1. Problem

Implement a program that will launch a specified process and periodically (with a provided time interval) collect the following data about it:

- CPU usage (percent).
- Memory consumption: Working Set and Private Bytes (for Windows systems) or Resident Set Size and Virtual Memory Size (for Linux systems).
- Number of open handles (for Windows systems) or file descriptors (for Linux systems).

Data collection should be performed all the time the process is running. Path to the executable file for the process and time interval between data collection iterations should be provided by user. Collected data should be stored on the disk. Format of stored data should support automated parsing to potentially allow, for example, drawing of charts.

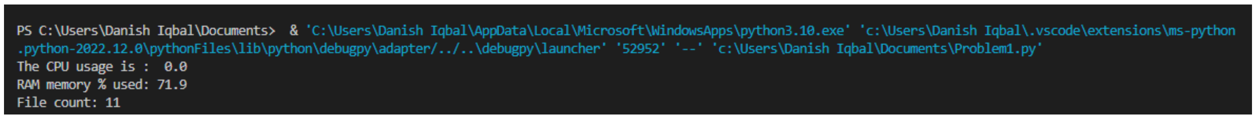
Solutions

1.1 Programming

```python
import os
import psutil

# Getting loadover10 minutes
load1, load5, load10 = psutil.getloadavg()

cpu_usage = (load10/os.cpu_count()) * 100

print("The CPU usage is : ", cpu_usage)

# Getting % usage of virtual_memory ( 3rd field)
print('RAM memory % used:', psutil.virtual_memory()[2])

# folder path
dir_path = r'c:\Users\Danish Iqbal'
count = 0
# Iterate directory
for path in os.listdir(dir_path):
    # check if current path is a file
    if os.path.isfile(os.path.join(dir_path, path)):
        count += 1
print('File count:', count)
```

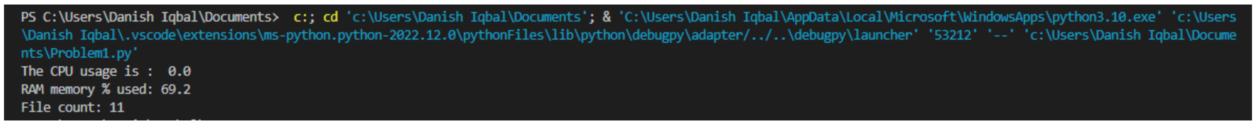
1.2 Output results:

1.2.1



```
PS C:\Users\Danish Iqbal\Documents>  & 'C:\Users\Danish Iqbal\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\Danish Iqbal\.vscode\extensions\ms-python
.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '52952' '--' 'c:\Users\Danish Iqbal\Documents\Problem1.py'
The CPU usage is :  0.0
RAM memory % used: 71.9
File count: 11
```

1.2.2



```
PS C:\Users\Danish Iqbal\Documents>  c:; cd 'c:\Users\Danish Iqbal\Documents'; & 'C:\Users\Danish Iqbal\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users
\Danish Iqbal\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53212' '--' 'c:\Users\Danish Iqbal\Docume
nts\Problem1.py'
The CPU usage is :  0.0
RAM memory % used: 69.2
File count: 11
```

1.3 Explanation:

Please note that I have used the windows system. So, these tasks are solved on the windows system. In the python code, the standard libraries are imported such as os and psutil as showed in the code.

The CPU usage is a data which is based on the time interval. In 1.2.1 the time used is 10 minutes whereas in 1.2.2 the time used is 20 minutes.

The following python code also give results of how much memory is used. It also counts the number of files opened currently in the system. The following python code runs parallel in the system and gives all the information as asked in the questions.

2. Problem

Implement a program that synchronizes two folders: source and replica. The program should maintain a full, identical copy of destination folder at replica folder.

Requirements:

- Synchronization must be one-way: after the synchronization content of the replica folder should be modified to exactly match content of the source folder.
- Synchronization should be performed periodically.
- File creation/copying/removal operations should be logged to a file and to the console output.
- Folder paths, synchronization interval and log file path should be provided using the command line arguments.

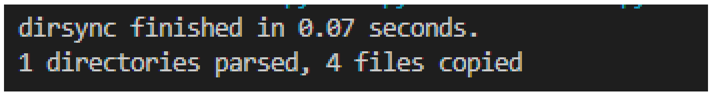
Solutions

Programming

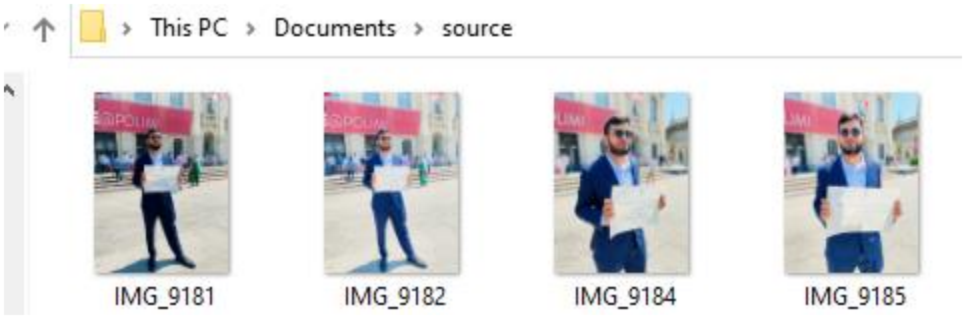
```
from dirsync import sync

sync('C:\\Users\Danish Iqbal\Documents\source', 'C:\\Users\Danish
Iqbal\Documents\Replica', 'sync', purge = True)
```

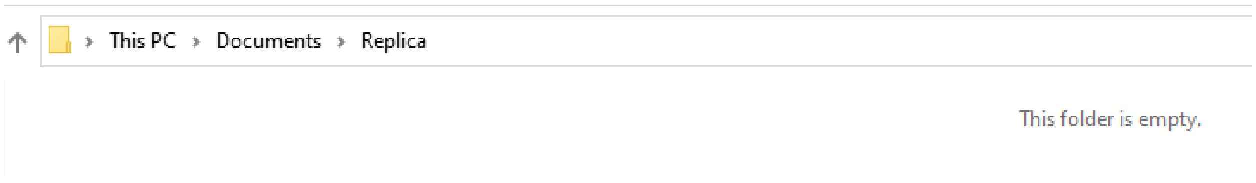
2.1 Output

2.2.1 Example

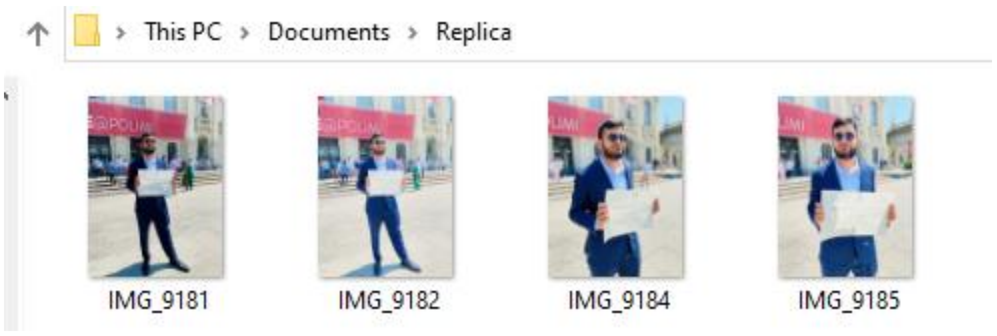


```
dirsync finished in 0.07 seconds.
1 directories parsed, 4 files copied
```

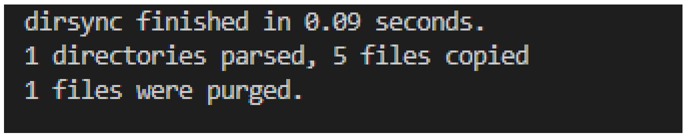




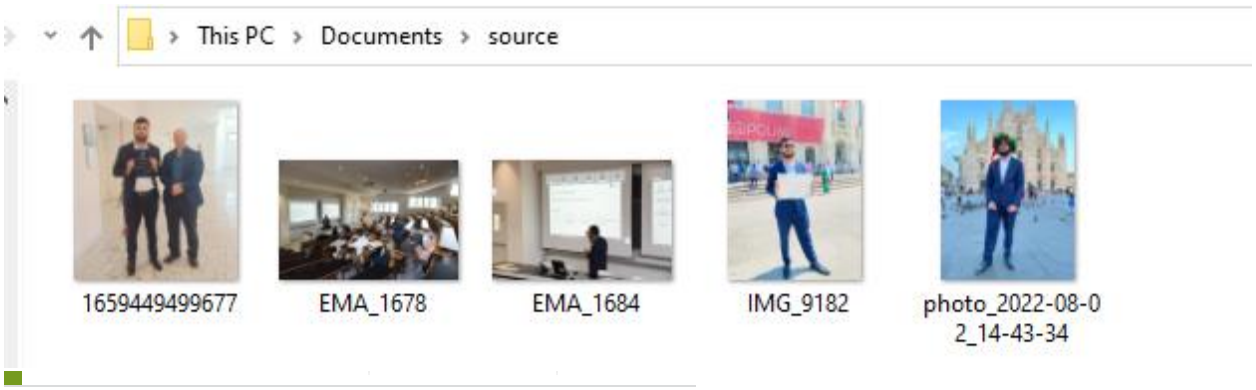
This folder is empty.

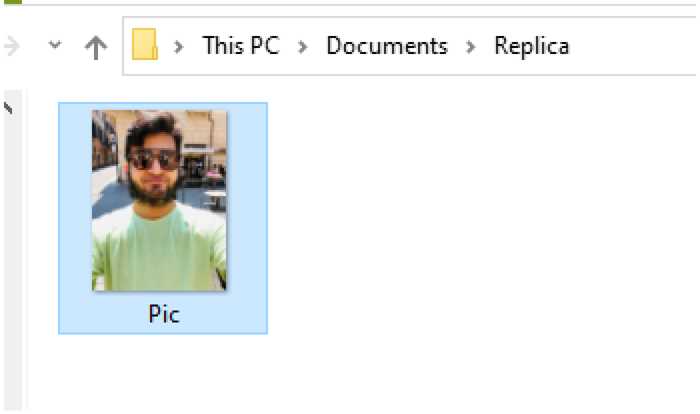


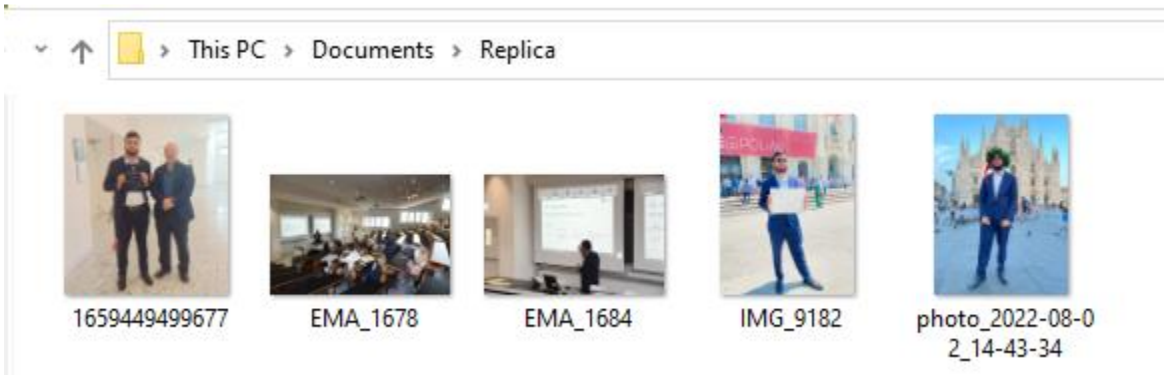
2.2.2 Example 2 (for case where already there are file in Replica)



```
dirsync finished in 0.09 seconds.
1 directories parsed, 5 files copied
1 files were purged.
```







2.2 Explanation

As shown in the example 2.2.1, python code can successfully copy the file from source to replica. It also gives the time taken to complete the task. Also, it shows the number of files. As you can see, in 2.2.2 there are no files initially in the replica file and there are four files which are images in source. It totally copies to replica file

Moreover, it deletes the file which are not in the source file as shown in the example 2 in the output section of 2.22.

If we want to synchronization at a regular interval, we can do easily by adding task scheduler in windows 10, by creating task and giving a fixed time interval, it will run synchronize at a regular interval of time.

3. Problem

Implement a client-server application that follows the next algorithm:

1. Server keeps ports 8000 and 8001 open.
2. Each client generates a unique identifier for itself.
3. Client connects to server port 8000, provides its unique identifier and gets a unique code from the server.
4. Client connects to server port 8001, provides a text message, its identifier and code that it received on step 2.
5. If client code does not match client identifier, server returns an error to the client.
6. If client code is correct, server writes the provided text message to a log file.

Server should be able to simultaneously work with at least 50 clients.
It is acceptable (although not required) to use a high-level protocol (e. g. HTTP) for communication between client and server.

Solutions

3.1 Programming

3.1.1 Server Side

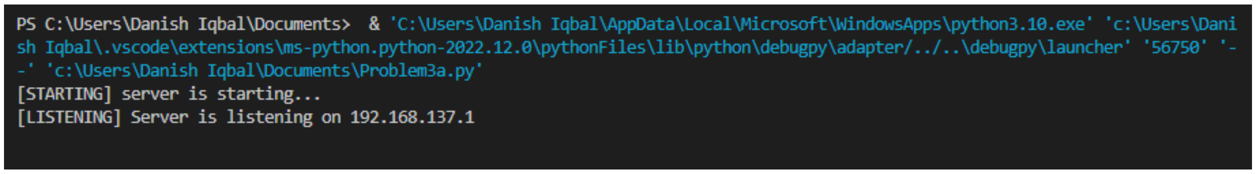
```python
import socket
import threading

HEADER = 64
PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")

    connected = True
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False
            print(f"[{addr}] {msg}")
            conn.send("Msg received".encode(FORMAT))

    conn.close()


def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")


print("[STARTING] server is starting...")
start()
```

3.1.2 Client Side

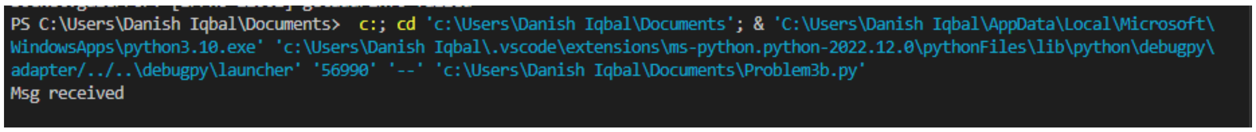
```
import socket

HEADER = 64
PORT = 5050
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
SERVER = "192.168.137.1"
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)
    print(client.recv(2048).decode(FORMAT))

send("Welcome to Czec")
input()
send("Hello Veeam Software!")
input()
```

```
send("Hello Danish Iqbal!")

send(DISCONNECT_MESSAGE)
```

3.2 Outputs



```
PS C:\Users\Danish Iqbal\Documents>  & 'C:\Users\Danish Iqbal\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\Dani
sh Iqbal\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56750' '-
-' 'c:\Users\Danish Iqbal\Documents\Problem3a.py'
[STARTING] server is starting...
[LISTENING] Server is listening on 192.168.137.1
```



```
PS C:\Users\Danish Iqbal\Documents>  c:; cd 'c:\Users\Danish Iqbal\Documents'; & 'C:\Users\Danish Iqbal\AppData\Local\Microsoft\
WindowsApps\python3.10.exe' 'c:\Users\Danish Iqbal\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\
adapter/../..\debugpy\launcher' '56990' '--' 'c:\Users\Danish Iqbal\Documents\Problem3b.py'
Msg received
```

3.3 Explanation

The python code for server side and client side are written and successfully run-in visual studio code. As you can see from output section server is starting and a proper connection is established between server and client. It can work for one server and multiple clients also for 50 clients.