Eduardo Knabben Tiyo - 2551748
Pedro Chouery Grigolli - 2551845
Felipe Martins Sanches - 2390809
Ingrid Reupke Sbeguen Moran - 2349388

Manipulação de threads

Relatório técnico de atividade prática solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Computação – DACOM
Bacharelado em Ciência da Computação – BCC

Campo Mourão
Abril / 2025

Resumo

Este relatório apresenta a análise e implementação de aplicações com uso de múltiplas threads em ambiente GNU/Linux, utilizando a biblioteca *POSIX Threads* (*pthreads*) na linguagem de programação `C`. Foram realizadas atividades de identificação de *threads* em execução no sistema, análise de desempenho com diferentes quantidades de *threads* e o desenvolvimento de aplicações paralelas para busca em vetores e cálculo de médias em matrizes. Os resultados obtidos demonstram os benefícios do paralelismo e evidenciam a importância da programação concorrente para o aproveitamento eficiente dos recursos computacionais modernos.

Palavras-chave: Threads. Paralelismo. Pthreads. GNU/Linux. Sistemas Operacionais.

Sumário

1 Introdução

Este relatório refere-se ao desenvolvimento e análise de programas multithread no sistema operacional GNU/Linux. O documento está estruturado em: objetivos, fundamentação, materiais utilizados, descrição dos procedimentos, análise dos resultados obtidos e conclusão.

2 Objetivos

- Compreender as principais operações usadas em *threads*.
- Desenvolver aplicações usando *threads*.
- Explorar programação com *Threads POSIX* **(pthreads)**

3 Fundamentação

Threads são unidades de execução dentro de um processo que compartilham o mesmo espaço de memória, permitindo a realização de múltiplas tarefas de forma simultânea. O uso de *threads* é uma abordagem eficiente para implementar paralelismo em sistemas com múltiplos núcleos de processamento, aumentando a performance de aplicações que demandam alto poder computacional.

A biblioteca *POSIX Threads*, conhecida como `pthreads`, fornece uma interface padronizada para criação, sincronização e gerenciamento de *threads* em sistemas baseados em Unix, como o GNU/Linux. Com ela, é possível dividir uma tarefa em múltiplas subtarefas executadas em paralelo, otimizando o tempo de execução.

No contexto deste laboratório, foram abordadas tanto a análise do comportamento de *threads* no sistema operacional, quanto a implementação prática de aplicações que se beneficiam da paralelização.

4 Materiais

- Distribução GNU/Linux Debian 12.10.
- Kernel Linux 6.14.
- Intel Core i5-1135G7 @4.20GHz.
- Intel TigerLake-LP GT2 [Iris Xe Graphics].
- 8GB RAM DDR4 3200MHz.

5 Procedimentos e Resultados

5.1 Identifique no seu sistema Linux quantas *threads* estão em execução. Qual o processo com o maior número de *threads*?

Para identificar as *threads* em execução, foi executado comando `top -H`. No canto superior esquerdo, é possível visualizar as *threads* totais, em execução, e as que estão "dormindo". Existem 1013 *threads* totais, 1 executando, 1012 dormindo, 0 paradas e 0 *zombies* (Figura 1).

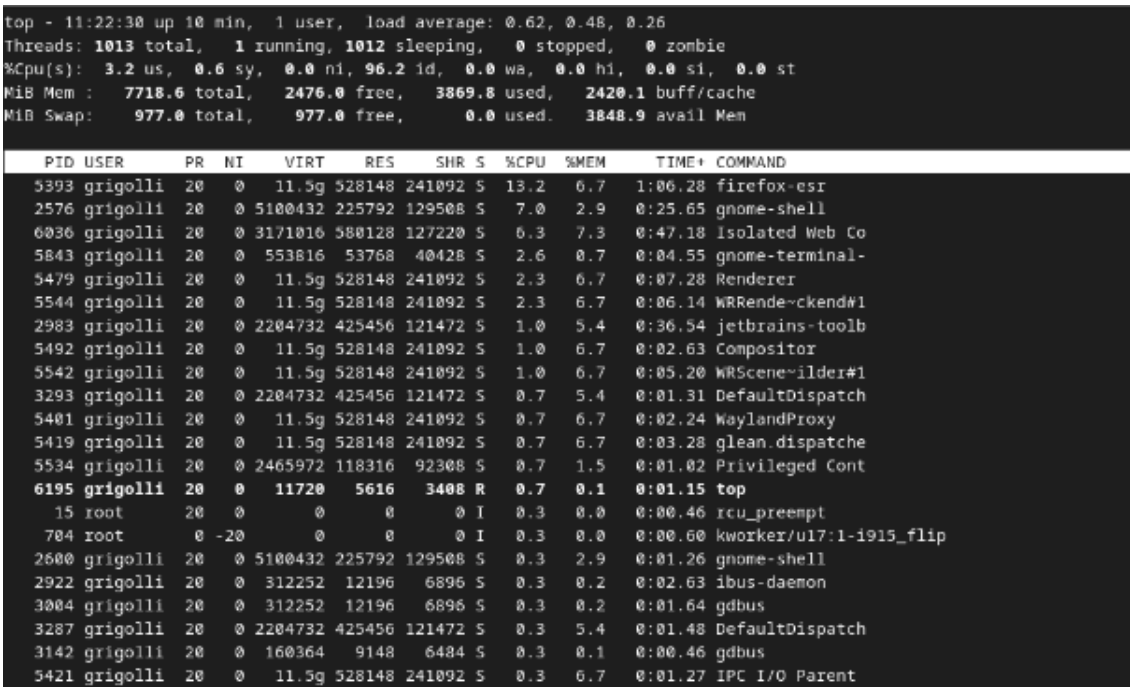

Figura 1 – Execução do comando `top -H`

Ao executar o comando `ps -eLo pid,ppid,cmd,nlwp -sort=-nlwp | head`, é possível visualizar os processos e a quantidade de NLWP (*Number of Light Weight Processes*) por processo. Como pode-ver visualizar na figura 2, o processo que possui a maior quantidade de *threads* é o *Firefox*, com 120 das 1013 *threads* totais.

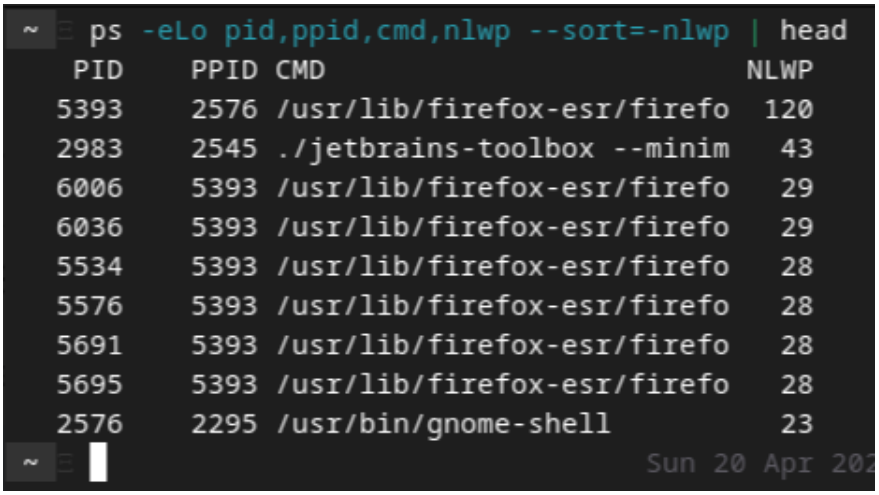

Figura 2 – Execução do comando `ps -eLo pid,ppid,cmd,nlwp -sort=-nlwp | head`

5.2 Qual o número máximo de *threads* que o seu sistema suporta?

Para verificar o número máximo de *threads* que o *kernel* do Linux suporta, utilizou-se o comando `cat /proc/sys/kernel/threads-max`. Esse valor representa o limite global de *threads* que podem estar em execução simultaneamente no sistema. Veja a execução na figura 3

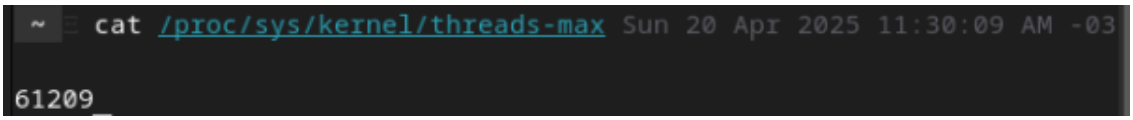

Figura 3 – Execução do comando `cat /proc/sys/kernel/threads-max`

Neste sistema, o valor obtido foi **61209**, indicando o total de *threads* que podem ser criadas no sistema como um todo. Para verificar o número de *threads* que um único usuário pode criar, usamos o comando `ulimit -u`, que retorna o número máximo de processos permitidos por usuário.

Figura 4 – Execução do comando `ulimit -u`

5.3 Verifique o tempo de execução do programa da questão 2, parte 2, considerando: 1 *thread*, 2 *threads*, 4 *threads*, 8 *threads* e 16 *threads*. Descreva o hardware (processador, memória e número de núcleos, tamanho da matriz usada nos testes e o tempo de execução para cada teste).

Para verificar o tempo de execução do programa, foi utilizado um processador 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, com 8Gb DDR4 3200 MHz de memória e 4 núcleos físicos. A tabela 1 exibe os resultados em segundos.

Tabela 1 – Tabela com os tempos de execução em segundos

Matriz 10x10						
Qtde. Threads	**Média (s)**	**Teste 1**	**Teste 2**	**Teste 3**	**Teste 4**	**Teste 5**
1	0,001098	0,00105	0,00102	0,00121	0,00123	0,00100
2	0,001612	0,00156	0,00206	0,00198	0,00111	0,00127
4	0,001926	0,00172	0,00089	0,00248	0,00163	0,00201
8	0,003139	0,003276	0,00310	0,00290	0,00386	0,00361
16	0,003676	0,003888	0,00423	0,00513	0,00470	0,00161
Matriz 100x100						
1	0,004392	0,004392	0,00426	0,00440	0,00444	0,00422
2	0,004984	0,004948	0,00499	0,00498	0,00511	0,00443
4	0,005528	0,005428	0,00510	0,00675	0,00448	0,00434
8	0,003938	0,003904	0,00452	0,00456	0,00648	0,00201
16	0,006662	0,006762	0,00262	0,00755	0,01064	0,00781
Matriz 1000x1000						
1	0,117268	0,11701	0,11308	0,12362	0,11129	0,11510
2	0,118226	0,11865	0,11221	0,12684	0,11783	0,11498
4	0,112582	0,110446	0,11655	0,11330	0,10792	0,11052
8	0,110170	0,109582	0,10820	0,11843	0,10870	0,11098
16	0,117084	0,116096	0,12521	0,11730	0,11749	0,10845

6 Discussão dos Resultados

A realização deste laboratório possibilitou uma compreensão prática do gerenciamento de *threads* no sistema operacional GNU/Linux. Inicialmente, a observação da quantidade de *threads* em execução e a identificação do processo com maior número de *threads* evidenciaram como o sistema distribui tarefas internas, sendo comum que navegadores, editores de texto ou ambientes de desenvolvimento apresentem grande paralelismo interno.

A consulta ao arquivo `/proc/sys/kernel/threads-max` revelou o número máximo de *threads* que o sistema suporta, refletindo um limite global estabelecido pelo *kernel*.

Esse número é relevante para manter a estabilidade do sistema, evitando que processos ou usuários comprometam os recursos disponíveis. Em paralelo, o comando `ulimit -u` demonstrou a quantidade máxima de processos (e, consequentemente, *threads*) que um único usuário pode criar.

A última questão envolveu a execução de um programa com diferentes quantidades de *threads* para processar matrizes de tamanhos variados. Foram realizados cinco testes para cada configuração, e os tempos de execução foram registrados para avaliação do impacto do paralelismo.

Os resultados mostram que a utilização de *threads* é mais vantajosa em contextos onde há uma maior carga de processamento. Para dados pequenos, o custo da paralelização tende a superar seus benefícios. Já para grandes volumes, a divisão das tarefas entre múltiplas *threads* pode proporcionar reduções significativas no tempo de execução.

7 Conclusões

Em geral, após a prática deste laboratório, foi possível compreender de maneira mais aprofundada o uso de *threads* no GNU/Linux, principalmente o uso de *pthreads*. Foi possível observar como a criação e manipulação de *threads* impacta no desempenho das aplicações, melhorando, assim, a utilização de recursos do sistema. Na parte de análise, foi possível observar o comportamento das *threads* em tempo real, quais processos possuem um maior número de *threads* e seus limites, além de conseguir analisar a importância da quantidade de *threads* para com a capacidade do hardware. Já na parte da programação, foi possível colocar em prática o que foi analisado e simular cenários reais de aplicações que lidam com grandes volumes de dados. Portanto, a realização desse laboratório foi essencial para reforçar os conceitos de *threads* e reconhecer a importância de um gerenciamento eficiente delas para, assim, garantir desempenho e estabilidade no sistema.

8 Referências