INSTITUTO FEDERAL DE SANTA MARIA
CAMPUS CAMOBI

Leandro Oliveira Galbarino do Nascimento

RELATÓRIO – SISTEMAS OPERACIONAIS

SANTA MARIA- RS
2024

1-Objetivo

O objetivo deste experimento é compreender o funcionamento de um sistema operacional por meio da implementação prática de uma simulação de seus principais componentes e do hardware associado. Este processo visa explorar as principais funcionalidades e componentes que garantem o funcionamento correto de um sistema operacional, incluindo:

- Gerenciamento de processos, desde a criação até a remoção.
- Escalonamento de processos, utilizando diferentes algoritmos para simular cenários reais.
- Controle de estados de processos (pronto, executando, bloqueado, etc.).
- Gerenciamento de filas e tabelas de processos de forma eficiente.
- Monitoramento de métricas operacionais e desempenho do sistema.

Essa abordagem experimental permite uma visão prática e detalhada das decisões e desafios envolvidos na implementação de um sistema operacional funcional, preparando para uma melhor compreensão de conceitos teóricos e aplicação em cenários reais.

2-Materiais e Métodos

Neste experimento, não foram utilizados hardwares físicos. Em vez de utilizar hardware físico, foram empregados códigos em C e Assembly que simulam o funcionamento de componentes de hardware, como processador e memória, porém com uma arquitetura simplificada, adequada para aprendizado e experimentação.

A implementação do sistema operacional (SO) foi feita majoritariamente na linguagem de programação C.

Materiais Utilizados:

- **Ambiente de Desenvolvimento:** Sistema Operacional Windows com WLS(Subsistema do Windows para Linux), Visual Studio Code e compilador GCC.
- **Simulador de Hardware:** conjunto de funções em C e Assembly que imitam componentes como processador, memória e dispositivos de entrada/saída.
- **Ferramentas de Análise:** console para logs e métricas do SO, permitindo a depuração e avaliação do desempenho.

Métodos:

1. **Simulação de Hardware:**
   Foi implementada uma camada simulada contendo dispositivos como terminal de entrada e saída, relógio, memória, etc. Cada dispositivo interagia com o SO via interrupções.

2. Estrutura do Sistema Operacional (SO)

A estrutura do sistema operacional simulado foi implementada com foco na manipulação eficiente de processos e gerenciamento dos diferentes estados que um processo pode ter durante sua execução. Os principais componentes incluem:

Filas e Tabela de Processos:

- Filas de Processos Prontos: Contém os processos que aguardam execução. Eles podem ser inseridos na fila quando o processo é criado ou quando é desbloqueado.
- **Tabela de Processos:** Armazena todos os processos, independentemente do estado em que se encontram. Quando um processo é criado, ele é adicionado na tabela de processos.

Estados de Processos:

- Pronto: O processo está pronto para ser executado assim que a CPU estiver disponível.
- Executando: O processo está atualmente em execução pela CPU.
- Bloqueado: O processo não pode continuar a execução até que uma condição externa seja atendida (como a conclusão de uma operação de E/S).
- Morto: O processo terminou sua execução e não pode mais ser alterado.

Manipulação de Processos:

- Criação: A criação de um novo processo envolve a alocação, a inicialização e a inserção do processo tanto na tabela de processos quanto na fila de processos prontos, onde aguardará sua vez para ser executado.
- Escalonamento: A função de escalonamento seleciona o processo que será executado a seguir. Foram implementados diferentes algoritmos de escalonamento, como o Escalonador Simples (sequencial), Round Robin e o Escalonador Circular com Prioridade.
- Bloqueio e Desbloqueio: O processo pode ser bloqueado quando aguarda um evento externo. O desbloqueio ocorre assim que o evento ocorre, movendo o processo de volta para a fila de processos prontos.

Tipos de Bloqueio:

- Bloqueio por E/S: O processo fica bloqueado enquanto espera pela conclusão de uma operação de entrada/saída.
- Bloqueio por Espera de Outro Processo: O processo fica bloqueado enquanto o processo solicitado não tiver terminado sua execução.

3-Métricas

O sistema operacional foi programado para registrar métricas importantes, tanto do sistema quanto dos processos criados. Para o sistema, as métricas incluem: tempo total de execução, tempo em que o sistema ficou ocioso, número de interrupções de cada tipo e o número total de preempções. Para os processos, as métricas incluem: tempo de retorno, número de preempções do processo, tempo em cada estado, número de vezes em cada estado e o tempo de resposta.

4-Dados de Desempenho dos Escalonadores

Os dados apresentados foram obtidos após a execução do sistema operacional com diferentes configurações de escalonadores. As simulações variaram o comportamento de escalonamento, alternando entre três estratégias:

1. **Escalonador Simples**
    - O escalonador sequencial é o mais simples dos três e realiza a execução dos processos de forma linear. Ele seleciona o primeiro processo "pronto" na tabela de processos e executa até que o processo termine sua execução ou seja bloqueado.
    - Este escalonador não faz uso de quantum ou prioridade, e os processos são executados na ordem em que chegam na tabela.
    - A principal vantagem desse algoritmo é a simplicidade de implementação, mas sua desvantagem é a falta de flexibilidade e a possibilidade de baixa performance.
2. **Escalonador Round Robin**
    - O algoritmo Round Robin é um dos mais comuns em sistemas operacionais modernos e é baseado na distribuição igualitária de tempo de CPU entre os processos.
    - Cada processo recebe um **quantum de tempo** (intervalo de tempo fixo) para execução. Se o processo não terminar dentro desse tempo, ele é interrompido e retorna à fila de "pronto", sendo então escolhido o próximo processo na fila.
    - O Round Robin é eficaz para sistemas onde a equidade na alocação de CPU é importante, garantindo que todos os processos recebam uma chance de execução de maneira cíclica.
    - A principal desvantagem do Round Robin é que ele pode ser ineficiente para processos que exigem muito tempo de CPU, pois a troca frequente de contexto pode causar overhead.

3. **Escalonador Circular com Prioridade**
    - O escalonador Circular com Prioridade é uma variação do Round Robin que introduz uma priorização entre os processos. Em vez de simplesmente seguir a ordem da fila, cada processo possui uma prioridade associada.

- Os processos com maior prioridade são executados antes dos processos de menor prioridade. Caso haja processos com a mesma prioridade, o algoritmo ainda segue o comportamento Round Robin, dando uma fatia de tempo para cada processo dentro da sua classe de prioridade.
- Esse algoritmo permite uma execução mais controlada de processos importantes, reduzindo a latência de processos críticos e permitindo uma maior flexibilidade na alocação de recursos.
- A desvantagem é que processos de menor prioridade podem ser desconsiderados por longos períodos, dependendo da quantidade de processos de alta prioridade na fila.
- No contexto do nosso sistema operacional, os processos com menor número de prioridade são executados primeiro.

Para comparar o desempenho de cada algoritmo, medimos o tempo médio de espera, tempo de resposta e taxa de utilização da CPU para diferentes cenários de carga de processos. Cada algoritmo teve um desempenho distinto dependendo da natureza dos processos no sistema, com o Round Robin sendo mais equilibrado em termos de tempo de resposta para todos os processos, o Circular com Prioridade se destacando para processos críticos, e o sequencial apresentando boa performance em cenários simples.

A seguir, apresentamos os dados obtidos. Embora tenham sido registrados como se fossem em segundos (s), é importante observar que não representam segundos reais, mas sim o tempo de clock do processador.

Simulação com Escalonador Simples	
Métricas do Sistema Operacional	
Processos Criados	4
Tempo total de execução	20337s
Tempo total do sistema ocioso	3730s
Número de interrupções por Reset	1
Número de interrupções por Erro de execução	0
Número de interrupções Chamada de sistema	460
Número de interrupções E/S: relógio	674
Número de interrupções E/S: teclado	0
Número de interrupções E/S: console	0
Número de preempções	0

Métricas dos Processos				
Processo	0	1	2	3
Tempo retorno	20337s	10156s	14896s	19586s
Preempções do processo	0	0	0	0
Estado executando	71	410	238	168
Tempo executando	773s	9573s	8033s	4284s
Estado pronto	4	9	21	29
Tempo pronto	0s	24s	5476s	11572
Estado bloqueado	3	8	20	113
Tempo bloqueado	19564s	559s	1387s	3730s
Tempo de resposta	0s	2s	260s	399s

Simulação com Escalonador Circular com Prioridade	
Métricas do Sistema Operacional	
Processos Criados	4
Tempo total de execução	20337s
Tempo total do sistema ocioso	1714s
Número de interrupções por Reset	1
Número de interrupções por Erro de execução	0
Número de interrupções Chamada de sistema	460
Número de interrupções E/S: relógio	674
Número de interrupções E/S: teclado	0
Número de interrupções E/S: console	0
Número de preempções	45

Métricas dos Processos				
Processo	0	1	2	3
Tempo retorno	20337s	17683s	11292s	19289s
Preempções do processo	1	32	7	5
Estado executando	4	39	22	60
Tempo executando	773s	9557s	3901s	4392s
Estado pronto	4	39	22	60
Tempo pronto	297s	7719s	6481s	10816s
Estado bloqueado	2	8	14	93
Tempo bloqueado	19270s	407s	910s	4081s
Tempo de resposta	74s	197s	294s	180s

Simulação com Escalonador Round Robin	
Métricas do Sistema Operacional	
Processos Criados	4
Tempo total de execução	20340s
Tempo total do sistema ocioso	2228s
Número de interrupções por Reset	1
Número de interrupções por Erro de execução	0
Número de interrupções Chamada de sistema	460
Número de interrupções E/S: relógio	675
Número de interrupções E/S: teclado	0
Número de interrupções E/S: console	0
Número de preempções	44

Métricas dos Processos				
Processo	0	1	2	3
Tempo retorno	20340s	16912s	11541s	19292s
Preempções do processo	1	32	6	5
Estado executando	4	37	20	55
Tempo executando	773s	9554s	3895s	3890s
Estado pronto	4	37	20	55
Tempo pronto	297s	7083s	6773s	11412s
Estado bloqueado	2	7	13	99
Tempo Bloqueado	19270s	295s	873s	3990s
Tempo de resposta	74s	191s	338s	207s

5-Resultados

O tempo de execução dos três escalonadores foi praticamente igual, apresentando valores próximos de 20337s para todos. No entanto, as diferenças mais relevantes aparecem em outras métricas, como o tempo total em que o sistema ficou ocioso. Os resultados foram:

- Escalonador Simples: 3730s de ociosidade.
- Escalonador Round Robin: 2228s de ociosidade, uma redução de 40,26% em relação ao Simples.
- Escalonador Circular com Prioridade: 1714s de ociosidade, representando uma redução adicional de 23,07% em relação ao Round Robin e 54,07% em relação ao Escalonador Simples.

Esses números mostram que os algoritmos circulares conseguem aproveitar melhor os ciclos da CPU, reduzindo significativamente o tempo ocioso. No nosso teste, o Circular com Prioridade se saiu melhor que o Round Robin, mas isso não ocorre sempre, sendo que o desempenho pode variar conforme o cenário.

Ao analisarmos o comportamento de cada escalonador, percebemos que:

- O Escalonador Simples é sequencial e não utiliza preempções. Ele executa um processo por vez até que este seja bloqueado ou finalizado. Essa abordagem resulta em maior ociosidade da CPU, pois a execução é interrompida sempre que um processo aguarda recursos ou eventos externos, sem a possibilidade de adiantar outros processos prontos.

- O Escalonador Round Robin e o Circular com Prioridade, por outro lado, fazem uso de preempções para alternar entre os processos, garantindo uma utilização mais eficiente da CPU e permitindo que os processos avancem de maneira mais equilibrada.

As métricas gerais do sistema operacional são semelhantes mesmo com os diferentes escalonadores, como o número de interrupções e chamadas de sistema, porém a grande diferença está nas métricas dos processos individuais.

No Escalonador Simples, o tempo de retorno e o tempo de resposta dos processos tendem a ser mais altos para processos que não são iniciados imediatamente. Como ele realiza etapas de maneira linear (um processo é executado até ser bloqueado ou finalizado antes de passar para o próximo), há uma forte penalização para processos que precisam esperar longos períodos até sua vez de executar. Isso contrasta com os escalonadores circulares, que alternam entre os processos, reduzindo o impacto desse tempo de espera.

Por exemplo, os tempos de resposta nos escalonadores circulares são mais equilibrados, com valores significativamente menores para processos que não sejam o inicial (init). Além disso, os tempos de retorno dos processos nos algoritmos circulares também são mais uniformes, indicando que todos os processos terminam em intervalos

de tempo mais próximos, ao contrário do escalonador simples, que pode deixar processos esperando muito tempo.

Essas características tornam os escalonadores circulares mais adequados para cenários em que a responsividade e a utilização eficiente da CPU são prioritárias.

6-Conclusões

Os escalonadores circulares, como o Round Robin e o Circular com Prioridade, se destacam por buscar um melhor aproveitamento do tempo da CPU. Eles tentam equilibrar a execução dos processos, de forma que todos tenham chances de progredir e os recursos sejam utilizados de forma eficiente. Já o Escalonador Simples, apesar de ser mais direto e menos custoso em termos de implementação, não consegue aproveitar tanto a capacidade da CPU, especialmente em sistemas com múltiplos processos aguardando recursos ou eventos.

7-Referências Bibliográficas

https://github.com/BenhurUFSM/so24b

Livro: Sistemas Operacionais Modernos - Andrew S. Tanenbaum