O Serviço HTTP

O objetivo desta aula é compreender o funcionamento básico do serviço Web, explorando suas principais possibilidades de configuração.

Conceitos básicos

A World-Wide-Web é baseada em um relacionamento cliente/servidor, no qual o cliente é um navegador, que tem como característica principal permitir a visualização de diversas mídias, além do padrão básico HTML. O servidor WWW é normalmente acessado através do protocolo HTTP, que permite o tráfego das informações desejadas através de uma conexão TCP.

O protocolo HTTP permite o transporte dos mais diversos tipos de arquivos, como textos HTML, imagens, documentos em formatos específicos, applets Java, etc. Este serviço pode ser organizado em uma arquitetura *two-tier*, onde são servidas páginas web estáticas, ou *three-tier*, onde o servidor Web busca informações de outras fontes (bancos de dados ou outros serviços) para construir as páginas solicitadas, dinamicamente.

URLs

Os recursos disponíveis na Web são acessados através de referências denominadas URLs - *Uniform Resource Locators*. Uma URL define completamente um recurso na Internet, e tem os seguintes componentes:

- protocolo: forma de comunicação usada para acessar o recurso. Pode ser FTP, HTTP, LDAP, ...
- servidor: nome da máquina que provê o serviço desejado.
- domínio: complemento do nome do servidor, especificando em que rede se encontra.
- porta: porta a conectar no servidor. Pode ser omitida se for a porta default para o serviço desejado.
- caminho: localização do recurso dentro do servidor.
- recurso: nome do recurso dentro do servidor.

Considerando as seguintes URLs de exemplo, seus componentes têm os seguintes significados:

URL	protocolo	servidor	domínio	porta	caminho	recurso
http://www.uol.com.br	http	www	uol.com.br	default (80)	/	index.html
http://espec.ppgia.pucpr.br:4700/~maziero/teste.gif	http	espec	ppgia.pucpr.br	4700	/~maziero/	teste.gif
ftp://ftp.pucpr.br/pub/cnpq/lattes/atencao.txt	ftp	ftp	pucpr.br	default (21)	/pub/cnpq/lattes/	atencao.txt

O protocolo HTTP

O HTTP - *HyperText Transport Protocol* (versão 1.0 definida na RFC 1945 e versão 1.1 na RFC 2616) - é um protocolo bastante simples, com comandos em formato texto, transportado através de uma conexão TCP (default na porta 80), que tem o objetivo de enviar requisições a servidores, na forma de URLs e obter respostas na forma de conteúdos especificados pelo padrão MIME.

Um exemplo típico de solicitação em HTTP, enviada pelo navegador ao servidor Web, é indicado a seguir:

```
GET /teste.html HTTP/1.0
Host: 20.0.0.29:8081
Accept: text/html, text/plain, image/*, application/pdf, text/sgml, */*
```

```
Accept-Encoding: gzip, compress
Accept-Language: en
User-Agent: Lynx/2.8.5dev.7 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.7
```

Observe que a última linha da solicitação é vazia (sem conteúdo). Ao receber uma linha vazia, o servidor compreende que a solicitação terminou e começa a preparar uma resposta. A resposta do servidor a essa solicitação, enviada de volta ao navegador, poderia ser a seguinte:

```
HTTP/1.1 200 OK
Date: Tue, 01 Jun 2004 01:41:08 GMT
Server: Apache/2.0.40 (Red Hat Linux)
Last-Modified: Tue, 02 Apr 2002 19:49:45 GMT
ETag: "1c5e4-35-aba6c840"
Accept-Ranges: bytes
Content-Length: 53
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<html>
<body>
<h1>Funcionou !</h1>
</body>
</html>
```

Deve-se observar que a resposta do servidor é dividida em duas seções, separadas por uma linha vazia. A primeira seção é denominada *cabeçalho* (*header*) e contém informações do servidor sobre a URL solicitada: status da resposta, tipo e tamanho da resposta, configuração do servidor, etc. A segunda seção, denominada *corpo* (*body*), contém o recurso propriamente dito, solicitado pelo navegador.

Os principais métodos (comandos) do protocolo HTTP são:

Método	**Descrição**
GET	busca um objeto definido por uma URL do servidor
PUT	indica que os dados no corpo da consulta devem ser armazenados na URL especificada
POST	cria um novo objeto ligado ao objeto especificado na URL. Uma URL será alocada pelo servidor e retornada ao cliente. O conteúdo do novo objeto é o corpo de dados da consulta
HEAD	similar ao método GET, mas retorna somente o cabeçalho da resposta do servidor
DELETE	solicita ao servidor a remoção da informação correspondente à URL indicada

Os principais status de retorno de métodos HTTP, informados pelo servidor no cabeçalho da resposta, são indicados abaixo (veja a RFC 2616 para a lista completa):

Categoria	**Descrição**	**Código**	**Descrição**
1 - -	Informational	100	Continue
		101	Switching protocols
2 - -	Success	200	Ok
		201	Created
		202	Accepted
		203	Non-authoritive information
3 - -	Redirection	300	Multiple choices
		301	Moved permanently
		302	Moved temporarily
4 - -	Client error	400	Bad request
		401	Unauthorized

Categoria	Descrição	Código	Descrição
		402	Payment required
		403	Forbidden
		404	Not found
		415	Unsupported media type
		426	Upgrade Required
5 - -	Server error	500	Internal server error
		501	Not implemented
		502	Bad gateway
		503	Service unavailable
		505	HTTP version not supported

Os tipos MIME usados para identificar o tipo do conteúdo da resposta do servidor (na linha `Content-type` do cabeçalho da resposta) seguem um padrão especificado nas RFCs 2045 a 2048. Alguns exemplos de tipos MIME comuns são (veja uma lista mais extensa no arquivo `/etc/mime.types`):

Tipo MIME	Significado
`text/plain`	arquivo de texto puro
`text/html`	arquivo de texto em formato HTML
`image/gif`	arquivo de imagem em formato GIF
`image/jpeg`	arquivo de imagem em formato JPEG
`application/pdf`	arquivo de aplicação em formato PDF
`video/quicktime`	arquivo de vídeo em formato QuickTime

O servidor Apache

Funcionamento básico do servidor

O servidor WWW que será usado neste curso é o http://www.apache.org Apache, certamente o mais utilizado atualmente no mundo inteiro. Existem diversos outros servidores WWW para UNIX, gratuitos ou comerciais, mas nenhum tem a base instalada do Apache, e certamente poucos têm sua flexibilidade, desempenho e segurança. Ele é gratuito e está presente em quase todas as distribuições Linux.

O servidor Apache é executado na forma de um processo *daemon* chamado `httpd`, que geralmente é lançado durante o processo de carga (*boot*) da máquina. Ao ser lançado, o *daemon* mestre cria um conjunto de *daemons* escravos (processos filhos) para atender as requisições de páginas WWW. Isso pode ser visto na seguinte listagem de processos, obtida através do comando `ps -aux`:

```
$ ps aux|grep -i http
...
root      1084  0.0  0.3 258800 15112 ?        S<s  Jun13   0:00 /usr/sbin/httpd
apache    4191  0.0  0.4 266960 17952 ?        S<   Jun30   0:01 /usr/sbin/httpd
apache    4192  0.0  0.3 264448 15336 ?        S<   Jun30   0:01 /usr/sbin/httpd
apache    4193  0.0  0.3 264396 15276 ?        S<   Jun30   0:01 /usr/sbin/httpd
apache    5611  0.0  0.3 263180 14000 ?        S<   Jul01   0:00 /usr/sbin/httpd
apache    6476  0.0  0.3 263212 14084 ?        S<   Jun29   0:01 /usr/sbin/httpd
apache    6477  0.0  0.3 264312 15344 ?        S<   Jun29   0:01 /usr/sbin/httpd
apache    6478  0.0  0.4 266876 17884 ?        S<   Jun29   0:01 /usr/sbin/httpd
apache    6479  0.0  0.2 259576 10028 ?        S<   Jun29   0:01 /usr/sbin/httpd
apache    9857  0.0  0.4 267164 18140 ?        S<   Jul02   0:00 /usr/sbin/httpd
apache   18931  0.0  0.3 266472 16068 ?        S<   Jun29   0:01 /usr/sbin/httpd
```

```
...
```

A estrutura em vários processos do servidor Apache tem vários objetivos:

- **Maior segurança**, pois os processos filhos executam sob a identidade de um usuário especial (normalmente `apache`, `httpd` ou `nobody`), com menos privilégios que o administrador do sistema (*root*);
- **Melhor desempenho**, pois cada processo filho pode tratar uma requisição diferente;
- **Maior robustez**, pois cada filho trata um número máximo de requisições e em seguida encerra sua execução, sendo substituido por um novo filho.

Os principais arquivos e diretórios usados pelo Apache (na distribuição Linux RedHat) são:

- `/etc/httpd/conf/httpd.conf` : principal arquivo de configuração
- `/etc/mime.types` : tipos MIME e extensões de arquivos conhecidos pelo servidor
- `/var/log/httpd/` : arquivos de logs de acessos e erros
- `/var/www/` : diretório principal das páginas locais oferecidas pelo servidor
- `/var/www/html/` : páginas default do servidor (index.html)
- `/var/www/cgi-bin/` : programas executáveis no padrão CGI
- `/usr/lib/httpd/modules/` : módulos dinâmicos usados pelo servidor (plugins)
- `$HOME/public_html/` : páginas pessoais de cada usuário

A localização dos diretórios pode ser alterada editando-se o arquivo de configuração.

É importante observar que, como os *daemons* do servidor executam sob a identidade de um usuário menos privilegiado, todos os arquivos e diretórios acessíveis aos *daemons* (páginas HTML, CGIs, imagens) devem ter permissões de acesso para terceiros (*others*).

Configuração básica

O principal arquivo de configuração do Apache é o `/etc/httpd/conf/httpd.conf`. É um arquivo texto bastante extenso, mas muito documentado e de configuração relativamente simples. Um dos pontos mais importantes desse arquivo é a definição dos serviços acessíveis, que segue uma forma padrão. Eis o exemplo de definição do diretório raiz das páginas locais:

```
DocumentRoot "/var/www/html"

<Directory "/var/www/html">

#   Defines how this directory can be accessed by clients.
#   This may also be "None", "All", or any combination of "Indexes",
#   "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".

    Options Indexes FollowSymLinks ExecCGI

#   This controls which options the .htaccess files in directories can
#   override. Can also be "All", or any combination of "Options", "FileInfo",
#   "AuthConfig", and "Limit"

    AllowOverride None

#   Controls who can get stuff from this server.

    Order allow,deny
    Allow from all
```

```
</Directory>
```

Controle de acesso a páginas

O controle de acesso à páginas pelos clientes é feito com base em diretórios. Além das definições presentes no arquivo de configuração global (cláusulas `order`, `allow` e `deny` dentro das definições dos serviços), restrições de acesso podem ser definidas em arquivos separados, para cada diretório. Para isso, deve-se criar um arquivo escondido (com nome `.htaccess` por default) no diretório, contendo as definições de acessibilidade para aquele diretório e seus sub-diretórios. Vejamos alguns exemplos:

Para controlar o acesso ao diretório baseado em hosts:

```
order deny,allow
deny from all
allow from .ppgia.pucpr.br
allow from 200.123.123.123
allow from 10.1
allow from 10.2.0.0/255.255.0.0
allow from 10.3.0.0/16
```

Para controlar o acesso baseado em usuários/senhas (o arquivo de usuários/senhas deve ser criado através da aplicação `htpasswd`):

```
AuthUserFile /home/prof/maziero/passwords.txt
AuthName "Página dos usuários autorizados"
AuthType Basic
require valid-user
```

Informações mais detalhadas sobre controle de acesso no Apache podem ser encontradas aqui.

Para ter mensagens de erro personalizadas:

```
ErrorDocument 404 /notfound.html
```

Para usar outra página default para um determinado diretório:

```
DirectoryIndex filename.html
```

É importante ressaltar que o uso dos arquivos `.htaccess` é regulado pela diretiva de configuração `AllowOverride`, definida para cada diretório no arquivo de configuração principal.

Uma série de opções interessantes, incluindo o redirecionamento automático de páginas, *mirroring*, etc, podem ser efetuadas através do módulo Apache Rewriting, cuja explanação foge ao escopo deste curso.

Páginas pessoais

O Apache considera que um recurso designado no formato "`/~usuário/caminho/recurso`" se localiza dentro do diretório pessoal (`$HOME`) do usuário indicado, em um sub-diretório de informações públicas. Por default esse diretório tem o nome "`public_html`", e deve ficar dentro do diretório "`$HOME`" do usuário.

É importante observar que os diretórios contendo páginas pessoais devem estar acessíveis aos *daemons* do Apache. Como estes executam sob a identidade de um usuário não-*root*, as permissões de acesso aos diretórios

e arquivos das páginas pessoais devem autorizar o acesso a terceiros.

From:
https://wiki.inf.ufpr.br/maziero/ - **Prof. Carlos Maziero**

Permanent link:
https://wiki.inf.ufpr.br/maziero/doku.php?id=espec:servico_http

Last update: **2020/08/18 19:51**