Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

O ICMSC na World-Wide Web

MARIA ALICE SOARES DE CASTRO
EDSON DOS SANTOS MOREIRA

Nº 35

RELATÓRIOS TÉCNICOS DO ICMSC

São Carlos
Out./1995

SYSNO 0894016
DATA / /
ICMC - SBAB

Introdução

Este trabalho tem por objetivo apresentar aos estudantes e professores do ICMSC a World-Wide Web: seus fundamentos e organização.

Os usuários, entusiasmados com os recursos oferecidos pelas interfaces gráficas, por vezes desconhecem o que se passa por trás do simples fato de acionar o mouse sobre uma figura ou texto, e viajar através de telas em busca de um tema de seu interesse.

O que é a World-Wide Web (WWW, W3 ou Web)? Como as páginas são produzidas? O que acontece quando se aciona um "link" na tela?

Este texto reúne as informações introdutórias desse sistema distribuído. Uma idéia geral a respeito de bancos de dados distribuídos e redes de computadores antecipam a apresentação da WWW, para melhor compreensão de sua arquitetura. É apresentada também a implementação do servidor WWW do ICMSC.

Complementa este trabalho um tutorial HTML instalado em nosso servidor WWW.

O ICMSC NA WORLD-WIDE WEB

Maria Alice Soares de Castro (*masc@icmsc.sc.usp.br*)
Edson dos Santos Moreira (*edsmorei@icmsc.sc.usp.br*)

Departamento de Ciências de Computação e Estatística
Instituto de Ciências Matemáticas de São Carlos - USP
Av. Dr. Carlos Botelho, 1465
Caixa Postal 668
São Carlos - SP
13560-970

Outubro de 1995

Índice

1 - Bancos de Dados Distribuídos **1**
1.1 - Características 1
1.2 - Principais Vantagens 2
1.3 - Principais Problemas 2
1.4 - Nós de Computação 2
1.5 - Computação de Transações Distribuídas 3
1.6 - Consultas 3
1.6.1 - Fatores Críticos no Processamento de Consultas 4
1.7 - Distribuição de Arquivos e Diretórios 4
1.7.1 - O Problema da Replicação de Dados 4
1.7.2 - Distribuição de Arquivos 5
1.7.3 - Distribuição de Diretórios 5
1.8 - Modelo Orientado a Objeto 5
1.8.1 - Sistemas Multimídia 5

2 - Redes de Computadores: A Internet **7**
2.1 - Interconexão de Redes (Internetworking) 7
2.1.1 - Comunicação em Redes 8
2.1.1.1 - Chaveamento por Circuito (circuit switching) 8
2.1.1.2 - Chaveamento por Pacote (packet switching) 8
2.2 - O Conjunto de Protocolos TCP/IP 9
2.2.1 - IP (Internet Protocol): Entrega Desconectada de Datagramas 10
2.2.1.1 - O Datagrama Internet 11
2.2.1.2 - Endereços Internet 12
2.2.2 - TCP (Transmission Control Protocol): Serviço Confiável de Transporte de Dados 12
2.2.2.1 - O Conceito de Janelas Deslizantes 13
2.2.2.2 - Portas e Conexões 13
2.3 - O Modelo de Interação Cliente-Servidor 13
2.3.1 - O Modelo Cliente-Servidor 13
2.3.2 - Comunicação Interprocessos 14
2.3.3 - A Complexidade dos Servidores 15

3 - A World-Wide Web **17**
3.1 - O que é a WWW? 17
3.1.1 - Acesso à World-Wide Web 17
3.2 - Arquitetura 18
3.2.1 - Browsers 18
3.3 - Fundamentos da WWW 19
3.3.1 - Universal Resource Identifier 19
3.3.1.1 - URL - Uniform Resource Locator 19
3.3.1.2 - Esquemas URL 20
3.3.1.3 - URLs para Servidores HTTP 21
i. URLs Relativos 21
ii. MIME - Multipurpose Internet Mail Extensions 21
iii. Passagem de Parâmetros para o Servidor 22
3.3.2 - Protocolo de Transferência de Hipertexto 22
3.3.2.1 - Transações 23

3.3.2.2 - Requisição 23
i. Métodos 23
ii. Request Header Fields 24
3.3.2.3 - Resposta 26
i. Response Header Fields 28
ii. Entidade 28
3.3.2.4 - Negociação de Conteúdo 30
3.3.2.5 - Considerações de Segurança 30
3.3.3 - Linguagem de Formatação de Hipertexto 31
3.3.3.1 - A Linguagem HTML 32
3.3.3.2 - Identificação de Documentos HTML 32
3.3.3.3 - Elementos de HTML 32
3.3.3.4 - Elemento `<HTML>` 33
3.3.3.5 - Elemento `<HEAD>` 33
3.3.3.6 - Elemento `<BODY>` 34
i. Cabeçalhos 35
ii. Parágrafos 35
iii. Textos Especiais 35
iv. Listas 36
v. Formatação de Caracteres 37
vi. Ligação a outros Documentos 39
vii. Quebra de Linha 40
viii. Linha Horizontal 40
ix. Inserção de Imagens 40
x. Tabelas 41
xi. Formatação de Equações 41
xii. Imagens Externas, Sons e Animações 41
xiii. Comentários em HTML 41
xiv. Formulários 42
3.4 - A Linguagem HTML 3.0 43
3.5 - O Servidor WWW 45
3.5.1 - Arquivos do Servidor 45
3.5.2 - Common Gateway Interface 46
3.5.2.1 - Usando `ISINDEX` para Buscas no Servidor 46
3.5.3 - WWW e Outros Sistemas 47
3.5.4 - Hyper-G: Além da WWW 48

4 - O Servidor WWW do ICMSC 49
4.1 - Árvore do Servidor 49
4.2 - Grafos de Navegação 50
4.2.1 - Distribuição dos Documentos 50
4.3 - Diretrizes para as Páginas Hipertexto 51
4.3.1 - Edição 51
4.4 - Homepages 52

Referências na WWW 53
Referências Bibliográficas 54

1 - BANCOS DE DADOS DISTRIBUÍDOS

INTRODUÇÃO

Os Bancos de Dados Centralizados surgiram no final dos anos 60, apresentando contínuo desenvolvimento desde então. A implementação desses sistemas sempre considerou que os dados ficariam em um mainframe, sendo utilizados somente por usuários que tivessem acesso a ele. Entretanto, em virtude das características de uma organização, os usuários podem desejar ter um pedaço do banco de dados "à mão", em seus departamentos - o que pode levar o sistema de informação a se desintegrar em um grande número de diferentes sistemas, sendo que um usuário não tem acesso ao banco de dados do outro.

A estratégia alternativa é possibilitar uma distribuição geográfica planejada do banco de dados, na qual os diversos computadores que mantêm partes do banco de dados estão interconectados por uma rede de comunicação; ou seja: implementar um sistema de banco de dados distribuído.

1.1 - CARACTERÍSTICAS

Uma definição comum de bancos de dados distribuídos é: *uma coleção de dados logicamente integrados que se apresentam fisicamente espalhados sobre os nós de uma rede de computadores.*

Esta definição não é muito precisa, mas salienta duas características importantes de um banco de dados distribuído:

*1. **Distribuição** - os dados não estão residentes no mesmo lugar*; distinguimos, assim, um banco de dados distribuído de um banco de dados centralizado.

*2. **Correlação lógica** - os dados apresentam algumas propriedades comuns*; isto distingue um banco de dados distribuído de um conjunto de bancos de dados ou arquivos locais que estejam residentes em diferentes pontos de uma rede de computadores.

Uma definição que considera características específicas de bancos de dados distribuídos é:

> ***"Um banco de dados distribuído é uma coleção de dados que são distribuídos por diferentes computadores em uma rede. Cada ponto da rede tem capacidade de processamento autônomo e pode desempenhar aplicações locais. Cada ponto também participa na execução de pelo menos uma aplicação global, que requer acesso a dados de diversos pontos usando um subsistema de comunicação."*** [Ceri_84]

No contexto deste trabalho, temos a World-Wide Web como um sistema distribuído de bancos de dados que se comunicam sobre a Internet.

Três características importantes distinguem bancos de dados distribuídos dos tradicionais:

• ***Estrutura de controle hierárquico***, baseada em um Administrador do Banco de Dados Global, responsável por todo o banco, e em Administradores dos Bancos de Dados Locais, responsáveis por seus bancos de dados respectivos. O grau de autonomia local pode variar da completa autonomia sem qualquer DBA[1] até controle quase totalmente centralizado.

• ***Independência de dados***, tal como nos bancos de dados comuns, *e transparência de distribuição*: os programas podem ser escritos como se o banco de dados não fosse distribuído. A correção dos programas não é afetada pelo movimento de dados de um ponto a outro, embora sua velocidade de execução o seja.

• ***Redundância controlada de dados*** é uma característica desejável, pois:

---

[1] DBA: "*Data Base Administrator*" ou Administrador de Banco de Dados

1 - a disponibilidade do sistema pode ser aumentada: uma falha num ponto não pára a execução de aplicações em outros pontos. Havendo diversas cópias de um item, a recuperação pode ser feita em qualquer cópia;

2- o sistema se torna mais eficiente quando os dados mais freqüentemente acessados em um local são armazenados no próprio local. Desse modo, há diminuição da concorrência, do volume e custo de transmissão.

1.2 - PRINCIPAIS VANTAGENS

• *Organização:* este sistema tem aplicação prática nas organizações descentralizadas compostas por um número significativo de subunidades, tais como bancos, universidades, etc.

• *Interconexão de bancos de dados existentes:* o banco de dados distribuído é criado a partir dos bancos de dados pré-existentes (bottom-up). Este processo implica num certo grau de reestruturação local que requer esforço muito menor do que o necessário para a criação de um banco de dados centralizado.

• *Crescimento gradual:* em uma organização que cresce acrescentando novas unidades relativamente autônomas (p. ex., filiais), um banco de dados distribuído suporta um aumento gradual com um mínimo impacto nas unidades já existentes. Na abordagem centralizada, o crescimento sempre causa grande impacto, por ser difícil prever as futuras dimensões do sistema logo no início do projeto.

• *Redução da sobrecarga de comunicação:* em um banco de dados geograficamente distribuído, o fato de muitas aplicações serem locais reduz claramente a sobrecarga de comunicação em relação a um banco de dados centralizado.

• *Desempenho:* a existência de diversos processadores autônomos resulta em aumento do desempenho através de um alto grau de paralelismo. A carga é compartilhada entre os processadores, sendo evitados gargalos críticos tais como a comunicação em rede ou serviços comuns a todo o sistema.

• *Confiabilidade/Disponibilidade:* o sistema pode continuar a ser acessível mesmo no caso de falhas locais ou de ligação. A capacidade de processamento autônomo dos diferentes pontos garante uma propriedade de degradação suave ("graceful degradation"). O efeito de cada falha é confinado àquelas aplicações que usam os dados do ponto falho, e a queda do sistema completo é rara.

1.3 - PRINCIPAIS PROBLEMAS

• Complexidade: a distribuição dos arquivos e diretórios dentro da rede deve satisfazer de forma eficiente as necessidades dos usuários; o processamento concorrente de transações deve preservar a integridade do sistema;

• são caros em termos de desenvolvimento de hardware e software, operação e manutenção;

• falta de experiência em desenvolvimento desse tipo de banco de dados: sendo um campo relativamente novo, com mais protótipos e experimentos acadêmicos do que sistemas comerciais disponíveis, existem ainda muitas questões e detalhes a resolver satisfatoriamente. Além disso, problemas de segurança são intrínsecos a sistemas distribuídos em geral, porque redes de comunicação representam um ponto fraco em relação a proteção;

• falta de padrões de interoperabilidade: cada desenvolvedor cria seu próprio padrão, e somente com a necessidade dos usuários surge a preocupação de "traduzir" os dados de um padrão para outro, o que não é uma tarefa simples;

• são difíceis de serem mudados; pelo próprio fato de serem distribuídos, os Bancos de Dados Distribuídos permitem uma grande flexibilidade local, mas dificultam mudanças globais.

1.4 - NÓS DE COMPUTAÇÃO

Denomina-se *nó de computação* um computador localizado numa área da organização (*computer site*) com certas facilidades de processamento. Em cada um desses nós o software do Sistema de Gerenciamento de Banco de Dados Distribuído (SGBDD) consiste pelo menos de:

- sistema de gerenciamento de banco de dados local;
- sistema operacional;
- gerenciador de comunicação.

O gerenciador de comunicação permite a troca mútua de informação entre programas remotos. Este gerenciador inclui três níveis de controle:

- controle da transmissão física dos dados;
- controle da ligação lógica de mensagens entre processos;
- controle de transferência e arquivos.

1.5 - COMPUTAÇÃO DE TRANSAÇÕES DISTRIBUÍDAS

Chamamos transação a execução de uma seqüência de operações que acessa um banco de dados compartilhado. Há quatro propriedades denominadas "ACIDez de Transações" [Helal_94]:

- ***Atomicidade:*** as operações da transação ou são realizadas completamente, ou não são realizadas, mesmo na presença de falhas do sistema distribuído;
- ***Consistência:*** transações não devem cometer nenhuma violação das restrições de integridade do banco de dados;
- ***Isolamento:*** mudanças concorrentes são invisíveis (deve ser possível realizar mudanças em série);
- ***Durabilidade:*** as atualizações realizadas por transações são persistentes.

Transações atômicas são o meio de se obter integridade do banco de dados, porque asseguram que ou todas as ações que o transformam de um estado consistente a outro são realizadas, ou o estado consistente inicial é mantido. Existem dois fatores que ameaçam a atomicidade de transações: o primeiro é a falha de processos e computadores, e é tratada por recuperação; o segundo é a execução concorrente de transações e é tratado por controle de concorrência.

Há muitos aspectos a considerar em relação à computação e comunicação de uma transação distribuída. Citamos aqui dois modelos:

i. PROCESSOS E SERVIDORES - existe um processo servidor independente de transações; as transações requisitam serviços do servidor enviando mensagens de pedidos. Uma transação se amarra a um servidor dinamicamente, passando a ele o identificador da transação; o servidor pode servir transações de maneira intercalada. Em um mesmo nó de computação, duas alternativas são: criar um processo para servir cada requisição que chega, ou criar somente um processo para a primeira consulta e reter aquele processo durante toda a execução daquela transação.

ii. SESSÕES E DATAGRAMAS - a comunicação pode ocorrer através de sessões e/ou datagramas. Com um modelo de processos e servidores, sessões são mais convenientes se for mantido um processo para cada localidade em vez de se criar um novo processo para cada requisição. Isso cria um tipo de estrutura de computação distribuída estável que é mantida tanto tempo quanto se queira; neste caso, é significativo estabelecer a sessão quando o processo remoto é criado, e mantê-lo até que ele seja removido.

1.6 - CONSULTAS

Uma consulta é uma expressão em uma linguagem apropriada que define uma porção dos dados contidos na base de dados. Pode ser usada para definir a semântica de uma aplicação, ou para definir a função realizada por um programa que acessa a base de dados.

1.6.1 - FATORES CRÍTICOS NO PROCESSAMENTO DE CONSULTAS

Um dos problemas mais importantes no processamento de consultas num ambiente distribuído é a eficiência.

Dentre os fatores críticos que podem afetar essa eficiência temos, em primeiro lugar, a transferência de dados entre nós, que deve ser minimizada.

Em segundo lugar, temos o custo de transmissão de dados entre dois nós, que é significativamente maior do que o custo de processamento local e do que a transferência de dados entre os dispositivos de armazenamento de um nó. A distribuição do sistema torna possível a diminuição do tempo de resposta pelo paralelismo de processamento de consultas e subconsultas.

Por último, o tráfego da rede constitui outro fator crítico devido à tecnologia de comunicações de dados não ter atingido o mesmo nível da tecnologia de computadores, em termos de redução de custo e aumento de desempenho.

1.7 - DISTRIBUIÇÃO DE ARQUIVOS E DIRETÓRIOS

Um elemento importante num ambiente distribuído é o diretório global, que é o meio pelo qual o SGBDD determina que nós precisam ser acessados com a finalidade de atender uma consulta particular. Sua localização física (cetralizado ou distribuído) é importante no desempenho do sistema.

Fatores a considerar na alocação de arquivos e diretórios:

- freqüência de acesso
- custo de transmissão
- custo de armazenamento
- atividades inter-nós
- atividades intra-nós
- compartilhamento dos dados
- tamanho do diretório
- freqüência de atualização do diretório

1.7.1 - O PROBLEMA DA REPLICAÇÃO DE DADOS

O grau de replicação dos dados tem um peso fundamental na distribuição de arquivos e diretórios. Um banco de dados distribuído *não precisa ter*, necessariamente, dados redundantes: a redundância é desejável enquanto promove uma melhoria na disponibilidade e confiabilidade do sistema.

Um fator principal contra a replicação é a dificuldade de se garantir a consistência dos dados. [Helal_94] apresenta alguns métodos de controle de replicação que se propõem a manter a correção das cópias, ameaçadas por atualizações e falhas:

1- Atualizações - deseja-se atingir consistência mútua, assegurando que todas as cópias dos dados apresentem o mesmo valor;

2- falhas em nós - deve-se assegurar a consistência mútua sob condições de falha, incluindo falhas locais e particionamento da rede.

Devemos observar que, no caso específico da WWW, os dados são páginas de hipertexto que o servidor oferece aos usuários. O problema de atualização de dados é de certa forma restrito a modificações feitas pelos autores dos documentos, já que não é possível alterar uma página feita por outra pessoa em um computador remoto; se um mesmo servidor tiver mais de uma cópia de

determinadas páginas, numa mesma máquina ou em máquinas diferentes, o responsável pelo sistema deve se preocupar em manter todas as cópias atualizadas de forma consistente.

1.7.2 - DISTRIBUIÇÃO DE ARQUIVOS

Há diversos tipos de arquivos, cada qual com um método de distribuição de acordo com suas características:

Tipo de Arquivo	Distribuição
• arquivos individuais	▫ localização geográfica - o arquivo deve estar o mais próximo possível dos usuários interessados em seu conteúdo; ▫ localização pelo tipo - aglomerar arquivos individuais de tipos iguais (texto simples, imagens, etc).
• arquivos replicados	▫ arquivos com duas ou mais cópias no sistema devem ser distribuídos segundo o Modelo Relacional.
• arquivos particionados	Casos de particionamento de arquivos: ▫ o arquivo não pode ser manipulado por um único nó (é muito grande ou possui taxa de acesso muito alta); ▫ freqüência de acesso diferente para partes do arquivo.

Este estudo de distribuição é muito importante para a organização do servidor WWW.

1.7.3 - DISTRIBUIÇÃO DE DIRETÓRIOS

Um diretório é uma listagem dos arquivos disponíveis para os usuários da rede. Esse diretório permitirá a um usuário de qualquer nó determinar em que lugar da rede existe um arquivo compartilhável.

▫ diretório centralizado: todos os arquivos da rede são descritos em um único nó; cada nó local possui um diretório de seus arquivos locais;

▫ diretórios locais: descrevem o conteúdo local de um nó;

▫ diretório distribuído: cada nó tem uma cópia de todo o diretório da rede;

▫ diretório centralizado estendido: uma vez que o usuário acha a localização ou descrição de um arquivo, ele pode acrescentar esta informação ao seu diretório local;

▫ localização pelo nome do arquivo: o nome do arquivo pode ser usado para indicar a localização deste arquivo, prefixando-se o mesmo com um código de localização.

O servidor WWW possui uma estrutura de diretório unificado, ou seja: todos os arquivos de um servidor se encontram disponíveis a partir de um único diretório. Podemos pensar em diretório estendido quando consideramos a possibilidade de criarmos listas de preferência particulares, com endereços de documentos, através dos *browsers* (ver Capítulo 3).

1.8 - MODELO ORIENTADO A OBJETO

Os modelos de dados orientados a objetos foram projetados para a criação e representação de estruturas bem mais complexas de uma maneira uniforme, sendo mais flexíveis que o modelo relacional: o objetivo da orientação a objetos é modelar o mundo real de maneira mais direta.

Em um SGBD orientado a objeto, todos os dados são representados como um objeto, e o SGBD precisa suportar conceitos de linguagens de programação orientadas a objeto tais como: objetos, classes de objetos, abstração, encapsulamento, hierarquia, e herança de atributos e métodos.

Os objetos podem ser armazenados e manipulados diretamente: não há necessidade de transformar objetos de aplicação em tabelas, ou aplicar formas de normalização; o usuário pode definir tipos de dados, sem se restringir a tipos pré-definidos.

A linguagem de banco de dados pode ser integrada com uma linguagem de programação orientada a objetos, o que significa que o programador lida com um único modelo uniforme de objetos. Não é necessário haver linguagens diferentes para acesso ao banco de dados (como SQL) e uma linguagem host (como C) para as tarefas complexas de programação usando os dados coletados.

Assim, dizemos que Bancos de Dados Orientados a Objeto atribuem funcionalidade de SGBD para uma Linguagem de Programação Orientada a Objeto (LPOO): as estruturas e manipulação de objetos são realizadas por uma LPOO, que faz as vezes de DDL (Linguagem de Definição de Dados) e DML (Linguagem de Manipulação de Dados).

1.8.1 - SISTEMAS MULTIMÍDIA

Sistemas multimídia podem ser baseados em objetos. Tais sistemas estendem o conceito de um banco de dados para incluir gráficos, imagens, vídeo, som, bem como conjuntos numéricos e de caracteres, e texto.

Um sistema composto dessa maneira oferece muitos desafios, pelas diversas categorias de dados. Enquanto dados tradicionais são discretos, baseados em registros, e requerem métodos de busca e recuperação, um texto (p.ex.) não tem campos ou registros discretos: ele tem elementos com significado e contexto, e não apenas palavras-chave.

Gráficos, por sua vez, são armazenados como um mapa de bits, lentos e difíceis de serem recuperados. Imagens, como fotografias, também são mapeadas por bits, e requerem uma capacidade de armazenamento e poder de processamento ainda maiores. Vídeo não é um elemento estático, e exige uma enorme capacidade de armazenamento, bem como o som. A tecnologia de armazenamento ótico (CD-ROM e WORM - Write-Once, Read-Many), entre outros equipamentos de hardware, e softwares que suportam sistemas multimídia, permitem novas maneiras de se analisar e apresentar informações.

2 - REDES DE COMPUTADORES: A INTERNET

INTRODUÇÃO

A Internet nasceu nos Estados Unidos como solução para um problema de estratégia militar, nos anos 60. A idéia era criar uma rede que não fosse destruída por bombardeios e que ligasse pontos estratégicos, tais como centros de pesquisa e tecnologia. A questão parecia insolúvel, pois por melhores que fossem as conexões tudo estaria perdido se a central fosse destruída.

A solução surgiu em 1964: uma rede sem centro. Ela funcionaria de forma que todos os nós teriam o mesmo status, e qualquer computador que falhasse poderia ser substituído por outro; além disso, a informação viajaria em qualquer sentido com rotas intercambiáveis. A implementação dessa idéia se materializou em 1969, em uma rede de quatro nós: a ARPAnet[2].

A ARPAnet cresceu nos anos 70, com o desenvolvimento final de seus protocolos TCP/IP; o DARPA[3] então já realizava pesquisas pioneiras em comunicação packet-switched. Em 83, a ARPAnet se dividiu em duas redes distintas: a ARPANET e a MILNET, esta última de uso exclusivo das forças armadas americanas: os primeiro pedaços da Internet se juntavam. Nessa época, os protocolos TCP/IP se tornaram obrigatórios para todoss os computadores ligados à ARPANET. Com a distribuição desses protocolos de domínio público entre as universidades (bem como programas utilitários e sockets, que popularizaram esses protocolos entre os pesquisadores), e o maior desenvolvimento da tecnologia de interconexão de redes, a Internet explodiu além da ARPANET e das redes universitárias, atingindo centros de supercomputação através da ligação com a NSFnet[4]. Em poucos anos, a Internet atingiu a Europa, conectando computadores de universidades e laboratórios de pesquisa governamentais.

Mais e mais computadores começaram a se utilizar de TCP/IP, e praticamente todas as universidades do mundo se interligam pela Internet, espalhando-se agora entre os meios comerciais e usuários "civis comuns".

A tendência de colocar à disposição mais e mais serviços e informações em rede torna o campo de interconexão de redes cada vez mais dinâmico, provocando o surgimento de novas tecnologias e aprimoramento das já existentes. Como todos os sistemas de comunicação, os protocolos TCP/IP são um sistema não acabado, e sendo ainda desenvolvidos para vir ao encontro das novas necessidades.

2.1 - INTERCONEXÃO DE REDES (INTERNETWORKING)

O modelo de referência OSI (Open Systems Interconection) causou grande impacto na terminologia usada em interconexão de redes. O conceito fundamental é o de organização em camadas: um serviço[5] relativamente simples pode ser aumentado para oferecer serviços mais poderosos na camada imediatamente acima. Esse processo pode continuar indefinidamente até que o nível de abstração e poder desejado seja atingido.

No modelo OSI, o serviço oferecido por uma camada é ou orientado a conexão ou desconectado.

O serviço orientado a conexão tem três fases:
- estabelecimento da conexão
- transferência de dados
- liberação da conexão

O serviço desconectado tem apenas uma fase:
- transferência de dados

---

[2] ARPAnet - Advanced Projects Research Agency network.
[3] DARPA - Defense Advanced Projects Research Agency.
[4] NSFnet - National Science Foundation network.

[5] Um serviço representa um conjunto de funções oferecidas a um usuário.

Aplicação	Gerencia comunicações entre aplicações
Apresentação	Adiciona estruturas aos dados sendo trocados
Sessão	Mecanismos de controle para troca de dados
Transporte	Responsável pela confiabilidade da comunicação
Rede	Responsável pela transferência de dados na rede
Junção de Dados	Responsável pela transmissão e controle de erros
Física	Interface com o meio de comunicação

As 7 camadas do Modelo de Referência OSI [Rose_91]

Em uma internet[6], diversas redes estão conectadas através do uso de gateways e um protocolo de interconexão de redes. Pelo uso desse protocolo, os gateways (ou roteadores) escondem os detalhes das redes subjacentes, fornecendo um serviço uniforme através delas.

2.1.1 - COMUNICAÇÃO EM REDES

O modo como as redes de comunicação provêem conexões entre um computador e outro ou entre terminais e computadores pode ser dividido em dois tipos básicos: chaveado por circuito (*circuit-switched*) e chaveado por pacote (*packet-switched*).

2.1.1.1 - CHAVEAMENTO POR CIRCUITO (circuit switching)

As redes que utilizam esta tecnologia são geralmente as que transmitem voz ou dados (exemplo típico: redes telefônicas). Um caminho dedicado de transmissão é estabelecido entre qualquer par ou grupo de usuários que querem se comunicar e é mantido tanto tempo quanto for preciso para a transmissão.

2.1.1.2 - CHAVEAMENTO POR PACOTE (packet switching)

O tráfego na rede é dividido em pacotes (blocos de dados). Um pacote carrega identificação que permite aos computadores da rede saber se ele o é destinado ou como enviá-lo adiante para seu destino correto. Um único arquivo pode ser quebrado em vários pacotes para a destinação especificada, onde o software de rede o monta em um único arquivo novamente.

Dois tipos de transmissão de dados packet-switched podem ser distinguidos:

- transmissão por circuito virtual - um caminho é estabelecido entre os dois computadores comunicantes através da rede: os pacotes atravessam o caminho escolhido e chegam ao nó destino na seqüência em que foram transmitidos. Este método também é chamado transmissão orientada a objeto.
- modo desconectado - datagramas individuais se movem através da rede entre os nós fonte e destino. Nenhuma conexão inicial é estabelecida neste caso: o roteamento em nós intermediários é baseado, normalmente, no endereço de destino de cada datagrama. Datagramas não chegam necessariamente ao destino na mesma ordem em que foram transmitidos.

---

[6] Encontramos na literatura os termos:
internet, que se refere genericamente a redes que se utilizam da tecnologia de interconexão de redes, e
Internet, que se refere especificamente à rede mundial que se derivou da ARPAnet.

A vantagem principal de packet-switching é que muitas comunicações podem ocorrer concorrentemente. A desvantagem é que à medida que a atividade aumenta os computadores que usam a rede têm que esperar antes de poder enviar pacotes adicionais. As motivações para adotar packet-switching são custo e desempenho: como diversas máquinas podem compartilhar uma rede, menos interconexões são necessárias, e o custo é mantido baixo. A maior parte das redes atuais se utiliza de chaveamento por pacote (a exemplo da Internet).

TECNOLOGIAS DE REDE - WANs, MANs, LANs

Tecnologias packet switched são divididas em três categorias: Wide Area Networks (WANs), Metropolitan Area Networks (MANs), e Local Area Networks (LANs). As tecnologias que fornecem maior velocidade de comunicação operam sobre distâncias menores. Existem ainda outras diferenças.

Na tecnologia LAN, cada computador normalmente contém um dispositivo de interface de rede que conecta a máquina diretamente com o meio da rede (isto é, um fio passivo de cobre ou um cabo coaxial). Normalmente, a própria rede é passiva, dependendo de dispositivos eletrônicos nos computadores acoplados para gerar e receber os sinais elétricos necessários.

Na tecnologia MAN, uma rede contém elementos de chaveamento ativos que introduzem pequenos atrasos quando roteiam dados a seus destinos.

Nas tecnologias WAN, uma rede consiste de uma série de chaveadores de pacotes complexos interconectados por linhas de comunicação. O tamanho da rede pode ser estendido adicionando-se um novo chaveamento e outra linha de comunicação. Os' chaveadores introduzem atrasos signficativos, portanto quando maior a WAN se torna, mais ela demora para rotear tráfego.

2.2 - O CONJUNTO DE PROTOCOLOS TCP/IP

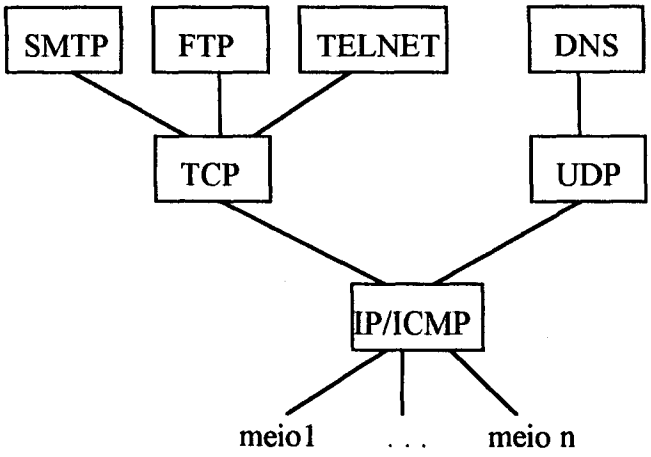
Todos os serviços de uma rede são descritos por protocolos, que dão as fórmulas para passagem de mensagens, especificam detalhes de formatos dessas mensagens, e descrevem como se manipulam condições de erro. Tudo sem precisarmos nos preocupar com detalhes do hardware de rede.

A geração atual de protocolos está baseada em:

- um serviço de transporte orientado a conexão, fornecido pelo Protocolo de Controle de Transmissão **(Transmission Control Protocol - TCP)**;
- um serviço de rede em modo desconectado, pelo Protocolo Internet **(Internet Protocol - IP)**.

A nível de aplicação, os protocolos disponíveis mais conhecidos são:

- Simple Mail Transfer Protocol (SMTP) - correio eletrônico: contacta diretamente a máquina do remetente à do destinatário;
- File Transfer Protocol (FTP) - permite cópias de grandes quantidades de dados ou programas extensos;
- TELNET - fornece serviços de terminal virtual pelo estabelecimento de uma sessão com uma máquina remota;
- Domain Name System (DNS) - realiza mapeamento entre nomes de hosts e endereços de rede.

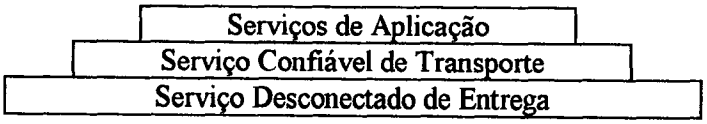


Protocolos de Rede [Rose_91]

Na figura acima vemos mais dois protocolos:

▫ UDP - User Datagram Protocol - protocolo de transporte de modo desconectado que faz pouco além de repassar dados para o IP;
▫ ICMP - Internet Control Message Protocol - avisa sobre a "saúde" da camada internet.

Conceitualmente, uma internet TCP/IP fornece três conjuntos de serviços:



O serviço desconectado de entrega serve como base para os outros serviços.

Como o modelo OSI, existe um modelo de arquitetura para o conjunto de protocolos Internet. Usaremos aqui uma visão com quatro camadas [Rose_91]:
▫ camada de interface - descreve tecnologias físicas e de junção de dados utilizadas para realizar transmissões a nível de hardware;
▫ camada internet - descreve as tecnologias "end-to-end" utilizadas para realizar a abstração de interconexão de redes;
▫ camada de transporte - descreve as tecnologias "end-to-end" utilizadas para realizar comunicações confiáveis entre hosts;
▫ camada de aplicação - descreve as tecnologias utilizadas para fornecer serviços para o usuário final.

2.2.1 - IP (INTERNET PROTOCOL) : ENTREGA DESCONECTADA DE DATAGRAMAS

O protocolo IP é responsável pelo endereçamento e transmissão dos dados. Tecnicamente, esse serviço é definido como um sistema de entrega de pacotes desconectada, não confiável, de melhor esforço.

É chamado "não-confiável" porque a entrega não é garantida. O pacote pode ser perdido, duplicado, demorado, ou entregue fora de ordem, mas o serviço não irá detectar essas condições, nem irá informar o remetente ou o destinatário.

"Desconectado" porque cada pacote é tratado independentemente dos outros. Uma seqüência de pacotes de uma máquina para outra pode viajar sobre diferentes caminhos, ou alguns podem ser perdidos enquanto outros são entregues.

Finalmente, o serviço é dito usar entrega de "melhor esforço" porque o software internet faz realmente uma tentativa de entregar pacotes, não descartando pacotes caprichosamente: a não-confiabilidade surge somente quando os recursos estão exauridos ou quando as redes subjacentes falham.

IP fornece três definições importantes:

Primeira, o protocolo IP define a unidade básica de transferência de dados através de uma internet TCP/IP.

Segunda, o software IP realiza a função de roteamento, escolhendo um caminho através do qual os dados serão enviados.

Terceira, IP inclui um conjunto de regras que incorporam a idéia de entrega de pacotes não-confiável. As regras caracterizam como hosts e gateways devem processar pacotes, como e quando devem ser geradas mensagens de erro, e as condições sob as quais os pacotes podem ser descartados.

2.2.1.1 O DATAGRAMA INTERNET

Em uma rede, a unidade de transferência é um frame que contém um cabeçalho e dados; o cabeçalho dá informações tais como os endereços (físicos) da origem e do destino. A Internet chama sua unidade básica de transferência de datagrama Internet. A diferença é que o cabeçalho do datagrama contém endereços IP enquanto o cabeçalho do frame contém endereços físicos.

Frames devem ser reconhecidos por hardware, enquanto que datagramas são manipulados por software. Como os datagramas se movem de uma máquina para outra, eles sempre têm que ser transportados em frames físicos. A idéia de se carregar datagramas em um frame de rede é chamada encapsulamento. Para a rede subjacente, um datagrama é igual a qualquer outra mensagem enviada de uma máquina para outra. No caso ideal, o datagrama inteiro cabe em um frame, tornando a transmissão eficiente.

Toda rede packet-switching estabelece um limite máximo para o tamanho do campo de dados utilizado para encapsular um datagrama IP. Esse limite é chamado *maximum transfer unit* ou MTU, medido em octetos. Quando se deseja enviar um datagrama maior que o MTU da interface local, é preciso dividir datagramas grandes em pedaços chamados fragmentos. Esses fragmentos são transmitidos pela rede e então montados novamente no destino em um datagrama completo.

Na Internet, uma vez que um datagrama foi fragmentado, os fragmentos viajam como datagramas separados por todo o caminho até seu destino, onde são montados. Se algum fragmento for perdido, o datagrama não pode ser montado e os pedaços que chegaram ao destino serão eventualmente descartados sem processamento.

O formato de um datagrama apresenta campos que descrevem, em geral:

- a versão do protocolo IP (identifica o formato usado);
- tamanho do cabeçalho do datagrama;
- o tipo de serviço, que especifica como o datagrama deve ser manipulado;
- o tamanho total do datagrama, em octetos;
- identificação, fornecida pelo originador do datagrama, e utilizada na montagem;
- controle de fragmentação e montagem do datagrama;
- tempo de vida do datagrama, isto é, por quanto tempo o datagrama pode permanecer no sistema Internet: esse campo é decrementado pelos gateways, que descartam o datagrama quando esse tempo atinge zero;
- o protocolo que especifica o formato e o conteúdo dos dados, em uma camada superior;
- controle do header, recalculado a cada envio do datagrama (detecção de erros);
- endereços Internet de fonte e destino do datagrama;
- os dados do usuário.

2.2.1.2. ENDEREÇOS INTERNET

Sendo a Internet uma estrutura virtual, seus endereços não são definidos por hardware: a cada computador é associado um endereço de 32 bits, chamado endereço IP. Esse endereço tem de ser único em toda a Internet, de modo a evitar conflitos. O órgão responsável pela atribuição desse endereço é o Network Information Center (InterNIC Registration Services).

O endereço IP é normalmente escrito como quatro (4) números decimais separados por pontos (por exemplo, `143.107.231.1`). Cada número decimal representa um dos quatro bytes que existem nos 32 bits do endereço. Conceitualmente, o endereço IP é dividido em duas partes: uma, referente à rede; outra, referente ao nó.

Cada endereço é um par (netid,hostid), onde netid identifica uma rede e hostid identifica um host naquela rede. Dizemos, com maior rigor, que um endereço especifica uma conexão de rede de um host. Assim, um gateway conectando n redes tem n endereços Internet distintos, um para cada conexão de rede.

Existe um mecanismo que associa um nome a cada endereço IP (por exemplo, o número `143.107.231.1` está associado a `xavante.icmsc.sc.usp.br`); esse nome é escolhido pelo administrador da máquina e deve ser único dentro de uma estrutura hierárquica de nomes (do tipo `máquina.rede.país`). Essa estrutura hierárquica é denominada *domain name* ou *domínio*, e reflete uma estrutura das organizações que compõem a Internet, e não uma estrutura física.

Os níveis mais altos de domínio são:
- `com` - organizações comerciais;
- `edu` - instituições educacionais;
- `gov` - instituições governamentais;
- `mil` - grupos militares;
- `net` - principais centros de suporte de rede;
- `org` - outras organizações (p. ex., entidades não-governamentais);
- `xy` - duas letras representando um país.

No Brasil, o domínio mais alto é `.br`. Seguindo essa convenção de nomenclatura de domínios, que surgiu nos Estados Unidos, outros países, como o Brasil, utilizam formas como: `.org.br`, `.gov.br`, etc (por exemplo, `embratel.net.br`, `senado.gov.br`).

Na prática, a tradução desses nomes para endereços IP se processa do domínio mais alto para o mais baixo, através de *nameservers* (servidores de nomes). Quando um *nameserver* recebe um nome para ser traduzido, ele verifica se esse nome pertence a seu subdomínio, através de tabelas de tradução. Se tiver sucessso, retorna um endereço IP; caso contrário, contacta outro servidor de nomes que possa realizar essa tradução.

2.2.2 - TCP (TRANSMISSION CONTROL PROTOCOL): SERVIÇO CONFIÁVEL DE TRANSPORTE DE DADOS

Embora TCP seja apresentado como parte do conjunto de protocolos Internet TCP/IP, ele é um protocolo independente, de propósito geral, que pode ser adaptado para uso com outros sistemas de entrega de pacotes. Como TCP assume muito poucas coisas a respeito da rede subjacente, é possível usá-lo sobre uma única rede tal como uma Ethernet, bem como sobre uma internet complexa.

Basicamente, TCP quebra a transmissão de dados em pacotes, rearranja-os no final da transmissão na ordem correta, e envia de volta partes incorretamente transmitidas.

2.2.2.1 - O CONCEITO DE JANELAS DESLIZANTES

O conceito conhecido por *sliding window* torna eficiente a transmissão dos dados: o remetente transmite um pacote e então espera por um sinal reconhecimento (confirmação de recebimento); enquanto espera, o emissor pode transmitir outros pacotes. Visualizando uma janela em uma seqüência de pacotes a serem transmitidos, teremos que os pacotes dentro da janela serão transmitidos pelo protocolo.

Por exemplo, um emissor com janela tamanho N pode enviar pacotes do quadro 0 até N-1 sem que tenha recebido confirmação; no entanto, para enviar o pacote N é necessário que o primeiro pacote enviado (número 0) seja confirmado. Quando essa confirmação ocorre, a janela desliza, passando a incluir os pacotes de 1 a N; quando o pacote 1 é confirmado, a janela passa a incluir os pacotes de 2 a N+1. Normalmente, a numeração de seqüência é circular.

Se um pacote transmitido não for reconhecido em seu destino - isto é, não for confirmado -, é utilizada uma temporização para que seja retransmitido.

2.2.2.2 - PORTAS E CONEXÕES

TCP incorpora objetos abstratos chamados *portas* que identificam o destino final dentro de uma máquina. A cada porta existe um número associado. Como números de porta são locais a uma dada máquina e únicos dentro dela, o destino final para o tráfego TCP é especificado unicamente dando-se um endereço IP para um host destino bem como um número de porta TCP naquele host.

TCP é um protocolo que precisa de dois pontos (inicial e final) para realizar sua comunicação. Antes que o tráfego TCP possa passar através da Internet, programas de aplicação nas duas extremidades da conexão devem concordar que a conexão é desejada.

Para isso, uma extremidade realiza uma função *open* passiva contactando seu sistema operacional e indicando que ele receberá uma conexão de fora. Nessa hora, o sistema operacional estabelece um *port number* nessa extremidade para essa conexão. O programa de aplicação na outra extremidade deve então contactar seu sistema operacional usando um pedido open ativo para estabelecer uma conexão. Os dois módulos TCP se comunicam para verificar se a conexão foi estabelecida. Uma vez criada a conexão, os módulos TCP em cada extremidade podem começar a passar dados.

2.3 - O MODELO DE INTERAÇÃO CLIENTE-SERVIDOR

O primeiro padrão de interação entre aplicações cooperativas é conhecido como o paradigma cliente-servidor. Ele forma a base da maior parte da comunicação em rede e é fundamental porque nos ajuda a entender o fundamento no qual os algoritmos distribuídos são construídos.

2.3.1 - O MODELO CLIENTE-SERVIDOR

O termo "servidor" se aplica a qualquer programa que oferece um serviço que pode ser atingido através da rede. Servidores aceitam pedidos que chegam através da rede, desempenham seu serviço, e retornam o resultado para o requisitante. Para os serviços mais simples, cada pedido chega em um único datagrama IP e o servidor retorna sua resposta em outro datagrama.

Um programa sendo executado se torna um "cliente" quando ele envia um pedido para um servidor e espera por uma resposta. Como o modelo cliente-servidor é uma extensão conveniente e natural de comunicação interprocessos em uma única máquina, é fácil construir programas que o utilizam para interagir.

Servidores podem desempenhar tarefas simples ou complexas. Por exemplo, um servidor "hora do dia" simplesmente retorna a hora atual sempre que um cliente envia um pacote para ele. Um servidor de arquivos recebe pedidos para desempenhar operações que armazenam ou recuperam informações de um arquivo; o servidor desempenha a operação e retorna o resultado. Ao cliente cabe a tarefa de enviar pedidos para o servidor, e apresentar a resposta obtida.

Normalmente, os servidores são implementados como programas a nível de aplicação. A vantagem disso é que eles podem ser executados em qualquer sistema de computadores que suporte comunicação TCP/IP.

Portanto, o servidor para um serviço particular pode ser executado em um sistema de compartilhamento de tempo (timesharing system) junto com outros programas, ou pode ser executado em um PC. Diversos servidores podem oferecer o mesmo serviço, e podem ser executados na mesma máquina ou em diversas máquinas. Se o principal propósito de uma máquina é suportar um programa servidor particular, o termo "servidor" pode ser aplicado à máquina bem como ao programa servidor.

2.3.2 - COMUNICAÇÃO INTERPROCESSOS

Um *socket* é um ponto de referência para o qual podem ser enviadas/recebidas mensagens Quando um socket é criado, um protocolo deve ser especificado - pode ser um datagrama (usando UDP) ou uma corrente de dados (usando TCP/IP). Para ocorrer comunicação entre um par de processos, cada um deve criar seu próprio socket, sendo os sockets usados em pares.

Na comunicação dentro de um computador, sockets são chamados *pathnames*; na comunicação entre computadores separados, os sockets são referenciados por um endereço IP e um número de porta TCP.

Na comunicação por *datagrama*, um par de sockets é identificado toda vez que realiza uma comunicação. Na comunicação baseada em *conexão*, dois processos devem, primeiro, estabelecer uma conexão entre seu par de sockets: um servidor cria um socket e associa um identificador a ele; clientes podem então pedir conexões para o socket do servidor. Quando uma conexão é aceita, é criado um novo socket para fazer par com o socket do cliente, assim o servidor pode continuar a receber pedidos de conexão de outros clientes através do socket original. A partir daí, servidor e cliente podem trocar dados até que a conexão seja fechada.

i. TRANSAÇÕES

Transações permitem que os clientes especifiquem seqüências de operações que serão desempenhadas como uma unidade atômica. Como já vimos no Capítulo 1, há dois fatores que ameaçam a atomicidade de transações: o primeiro é a falha de processos e computadores, e é tratada por recuperação; o segundo é a execução concorrente de transações e é tratado por controle de concorrência.

ii. PROTEÇÃO

Em um sistema distribuído cliente-servidor, cada serviço gerencia uma coleção de objetos com um conjunto de domínios de proteção associado. O direito de receber mensagens em portas de serviço pode ser restrito aos processos do servidor, para evitar que clientes não autorizados enviem respostas a outros clientes.

2.3.3 - A COMPLEXIDADE DOS SERVIDORES

Na prática, os servidores são normalmente mais difíceis de serem construídos do que os clientes, porque eles precisam acomodar diversos pedidos concorrentes. Normalmente, os servidores têm duas partes: um único mestre que é responsável por receber novos pedidos, e um conjunto de escravos que reponsáveis por lidar com pedidos individuais. O servidor mestre desempenha os seguintes 5 passos:

Open port	O mestre abre a porta na qual ele pode ser acessado
Wait for client	O mestre espera que um novo cliente envie um pedido.
Choose port	Se necessário, o mestre aloca uma nova porta de protocolo local para este pedido e informa o cliente (passo desnecessário com TCP).
Start Slave	O mestre inicia um escravo independente concorrente para lidar com este pedido (p.ex., em UNIX ele realiza um fork do processo servidor). O escravo não aceita mais pedidos da porta do servidor - somente lida com um pedido e então termina.
Continue	O mestre retorna ao passo de espera e continua a aceitar novos pedidos enquanto o escravo que acabou de ser criado lida com o pedido prévio concorrentemente.

Como o mestre inicia um escravo para cada novo pedido, o processamento se dá concorrentemente. Portanto, os pedidos que requerem pouco tempo para serem completados podem terminar antes dos pedidos que demoram mais, independentemente da ordem em que são iniciados.

Além da complexidade resultante do fato dos servidores terem que lidar com pedidos concorrentes, a complexidade também surge porque os servidores devem reforçar as regras de autorização e proteção. Os programas servidores normalmente precisam ser executados com altos privilégios porque eles precisam ler *system files*, manter *logs* (registros de acessos), e acessar dados protegidos. O sistema operacional não irá restringir um programa servidor se ele tentar acessar arquivos de usuários. Portanto, os servidores não podem servir pedidos de outros locais cegamente. Ao invés disso, cada servidor tem a responsabilidade de reforçar as políticas de acessos ao sistema e proteção.

Finalmente, os servidores devem se proteger contra pedidos mal formulados ou contra pedidos que podem causar a interrupção do programa servidor. É difícil prever tais problemas, uma vez que comandos simples e aparentemente ingênuos podem provocar graves efeitos colaterais.

3 - A WORLD-WIDE WEB

INTRODUÇÃO

A World-Wide Web, criada em 1990, gera uma teia viva de bancos de dados distribuídos que apresentam informações na Internet em diversos meios: textos, imagens, sons. As informações são dinâmicas tal como esse sistema: novos formatos, padrões de protocolos e linguagem são propostos e utilizados, em uma ampla troca de informações pelo mundo todo.

Grande parte das informações sobre a WWW é coletada na própria WWW. Este capítulo não esgota, absolutamente, tudo o que é discutido sobre esse sistema distribuído, mas procura abranger boa parte das informações necessárias para a compreensão de seu mecanismo: como as informações são acessadas e recuperadas, utilização da linguagem HTML (exemplos práticos se encontram no tutorial on-line) e mecanismos do protocolo HTTP.

O conteúdo teórico deste capítulo foi baseado em RFCs[7] e Internet drafts[8] disponíveis pela Web - os endereços consultados se encontram na seção ***Referências na WWW***.

3.1 - O QUE É A WWW?

Devemos partir dos seguintes conceitos:

• *Hipertexto:* permite que um pedaço de texto seja referenciado através de outros pedaços de texto, seja num mesmo documento ou em outros documentos em uma rede de computadores (no caso da W3, na Internet).

• *Hipermídia*: um superconjunto de hipertexto - é qualquer meio (texto, imagem, som, animações) com ponteiros para outro meio (animações, som, imagem, texto).

A idéia da World-Wide Web surgiu de uma experiência bem-sucedida no CERN (*Centre d'Etudes et Recherches Nucleaires* - Centro de Estudos e Pesquisas Nucleares, em Genebra, Suíça) com um pequeno sistema em hipertexto usado para manter informações em um projeto distribuído.

O ***Projeto WWW*** surgiu a partir dessa experiência, visando a construção de um sistema de hipermídia distribuído: a World-Wide Web ("Ampla Teia Mundial"[9]), também chamada W3 ou WWW, ou Web - um *pool* de conhecimento que permita a colaboradores, em locais remotos, compartilhar suas idéias e todos os aspectos de um projeto comum.

A estrutura foi projetada de modo que, se a Web estivesse sendo usada independentemente para dois projetos, e depois fossem surgindo relacionamentos entre esses projetos, nenhuma grande mudança centralizada teria de ser feita: as informações poderiam ser levemente remodeladas para representar esse novo estado de conhecimento. Essa propriedade permitiu que a WWW se expandisse rapidamente de suas origens no CERN através da Internet.

3.1.1 - ACESSO À WORLD-WIDE WEB

Para se acessar a Web, roda-se um *programa paginador* (*browser*). Logo que um browser é iniciado, ele mostra um objeto, normalmente com texto e, provavelmente, imagens. Algumas das imagens e frases são destacadas: em azul, ou verde, ou dentro de caixas, sublinhadas ou numeradas, dependendo do tipo de display e configuração do browser utilizado. Os documentos que os browsers apresentam são documentos hipertexto. Selecionar uma área destacada (chamada *âncora*), faz com que

---

[7] RFC (Request for Comments) é um conjunto informal de artigos que sintetizam propostas de grupos de pesquisadores, com a preocupação de disseminar uma discussão sobre determinado tema. As propostas depuradas e consideradas boas recebem números.

[8] *Internet drafts* são documentos sobre um trabalho em desenvolvimennto, válidos por até seis meses.

[9] Tradução segundo [Barreira_94].

o browser recupere outro objeto, provavelmente de um outro computador remoto. O objeto recuperado está normalmente em formato hipertexto também, então o processo de navegação continua. A seleção é realizada ou através do acionamento do mouse sobre uma área destacada, ou conforme as caracterísitcas do browser utilizado. Os browsers podem acessar arquivos por FTP, NNTP, Gopher, e por diversos outros métodos. Além disso, se o servidor acessado permite a realização de buscas, os browsers permitirão buscas de documentos e bancos de dados.

Os browsers que acessam a World-Wide Web navegam sobre um banco de informações relacionadas, organizado somente pelo critério de associação temática: não existe ordem fixa, ou uma hierarquia. De página em página o usuário vai explorando assuntos relacionados. As páginas que compõem esse banco de informações estão espalhadas por todo o mundo, como um hipertexto em escala planetária. Se for desenhado um diagrama mostrando as relações entre todas as páginas, obter-se-á uma figura que lembra uma teia de aranha de proporções mundiais.

3.2 - ARQUITETURA

A comunicação na WWW funciona no modelo cliente-servidor: a base dessa teia são os programas servidores de arquivos, que atendem requisições de outros programas, chamados clientes - ***não é*** um serviço orientado à conexão, como o telnet.

Os clientes da Web são os browsers: há muitos tipos de programas clientes. Como a informação hipertexto é transmitida pela rede de maneira já formatada, cada cliente pode interpretá-la de maneira natural para sua própria plataforma, fazendo uso ótimo de fontes, cores e outras interfaces disponíveis. Portanto, um mesmo documento pode ser interpretado de maneira diferente (fontes, gráficos, etc) de acordo com a plataforma em que é apresentado.

Um servidor W3 é um programa que responde a uma conexão TCP de chegada e fornece serviços para o programa chamador. Há muitas variações do software do servidor W3 para servir diferentes formas de dados. Qualquer que seja o tipo do servidor, a interface com o usuário é a mesma, portanto não é preciso entender as diferenças entre os vários protocolos em uso.

3.2.1 - BROWSERS

O tipo de interface oferecida para o acesso aos serviços e dados na Internet resumia-se a um software emulador de terminal, que essencialmente transformava o computador do usuário em um terminal burro de um servidor de informações. A interface resumia-se a uma tela limpa, com um prompt e um cursor, e era preciso conhecer os comandos de acesso aos serviços desejados.

Essa interface restringia o acesso de usuários leigos à rede. A teia Gopher surgiu com a primeira interface que facilitou a navegação pela rede mundial, tornando transparente aos usuários a manipulação de códigos de acesso e a tradução dos diferentes formatos de informações disponíveis. As ligações Gopher levam a menus, textos, índices ou conexões Telnet.

A WWW, além de oferecer tudo isso, apresenta textos mais elaborados (com cabeçalhos, listas, textos enfatizados) de maneira padronizada, além de recursos multimídia. Mas a comunidade Internet só teve acesso a uma interface gráfica em 1993, portanto 3 anos após o nascimento da Web: o Mosaic (desenvolvido no *NCSA* - National Center for Supercomputing Applications - centro de computação da University of Illinois at Urbana-Champaign). Antes do Mosaic, os browsers utilizados eram de modo texto, pouco diferindo da interface Gopher.

Deve ficar bem claro que todos os browsers que acessam a WWW podem acessar os mesmos documentos: o que muda é o modo de apresentarem esses documentos, que depende da plataforma do browser. A Teia é uma só, como trilhos que unem um banco de dados distribuído, e os browsers navegam sobre esse banco de dados "como um trem se move sobre trilhos" [Ramalho_94].

O papel dos programas paginadores é enviar requisições para os servidores e processar os documentos recebidos para a apresentação do conteúdo ao usuário. Quando recebe um documento em hipertexto, o browser realiza uma "análise sintática" (*parsing*) do conteúdo. Os elementos de

formatação do documento, entendidos pelo browser, são aplicados; os elementos desconhecidos ou não disponíveis são ignorados.

Há muitos produtos semelhantes, disponíveis para diversas plataformas - como é o caso do *Web Explorer* para OS2/Warp, o *InternetWorks* para Windows, o *Hot Java* da Sun para Solaris e Windows 95 (que permite extensões interativas através de programas na linguagem *java*) - sendo a maioria de domínio público, podendo ser encontrados na rede por ftp. O browser mais "famoso" na atualidade é o Netscape Navigator, que reconhece formatações mais avançadas em relação aos outros browsers, sendo assim capaz de apresentar documentos mais elaborados.

3.3 - FUNDAMENTOS DA WWW

A WWW define:

• a idéia de um mundo de informações sem fronteiras no qual todos os itens têm uma referência pela qual podem ser recuperados;

• o sistema de endereços (URI) que o projeto implementou para tornar esse mundo possível, apesar dos muitos protocolos diferentes;

• um protocolo de rede (HTTP) usado por servidores WWW dando o desempenho e características que não estariam disponíveis de outra forma;

• uma linguagem de formatação (HTML) que todo cliente WWW entende, e que é usada para a transmissão de coisas básicas como texto, menus e informações on-line através da rede.

Veremos como estão implementados e como funcionam estes fundamentos.

3.3.1 - UNIVERSAL RESOURCE IDENTIFIER

A sintaxe **URI** (Universal Resource Identifier - Identificador Universal de Recursos) permite o acesso a objetos que estejam disponíveis por qualquer protocolo já existente ou que venha a ser desenvolvido, de forma legível para o homem. Essa sintaxe é seguida pelos métodos de endereçamento:

URL (Uniform Resource Locator - Localizador Uniforme de Recursos) é um URI usado para localizar recursos que podem ser acessados através de protocolos existentes, de maneira concisa o suficiente para ser chamado de "endereço" (uma identificação abstrata da localização do recurso).

URLs são fundamentais na arquitetura WWW. O fato de ser fácil endereçar um objeto em qualquer lugar na Internet é essencial para o aumento do sistema, e para o espaço de informação ser independente da topologia da rede e do servidor.

Não há garantias de que um URL, que num momento esteja apontando um dado objeto, continue a fazê-lo depois de certo tempo, devido à mobilidade dos objetos nos servidores.

URN (Uniform Resource Names - Nomes Uniformes de Recursos) é um nome com escopo global que tem o mesmo significado independentemente da localização. Ainda em discussão, define um espaço de nomes mais persistente que os URLs.

3.3.1.1 - URL - UNIFORM RESOURCE LOCATOR

O URL contém as seguintes informações:

• o tipo de serviço pelo qual o recurso é servido (http, gopher, wais,...); esse especificador de serviço indica o método de acesso.

• o nome internet do local (site) que contém o recurso;

• o port number do serviço. Se omitido, o browser assume um valor default de acordo com o método de acesso;

• a localização do recurso na estrutura de diretórios do servidor.

A sintaxe genérica do URL consiste de seis componentes:

```
<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>
```

onde:

```
scheme:        ::= nome do esquema
//<net_loc>    ::= localização na rede
/<path>        ::= caminho dentro do servidor
;<params>      ::= parâmetros de um objeto
?<query>       ::= informações para uma consulta
      <query> ::= <argument>{+<argument>}
#<fragment>    ::= identificador de fragmento
```

O fragmento (e o # precedente) não é considerado parte do URL. A cadeia `#<fragment>` não é enviada para o servidor, mas é retida pelo cliente e usada quando o objeto todo foi recuperado.

Seguindo o esquema de codificação do URI, quando o esquema de nomeação local user caracteres não permitidos, eles devem ser representados no URL por um `%` seguido por dois dígitos hexadecimais dando o código ISO Latin-1 para aquele caracter. Por exemplo, um endereço contendo `~gusfaria` será codificado, em alguns sistemas, como `%7gusfaria`.

3.3.1.2. ESQUEMAS URL

FTP File Transfer Protocol
`ftp://[user[:password]@]host/path/file.ext`

HTTP HyperText Transfer Protocol
`http://host[:port]/path/`

Gopher protocolo Gopher
`gopher://host/`

Mail endereço e-mail
`mailto:user@host`

News USENET news
`news:<newsgroup-name>`

News news por acesso NNTP
`nntp://<host>:<port>/<newsgroup-name><control-number>`

Telnet sessão interativa
`telnet://[user[:password]@]host[:port]`

rlogin sessão interativa
`rlogin://username@host`

WAIS Wide Area Information Servers
`wais://<host>:<port>/<database>[?search][/<type>/<path>]`

file arquivo local
`file://host/path/arq.ext`

3.3.1.3. URLs PARA SERVIDORES HTTP

A porta 80 é default para servidores http.

Se o nome de um arquivo não for especificado, o servidor tenta enviar um arquivo default do diretório indicado. Normalmente o default é o `index.html`, mas isso pode ser modificado pelos arquivos de configuração do servidor.

O recurso é especificado por um caminho relativo ao diretório raiz do servidor. Os usuários podem ter documentos em seus próprios diretórios home, fora da hierarquia do servidor.

i. URLs Relativos

Um uso comum para URLs é inserí-los em um documento (chamado documento base) para o propósito de se identificar outros recursos acessíveis pela Internet; em documentos hipertexto, os URLs podem ser usados também como identificadores para destinos de links hipertexto.

Quando inserido em um documento base, um URL em sua forma absoluta pode conter informações que já são conhecidas no contexto da recuperação do documento base, incluindo o esquema, localização na rede e partes do caminho de acesso; nessas situações, em que o URL básico é bem definido e conhecido, é útil poder inserir referências URL que herdem o contexto em vez de reespecifiar tudo em todas as instâncias.

Como os servidores WWW se valem de uma estrutura hierárquica de arquivos, os endereços relativos de URLs permitem que árvores inteiras de documentos sejam parcialmente independentes de sua localização e esquema de acesso; desse modo é possível movê-las como um todo, sem mudar nenhum dos URLs inseridos. Essa habilidade é necessária para o uso de URLs relativos a longo prazo.

ii. MIME - Multipurpose Internet Mail Extensions

O RFC822 define um protocolo de representação de mensagens que especifica detalhes sobre as mensagens e conteúdo de mensagens de mail na Internet, permitindo sua representação e intercâmbio sem perda de informações. MIME (RFC1521) é uma extensão do SMTP (Simple Mail Transfer Protocol) que permite a transmissão de objetos em uma grande variedade de representações:

- *MIME-Version* - usa número de versão para declarar que uma mensagem está conforme uma especificação MIME;
- *Content-Type* - pode ser usado para especificar o tipo e subtipo dos dados contidos na mensagem e a representação (codificação) desses dados. Valores possíveis para esse campo: *text, multipart, application, message, image, audio, video.*

Quando o browser recupera um arquivo, ele deve saber o que fazer com ele. Em geral isso é feito através da extensão do arquivo. Se o browser não conseguir identificar o tipo do arquivo, esse arquivo será apresentado como arquivo texto.

Os browsers geralmente usam o tipo MIME do documento para determinar como ele será tratado. Os tipos MIME são usados para mapear tipos particulares de arquivos (determinados pelas extensões ou algum tipo de exame de conteúdo) para determinadas funções (tais como processadores de som, mostradores de imagem, etc.). Servidores HTTP acoplam cabeçalhos MIME a cada arquivo, que servem para o browser, de modo que o browser saiba que tipo de arquivo é e o que fazer com ele.

Os tipos de arquivos reconhecidos pelo servidor HTTP são, por extensão:

`.txt` ou `.text`	texto ASCII (plain text) - o browser apresenta o arquivo como um bloco de texto e não o processa seguindo instruções de formatação

`.html`	documento HTML
`.gif`	imagem GIF
`.xbm`	imagem X-Bitmap
`.tiff`	imagem tiff
`.xpm`	X-Pixmap (colorida)
`.jpg ou .jpeg`	imagem JPEG
`.ps`	arquivo PostScript
`.Z`	arquivo comprimido
`.gz`	arquivo comprimido
`.aiff`	som AIFF
`.au`	som AU
`.mov`	filme QuickTime[10]
`.mpeg ou .mpg`	filme MPEG

iii. PASSAGEM DE PARÂMETROS PARA O SERVIDOR

O protocolo HTTP suporta a passagem de argumentos para o servidor. O formato geral é

```
http://servidor/cgi-bin/nomescript?arg1+arg2+arg3
```

/cgi-bin/ é o diretório que contém programas ou scripts de interface com a WWW - um URL pode acessar e passar argumentos para programas/scripts nesse diretório, e esses programas/scripts podem agir nesses argumentos e retornar informações, documentos, etc, para o browser.

Os argumentos são anexados ao URL após um ponto de interrogação. Pode ser enviado mais de um argumento, separados por um sinal de mais ("+").

3.3.2 - PROTOCOLO DE TRANSFERÊNCIA DE HIPERTEXTO

O que é HTTP (*Hypertext Transfer Protocol*):

• protocolo para transferência de informação com a eficiência que é necessária para se fazer saltos em hipertexto. Os dados transferidos podem ser texto ASCII (*plain text*), hipertexto, imagens, animações, som;

• protocolo Internet "stateless" - isto significa que ele roda sobre uma conexão TCP que é mantida apenas durante uma operação. O modelo stateless é eficiente quando uma ligação de um objeto pode levar igualmente a um objeto armazenado no mesmo servidor, ou a um outro servidor distante;

• protocolo orientado a objeto que pode ser usado para muitas tarefas tais como servidores de nome e sistemas orientados a objeto distribuídos, pela extensão dos comandos ou métodos utilizados;

• protocolo baseado no paradigma de requisição/resposta que suporta a transmissão de dados em um conjunto ilimitado e ampliável de formatos. Essa habilidade de negociação de formatos é conseguida da seguinte maneira: o cliente envia uma lista dos formatos que consegue manipular, e o servidor responde com dados em qualquer dos formatos que consegue produzir.

Seu overhead é baixo, por não ter de suportar caracterísitcas completas de navegação como fazem os protocolos FTP e Gopher: a informação de navegação é inserida nos documentos diretamente, via HTML.

As mensagens são passadas em formato similar àquele usado pelo RFC822 STD11 (Internet Mail) e MIME (RFC1521). O HTTP/1.0 também é usado para comunicação entre clientes ("user agents") e diversos gateways, permitindo acesso hipermídia a protocolos como SMTP, NNTP, FTP, Gopher e Wais.

---

[10] Muitas workstations UNIX não conseguem "ver" filmes QuickTime

3.3.2..1 - TRANSAÇÕES

Uma transação consiste de: conexão, consulta, resposta e fechamento.

Conexão	O estabelecimento de uma conexão pelo cliente para o servidor - usando TCP/IP, port 80 é a porta conhecida, mas outras portas não reservadas podem ser especificadas no URL;
Requisição	O envio, pelo cliente, de uma requisição de consulta para o servidor;
Resposta	O envio, pelo servidor, de uma resposta para o cliente;
Fechamento	O fechamento da conexão por qualquer das duas partes.

3.3.2.2. REQUISIÇÃO

A requisição HTTP do cliente começa com um código de operação (conhecido como método, de acordo com a terminologia orientada a objeto), o URI do objeto, e a versão do protocolo em uso, seguido por mais informações codificadas no estilo de cabeçalho RFC822. O formato da consulta é:

```
Request          = SimpleRequest | FullRequest
SimpleRequest    = GET <Request-uri> CrLf
FullRequest      = Method <Request-uri> ProtocolVersion CrLf
                   [*<HTRQ Header>],
                   [<CrLf> <data>]
<Method>         = GET | HEAD | POST | PUT | DELETE | LINK |
                 UNLINK | ...
ProtocolVersion  = HTTP/V1.0
<HTRQ Header>    = <Fieldname> : <Value> <CrLf>
<data>           = mensagem em formato MIME
```

O campo *ProtocolVersion* define o formato da consulta. Se a versão do protocolo não estiver especificada, o servidor assume que o paginador usa o HTTP versão 0.9.

O campo *método* indica o método a ser aplicado no objeto indentificado pelo URI.

HTQR - cabeçalhos de consulta RFC822 com nomes de campos especiais, bem como qualquer outro cabeçalho de objeto ou cabeçalho MIME. Quando objetos são transferidos sobre a rede, informações sobre eles (metainformações) são transferidas em cabeçalhos HTTP. O conjunto de cabeçalhos é uma extensão do conjunto Multipurpose Internet Mail Extensions (MIME), que permite a transmissão de objetos em uma grande variedade de representações.

O conteúdo de um objeto é enviado (dependendo do método) com o pedido e/ou resposta.

i. MÉTODOS

O campo Método em HTTP indica o método a ser aplicado no objeto identificado pelo URL. O método GET é sempre suportado. Nomes de métodos são sensíveis à caixa: com um conjunto limitado de métodos, a falta de sensibilidade à caixa seria tolerável, mas o objetivo de HTTP é cobrir um sistema arbitrário de objetos subjacentes; em tal sistema, nomes de métodos podem ser sensíveis à caixa, e então isso foi preservado em HTTP.

Métodos especificados ou em discussão atualmente:

GET	recupera quaisquer dados identificados pelo URL; o URL se refere a um processo de produção de dados, são esses dados que serão retornados, e não o texto fonte do script ou processo. Também usado para buscas.
HEAD	é o mesmo que GET mas retorna somente cabeçalhos HTTP e não conteúdo de documentos.
POST	Cria um novo objeto ligado ao objeto especificado O campo message-id do novo objeto pode ser estabelecido pelo cliente ou será dado pelo servidor. Um URL será alocado pelo servidor e retornado para o cliente. O novo documento é a parte de dados do pedido. É considerado ser subordinado ao objeto especificado, no sentido em que um arquivo é subordinado a um diretório que o contém, um artigo news é subordinado a um newsgroup no qual ele é postado.
PUT	Especifica que os dados na seção do corpo serão armazenados no dado URL. O URL precisa existir previamente. O novo conteúdo do documento é a parte de dados do pedido. POST e REPLY podem ser usados para criar novos documentos.
DELETE	Pede que o servidor apague as informações correspondentes ao dado URL. Após um método DELETE bem sucedido, o URL se torna inválido para qualquer método futuro.
LINK	Liga um objeto existente ao objeto especificado.
UNLINK	Remove informações de ligação (ou outra meta-informação) de um objeto.

O método GET é definido de modo a preservar o estado da Web. Ele envia uma representação do objeto para o cliente.

O método POST pode ser usado para as seguintes funções:

- anotar recursos existentes;
- postar uma mensagem em um newsgroup, mailing list, etc,;
- prover dados para um processo manipulador de dados;
- ampliar uma base de dados através de uma operação de *append*.

O método LINK estabelece um ou mais relacionamentos de ligaçãoentre o recurso existente identificado pelo *Request URI* e outros recursos existentes. Não resulta na criação de novos recursos.

ii. REQUEST HEADER FIELDS

Os campos do cabeçalho de requisição são enviados pelo cliente em uma transação de protocolo HTTP. Todas as linhas são cabeçalhos em formato RFC822.

Esses campos permitem que o cliente passe informações adicionais sobre a consulta (e sobre o próprio cliente) para o servidor. Todos esses campos são opcionais.

```
Request-Header =   Accept
                   | Accept-Charset
                   | Accept-Encoding
                   | Accept-Language
                   | Authorization
                   | From
                   | If-Modified-Since
                   | Pragma
                   | Referer
                   | User-Agent
                   | <a ser especificado>
```

Request Header	Descrição
Accept	contém uma lista de esquemas de representação, separados por ponto-e-vírgulas, que serão aceitos em resposta a este pedido. Se nenhum campo `Accept:` estiver presente, então assume-se que text/plain e text/html são aceitos. Ex.: Accept: text/plain; text/html (significa: "se tiver text/html, envie-me, senão envie text/plain"); Accept: */* ("aceito qualquer conteúdo").
Accept-Charset	lista conjuntos de caracteres aceitos pelo cliente. Exemplo: Accept-Charset: iso-8859-5, unicode-1-1. O valor desse campo não deve incluir "US-ASCII" ou "ISO-8859-1", que já são assumidos como padrão.
Accept-Encoding	similar a `ACCEPT`, mas lista os mecanismos de codificação e valores de transferência que são aceitos na resposta. Ex.: Accept-Encoding: compress; gzip.
Accept-Language	similar a `ACCEPT`, mas lista o conjunto linguagens naturais que são preferíveis na resposta. Uma resposta em linguagem não especificada não será ilegal. Ex.: Accept-Language: dk, en-gb ("envie-me versão em dinamarquês, se tiver; senão, uma versão em inglês britânico").
Authorization	o valor deste campo consiste de credenciais contendo as informações de autenticação do *user agent* para o domínio do recurso sendo solicitado. Normalmente, o *user agent* procura se autenticar com um servidor depois de receber uma resposta de "acesso não autorizado".
From	contém um endereço e-mail Internet (RFC822) do usuário que controla o agente requisitante. Este campo pode ser usado para propósitos de loging e como meio de identificar a fonte de requisições inválidas ou indesejadas. O cliente não deve enviar o campo `FROM` sem aprovação do usuário, por questões de segurança e privacidade. O usuário deve ser capaz de desabilitá-lo. Ex.: From: jr@icmsc.sc.usp.br.
If-Modified-Since	campo utilizado com o método `GET` para torná-lo condicional: se o documento pedido não hover sido modificado desde o tempo especificado neste campo, o documento não será enviado, mas ao invés dele uma mensagem *Not Modified 304*. O propósito desta caracterísitca é permitir atualizações eficientes das informações do cache local, com um mínimo overhead de transações.
Pragma	sintaxe igual à dos outros campos multivalorados em HTTP, com uma lista de entradas separadas por vírgulas, para os quais os parâmetros opcionais são separados por ponto-e-vírgulas. Diretivas pragma devem ser compreendidas por servidores para os quais forem relevantes (por exemplo, um servidor proxy[11]). Até o momento, somente um pragma foi definido: ♦ *no-cache*: quando presente, o proxy não deve retornar um documento do cache, mesmo que não tenha expirado, mas deve sempre pedir o documento do servidor sendo acessado.

[11] *Proxy* é um servidor HTTP rodando em uma máquina firewall, fornecendo acesso ao mundo exterior para as pessoas que estão dentro da firewall. Além disso ele é capaz de trabalhar com cache para documentos, resultando em tempos de resposta mais rápidos.

Request Header	Descrição
Refereer	é um campo opcional que permite ao cliente especificar, para o bem do servidor, o endereço (URI) do documento (ou elemento de documento) do qual foi obtido o URI do qual se originou a consulta. Isto permite ao servidor gerar listas de back-links para documentos, logging, etc. Permite que links ruins sejam verificados para manutenção. Se um URI parcial for dado, ele deverá ser analisado em relação ao URI do objeto da consulta. Ex.: http://www.w3.org/hypertext/DataSources/Overview.html
User-Agent	se presente, dá o programa (software) usado pelo cliente original. Isto é somente para propósitos estatísticos e para verificar violações de protocolo.

3.3.2.3. RESPOSTA

A resposta do servidor deve começar com a seguinte sintaxe:

```
Response            = Simple-Reponse | Full-Response
Simple-Response     = [Entity-Body]
Full-Response       = Status-Line *(General-Header | Response-Header
                    | Entity-Header) CRLF [Entity-Body]
Status-Line         = HTTP-Version Status-Code Reason-Line>
                      CRLF
```

onde:

`<http version>`: identifica a versão do protocolo HTTP usado pelo servidor (default: HTTP/1.0)

`<status code>`: resultados codificados da tentativa de se entender e satisfazer o pedido;

`<reason string>`: fornece uma explicação para o usuário.

Os campos na linha de status são delimitados por um espaço em branco. Valores possíveis para o código de status:

Response headers - cabeçalhos dos objetos retornados em formato RFC822, bem como qualquer cabeçalho compatível com MIME: mesmas condições dos cabeçalhos de consulta.

Response data - informações adicionais, no formato de um corpo de mensagem MIME. O tipo de conteúdo usado para os dados pode ser qualquer "content-type" que o cliente tenha especificado ser capaz de receber. Assume-se que o cliente sempre pode manipular text/plain e text/html.

Uma resposta simples (*Simple-Response*) deve ser enviada somente em resposta a uma consulta HTTP 0.9 ou se o servidor suporta apenas o protocolo HTTP 0.9, mais limitado.

O elemento *Status-Code* é um código de três dígitos resultante da tentativa de se entender e satistafazer a consulta. O texto que o acompanha (*Reason-Phrase*) dá uma descrição desse código para o usuário.

O primeiro dígito do código de status define a classe da resposta:

Códigos de Status	Significado
INFORMATIONAL 1XX	Não utilizados, mas reservados para uso futuro
SUCCESS 2XX	A requisição foi recebida, entendida e aceita com sucesso.
200 OK	A requisição foi atendida.
201 Created	A requisição foi atendida e resultou na criação de um novo recurso, que pode ser acessado pelo URI retornado na resposta.
202 Accepted	A consulta foi aceita para processamento, mas o processamento não foi completado. A consulta pode estar ou não acontecendo no momento, já que ela pode estar desabilitada quando o processamento acontecer. Não existem facilidades de retorno de status de operações assíncronas como esta.
203 Provisional Information	Quando recebido em resposta a um comando GET, indica que a metainformação retonada não é um conjunto definitivo de um servidor com uma cópia do objeto, mas sim de uma teia privada. Pode conter, por exemplo, informações anotadas sobre o objeto.
204 No Content	O servidor atendeu a requisição mas não há novas informações para retornar.
REDIRECTION 3XX	Alguma outra ação deve ser feita pelo cliente para completar a consulta.
300 Multiple Choices	O recurso requisitado está disponível em uma ou mais versões e uma versão de preferência não pôde ser determinada via algoritmo de negociação de conteúdo.
301 Permanently Moved	O recurso pedido foi associado a um novo URI, e essa mudança é permanente. (browsers com capacidade de edição devem realizar essa religação à nova referência automaticamente, onde possível)
302 Temporarily Moved	O dado pedido na verdade está em um URI diferente, mas o redirecionamento pode ser alterado .
304 Not modified	O documento não foi modificado desde a data especificada em *If-Modified-Since*, assim o servidor não envia o documento para o cliente.
ERROR 4XX	Erro do cliente: a consulta contém má sintaxe ou não pode ser atendida
400 Bad request	Requisição com sintaxe errada ou impossível de ser atendida.
401 Unauthorized	O cliente pode tentar a consulta novamente, com um cabeçalho de autorização apropriado.
402 PaymentRequired	O cliente pode tentar de novo o pedido com um cabeçalho ChargeTo apropriado (para uso futuro, ainda não implementado).
403 Forbidden	Não existe autorização para a consulta.
404 Not Found	O servidor não encontrou algo que combinasse com o URI dado.
405 Method Not Allowed	O método especificado não é permitido para o recurso em questão.
406 None Acceptable	O servidor encontrou recursos no URI especificado, mas nenhum que satisfaça as condições dos campos *Accept*
407 Proxy Authentication Required	Similar ao código 401, referente a um proxy (a ser implementado).
408 Request Timeout	O cliente não produziu uma requisição dentro do tempo que o servidor estava preparado para esperar.
409 Conflict	A requisição não pôde ser completada devido a um conflito com o estado atual do recurso. A resposta deve incluir informações suficientes para que o usuário identifique a origem do conflito para resolvê-lo e submeter novamnte a requisição.
410 Gone	O recurso pedido não está mais disponível no servidor e não se conhece um enderço para redirecionar a requisição; esta condição é permanente.

Códigos de Status	Significado
ERROR 5XX	Erro do servidor: o servidor falhou em atender a uma consulta aparentemente válida.
500 Internal Server Error	O servidor encontrou uma condição inesperada que o impediu de atender o pedido.
501 Not implemented	O servidor não suporta a consulta.
502 Bad Gateway	O servidor recebeu uma resposta inválida de um gateway acessado na tentativa de completar a consulta.
503 Service Unavailable	O servidor não está sendo capaz de manipular a requisição no momento devido a sobrecarga ou manutenção do servidor
504 Gateway Timeout	O servidor não recebeu resposta de um gateway, dentro de determinado tempo, que foi acessado na tentativa de completar a consulta.

i. RESPONSE HEADER FIELDS

Os campos do cabeçalho relacionados a objetos em HTTP são todos opcionais. Estes cabeçalhos especificam metainformações: isto é, informações sobre o objeto, não informações sobre o que está contido no objeto.

```
Response-Header     = Public
                    | Retry-After
                    | Server
                    | WWW-Authenticate
                    | <a ser definido>
```

Response Header	Descrição
Public	Lista o conjunto de métodos suportados pelo servidor (além dos métodos comuns: GET, HEAD, ...) na conexão em andamento.
Retry-After	Indica quanto tempo o serviço poderá ficar indisponível.
Server	Contém informações sobre o software sendo usado pelo programa servidor.
WWW-Authenticate	Indica o esquema de validação para acesso ao URI.

ii. ENTIDADE

Mensagens *Full-Request* e *Full-Response* podem transferir uma entidade dentro de algumas requisições e respostas.

```
Entity          = Entity-Header Entity-Body

Entity-Header   = Allow
                | Content-Encoding
                | Content-Language
                | Content-Lenght
                | Content-Transfer-Enconding
                | Content-Type
                | Derived-From
                | Expires
                | Last-Modified
                | Link
```

```
| Location
| Title
| URI-header
| Version
| <a ser especificado>
```

O *Entity-Body* é incluído em uma mensagem de requisição somente quando o método exige (atuallmente, somente POST e PUT). O tipo dos dados contido numa entidade é determinado por campos de cabeçalho segundo o modelo:

```
Entity-Body ← Content-Transfer-Encoding(Content-Encoding(Content-Type))
```

O comprimento do conteúdo da entidade é determinado pelos campos `Content-Lenght`, `Content-Type`, `Content-Transfer-Encoding` ou pelo fechamento da conexão pelo servidor.

Foi proposto que qualquer elemento de metainformação HTML (permitido dentro do elemento `<HEAD>`) seja um candidato válido para cabeçalho de entidade. *Link* e *Title* são exemplos; *Base* será incluído em versões futuras.

Entity-Header	Ação
Allow	Lista o conjunto de métodos suportados pelo recurso identificado pelo URI.
Content-encoding	Especifica o mecanismo de codificação aplicado ao recurso, e qual decodificador deve ser usado.
Content-language	Descreve a linguagem natural da entidade.
Content-lenght	Indica o tamanho do conteúdo da entidade enviada (em números decimais ou octetos). Deve ser usado sempre que o comprimento da entidade possa ser determinado antes da transferência.
Content-transfer-encoding	Indica que tipo de transformação foi aplicada à entidade para transferí-la com segurança (é uma propriedade da mensagem e não do recurso).
Content-type	Indica o tipo de mídia do conteúdo da entidade enviada. Somente quando este campo não estiver presente (no caso de respostas simples), o destinatário deve tentar adivinhar o tipo do conteúdo por inspeção e/ou extensão do URL usado no acesso ao recurso.
Cost	Custo do objeto recuperado (um formato está para ser especificado).
Derived-From	Quando um objeto editado é submetido novamente, este campo dá o valor da versão.
Expires	Data após a qual as informações deixam de ser válidas e que não devem ser armazenadas em cache. Não significa que o objeto original deixará de existir.
Last-Modified	Última vez em que o objeto foi modificado
Link	Descreve um relacionamento entre a entidade e outro(s) recusro(s). Equivalente ao elemento `<LINK>` de HTML.
Title	Título do documento, considerado equivalente ao elemento `<TITLE>` de HTML.
URI	Pode conter um ou mais URIs pelos quais a origem do recurso contido na entidade pode ser identificado.
Version	Define a versão de um objeto; um cliente pode pedir determinada versão como resposta a uma consulta.

3.3.2.4 - NEGOCIAÇÃO DE CONTEÚDO

A negociação de conteúdo é uma caracterísitca opcional do protocolo HTTP. Ela foi projetada para permitir uma seleção preemptiva de uma representação de maior preferência para o conteúdo, dentro de uma comunicação requisição-resposta, sem intervenção do usuário.

O primeiro passo no algoritmo de negociação é a determinação, pelo servidor, da existência ou não de variantes para o recurso requisitado. Se não houver formas variantes do recurso, a negociação é limitada à aceitação daquele tipo único de acordo com as restrições especificadas no campo `Accept` (quando existir).

Havendo variantes disponíveis, elas são comparadas com os tipos especificados em *Accept-Encoding*, *Accept-Charset* e *Accept*. Se restarem variantes aceitáveis, é analisado o campo *Accept-Language*. Restando ainda mais de uma variante aceitável, a seleção é feita por cálculo e comparação da qualidade relativa dos tipos disponíveis.

Os pesos caculados são normalizados em um número real entre 0 (mínimo) e 1 (máximo), Os cálculos envolvem os seguintes parâmetros:

`q`	Fator de qualidade escolhido pelo *user-agent* (configurável pelo usuário) representando a receptividade do tipo de mídia do recurso. Essa receptividade é uma medida da capacidade de apresentação dos conteúdos desse tipo de mídia ao usuário. Default: 1.
`qs`	Fator de qualidade escolhido pelo servidor (através de algum mecanismo ainda não especificado) para representar a qualidade relativa de uma determinada variante de representação do recurso.
`mxb`	número máximo de bytes aceitável pelo cliente, no *Entity-Body*. Default: infinito (*undefined*)
`bs`	número de bytes no *Entity-Body* para uma dada representação da fonte. Deve ser igual ao valor de *Context-Lenght*. Default: 0.

Essa função é definida como:

```
                    { if mxb=undefined, then (qs*q) }
Q(q,qs,mxb,bs)=     { if mxb >= bs, then (qs*q) }
                    { if mxb < bs, then 0 }
```

As variantes com maior valor para a função `Q` serão as representações preferidas para a entidade. Existindo diversas representações para um único tipo de mídia (variando apenas pelo campo *Content-Encoding*), então a representação menor (com menor bs) é a preferida.

Finalmente, restando ainda escolhas disponíveis, o servidor pode escolher uma variante (aleatoriamente) e enviá-la, ou responder com uma mensagem *300 Multiple Choices*.

3.3.2.5 - CONSIDERAÇÕES DE SEGURANÇA

O protocolo HTTP permite consultas para comunicação com máquinas servidoras remotas, e aqui se aplicam todas as considerações de segurança para sistemas cliente-servidor.

◊ *Validação de Acesso*

O protocolo HTTP provê um mecanismo simples de autorização que pode ser usado por um servidor para recusar uma requisição (através de resposta *401 Unauthorized*), e por um cliente para dar informações de validação. Um valor de domínio, mais o URL raiz do servidor acessado, definem o espaço de validação. Estes domínios permitem que os recursos protegidos sejam particionados em um conjunto de espaços de autorização, cada um com seu próprio esquema de validação e/ou banco de dados.

O esquema básico de validação segue o modelo em que um cliente deve se identificar através de um par *user-ID/password* para cada domínio. Não é um método seguro para filtrar acessos não-autorizados, pois assume que a conexão entre o cliente e o servidor seja segura - o que não é verdade, geralmente, em uma rede aberta.

O HTTP/1.0 não impede que outros esquemas e mecanismos de criptografia sejam utilizados, além do esquema básico de validação, para melhorar a segurança.

◊ *Métodos*

O software do cliente representa o usuário em suas interações sobre a rede; os autores desse software devem ter o cuidade de permitir que o usuário tome conhecimento de todas as ações que podem ser tomadas e que causem efeitos inesperados.

◊ *Abuso de informação de log*

Dois campos de cabeçalho são importantes neste contexto: *Referer* e *From*.

O campo *Referer* permite que se estudem padrões de leitura, e que sejam rastreados links reversos. Embora seja bastante útil, mesmo removendo informações do usuário esse campo pode acabar identificando um documento que se espera estar sob segurança.

As informações enviadas no campo *From* podem conflitar com os interesses de privacidade do usuário ou da segurança de seu site, e portanto não devem ser transmitidas sem que o usuário seja capaz de habilitá-lo, desabilitá-lo, e modificar seu conteúdo.

3.3.3 - LINGUAGEM DE FORMATAÇÃO DE HIPERTEXTO

Antes de conhecermos a linguagem HTML, é preciso apresentar brevemente dois padrões fundamentais:

♦ HyTime - Hypermedia/Time-based Document Structuring Language

HyTime (ISO 10744:1992) - padrão ISO para representação estruturada de hipermídia e informação baseada em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (áudio, vídeo, etc.), conectados por webs ou hiperlinks.

O padrão HyTime é independente dos padrões de processamento de texto em geral. Ele fornece a base para a construção de sistemas hipertexto padronizados, consistindo de documentos que aplicam os padrões de maneira particular.

♦ SGML - Standard Generalized Markup Language

SGML (ISO 8879:1986) é um dos padrões gerados pelo programa CALS - Computer-aided Acquisition and Logistics Support (programa do Departamento de Defesa Americano para adquirir e gerenciar eletronicamente informações técnicas sobre sistemas de armamento). O SGML é um padrão de formatação de textos: não foi desenvolvido para hipertexto, mas torna-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações.

O padrão SGML não é aplicado de maneira padronizada: todos os produtos SGML têm seu próprio sistema para traduzir as etiquetas de formatação para um particular formatador de texto.

◊ **Document Type Definition (DTD)**

O DTD é uma definição formal da sintaxe para uma dada classe de documentos, em termos do padrão SGML. Um DTD ou uma referência para um DTD deve estar contido em qualquer documento que siga esse padrão. Ele não contém informação sobre como processar um documento ou como ele deve ser apresentado.

3.3.3.1 - A LINGUAGEM HTML

Em uso desde 1990, a HTML (HyperText Markup Language) é uma linguagem simples de formatação, usada para criar documentos hipertexto independentes de plataforma. HTML é uma aplicação do padrão SGML.

A especificação da linguagem HTML foi escrita pela última vez como um *Internet draft* em agosto de 1995. Há um conjunto de características relativamente estável na prática atualmente, mas não existe documento que os especifique. Uma versão final do documento de definição da linguagem (DTD) continua em discussão.

Uma linguagem HTML nível 1 é exigida para todos os clientes WWW; baseia-se na linguagem entendida pelos primeiros browsers, mais a inclusão de imagens (característica inserida pelo Mosaic). A HTML nível 2 é um superconjunto do nível 1, incluindo também formulários para entrada de dados pelo usuário. Esse nível da linguagem está sendo refinado.

Acima do nível 2, a linguagem HTML inclui características tais como tabelas, figuras, e equações matemáticas - versão 3.0 da linguagem.

A linguagem é definida para ser uma linguagem de comunicação, que realmente percorra a rede. Não é obrigatório que os arquivos sejam armazenados em HTML. Os servidores podem armazenar arquivos em outros formatos, ou em variações dessa linguagem que incluem informações extras que tenham interesse somente local, e então gerar HTML durante a resposta a uma consulta.

HTML não funciona como um processador de textos WYSIWYG porque o mesmo documento pode ser visto por muitos browsers diferentes com diferentes características. Portanto, um elemento da linguagem que marca um parágrafo pode ser interpretado de maneiras particulares diferentes pelos programas que irão mostrar aquele documento para o usuário: um browser pode tabular o início de um parágrafo, enquanto outro pode apenas deixar uma linha em branco.

Com a falta de uma padronização bem definida da linguagem, foi possível o desenvolvimento de extensões paralelamente ao trabalho desenvolvido pelo CERN e IETF[12]. O exemplo trivial dessa situação são as extensões aceitas pelo Netscape Navigator: implementaram-se algumas das formatações do HTML 3.0, juntando-se características próprias desse browser. Em sua versão 2.0, além do browser ser também um editor WYSIWYG, ele incorpora elementos da linguagem *Java* e *scripts* especiais. A distribuição massiva desse programa leva autores de hipertexto a se esquecerem dos diversos usuários de outros browsers, a tal ponto que seu produto fique restrito aos usuários do Netscape.

Uma maneira de contornar esse problema é empregar essas extensões tendo em vista não prejudicar a apresentação dos documentos para usuários dos diversos browsers.

3.3.3.2 - IDENTIFICAÇÃO DE DOCUMENTOS HTML

Para identificar um documento como um documento que segue uma versão determinada de HTML, ele deve ser iniciado com uma declaração de versão; quando não estiver indicado o tipo de documento, será admitida a versão HTML 2.0 mais geral. Algumas declarações (da mais geral para a mais específica):

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN//">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN//">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 2//EN//">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Level 2//EN//">
```

Documentos que seguem HTML 3.0 são identificados por:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0 //EN//">
```

3.3.3.3 - ELEMENTOS DE HTML

Instruções HTML são chamadas ***elementos***, e podem ser divididas em duas categorias:

---

[12] *IETF* - Internet Engineering Task Force

• elementos que definem como o corpo do documento será mostrado pelo browser, e
• elementos que definem informações sobre o documento, tal como o título ou relacionamentos com outros documentos.

Os elementos da linguagem são denotados por *etiquetas* (*tags*). Elas consistem de um símbolo "<", seguido pelo nome da etiqueta, e fechada por um ">". A maioria dos elementos marca blocos do documento para propósitos específicos ou de formatação. Nesse caso, um elemento `<nome-do-elemento>` marca o início de uma seção de texto, e `</nome-do-elemento>` marca o fim dessa seção.

Alguns elementos são *vazios*: isto é, eles não afetam a formatação de um bloco do documento e não têm etiqueta de fechamento; outros, tais como itens de listas, dispensam essa etiqueta.

Muitos elementos podem ter argumentos que passam parâmetros para o interpretador que os apresenta. Esses argumentos são chamados *atributos* do elemento; veremos exemplos de atributos na definição do elemento `<A>`, que marca uma região do documento para ligação de hipertexto.

Nem todos os browsers suportam todas as etiquetas. Se um browser não suportar determinada *tag* de formatação, ele simplesmente a ignora.

Os nomes das etiquetas não são sensíveis à caixa.

3.3.3.4. ELEMENTO <HTML>

Um ***documento*** HTML é composto por um elemento `<html>...</html>` composto por `<head>...</head>` e `<body>...</body>`:

```
<html>
<head>...</head>
<body>...</body>
</html>
```

3.3.3.5. ELEMENTO <HEAD>

O elemento `<head>` contém informações gerais sobre o documento, que não fazem parte do documento mostrado ao usuário. A versão 2.0 apresenta os seguintes elementos dentro de `<head>`:

`<title>... <\title>`

Título do documento. Todo documento HTML deve ter um título, que geralmente identifica a janela em que é mostrado, e é usado para identificação dos documentos em outros documentos (p.ex., em uma busca WAIS, ou em uma lista *bookmarks* de browsers gráficos). Cada documento tem apenas um título, que deve ser em texto simples - sem links. Recomenda-se que os títulos sejam sugestivos (evitando-se títulos tais como: "*Introdução*"), com menos de 64 caracteres para que não sejam truncados em alguns aplicativos, e sem símbolos especiais (acentos, por exemplo)

`<isindex>`

Normalmente gerado automaticamente pelo servidor ou por um *server script*, ou por um programa do servidor, permite buscas por palavra-chave. Um servidor não sabe como realizar buscas; ele precisa acessar um programa projetado para buscar arquivos ou pesquisar bancos de dados. A interface entre esses programas e o servidor http é um *script*, normalmente localizado no diretório /cgi-bin/ e acessado por

```
http://server/cgi-bin/progr[?arg1+arg2]
```

Um exemplo será apresentado mais adiante, sob *Common Gateway Interface (CGI)*.

<nextid>

Incluído apenas por motivos históricos, sugere um nome para o próximo identificador de uma cabeça de âncora (*ver adiante, sobre hiperlinks*).

<link>

Define os relacionamentos de um documento com outros documentos; um documento pode ter diversos LINKs - por exemplo, indicando o documento "pai", "próximo", "anterior". Os atributos são os mesmos dos elementos de âncora.

↳ Em HTML 3.0, LINK é muito utilizado, definindo os relacionamentos de um documento com outros - glossários, índices, notas de rodapé, por exemplo - principalmente através do atributo REL. Ver ítens sobre *links* e *style sheets*, mais adiante.

<base>

URL original absoluto do documento. Permite que se mova o documento para outro diretório ou mesmo outro *site* e se realizem acessos relativos em relação ao URL em BASE.

<meta>

Identifica metainformações especializadas associadas ao documento. Cada elemento meta especifica um par nome/valor; o método pelo qual um servidor extrai essas metainformações não está especificado, sendo seu efeito dependente da implementação do servidor e do *user-agent*.

Atributos:

HTTP-EQUIV
: Liga o elemento a um campo de cabeçalho HTTP. Um servidor HTTP pode usar esta informação para processar o documento.

NAME
: Especifica o nome do par nome/valor.

CONTENT
: Especifica o valor do par nome/valor.

Exemplos:

```
<META HTTP-EQUIV="Expires" Content="Tue, 13 Sep 1995 21:00:32 GMT">
<META HTTP-EQUIV="Reply-to" Content="edsmorei@icmsc.sc.usp.br">
```

Essas informações são incluídas pelo servidor como campos de cabeçalho, como parte da resposta para um GET ou HEAD para aquele documento.

3.3.3.6. ELEMENTO <BODY>

O elemento <body> abrange todo o conteúdo de um documento. BODY apresenta elementos que formatam o texto, indicando cabeçalhos, parágrafos, listas, ligações hipertexto e imagens.

↳ Em HTML 3.0, <body> ganha diversos atributos:

background - permite a inserção de uma cor (padrão RGB) ou desenho (imagem) para o fundo do documento.

banner - seção que aparece no topo da janela e não muda com o *scroll* do conteúdo da página. Útil para logotipos, declarações de copyright, avisos, controles de navegação especiais.

spot - insere "cabeças de link" em pontos arbitrários
div - usado para representar classes diferentes de texto, por exemplo: capítulo, seção, resumo, apêndice.

i. CABEÇALHOS

Há seis níveis de cabeçalhos, numerados de 1 a 6; H1 é o principal, apresentado na maior fonte disponível, e H6 com a menor fonte. A sintaxe de cabeçalho é: <Hnum> texto do cabeçalho </Hnum>, para num = 1,...,6.

↳ Em HTML 3.0, os cabeçalhos ganham diversos atributos para alinhamento, numeração, e inserção de imagens.

ii. PARÁGRAFOS

Diferentemente de documentos na maioria dos processadores de texto, quebra de linha (*line feed*) não é significante em arquivos HTML, sendo o comprimento da linha determinado pela largura da janela aberta pelo browser.

A tag <P> é utilizada sempre que se quer separar dois blocos de texto em parágrafos lógicos; em geral, não se usa parágrafo dentro de cabeçalhos, listas, ou blocos com outro tipo de formatação: esses elementos gerenciam seu próprio espaçamento. Parágrafos não contêm cabeçalhos, mas podem conter formatações de caracteres e listas, inclusive com aninhamento.

↳ Em HTML 3.0, é sugerido (mas não exigido) que todo parágrafo seja formatado com a tag inicial <P> e com a tag final </P>. Formalmente, essa mudança apenas reflete o fato de que o elemento "parágrafo" não é *vazio*. A maioria dos browsers aceita ambas as formas, dentro do critério de compatibilidade de versões estipulada pela WWW.

É possível controlar o alinhamento dos parágrafos (atualmente, é feito somente à esquerda, embora o Netscape 1.1 aceite o alinhamento centralizado). Exemplos:

```
<P ALIGN=center>...</P>
<P ALIGN=left>...</P>
<P ALIGN=right>...</P>
<P ALIGN=justify>...</P>
```

iii. TEXTOS ESPECIAIS

HTML permite três modos texto especiais. São chamados modos lógicos, porque definem blocos de texto por seus propósitos lógicos. A maneira em que os blocos serão mostrados é determinada pelo browser.

▪ *TEXTO PRÉ-FORMATADO*

O elemento <PRE> é usado para mostrar um texto exatamente da maneira em que foi escrito, respeitando mudanças de linha e tabulações; é útil para exibir texto que tenha sido formatado para um teletipo, ou para outro display com tamanho de caracteres fixo.

Como o HTML padrão (até 2.0) não tem o elemento <TABLE>, somente <PRE> consegue formatar tabelas estruturadas de modo apropriado.

<PRE> tem um atributo opcional WIDTH, que especifica o número máximo de caracteres que podem ser escritos em uma única linha. Um valor default de 80 sempre é assumido, mas isso depende do browser (o Mosaic, por exemplo, ignora completamente esse atributo).

Não se deve usar elementos que definem formatação de parágrafos dentro do elemento <PRE>. É permitido o uso de âncoras.

↳ Em HTML 3.0, existem elementos para formatação de tabelas; o Netscape aceita essa formatação, porém nem todos os browsers apresentam tabelas corretamente. Ver seção sobre tabelas em HTML 3.0.

▪ *ENDEREÇOS*

<ADDRESS> é geralmente usado para informações de endereço, assinaturas, autoria, etc. Normalmente, é o último (ou primeiro) item de um documento, especificando o autor do mesmo, e um meio de contactá-lo (endereço E-mail, por exemplo). Normalmente mostrado em itálico.

▪ *BLOCOS ENTRE ASPAS*

A tag <BLOCKQUOTE> permite que textos de citações (trechos de livros, frases históricas, etc.) sejam mostrados de maneira diferenciada. Normalmente, com uma tabulação, ou em itálico. Também causa uma quebra de parágrafo, e força um espaço em branco antes e depois do bloco.

iv. LISTAS

HTML suporta diversos elementos para fazer listas. Elas podem ser divididas em dois tipos:

- listas de definição - denotadas pelo elemento <DL>;
- listas regulares - denotadas pelos elementos <UL>, <OL>, <MENU>, e <DIR>.

As listas podem ser aninhadas, portanto pode-se ter uma lista regular dentro de uma lista regular, uma lista regular dentro de uma lista de glossário, e assim por diante. O atributo COMPACT indica uma apresentação compacta da lista. Esse atributo, entretanto, é ignorado pela maioria dos browsers.

▪ *LISTAS REGULARES*

Uma lista regular é uma seqüência de parágrafos <LI>, cada qual podendo ser precedido ou não por uma marca especial ou número de seqüência. As listas têm etiquetas de fechamento.

<UL>	*Lista não ordenada*	seqüência de parágrafos com diversas linhas, listados separadamente e normalmente marcados por um *bullet* ou símbolo semelhante.
<OL>	*Lista numerada* (ou *ordenada*)	seqüência de parágrafos com diversas linhas, listados separadamente e ordenados numericamente.
<MENU>	*Lista MENU*	lista não ordenada com ítens mais curtos; é similar a <UL>, mas formatado de maneira compacta (quando possível).
<DIR>	*Lista DIR*	lista de elementos curtos, com até 20 caracteres. Os ítens podem ser mostrados em colunas ao longo da página em vez de um abaixo do outro. Depende do browser utilizado.

↳ Em HTML 3.0, o autor pode escolher os estilos de numeração, ícones para substituir os *bullets*, e listas com várias colunas. Ver adiante o conceito de *"style sheets"*. Também está em discussão a criação de um elemento <LH> (*list header*). HTML 3.0 elimina os elementos <MENU> e <DIR>.

▪ *LISTA DE DEFINIÇÃO*

Usada para apresentar uma lista de ítens com respectivos parágrafos descritivos. Pode ser usada para glossários, mas também é útil para apresentar listas de ítens e seus significados. A lista de definição é formatada pelo elemento <DL>. Os ítens da lista são introduzidos pelos elementos:

- <DT> - o "termo", em uma linha única;
- <DD> - a "definição", que pode ter diversas linhas.

v. *FORMATAÇÃO DE CARACTERES*

Há dois tipos de formatação de caracteres: *lógico* e *físico*. Os efeitos obtidos na tela são os mesmos, porém o motivo da distinção entre os dois é ligado à filosofia de SGML: num universo ideal para esse tipo de linguagem, o conteúdo está divorciado da apresentação.

A formatação lógica segue o significado lógico do texto marcado. A formatação física especifica explicitamente o estilo que se quer para o texto: em itálico, grifado, sublinhado, etc.

Portanto, quando a linguagem especifica um cabeçalho de nível 1, por exemplo, ela apenas especifica um cabeçalho de nível 1, sem especificar que esse tipo de cabeçalho deve estar "centralizado no topo da página, no tipo Times New Roman de 24 pontos". A vantagem disso é que se o programador decidir mudar os cabeçalhos de nível 1 para "Courier de 20 pontos justificado à esquerda", essa mudança só precisa ser feita na definição dos cabeçalhos de nível 1 no dispositivo de apresentação - isto é, no browser.

Recomenda-se o uso de formatações lógicas, pois elas serão interpretadas de maneira mais consistente de browser para browser.

▪ *ESTILOS LÓGICOS*

`<EM>` ênfase; normalmente em itálico
Ex.: `<EM>Área do usuário com 100% de ocupação!!</EM>`

`<CITE>` nomes ou títulos de livros, artigos, filmes, etc.; em itálico
Ex.: `Ver <CITE>Guia do servidor WWW</CITE>`

`<CODE>` exemplos de código
Ex.: `<CODE>if (true) then (do something)</CODE>`

`<KBD>` indica entrada por teclado
Ex.: `Entre <KBD>passwd<KBD> para acesso`

`<SAMP>` seqüência de caracteres
Ex.: `<SAMP>Erro de Aplicativo</SAMP>`

`<STRONG>` forte ênfase; em bold
Ex.: `<STRONG>Obs:</STRONG> indique a fonte consultada`

`<VAR>` nome de variável, mostra onde o usuário irá substituir com um valor específico; em itálico
Ex.: `<KBD>rm<KBD> <VAR>arquivo</VAR> remove o arquivo`

▪ *ESTILOS FÍSICOS*

`<B>` bold quando disponível (em alguns browsers, pode aparecer sublinhado)

`<I>` itálico (em alguns casos, caracteres inclinados)

`<TT>` tipo "teletype"

▪ *CARACTERES ESPECIAIS*

HTML permite que caracteres especiais sejam representados por seqüências de escape, indicadas por três partes: um "&" inicial, um número ou cadeia de caracteres correspondente ao caracter desejado, e um ";" final.

Três caracteres ASCII - <, >, & - têm significados especiais em HTML, e para usá-los em um documento é preciso entrar uma *seqüência de escape* correspondente:

Entidade	*Caracter*
<	<
>	>
&	&

Outras seqüências de escape suportam caracteres *ISO Latin1*, porém, nem todos os browsers suportam essas entidades. Abaixo, alguns caracteres utilizados em Português (note que as seqüências de escape são sensíveis à caixa):

Entidade	*Caracter*	*Entidade*	*Caracter*
Á	Á	á	á
Â	Â	â	â
À	À	à	à
Ã	Ã	ã	ã
Ç	Ç	ç	ç
É	É	é	é
Ê	Ê	ê	ê
Í	Í	í	í
Ó	Ó	ó	ó
Ô	Ô	ô	ô
Õ	Õ	õ	õ
Ú	Ú	ú	ú
Ü	Ü	ü	ü

☹ *Por que usar essas formatações?*

Os browsers costumam também mostrar corretamente os caracteres acentuados normalmente - á no lugar de `á` por exemplo. Essa prática, embora facilite sobremaneira a digitação dos documentos, não é recomendada devido a um problema relativo à transmissão desses caracteres.

Chama-se *conjunto de caracteres* uma representação digital de texto. Um caracter é uma unidade atômica de comunicação, um símbolo cujas diversas representações devem significar a mesma coisa para uma comunidade de pessoas. Na prática, porém, existem alguns conjuntos que associam dois números distintos a um mesmo caracter; é preciso seguir um padrão determinado que garanta a apresentação correta dos caracteres após a transmissão dos mesmos pela rede até o browser do usuário.

A aplicação da HTML na WWW ficou muito restrita devido ao fato de se basear no conjunto de caracteres ISO-8859-1, ou ISO Latin-1, apropriado somente para línguas da Europa Ocidental. Usuários de outras línguas têm utilizado extensões da linguagem HTML que consideram seus caracteres especiais. Entretanto, browsers de diferentes países não conseguem apresentar os caracteres especiais das diversas línguas por não haver um padrão incorporado ao HTML para todos esses caracteres especiais.

Uma proposta para resolver esse problema sugere a adoção do padrão *Base Multilingual Plane* - ISO 10646:1993, também conhecido por UCS-2 (idêntico ao padrão UNICODE). Desse modo, além de evitar interpretações ambíguas, o limite máximo de referências numéricas para caracteres passa de 255 para 65.533. A proposta inclui um modelo de reconhecimento do conjunto de caracteres antes do documento ser processado como documento HTML. O modelo faz com que as referências numéricas sejam sempre resolvidas para os mesmos caracteres, qualquer que seja a codificação externa usada.

Para garantir a interpretação apropriada de um documento, pode-se incluir uma indicação de esquema de codificação para suprir uma falha dos browsers e servidores atuais, através de um campo:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
```

↳ O HTML 3.0 inclui diversos símbolos especiais, tais como os de marca registrada (`™`), copyright (`©`), e outros símbolos-padrão que substituem ícones de uso mais comum. Inclui também elementos que padronizam o uso de caracteres gregos e símbolos matemáticos.

vi. LIGAÇÃO A OUTROS DOCUMENTOS

O grande poder de HTML vem de sua habilidade em ligar regiões de texto (e também imagens) a outro documento. O browser destaca essas regiões (com cor diferente ou sublinhando-as) para indicar que são ligações de hipertexto (também chamadas *hiperlinks* ou *links*).

Um *hiperlink* é um relacionamento entre duas âncoras, chamadas "cabeça" (região apontada pelo link) e "cauda" (região de onde parte o link).

A única etiqueta relacionada a hipertexto em HTML é `<A>` (de *anchor*: âncora). Uma âncora é um pedaço de texto ou outro objeto (por exemplo, uma imagem) que marca uma ligação de hipertexto.

▪ *ATRIBUTO "HREF"*

Para incluir uma âncora em um documento, escreve-se o parâmetro:

```
<A HREF = "nome-de-arquivo">hiperlink</A>
```

onde "nome-de-arquivo" é o documento que está sendo apontado, e hiperlink é o texto (ou imagem) que servirá de ligação hipertexto no documento sendo mostrado.

Uma referência `<A HREF = "servicos.html">Serviços</A>` faz do texto `Serviços` o hiperlink para o documento `servicos.html` que, neste caso, estaria no mesmo diretório do documento em que está esse link. Pode-se referenciar documentos de outros diretórios especificando o caminho relativo do documento atual até o documento relacionado. Estando no diretório anterior, poderíamos ter um link para um documento sobre a cidade de São Carlos, que está em outro diretório, definido da seguinte forma:

```
<A HREF = "/Sao_Carlos/index.html">Cidade de São Carlos</A>
```

(que significa o mesmo que:

```
<A HREF = "/Sao_Carlos/">Cidade de São Carlos</A>
```

porque o arquivo `index.html` é o arquivo *default* dentro de um diretório. Esse diretório está sendo referenciado com relação ao diretório raiz de um servidor.)

Esse caminho, do documento atual a outro documento, pode ser um URL (Universal Resource Locator), indicando documentos em outros servidores:

```
Veja também: <A HREF = "http://www.w3.org/">WWW</A>
```

Esta linha é mostrada desta forma (com WWW em destaque):

Veja também: **WWW**

A palavra `WWW` será um link para o servidor do W3 Consortium, na Suíça, que se encontra no endereço `http://www.w3.org/`.

▪ *ATRIBUTO "NAME"*

Marca a âncora como uma possível "cabeça" de link vindo de outro documento, ou dentro do mesmo documento. Podemos ligar um documento A.html a uma seção específica de um documento B.html, estabelecendo um "nome-de-âncora" no documento B.html:

```
<A NAME = "nome-de-âncora">Sobre hiperlinks</A>
```

No documento A.html, referenciamos essa seção deste modo:

```
<A HREF = "B.html#nome-de-âncora">Hiperlinks</A>
```

↳ No HTML 3.0, `NAME` é substituído pelo atributo `ID`.

▪ *ATRIBUTO "TITLE"*

Atributo pouco usado e suportado por poucos browsers atuais. Também é um atributo opcional e serve apenas para dar informação adicional. Se estiver presente, seu argumento deve ser o TITLE do documento cujo endereço é dado pelo atributo HREF.

Pode ser útil para:

• o browser mostrar o título do documento antes de buscá-lo, por exemplo em uma nota na margem, ou em uma caixinha quando o mouse estiver sobre a âncora.

• alguns documentos - principalmente os que não são em HTML, como gráficos, textos ASCII (*plain text*) e menus Gopher - não vêm com um título. Portanto, pôr um título no link é uma maneira de dar a eles um título significativo.

▪ *ATRIBUTO "REL"*

Usado para dar os relacionamentos descritos pela ligação de hipertexto - portanto, REL só pode ser usado na presença de HREF. O valor para REL é uma lista de valores de relacionamentos, separados por vírgulas.

▪ *ATRIBUTO "REV"*

REL e REV são raramente usados, mas terão importância crescente à medida em que documentos HTML e ambientes de desenvolvimento desses documentos se tornem mais sofisticados.

REV é semanticamente inverso a REL. Um link de A para B com REL="X" significa a mesma coisa que um link de B para A com REV="X".

▪ *ATRIBUTO "URN"*

Atributo não utilizado atualmente, nem suportado por browsers ou servidores atuais. É um identificador mais persistente para a âncora.

▪ *ATRIBUTO "METHODS"*

Não é muito usado, e é suportado por poucos browsers. Se presente, seu argumento é uma lista de métodos HTTP suportados pelo objeto e acessíveis pelo usuário, separados por vírgulas. Essa lista é dada pelo protocolo HTTP, mas pode ser útil incluir essa informação antes, na referência da âncora.

vii. QUEBRA DE LINHA

 força uma quebra de linha sem espaço extra entre linhas (diferente de <P>). Não tem etiqueta de fechamento (é um elemento *vazio*).

viii. LINHA HORIZONTAL

<HR> é usado para desenhar uma linha horizontal dividindo completamente a tela. Pode separar logicamente dois blocos de texto, ou listas de ícones de um corpo de texto. É um elemento vazio.

↳ HTML 3.0 permite o atributo SRC (ver em "Inserção de imagens") para usar em <HR> uma linha criada pelo autor do documento.

ix. INSERÇÃO DE IMAGENS

O elemento <IMG> permite que um documento com imagem seja inserido em um documento HTML, apresentando texto e imagem em uma mesma página, ao invés de usar links hipertexto para recuperar e mostrar imagens separadamente. Este elemento não pode ser usado para incluir outro texto HTML dentro de um dado documento. As imagens podem ser incluídas dentro de uma âncora de

hipertexto, fazendo com que a imagem funcione como um botão para acessar outros documentos. É um elemento vazio, isto é, não existe uma etiqueta de fechamento.

<IMG> tem os seguintes atributos:

• SRC="URL_da_imagem": SRC dá o URL do documento com a imagem a ser mostrada. O esquema é o mesmo das âncoras de hipertexto. Este atributo deve estar sempre presente.

• ALIGN= "alinhamento": diz ao browser como alinhar a imagem em relação ao texto circundante. Os valores BOTTOM, MIDDLE, e TOP fazem com que o alinhamento seja de acordo com o fim, o começo ou o meio do texto.

• ALT="texto alternativo": alguns browsers não podem mostrar imagens; ALT permite que seja especificado um texto alternativo para ser mostrado por browsers de modo texto.

• ISMAP: formata a imagem como um mapa ativo; desse modo, o usuário pode acionar o mouse sobre ela e obter diferentes resultados de acordo com a região da imagem sobre a qual houve o acionamento.

↳ Em HTML 3.0, existem ainda os atributos de alinhamento LEFT, RIGHT, HEIGHT (especifica altura para a imagem), WIDHT (especifica largura para a imagem) e UNITS (indica unidades - pixels, por exemplo - para altura e largura da imagem).

x. *TABELAS*

↳ Disponíveis somente em HTML 3.0. Ver seção adiante.

xi. *FORMATAÇÃO DE EQUAÇÕES*

Não há elementos HTML para formatação de equações e fórmulas matemáticas. Uma opção é usar o LaTex2HTML, que converte textos LaTex (que contenham equações) para arquivos .gif, e então transformá-los para formato GIF89a transparente. Resulta em uma imagem que pode ser inserida em documentos usando-se a etiqueta <IMG>.

↳ Ver sobre *elementos matemáticos* em HTML 3.0.

xii. *IMAGENS EXTERNAS, SONS E ANIMAÇÕES*

Pode-se querer ter um arquivo imagem aberto separadamente quando um usuário ativa um link. Isto é considerado uma imagem externa e é útil quando não se quer diminuir a velocidade de carregamento devido às imagens inseridas no texto do documento principal:

<A HREF = URL_da_imagem>âncora</A>

A mesma sintaxe é usada para ligações para animações, imagens externas e sons.

xiii. *COMENTÁRIOS EM HTML*

Indicados por <!--, terminados por -->. Esses símbolos devem ser colocados no começo e fim de toda linha a ser tratada como comentário.

Comentários não são aninhados e a seqüência "--" não deve aparecer dentro de um comentário exceto como parte da etiqueta de fechamento de comentário.

Alguns browsers, em particular o Mosaic, dão importância a etiquetas HTML que estejam dentro de comentários.

xiv. FORMULÁRIOS

Um formulário é um modelo para um conjunto de dados (inseridos no formulário), e um URI com método e ação associados. O elemento <FORM> contém uma seqüência de elementos de entrada e elementos de formatação do documento; esse elemento tem os seguintes atributos:

FORM ACTION	especifica o URI da ação (onde será desenpenhada a ação)
FORM METHOD	seleciona método de acesso ao URI
FORM ENCTYPE	especifica tipo de mídia usado para codificar os dados de entrada do formulário

O elemento <INPUT> representa um campo para entrada pelo usuário, com os seguintes atributos:

INPUT TYPE=TEXT	indica uma linha de entrada
INPUT TYPE=TEXTAREA	indica várias áreas de entrada
Atributos:	
NAME	nome do campo
Opcionais:	
MAXLENGHT	restringe número de caracteres que podem ser entrados
SIZE	especifica tamanho do espaço no display para mostrar o formulário
VALUE	valor inicial do campo
INPUT TYPE=PASSWORD	funciona como um campo texto, porém o valor é escondido na entrada dos dados.
INPUT TYPE=CHECKBOX	Um conjunto de elementos com mesmo nome representa um campo de múltipla escolha, que pode ter diversos valores
Atributos:	
NAME	nome simbólico para uma das alternativas
VALUE	valor do campo dado pelo elemento
Opcional:	
CHECKED	estado inicial "positivo"
INPUT TYPE=RADIO	representa escolha booleana para um campo que pode ter apenas um de vários valores. Possui os mesmos atributos de checkbox; apenas um dos "radio buttons" pode ficar em estado inicial "positivo".
INPUT TYPE=IMAGE	especifica uma imagem para ser mostrada, permitindo que sejam escolhidas coordenadas x e y de um pixel da imagem
INPUT TYPE=HIDDEN	representa um campo oculto
INPUT TYPE=SUBMIT	representa a submissão do formulário pelo *user agent* (*browser*)
Opcional:	
NAME	indica que esse elemento também contribui com um campo para o formulário
INPUT TYPE=RESET	instrui o *user agent* a restabelecer os valores iniciais do formulário

O elemento <SELECT> restringe o formulário a uma lista enumerada de valores, dados pelo elemento <OPTION>. Atributos:

MULTIPLE	indica que mais de uma opção pode ser incluída no valor
NAME	especifica nome do campo
SIZE	especifica número de ítens visíves
OPTION	
Atributos:	
SELECTED	campo selecionado inicialmente
VALUE	valor a ser retornado (default: conteúdo de OPTION)

O elemento <SELECT> restringe o formulário a uma lista enumerada de valores, dados pelo elemento <OPTION>. Atributos:

MULTIPLE		indica que mais de uma opção pode ser incluída no valor
NAME		especifica nome do campo
SIZE		especifica número de ítens visíves
OPTION		
Atributos:		
	SELECTED	campo selecionado inicialmente
	VALUE	valor a ser retornado (default: conteúdo de OPTION)
TEXTAREA		
Atributos:		
	COLS	número de colunas visíveis
	NAME	nome do campo
	ROWS	número de linhas visíveis
METHOD=GET		O processamento do formulário não implica em ações que modificam o estado de quaisquer informações
METHOD=POST		O processamento do formulário implica em ações que modificam informações (por exemplo, bases de dados ou assinatura de alguma lista)

Extensões do HTML 3.0 prevêem também:

- controle do domínio de um atributo SIZE;
- envio de arquivos (inclusive sons) para o servidor;
- botões SUBMIT/RESET modificados com imagem;
- suporte a menus SELECT gráficos.

3.4 - A LINGUAGEM HTML 3.0

O browser Arena está sendo desenvolvido no CERN para teste dos novos elementos da linguagem HTML 3.0. A maior parte dos elementos e seus atributos está em fase de desenvolvimento.

▫ Tabelas

<TABLE> permite a definição de um título e a diferenciação de cabeçalhos e células. As células podem conter texto, diversos parágrafos, listas, e cabeçalhos. Células adjacentes podem ser unidas, por exemplo, para definir um cabeçalho que abranja duas colunas.

A formatação de uma tabela pode conter os seguintes elementos:

- <TABLE BORDER>: o atributo BORDER faz com que o browser desenhe linhas englobando cada célula da tabela;
- <CAPTION>: define o título da tabela;
- <TH> (table header): precede texto de célula cabeçalho. O texto do cabeçalho é mostrado de modo enfatizado

  <TH ROWSPAN=n>: o atributo ROWSPAN faz com que n linhas de células de dados se fundam para formar uma célula de cabeçalho;

  <TH COLSPAN=n>: da mesma forma que ROWSPAN, COLSPAN une n colunas de células;
- <TD> (table data): precede texto de célula de dados

  <TD ALIGN=...> ou <TH ALIGN=...>: por default, o texto é centralizado dentro de cada célula, mas é possível formatar alinhamentos pelo atributo ALIGN, com valores LEFT, RIGHT, ou CENTER;
- <TR> (table row): usado para separar linhas da tabela.

Exemplo de formatação:

```
<table border=4>
<th colspan=2>Colunas 1 e 2
<tr><td>linha 1, coluna 1</td><td>linha 1, coluna 2</td></tr>
<tr><td>linha 2, coluna 1</td><td>linha 2, coluna 2</td></tr>
<th rowspan=3>3 linhas
<td>uma linha</td><tr><td>duas linhas</td></tr>
<tr><td>tres linhas</td></tr>
</table>
```

▫ Elementos Matemáticos

O elemento <MATH> indica uma seção de texto com formatação de elementos matemáticos.

Elemento:	Significado:	Exemplo:
<SUP>	sobrescritos (expoentes)	X_i = X_i
<SUB>	subscritos (índices, por exemplo)	X² = X^2
<BOX>	chaves e outros delimitadores	<box>a+b</box> = {a+b}
<OVER>	frações	1+X <over> Y = $\frac{1+X}{Y}$

Existem ainda outros elementos, tais como <SQRT> para raiz quadrada, <ARRAY> para vetores e matrizes, a maioria ainda em desenvolvimento.

Uma integral

$\int_a^b f(x)$ é escrita: `<math>∫<sub>a</sub><sup>b</sup>{f(x)}</math>`

▫ Figuras

O elemento <FIG> é similar a IMG, mas age como um parágrafo.

O atributo ALIGN pode valer LEFT (default), CENTER, RIGHT ou FLOAT, determinando onde a figura deve ser carregada. Se valer FLOAT, a figura pode ser colocada em outra posição mais conveniente (e possivelmente aumentada ou reduzida no processo). Um título pode ser definido com o elemento CAPTION e seguido por um texto que descreve a figura para os leitores que usam apenas browsers do tipo texto. O browser pode posicionar o título acima, abaixo, ou ao lado da figura.

▫ Inserção de observações

O elemento <NOTE> é usado para introduzir observações. O atributo CLASS diferencia classes de observações: notas, avisos, etc.

▫ Notas de rodapé

Mostradas normalmente em estilo *popup*, em uma janela sobreposta ao browser. Esses elementos FN (*footnote*) são referenciados por links com o elemento âncora.

▫ *Style Sheets*

Uma *"folha de estilo"* pode ser asssociada a um documento, para descrever como os elementos devem ser apresentados. Isso é feito usando o elemento LINK, por exemplo

```
<LINK rel=style href="estilo.dsssl">, ou
<style notation=dsssl-lite>Texto em dsssl-lite</style>.
```

Pode-se atribuir estilos a elementos de formatação específicos:

```
<h2 class=bigcaps>Texto com fonte maior que a normal</h2>
<p class=abstract>Par&aacute;grafo estilo resumo</p>
```

HTML 3.0 permite a definição de diversos elementos `BODY` em um documento, cada qual com seu próprio `background`, `style sheet`, etc. Os estilos declarados diretamente nos elementos (vide exemplos acima), têm prioridade sobre o estilo geral do documento.

3.5 - O SERVIDOR WWW

Um servidor WWW é normalmente usado como base para um servidor mais especifico para uma dada aplicação. O servidor que permite que um mundo de dados seja visto como parte do universo W3 é conhecido como gateway. (Muitos servidores poderiam ser considerados gateways, mas o termo implica uma conversão ou mapeamento entre mundos diferentes - ver seção seguinte).

Um httpd (*HyperText Transfer Protocol Daemon*), além de realizar funções complexas, pode permitir a inclusão da saída de comandos ou outros arquivos em seus documentos HTML.

Um *daemon* básico é essencialmente um servidor de arquivos, executando a função de recuperação de documentos, após o recebimento de uma mensagem do cliente. Essa função tem três parâmetros de entrada:

1- referência do documento a ser recuperado, enviada pelo cliente (através da referência hipertexto);
2- uma lista de palavras-chave separadas por sinais de "+" (vazia, exceto quando é pedida uma busca indexada);
3- o socket para o qual a resposta deve ser enviada.

3.5.1 - ARQUIVOS DO SERVIDOR

O *HTTPD* é basicamente acompanhado dos seguintes diretórios e arquivos:

Diretórios	Arquivos	Função
/httpd/conf	httpd.conf	arquivo principal de configuração do servidor. Define, em especial: • tipo do servidor (*inetd* ou *standalone*); • porta para o servidor standalone (default: port 80); • diretório raiz para os arquivos de configuração, erros e acessos.

	srm.conf	configura o espaço visível do servidor:
		• documento raiz do servidor. Ex.:
		`DocumentRoot /xavante/usr/httpd/docs`
		• documento default dos diretórios. Ex.:
		`DirectoryIndex index.html`
		• tipo de documento default para o caso do servidor não conseguir identificar o tipo pela extensão do nome-de-arquivo;
		• redirecionamento de URLs.

	access.conf	configura acesso global ao servidor, permitindo o controle de acesso a determinados diretórios de acordo com o domínio da máquina requisitante.
	mime.types	tipos de arquivo aceitos pelo servidor, de acordo com a extensão do nome-de-arquivo.

Diretórios	Arquivos	Função
/httpd/logs	accesss_log	mantém informações de acesso aos arquivos do servidor
	agent_log	informações sobre os browsers utilizados no acesso ao servidor
	referer_log	informações sobre a origem dos acessos ao servidor
	error_log	registro de erros ocorridos no acesso ao servidor
/httpd/support		mantém arquivos auxiliares do servidor
/httpd/src		mantém programas auxiliares do servidor
/httpd/cgi-src		mantém arquivos de interface para os programas de suporte
/httpd/cgi-bin		mantém arquivos de interface para scripts
/httpd/docs		mantém a árvore de documentos html, ícones, sons, animações e imagens disponíveis através do servidor

3.5.2 - COMMON GATEWAY INTERFACE

CGI (Common Gateway Interface) é o padrão pelo qual programas realizam interface com os servidores HTTP. Os programas CGI agem como *gateways* entre browsers e uma máquina servidora.

Por exemplo, um programa CGI pode ser projetado para lidar com um pedido de informação para um banco de dados, pegar os resultados e montar com eles um documento HTML, retornando esse documento para o browser. Tais programas podem ser escritos em diversas linguagens de programação, como *Perl*, ou na própria *Bourne shell*.

Conseqüentemente, os programas CGI são usados para processar informações obtidas via consultas `ISINDEX`, `FORM` ou `ISMAP`. Essas características dependem muito do servidor sendo usado.

3.5.2.1. USANDO `ISINDEX` PARA BUSCAS NO SERVIDOR

Aqui apresentamos um exemplo com o programa finger. O script também é chamado *finger*, e se encontra no diretório cgi-bin de um servidor. Ele é acessado pelo URL:

`http://www.servidor.br/cgi-bin/finger`

sendo o script *finger* com o seguinte conteúdo:

```
#!/bin/sh
echo Content-type: text/html
echo
if [ -x $FINGER ]; then
      if [ $# = 0 ]; then
            cat << EOM
<HTML><HEAD><TITLE>Finger Gateway</TITLE></HEAD><BODY>
<H1>Finger Gateway</H1>
<ISINDEX>
This is a gateway to "finger". Type a user@host combination in your browser's
search dialog.<p>
EOM
      else
            echo \<PRE\>
            $FINGER "$*"
            echo \</pre\>
      fi
else
      echo Cannot find finger on this system.
fi
cat << EOM
</BODY> </HTML>
EOM
```

Suponhamos que alguém acesse o `http://www.servidor.br/cgi-bin/finger`. Quando o script é acessado, ele sempre ecoa a linha `Content-type:text/html`. Essa linha é enviada de volta para o browser, de modo que o material "ecoado" se torna o novo documento. Em particular, esta linha diz ao browser para esperar um documento text/html.

A sentença `if` verifica se existem argumentos para o script. Esses argumentos são enviados através do URL, anexados após um ponto de interrogação. Se não houver argumentos, algumas definições em HMTL são ecoadas e é enviado o código `ISINDEX`. Isto diz ao browser que isso é uma busca.

O browser mostra, então, o documento recebido e pede uma cadeia de busca, apresentando um *template* (modelo) para ser preenchido. Quando a tecla *return* for pressionada, o browser acessará o mesmo URL anterior, mas dessa vez com a cadeia de busca anexada:

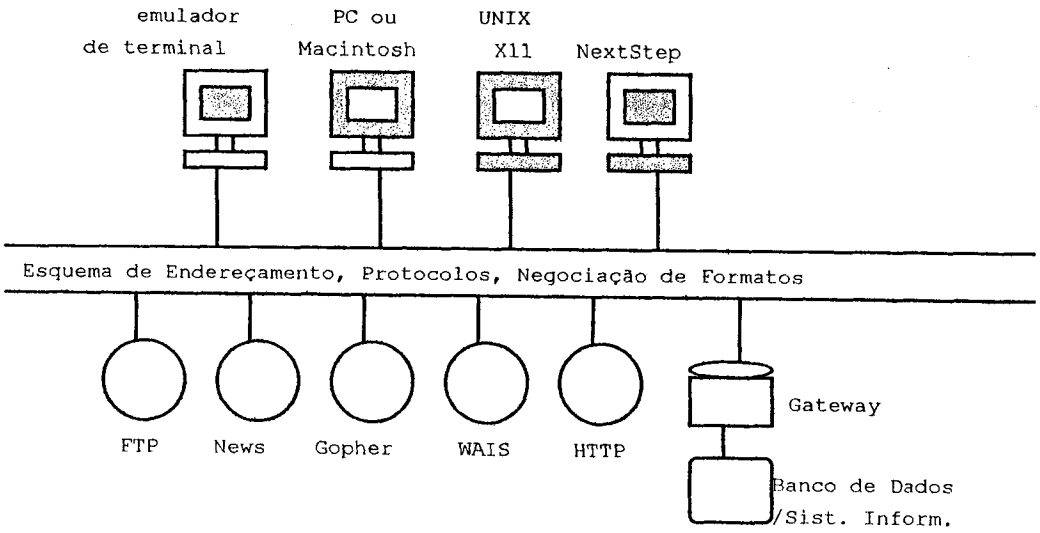
`http://www.servidor.br/cgi-bin/finger?cadeia_de_busca`

Dessa forma, o segundo ramo do `if` é acessado, ecoando outros cabeçalhos indicando a busca que está sendo realizada, e roda o programa de busca. Por default o resultado do finger é apresentado, e os resultados da busca são enviados para o browser. Não se inclui `ISINDEX` nesse ramo do `if`, porque ele já fornece os resultados da busca, apresentados no documento final.

3.5.3 - WWW E OUTROS SISTEMAS

Dois outros sistemas, WAIS e Gopher, compartilham da arquitetura cliente-servidor de WWW.

O protocolo ***WAIS*** é influenciado pelo protocolo z39.50, projetado para catálogos de bilbioteca em rede. Ele permite busca baseada em texto, e recuperação seguindo uma busca. Os índices a serem procurados são encontrados por uma procura em um *índice master*. Esta busca em dois estágios provou ser suficientemente poderosa para cobrir o mundo de dados atual de WAIS. Entretanto, não há ferramentas navegavionais para permitir ao usuário verificar os recursos disponíveis, ou para guiá-lo através dos dados: o usuário cai "de repente" em um local de dados no mundo de informações, e fica sem contexto.



A arquitetura cliente-servidor da World-Wide Web [Berners-Lee_94]

Gopher apresenta um mecanismo de busca livre de texto, mas se utiliza principalmente de menus. Um menu é uma lista de títulos, dos quais o usuário pode escolher um. Enquanto o espaço gopher é de fato uma teia (web) contendo diversos loops, o sistema de menus dá ao usuário a impressão de uma árvore. O servidor *Veronica* fornece um índice master para o espaço gopher.

O modelo de dados WWW é similar ao modelo gopher, exceto que os menus são generalizados para documentos hipertexto. Em ambos os casos, simples servidores de arquivo geram os menus ou hipertexto diretamente da estrutura de arquivos de um servidor. O modelo hipertexto de WWW dá ao programa maior poder para comunicar as opções disponíveis ao usuário incluindo, por exemplo, cabeçalhos e diversas formas de estrutura de listas no hipertexto.

Todos os três sistemas permitem que se apresente gráficos, som e vídeo, embora o sistema WAIS tenha que ter texto associado aos arquivos gráficos, devido a seu acesso ser somente por busca de texto.

3.5.4 - HYPER-G: ALÉM DA WWW

A WWW apresenta diversas limitações que se tornam aparentes quando se considera tarefas mais complexas; por exemplo, o servidor é incapaz de realizar buscas completas. Não é possível realizar buscas além das fronteiras dos servidores WWW; faltam características de autorização de acesso, entre outros. Tendo pavimentado o caminho para uma utilização mais efetiva da Internet, a WWW serviu de base para trabalhos que visam implementar uma "segunda geração" de sistemas hipermídia.

Na WWW, os links são unidirecionais. A bidirecionalidade é essencial para descobrir quais documentos estão apontando para o documento atual, informação crítica para desenvolvimento de browsers gráficos e para atualização de links.

Considerando os conceitos que faltam na WWW, examinaram-se idéias, estruturas e caraterísticas básicas de grandes servidores distribuídos, dando surgimento ao Hyper-G, projeto da Universidade de Tecnologia de Graz (Áustria) em 1993.

A diferença principal entre servidores WWW e Hyper-G é que o Hyper-G provê muito mais funcionalidade integrada, e portanto uniforme, implementada sobre a WWW. Outras características:

• o conceito de conjuntos (*clusters*) e coleções (*collections*) de documentos como elementos fundamentais;

• os servidores Hyper-G trabalham sobre uma plataforma realmente distribuída: um usuário pode acionar diversos servidores Hyper-G de modo que a união de todos os bancos de dados envolvidos aparece como se fosse um único banco de dados;

• é possível lidar com contas de usuários: quem pode acessar o quê, onde, e por quanto tempo; isso se torna útil para pesquisas, marketing, cobranças, etc;

• o autor é capaz de especificar a vida útil de documentos no momento da criação;

• a característica de mapeamento global é uma nova abordagem para navegação por informações, além da visão de caminho único da WWW. Em Hyper-G todos os links *para* ou *de* algum documento são conhecidos (sem que se veja o conteúdo dos documentos), e assim é possível seguir links em qualquer direção, através de qualquer número de servidores.

Conceitos importantes têm sido considerados no projeto de Hyper-G:

•grandes sistemas hipermídia devem ter uma estrutura: um grafo plano e simples, sem significado semântico para links, não será suficiente para sistemas grandes;

•atividades em grandes sistemas multimídia em rede precisam ser restritos a escopos definidos pelos usuários;

•as atividades consideradas centrais (tais como busca) devem ser integradas ao sistema básico para evitar a proliferação de características conflitantes com a implementação do sistema.

A construção de Hyper-G é considerada a primeira experiência em sistemas hipermídia de "segunda geração" a se tornar amplamente disponível. A Universidade de Tecnologia de Graz vem trabalhando em coopereção com outras universidades para o desenvolvimento do sistema, estando ainda aberta a novos colaboradores para projetos de clientes e servidores.

4 - O SERVIDOR WWW DO ICMSC

O servidor NCSA instalado no ICMSC conta com uma árvore de mais de 30 diretórios, contendo documentos hipertexto, imagens e fotos, todos disponíveis . Grande parte das informações se encontra tanto em inglês como em português; existem listas com links para servidores FTP, Gopher e HTTP no Brasil, e servidores HTTP de diversos países.

A árvore de diretórios do servidor tem sua raiz atual no diretório de rede */xavante/usr/httpd/docs/*. Dessa maneira, quando um browser pedir a recuperação de um documento do servidor no endereço: `http://www.icmsc.sc.usp.br/`, ele estará recuperando um arquivo do diretório `/xavante/usr/httpd/docs/`.

Os documentos disponíveis pelo servidor apresentam informações variadas: apresentações pessoais de alunos e professores, grupos de trabalho, trabalhos de pós-graduação, publicações de docentes, tutoriais, informações retiradas de listas de discussão do ICMSC, e recursos didáticos para cursos de graduação.

4.1 - ÁRVORE DO SERVIDOR

Os documentos disponíveis no servidor estão separados em dois ramos, sendo o principal (a partir da raiz) com informações em inglês. A maior parte dos documentos em português se encontra sob o diretório `Portugues`; nem todas as informações disponíveis têm correspondente nas duas línguas, sendo a parte predominante em português.

A organização do servidor segue a seguinte estrutura:

Diretório	Informações
Campus	dados sobre o Campus da USP em São Carlos; resumos sobre a história do Campus e sobre a USP; links para outros servidores WWW no Campus
Grupos	apresentação dos grupos de pesquisa do ICMSC
ICMSC	histórico, cursos, espaço físico, bolsas de pesquisa
News	links para jornais e revistas relacionadas a Matemática, Computação e Estatística
Phd	informações sobre o curso de Doutorado
Portugues	espelha informações do ramo principal, com maior quantidade de dados
ProtocolStuff	informações sobre o protocolo HTTP
Sao_Carlos	apresentação da cidade de São Carlos; história, características geográficas e econômicas, turismo, rota rodoviária São Paulo - São Carlos
cgi	informações sobre *Common Gateway Interface*
eventos	eventos no ICMSC: palestras, simpósios
fotos	fotos e fundos de página
icones	ícones diversos
links	links para servidores brasileiros, acadêmicos, e de outros países
manuals	manuais e tutoriais sobre a utilização da Internet, linguagem C, HTML, WWW
people	informações sobre os reponsáveis pela manutenção do servidor (os *webmasters*); lista de homepages disponíveis pelo servidor
staff	apresentação do corpo docente; links para homepages
tutorials	tutoriais sobre o servidor NCSA
util	lista telefônica do ICMSC

Todas as informações estão em fase de atualização contínua e extensão. A partir da árvore de documentos também partem links-diretório para áreas particulares de grupos de pesquisa - desse modo, essas informações são colocadas no servidor sem intervenção direta dos webmasters.

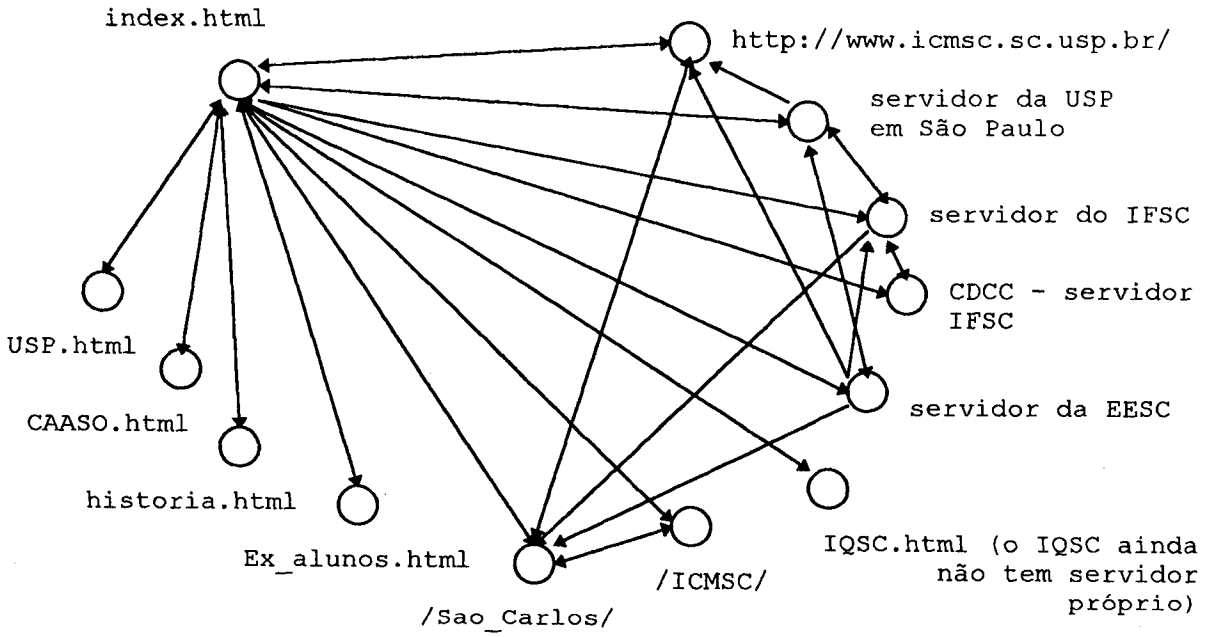
4.2 - GRAFOS DE NAVEGAÇÃO

Tendo em vista proporcionar um caminho básico de acesso a todas as informações contidas no servidor, os diretórios de documentos foram assentados sobre um *grafo de navegação*.

Embora todo servidor possua uma página principal de apresentação, qualquer página é virtual candidata a ponto de entrada aos seus documentos - através de bookmarks, hotlists, buscas na rede, etc. O grafo de navegação não é um caminho que todos os leitores seguem, obrigatoriamente: apenas sugere um caminho de navegação.

Existem três documentos principais que servem de porta de entrada ao nosso servidor: a página de apresentação, a do Campus, e a da cidade de São Carlos. Os links "orientam" a navegação entre essas páginas de modo a oferecer razoável quantidade de informações em poucas seleções via mouse.

Tomando como exemplo a sub-árvore de documentos do Campus, temos o seguinte grafo:



Grafo de navegação a partir de /Campus/

As linhas bidirecionais não indicam links bidirecionais (que não existem em HTML), apenas representam que os documentos fazem referência um ao outro. No caso dos servidores, essa referência não é direta (a partir do documento raiz), e sim através de documentos imediatamente abaixo da raiz. Um tipo de reestruturação desse grafo implica a introdução de mais links, permitindo a navegação direta entre os documentos que estão num mesmo nível hierárquico: esses links evitam que se tenha de voltar sempre a uma página principal para obter acesso a outras informações relacionadas.

4.2.1 - DISTRIBUIÇÃO DOS DOCUMENTOS

A divisão dos textos em diversos arquivos é baseada em conceitos de bancos de dados distribuídos. Mudanças nessa estrutura são feitas também de acordo com as estatísticas de acesso e com as referências recebidas "de fora" (links de outros servidores para documentos de nosso servidor). Conforme o tipo de informação, é prático deixar um texto mais extenso e completo para ser recuperado de uma só vez; em outros casos, é melhor dar ao leitor a opção de escolher apenas os trechos de informação que mais lhe interessarem.

Conforme o tipo de informação, é prático deixar um texto mais extenso e completo para ser recuperado de uma só vez; em outros casos, é melhor dar ao leitor a opção de escolher apenas os trechos de informação que mais lhe interessarem.

Atualmente, a página do Campus funciona como um índice para os servidores do Campus e arquivos com informações relacionadas (Centro Acadêmico, história, etc.). No início, porém, essa página continha muitas informações juntas e links somente para arquivos sobre o ICMSC. Como outros servidores WWW foram sendo instalados no Campus, essa página foi totalmente modificada, por não ter que se ocupar com os dados específicos das outras unidades.

A página do ICMSC passou por transformação semelhante: antes composta por textos, imagens e links, atualmente apresenta uma foto, o endereço físico, e links para informações específicas: histórico, departamentos, áreas de pesquisa, biblioteca, Campus e cidade de São Carlos. Essa estrutura visa uma apresentação menos carregada, e maiores possibilidades de navegação pelas informações.

Já a página de São Carlos passou por três etapas diferentes: na primeira, todos os dados referentes à cidade eram recuperados de uma só vez; na segunda, a página de apresentação era somente um índice de links para tópicos separados em diversos arquivos; na etapa atual, essa página contém links para tópicos principais (geografia, história, etc.) seguidos de resumos do conteúdo apontado pelos links. Essa mudança procura atrair o leitor para as outras páginas, uma vez que a "página-índice" não chamava a atenção do navegador "impaciente" (de acordo com o verificado nos logs de acesso).

4.3 - DIRETRIZES PARA AS PÁGINAS HIPERTEXTO

Existem algumas normas a seguir para escrever um documento WWW:

- as informações dadas devem ser confiáveis;
- todo texto deve ser assinado, isto é, o autor do texto deve ser identificado, com nome e endereço (email ou físico) para contato;
- a origem das informações deve ser especificada: livros, jornais, revistas, artigos, etc.
- o status dos documentos e/ou do servidor deve ser especificado: se está "em construção", data da última atualização, tudo que situe a condição atual do servidor ou de um conjunto de documentos para que o usuário não perca tempo pedindo dados que ainda não estejam disponíveis (é aconselhável não deixar links para páginas ainda não instaladas);
- uso da língua inglesa - a "língua oficial da Internet" deve ser utilizada em pelo menos uma versão dos documentos de um servidor;
- qualquer citação à WWW pode ser um link para o URL `http://www.w3.org/`, que é o ponto de entrada para se obter informações sobre o projeto WWW, e para uma visão geral da Web através de um catálogo dos bancos de dados ordenados por assunto (pela *Virtual Library*).

Repetimos aqui a preocupação com os diversos browsers em uso: uma página que utiliza recursos específicos de determinado browser poderá parecer muito mal feita para usuários de outros dispositivos. Além disso, os leitores podem modificar em seus browsers a fonte do texto, e desativar os gráficos das páginas, alterando o resultado final esperado pelo autor. A maior preocupação deve ser a estrutura das páginas, garantindo uma boa visualização em qualquer browser. A utilização de gráficos deve ser administrada tendo-se em vista que uma página muito "pesada" (com muitos gráficos grandes) levará muito tempo para ser transmitida, e o leitor que utiliza conexões lentas acabará desistindo de acessá-la.

4.3.1 - EDIÇÃO

Um arquivo em HTML é um arquivo ASCII comum. Pode ser editado, por exemplo, em vi ou emacs (que já tem uma versão para editar em html). Para evitar a sobrecarga de digitação - devido às

etiquetas da linguagem, foram desenvolvidos editores HTML, que inserem esses elementos com menor esforço para o autor. Para ambiente Windows, existem os editores HoTMetaL, HTML Writer, HTMLed, e HTMLAssistant (entre os mais conhecidos). Para Macintosh e UNIX se encontram diversos outros editores específicos. Para UNIX, foi testado o HoTMetaL, que verifica a sintaxe do documento em tempo de edição, segundo regras SGML para HTML e "HTML+"; permite que se escreva documentos estritamente corretos, porém ocupa muita memória e tem baixa performance. A maioria desses editores é adquirido via ftp.

A versão 6.0a do Word já vem com um módulo de edição para HTML. Também há diversos "tradutores" que convertem arquivos do Word for Windows, e de outros formatadores de texto, para HTML.

4.4 - HOMEPAGES

Para criar suas próprias páginas, a primeira coisa a fazer é criar um diretório ***WWW*** (com letras maiúsculas) em sua área na rede. A partir do momento da criação desse diretório, o URL:

`http://www.icmsc.sc.usp.br/~seulogin/`

passará a ser acessível pelo servidor. Nesse diretório deve haver um arquivo ***index.html*** (já explicado no capítulo anterior), que servirá de página de apresentação e índice das informações que estarão disponíveis em sua área.

Isso é possível devido à diretiva `UserDir WWW` presente no arquivo de configuração `srm.conf`. Depois de ter suas páginas prontas (com os links corretos), é preciso apenas contactar um dos Webmasters para que sua página seja referenciada através de uma das páginas principais do servidor - página de homepages, por exemplo.

O tutorial HTML on-line encontra-se provisoriamente no endereço

`http://www.icmsc.sc.usp.br/~masc/notasHTML/`

com passos básicos de aprendizado, e ítens avançados em implementação.

REFERÊNCIAS NA WWW

Assunto/Site	URL
CERN Server User Guide	http://www.w3.org/hypertext/WWW/Daemon/User_3.0/
How to write HTML files	http://kcgl1.eng.ohio-state.edu/www/doc/htmldoc.html
HTML (Internet Draft)	http://www.w3.org/hypertext/WWW/MarkUp/HTML.html
HTTP: A protocol for networked information	http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html
Hyper-G Papers(collection)	http://hmul.hmu.auckland.ac.nz:80/C912167A/CHGpapers
IETF Working Groups (mirror)	http://www.ics.uci.edu/pub/ietf/
HotJava	http://www.java.com/
Mosaic Home	http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
NCSA CGI	http://hoohoo.ncsa.uiuc.edu/cgi/overview.html
NCSA HTTPD	http://hoohoo.ncsa.uiuc.edu/
Netscape	http://www.netscape.com/
PC Magazine on the Web	http://www.ziff.com/~pcmag/
The HTML Sourcebook	http://www.utirc.utoronto.ca/HTMLdocs/NewHTML/htmlindex.html
The World-Wide Web Virtual Library	http://www10.w3.org/hypertext/DataSources/bySubject/
W3 manuals available in printed form	http://www.w3.org/hypertext/WWW/Paper/PaperManuals.html
World-Wide Web Consortium (W3C)	http://www.w3.org/
World-Wide Web Primer	http://www.vuw.ac.nz/who/Nathan.Torkington/ideas/www-primer.html

REFERÊNCIAS BIBLIOGRÁFICAS

•[Avison_92] Avison, David E. - **Information Systems Development: A Database Approach** - Blackwell Scientific Publications, 1992 - 2nd Edition

•[Azevedo_93] Azevedo, Carlos E. Mendez - **A Internet e seu uso - in: Redes de Computadoras** - VI Escuela Brasileño Argentina de Informatica - julho 1993

•[Barreira_94] Barreira, Wagner - **Internet S.A.** - SUPERINTERESSANTE - ano 8 n°11, novembro 1994, pp 60-63

•[Berners-Lee_94] Berners-Lee, Tim - **Universal Resource Identifiers in WWW** - RFC1630 - CERN - junho de 1994

•[Berners-Lee_94] Berners-Lee, Tim; Cailliau, Robert; Luotonen, Ari; Nielsen, Henrik Frystyk; Secret, Arthur - **The World-Wide Web** - Communications of the ACM - vol.37, n°8, 1994, pp 76-82

•[Berners-Lee_94] Berners-Lee, Tim; Masinter, Larry; McCahill, M. - **Uniform Resource Locators (URL)** - RFC1738 - CERN, Xerox Corporation, University of Minnesota - dezembro de 1994

•[Berners-Lee_95] Berners-Lee, Tim; Connoly, Daniel W. - **Hipertext Markup Language - 2.0** - Internet draft - MIT Laboratory for Computer Science, W3 Consortium - 8 de agosto de 1995

•[Berners-Lee_95] Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk - **Hipertext Transfer Protocol - HTTP/1.0** - Internet draft - MIT Laboratory for Computer Science, University of California, CERN - março de 1995

•[Ceri_84] Ceri, Stefano; Pelagatti, Giuseppe - **Distributed Databases: Principles and Systems** - McGraw-Hill, 1984

•[Comer_91] Comer, Douglas E. - **Internetworking with TCP/IP - vol 1 - Principles, Protocols and Architecture** - Prentice Hall, 1991 - 2ª. ed.

•[Connoly_95] Connoly, Dan - **Character Set Considered Harmful** - Internet draft - MIT/W3C - 2 de maio de 1995

•[Couceiro_84] Couceiro, Luiz Antonio C. C.; Barrenecha, Hugo F. S. - **Sistemas de Gerência de Banco de Dados Distribuídos** - LTC, 1984

•[Coulouris_88] Coulouris, George F.; Dollimore, Jean - **Distributed Systems: Concepts and Design** - Addison-Wesley, 1988

•[EARN_93] EARN Association-**Guide to Network Resource Tools** -Sept 15, 1993, n°2.0

•[Fielding_95] Fielding, R. J. - **Relative Uniform Resource Locators** - RFC1808 - University of California - junho de 1995

•[Frentzen_95] Frentzen, Jeff; Garfinkel, Simson; Seltzer, Larry; Sullivan, Eamonn - **Montando um Negócio na Internet** - Windows Computing - vol.2, n°6, pp 38-79

•[Furtado_88] Furtado, Antonio Luz; dos Santos, Clésio Saraiva - **Organização de Bancos de Dados** - Editora Campus, 1988 - 7ª. edição

•[FSP_94] Folha de S. Paulo - **A Superinfovia do Futuro** - diversos artigos - caderno Mais! - pp 6-4 - 6-8 - 17 de julho de 1994

•[Helal_94] Helal, Abdelsalam; Elmasri, Ramez - **Distributed & Object-Oriented Database Systems** - minicurso, IX Simpósio Brasileiro de Banco de Dados, São Carlos-SP, setembro 1994

•[Krishnamurthy_89] Krishnamurthy, E. V. - **Parallel Processing: Principles and Practice** - Addison-Wesley, 1989

•[Lages_86] Lages, Newton A. de Castilho; Nogueira, José M. S. - **Introdução aos Sistemas Distribuídos** - Editora da UNICAMP/Editora Papirus, 1986

•[Masinter_94] Masinter, Larry; Sollins, Karen - **Functional Requirements for Uniform Resource Names** - RFC1737 - Xerox Palo Alto Research Center, MIT Laboratory for Computer Science - dezembro de 1994

•[Raggett_95] Raggett, Dave - **HyperText Markup Language Specification Version 3.0** - Internet draft - W3 Consortium - 28 de março de 1995

•[Ramalho_94] Ramalho, Luciano - **Fios da Teia Global** - Informática Exame - Ano 9 - n°105 - dezembro 1994

•[Rose_91] Rose, Marshall T. - **The Simple Book: An Introduction to Management of TCP/IP-based internets** - Prentice Hall, 1991

•[Schwartz_88] Schwartz, Mischa - **Telecommunication Networks: Protocols, Modeling and Analysis** - Addison-Wesley, 1988

•[Shepherd_90] Shepherd, Michael A.; Watters, C. R.; Cai, Yao - **Transient Hypergraphs for Citation Networks** - Information Processing & Management vol.26, n°3, 1990, pp 395-412

•[Traina_94] Traina Jr., Caetano; Ferreira, João Eduardo - **Adding Conceptual Constructs to Support Distribution in an Object Oriented Model** - Anais do IX Simpósio Brasileiro de Banco de Dados, São Carlos-SP,setembro 1994

•[Yergeau_95] Yergeau, Franois; Nicol, Gavin Thomas; Adams, Glenn; Duerst, Martin J. - **Internationalization of the Hypertext Markup Language** - Internet draft - Alis Technology, Electronic Book Technologies, Universidade de Zurich - 15 de agosto de 1995