

### TESTE DE PERFORMANCE 3



**Engenharia da Computação**  
**Projeto em Arquitetura de Computadores,**  
**Sistemas Operacionais e Redes**  
Kaio Henrique Silva da Cunha  
Prof.: Alcione Dolavale

Fortaleza, CE  
18/09/2021

## Atividade 1

A aplicação começa por importar os módulos necessários para capturar informações da arquitetura do computador e para exibir interfaces gráficas.

```
Segundo Bloco > Projeto em Arquitetura
1  import pygame
2  import psutil
3  import cpuinfo
```

Em seguida são declaradas as funções para formatar as informações da máquina, exibir informações da CPU, utilização dos núcleos, do disco e da memória, respectivamente. É importante observar que a captura da frequência(freq) me causou muitos problemas. Por algum motivo, na minha máquina virtual(Linux) ele estava retornando 'None'. Acabei decidindo por enviar com uma solução temporária, pois já estava muito atrasado na entrega. Eu precisei utilizar uma máquina virtual porque tive outros problemas de

compatibilidade na minha máquina que me impediam de usar os módulos.

```
5 # Mostra texto de acordo com uma chave:
6 def mostra_texto(s1, nome, chave, pos_y):
7     text = font.render(nome, True, preto)
8     s1.blit(text, (10, pos_y))
9     if chave == "freq":
10         freq = psutil.cpu_freq()
11         if freq:
12             s = freq.current
13         else:
14             s = "unknown"
15         #s = str(round(psutil.cpu_freq().current, 2))
16     elif chave == "nucleos":
17         s = str(psutil.cpu_count())
18         s = s + " (" + str(psutil.cpu_count(logical=False)) + ")"
19     elif chave == "ip":
20         dic_interfaces = psutil.net_if_addrs()
21         s = str(dic_interfaces['ens33'][0].address)
22
23     else:
24         s = str(info_cpu[chave])
25     text = font.render(s, True, cinza)
26     s1.blit(text, (220, pos_y))
27
28 # Obtém informações da CPU
29 info_cpu = cpuinfo.get_cpu_info()
30
31 # Mostra as informações de CPU escolhidas + IP:
32 def mostra_info_cpu():
33     s1.fill(branco)
34     mostra_texto(s1, "Nome:", "brand", 10)
35     mostra_texto(s1, "Arquitetura:", "arch", 30)
36     mostra_texto(s1, "Palavra (bits):", "bits", 50)
37     mostra_texto(s1, "Frequência (MHz):", "freq", 70)
38     mostra_texto(s1, "Núcleos (físicos):", "nucleos", 90)
39     mostra_texto(s1, "IP(ens33):", "ip", 110)
40     tela.blit(s1, (0, 0))
```

```

43 def mostra_uso_cpu():
44     capacidade = psutil.cpu_percent(interval=0)
45     larg = largura_tela #- 2*20
46     s5 = pygame.surface.Surface((larg, altura_tela/3))
47     pygame.draw.rect(s5, azul, (20, 50, larg, 70))
48     larg = larg*capacidade/100
49     s6 = pygame.surface.Surface((larg, altura_tela/3))
50     pygame.draw.rect(s6, amarelo, (20, 50, larg, 70))
51     tela.blit(s5, (0, 370))
52     tela.blit(s6, (0, 370))
53     text = font.render("Uso de CPU 1:", 1, branco)
54     tela.blit(text, (10,370))
55
56     capacidade = psutil.cpu_percent(interval=1)
57     larg = largura_tela #- 2*20
58     s7 = pygame.surface.Surface((larg, altura_tela/3))
59     pygame.draw.rect(s7, azul, (20, 50, larg, 70))
60     larg = larg*capacidade/100
61     s8 = pygame.surface.Surface((larg, altura_tela/3))
62     pygame.draw.rect(s8, amarelo, (20, 50, larg, 70))
63     tela.blit(s7,(0, 490))
64     tela.blit(s8, (0, 490))
65     text = font.render("Uso de CPU 2:", 1, branco)
66     tela.blit(text, (10, 490))
67
68 def mostra_uso_disco():
69     disco = psutil.disk_usage('.')
70     larg = largura_tela #- 2*20
71     s3 = pygame.surface.Surface((larg, altura_tela/10))
72     pygame.draw.rect(s3, azul, (20, 50, larg, 70))
73     larg = larg*disco.percent/100
74     s4 = pygame.surface.Surface((larg, altura_tela/10))
75     pygame.draw.rect(s4, vermelho, (20, 50, larg, 70))
76     tela.blit(s3, (0, 250))
77     tela.blit(s4, (0, 250))
78     total = round(disco.total/(1024*1024*1024), 2)
79     texto_barra = "Uso de Disco: (Total: " + str(total) + "GB):"
80     text = font.render(texto_barra, 1, branco)
81     tela.blit(text, (10, 255))

```

```

83 # Mostar uso de memória
84 def mostra_uso_memoria():
85     mem = psutil.virtual_memory()
86     larg = largura_tela #- 2*20
87     s1 = pygame.surface.Surface((larg, altura_tela/10))
88     pygame.draw.rect(s1, azul, (20, 50, larg, 70))
89     larg = larg*mem.percent/100
90     s2 = pygame.surface.Surface((larg, altura_tela/10))
91     pygame.draw.rect(s2, vermelho, (20, 50, larg, 70))
92     tela.blit(s1, (0, 130))
93     tela.blit(s2, (0, 130))
94     total = round(mem.total/(1024*1024*1024),2)
95     texto_barra = "Uso de Memória (Total: " + str(total) + "GB):"
96     text = font.render(texto_barra, 1, branco)
97     tela.blit(text, (10, 130))

```

Entre as linhas 100 e 122 são declaradas as variáveis que serão úteis para a aplicação como um todo, como largura e altura da tela, surfaces, cores, relógio e contador. É também onde a tela é iniciada.

```

99 # Iniciando a janela principal
100 largura_tela = 1600
101 altura_tela = 1200
102 # Superfície para mostrar as informações:
103 s1 = pygame.surface.Surface((largura_tela, altura_tela))
104 s = pygame.surface.Surface((largura_tela, altura_tela))
105 pygame.font.init()
106 font = pygame.font.Font(None, 32)
107 tela = pygame.display.set_mode((largura_tela, altura_tela))
108 pygame.display.set_caption("Uso de Recursos")
109 pygame.display.init()
110
111 # Cores:
112 preto = (0, 0, 0)
113 branco = (255, 255, 255)
114 cinza = (100, 100, 100)
115 azul = (0, 0, 255)
116 vermelho = (255, 0, 0)
117 amarelo = (255, 255, 53)
118
119 # Cria relógio
120 clock = pygame.time.Clock()
121 # Contador de tempo
122 cont = 60

```

No trecho 124-141 inicia-se um loop onde eventos são captados para terminar a aplicação ou não. É também onde as funções são chamadas e o relógio é utilizado para atualizar a tela e exibir o uso dos recursos variando.

## **Atividade 2**

Os processadores são um mundo à parte dentro do universo da computação. Eles podem ser single-core ou multi-core. Essa característica indica a quantidade de núcleos de processamento, que pode ir de 1 até 32. Quanto maior a quantidade, maior a capacidade de realizar tarefas ao mesmo tempo.

Eles podem ser também de 32 ou 64 bits. Isso diz respeito à capacidade de processamento. Os de 64 aproveitam capacidades maiores de memória RAM. As duas empresas mais conhecidas na fabricação de processadores são Intel e AMD. Foi a Intel, inclusive, que criou o recurso de hyperthreading, que simula, em cada núcleo, 2 cores diferentes.

## **Atividade 3**

Uma palavra é basicamente uma sequência de bits que são processados em conjunto.

## **Atividade 4**

É a diferença entre um núcleo físico, real, e um núcleo virtual, emulado pelo próprio processador para auxiliar os físicos. Como dito anteriormente, estão presentes na linha Core da Intel. Ex: i3, i5 e i7.

## **Referências**

Material da disciplina no Moodle e gravações das aulas. Etapa 3. Acesso em set. 2021.