INSTITUTO INFNET

CARLOS HENRIQUE DOS SANTOS RODRIGUES JUNIOR

ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

PROJETO DE BLOCO – TP(?)

Rio de Janeiro, novembro

2020

CARLOS HENRIQUE DOS SANTOS RODRIGUES JUNIOR

ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

PROJETO DE BLOCO – TP(?)

Trabalho apresentado no curso de graduação do Instituto Infnet.

Professor: Adriano Saad

Rio de Janeiro, novembro

2020

Sumário

[Capítulo 1 5](#_Toc56108585)

[Objetivo 5](#_Toc56108586)

[Bibliotecas 5](#_Toc56108587)

[Desenvolvimento 5](#_Toc56108588)

[Problemas observados: 6](#_Toc56108589)

[Soluções de problemas anteriores: 6](#_Toc56108590)

[Capítulo 2 7](#_Toc56108591)

[Objetivo 7](#_Toc56108592)

[Bibliotecas 7](#_Toc56108593)

[Desenvolvimento 7](#_Toc56108594)

[Problemas observados 9](#_Toc56108595)

[Soluções de problemas anteriores 9](#_Toc56108596)

[Capítulo 3 10](#_Toc56108597)

[Objetivo 10](#_Toc56108598)

[Bibliotecas 10](#_Toc56108599)

[Desenvolvimento 10](#_Toc56108600)

[Problemas observados: 11](#_Toc56108601)

[Soluções de problemas anteriores: 11](#_Toc56108602)

[Correção proporção do tamanho do gráfico de Disco 11](#_Toc56108603)

[Correção proporção do tamanho do gráfico de Memória 12](#_Toc56108604)

[Correção carrossel 12](#_Toc56108605)

[Capítulo 4 13](#_Toc56108606)

[Objetivo 13](#_Toc56108607)

[Bibliotecas 13](#_Toc56108608)

[Desenvolvimento 14](#_Toc56108609)

[Problemas observados: 14](#_Toc56108610)

[Soluções de problemas anteriores: 14](#_Toc56108611)

[Capítulo 5 15](#_Toc56108612)

[Objetivo 15](#_Toc56108613)

[Bibliotecas 15](#_Toc56108614)

[Desenvolvimento 16](#_Toc56108615)

[Infraestrutura 16](#_Toc56108616)

[Problemas observados 16](#_Toc56108617)

[Soluções de problemas anteriores 16](#_Toc56108618)

[Capítulo 6 17](#_Toc56108619)

[Objetivo 17](#_Toc56108620)

[Bibliotecas 17](#_Toc56108621)

[Desenvolvimento 18](#_Toc56108622)

[Problemas observados: 18](#_Toc56108623)

[Soluções de problemas anteriores: 18](#_Toc56108624)

[Capítulo 7 19](#_Toc56108625)

[Objetivo 19](#_Toc56108626)

[Bibliotecas 19](#_Toc56108627)

[Desenvolvimento 19](#_Toc56108628)

[Problemas observados: 19](#_Toc56108629)

[Soluções de problemas anteriores: 19](#_Toc56108630)

[Anexos 22](#_Toc56108631)

[Código fonte TP2 23](#_Toc56108632)

[Código fonte TP3 29](#_Toc56108633)

[Código fonte TP4 36](#_Toc56108634)

[Código fonte TP5 45](#_Toc56108635)

[Código fonte TP6 56](#_Toc56108636)

[Código fonte TP7 68](#_Toc56108637)

[Evidências 80](#_Toc56108638)

[Evidência TP2 81](#_Toc56108639)

[Evidências TP3 83](#_Toc56108640)

[Evidências TP4 87](#_Toc56108641)

[Evidências TP5 91](#_Toc56108642)

# Capítulo 1

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP2 de projeto de bloco.

## Objetivo

* Criar uma barra de indicação da porcentagem do uso de memória
* Criar uma barra de indicação da porcentagem do uso de CPU
* Criar uma barra de indicação da porcentagem de uso do Disco
* Informar o IP da máquina

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform

## Desenvolvimento

A aplicação entregue, consiste em um utilitário onde é possível visualizar os dados de IP da rede, uso da CPU e quantidade de memória utilizada.

Para se obter o IP (Internet Protocol) da máquina, foi criado uma função chamada getIp. Essa função precisa do pacote ***plataform*** e ***psutil*** instalado. O pacote ***plataform*** se faz necessário para se obter o sistema operacional em que o usuário está executando a aplicação para que concomitante a função ***net\_if\_addrs()*** da biblioteca ***psutil*** seja possível obter o IP. A função ***net\_if\_addrs()*** do pacote ***psutil*** retorna informações diferentes baseado no sistema operacional em execução. Dessa forma faz se necessário saber qual chave utilizar obter a informação correta.

Para obter o percentual de uso da CPU foi utilizado a função ***cpu\_percent(interval=0)***. Essa função retorna um ***float*** representando a utilização do sistema, ela pode receber dois parâmetros: ***interval*** e ***percpu***. ***Interval*** configura a velocidade em que os dados serão atualizados. Já o ***percpu*** recebe um booleano e quando igual a True, retorna um vetor contendo floats referente a utilização do sistema no tempo.

Afim de obter os dados da memória, foi utilizado a função ***virtual\_memory()***, que retorna dados estatísticos da memória do computador. Essa função retorna um objteto contendo os seguintes atributos: ***total*** e ***available***.

* Total: representa o espaço físico total
* Available: representa o espaço disponível não incluindo o swap

Por fim, para se obter os dados de disco, foi utilizado a função ***disk\_usage(‘/’)*** também do pacote ***psutil***. Essa função retorna as seguintes propriedades: total, used, free e percent.

* Total: representa o volume total da partição
* Used: representa o valor total usado pela partição
* Free: representa o espaço livre da partição
* Percent: representa o percentual usado pela partição

## Problemas observados:

Sem problemas observados.

## Soluções de problemas anteriores:

Sem soluções anteriores.

# Capítulo 2

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP3 de projeto de bloco.

## Objetivo

* Extrair o código anterior e criar 5 visualizações diferentes:
  + Tela com informações do processador
    - Informar o modelo/nome da CPU
    - Informar o tipo de arquitetura
    - Adicionar informação da palavra do processador
    - Informar a frequência total e frequência de uso da CPU
    - Informar o número total de núcleos (físico) e threads (lógico)
  + Tela com informações de memória
  + Tela com informações de disco
  + Tela com informações de IP
* Implementar uma navegação em slide. Sempre que o usuário clicar nas setas da direita ou esquerda a tela deve mudar como um carrossel.
* Implementar uma tela de resumo

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform

## Desenvolvimento

Nessa etapa, foi solicitado que os dados coletados no capítulo anterior, fossem apresentados em telas diferentes. Para isso, foram criadas 8 superfícies:

* Superficie\_info\_cpu
  + Superficie\_grafico\_cpu
* Superficie\_info\_disco
  + Superficie\_grafico\_disco
* Superficie\_info\_memoria
  + Superficie\_grafico\_memoria
* Superficie\_info\_rede
* Superficie\_resumo

Onde a superfície com o prefixo “info” seria responsável por exibir em tela os dados e as superfícies com o prefixo “gráfico” seriam responsáveis pela apresentação dos gráficos respectivos. Para alcançar tal objetivo, foi utilizado o módulo do ***pygame*** e a função ***Suface((largura\_tela, altura\_tela))***. As dimensões definidas para esse projeto foram 800x600, dessa forma todas as superfícies de informação a altura definida foi 600/3 ou senha 200 pixels.

Foi solicitado também que fosse implementado um carrossel para a troca de tela. Sempre que o usuário clicar na seta da direita ou esquerda a aplicação deve reagir a essa interação apresentando a nova tela. Para alcançar esse objetivo, foi criado uma variável auxiliar chamada de “***posicao\_atual***”. Durante o processo de renderização do pygame é feito um monitoramento dos eventos gerados pelo usuário, sempre que o evento “***pygame\_.KEYDOWN***” e “***pygame.K\_RIGHT***” ou “***pygame.K\_LEFT***” ou “***pygame.K\_SPACE***” for identificado é incrementado ou decrementado o valor 1 a variável “***posicao\_atual***”. Essa variável é passada pra função “***getEnvolucro(posicao\_atual)***” que é responsável por renderizar as superfícies respectivas.

|  |  |
| --- | --- |
| Posição atual | Superfície |
| 0 | Envolucro\_dados\_cpu() |
| 1 | Envolucro\_dados\_memoria() |
| 2 | Envolucro\_dados\_disco() |
| 3 | Envolucro\_dados\_rede() |
| 4 | Resumo() |

Para obter os dados de CPU, foi utilizado a biblioteca ***“cpuinfo”*** e a função “***get\_cpu\_info()”***. Onde o retorno obtido foi:

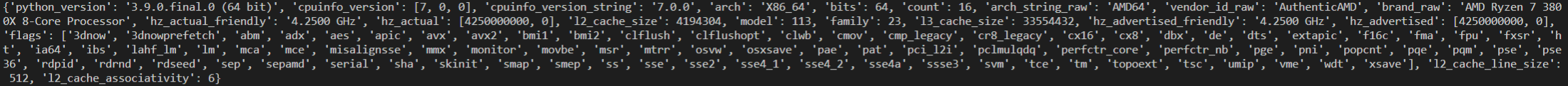


Figura 1 Resposta CpuInfo

Para se obter as informações exigidas nesse capitulo, foi necessário obter os seguintes valores da chave do “***cpuinfo***”:

* Brand\_raw = Nome Processador
* Arch = Arquitetura do processador
* Bits = Bits do processador
* Hz\_actual\_friendly = Frequência de atuação do processador
* Count = Núcleos físicos do processador

Para se obter os dados do disco, também foi utilizado a biblioteca “***psutil***” com a função “***disk\_usage(‘/’)”***  onde a “**/**” informada como parâmetro faz referência ao disco principal da máquina.O resultado obtido foi:

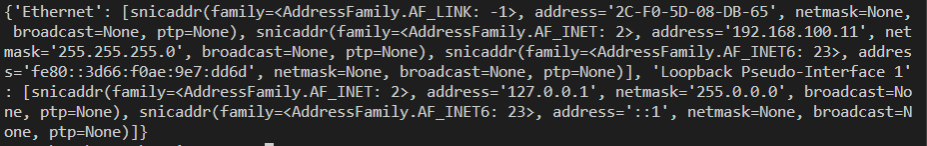


Para se obter as informações exigidas nesse capitulo, foi necessário obter os seguintes valores da chave do “***disk\_usage***”:

* Percent = Percentual usado
* Used = Kb usado
* Free = Kb livre

Para se obter o valor usado e livre em GB foi aplicado o seguinte cálculo:

Para se obter o endereço IP da máquina, também foi utilizado a biblioteca “***psutil***” e a função “***net\_if\_addrs()”.*** O resultado obtido pela função “***net\_if\_addrs()***”, foi:



Entretanto para se obter o endereço IP da máquina do usuário, faz se necessário saber em qual sistema operacional a aplicação está executando, pois a chave presente no JSON pode variar entre:

* Wlp3s0 para Linux
* Ethernet para Windows

## Problemas observados

* A proporção do tamanho das barras da memória e disco (HD) estão desproporcionais
* Ao chegar a última tela, o carrossel não retorna para a primeira página

## Soluções de problemas anteriores

Sem soluções anteriores.

# Capítulo 3

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP4 de projeto de bloco.

## Objetivo

* Criar uma ou mais funções que retornem ou apresentem informações sobre diretórios e arquivos
* Criar uma ou mais funções que retornem ou apresentem informações sobre processos do sistema. As informações podem ser: PID, nome do executável, consumo de processamento, consumo de memória, entre outras

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Subprocess
* Os
* Time
* Socket

## Desenvolvimento

Para atender ao requisito de obtenção dos dados de arquivos e diretórios, foi criado uma função chamada “***mostrar\_dados\_diretorio()***”, essa função retorna uma lista contendo os arquivos e diretórios presentes no path do projeto. Ela utiliza a biblioteca “***os***” e a função “***listdir()***”. O resultado obtido pela função foi:



Para se obter os detalhes dos documentos também foi utilizado a biblioteca “***os***”. Para se obter os valores solicitados, foram utilizadas as seguintes funções:

* St\_size: obter informações do tamanho do arquivo.
* St\_ctime: em alguns sistemas (como Unix) é a hora da última mudança de metadados e, em outros (como Windows), é a hora de criação.
* St\_mtime: hora da última modificação.

Por fim, para se obter os dados do processo, foi utilizado a biblioteca “***subprocess***” e a função “**popen().pid**”. Essa função inicia um processo e retorna o PID do mesmo. PIP é o código identificado do processo e é gerenciado pelo sistema operacional.

Para se obter os dados desejados do processo, fez se necessário obter os seguintes atributos:

* Memory\_percent
* Memory\_info.rss
* Create\_time()

## Problemas observados:

* A função responsável por obter o endereço IP da máquina do usuário, só está retornando o IP da posição inicial da tupla. Caso o usuário possua duas ou mais interfaces a função não obtém os demais endereços.

## Soluções de problemas anteriores:

### Correção proporção do tamanho do gráfico de Disco

O novo código para apresentar o gráfico do consumo do disco foi alterado para:

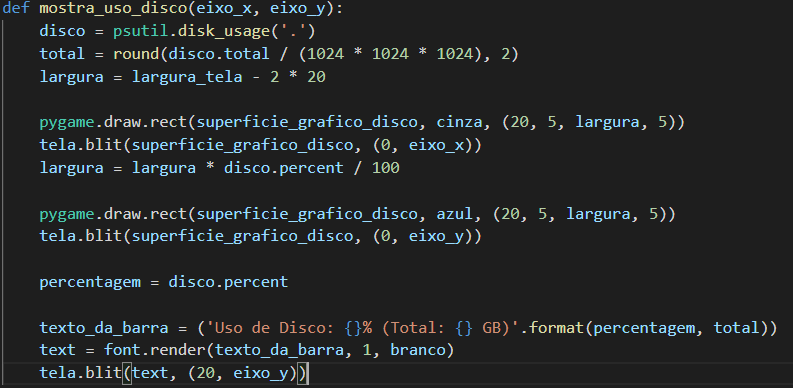


Figura 2 Código para apresentar o consumo do disco

### Correção proporção do tamanho do gráfico de Memória

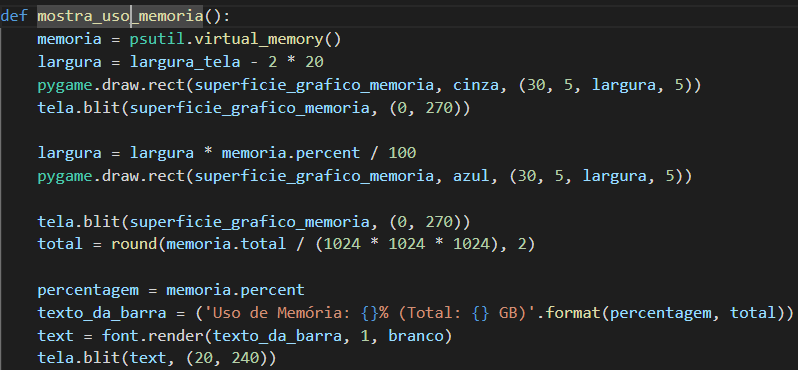
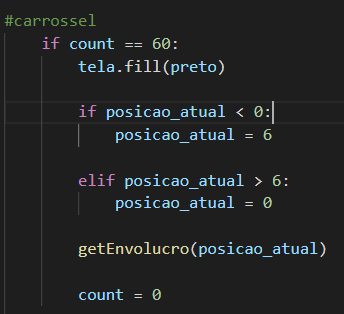


Figura 3 Código para apresentar o consumo de memória

### Correção carrossel

Foi identificado um BUG presente na incrementação da variável global “**posicao\_atual**” onde não era verificado se essa variável era igual ao maior ou menor número de telas existentes. Sendo assim o código corrigido ficou da seguinte maneira:



# Capítulo 4

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP5 de projeto de bloco.

## Objetivo

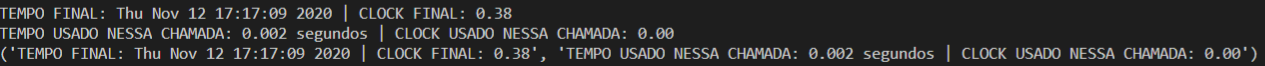
* Utilizar o módulo ‘sched’ para chamar as funções criadas no TP4 que retornam as informações sobre diretórios e arquivos.
* Realizar um escalonamento das chamadas das funções com o módulo ‘sched’ e medir o tempo total utilizado por cada chamada com o módulo ‘time’. Você pode escolher com quais funções do seu projeto realizar o escalonamento, deixando indicado no relatório.

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Nmap

## Desenvolvimento

Para atender ao requisito da utilização do módulo “***shed***”, foi criado uma função chamada “***get\_shed\_sheduler\_arquivos()***”, que tem por finalidade medir o tempo de execução de uma função. Após instanciar a configuração do “***scheduler***”, faz se necessário utilizar a função “***enter()***”, a mesma recebe como parâmetro a função que deverá ser monitorada. Entretanto para se obter o resultado corretamente, é necessário criar variáveis auxiliares para medir o tempo passado. O resultado exibido no console, mediante as medições foi:

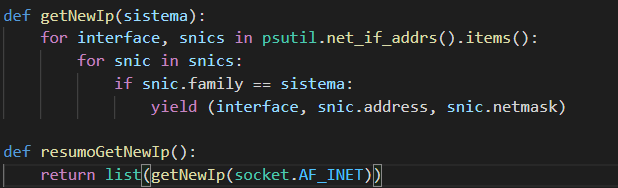


## Problemas observados:

* No capitulo passado, foi solicitado a listagem dos processos em execução e não iniciar um processo como foi feito.

## Soluções de problemas anteriores:

* Correção obtenção IPs da máquina do usuário:



Resultado obtido:

# Capítulo 5

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP6 de projeto de bloco.

## Objetivo

* Criar funções que retornem ou apresentem informações sobre as máquinas pertencentes à sub rede do IP específico
* Criar funções que retornem os status das portas dos diferentes IPs obtidos nessa sub rede.

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Sched
* Nmap
* Threading
* Time

## Desenvolvimento

### Infraestrutura

Para obter sucesso durante a varredura dos hosts da sub rede foi necessário ter instalado previamente o software Nmap além da biblioteca nmap.

Para obter sucesso durante a varredura dos hosts pertencentes a rede foi necessário ter previamente o Ip da máquina do usuário. Em seguida, realizar um ping em todos os ips possíveis. Os hosts que responderam ao procedimento são postos em uma lista e em seguida é feito a verificação das portas ativas do mesmo. Para realizar essa etapa, foi utilizado a biblioteca ***nmap.PortScanner().*** Em seguida é utilizado a função **scan** para obter o status da porta.

## Problemas observados

* Ao executar a aplicação, o software entregue apresenta uma tela preta. Essa tela representa o travamento da aplicação ao iniciar a varredura dos hosts da rede. Uma possível solução para essa questão, seria iniciar a leitura da rede em um thread liberando o processo e informar ao usuário que os dados da rede estão sendo coletados.

## Soluções de problemas anteriores

* Processos em execução

Para solucionar esse problema, foram criadas duas classes: Processo e Porta.

A classe Processo possui os seguintes atributos:

* + Pid
  + Nome
  + Percentual\_uso
  + Memoria\_usada
  + Threads\_processo
  + Tempo\_usuario
  + Data\_criacao

Já a classe Porta, possui os seguintes atributos:

* + Porta
  + State

Dessa forma, para se obter os dados dos processos em execução, foi utilizado a biblioteca **psutil** e a função **pids()**. Os PIDs obtidos foram postos em uma lista. Por fim, foi utilizado a biblioteca **psutil** novamente juntamente com a função ***Process*** passando como paramento o PID obtido. Os dados propostos foram obtidos através das seguintes funções:

* + Memory\_percent()
  + Memory\_info().rss
  + Num\_threads()
  + Cpu\_time().user
  + Create\_time

# Capítulo 6

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP7 de projeto de bloco.

## Objetivo

* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: IP, *gateway*, máscara de subrede.
* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: Uso de dados de rede por interface.
* Crie uma ou mais funções que retornem ou apresentem as seguintes informações de redes: Uso de dados de rede por processos.

## Bibliotecas

* Pygame
* Psutil
* Cpuinfo
* Platform
* Subprocess
* Os
* Time
* Socket
* Sched
* Nmap
* Threading
* Time

## Desenvolvimento

## Problemas observados:

* Travamento da aplicação ao inicializar:

Para solucionar esse problema, foi criado uma classe chamada **ThreadRede**. Ela herda da biblioteca **threading.Thread** nela foi criado uma função chamado **run**, que é responsável por inicializar a leitura dos hosts da rede, através da função **get\_hosts()**. Foi abordado essa solução, pois executando essa varredura em uma thread separada, o software será liberado permitindo então a execução de outras funções.

## Soluções de problemas anteriores:

# Capítulo 7

Esta sessão contemplam todas as bibliotecas e funções utilizadas para atender aos requisitos do TP8 de projeto de bloco.

## Objetivo

## Bibliotecas

## Desenvolvimento

## Problemas observados:

## Soluções de problemas anteriores:

# Anexos

## Código fonte TP2

import pygame

import psutil

import cpuinfo

import platform

import os

import time

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Informações de CPU")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

####################################################################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(cinza)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width()- 2\*y - (num\_cpu+1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, vermelho, (d, y, larg, alt))

        pygame.draw.rect(s, azul, (d, y, larg, (1-i/100)\*alt))

        d = d + larg + desl

        tela.blit(s, (0, 140))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

####################################################################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + "%"

    total\_disco = str(round(disco.used/(1024\*\*3), 2)) + "GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco():

    disco = psutil.disk\_usage('/')

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 50, larg/2, 70))

    larg = larg \* disco.percent/100

    pygame.draw.rect(superficie\_grafico\_disco, Dim, (20, 50, larg/2, 70))

    total = round(disco.total/(1024\*\*3), 2)

    texto\_barra = "Uso de Disco: (Total: " + str(total) + "GB):"

    texto = font.render(texto\_barra, 1, branco)

    superficie\_grafico\_disco.blit(texto, (20, 10))

    tela.blit(superficie\_grafico\_disco, (0, 200))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco()

####################################################################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    mem = psutil.virtual\_memory()

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (20, 50, larg/2, 70))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    larg = larg \* mem.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, preto, (20, 5, larg, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(mem.total / (1024 \* 1024 \* 1024), 2)

    percent = mem.percent

    msg = ('Uso de Memória: {}% (Total: {} GB)'.format(percent, total))

    text = font.render(msg, 1, branco)

    tela.blit(text, (20, 270))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

####################################################################################################################

def getIp():

    plataforma = platform.system()

    dic\_interfaces = psutil.net\_if\_addrs()

    if plataforma == 'Linux':

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

    elif plataforma == "Windows":

        ip = (dic\_interfaces['Ethernet'][1].address)

        return ip

    else:

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

def mostrar\_texto\_rede():

  plataforma = "IP: " + getIp()

  mostra\_texto(superficie\_info\_rede, plataforma, 10)

  tela.blit(superficie\_info\_rede, (20, 0))

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

####################################################################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

####################################################################################################################

posicao\_atual = 0

posicao\_cpu = 0

posicao\_mem = 1

posicao\_dis = 2

posicao\_red = 3

posicao\_res = 3

######################################################################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < posicao\_cpu:

            posicao\_atual = 3

        elif posicao\_atual > posicao\_res:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte TP3

import pygame

import psutil

import cpuinfo

import platform

import os

import time

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Informações de CPU")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

####################################################################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(cinza)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width()- 2\*y - (num\_cpu+1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, vermelho, (d, y, larg, alt))

        pygame.draw.rect(s, azul, (d, y, larg, (1-i/100)\*alt))

        d = d + larg + desl

        tela.blit(s, (0, 140))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

####################################################################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + "%"

    total\_disco = str(round(disco.used/(1024\*\*3), 2)) + "GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco():

    disco = psutil.disk\_usage('/')

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 50, larg/2, 70))

    larg = larg \* disco.percent/100

    pygame.draw.rect(superficie\_grafico\_disco, Dim, (20, 50, larg/2, 70))

    total = round(disco.total/(1024\*\*3), 2)

    texto\_barra = "Uso de Disco: (Total: " + str(total) + "GB):"

    texto = font.render(texto\_barra, 1, branco)

    superficie\_grafico\_disco.blit(texto, (20, 10))

    tela.blit(superficie\_grafico\_disco, (0, 200))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco()

####################################################################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = getIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip, True, preto), (155, pos\_y))

####################################################################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    mem = psutil.virtual\_memory()

    larg = largura\_tela - 2\*20

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (20, 50, larg/2, 70))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    larg = larg \* mem.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, preto, (20, 5, larg, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(mem.total / (1024 \* 1024 \* 1024), 2)

    percent = mem.percent

    msg = ('Uso de Memória: {}% (Total: {} GB)'.format(percent, total))

    text = font.render(msg, 1, branco)

    tela.blit(text, (20, 270))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

####################################################################################################################

def getIp():

    plataforma = platform.system()

    dic\_interfaces = psutil.net\_if\_addrs()

    if plataforma == 'Linux':

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

    elif plataforma == "Windows":

        ip = (dic\_interfaces['Ethernet'][1].address)

        return ip

    else:

        ip = (dic\_interfaces['wlp3s0'][0].address)

        return ip

def mostrar\_texto\_rede():

  plataforma = "IP: " + getIp()

  mostra\_texto(superficie\_info\_rede, plataforma, 10)

  tela.blit(superficie\_info\_rede, (20, 0))

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

####################################################################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        resumo()

####################################################################################################################

posicao\_atual = 0

posicao\_cpu = 0

posicao\_mem = 1

posicao\_dis = 2

posicao\_red = 3

posicao\_res = 4

######################################################################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < posicao\_cpu:

            posicao\_atual = 5

        elif posicao\_atual > posicao\_res:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte TP4

import pygame

import psutil

import cpuinfo

import subprocess

import os

import time

import socket

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

####################################################################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

####################################################################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

####################################################################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

####################################################################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

####################################################################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

####################################################################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

meu\_ip = ''

hosts = []

ips = resumoGetNewIp()

###################################################################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

####################################################################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

####################################################################################################################

posicao\_atual = 0

######################################################################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte TP5

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

####################################################################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

####################################################################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

####################################################################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

####################################################################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

####################################################################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

####################################################################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

###################################################################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

####################################################################################################################

def getEnvolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

####################################################################################################################

posicao\_atual = 0

######################################################################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 6

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        getEnvolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte TP6

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

#superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    print("\nMapeamento pronto...")

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def get\_envolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 4

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        get\_envolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

## Código fonte TP7

import pygame

import psutil

import cpuinfo

import platform

import subprocess

import os

import time

import socket

import nmap

info\_cpu = cpuinfo.get\_cpu\_info()

psutil.cpu\_percent(interval=1, percpu=True)

preto = (0, 0, 0)

branco = (255, 255, 255)

cinza = (100, 100, 100)

Dim = (105,105,105)

azul = (0, 0, 255)

vermelho = (255, 0, 0)

largura\_tela = 800

altura\_tela = 600

# configurações da tela

tela = pygame.display.set\_mode((largura\_tela, altura\_tela))

pygame.display.set\_caption("Projeto de bloco - Carlos Henrique")

pygame.display.init()

# configurando a fonte

pygame.font.init()

font = pygame.font.SysFont(None, 20)

superficie\_info\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_cpu = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_disco = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_grafico\_memoria = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_rede = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

superficie\_info\_resumo = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

#superficie\_info\_ips = pygame.surface.Surface((largura\_tela, int(altura\_tela/3)))

clock = pygame.time.Clock()

terminou = False

count = 60

class Host:

    def \_\_init\_\_(self, ip, name):

        self.ip = ip

        self.name = name

        self.ports = []

class Port:

    def \_\_init\_\_(self, port, state):

        self.port = port

        self.state = state

meu\_ip = ''

hosts = []

#########################################################################

def mostra\_info\_cpu():

    superficie\_info\_cpu.fill(branco)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Nome:", "brand\_raw", 110)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Arquitetura:", "arch", 30)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Palavra (bits):", "bits", 50)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Frequência (MHz):", "hz\_actual\_friendly", 70)

    mostra\_texto\_cpu(superficie\_info\_cpu, "Núcleos (físicos):", "count", 90)

    tela.blit(superficie\_info\_cpu, (0, 0))

def mostra\_texto\_cpu(s1, nome, chave, pos\_y):

    text = font.render(nome, True, preto)

    s1.blit(text, (40, pos\_y))

    if chave == "freq":

        s = str(round(psutil.cpu\_freq().current, 2))

    elif chave == "nucleos":

        s = str(psutil.cpu\_count())

        s = s + " (" + str(psutil.cpu\_count(logical=False)) + ")"

    else:

        s = str(info\_cpu[chave])

        text = font.render(s, True, cinza)

        superficie\_info\_cpu.blit(text, (155, pos\_y))

# Desenha grafico

def mostrar\_uso\_cpu(s):

    l\_cpu\_percent = psutil.cpu\_percent(percpu=True)

    s.fill(preto)

    capacidade = psutil.cpu\_percent(interval=1)

    num\_cpu = len(l\_cpu\_percent)

    x = y = 10

    desl = 10

    alt = s.get\_height() - 2\*y

    larg = (s.get\_width() - 2 \* y - (num\_cpu + 1) \* desl) / num\_cpu

    d = x + desl

    for i in l\_cpu\_percent:

        pygame.draw.rect(s, azul, (d, y + 20, larg, alt))

        pygame.draw.rect(s, Dim, (d, y + 20, larg, (1 - i / 100) \* alt))

        d = d + larg + desl

    text = font.render("Uso de CPU por núcleo total: " + str(capacidade) + "%", 1, branco)

    s.blit(text, (20, 5))

    tela.blit(s, (0, 200))

def envolucro\_dados\_cpu():

    mostra\_info\_cpu()

    mostrar\_uso\_cpu(superficie\_grafico\_cpu)

#########################################################################

def mostrar\_info\_disco():

    superficie\_info\_disco.fill(branco)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_usado", "Usado:", 10)

    mostrar\_texto\_disco(superficie\_info\_disco, "total\_disco", "Total:", 30)

    mostrar\_texto\_disco(superficie\_info\_disco, "disco\_livre", "Livre:", 50)

    tela.blit(superficie\_info\_disco, (0, 0))

def mostrar\_texto\_disco(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('.')

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_usado= str(round(disco.percent, 2)) + " %"

    total\_disco = str(round(disco.total / (1024\*\*3), 2)) + " GB"

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + " GB"

    if chave == "total\_disco":

        s1.blit(font.render(total\_disco, True, preto), (155, pos\_y))

    elif chave == "disco\_usado":

        s1.blit(font.render(disco\_usado, True, preto), (155, pos\_y))

    elif chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

def mostra\_uso\_disco(eixo\_x, eixo\_y):

    disco = psutil.disk\_usage('.')

    total = round(disco.total / (1024 \* 1024 \* 1024), 2)

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_disco, cinza, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_x))

    largura = largura \* disco.percent / 100

    pygame.draw.rect(superficie\_grafico\_disco, azul, (20, 5, largura, 5))

    tela.blit(superficie\_grafico\_disco, (0, eixo\_y))

    percentagem = disco.percent

    texto\_da\_barra = ('Uso de Disco: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, eixo\_y))

def envolucro\_dados\_disco():

    mostrar\_info\_disco()

    mostra\_uso\_disco(410, 390)

#########################################################################

def resumo():

    superficie\_info\_resumo.fill(branco)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "disco\_livre", "Disco livre:", 10)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "memoria\_livre", "Memoria livre:", 30)

    mostrar\_texto\_resumo(superficie\_info\_resumo, "ip\_rede", "IP:", 50)

    tela.blit(superficie\_info\_resumo, (0, 0))

def mostrar\_texto\_resumo(s1, chave, nome, pos\_y):

    disco = psutil.disk\_usage('/')

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    disco\_livre = str(round(disco.free/(1024\*\*3), 2)) + "GB"

    memoria\_livre = str(round(mem.available/(1024\*\*3), 2))

    ip = resumoGetNewIp()

    if chave == "disco\_livre":

        s1.blit(font.render(disco\_livre, True, preto), (155, pos\_y))

    elif chave == "memoria\_livre":

        s1.blit(font.render(memoria\_livre, True, preto), (155, pos\_y))

    elif chave == "ip\_rede":

        s1.blit(font.render(ip[0][1], True, preto), (155, pos\_y))

#########################################################################

def mostrar\_info\_memoria():

    superficie\_info\_memoria.fill(branco)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "capacidade", "Capacidade:", 10)

    mostrar\_texto\_memoria(superficie\_info\_memoria, "disponivel", "Disponivel:", 30)

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostrar\_texto\_memoria(s1, chave, nome, pos\_y):

    mem = psutil.virtual\_memory()

    texto = font.render(nome, True, preto)

    s1.blit(texto, (40, pos\_y))

    capacidade = round(mem.total/(1024\*\*3),  2)

    disponivel = round(mem.available/(1024\*\*3), 2)

    memoria\_usada = str(capacidade) + "GB"

    memoria\_disponivel = str(disponivel) + "GB"

    if chave == "capacidade":

        s1.blit(font.render(memoria\_usada, True, preto), (155, pos\_y))

    elif chave == "disponivel":

        s1.blit(font.render(memoria\_disponivel, True, preto), (155, pos\_y))

    tela.blit(superficie\_info\_memoria, (0, 0))

def mostra\_uso\_memoria():

    memoria = psutil.virtual\_memory()

    largura = largura\_tela - 2 \* 20

    pygame.draw.rect(superficie\_grafico\_memoria, cinza, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    largura = largura \* memoria.percent / 100

    pygame.draw.rect(superficie\_grafico\_memoria, azul, (30, 5, largura, 5))

    tela.blit(superficie\_grafico\_memoria, (0, 270))

    total = round(memoria.total / (1024 \* 1024 \* 1024), 2)

    percentagem = memoria.percent

    texto\_da\_barra = ('Uso de Memória: {}% (Total: {} GB)'.format(percentagem, total))

    text = font.render(texto\_da\_barra, 1, branco)

    tela.blit(text, (20, 240))

def envolucro\_dados\_memoria():

    mostrar\_info\_memoria()

    mostra\_uso\_memoria()

#########################################################################

def envolucro\_arquivos():

    arquivos = mostrar\_dados\_diretorio()

    apresenta\_dados(arquivos)

def mostrar\_dados\_diretorio():

    lista = os.listdir()

    dados\_organizados = {}

    for i in lista:

        dados\_organizados[i] = []

        dados\_organizados[i].append(os.stat(i).st\_size)

        dados\_organizados[i].append(os.stat(i).st\_ctime)

        dados\_organizados[i].append(os.stat(i).st\_mtime)

    tituloInfo = 'Arquivos do diretório:'

    titulo\_tamanho = '{:>5}'.format('Tamanho')

    titulo\_data\_criacao = '{:>30}'.format('Criação')

    titulo\_data\_modificacao = '{:>38}'.format('Modificação')

    titulo\_nome = '{:>38}'.format('Nome')

    titulo = titulo\_tamanho + titulo\_data\_criacao + titulo\_data\_modificacao + titulo\_nome

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (20, 20))

    textoTitulo = font.render(titulo, 1, branco)

    tela.blit(textoTitulo, (20, 60))

    return dados\_organizados

def apresenta\_dados(arquivos):

    espacos = 100

    for i in arquivos:

        tamanho\_arquivo = arquivos[i][0]/1024

        tamanho\_formatado = '{:>10}'.format(str('{:.2f}'.format(tamanho\_arquivo) + 'KB'))

        data\_criacao = '{:>30}'.format(time.ctime(arquivos[i][0]))

        time\_mod = '{:>30}'.format(time.ctime(arquivos[i][1]))

        nomeArquivo = '{:>30}'.format(i)

        textoArqDir = font.render(tamanho\_formatado + data\_criacao + time\_mod + nomeArquivo, 1, branco)

        tela.blit(textoArqDir, (15, espacos))

        espacos += 25

#########################################################################

def mostrar\_texto\_rede():

    espacos = 100

    for ip in ips:

        texto = font.render(ip[0] + ': ' + ip[1] + ' - ' + ip[2], 1, branco)

        tela.blit(texto, (15, espacos))

        espacos += 25

    titulo = font.render("\*\* Rede \*\*" ,1, azul)

    tela.blit(titulo,(15, 20))

    titulo2 = font.render("\*\* Hosts da Rede \*\*" ,1, azul)

    tela.blit(titulo2,(15, 190))

    for host in hosts:

        host\_name = ""

        if host.name != "":

            host\_name = host.name

        else:

            host\_name = "NÃO LOCALIZADO"

        cor = ""

        if host\_name != "NÃO LOCALIZADO":

            cor = (255, 255, 255)

        else:

            cor = (255, 0, 0)

        texto = font.render(host.ip + ': Nome: ' + host\_name, 1, cor)

        tela.blit(texto, (15, espacos + 5))

        espacos += 15

        for porta in host.ports:

            detalhe\_porta = font.render("Porta: " + str(porta.port) + " - Estado: " + porta.state ,1, branco)

            tela.blit(detalhe\_porta, (15, espacos))

            espacos += 15

        espacos += 10

def mostra\_texto(s1, nome, pos\_y):

    text = font.render(nome, True, branco)

    superficie\_info\_rede.blit(text, (10, pos\_y))

def getNewIp(sistema):

    for interface, snics in psutil.net\_if\_addrs().items():

        for snic in snics:

            if snic.family == sistema:

                yield (interface, snic.address, snic.netmask)

def resumoGetNewIp():

    return list(getNewIp(socket.AF\_INET))

def retorna\_codigo\_ping(hostname):

    """Usa o utilitario ping do sistema operacional para encontrar   o host. ('-c 5') indica, em sistemas linux, que deve mandar 5   pacotes. ('-W 3') indica, em sistemas linux, que deve esperar 3   milisegundos por uma resposta. Esta funcao retorna o codigo de   resposta do ping """

    plataforma = platform.system()

    args = []

    if plataforma == "Windows":

        args = ["ping", "-n", "1", "-l", "1", "-w", "100", hostname]

    else:

        args = ['ping', '-c', '1', '-W', '1', hostname]

    ret\_cod = subprocess.call(args, stdout=open(os.devnull, 'w'), stderr=open(os.devnull, 'w'))

    return ret\_cod

def verifica\_hosts\_validos(base\_ip):

    """Verifica todos os host com a base\_ip entre 1 e 255 retorna uma lista com todos os host que tiveram resposta 0 (ativo)"""

    print("Mapeando\r")

    host\_validos = []

    return\_codes = dict()

    for i in range(1, 255):

        return\_codes[base\_ip + '{0}'.format(i)] = retorna\_codigo\_ping(base\_ip + '{0}'.format(i))

        if i %20 ==0:

            print(".", end = "")

        if return\_codes[base\_ip + '{0}'.format(i)] == 0:

            host\_validos.append(base\_ip + '{0}'.format(i))

    print("\nMapeamento pronto...")

    return host\_validos

def detalhar\_host(host\_validos):

    """Obtendo nome do host"""

    nm = nmap.PortScanner()

    for host in host\_validos:

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r')

        try:

            nm.scan(host)

            host\_ = Host(host, nm[host].hostname())

            #print('O IP', host, 'possui o nome', nm[host].hostname())

            for proto in nm[host].all\_protocols():

                print('----------')

                print('Protocolo : %s' % proto)

            lport = nm[host][proto].keys()

            for port in lport:

                port\_ = Port(port, nm[host][proto][port]['state'])

                host\_.ports.append(port\_)

                #print ('Porta: %s\t Estado: %s' % (port, nm[host][proto][port]['state']))

        except:

            print(':rocket: Exception')

            pass

        hosts.append(host\_)

executando = False

executou = False

ips = resumoGetNewIp()

def envolucro\_detalhar\_host():

    executando = True

    print('Iniciando coleta de dados da rede:', executando)

    meu\_ip = ips[0][1]

    print('Ip que será utilizado como base', meu\_ip)

    ip\_string = meu\_ip

    ip\_lista = ip\_string.split('.')

    base\_ip = ".".join(ip\_lista[0:3]) + '.'

    print("O teste será feito na sub rede: ", base\_ip)

    hosts\_localizados = verifica\_hosts\_validos(base\_ip)

    print ("Os host válidos são: ", hosts\_localizados)

    print('Verifica nome do host\r')

    detalhar\_host(hosts\_localizados)

    executou = True

    print('Processo finalizado', executou)

#########################################################################

def info\_processos():

    pid = subprocess.Popen('cmd.exe').pid

    p = psutil.Process(pid)

    perc\_mem = '{:.2f}'.format(p.memory\_percent())

    mem = '{:.2f}'.format(p.memory\_info().rss/1024/1024)

    tituloInfo = 'Informações sobre processos:'

    textoTituloInfo = font.render(tituloInfo, 1, branco)

    tela.blit(textoTituloInfo, (100, 160))

    texto1 = 'Nome: ' + p.name()

    texto01 = font.render(texto1, 1, branco)

    tela.blit(texto01, (100,200))

    texto2 = 'Executável: ' + p.exe()

    texto02 = font.render(texto2, 1, branco)

    tela.blit(texto02, (100,220))

    texto3 = 'Tempo de criação: ' + time.ctime(p.create\_time())

    texto03 = font.render(texto3, 1, branco)

    tela.blit(texto03, (100,240))

    texto4 = 'Tempo de usuário: ' + str(p.cpu\_times().user) + 's'

    texto04 = font.render(texto4, 1, branco)

    tela.blit(texto04, (100,260))

    texto5 = 'Tempo de sistema: ' + str(p.cpu\_times().system) + 's'

    texto05 = font.render(texto5, 1, branco)

    tela.blit(texto05, (100,280))

    texto6 = 'Percentual de uso de CPU: ' + str(p.cpu\_percent(interval=1.0)) + '%'

    texto06 = font.render(texto6, 1, branco)

    tela.blit(texto06, (100,300))

    texto7 = 'Percentual de uso de memória: ' + perc\_mem + '%'

    texto07 = font.render(texto7, 1, branco)

    tela.blit(texto07, (100,320))

    texto8 = 'Uso de memória: ' + mem + 'MB'

    texto08 = font.render(texto8, 1, branco)

    tela.blit(texto08, (100,340))

    texto9 = 'Número de threads: ' + str(p.num\_threads())

    texto09 = font.render(texto9, 1, branco)

    tela.blit(texto09, (100,360))

#########################################################################

def get\_envolucro(posicao):

    if posicao == 0:

        envolucro\_dados\_cpu()

        if len(hosts) == 0 and not executando and not executou:

            envolucro\_detalhar\_host()

    elif posicao == 1:

        envolucro\_dados\_memoria()

    elif posicao == 2:

        envolucro\_dados\_disco()

    elif posicao == 3:

        mostrar\_texto\_rede()

    elif posicao == 4:

        envolucro\_arquivos()

    elif posicao == 5:

        info\_processos()

    elif posicao == 6:

        resumo()

#########################################################################

posicao\_atual = 0

#########################################################################

while not terminou:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            terminou = True

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RIGHT:

                posicao\_atual = posicao\_atual + 1

            elif event.key == pygame.K\_LEFT:

                posicao\_atual = posicao\_atual - 1

            elif event.key == pygame.K\_SPACE:

                posicao\_atual = 6

#carrossel

    if count == 60:

        tela.fill(preto)

        if posicao\_atual < 0:

            posicao\_atual = 6

        elif posicao\_atual > 6:

            posicao\_atual = 0

        get\_envolucro(posicao\_atual)

        count = 0

    pygame.display.update()

    clock.tick(60)

    count = count + 1

pygame.display.quit()

# Evidências

## Evidência TP2

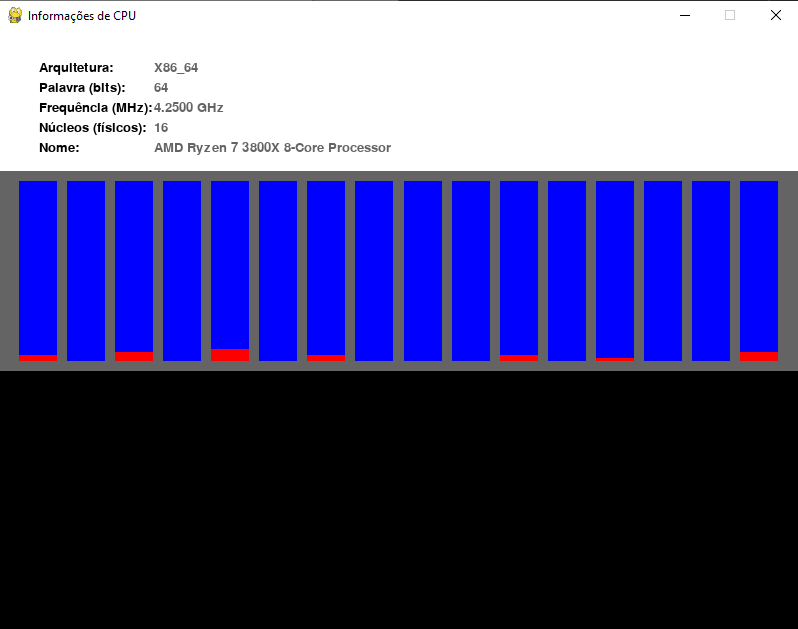


Figure 1 Processador

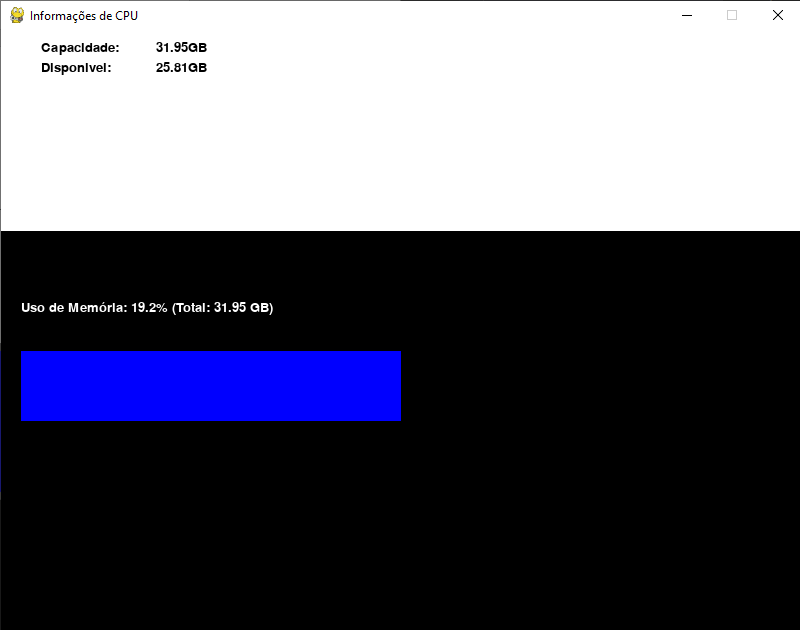


Figure 2 Memória



Figure 3 Disco



Figure 4 Rede

## Evidências TP3

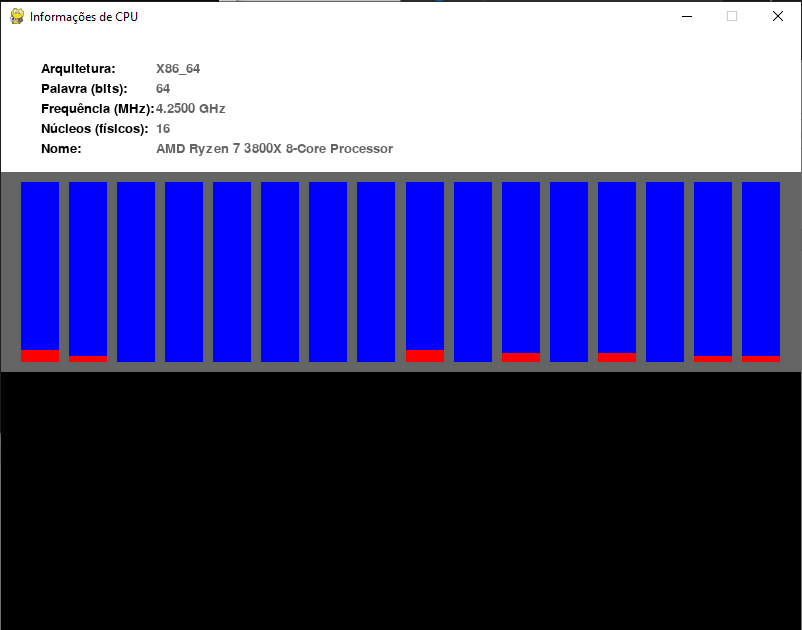


Figure 5 Processador



Figure 6 Memória



Figure 7 Disco

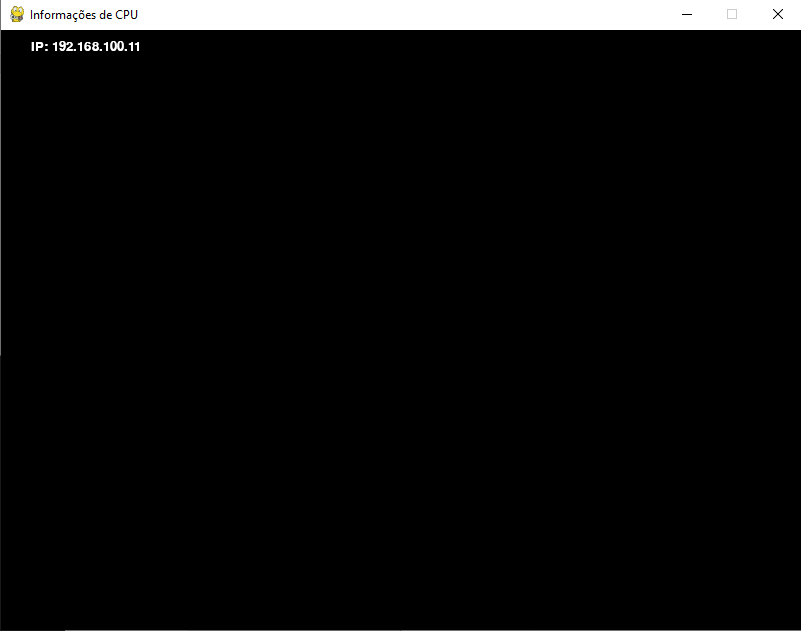


Figure 8 Rede

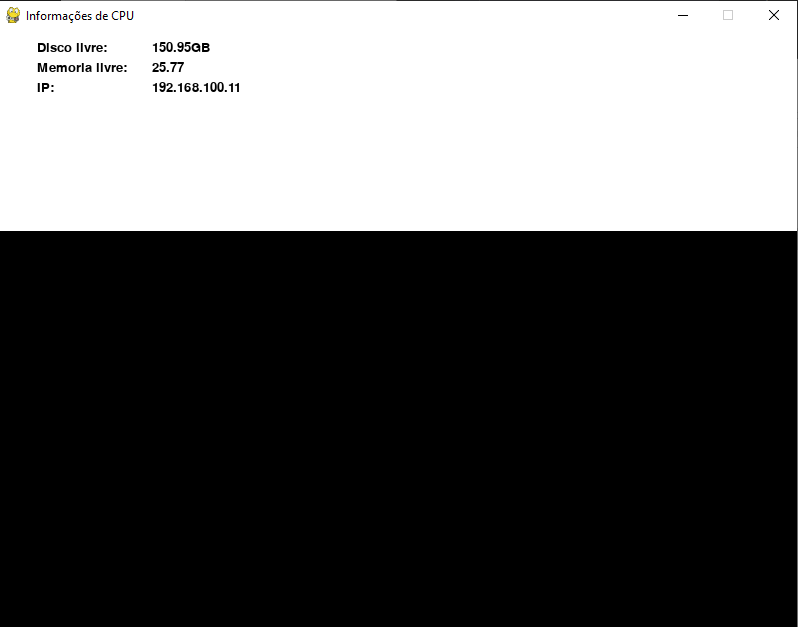


Figure 9 Resumo

## Evidências TP4

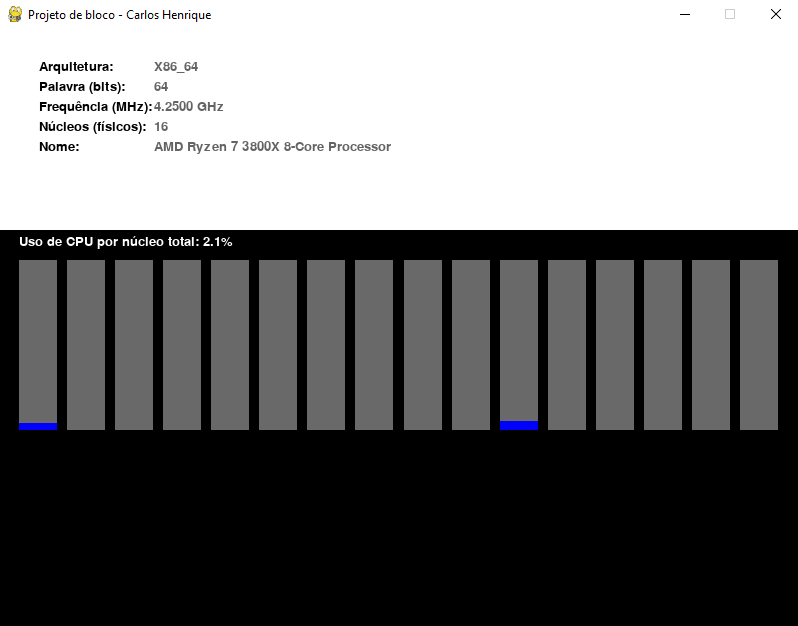


Figure 10 Processador

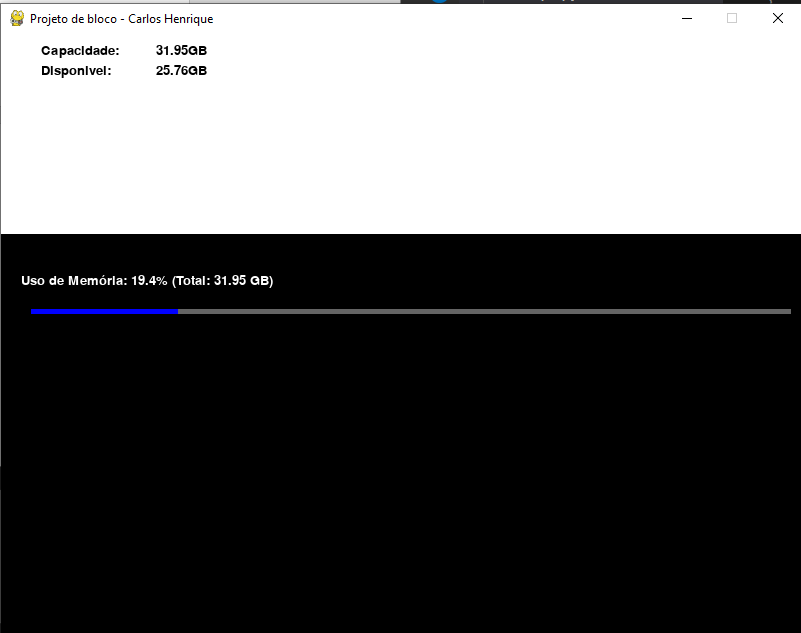


Figure 11 Memória

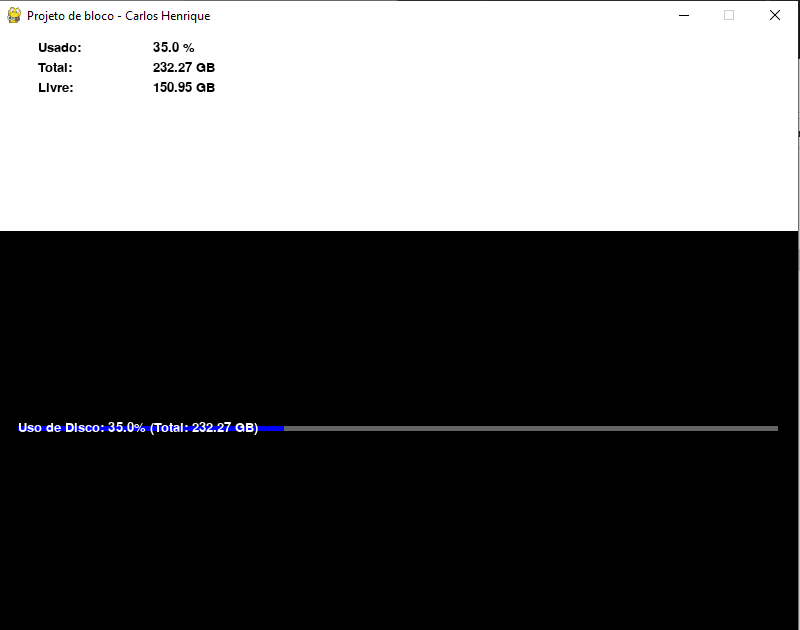


Figure 12 Disco

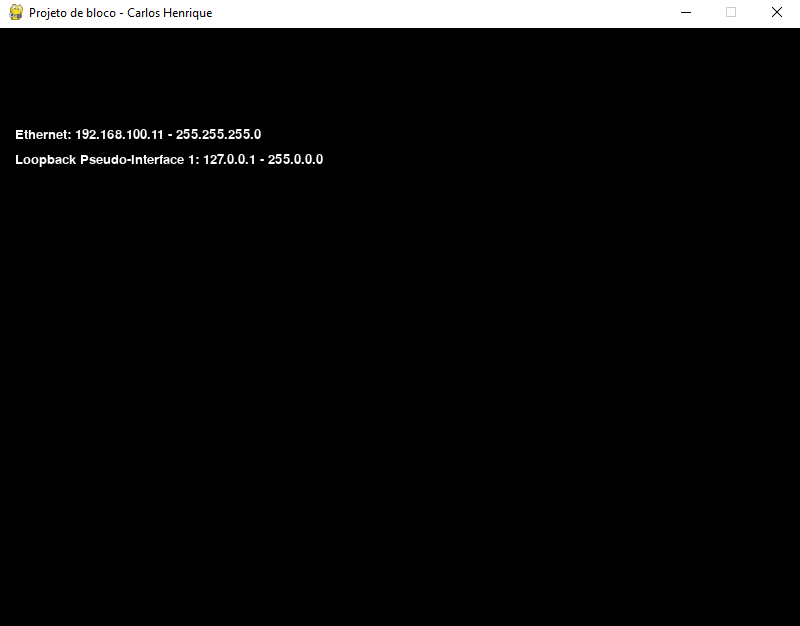


Figure 13 Rede

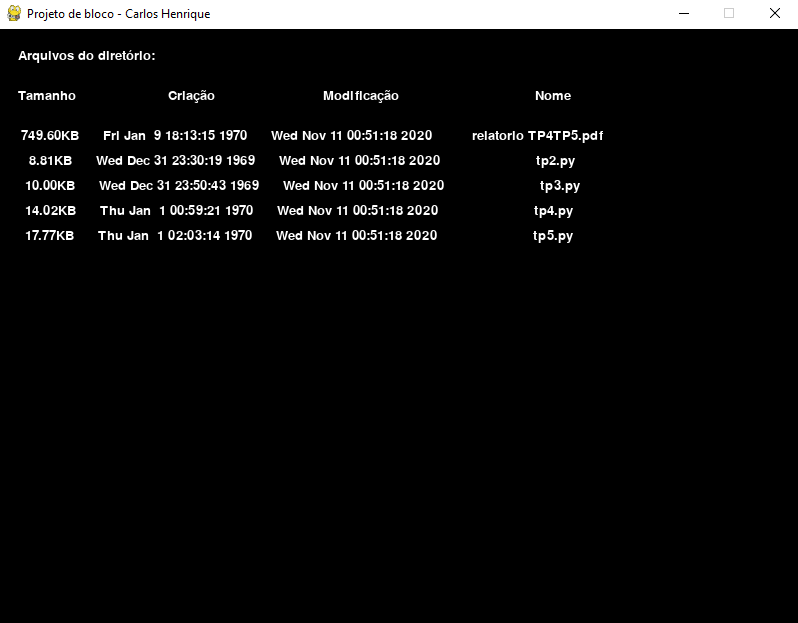


Figure 14 Diretórios

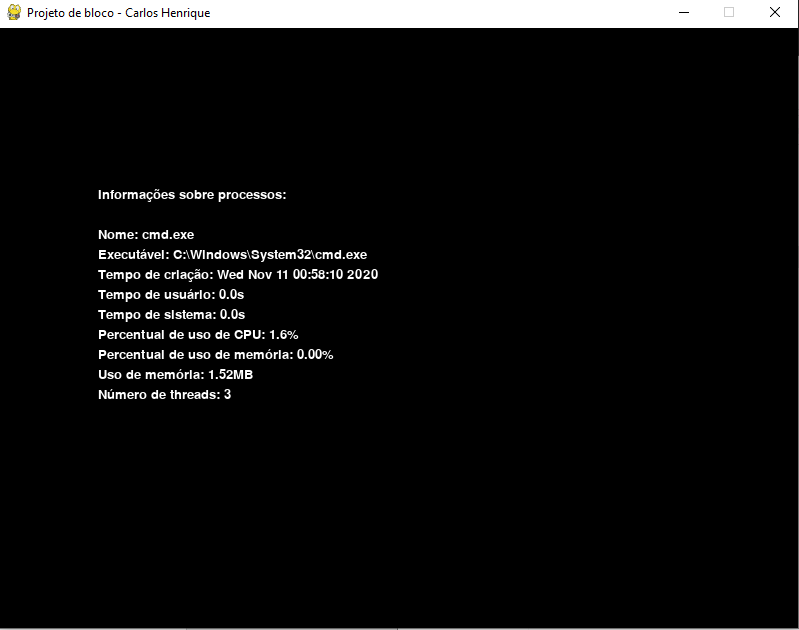


Figure 15 Processo

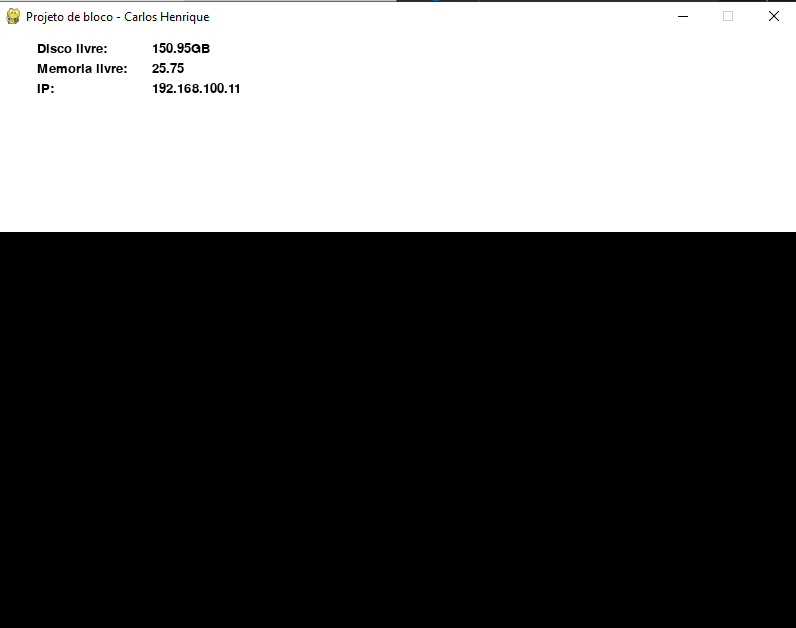


Figure 16 Resumo

## Evidências TP5

