникальным, и по мере роста числа пользователей адреса IPv4 заканчиваются. IPv6 использует 128-битные интернет-адреса. Это означает, что протокол может обрабатывать в общей сложности до 2^{128} IP-адресов, которые будут приблизительно составлять 340 282 366 920 938 000 000 000 000 000 000 000.

1.5 Сравнение с аналогами

1.5.1 Nagios

Nagios — это состоявшаяся программная система для мониторинга сети, которая уже многие годы находится в активной разработке. Написанная на языке С, она позволяет делать почти все, что может понадобится системным и сетевым администраторам от пакета прикладных программ для мониторинга. Веб-интерфейс этой программы является быстрым и интуитивно понятным, в то время его серверная часть — чрезвычайно надежной. Кроме того, функция отображения демонстрирует все контролируемые устройства в логическом представлении их размещения в сети, с цветовым кодированием, что позволяет показать проблемы по мере их возникновения.

Недостатком Nagios является конфигурация, так как ее лучше всего выполнять через командную строку, что значительно усложняет обучение новичков.

Плюсы:

- Большой функционал;
- Понятный интерфейс;
- Система уведомлений.

Недостатки:

- Обращение к программе через командную строку, что не делает её интуитивно понятной для новичков.

1.5.2 Observium

Этот инструмент сетевого мониторинга отличает способность к анализу производительности системы. Протокол SNMP, на котором основана работа системы, помогает с успехом анализировать любые сети, вне зависимости от их масштаба и архитектуры. Работает программа с оборудованием большинства

производителей и всеми основными операционными системами. Используется графический интерфейс, отличающийся точностью и богатством опций.

Плюсы:

Есть демоверсия;

Работает во многих ОС.

Недостатки:

Сложная установка;

Ограниченность бесплатной версии.

1.5.3 PRTG Network Monitor

Этот инструмент предназначен для мониторинга локальных сетей любого размера. Он работает только с Windows. Продукт распространяется по платной лицензии, сэкономить можно только на первом месяце использования с целью знакомства с возможностями программы.

Инструмент хорошо справится со всеми задачами мониторинга, просканирует техническое состояние всех подключенных к сети устройств, выявит кибератаки.

Среди функций, которые заинтересуют администратора корпоративной сети:

- проверка трафика;
- возможность сохранения данных статистики мониторинга сети в базу для последующего анализа;

Плюсы:

- Возможность вести мониторинг в режиме реального времени;
- Понятный интерфейс.

Недостатки:

- Высокая стоимость.

1.6 Вывод

В данном разделе была дана формализация задачи на курсовой проект. Рассмотрено понятие сокета. Рассмотрены встречающиеся протоколы. Проанализированы существующие аналоги разрабатываемого приложения, чьи основные недостатки – ограниченность или отсутствие бесплатной версии, а также сложный для нового пользователя интерфейс. В данной курсовой работе будет создано приложение, не имеющее данных недостатков.

2. Конструкторский раздел

2.1 Проектирование Use-Case диаграммы

На рисунке 2 представлена Use-Case диаграмма проекта. С системой может взаимодействовать один актер: пользователь. У него есть права на выполнение всех действий, предоставленных системой.

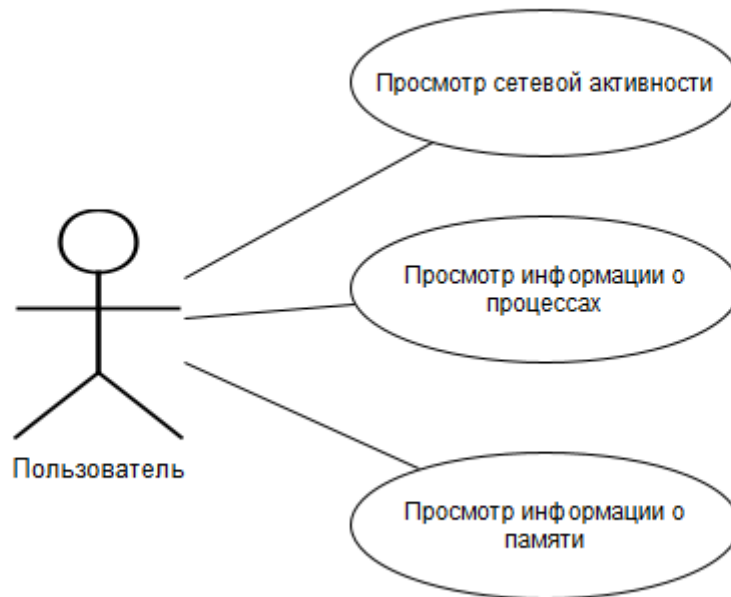


Рисунок 2. Use-case диаграмма

2.2 Функциональная модель

На рисунке 3 представлена функциональная модель приложения IDEF0 нулевого уровня.

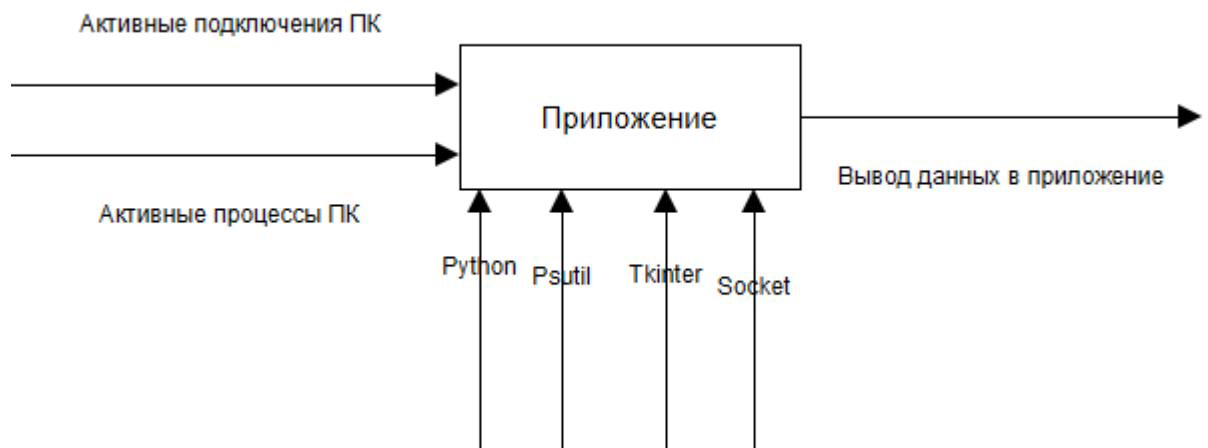


Рисунок 3. IDEF0

2.3 Клиентская часть

В данной работе необходимо разработать клиентскую часть, то есть, как пользователю взаимодействует с программой, чтобы получить результат. Пользователь выбирает, какие именно данные отображаются в окне приложения.

Существует множество возможностей реализации клиентской части. Для удобства использования программы было принято решение создать графический интерфейс, так как взаимодействие с консолью не всегда является удобным в эксплуатации и требует запоминания различных команд.

2.4 Вывод

В данном разделе представлена Use-Case диаграмма приложения и функциональная модель IDEF0.

3.Технологический раздел

3.1 Выбор инструментов разработки

Для реализации поставленных задач необходимо выбрать язык программирования и библиотеки для работы с сетевыми данными.

3.1.1 Выбор языка программирования

Самыми распространёнными языками программирования для при сетевого администрирования являются:

- C#;
- Python.

В данной работе используется Python, являющийся динамическим языком программирования. Его преимуществами являются кроссплатформенность, отсутствие этапа компиляции (за счёт чего уменьшается время разработки) а также лаконичность и легко читаемый код.

3.1.2 Выбор библиотек

Для реализации поставленных задач была выбрана библиотека psutil. Библиотека psutil предназначена для получения информации о запущенных процессах и использовании системы (процессор, память, диски, сеть). Предоставляет интерфейс для получения информации о запущенных процессах и использовании системы (CPU, память) портативным способом с помощью Python, реализуя множество функций, предлагаемых такими инструментами, как ps, top и Windows Task manager. В настоящее время он поддерживает Linux, Windows, OSX, Sun Solaris, FreeBSD, OpenBSD и NetBSD; как 32-разрядные, так и 64-разрядные архитектуры.

Для реализации интерфейса выбрана библиотека Tkinter. Tkinter расшифровывается как Tk interface. В Python есть довольно много GUI фреймворков (graphical user interface), однако только Tkinter встроен в стандартную библиотеку языка. У Tkinter есть несколько преимуществ. Он кроссплатформенный, поэтому один и тот же код можно использовать на

Windows, macOS и Linux. Визуальные элементы отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью Tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают.

Хотя Tkinter является популярным GUI фреймворком на Python, у него есть свои недостатки. Один из них заключается в том, что графические интерфейсы, созданные с использованием Tkinter, выглядят устаревшими. Однако это хороший выбор для задач, если большую роль играет функциональность и кроссплатформенная скорость.

3.2 Реализация доступа к данным

В результате выполнения данной курсовой работы реализованы функции, позволяющие мониторить сетевую активность, запущенные процессы и память ПК. Данные функции представлены в листингах.

```
AD = "-"
AF_INET6 = getattr(socket, 'AF_INET6', object())
proto_map = {
    (AF_INET, SOCK_STREAM): 'tcp',
    (AF_INET6, SOCK_STREAM): 'tcp6',
    (AF_INET, SOCK_DGRAM): 'udp',
    (AF_INET6, SOCK_DGRAM): 'udp6',
}
templ = "%-5s %-30s %-30s %-13s %-6s %s"
print(templ % (
    "Proto", "Local address", "Remote address", "Status", "PID",
    "Program name"))
proc_names = {}
for p in psutil.process_iter(['pid', 'name']):
    proc_names[p.info['pid']] = p.info['name']
```

```

for c in psutil.net_connections(kind='inet'):
    laddr = "%s:%s" % (c.laddr)
    raddr = ""
    if c.raddr:
        raddr = "%s:%s" % (c.raddr)
    name = proc_names.get(c.pid, '?') or "

```

Листинг 1. Вывод активных соединений

```

def show_users():
    users = psutil.users()
    for user in users:
        proc_name = psutil.Process(user.pid).name() if user.pid else ""
        print("%-12s %-10s %-10s %-14s %s" % (
            user.name,
            user.terminal or '-',
            datetime.fromtimestamp(user.started).strftime("%Y-%m-%d
%H:%M"),
            "(%s)" % user.host if user.host else "",
            proc_name
        ))

```

Листинг 2. Вывод активных пользователей

```

def show_netinterface():
    af_map = {
        socket.AF_INET: 'IPv4',
        socket.AF_INET6: 'IPv6',
        psutil.AF_LINK: 'MAC',
    }

    duplex_map = {

```

```

psutil.NIC_DUPLEX_FULL: "full",
psutil.NIC_DUPLEX_HALF: "half",
psutil.NIC_DUPLEX_UNKNOWN: "?",
}

stats = psutil.net_if_stats()
io_counters = psutil.net_io_counters(pernic=True)
for nic, addrs in psutil.net_if_addrs().items():
    print("%s:" % (nic))

    if nic in stats:
        st = stats[nic]
        print("    stats      : ", end="")
        print("speed=%sMB, duplex=%s, mtu=%s, up=%s" % (
            st.speed, duplex_map[st.duplex], st.mtu,
            "yes" if st.isup else "no"))
    if nic in io_counters:
        io = io_counters[nic]
        print("    incoming    : ", end="")
        print("bytes=%s, pkts=%s, errs=%s, drops=%s" % (
            bytes2human(io.bytes_recv), io.packets_recv, io.errin,
            io.dropin))
        print("    outgoing    : ", end="")
        print("bytes=%s, pkts=%s, errs=%s, drops=%s" % (
            bytes2human(io.bytes_sent), io.packets_sent, io.errout,
            io.dropout))
    for addr in addrs:
        print("    %-4s" % af_map.get(addr.family, addr.family), end="")
        print(" address   : %s" % addr.address)

```

```

if addr.broadcast:
    print("        broadcast : %s" % addr.broadcast)
if addr.netmask:
    print("        netmask  : %s" % addr.netmask)
if addr.ptp:
    print("        p2p      : %s" % addr.ptp)

```

Листинг 3. Вывод сетевых интерфейсов

```

def show_disks():
    templ = "%-17s %8s %8s %8s %5s%% %9s %s"
    print(templ % ("Device", "Total", "Used", "Free", "Use ", "Type",
                  "Mount"))
    for part in psutil.disk_partitions(all=False):
        if os.name == 'nt':
            usage = psutil.disk_usage(part.mountpoint)
            print(templ % (
                part.device,
                bytes2human(usage.total),
                bytes2human(usage.used),
                bytes2human(usage.free),
                int(usage.percent),
                part.fstype,
                part.mountpoint))

```

Листинг 4. Вывод монтированных дисков

```

def pprint_ntuple(nt):
    for name in nt._fields:
        value = getattr(nt, name)
        if name != 'percent':
            value = bytes2human(value)

```

```
print('%-10s : %7s' % (name.capitalize(), value))
```

Листинг 5. Вывод информации о памяти ПК

3.3 Интерфейс пользователя

В данном разделе представлены примеры GUI и работы программы.

Сеть Процессы Память Пользователи Информация

Рисунок 1. Панель программы

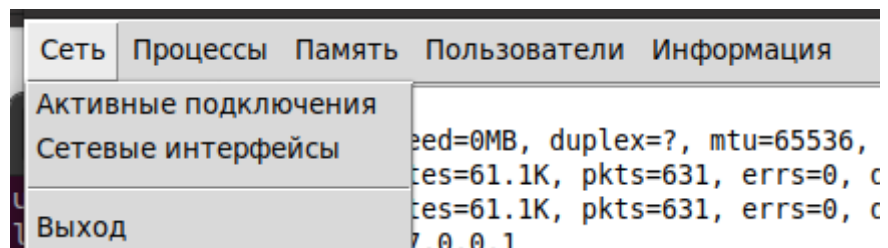


Рисунок 2. Выбор нужного действия

Сеть Процессы Память Пользователи Информация						
Proto	Local address	Remote address	Status	PID	Program name	
udp	0.0.0.0:54186	-	NONE	-	?	
udp	10.0.2.15:68	10.0.2.2:67	NONE	-	?	
tcp	127.0.0.53:53	-	LISTEN	-	?	
tcp6	:::1:631	-	LISTEN	-	?	
tcp	127.0.0.1:631	-	LISTEN	-	?	
udp	0.0.0.0:631	-	NONE	-	?	
udp	0.0.0.0:5353	-	NONE	-	?	
udp	127.0.0.53:53	-	NONE	-	?	
udp6	:::60980	-	NONE	-	?	
tcp	10.0.2.15:59844	54.244.7.161:443	ESTABLISHED	1535	MainThread	
udp6	:::5353	-	NONE	-	?	

Рисунок 3. Пример работы программы. Активные подключения

Сеть Процессы Память Пользователи Информация				
katya	:0	2020-12-27 20:31 (localhost)		gdm-x-session

Рисунок 4. Пример работы программы. Активные пользователи

Сеть	Процессы	Память	Пользователи	Информация					
USER	PID	%MEM	VSZ	RSS	NICE	STATUS	START	TIME	CMDLINE
root	1	0.7	163.6M	6.5M		sleep	20:31	00:25	/sbin/init splash
root	2	0.0	0.0B	0.0B		sleep	20:31	00:00	kthreadd
root	3	0.0	0.0B	0.0B	-20	idle	20:31	00:00	rcu_gp
root	4	0.0	0.0B	0.0B	-20	idle	20:31	00:00	rcu_par_gp
root	5	0.0	0.0B	0.0B		idle	20:31	00:00	kworker/0:0-events
root	6	0.0	0.0B	0.0B	-20	idle	20:31	00:00	kworker/0:0H-kblockd
root	9	0.0	0.0B	0.0B	-20	idle	20:31	00:00	mm_percpu_wq
root	10	0.0	0.0B	0.0B		sleep	20:31	00:00	ksoftirqd/0
root	11	0.0	0.0B	0.0B		idle	20:31	00:01	rcu_sched
root	12	0.0	0.0B	0.0B		sleep	20:31	00:00	migration/0
root	13	0.0	0.0B	0.0B		sleep	20:31	00:00	idle_inject/0
root	14	0.0	0.0B	0.0B		sleep	20:31	00:00	cpuhp/0
root	15	0.0	0.0B	0.0B		sleep	20:31	00:00	cpuhp/1
root	16	0.0	0.0B	0.0B		sleep	20:31	00:00	idle_inject/1
root	17	0.0	0.0B	0.0B		sleep	20:31	00:00	migration/1
root	18	0.0	0.0B	0.0B		sleep	20:31	00:00	ksoftirqd/1
root	20	0.0	0.0B	0.0B	-20	idle	20:31	00:00	kworker/1:0H-kblockd
root	21	0.0	0.0B	0.0B		sleep	20:31	00:00	kdevtmpfs
root	22	0.0	0.0B	0.0B	-20	idle	20:31	00:00	netns
root	23	0.0	0.0B	0.0B		sleep	20:31	00:00	rcu_tasks_kthre
root	24	0.0	0.0B	0.0B		sleep	20:31	00:00	kauditd
root	25	0.0	0.0B	0.0B		sleep	20:31	00:00	khungtaskd
root	26	0.0	0.0B	0.0B		sleep	20:31	00:00	oom_reaper
root	27	0.0	0.0B	0.0B	-20	idle	20:31	00:00	writeback
root	28	0.0	0.0B	0.0B		sleep	20:31	00:00	kcompactd0
root	29	0.0	0.0B	0.0B	5	sleep	20:31	00:00	ksmd
root	30	0.0	0.0B	0.0B	19	sleep	20:31	00:00	khugepaged
root	77	0.0	0.0B	0.0B	-20	idle	20:31	00:00	kintegrityd

Рисунок 5. Пример работы программы. Вывод процессов

Сеть	Процессы	Память	Пользователи	Информация		
Device	Total	Used	Free	Use %	Type	Mount
/dev/sda5	14.2G	6.5G	7.0G	48%	ext4	/
/dev/loop1	55.5M	55.5M	0.0B	100%	squashfs	/snap/core18/1944
/dev/loop0	55.0M	55.0M	0.0B	100%	squashfs	/snap/core18/1880
/dev/loop2	255.6M	255.6M	0.0B	100%	squashfs	/snap/gnome-3-34-1804/36
/dev/loop3	218.0M	218.0M	0.0B	100%	squashfs	/snap/gnome-3-34-1804/60
/dev/loop7	51.1M	51.1M	0.0B	100%	squashfs	/snap/snap-store/518
/dev/loop4	62.1M	62.1M	0.0B	100%	squashfs	/snap/gtk-common-themes/1506
/dev/loop5	64.9M	64.9M	0.0B	100%	squashfs	/snap/gtk-common-themes/1514
/dev/loop6	49.9M	49.9M	0.0B	100%	squashfs	/snap/snap-store/467
/dev/loop8	31.1M	31.1M	0.0B	100%	squashfs	/snap/snapd/10492
/dev/loop9	30.0M	30.0M	0.0B	100%	squashfs	/snap/snapd/8542
/dev/sda1	511.0M	4.0K	511.0M	0%	vfat	/boot/efi
/dev/sr0	58.2M	58.2M	0.0B	100%	iso9660	/media/katya/VBox_GAs_6.1.16

Рисунок 6. Пример работы программы. Монтированные диски

```
MEMORY
-----
Total      : 981.1M
Available  : 196.4M
Percent    : 80.0
Used       : 637.4M
Free       : 63.3M
Active     : 458.7M
Inactive   : 318.4M
Buffers    : 17.2M
Cached     : 263.2M
Shared     : 8.6M
Slab       : 88.8M
SWAP
-----
Total      : 687.5M
Used       : 20.3M
Free       : 667.2M
Percent    : 2.9
Sin        : 976.0K
Sout       : 20.0M
```

Рисунок 7. Пример работы программы. Информация о памяти

3.4 Вывод

В данном разделе приведены листинги реализованных функций. Показаны примеры реализованного интерфейса и его функций.

Заключение

В результате проведённой работы:

- Проведён анализ инструментов для данной задачи, в результате которого были выбраны язык программирования Python, библиотека psutil и Tkinter;
- Реализованы функции, обеспечивающие доступ к данным процессов, памяти и активных подключений ПК. Реализованы такие возможности, как просмотр активных подключений, просмотр процессов и информации о них, просмотр активных пользователей, просмотр монтированных дисков, просмотр состояния памяти;
- Реализован GUI приложения.

Дальнейшее развитие проекта:

- Мониторинг в режиме реального времени;
- Возможность сохранять данные в файл для их дальнейшего анализа.

Список использованной литературы

1. А. В. Боровский Сокеты. журнал Linux Format. Серия "Программирование для Linux", 2010 г.
2. W. Richard Stevens, Bill Fenner, Andrew M. Rudoff «Unix Network Programming Volume 1, Third edition: The sockets networking API» 2015 г.
3. А.А. Дубаков. Сетевое программирование, 2013 г.
4. Документация Tkinter [Электронный ресурс] URL: <https://python-scripts.com/tkinter> (Дата обращения: 20.10.2020)
5. Документация netstat [Электронный ресурс] URL: <https://putty.org.ru/articles/netstat-linux-examples.html> (Дата обращения: 20.10.2020)
6. Документация psutil [Электронный ресурс] URL: <https://psutil.readthedocs.io/en/latest/> (Дата обращения: 20.10.2020)
7. Сокеты в Python [Электронный ресурс] URL: <https://docs-python.ru/standart-library/modul-socket-setevoj-interfejs-python/> (Дата обращения: 20.12.2020)