Sistemas Operacionais
Trabalho I: Escalonador

Angelo Calebe, Gabriel Weich

[1]Escola Politécnica - Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Avenida Ipiranga, 6681 – 90619-900 – Porto Alegre – RS – Brasil

{angelo.calebe,gabriel.weich}@acad.pucrs.br

Abstract. *This article describes and documents the behavior of the process scheduling simulator developed for the first work of Operating Systems in the second semester of 2018, which consists of creating a process scheduling simulator.*

Resumo. *Este relatório descreve e documenta o funcionamento do simulador de escalonamento de processos desenvolvido para o trabalho 1 de Sistemas Operacionais, no segundo semestre de 2018, no qual consiste em criar um simulador de escalonamento de processos.*

1. Introdução

Este artigo descreve a solução proposta para o primeiro trabalho da disciplina de Sistemas Operacionais, o qual nos desafia a construir um programa que simule o algoritmo de escalonamento *Round-Robin*.

Sempre que a CPU estiver desocupada, o sistema operacional deve selecionar um dos processos que estão aguardando para serem executados e alocar no processador. A política utilizada para selecionar tal processo será abordada no decorrer do artigo, juntamente com o algoritmo capaz de simular a execução da mesma.

2. O escalonador

O escalonador pode tomar alguma ação nas seguintes circunstâncias: [Abraham Silberschatz and Gagne 2012]

- Quando um processo troca do estado executando para o estado de bloqueado, para, por exemplo, realizar uma operação de E/S.
- Quando um processo troca do estado executando para o estado de pronto para executar (por exemplo, quando uma interrupção ocorre).
- Quando um processo troca do estado de bloqueado para o estado de pronto para executar (por exemplo, ao completar E/S).
- Quando um processo finaliza.

Para as situações 1 e 4 não há escolha em termos de escalonamento, um novo processo é escolhido para executar.

O algoritmo *Round-Robin* é chamado preemptivo, pois um processo pode ser interrompido durante sua execução, quando, por exemplo, um segundo processo de maior prioridade estiver pronto para executar. Interrupções desse tipo são a causa do item 2.

O terceiro item acontece quando um processo termina uma operação de entrada e saída, nesse caso ele é recolocado na fila de prontos para executar.

Outra característica do escalonador é interromper o processo que estiver executando após uma dada fatia de tempo (*quantum*). Neste caso, o processo interrompido é colocado na fila de prontos para executar e o processo dessa fila que tiver maior prioridade é selecionado para executar.

3. Implementação

O simulador foi codificado usando a linguagem *Python 3.7*.

Foram criadas as seguintes classes:

- **Process:** Representa um processo. Possui índice, tempo de chegada, tempo de execução, prioridade e tempos de operação de entrada e saída (I/O);
- **CPU:** Representa o processador. Um processo pode ser alocado nele. Também possui um relógio incrementado a cada ciclo de execução.
- **Scheduler:** Representa o escalonador. Possui uma lista de processos, que são carregados a partir do arquivo de entrada, uma fila de processos prontos para executar e uma lista de processos bloqueados. O processador é inicializado aqui.

Parâmetros e regras do escalonador:

- **Tempo de uma operação de E/S:** 4
- **Algoritmo:** *Round Robin* com prioridade.
- **Política de fatia de tempo para processos interrompidos:** 1 período de tempo (correspondente a uma troca de contexto) é subtraído do tempo em que o processo permanece bloqueado para E/S. Ao voltar para o processador, o processo executará durante a fatia de tempo restante desde a última execução.
- **Relógio**: o relógio do processador começa no tempo 0, e é incrementado a cada ciclo de execução.

A fila de processos prontos para executar é composta por duas estruturas. A primeira é um dicionário, onde a chave é um número indicando prioridade e o valor é uma fila duplamente terminada, que recebe os processos daquela prioridade. Desta forma, é possível armazenar processos de mesma prioridade de forma ordenada, onde o primeiro a entrar é o primeiro a sair (*FIFO*). A segunda é uma fila de prioridade, que também recebe os processos. Quando um processo deve ser inserido no processador, é consultado o primeiro elemento da fila de prioridade, a fim de se obter a prioridade mais alta no momento. Então, o dicionário é consultado na chave correspondente à prioridade obtida, e o primeiro processo (o mais antigo) é retirado da fila.

O escalonador inicializa recebendo a fatia de tempo e uma lista de processos, denominada lista de processos inicial, e então começa a executar em um loop. Cada iteração dentro do loop representa uma unidade de tempo.

Antes que a CPU faça uma operação, são realizadas as seguintes verificações:

- A lista de processos bloqueados é percorrida, procurando por processos cujo tempo de operação de E/S já terminou e os inserindo de volta para a fila de prontos para executar;

- A lista de processos inicial é percorrida, para verificar se existem processos cujo tempo de chegada terminou. O tempo de chegada é comparado com o tempo do relógio interno da CPU, e, sendo igual ou menor, o processo é inserido na fila de prontos para executar;
- Agora o processo na CPU é verificado:
  - Caso o tempo de execução terminou, o processo será removido e "descartado" (não inserido em mais nenhuma fila ou lista);
  - Caso a fatia de tempo terminou, o processo será removido e reinserido na fila de prontos para executar;
  - Caso um processo com maior prioridade esteja na fila de prontos para executar, o processo executando será removido e inserido na fila, para que o processo com maior prioridade seja inserido na CPU;
  - Caso seja realizada uma operação de E/S, o processo será removido e inserido na lista de bloqueados, onde vai ficar durante o tempo de operação.

Caso uma das verificações acima tenha ocasionado remoção ou substituição do processo na CPU, uma troca de contexto é realizada. Se houver algum processo na fila de prontos para executar, o de maior prioridade é inserido na CPU.

Agora é executado um ciclo de processamento. Se estiver em troca de contexto, será mostrada a letra C. Se houver um processo em execução, o índice dele é mostrado, caso contrário, o símbolo "-".

Para que o escalonador não fique executando indefinidamente, o loop é mantido enquanto houver processos na fila de prontos para executar, lista de processos bloqueados, lista inicial ou CPU.

No final, é mostrado o gráfico em modo texto da execução, e os tempos médios de resposta e espera.

4. Exemplos

4.1. Primeiro exemplo

```
5
3
3 10 2
5 12 1
9 15 2
11 15 1
12 8 5 2
Tempo de espera médio: 39.0
Tempo de resposta médio: 20.8
---C1C222C222C444C222C444C222C444C444C444C11C333C111C333C111
C333C1C333C333C55C---C5C555C55C-
```

4.2. Segundo exemplo

```
5
```

```
7
1 10 1 5 9
1 12 1
35 15 2 4 8 12
35 15 1
35 8 1
Tempo de espera médio: 17.2
Tempo de resposta médio: 9.4
-C11111C2222222C11C22222C11C---C1C---C4444444C5555555C
4444444C5C4C3333C---C333C3C---C3333C---C33C3C-
```

4.3. Terceiro exemplo

```
5
5
1 10 2
3 12 1
10 15 1
30 15 2
32 8 2

Tempo de espera médio: 25.4
Tempo de resposta médio: 7.2
-C1C22222C22222C33333C22C33333C33333C1111C44444C55555C
11111C44444C555C44444C-
```

5. Conclusão

Através do desenvolvimento do trabalho melhoramos nosso entendimento sobre o funcionamento de escalonadores de processos, podendo compreender como o sistema operacional é capaz de gerenciar a execução dos processos com eficiência.

A solução desenvolvido para simular a alternância de processos através do algoritmo Round-robin apresentou-se simples e eficiente para o que foi proposto, os resultados foram precisos e código ficou claro e inteligível.

Referências

Abraham Silberschatz, P. B. G. and Gagne, G. (2012). In *Operating System Concepts*, pages 263–264. Wiley, 9th edition.