Sistemas Operacionais

Relatório do Laboratório 4 - Escalonamento

Introdução

Relatório de desempenho dos algoritmos de escalonamento de CPU:

- Prioridade
- *First Come, First Served*
- *Shortest Job First*
- *Round Robin*

Foram implementadas também as versões preemptivas dos algoritmos, quando aplicável.

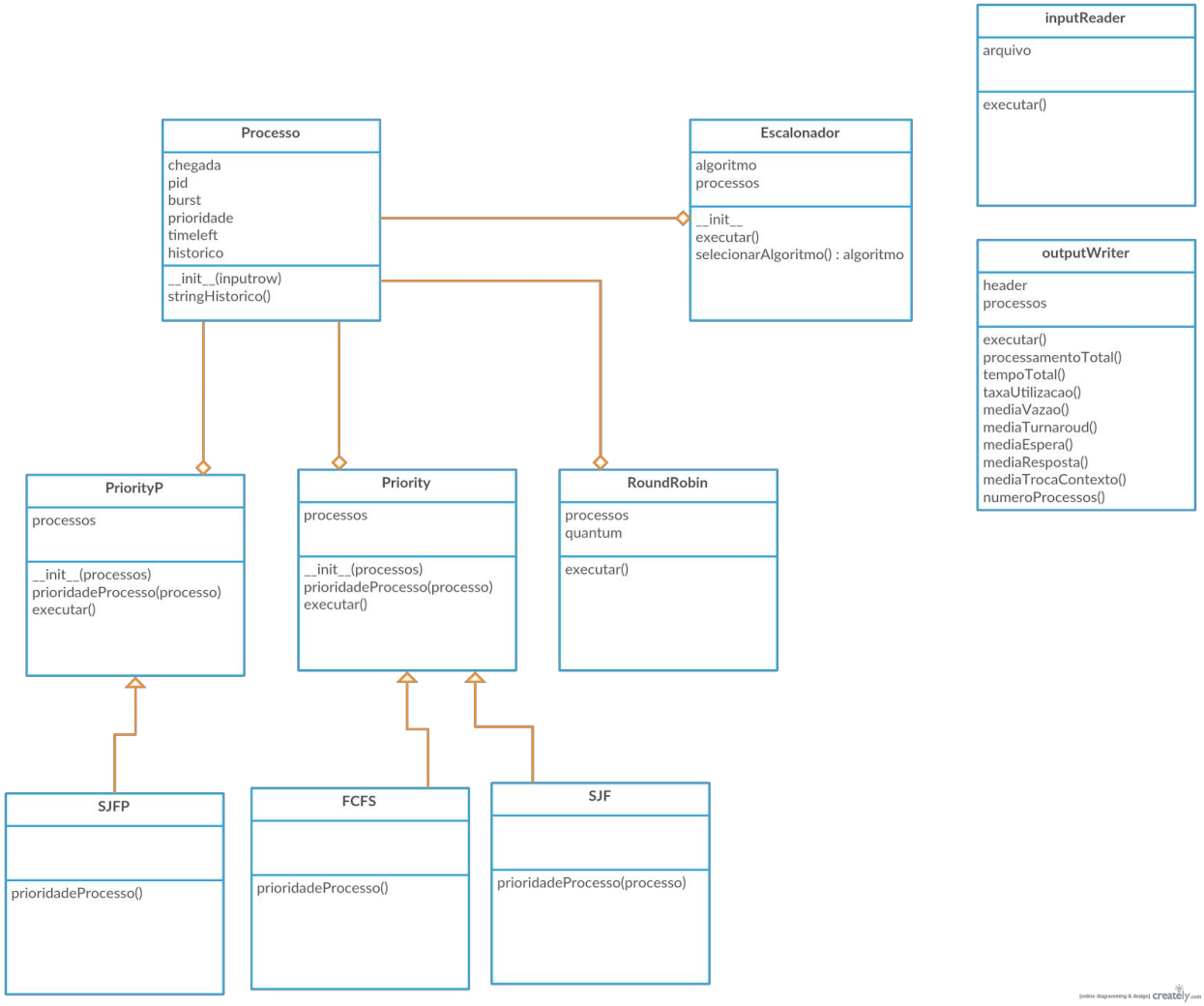
Foi utilizada a linguagem de programação Python por sua sintaxe simples e por ser orientada a objetos, o que facilitou a organização do programa.

Estrutura e funcionamento

A estrutura do programa é definida no diagrama de classes a seguir. Cada algoritmo foi implementado em uma classe própria, além disso observa-se que houve o uso da herança na definição de algumas das classes. Isso se deve ao fato de que os algoritmos FCFS (*First Come First Served*), SJF (*Shortest Job First*) e SJFP (*Shortest Job First Preemptivo*) são casos especiais dos algoritmos de Prioridade, em que as prioridades de cada processo assumem valores específicos. No caso do FCFS, todos os processos possuem a mesma prioridade, e no caso do SJF e do SJFP, a prioridade de um processo é definida como o tempo restante para que ele termine de executar.

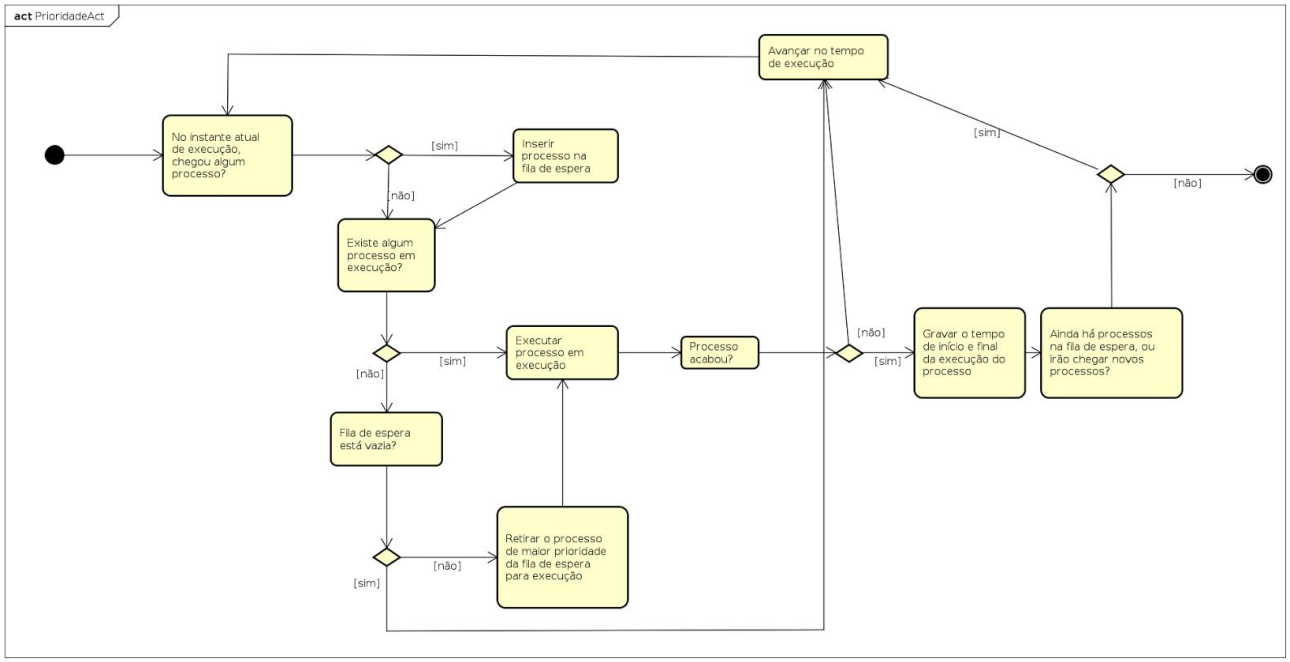
Além das classes de algoritmos, definimos também uma classe que representa um processo. Além dos atributos que definem um processo , como PID, prioridade, tempo de chegada e tempo restante, cada processo possui um histórico, que grava os intervalos de tempo nos quais ele esteve em execução, e é usado para obter as estatísticas do processo.

Outras classes definidas no programa são a classe InputReader, que lê o arquivo de entrada e retorna os valores lidos, a classe OutputWriter, que gera e grava estatísticas a partir dos históricos de um conjunto de processos, e a classe Escalonador, que age como classe principal do programa.

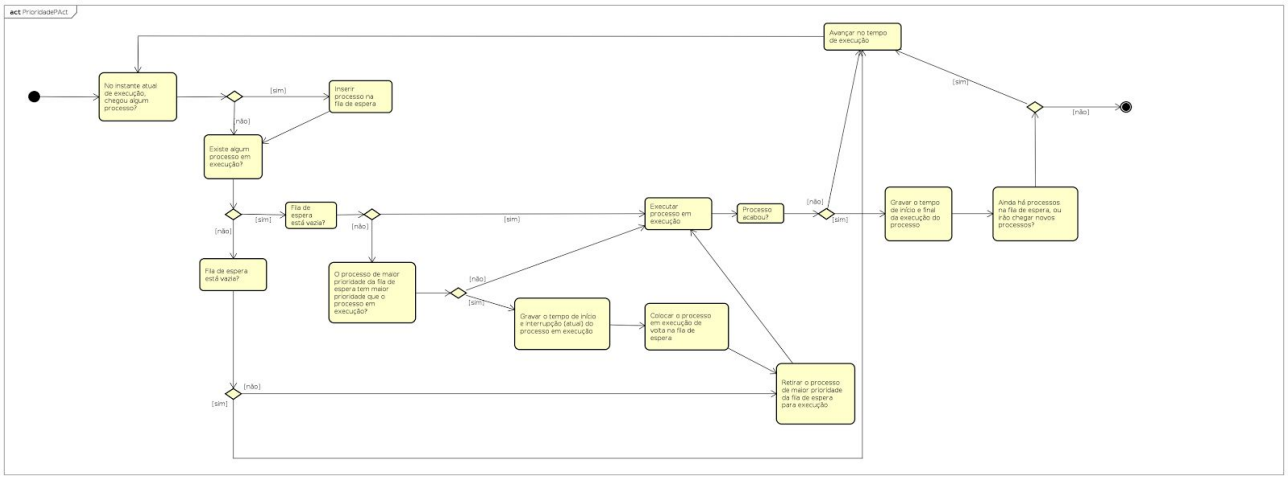


O funcionamento dos algoritmos é definido nos seguintes diagramas de atividade. Eles foram implementados na função *executar* de suas respectivas classes.

1. Diagrama de atividades do algoritmo Priority (aplica-se também ao FCFS e SJF)[1]

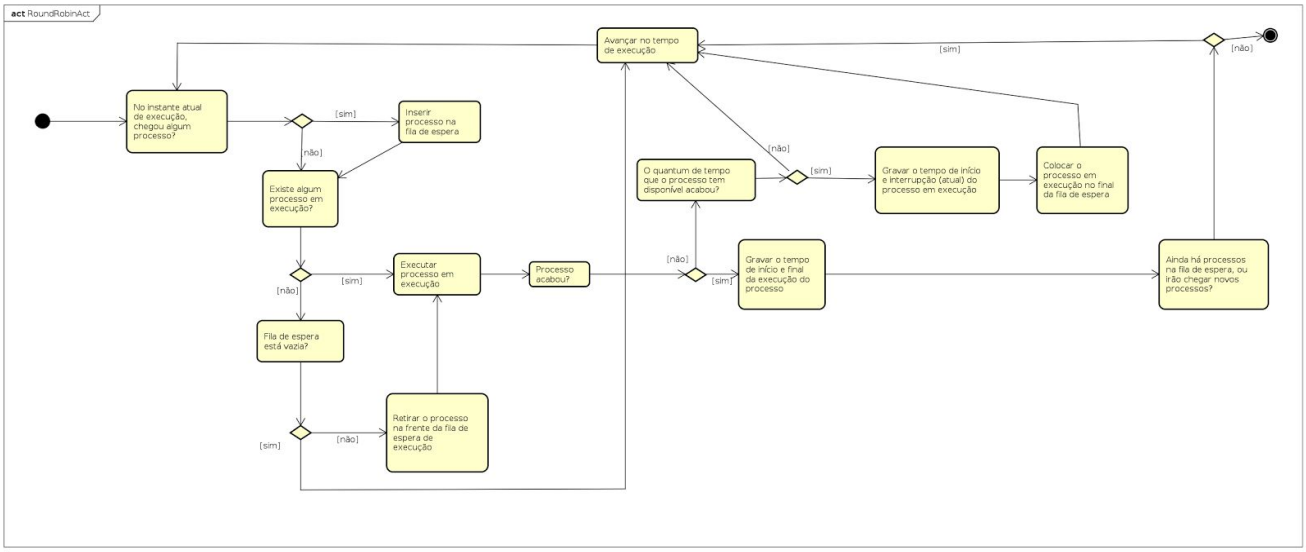


2. Diagrama de atividades do algoritmo PriorityP (aplica-se também ao SJFP)[2]



[1] Imagem original
[2] Imagem original

3. Diagrama de atividades do algoritmo Round Robin[3]



Análise

Ao executar os algoritmos com os testes criados e encontrados na pasta “tests”, alguns fatos foram percebidos e são detalhados a seguir.

O que mais chamou atenção inicialmente foi o fato de que o tempo total de processamento, o percentual de utilização da CPU, a média de *throughput* e o número de processos sempre apresentam o mesmo valor independente do algoritmo utilizado. A razão para isso é de que a soma de *bursts* dos processos é sempre a mesma, e os momentos em que a CPU fica ociosa ocorrem somente quando não há mais processos na fila de execução e por fim, o throughput é igual pois todos os processos são sempre executados após a mesma quantidade de tempo.

Comparando os algoritmos de prioridade e SJF com suas versões preemptivas, nota-se que as versões preemptivas apresentam menor média de espera, de resposta e de *turnaround*. Isso ocorre pois as versões preemptivas evitam que alguns processos fiquem na fila de execução após serem solicitados, o que diminui o tempo de espera, de resposta e portanto o de *turnaround* desses.

[3] Imagem original

Nem sempre o algoritmo de prioridade preemptivo apresenta resultados melhores que o algoritmo de prioridade normal, o que foi observado em dois casos de teste. Em um deles os resultados são iguais (Exemplo 1), pois os processos tem a mesma prioridade (assim como no algoritmo FCFS). No outro caso de teste há um processo de maior prioridade mas que também possui um tempo de burst muito maior que o dos outros processos, portanto os processos menores passam mais tempo esperando na fila de execução (Exemplo 2).

Exemplo 1:

```
========================================================================
Parâmetros: escalonador.py tests/teste3.txt priority
========================================================================
Tempo total de processamento:                                      53
Percentual de utilizacao da CPU:                                96.36%
Media de throughput:                                             0.15
Media de turnaround:                                            21.25
Media de espera:                                                14.62
Media de resposta:                                              14.62
Media de trocas de contexto:                                     0.00
Numero de processos:                                                8
========================================================================
Parâmetros: escalonador.py tests/teste3.txt priorityp
========================================================================
Tempo total de processamento:                                      53
Percentual de utilizacao da CPU:                                96.36%
Media de throughput:                                             0.15
Media de turnaround:                                            21.25
Media de espera:                                                14.62
Media de resposta:                                              14.62
Media de trocas de contexto:                                     0.00
Numero de processos:                                                8
```

Exemplo 2:

```
========================================================================
Parâmetros: escalonador.py tests/teste6.txt priority
========================================================================
Tempo total de processamento:                                      62
Percentual de utilizacao da CPU:                               100.00%
Media de throughput:                                             0.06
Media de turnaround:                                            44.00
Media de espera:                                                28.50
Media de resposta:                                              28.50
Media de trocas de contexto:                                     0.00
Numero de processos:                                                4
========================================================================
Parâmetros: escalonador.py tests/teste6.txt priorityp
========================================================================
Tempo total de processamento:                                     112
Percentual de utilizacao da CPU:                               100.00%
Media de throughput:                                             0.04
Media de turnaround:                                           106.00
Media de espera:                                                78.00
Media de resposta:                                              53.00
```

```
Media de trocas de contexto:                    0.25
Numero de processos:                              4
```

Quanto ao número de trocas de contexto, o algoritmo *Round Robin* na maioria dos casos de teste foi o que as usou em maior quantidade. Isso é explicado pois a lógica desse algoritmo é de separar o tempo dedicado à execução dos processos em intervalos definidos, o que faz com que eles sejam interrompidos muitas vezes. Os algoritmos com menores quantidades de trocas de contexto foram o FCFS, o SJF e o de prioridade. Isso é esperado pois esses algoritmos nunca interrompem seus processos. Um exemplo de um caso de teste em que o Round Robin precisa de mais trocas de contexto que os outros algoritmos é o demonstrado pela tabela abaixo.

Algoritmo	**Média de trocas de contexto**
FCFS	0
Prioridade	0
Prioridade Preemptivo	0.14
JSF	0
JSF Preemptivo	0.29
Round Robin	1.86

Equipe

- *André Novais - 344082*
  - *Implementação dos algoritmos SJF e FCFS*
  - *Documentação do código e tratamento de exceções*
- *Carolina Herbster - 354044*
  - *Projeto da hierarquia de classes do sistema.*
  - *Diagramas de classe e de atividades.*
  - *Implementação do PriorityP, RoundRobin, FCFS.*
  - *Escrita do relatório.*
- *Heitor Oliveira - 354065*
  - *Implementação do algoritmo Priority.*
  - *Implementação das funções de cálculo de desempenho, entrada e saída.*
  - *Escrita e revisão do relatório.*
- *Mariana Fontenele Lopes - 354081*
  - *Testes de correção.*

- *Escrita do relatório.*
- *Análise de desempenho dos algoritmos.*