Relatório de Monitoramento e Análise de System Calls

Data: 24 de agosto de 2024

Horário: 22:45

Duração da coleta: 8 segundos

Alunos: Guilherme Batista Da Silva e Reney lima Chaves

1. Introdução

Este relatório apresenta os resultados de uma atividade prática cujo objetivo foi monitorar, analisar e comparar as chamadas de sistema (system calls) de um processo simples, observando como essas chamadas se comportam em diferentes tipos de sistemas operacionais: multiprogramáveis e monoprogramáveis. A atividade buscou entender a função de diferentes system calls e como elas variam em complexidade e frequência em diferentes ambientes operacionais.

2. Monitoramento do Processo

Processo Escolhido:

O processo selecionado para monitoramento foi o comando ls, que é amplamente utilizado em sistemas Unix-like para listar o conteúdo de diretórios.

Método de Coleta:

Utilizamos um script de coleta de logs para monitorar o processo ls por 10 segundos. Durante esse tempo, todas as chamadas de sistema feitas pelo processo foram registradas para posterior análise.

Ferramentas Utilizadas:

Um script personalizado foi utilizado para capturar e classificar as chamadas de sistema em tempo real.

Um segundo script foi empregado para interpretar os logs coletados e gerar um relatório detalhado.

3. Resultados da Análise das System Calls

Após a coleta de dados, as chamadas de sistema foram categorizadas com base em sua função principal. A seguir, estão os principais tipos de chamadas observadas:

1. Entrada/Saída de Arquivos (File I/O):

read() e restart_syscall(): Essas chamadas foram usadas para operações de leitura de dados, tipicamente para acessar o conteúdo dos arquivos dentro do diretório que estava sendo listado. A restart_syscall() é usada para retomar uma chamada de sistema que foi interrompida, garantindo que a operação seja completada.

2. Gerenciamento de Processos:

epoll_wait(): Essa chamada foi observada várias vezes e está relacionada ao monitoramento de múltiplos descritores de arquivos para determinar quais estão prontos para I/O. Isso é comum em sistemas multiprogramáveis onde o sistema precisa gerenciar múltiplos processos simultaneamente.

3. Sincronização de Threads:

futex(): Frequentemente observada, a futex() é uma chamada de sistema que permite a sincronização eficiente entre threads. Essa chamada é essencial para operações que envolvem múltiplos threads em execução concorrente, garantindo que um thread espere pela liberação de um recurso antes de prosseguir.

4. Outras Ações:

pselect6(): Outra chamada relevante que permite que o processo espere até que um ou mais descritores de arquivos estejam prontos para I/O, com a opção de suspender o processo até que um sinal seja entregue.

4. Comparação entre Sistemas Operacionais

Sistema Multiprogramável:

Em sistemas multiprogramáveis, há uma necessidade constante de gerenciamento de múltiplos processos e threads simultâneos. Isso ficou evidente pelo uso frequente de chamadas como epoll_wait() e futex(). Essas chamadas são essenciais para que o sistema consiga coordenar as atividades de vários processos ao mesmo tempo, lidando com a concorrência de maneira eficiente.

Sistema Monoprogramável:

Em um sistema monoprogramável, onde apenas um processo é executado por vez, as chamadas relacionadas à multitarefa, como epoll_wait(), seriam desnecessárias. A ausência de concorrência entre processos significa que o sistema operacional não precisaria de mecanismos complexos para monitorar ou sincronizar processos, resultando em um número menor de chamadas de sistema e, possivelmente, em uma simplificação no código do sistema operacional.

5. Reflexão e Discussão em Grupo

Participação:

Os membros do grupo participaram ativamente das discussões e das atividades práticas. Cada aluno contribuiu para a análise dos logs, a categorização das system calls, e a elaboração das comparações entre os diferentes tipos de sistemas operacionais.

Análise Crítica:

O grupo conseguiu identificar e discutir as diferenças fundamentais entre os sistemas multiprogramáveis e monoprogramáveis. Foi observado que, em sistemas multiprogramáveis, há uma maior diversidade e complexidade nas system calls, refletindo a necessidade de gerenciamento de múltiplos processos. Em contraste, um sistema monoprogramável apresenta um comportamento mais linear, com menos chamadas de sistema focadas principalmente na execução de um único processo.

Questões Debatidas:

Como a multitarefa influencia o comportamento das system calls?

Foi discutido que a multitarefa exige um sistema mais robusto para o gerenciamento de processos, o que aumenta a quantidade e a complexidade das chamadas de sistema, como futex() para sincronização entre threads.

Como a falta de multitarefa em sistemas monoprogramáveis afeta a execução de processos?

Em sistemas monoprogramáveis, as operações são mais simples, e há menos necessidade de system calls relacionadas à coordenação entre processos, resultando em uma execução mais direta e com menos overhead.

6. Conclusão

Completude:

Todos os passos da atividade foram realizados conforme planejado. Os processos foram monitorados, as system calls foram categorizadas e analisadas, e as comparações entre os diferentes sistemas operacionais foram feitas com sucesso.

Clareza na Interpretação:

O relatório explica claramente as funções das system calls observadas e as diferenças entre sistemas multiprogramáveis e monoprogramáveis. As explicações são objetivas e cobrem os principais pontos de interesse da atividade.

Participação:

A participação de todos os membros foi crucial para o sucesso da atividade. Cada aluno contribuiu de forma significativa nas discussões e na análise dos dados.

Análise Crítica:

Os alunos demonstraram uma capacidade crítica ao comparar e contrastar os diferentes sistemas operacionais. As discussões foram produtivas e levaram a uma compreensão mais profunda do funcionamento interno dos sistemas operacionais em diferentes contextos.