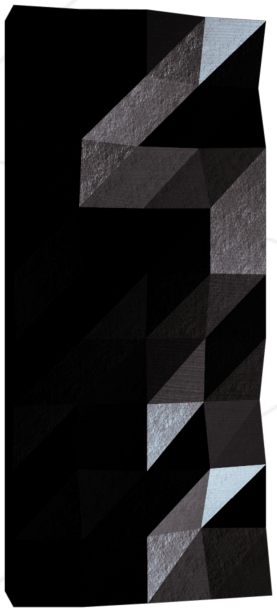

# Webserv

## É aqui que você finalmente entende por que URLs começam com HTTP

*Preâmbulo:*
*Este projeto é sobre escrever seu próprio servidor HTTP.*
*Você poderá testá-lo com um navegador real.*
*HTTP é um dos protocolos mais utilizados na internet.*
*Entender suas complexidades será útil, mesmo que você não trabalhe em um site.*

*Versão: 23.0*

# Sumário

# Capítulo I

# Introdução

O **Hypertext Transfer Protocol** (HTTP) é um protocolo de aplicação para sistemas de informação hipermídia distribuídos e colaborativos.

HTTP é a base da comunicação de dados para a World Wide Web, onde documentos de hipertexto incluem hiperlinks para outros recursos que o usuário pode acessar facilmente. Por exemplo, clicando em um botão do mouse ou tocando na tela em um navegador web.

O HTTP foi desenvolvido para suportar a funcionalidade de hipertexto e o crescimento da World Wide Web.

A função primária de um servidor web é armazenar, processar e entregar páginas web para clientes. A comunicação cliente-servidor ocorre através do Hypertext Transfer Protocol (HTTP).

As páginas entregues são mais frequentemente documentos HTML, que podem incluir imagens, folhas de estilo e scripts, além do conteúdo de texto.

Múltiplos servidores web podem ser usados para um site de alto tráfego, dividindo o tráfego entre várias máquinas físicas.

Um agente do usuário, comumente um navegador web ou um rastreador web, inicia a comunicação solicitando um recurso específico usando HTTP, e o servidor responde com o conteúdo desse recurso ou uma mensagem de erro se não conseguir fazê-lo. O recurso é tipicamente um arquivo real no armazenamento do servidor, ou o resultado de um programa. Mas este não é sempre o caso e pode realmente ser muitas outras coisas.

Embora sua função primária seja servir conteúdo, HTTP também permite que os clientes enviem dados. Este recurso é usado para enviar formulários web, incluindo o envio de arquivos.

# Capítulo II

# Regras gerais

- Seu programa não deve travar sob nenhuma circunstância (mesmo que fique sem memória) ou terminar inesperadamente.
  Se isso ocorrer, seu projeto será considerado não funcional e sua nota será `0`.

- Você deve enviar um `Makefile` que compile seus arquivos de origem. Ele não deve realizar uma religação desnecessária.

- Seu `Makefile` deve conter pelo menos as regras:
  `$(NAME)`, `all`, `clean`, `fclean` e `re`.

- Compile seu código com `c++` e as flags `-Wall -Wextra -Werror`

- Seu código deve estar em conformidade com o **padrão C++ 98** e ainda deve compilar ao adicionar a flag `-std=c++98`.

- Certifique-se de aproveitar o máximo possível dos recursos do `C++` (por exemplo, escolha `<cstring>` em vez de `<string.h>`). Você está autorizado a usar funções `C`, mas sempre prefira suas versões `C++` se possível.

- Qualquer biblioteca externa e bibliotecas `Boost` são proibidas.

# Capítulo III

# Instruções para IA

## Contexto

Durante sua jornada de aprendizado, a IA pode auxiliar em diversas tarefas. Dedique tempo para explorar as várias capacidades das ferramentas de IA e como elas podem apoiar seu trabalho. No entanto, sempre as utilize com cautela e avalie criticamente os resultados. Seja código, documentação, ideias ou explicações técnicas, você nunca pode ter certeza absoluta de que sua pergunta foi bem formulada ou que o conteúdo gerado é preciso. Seus colegas são um recurso valioso para ajudá-lo a evitar erros e pontos cegos.

## Mensagem principal

- Use a IA para reduzir tarefas repetitivas ou tediosas.
- Desenvolva habilidades de prompt — tanto para codificação quanto para outras tarefas — que beneficiarão sua carreira futura.
- Aprenda como os sistemas de IA funcionam para melhor antecipar e evitar riscos comuns, vieses e questões éticas.
- Continue desenvolvendo habilidades técnicas e interpessoais trabalhando com seus colegas.
- Use apenas conteúdo gerado por IA que você entenda completamente e pelo qual possa se responsabilizar.

## Regras para o aluno:

- Você deve dedicar tempo para explorar as ferramentas de IA e entender como elas funcionam, para que possa usá-las eticamente e reduzir potenciais vieses.
- Você deve refletir sobre seu problema antes de criar o prompt — isso ajuda você a escrever prompts mais claros, detalhados e relevantes, usando vocabulário preciso.

- Você deve desenvolver o hábito de verificar, revisar, questionar e testar sistematicamente tudo o que for gerado por IA.
- Você deve sempre buscar revisão por pares — não confie apenas em sua própria validação.

## Resultados da fase:

- Desenvolver habilidades de prompt tanto para uso geral quanto para áreas específicas.
- Aumentar sua produtividade com o uso eficaz de ferramentas de IA.
- Continuar fortalecendo o pensamento computacional, resolução de problemas, adaptabilidade e colaboração.

## Comentários e exemplos:

- Você encontrará regularmente situações — exames, avaliações e mais — onde você deve demonstrar compreensão real. Esteja preparado, continue desenvolvendo suas habilidades técnicas e interpessoais.
- Explicar seu raciocínio e debater com colegas frequentemente revela lacunas em sua compreensão. Priorize a aprendizagem colaborativa.
- As ferramentas de IA geralmente carecem do seu contexto específico e tendem a fornecer respostas genéricas. Seus colegas, que compartilham seu ambiente, podem oferecer insights mais relevantes e precisos.
- Onde a IA tende a gerar a resposta mais provável, seus colegas podem fornecer perspectivas alternativas e nuances valiosas. Conte com eles como um ponto de verificação de qualidade.

**✓ Boa prática:**

> Pergunto à IA: “Como testo uma função de ordenação?” Ela me dá algumas ideias. Eu as testo e reviso os resultados com um colega. Nós refinamos a abordagem juntos.

**✗ Má prática:**

> Peço à IA para escrever uma função inteira, copio e colo no meu projeto. Durante a avaliação por pares, não consigo explicar o que ela faz ou porquê. Perco credibilidade — e reprovo no meu projeto.

**✓ Boa prática:**

Uso a IA para ajudar a projetar um analisador sintático. Então, analiso a lógica com um colega. Detectamos dois erros e reescrevemos juntos — melhor, mais limpo e totalmente compreendido.

**✗ Má prática:**

Deixo o Copilot gerar meu código para uma parte importante do meu projeto. Ele compila, mas não consigo explicar como ele lida com pipes. Durante a avaliação, não consigo justificar e reprovo no meu projeto.

# Capítulo IV

# Parte obrigatória

| | |
|---|---|
| **Nome do programa** | webserv |
| **Arquivos para entregar** | Makefile, *.{h, hpp}, *.cpp, *.tpp, *.ipp, arquivos de configuração |
| **Makefile** | NAME, all, clean, fclean, re |
| **Argumentos** | [Um arquivo de configuração] |
| **Funções externas autorizadas** | Toda a funcionalidade deve ser implementada em C++ 98.<br>execve, pipe, strerror, gai_strerror, errno, dup, dup2, fork, socketpair, htons, htonl, ntohs, ntohl, select, poll, epoll (epoll_create, epoll_ctl, epoll_wait), kqueue (kqueue, kevent), socket, accept, listen, send, recv, chdir, bind, connect, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl, close, read, write, waitpid, kill, signal, access, stat, open, opendir, readdir e closedir. |
| **Libft autorizada** | n/a |
| **Descrição** | Um servidor HTTP em C++ 98 |

Você deve escrever um servidor HTTP em C++ 98.

Seu executável deve ser executado da seguinte forma:

```
./webserv [arquivo de configuração]
```

Mesmo que poll() seja mencionado no subject e na régua de avaliação, você pode usar qualquer função equivalente, como select(), kqueue() ou epoll().

Por favor, leia os RFCs que definem o protocolo HTTP e execute testes com telnet e NGINX antes de começar este projeto.
Embora você não seja obrigado a implementar todos os RFCs, lê-los ajudará você a desenvolver os recursos necessários.
O HTTP 1.0 é sugerido como um ponto de referência, mas não é obrigatório.

## IV.1 Requisitos

- Seu programa deve usar um arquivo de configuração, fornecido como um argumento na linha de comando, ou disponível em um caminho padrão.
- Você não pode `execve` outro servidor web.
- Seu servidor deve permanecer não bloqueante em todos os momentos e lidar adequadamente com desconexões de clientes quando necessário.
- Ele deve ser não bloqueante e usar apenas **1** `poll()` (ou equivalente) para todas as operações de I/O entre os clientes e o servidor (incluindo listen).
- `poll()` (ou equivalente) deve monitorar a leitura e a escrita simultaneamente.
- Você nunca deve fazer uma operação de leitura ou escrita sem passar por `poll()` (ou equivalente).
- Verificar o valor de `errno` para ajustar o comportamento do servidor é estritamente proibido após realizar uma operação de leitura ou escrita.
- Você não é obrigado a usar `poll()` (ou equivalente) antes de read() para recuperar seu arquivo de configuração.

Como você tem que usar descritores de arquivo não bloqueantes, é possível usar funções read/recv ou write/send sem poll() (ou equivalente), e seu servidor não ficaria bloqueando.
Mas isso consumiria mais recursos do sistema.
Assim, se você tentar read/recv ou write/send em qualquer descritor de arquivo sem usar poll() (ou equivalente), sua nota será 0.

- Ao usar poll() ou qualquer chamada equivalente, você pode usar todas as macros ou funções auxiliares associadas (por exemplo, FD_SET para select()).
- Uma solicitação ao seu servidor nunca deve travar indefinidamente.
- Seu servidor deve ser compatível com **navegadores web** padrão de sua escolha.
- NGINX pode ser usado para comparar cabeçalhos e comportamentos de resposta (preste atenção às diferenças entre as versões HTTP).
- Seus códigos de status de resposta HTTP devem ser precisos.
- Seu servidor deve ter **páginas de erro padrão** se nenhuma for fornecida.
- Você não pode usar fork para nada além de CGI (como PHP ou Python, e assim por diante).

- Você deve ser capaz de **servir um site totalmente estático**.
- Os clientes devem ser capazes de **enviar arquivos**.
- Você precisa de pelo menos os métodos `GET`, `POST` e `DELETE`.
- Teste seu servidor para garantir que ele permaneça disponível em todos os momentos.
- Seu servidor deve ser capaz de ouvir várias portas para fornecer conteúdo diferente (ver *Arquivo de configuração*).

`Escolhemos deliberadamente oferecer apenas um subconjunto do HTTP RFC. Neste contexto, o recurso de host virtual é considerado fora do escopo. Mas você pode implementá-lo se quiser.`

## IV.2 Apenas para MacOS

Como o macOS lida com write() de forma diferente de outros sistemas operacionais baseados em Unix, você está autorizado a usar fcntl(). Você deve usar descritores de arquivo no modo não bloqueante para obter um comportamento semelhante ao de outros sistemas operacionais Unix.

No entanto, você só pode usar fcntl() com as seguintes flags: F_SETFL, O_NONBLOCK e FD_CLOEXEC.
Qualquer outra flag é proibida.

## IV.3 Arquivo de configuração

Você pode se inspirar na seção 'server' do arquivo de configuração do NGINX.

No arquivo de configuração, você deve ser capaz de:

- Definir todos os pares interface:porta nos quais seu servidor irá ouvir (definindo vários sites servidos pelo seu programa).
- Configurar páginas de erro padrão.
- Definir o tamanho máximo permitido para os corpos de solicitação do cliente.
- Especificar regras ou configurações em um URL/rota (nenhum regex é necessário aqui), para um site, entre os seguintes:
  - Lista de métodos HTTP aceitos para a rota.
  - Redirecionamento HTTP.
  - Diretório onde o arquivo solicitado deve estar localizado (por exemplo, se o URL `/kapouet` estiver enraizado em `/tmp/www`, o URL `/kapouet/pouic/toto/pouet` procurará por `/tmp/www/pouic/toto/pouet`).
  - Ativar ou desativar a listagem de diretórios.

- Arquivo padrão a ser servido quando o recurso solicitado é um diretório.
- O envio de arquivos dos clientes para o servidor é autorizado e o local de armazenamento é fornecido.
- Execução de CGI, com base na extensão do arquivo (por exemplo, .php). Aqui estão algumas observações específicas sobre CGIs:
  * Você está se perguntando o que é um CGI?
  * Observe atentamente as variáveis de ambiente envolvidas na comunicação servidor web-CGI. A solicitação completa e os argumentos fornecidos pelo cliente devem estar disponíveis para o CGI.
  * Apenas lembre-se de que, para solicitações em partes, seu servidor precisa desagrupá-las, o CGI esperará `EOF` como o final do corpo.
  * O mesmo se aplica à saída do CGI. Se nenhum `content_length` for retornado do CGI, `EOF` marcará o final dos dados retornados.
  * O CGI deve ser executado no diretório correto para acesso a arquivos de caminho relativo.
  * Seu servidor deve suportar pelo menos um CGI (php-CGI, Python e assim por diante).

Você deve fornecer arquivos de configuração e arquivos padrão para testar e demonstrar que todos os recursos funcionam durante a avaliação.

Você pode ter outras regras ou informações de configuração em seu arquivo (por exemplo, um nome de servidor para um site se você planeja implementar hosts virtuais).

Se você tiver uma pergunta sobre um comportamento específico, pode comparar o comportamento do seu programa com o do NGINX. Fornecemos um pequeno testador. Usá-lo não é obrigatório se tudo funcionar bem com seu navegador e testes, mas pode ajudá-lo a encontrar e corrigir bugs.

A resiliência é fundamental. Seu servidor deve permanecer operacional em todos os momentos.

Não teste com apenas um programa. Escreva seus testes em uma linguagem mais adequada, como Python ou Golang, entre outras, mesmo em C ou C++ se você preferir.

# Capítulo V

# Parte bônus

Aqui estão alguns recursos adicionais que você pode implementar:

- Suporte a cookies e gerenciamento de sessão (forneça exemplos simples).
- Lidar com vários tipos de CGI.

A parte bônus só será avaliada se a parte obrigatória estiver totalmente concluída sem problemas. Se você não atender a todos os requisitos obrigatórios, sua parte bônus não será avaliada.

# Capítulo VI

# Entrega e avaliação por pares

Entregue seu projeto no seu repositório `Git` como de costume. Apenas o conteúdo do seu repositório será avaliado durante a defesa. Certifique-se de verificar novamente os nomes de seus arquivos para garantir que estejam corretos.

Durante a avaliação, uma breve **modificação do projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa visa verificar sua compreensão real de uma parte específica do projeto. A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual), e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.
Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, alvo, etc.) serão especificados nas **diretrizes de avaliação** e podem variar de uma avaliação para outra para o mesmo projeto.

```
16D85ACC441674FBA2DF65190663F42A3832CEA21E024516795E1223BBA77916734D1
26120A16827E1B16612137E59ECD492E46EAB67D109B142D49054A7C281404901890F
619D682524F5
```