

# Redes de Computadores – 2011

## Primeiro Trabalho

### Servidor HTTP

Prof. Ronaldo Alves Ferreira

## 1 Descrição do Trabalho

Neste trabalho você implementará um servidor HTTP de acordo com a RFC 2616. O servidor deve permitir que clientes HTTP (Firefox, IE, etc.) se conectem ao servidor e façam downloads de arquivos. A implementação deverá ser em C ou C++.

Um cliente HTTP envia uma requisição GET para o servidor para recuperar um arquivo. A sintaxe geral da requisição é dada abaixo:

```
GET <sp> <Documento> <sp> HTTP/1.1 <crLf>
{Outras informações de cabeçalho}*
<crLf>
```

A função do servidor HTTP é analisar a requisição acima, identificar o arquivo sendo solicitado e enviar o arquivo pela rede para o cliente. Entretanto, antes de enviar o documento, o servidor HTTP deve enviar um cabeçalho de resposta para o cliente. Uma resposta típica de um servidor HTTP pode ser vista abaixo, neste caso o arquivo foi encontrado no servidor:

```
HTTP/1.1 <sp> 200 <sp> Document <sp> follows <crLf>
Server: <sp> <Server-Type> <crLf>
Content-type: <sp> <Document-Type> <crLf>
{Outras informações de cabeçalho}*
<crLf>
<Dados do Documento>
```

Server-Type: identifica o fabricante/versão do servidor. Você pode colocar o valor FACOM-MCC-RC-2011/1.0.

Document-Type: indica o tipo de documento sendo enviado. O valor deve ser “text/html” para um documento html, “image/gif” para um arquivo gif, “text/plain” para arquivo texto comum, etc.

Se o arquivo solicitado não puder ser encontrado no servidor, o servidor deve enviar um cabeçalho indicando o erro. Uma resposta típica seria:

```
HTTP/1.1 <sp> 404 File Not Found <crLf>
Server: <sp> <Server-Type> <crLf>
Content-type: <sp> <Document-Type> <crLf>
<crLf>
<Error Message>
```

Document-Type: indicat o tipo de documento. Neste caso, o valor deve ser “text/plain.”

Error Message: é uma descrição do erro em texto comum (por exemplo, “Could not find the specified URL”).

## 2 Funcionamento Básico

Você deverá implementar dois modos de operação do servidor de acordo com os parâmetros passados na linha de comando.

- **httpd -f porta**

neste caso, o servidor deve criar um novo processo para cada nova conexão. Você deve criar um tratador de sinais que trate o sinal SIGCHLD para evitar o aparecimento de processos zumbis.

- **http -t N porta**

neste caso, você deve criar uma thread principal que irá colocar o socket principal em modo de escuta (listen) e criará um pool de N threads. Cada thread deve ter um loop infinito que chama a função `accept()` e processa a requisição do cliente. A idéia é ter um servidor interativo rodando em cada thread. Para evitar que várias threads chamem `accept()` ao mesmo tempo, crie um semáforo para coordenar o acesso a essa função.

Nos exemplos acima, porta indica o número da porta em que o servidor receberá conexões. Se nenhuma porta for especificada, o servidor deve escutar na porta 8080.

### 2.1 Diretórios

Você deve adicionar ao seu servidor a capacidade de navegação em diretórios. Se o documento requisitado for um diretório, seu servidor HTTP deve retornar um documento HTML com links para os arquivos/diretórios presentes no diretório. Você deve permitir navegação recursiva de diretórios. Estudem as páginas de manual das funções *opendir* e *readdir*.

### 2.2 CGI-BIN

Você deve implementar a interface de comunicação de processos CGI-BIN. Quando uma requisição for do tipo:

```
GET <sp> /cgi-bin/<script>?{<var>=<val>}*{<var>=<val>}<sp>HTTP/1.1<crlf>
{Outras informações de cabeçalho}<crlf>*
<crlf>
```

o processo que está executando a requisição chamará a função *execv*, após chamar a função *fork*, para executar o programa em *cgi-bin/<script>*. Seu programa deve tratar o sinal SIG-CHLD.

Existem duas maneiras de se passar variáveis para scripts cgi-bin: o método GET e o método POST. Você deve implementar somente o método GET. No método GET, a cadeia de variáveis é passada para o programa *<script>* como uma variável de ambiente QUERY\_STRING. O programa *<script>* é quem deve decodificar esta cadeia. A saída do programa *<script>* deverá ser enviada para o cliente HTTP.

## 2.3 Conexões Persistentes

O seu servidor deve tratar solicitações de conexões persistentes. Caso o navegador envie a diretiva de conexão persistente, o seu servidor deve manter o socket aberto e novas requisições devem ser tratadas.

## 2.4 Funcionalidades Adicionais

Os trabalhos serão avaliados comparativamente. Portanto, não há limite para funcionalidades adicionais, você pode implementar outras funcionalidades não descritas neste trabalho, como, por exemplo, o método POST. Leia a RFC 2616 e decida as funcionalidades adicionais a serem implementadas.

# 3 Fontes para Consultas

Além da RFC mencionada acima, consulte as páginas de manual para as funções da API socket discutidas em sala. Outras funções importantes são: *fork*, *signal*, *sem\_wait*, *sem\_post*, *kill*, *pthread\_create*.

Guia para programação utilizando sockets:  
<http://beej.us/guide/bgnet/>

# 4 Entrega do Trabalho

O trabalho deve ser entregue no dia 1 de setembro. A entrega do trabalho consistirá em uma demonstração das funcionalidades do seu servidor no laboratório. Você deve preparar uma breve apresentação e um roteiro para a demonstração. Você deve indicar explicitamente as funcionalidades que foram implementadas e as que não foram. Na demonstração, você deve incluir casos que demonstrem claramente as funcionalidades implementadas, como, por exemplo, o tratamento de várias conexões simultâneas, conexões persistentes, etc. Além disso, você deve

entregar um breve relatório descrevendo o trabalho. Neste relatório, você deve incluir uma breve introdução, decisões de implementação, funcionalidades não implementadas, problemas enfrentados na implementação, etc. O relatório deve ser entregue em um arquivo PDF.

O trabalho pode ser feito em grupos de no máximo três alunos. Casos de plágio serão tratados com rigor. Caso você faça o trabalho em grupo, submeta apenas um trabalho e identifique os componentes do grupo no relatório e no código fonte.

## 5 Avaliação

Além da correção do programa, o professor fará perguntas durante a apresentação do trabalho. Durante a apresentação, o grupo deverá explicar o funcionamento do programa e responder a perguntas relativas ao projeto.