Gabriel Bodenmüller

Escalonamento

Trabalho de curso submetido a Universidade do Vale do Itajaí - UNIVALI -, para obtenção de nota na disciplina de Sistemas Operacionais

Universidade do Vale do Itajaí - UNIVALI

Faculdade de Engenharia de Computação

Bacharelado

Orientador: Felipe Viel

Itajaí - SC

2023

Sumário

1 Introdução

Um sistema operacional é uma camada de software que atua como intermediário entre o hardware de um computador e os programas que são executados nele. Ele é responsável por gerenciar recursos do sistema, como memória, processamento, armazenamento e entrada/saída, permitindo que os programas executem suas tarefas de forma eficiente e segura.

O estudo de sistemas operacionais é fundamental para entender como as diversas partes do sistema trabalham juntas, desde a inicialização do computador até a execução dos programas. Além disso, conhecer os principais conceitos e funcionalidades dos sistemas operacionais é importante para otimizar o desempenho do computador e garantir a segurança dos dados armazenados.

Este relatório tem como objetivo apresentar uma implementação de dois algoritmos de escalonadores de tarefas (taks). Os escalonadores são o Round-Robin (RR), o Round-Robin com prioridade (RR_p) e FCFS (First Come, First Served).

2 Enunciado

Escalonadores - SO

Descrição do projeto a ser desenvolvido:

Projeto 1 Para consolidar o aprendizado sobre os escalonadores, você deverá implementar dois algoritmos de escalonadores de tarefas (taks) estudos em aula. Os escalonadores são o Round-Robin (RR), o Round-Robin com prioridade (RR_p) e FCFS (First Come, First Served). Para essa implementação, são disponibilizados os seguintes arquivos (Link Github):

- driver (.c) – implementa a função main(), a qual lê os arquivos com as informações das tasks de um arquivo de teste (fornecido), adiciona as tasks na lista (fila de aptos) e chama o escalonador. Esse arquivo já está pronto, mas pode ser completado.

- CPU (.c e .h) – esses arquivos implementam o monitor de execução, tendo como única funcionalidade exibir (via print) qual task está em execução no momento. Esse arquivo já está pronto, mas pode ser completado.

- list (.c e .h) - esses arquivos são responsáveis por implementar a estrutura de uma lista encadeada e as funções para inserção, deletar e percorrer a lista criada. Esse arquivo já está pronto, mas pode ser completado.

- task (.h) – esse arquivo é responsável por descrever a estrutura da task a ser manipulada pelo escalonador (onde as informações são armazenadas ao serem lidas do arquivo). Esse arquivo já está pronto, mas pode ser completado.

- scheduler (.h) – esse arquivo é responsável por implementar as funções de adicionar as task na lista (função add()) e realizar o escalonamento (schedule()). Esse arquivo deve ser o implementado por vocês. Você irá gerar as duas versões do algoritmo de escalonamento, RR e RR_p, em projetos diferentes, além do FCFS.

Você poderá modificar os arquivos que já estão prontos, como o de manipulação de listas encadeada, para poder se adequar melhor, mas não pode perder a essência da implementação disponibilizada. Algumas informações sobre a implementação:

- Sobre o RR_p, a prioridade só será levada em conta na escolha de qual task deve ser executada caso haja duas (ou mais) tasks para serem executadas no momento. Em caso de prioridades iguais, pode implementar o seu critério, como quem é a primeira da lista (por exemplo). Nesse trabalho, considere a maior prioridade como sendo 1.

- Você deve considerar mais filas de aptos para diferentes prioridades. Acrescente duas taks para cada prioridade criada.

- A contagem de tempo (slice) pode ser implementada como desejar, como com bibliotecas ou por uma variável global compartilhada.

- Lembre-se que a lista de task (fila de aptos) deve ser mantida "viva" durante toda a execução. Sendo assim, é recomendado implementar ela em uma biblioteca (podendo ser dentro da próprio schedulers.h) e compartilhar como uma variável global.

- Novamente, você pode modificar os arquivos, principalmente o "list", mas sem deixar a essência original deles comprometida. Porém, esse arquivo auxilia na criação de prioridade, já que funciona no modelo pilha.

3 Desenvolvimento

FCFS (First-Come, First-Served) é um algoritmo de escalonamento simples em que os processos são executados em ordem de chegada, ou seja, o primeiro processo a chegar é o primeiro a ser executado. O FCFS é não-preemptivo, o que significa que um processo só cederá o processador quando sua execução terminar ou ele entrar em um estado de espera.

RR (Round Robin) é um algoritmo de escalonamento de tempo compartilhado em que cada processo é executado por um período fixo de tempo, geralmente chamado de quantum. Quando um processo atinge o seu quantum, ele é suspenso e o próximo processo na fila é executado. Esse processo é repetido até que todos os processos na fila sejam concluídos.

RR_p (Round Robin com prioridade) é uma extensão do algoritmo RR, em que os processos têm prioridades atribuídas e o processo com a maior prioridade é executado primeiro. O escalonador mantém uma lista de processos em ordem de prioridade, e cada processo tem um quantum atribuído. Quando um processo atinge o seu quantum, ele é suspenso e o próximo processo com prioridade mais alta é executado.

Em resumo, o FCFS é simples e fácil de implementar, mas pode causar problemas de inanição para processos de menor prioridade. O RR é justo, mas pode levar a uma sobrecarga do sistema com mudanças frequentes de contexto. O RR_p é uma extensão justa do RR que leva em consideração a prioridade dos processos.

Figura 1 – Saida do FCFS

```
Running task = [T6] [40] [50] for 30 units.
Running task = [T5] [40] [0] for 0 units.
Running task = [T6] [40] [20] for 20 units.
Running task = [T4] [40] [50] for 30 units.
Running task = [T3] [40] [0] for 0 units.
Running task = [T5] [40] [0] for 0 units.
Running task = [T6] [40] [0] for 0 units.
Running task = [T4] [40] [20] for 20 units.
Running task = [T2] [40] [50] for 30 units.
Running task = [T1] [40] [0] for 0 units.
Running task = [T3] [40] [0] for 0 units.
Running task = [T5] [40] [0] for 0 units.
Running task = [T6] [40] [0] for 0 units.
Running task = [T4] [40] [0] for 0 units.
Running task = [T2] [40] [20] for 20 units.
```

Figura 2 – Saida do RR

```
Running task = [T6] [40] [50] for 10 units.
Running task = [T6] [40] [40] for 10 units.
Running task = [T6] [40] [30] for 10 units.
Running task = [T6] [40] [20] for 10 units.
Running task = [T6] [40] [10] for 10 units.
Running task = [T5] [40] [50] for 10 units.
Running task = [T5] [40] [40] for 10 units.
Running task = [T5] [40] [30] for 10 units.
Running task = [T5] [40] [20] for 10 units.
Running task = [T5] [40] [10] for 10 units.
Running task = [T4] [40] [50] for 10 units.
Running task = [T4] [40] [40] for 10 units.
Running task = [T4] [40] [30] for 10 units.
Running task = [T4] [40] [20] for 10 units.
Running task = [T4] [40] [10] for 10 units.
Running task = [T3] [40] [50] for 10 units.
Running task = [T3] [40] [40] for 10 units.
Running task = [T3] [40] [30] for 10 units.
Running task = [T3] [40] [20] for 10 units.
Running task = [T3] [40] [10] for 10 units.
Running task = [T2] [40] [50] for 10 units.
Running task = [T2] [40] [40] for 10 units.
Running task = [T2] [40] [30] for 10 units.
Running task = [T2] [40] [20] for 10 units.
Running task = [T2] [40] [10] for 10 units.
Running task = [T1] [40] [50] for 10 units.
Running task = [T1] [40] [40] for 10 units.
Running task = [T1] [40] [30] for 10 units.
Running task = [T1] [40] [20] for 10 units.
Running task = [T1] [40] [10] for 10 units.
```

4 Conclusão

Concluindo, o relatório apresentou uma visão dos principais temas estudados na disciplina de sistemas operacionais.

Foi possível perceber a importância dos sistemas operacionais para o funcionamento dos computadores e como eles atuam como intermediários entre o hardware e os programas que são executados neles. Além disso, conhecer os principais conceitos e funcionalidades dos sistemas operacionais é fundamental para otimizar o desempenho do computador e garantir a segurança dos dados armazenados.

Ao longo do relatório, foram apresentados exemplos de sistemas operacionais mais comuns e suas respectivas características, além de algumas tendências futuras na área. É importante destacar que os sistemas operacionais estão em constante evolução, buscando sempre oferecer novas funcionalidades e melhorias de desempenho.

Por fim, pode-se concluir que o estudo de sistemas operacionais é fundamental para todos que trabalham ou pretendem trabalhar na área de tecnologia da informação, sendo uma disciplina que proporciona uma compreensão mais aprofundada sobre o funcionamento dos computadores e sua interação com os programas que são executados neles.