## 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. Programming

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

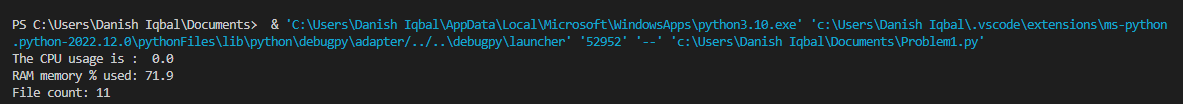
    if os.path.isfile(os.path.join(dir\_path, path)):

        count += 1

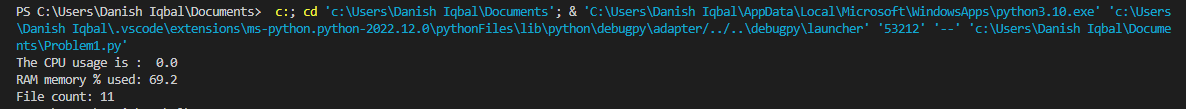
print('File count:', count)

* 1. Output results:

1.2.1



1.2.2



* 1. 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.

## 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)

* 1. Output

2.2.1 Example



Graphical user interface, application

Description automatically generated

A picture containing table

Description automatically generated

Graphical user interface, application

Description automatically generated

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

Text

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* 1. 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.

## 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

* 1. Programming
     1. Server Side

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()

* + 1. 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()

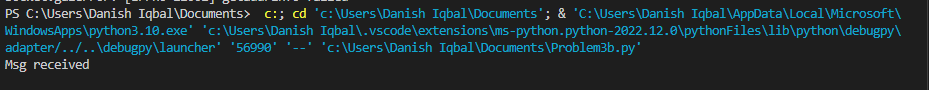
send("Hello Danish Iqbal!")

send(DISCONNECT\_MESSAGE)

* 1. Outputs

Text

Description automatically generated



* 1. 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.