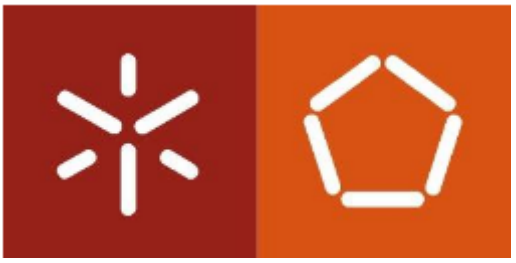

Universidade do Minho

Licenciatura em Engenharia Informática

Sistemas Operativos - Trabalho Prático

Grupo 29

Ano Letivo 24/25

Gonçalo Amaro (A106803) Gonçalo Sá (A107376)

Octávio Henriques (A104277)

Conteúdo

1. Introdução

Este projeto consiste no desenvolvimento de um sistema cliente-servidor para indexação e pesquisa de documentos de texto guardados localmente. O servidor (“dserver.c”) gere a meta-informação dos documentos, armazenando-a em memória e em disco, e processa pedidos enviados pelo cliente (“dclient.c”) através de pipes nomeados. O cliente permite adicionar, consultar, remover e pesquisar documentos de forma simples, executando uma operação por invocação.

O sistema suporta operações concorrentes, persistência dos dados e um mecanismo de cache configurável para melhorar o desempenho. Este relatório descreve a implementação das funcionalidades, a arquitetura adotada e as otimizações realizadas.

2. Funcionalidades Implementadas

O sistema desenvolvido é composto por dois programas principais: o servidor (“dserver.c”) e o cliente (“dclient.c”). A comunicação entre ambos é realizada através de pipes com nome (FIFOs), assegurando a separação clara de responsabilidades e a robustez do serviço. As principais funcionalidades implementadas são:

Indexação de documentos:

A atribuição dos IDs dos documentos é realizada dinamicamente durante a operação de indexação. Como os novos documentos são sempre adicionados com um ID maior que todos os existentes, o maior ID está sempre na cabeça da lista ligada, que representa o documento mais recentemente inserido. Assim, para determinar o próximo ID disponível, o servidor simplesmente incrementa o ID do documento na cabeça da lista. Essa abordagem garante unicidade e coerência dos IDs sem a necessidade de percorrer toda a lista, além de assegurar que não é possível adicionar um documento com um ID inferior a qualquer já existente.

Consulta de informação:

Permite consultar a meta-informação de um documento indexado através do seu identificador único.

Remoção de informação:

Permite remover a indexação da meta-informação de um documento da estrutura de dados e da cache, sem apagar o ficheiro original.

Pesquisa no conteúdo:

Possibilita pesquisar o número de linhas de um documento que contenham uma palavra-chave, assim como listar todos os documentos que a contenham.

Pesquisa concorrente:

Suporta pesquisas paralelas, realizadas através da criação de múltiplos processos filhos, otimizando o desempenho para operações em larga escala e garantindo a correta sincronização e recolha dos resultados.

Persistência:

Toda a meta-informação dos documentos é serializada e armazenada num ficheiro binário ("/tmp/documents.bin"). No arranque do servidor, a informação é carregada para a estrutura de dados em memória, mantendo a continuidade entre sessões.

Cache configurável:

O servidor mantém uma cache em memória com as entradas mais recentes, de tamanho configurável na inicialização. A cache utiliza uma política simples de substituição, onde documentos mais antigos são removidos quando o limite é ultrapassado.

Comando de desligar:

O cliente pode enviar um comando especial (-f) para desligar o servidor de forma controlada, garantindo a persistência dos dados em disco, libertação de memória e encerramento correto dos FIFOs.

Funcionalidades de visualização e limpeza do ficheiro de persistência:

Ao fazer o comando "./bin/dserver", no nosso projeto, é possivel adicionar um quarto argumento. Podendo este argumento ser "check", "erase" ou "check_erase". "check" faz com que no fim de cada operação efetuado pelo cliente e no inicio do programa "dserver.c" seja listado os conteúdos da cache, lista ligada e ficheiro de persistência. "erase" limpa o ficheiro de persistência no inicio do programa "dserver.c". "check_erase" ativa as duas funções mencionadas anteriormente.

3. Testes

Foram desenvolvidos diversos scripts em bash para automatizar as operações e recolher métricas. Um script fornecido pelo docente foi usado para indexar documentos, enquanto scripts próprios foram criados para testar a operação de pesquisa (-s) com diferentes números de processos. Adicionalmente, foram realizados testes de desempenho da cache, tanto num conjunto pequeno de documentos (docs/my docs) como num conjunto extenso (docs/Gdataset), medindo o tempo de execução em função do tamanho da cache, estes scripts também usam operações “-s” para testar. Todos os testes feitos pelo grupo retornam o tempo de execução dos varios casos. Para os testes de cache foram usadas operações -s com keyword “praia” e 4 processos. Aqui estão os resultados:

O comando "./tests/test_cache_small.sh" chegou a estes resultados:

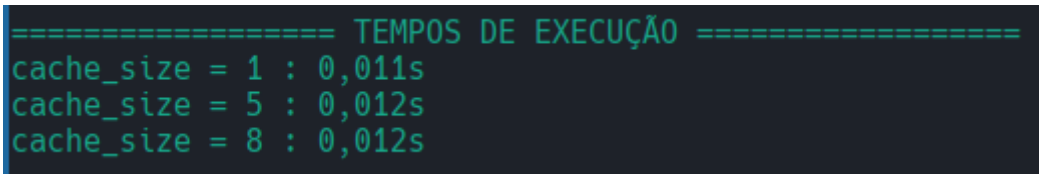

Figura 1: Resultado do comando "./tests/test_cache_small.sh"

O comando "./tests/test_cache_big.sh" chegou a estes resultados:

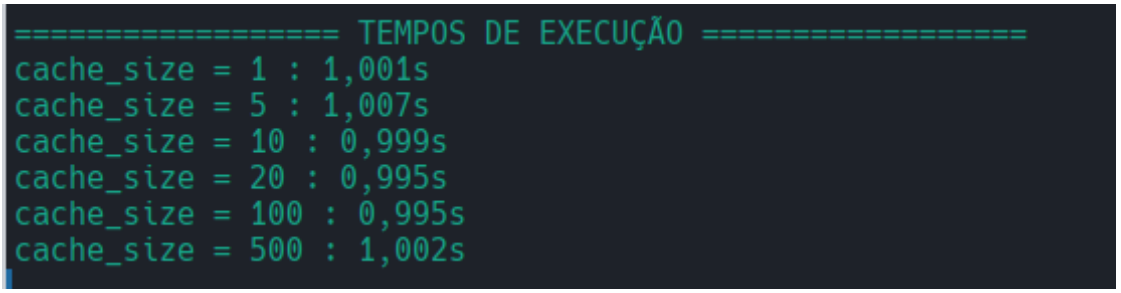

Figura 2: Resultado do comando "./tests/test_cache_big.sh"

O comando "./tests/test_search.sh" chegou a estes resultados (neste teste a keyword a procurar é “huge” no folder “docs/Gdataset”):

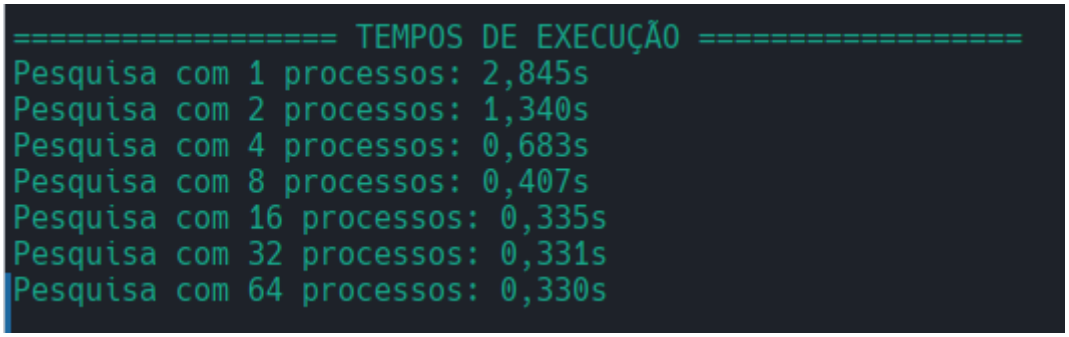

Figura 3: Resultado do comando "./tests/test_search.sh" com a procura da keyword “huge”

Os conjuntos de documentos fornecidos e outros criados para o efeito foram utilizados para testar a robustez e o desempenho do sistema. Os resultados demonstraram a correta implementação das funcionalidades e a eficiência das otimizações introduzidas.

4. Decisões Tomadas

Durante o desenvolvimento do projeto, foram tomadas várias decisões técnicas importantes:

Estruturas de dados:

A meta-informação dos documentos é armazenada em estruturas eficientes em memória, nomeadamente listas ligadas, que permitem um acesso dinâmico e organizado aos dados.

Mecanismos de Comunicação.

A comunicação cliente-servidor é realizada através de pipes nomeados (FIFOs). O servidor cria dois FIFOs globais, um para receber pedidos e outro para enviar respostas. Os clientes enviam as mensagens pelo FIFO de pedidos, e o servidor processa cada mensagem conforme a operação indicada. Para operações normais, o servidor responde diretamente pelo FIFO de respostas. A operação de término do servidor (-f) é tratada internamente no loop principal: ao receber este comando, o servidor guarda a informação necessária, limpa recursos como cache e meta-informação, fecha os FIFOs, remove os arquivos FIFO do sistema e termina o processo principal com exit(0), garantindo um encerramento limpo e controlado.

Política de cache:

O sistema implementa uma cache em memória para armazenar meta-informação de documentos acedidos recentemente, com o objetivo de reduzir o custo de acesso à estrutura de dados principal. A cache é gerida como uma lista ligada simples, onde os documentos mais recentemente acedidos são inseridos na cabeça da lista. Caso o documento já exista na cache, não é reinserido, evitando duplicações. O sistema utiliza uma cache com política de substituição FIFO simples, onde documentos são adicionados à cabeça da lista e o mais antigo (no fim) é removido quando o limite é atingido. A cache é inicializada com a função “init_cache”, limpa com “free_cache”, e pode ser monitorizada com a função “list_documents_cache”, útil para fins de debug e análise de desempenho durante o desenvolvimento.

Paralelismo:

A pesquisa concorrente foi implementada através da criação de múltiplos processos filhos usando “fork()”. Cada operação de pesquisa (consultar, contar, pesquisar) é executada num processo separado, permitindo o processamento paralelo de pedidos e melhor

aproveitamento dos recursos do sistema. A sincronização e recolha de resultados é feita de forma controlada, garantindo que o servidor continua a funcionar sem bloqueios.

Persistência:

A informação sobre os documentos é serializada e guardada num ficheiro binário (“/tmp/documents.bin”), contendo a meta-informação presente na estrutura de dados em memória. Esta informação é carregada no arranque do servidor e atualizada no encerramento, garantindo a integridade e continuidade dos dados entre sessões.

5. Conclusão

O projeto permitiu consolidar conhecimentos sobre sistemas operativos, comunicação entre processos, gestão de memória e persistência de dados. Todas as funcionalidades propostas foram implementadas e testadas com sucesso. O sistema revelou-se eficiente e robusto, sendo possível identificar oportunidades de melhoria futura, como a implementação de novas políticas de cache ou a modificação das funcionalidades de pesquisa.