IPS Instituto Politécnico de Setúbal Escola Superior de Tecnologia de Setúbal	COMPUTAÇÃO PARALELA E DISTRIBUÍDA 2022/2023
	Laboratório 04: Utilização de Processos

Objetivos do laboratório

Pretende-se que o aluno:

1. Compreenda a comunicação entre os processos utilizando “pipes” e “queues"
2. Perceba o conceito de processo
3. Perceba a comunicação entre processos
4. Utilize multiprocessamento para obter termos da série de Fibonacci

Nível 1

Escreva o código apresentado e responda as questões colocadas.

```
from multiprocessing import Process, Queue, Pipe
import os
import random


def funcao_comunicacao_queue(q):
    print(f"\tEstou na função funcao_comunicacao_queue e sou o processo {os.getpid()}")
    valor = random.randint(1, 10)
    q.put(valor)


def funcao_comunicacao_pipe(conn):
    print(f"\tEstou na função funcao_comunicacao_pipe e sou o processo {os.getpid()}")
    conn.send(['Olá', 'mundo', 'abril', 2022])
    conn.close()


if __name__ == '__main__':
    print(f"processo principal {os.getpid()}")
    # utilizando queue
    q = Queue()
    processo_1 = Process(target=funcao_comunicacao_queue, args=(q,))
    processo_1.start()
    valor = q.get()
    print(f"valor recebido: {valor}")
    processo_1.join()

    # utilizando pipe
    conn_pai, conn_filho = Pipe()
    processo_2 = Process(target=funcao_comunicacao_pipe, args=(conn_filho,))
    processo_2.start()
    valor = conn_pai.recv()
    print(f"valor recebido: {valor}")
    processo_2.join()
```

- Diga o que faz cada uma das seguintes instruções:
    - q = Queue()
    - q.put(valor)
    - valor = q.get()
    - conn_pai, conn_filho = Pipe()
    - conn.send(['Olá', 'mundo', 'abril', 2022])
    - conn.recv()
    - conn.close()

Nível 2

Escreva um programa que crie dois processos produtores (**produtor_pipe** e **produtor_queue**) e dois processos consumidores (**consumidor_pipe** e **consumidor_queue**). O processo **produtor_pipe** deverá enviar os nomes dos alunos do grupo para o processo **consumidor_pipe** utilizando comunicação com pipe. O processo produtor_queue deverá enviar os nomes dos alunos do grupo para o processo **consumidor_queue** utilizando comunicação com queue. Os processos consumidores deverão apresentar a informação recebida.

O output do programa deverá apresentar o seguinte formato:

```
processo (MainProcess)
Comunicação com Pipe
        ['primeiro nome', 'segundo nome'] enviado pelo subprocesso produtor_pipe (Process-1)
        ['primeiro nome', 'segundo nome'] recebido pelo subprocesso consumidor_pipe (Process-2)
Comunicação com Queue
        ['primeiro nome', 'segundo nome'] enviado pelo subprocesso produtor_queue (Process-3)
        ['primeiro nome', 'segundo nome'] recebido pelo subprocesso consumidor_pipe (Process-4)
```

Nível 3

Escreva o código apresentado e responda as questões colocadas.

```
import time

def fibonacci(n):
    a, b = 0, 1
    for item in range(n):
        a, b = b, a + b
    return a

def preencher_dicionario_fibonacci_v1(dicionario, qtd):
    for i in range(qtd):
        dicionario[i+1] = fibonacci(i+1)

def preencher_dicionario_fibonacci_v2(dicionario, qtd):
    for i in range(qtd):
        if i < 2:
            dicionario[i+1] = 1
        else:
            dicionario[i+1] = dicionario[i] + dicionario[i-1]

def mostrar_dicionario(dicionario):
    for n in dicionario.keys():
        print(f'Fib({n}) = {dicionario[n]}')

if __name__ == '__main__':
    qtd_valores = 10
    inicio = time.time()
    dicionario_fibonacci = {}
    preencher_dicionario_fibonacci_v1(dicionario_fibonacci, qtd_valores)
    tempo = time.time() - inicio
    print(f'{tempo:.10f}s para calcular os primeiros {qtd_valores} termos da série de Fibonacci de forma
sequencial')
    mostrar_dicionario(dicionario_fibonacci)
    inicio = time.time()
    dicionario_fibonacci = {}
    preencher_dicionario_fibonacci_v2(dicionario_fibonacci, qtd_valores)
    tempo = time.time() - inicio
    print(f'{tempo:.10f}s para calcular os primeiros {qtd_valores} termos da série de Fibonacci de forma
sequencial')
    mostrar_dicionario(dicionario_fibonacci)
```

- Diga o que faz o programa.
- O que é um dicionário em **Phyton**?
- Coloque como comentário as instruções **mostrar_dicionario(dicionario_fibonacci)**, altere para 10000 o valor da **qtd_valores** e execute o programa. Explique a diferença de desempenho observada.

Nível 4

Escreva o código apresentado e responda as questões colocadas.

```
import time
from multiprocessing import Process, Queue, current_process, Manager

def fibonacci(n):
    a, b = 0, 1
    for item in range(n):
        a, b = b, a + b
    return a

def mostrar_dicionario(dicionario):
    for n in dicionario.keys():
        print(f"Fib({n}) = {dicionario[n]}")

def gerar_chaves(fibo_dict, qtd):
    for i in range(qtd):
        fibo_dict[i+1] = None

def calcular_valores(q, fibo_dict):
    cont = 0
    while not q.empty():
        n = q.get()
        if n:
            fibo_dict[n] = fibonacci(n)
            cont = cont + 1
    print(f"{current_process().name} - calculou {cont} termo(s) da série")

def calcular_valores_pool(fibo_dict, n):
    # print(f"Processo {current_process().name} a calcular Fib({n})")
    fibo_dict[n] = fibonacci(n)

if __name__ == '__main__':
    qtd_valores =  20
    numero_de_processos = 4
    inicio = time.time()
    manager = Manager()
    dicionario_fibonacci_par = manager.dict()
    gerar_chaves(dicionario_fibonacci_par, qtd_valores)
    fila_de_valores_fibonacci = Queue()
    # print(dicionario_fibonacci_par.keys())
    for k in dicionario_fibonacci_par.keys():
        fila_de_valores_fibonacci.put(k)
    lista_de_processos = []
    for _ in range(numero_de_processos):
        p = Process(target=calcular_valores, args=(fila_de_valores_fibonacci, dicionario_fibonacci_par))
        p.start()
        lista_de_processos.append(p)
    [p.join() for p in lista_de_processos]
    fim = time.time()
    print(f"Tempo utilizado para calcular os primeiros {qtd_valores} termos da série de Fibonacci
utilizando "
          f" multiprocessamento {fim - inicio:.10f}s")
    mostrar_dicionario(dicionario_fibonacci_par)
```

- Diga o que faz cada uma das seguintes instruções:
    - **manager = Manager()**
    - **dicionario_fibonacci_par = manager.dict()**
    - **gerar_chaves(dicionario_fibonacci_par, qtd_valores)**
    - **[p.join() for p in lista_de_processos]**
- Coloque como comentário as instruções **mostrar_dicionario(dicionario_fibonacci),** altere para 10000 o valor da **qtd_valores** e execute o programa.
- Realize uma análise critica do trabalho realizado por cada um dos processos criados relacionando com a quantidade de termos a calcular e com a utilização da "**queue**".

Nível 5

Altere o programa do nível 4 implementando o modo mais eficiente de obter os termos da sequência de **Fibonacci** observado no nível 3.