

Manual de Utilização Servidor Web Multi-Thread

Membros:

Alexandre Regalado 124572

Cristiane Moreno 128076

Visão Geral.....	3
Requisitos do Sistema	3
Sistema Operativo	3
Ferramentas de compilação	3
Compilação.....	4
Comandos Disponíveis:.....	4
Configuração.....	4
Exemplo de Configuração.....	4
Parâmetros Disponíveis.....	4
Execução do Servidor	5
Endpoints disponíveis	5
Serviço de ficheiros Estáticos	5
Endpoints Especiais.....	5
Encerramento do Servidor	5
Resolução de problemas	6
Conclusão.....	6

Visão Geral

O sistema consiste num servidor HTTP implementado na linguagem C para distribuir ficheiros.

Este projeto foi desenvolvido no âmbito da cadeira de Sistemas Operativos, aplicando conceitos como processos, threads, sincronização e comunicação entre processos.

O projeto tem como objetivo a distribuição de ficheiros (html, css, imagens) de forma concorrente e controlada.

O sistema implementa:

- Suporte ao protocolo HTTP com método GET
- Arquitetura prefork com múltiplos processos worker
- Pool de threads por worker para a paralelização de processamento
- Shared queue para distribuição de conexões
- Cache LRU de ficheiros estáticos
- Estatísticas globais em memória partilhada
- Endpoints de monitorização

Requisitos do Sistema

Sistema Operativo

O servidor foi desenvolvido para sistemas Linux compatíveis com POSIX, sendo necessário suporte para processos, threads, memória partilhada e semáforos POSIX.

Ferramentas de compilação

São necessárias as seguintes ferramentas:

- GCC
- GNU Make

Em sistemas baseados em Debian, a instalação pode ser realizada com:

“sudo apt-get install build-essential”

Compilação

Apos obter o código fonte, a compilação do servido e realizada através do *Makefile* fornecido.

Compilação: “*make*”

O binário gerado encontra-se no diretório *bin/*.

Comandos Disponíveis:

Comando	Descrição
make	Compila o servido
make clean	Remove ficheiros gerados
make run	Compila e executa com configuração padrão

Configuração

O comportamento do servidor e controlado por um ficheiro de configuração no formato de CHAVE=VALOR, chamado por *server.conf*.

Exemplo de Configuração

PORT=8080

DOCUMENT_ROOT=/var/www/html

NUM_WORKERS=4

THREADS_PER_WORKER=10

TIMEOUT_SECONDS=30

CACHE_SIZE_MB=10

Parâmetros Disponíveis

Parâmetro	Descrição	Valor Exemplo
-----------	-----------	---------------

PORT	Porta TCP de escuta	8080
DOCUMENT_ROOT	Diretório raiz dos ficheiros	/var/www/html
NUM_WORKERS	Número de processos worker	4
THREADS_PER_WORKER	Threads por worker	10
TIMEOUT_SECONDS	Timeout de sockets	30
CACHE_SIZE_MB	Cache por worker (MB)	10

Execução do Servidor

Com o ficheiro de configuração explícito: “./bin/concurrent-http-server server.conf”

Endpoints disponíveis

Serviço de ficheiros Estáticos

Pedidos GET e HEAD para caminhos relativos ao DOCUMENT_ROOT resultam na devolução do ficheiro.

Em caso de omissão de um caminho é devolvido o ficheiro index.html que se encontra no DOCUMENT_ROOT

Endpoints Especiais

- GET /health: Devolve um JSON com o estado geral do sistema
- GET /stats: Devolve um JSON com as estatísticas globais do servidor
- GET /metrics: Devolve um ficheiro de texto que pode ser usado com o Prometheus para monitorizar o servidor

Encerramento do Servidor

O servidor suporta encerramento através de sinais POSIX

“kill -SIGINT <pid>”

Resolução de problemas

Alguns problemas incluem

- Porta TCP já em uso
- Permissões insuficientes no diretório de documentos
- Configuração incompatível com os recursos do sistema

Conclusão

Este manual apresenta os passos para a correta utilização do servidor HTTP.

Para detalhes adicionais devem ser consultados o relatório técnico e o documento de design.

Nota

Está disponível em <https://github.com/Alxit0/so-websocket-ipc/tree/main> uma versão com docker-compose para compilar e correr o servidor em qualquer ambiente.