

MANUAL DE UTILIZAÇÃO E COMPILAÇÃO

BSB Compute — Sistema de Orquestração e Escalonamento

Davi Ribeiro Nobre - 22401443
Gabriella Lourenço - 22405748
Lara Alves Korogi - 22400272
Larissa Alves Neves - 22511353

1. Introdução

Este manual descreve detalhadamente como instalar dependências, executar o sistema de orquestração, selecionar algoritmos, analisar resultados e utilizar os scripts auxiliares incluídos no projeto.

Todo o sistema foi desenvolvido em Python, utilizando arquivos JSON como entrada, scripts modulares e ferramentas modernas de gerenciamento de ambiente.

2. Requisitos do Sistema

Para rodar o projeto corretamente, são necessários:

2.1 Linguagem

- Python 3.10+

2.2 Ferramentas recomendadas

- UV (gerenciador de dependências)
- Terminal Linux ou Windows com WSL
- Editor como VSCode (opcional)

2.3 Estrutura de Diretórios

O projeto deve manter a seguinte organização:

/projeto

| — launcher.py

| — main.py

| — comparador.py

| — config.json

| — metricas.json

| — resultados/

| — pyproject.toml

| — uv.lock

3. Preparação do Ambiente

Antes da primeira execução, é necessário configurar o ambiente Python e instalar as dependências descritas no arquivo `pyproject.toml`.

3.1 Instalação do UV

O UV é responsável por gerenciar ambientes virtuais e dependências do projeto.

- Linux/macOS: `curl -LsSf https://astral.sh/uv/install.sh | sh`
- Windows (PowerShell) : `irm https://astral.sh/uv/install.ps1 | iex`

3.2 Instalação das Dependências

No diretório raiz do projeto: `uv sync`

Este comando irá:

- criar um ambiente virtual isolado
- instalar bibliotecas necessárias
- preparar o sistema para execução

4. Arquivo de Configuração (config.json)

O funcionamento do sistema está centralizado no arquivo `config.json`, que define os parâmetros da simulação, as tarefas a serem processadas e o algoritmo escolhido.

```
{  
    "algoritmo": "sjf",  
    "tarefas": [  
        { "id": 1, "tempo": 8, "prioridade": 1 },  
        { "id": 2, "tempo": 3, "prioridade": 3 }  
    ]  
}
```

O JSON permite definir:

- o algoritmo de escalonamento
- a quantidade de tarefas
- o tempo de execução de cada tarefa
- prioridades (quando aplicável)
- parâmetros adicionais da simulação

5. Execução do Sistema

O fluxo operacional é iniciado pelo arquivo `launcher.py`, que interpreta o arquivo de configuração, prepara o ambiente e aciona `main.py`.

5.1 Execução Padrão

Com o ambiente preparado: `uv run launcher.py`

6. Seleção de Algoritmos

O sistema suporta três algoritmos clássicos de escalonamento:

- **RR — Round Robin**
- **SJF — Shortest Job First**
- **Prioridade**

A seleção pode ser realizada diretamente na execução:

Round Robin: `uv run launcher.py --alg rr`

Shortest Job First: `uv run launcher.py --alg sjf`

Por Prioridade: `uv run launcher.py --alg prioridade`

7. Funcionamento da Simulação

Durante a execução, o sistema:

1. lê o arquivo de configuração;
2. carrega as tarefas;
3. organiza as filas de acordo com o algoritmo selecionado;
4. simula o tempo de execução das tarefas (`time.sleep()`);
5. registra o tempo total de execução;
6. gerar métricas detalhadas;
7. salva os resultados no diretório `resultados/`.

O usuário pode acompanhar o andamento por meio de mensagens exibidas no terminal.

8. Interpretação dos Resultados

Ao final de cada simulação, dois tipos de arquivos são gerados:

8.1 metricas.json

Arquivo consolidado contendo:

- algoritmo utilizado
- tempo médio de execução
- tempo total de simulação
- quantidade de tarefas processadas
- ordem real de execução

8.2 Pasta resultados/

Contém arquivos adicionais para cada execução, permitindo análises posteriores.

9. Comparação de Algoritmos

O arquivo `comparador.py` permite realizar análises comparativas entre execuções de algoritmos diferentes.

Para rodar o comparador:`uv run comparador.py`

Ele produzirá um relatório comparando:

- tempo médio de cada algoritmo
- eficiência
- impacto das prioridades
- comportamento sob diferentes cargas

10. Processo Completo de Utilização

1. Instalar o UV
2. Rodar `uv sync`
3. Configurar `config.json`
4. Executar o algoritmo desejado com `launcher.py`
5. Verificar resultados em `metricas.json` e `resultados/`
6. Executar `comparador.py` para analisar desempenhos

11. Encerramento

Este manual fornece todas as informações necessárias para executar o sistema de orquestração, compreender sua saída e alternar entre diferentes algoritmos de escalonamento. O ambiente em Python torna o sistema acessível, extensível e adequado para fins acadêmicos, permitindo a realização de experimentos e comparações de forma simples e estruturada.

