

Tarefa 10 - Computação Distribuída

Aluno: Rafael Amauri Diniz Augusto // 651047

Implemente uma simulação de X processos e um detector de falhas com consenso, de maneira simplificada. Mostre um exemplo de execução, simulando que um dos processos falhou e a falha foi detectada. Mostre o consenso final. Lembre-se da síndrome que indica o estado dos processos (veja o material sobre falhas).

Meu código:

```
import numpy as np

class Processo:

    def __init__(self, id):

        self.id = id

        self.received_heartbeat = False

        self.is_timedout = False

    def __eq__(self, p2):

        return self.id == p2.id

    def has_received_heartbeat(self):

        return self.received_heartbeat

    def receive_heartbeat(self, value):

        self.received_heartbeat = value

    def send_heartbeat(self, p):

        # Simulando que o processo 2 está falho

        if p.id != 2:

            p.received_heartbeat = True
```

```

def has_timedout(self):

    return self.is_timedout

def update_timeout(self):

    self.is_timedout = False

def compute_timeout(self):

    pass

def main():

    suspeitos = np.empty(0, dtype = Processo)

    falhos = np.empty(0, dtype = np.int32)

    n_processos = 4

    '''

    Estamos simulando que, dos 4 processos, o de ID 2 está falho!

    '''

    for id in range(n_processos):

        suspeitos = np.append(suspeitos, Processo(id))

    for i in suspeitos:

        for j in suspeitos:

            j.received_heartbeat = False

            j.is_timedout = False

        for j in suspeitos:

            if i == j:

```

```
        continue

    j.compute_timeout()

    i.send_heartbeat(j)

    if j.has_received_heartbeat():

        j.update_timeout()

    else:

        j.is_timedout = True

    if j.has_timedout():

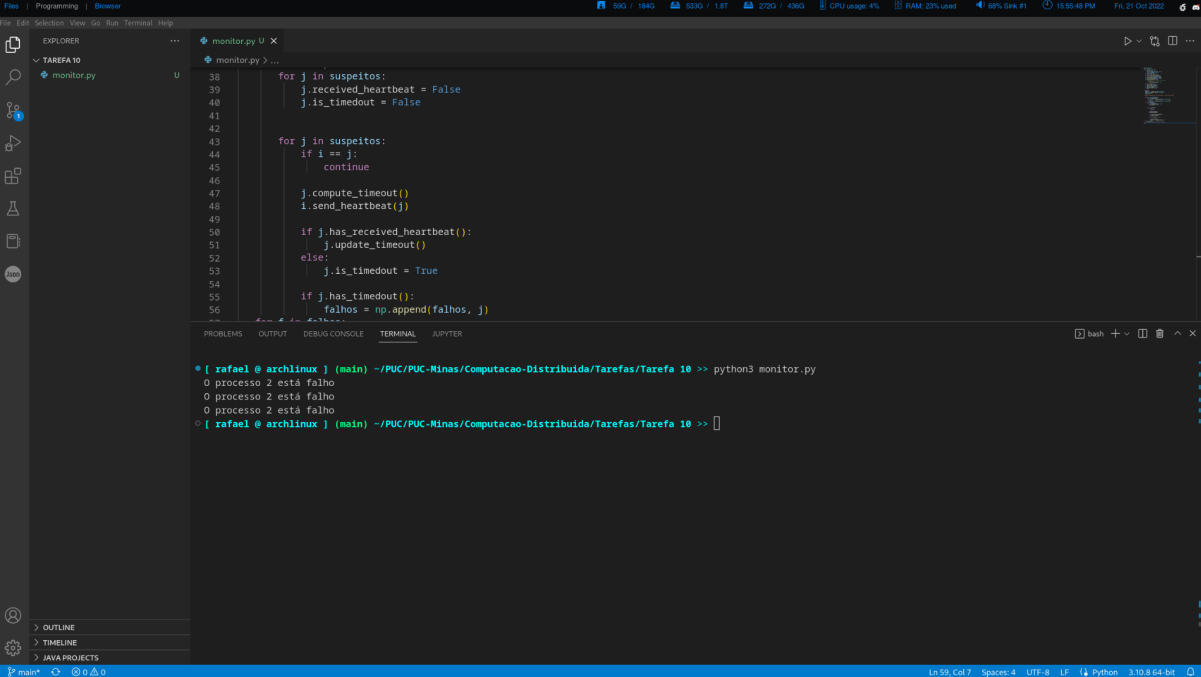
        falhos = np.append(falhos, j)

for f in falhos:

    print(f"O processo {f.id} está falho")

main()
```

Resultado após rodar:



The screenshot shows a Visual Studio Code editor window with a Python file named `monitor.py` open. The file contains a loop that checks the status of four processes (IDs 1, 2, 3, and 4). The terminal output shows the command `python3 monitor.py` being executed, and the output indicates that all four processes are running correctly.

```
38         for j in suspeitos:
39             j.received_heartbeat = False
40             j.is_timedout = False
41
42     for j in suspeitos:
43         if i == j:
44             continue
45
46         j.compute_timeout()
47         i.send_heartbeat(j)
48
49         if j.has_received_heartbeat():
50             j.update_timeout()
51         else:
52             j.is_timedout = True
53
54         if j.has_timedout():
55             falhos = np.append(falhos, j)
56
```

```
*[ rafaél @ archlinux ] (main) ~/PUC/PUC-Minas/Computacao-Distribuida/Tarefas/Tarefa 10 >> python3 monitor.py
0 processo 2 está falho
0 processo 2 está falho
0 processo 2 está falho
0 processo 2 está falho
*[ rafaél @ archlinux ] (main) ~/PUC/PUC-Minas/Computacao-Distribuida/Tarefas/Tarefa 10 >> ]
```

Como pode ser visto, temos 4 processos e todos detectaram que o processo de ID 2 está falho.