Sistema Operacional Unix

André F. Silveira, Gustavo H. P. Kwaczynski, Lucas E. Dematte, Rodrigo Miss

Universidade Tecnológica Federal do Paraná (UTFPR)

Guarapuava – Pr – Brasil

Abstract*. Unix is a free operating system platform with multi-user and multitasking. Due to its ease of use and its great performance, can be used on a personal computer or even server files or applications from its advances and its various offshoots born modulated structure and new systems.*

The whole HTML, CSS and JavaScript work together to terms available sites with good looks and great performance. A new horizon with numerous functions is presented to web programmers with the arrival of HTML5 and CSS3.

Key-words: *UNIX, system, operational, programmers, looks.*

Resumo. *Unix é um sistema Operacional livre com plataforma Multiusuário e Multitarefa. Devido à facilidade de sua utilização e seu grande desempenho, pode ser utilizado em um computador pessoal ou até mesmo em servidor de arquivos ou aplicações, a partir de seus avanços e de sua estrutura modulada nasceram diversas ramificações e novos sistemas.*

O conjunto HTML, CSS e JavaScript trabalham em conjunto para termos disponíveis sites com boa aparência e ótimo desempenho. Um novo horizonte com inúmeras funções é apresentado aos programadores web com chegada do HTML5 e CSS3.

Palavras-chave: *UNIX, sistema, operacional, desenvolvedores, aparência .*

1. Introdução

O UNIX foi o precursor de diversos sistemas operacionais tais como o Linux, Minix, Solaris, Mac OS X, e diversos outros pequenos sistemas como por exemplo sistemas automotivos e pequenas máquinas. Isto aconteceu devido sua facilidade de implementação, possuir código aberto e bibliotecas o que permite uma maior facilidade em reescrevê-lo ou moldá-lo.

Foi desenvolvido com o objetivo inicial de proporcionar um ambiente onde os programadores pudessem criar novos programas, mas devido a sua grande popularidade logo já estava em uso por universidades e empresas. Com a rápida disseminação logo havia várias implementações diferentes, ou seja, várias adaptações do sistema.

Neste artigo será abordada a história, a estrutura e a organização do sistema operacional UNIX, bem como um resumo de seu nascimento, empresas envolvidas, sua disseminação e contexto no cenário mundial.

Cada vez mais estamos dependentes de utilizar serviços disponíveis na internet. E com isso, cada vez mais sites surgem na rede. E para que os sites existam, alguns elementos fundamentais são necessários, e três deles são: HTML, CSS, JavaScript. São linguagens de programação que trabalham juntas para o funcionamento dos sites.

Enquanto o HTML faz a marcação para formatação das páginas web, o CSS trabalha otimizando o visual e o JavaScript torna dinâmico o que era estático.

Neste artigo serão abordados conceitos e informações das linguagens HTML, JavaScript e CSS.

2. Unix

Em 1960 as organizações MIT, Bell Labs e General Eletric se uniram para a criação de um novo sistema operacional que viesse a substituir os sistemas Batch. Ao decorrer do projeto a empresa Bell Labs se retirou, porém dois de seus pesquisadores Ken Thomposon e Dennis Ritchie continuam o projeto e vieram a desenvolver o sistema UNICS, que no futuro se tornaria o sistema UNIX. [Machado & Maia, 2008]

Após o sistema já estar operacional, a empresa Bell Labs que era uma subsidiaria da AT&T não podia comercializar o mais novo sistema devido às leis de Antimonopólio. Em 1974 os criadores do UNIX lançaram um documento descrevendo o novo sistema, o que despertou a curiosidade das Universidades em fazer uso deste moderno Sistema operacional, sendo útil para a AT&T já que as universidades poderiam licenciar o UNIX. Assim a Universidade de Berkeley na Califórnia recebeu cópias do sistema bem como seu código fonte, realizaram diversas modificações criando novas releases. Grandes melhorias ocorreram com as modificações da universidade entre elas a criação do Protocolo TCP/IP, memória virtual e Shell.

Com um sistema mais estável e melhorias constantes, as empresas abriram as suas portas e inseriram o UNIX em seus produtos. Entre as grandes empresas estão HP e IBM. Em 1982 a AT&T é liberada para comercializar o sistema, então a AT&T e Universidade de Berkeley lançavam atualizações diferentes e nunca houve unificação entre ambas, apesar de usarem a mesma base para o sistema. A AT&T continuou sua comercialização com o sucesso da release (SVR4), enquanto Berkeley criou a *Open Software Foundation*, disponibilizando seu sistema de forma livre.

2.1. Características

Existem vários sistemas operacionais, porém muitas vezes o fator que determina seu sucesso ou fracasso, são suas peculiaridades. No Unix, destacam-se duas dessas características: é um sistema multitarefa e multiusuário.

2.1.1. Sistema operacional multitarefa

Sistemas operacionais multitarefa são sistemas capazes de executar um ou mais processos simultaneamente, enquanto sistemas monotarefa permitem executar apenas um processo por vez, sendo executado sequencialmente de forma tão rápida causando a falsa ilusão de que estão sendo executados simultaneamente [Machado & Maia 2008].

O UNIX é um sistema operacional multitarefa com preempção, o que significa que um intervalo de tempo chamado **quantum** é que define o tempo que cada processo possui para utilizar o processador e executar suas tarefas. Após esgotar o quantum, o UNIX suspende a execução do processo e salva as informações necessárias para posterior execução (contexto do processo), e coloca em execução o próximo processo da fila de espera. [Machado & Maia, 2008].

2.1.2. Sistema operacional multiusuário

O UNIX é um sistema operacional multiusuário, pois permite que dois ou mais usuários utilizem o mesmo computador simultaneamente para executar diferentes aplicações concorrentemente e independentemente, geralmente através de terminais conectados ao determinado computador. [Machado & Maia, 2008]

O Unix possui dois tipos de usuários, o usuário root ou superusuário que possui total permissão sobre o sistema, e os usuários comuns que possuem permissões limitadas e configuráveis.

2.2. Estrutura

Quase em totalidade o código fonte do sistema Unix foi desenvolvido em linguagem C, sendo a minoria restante escrita em *assembly*, favorecendo o sistema quando se deseja fazer a portabilidade para outras plataformas de hardware [Srirengan, 1998].

A estrutura do UNIX é baseada em camadas (camada usuário, utilitários, biblioteca padrão, *sytem calls* ou chamada de sistema e camada de hardware),e o modo de acesso ao sistema se dá por meio de dois níveis, usuário e kernel.[Machado & Maia, 2008]

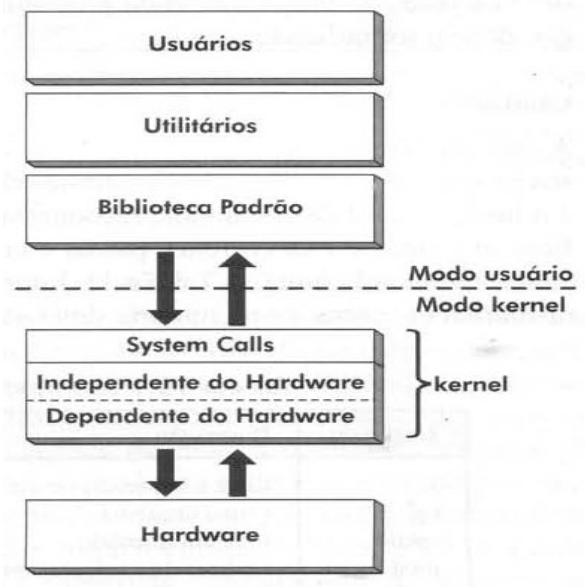

Figura 1. Estrutura do Unix [Machado & Maia, 2008]

2.3. Kernel

O kernel é uma fatia do sistema que é responsável por controlar o hardware e fornecer as chamadas de sistema para que as aplicações do usuário tenham acesso às funções do sistema operacional.

Possui uma parte dependente do hardware e outra independente. A parte dependente está atrelada as rotinas de tratamento de interrupções e exceções*, drivers* e tratamento de sinais. Portanto, deve ser reescrita quando estiver sendo portado o sistema UNIX para outra plataforma de hardware. A parte independente não possui nenhum vínculo com a plataforma que está portada, e é responsável pelo tratamento das *systemcalls*, gerência de processos, gerência de memória, escalonamento, *pipe*s, paginação, swapping e sistemas de arquivos.[Machado & Maia, 2008]

O kernel do UNIX, se comparado com os demais sistemas operacionais, é dotado de um conjunto pequeno de *system calls*, porém, a partir delas podem ser desenvolvidas funções de enorme complexidade. Isso foi uma estratégia adotada quando o UNIX foi criado, pois novos utilitários podem ser facilmente integrados ao sistema, sem que o kernel sofra quaisquer alterações.

2.4. Biblioteca Padrão

Para cada rotina do sistema, existe uma função na biblioteca padrão do Unix, que permite ocultar os detalhes do modo de acesso usuário/kernel ao sistema. A biblioteca fornece uma interface entre os programas e o sistema operacional, fazendo com que as chamadas de sistemas sejam invocadas.[Machado & Maia, 2008]

2.5. Utilitários

A camada de mais alto nível é a interface com o usuário, e é nela que estão implementados vários softwares, como editores de texto, navegadores de internet, compiladores e o shell.

O shell é um interpretador de comandos, responsável por ler e verificar a sintaxe dos comandos introduzidos pelo usuário e finalmente passar o controle para outros programas que realizaram a função solicitada. O Unix possui três interpretadores de comandos populares, a saber, Bourne Shell (sh) que foi o primeiro shell disponível e está presente em todas as versões do Unix, o C Shell (csh) e o Korn Shell.[Machado & Maia, 2008]

Além das várias interfaces em modo texto, também existem várias interfaces gráficas, como por exemplo a X Windows System.

2.6. Sistema de Arquivos

A organização dos diretórios é chamada de *filesystem, e a* sua estrutura lembra uma árvore, um organograma, chamado de *root directory e se inicia pela "/".* Programas executáveis são alocados na pasta "/bin", arquivos provenientes de dispositivos de entrada e saída "/dev", bibliotecas "/lib", e usuários "/usr". Como no UNIX a maioria das funções utilizam arquivos, é necessário um sistema de propriedade e proteção que garantam a integridade dos arquivos. Cada arquivo é somente um conjunto de bytes, pois ele não diferencia um arquivo de texto de um executável. Assim a função do sistema é simplesmente gerar o acesso a estes arquivos, e a interpretação dos mesmos fica por parte da aplicação.

A localização de um arquivo no diretório é indicada através de um *pathname* que pode ser relativo ou absoluto. *Pathname* relativo é a partir de onde o usuário se encontra e absoluto é a localização completa partindo do "/". Uma curiosidade do sistema de arquivos do UNIX é que ele mantém a sua estrutura lógica mesmo sendo adicionados outros discos físicos, ou seja, independente de quantos discos o computador possua, para o usuário é como se existisse somente um.

As informações dos arquivos ficam armazenadas nos *i-nodes*, isto é, existe um bloco onde os *i-nodes* ficam, e eles armazenam a data de criação, modificação, inserção entre outras informações do arquivo. Cada*i-node* possui 64 bytes para armazenar estas informações. Assim quando um arquivo possui uma enorme quantidade de informações, extrapolando o tamanho dos *i-nodes*, este é dividido entre quantos forem necessários, e ao fim de cada*i-node* existe uma espécie de ponteiro para o próximo, formando uma estrutura encadeada.

2.7. Gerências

Gerencia como o próprio nome diz, são responsáveis por manter a ordem e o funcionamento de setores de um S.O cada qual com um objetivo especifico, mas em geral de manter o sistema em perfeito funcionamento e de forma ágil, realizar trocas sem que o usuário sinta qualquer impacto de seus esforços, deixando assim o sistema totalmente funcional e a postos para atender as necessidades de seus usuários.

2.7.1. Gerência do Processador

O sistema UNIX utiliza dois tipos de escalonamento, circular com prioridades e prioridades. [Machado & Maia, 2008]

No UNIX, os processos podem ter prioridades entre 0 e 127 (quanto menor o valor, maior a prioridade). Processos do modo usuário geralmente tem prioridades entre 50 e 127, e os do modo kernel entre 0 e 49. Quando estão no estado de pronto, os processos aguardam em filas para serem escalonados, onde cada fila está associada a uma propriedade.

Depois de escalonado, o processo pode ficar no processador por até 100 milissegundos. Quando o seu quantum termina, o processo retorna para a fila associada à sua prioridade. A prioridade dos processos no estado de pronto é recalculada periodicamente.

2.7.2. Gerência de Memória

As primeiras versões do UNIX utilizavam um gerenciamento de memória do tipo swapping. Após a versão 3BSD o sistema adotado para gerenciar a memória foi paginação por demanda e atualmente é utilizado paginação com swapping. [Toscani, 2010]

O espaço de endereçamento da memória do UNIX é dividido em três partes:

- **Texto:** Parte onde ficam armazenados os códigos executáveis dos programas. É uma área protegida contra gravação. É estático e pode ser compartilhado utilizando um esquema de memória compartilhada.

- **Dados:** Diferente da parte de texto, a área de Dados é o local onde se encontram as variáveis de um programa. É uma área dinâmica e permite alterações e gravações constantes.

- **Pilha:** "Armazena informações do controle de ambiente dos processos e procedimentos, ela cresce dinamicamente do endereço virtual mais alto para o mais baixo".[Toscani, 2010]

A paginação por demanda no UNIX ocorre somente para páginas que são referenciadas em suas execuções. O sistema gerencia duas listas: uma contendo as páginas livres, ou seja, que estão disponíveis, e outra que contém as paginas já em uso. Quando uma página é referenciada e não se encontra na memória principal ocorre o *pagefault*. Esta verificação ocorre através do bit de validade, e quando isto ocorre, a gerência de memória retira uma página que está na lista de livres e carrega ela na memória e ela passa a constar na lista de páginas em execução.

O Kernel é responsável por implementar a paginação, enquanto um segundo processo chamado *Daemon* é responsável por verificar a lista de páginas livres a cada determinado período de tempo. Caso esta lista esteja quase vazia, ele é responsável por realizar uma busca nas páginas que estão carregadas e verificar quais estão disponíveis para voltarem para a lista de livres.

As páginas de texto podem ser transferidas sem qualquer verificação, pois são automaticamente recarregadas quando o arquivo executável é ativado. Já as páginas de dados devem ser verificadas se ocorreram modificações, através do bit de modificação. Caso o bit informe que ocorreram modificações, estas devem ser salvas em disco antes de serem transportadas para a lista de páginas livres.

A substituição de páginas ocorre por um algoritmo FIFO Circular, porém com dois ponteiros em vez de um, como é o comum. O primeiro ponteiro, que sempre fica a frente do segundo, é responsável por desligar o bit de referências das páginas, enquanto o segundo ponteiro, que fica a certa distância do primeiro, é responsável por verificar o estado da página. Caso a página não tenha sido referenciada desde que o primeiroponteiro passou por ela, ela é selecionada para passar a lista de páginas livres, é o que mostra a figura 2.

Se por acaso o sistema não conseguir manter um número mínimo de páginas livres a área de swapping é ativada e ponteiros semelhantes aos anteriores verificam os processos que estão a mais tempo inativos, e então os seleciona e também faz uma lista dos 4 processos que mais fazem uso da memória. O que permanecer por mais tempo inativo também é selecionado. Isto ocorre até que a lista de páginas livres volte ao seu tamanho normal.

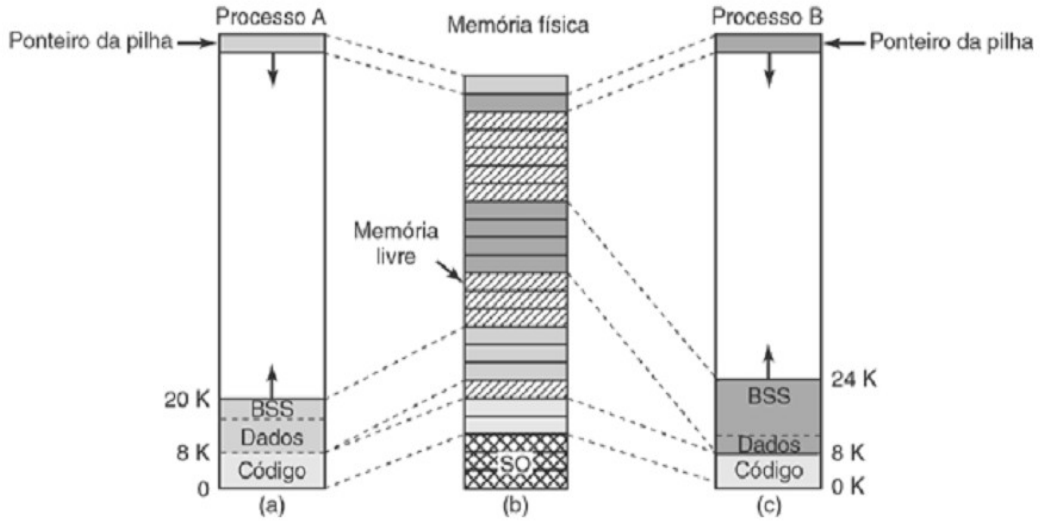

figura 2 - Gerência de memória com dois ponteiros [SOM - A.S Tanenbaum -cap 10]

Para controlar isto, uma estrutura com todas estas informações fica fora da memória junto ao Kernel do sistema, verificando o número de páginas livres e em uso, e realizando a gerência de memória.

2.7.3. Gerencia de Entrada / Saída

Conforme mencionado em Sistema de Arquivos, cada dispositivo de entrada e saída possui um ou mais arquivos especiais, que contém seus registros. A gerência de Entrada e Saída ocorre através destes arquivos, assim todas as operações de entrada e saída são realizadas como uma sequência de bytes, sendo assim é possível comunicar-se com mais de um dispositivo por vez. A parte de buffer e controle de acesso ficam por conta do kernel. Para manter um equilíbrio entre o kernel e os drives, o sistema implementa uma interface que padroniza o acesso por parte tanto do kernel (DKI – Driver Kernel Interface) quanto do *Driver* (DDI – *Device Driver Interface*).[Barbetti, 1997]

O DDI tem a função de isolar o dispositivo de Entrada e Saída do kernel, tornando-o independente. Somente após ser acoplado ao sistema é que o kernel é gerado. O sistema UNIX necessita de uma reinicialização do sistema para dar funcionalidade aos novos hardwares.

Os dispositivos que transmitem um grande bloco de informações recebem uma atenção especial do sistema operacional para evitar demoras com busca de informações no dispositivo e transferência para a memória. Para isto é utilizado um buffer na memória principal onde os blocos recebidos recentemente do dispositivo ficam armazenados por um determinado período de tempo. Assim, quando ocorrer uma nova operação de leitura, primeiramente é verificado o buffer para ver se lá não se encontra a informação necessária, para então fazer acesso ao dispositivo propriamente dito.

2.8. Processos e Threads

Na inicialização do sistema um processo denominado "0" é iniciado, e por sua vez inicia o processo "1" chamado de *init*. Esse processo é responsável por iniciar os processos posteriores, ou seja, sempre que um usuário logar, é ele quem inicia o shell para o novo usuário. Após a inicialização do shell os próximos processos criados pelo usuário são executados pelo próprio shell ou o shell cria um processo para executá-los. [Magalhaes, 2002]

Cada comando interpretado pelo Shell gera um processo com um identificador de processo (PID). A partir da criação deste a *systemcallfork,* cria os sub-processos, ou processos filhos, e os associa ao processo pai. O Sistema utiliza os PID's para acompanhar o status de cada processo.

Os status dos processos podem ser: [Barbetti, 1997]

- Executando – Processo carregado na memória e em execução.
- Bloqueado – Processo bloqueado pelo sistema operacional ou pelos dispositivos externos. Isto ocorre quando outro processo mais prioritário chega ou ocorre uma preempção então o sistema operacional guarda os dados do processo que estava executando em um cache e ele entra em estado de bloqueado até o outro processo mais prioritário ou a interrupção terminar.
- Espera – Aguardando a vez para fazer uso do processador ou aguardando Entrada e Saída de dados (I/O)
- Concluído – Processo já executado.
- Pronto: é o processo que está carregado na memória somente a espera de seu uso do processador.

Caso um processo seja muito grande e seja necessário executar outro processo, é possível deixá-lo para executar em *background,* ou segundo plano. Para iniciar ou transferir um processo para background basta somente colocar um "&" ao fim da linha de comando, por exemplo: ping 127.0.0.1 &.

Os processos pertinentes ao próprio sistema são denominados daemons. Eles são responsáveis por tarefas como gerenciamento de *logs*, escalonamento de tarefas, gerências de filas de impressão, e outras tarefas administrativas. O UNIX implementa o uso de sinais, que nada mais são do que avisos aos processos sobre possíveis erros em

sua execução. Os processos por sua vez podem aceitar estes sinais e iniciarem um tratamento especifico ou simplesmente ignorá-lo e seguir sua execução.

O sistema UNIX também implementa um vetor de processos, que é um tipo de tabela onde ficam representadas a estrutura dos processos em execução. Este vetor tem um tamanho definido que limita a quantidade de processos que podem estar executando. Quando um FORK é executado o sistema busca no vetor de processos um espaço livre para criar o processo filho, e só então é que são copiadas as informações do processo pai.

Inicialmente os sistemas UNIX não utilizavam Threads, mas com o crescimento de processos em execução não havia outra saída se não a de adotar processos concorrentes(threads) para otimizar o tempo de processamento. Em 1995 as Threads foram implantadas no sistema. Foi criado então o padrão POSIX, que definia a sincronização entre as threads como semáforos e variáveis condicionais.

3. HTML

HyperText Markup Language (Linguagem para marcação de hipertexto), “hipertexto é um conteúdo inserido em um documento para web e que tem como principal característica a possibilidade de se integrar a outros documentos web” [Silva, 2011].

Em 1992, a word wide *web* (tradução livre: 'Maior Teia Mundial') foi criada e acreditava-se que computadores diferentes poderiam compartilhar um mesmo hipertexto através de links globais chamados de hiperlinks. Após esta criação nasceu o *hypertext transferprotocol*(HTTP – Transferência para protocolos de Hipertexto). É um protocolo para recuperar hipertextos, e o formato de texto utilizado para criar o HTTP foi o HTML.[Silva, 2011].

Desde seu nascimento em 1990 até hoje a HTML já passou por 7versões , são elas:

- HTML: 1990;
- HTML+: 1993;
- HTML 2.0: 1995;
- HTML 3.2: 1997;
- HTML 4.0: 1997;
- HTML 4.1: 1999;
- HTML 5: 2007;

3.1. Um pouco mais sobre HTML5

Em 1994 nasce a W3C (WORD WIDE WEB Consortium). Um conjunto de empresas, pesquisadores e instituições com o objetivo de alavancar os estudos referentes à web, é também uma instituição normalizadora das normas que regem o mundo web.

Após a sua última recomendação sobre o HTML 4.1 em 1999 a W3C decidiu abandonar o desenvolvimento HTML já que a mesma havia se fundido com XML (Linguagem de Marcação extensível) formando o XHTML, mas em 2007 retomou os estudos no projeto HTML5 que estava nas mãos do grupo de trabalho para tecnologias de hipertexto em aplicações web (WHATWG).

Em janeiro de 2011 foi publicada uma matéria que continha a especificação de que o modelo HTML5 é de desenvolvimento exclusivo da W3C e o WHATWG desenvolveria tecnologias em geral para HTML sem sufixo da versão.

3.2. Recursos e Estrutura do HTML5

O HTML é uma linguagem interpretada pelos browsers (navegadores) sua função é criar a estrutura de uma pagina web.

Com o HTML5 foram criados diversos recursos que possibilitam uma estrutura muito mais organizada e com uma maior facilidade de criação. Além disto, o HTML5 fornece ferramentas para o CSS e JavaScript fazerem suas tarefas da melhor forma possível. Isto tudo através de suas APIs,ou seja, de suas novas funcionalidades que trabalham juntamente com o DOM *(DocumentObjectModel*). Algumas de suas APIs são:

- Canvas : Desenhos em tempo real em uma pagina;
- Validação de Formulários: Validação já nos campos de input ;
- Controles de Áudio e Vídeo: web players integrados;
- *DragandDrop*: arraste e solte elementos na pagina;

Outra novidade são *tags* (marcações) que foram criadas, pois nas versões anteriores do HTML não existia um padrão universal para rodapés, cabeçalhos, menus, entre outros. E também elementos como “B” e ”I”, que foram descontinuados nas versões anteriores, ressurgem com novo sentido no HTML5.[Silva- 2011].

Um Exemplo da Estrutura básica do HTML5:

```
1.<!DOCTYPE HTML>
2.<html lang=”pt-br”>
3.    <head>
4.        <meta charset=”UTF-8”>
```

5. **<link rel=”stylesheet” type=”text/css” href=”estilo.css”>**

6. **<title></title>**

</head>

7. **<body>**

8. **<!-- Conteudo -->**

9. </body>

10.</html>

1. DOCTYPE - Deve ser a primeira linha a ser escrita, pois informa ao navegador qual especificação de código deve utilizar

2. HTML – na linguagem HTML um elemento é filho de outro formando uma hierarquia e o maior elemento nesta hierarquia é o HTML este elemento vem acompanhado do atributo “lang” que informa aos navegadores qual a língua em que o documento foi escrita.

3. HEAD – É a parte da pagina que contem toda a informação sobre o conteúdo que foi publicado.

4. META charset – é responsável por apresentar ao navegador qual a tabela de caracteres que a pagina esta usando.

5. Link – referencia um arquivo esterno de CSS.

6. Title – Nesta parte fica o titulo do pagina o qual se apresenta na aba dos navegadores.

   Body – É o corpo da página, nela são colocados as tags e demais conteúdo que se apresentaram em forma de um website.

4. CSS

CascadingStyleSheet(Folhas de estilo em cascata), o CSS tem como finalidade dar vida as páginas HTML e devolver o propósito do HTML/XML que é o de estruturar uma página e não de se preocupar com cores, posições e coisas do gênero.

O CSS foi criado em 1994 por Tim Berners-Lee, e em 1996 CSS1 foi lançado e com recomendação da W3C. A Versão 2 Foi lançada em 1998, e a mais recente versão, a CSS3, foi lançada em 2008.

4.1. Recursos e Estrutura do CSS3

Através do CSS é possível alterar o tamanho de um elemento, dispô-lo em um local desejado na pagina, alterar suas cores, bordas, entre outros efeitos de estilo. Para isto o CSS utiliza uma regra que nada mais é que uma unidade básica de uma folha de estilo, ou seja, a menor porção de código capaz de produzir uma estilização na página.

A regra é composta por duas partes são elas:

1. Seletor: é o elemento HTML sobre o qual o efeito será aplicado.
2. Declaração: o efeito em si que se apresentará na pagina.
2.1 Propriedade: a característica a ser implantada.
2.2 Valor: é a quantidade ou tipo a ser estilizado.

Exemplos de estrutura:
No mesmo arquivo, deve ser adicionada logo após o elemento HTML <meta>:

```
<style>
div{
    background-color: Black;
    position: left;
}
</style>
```

Ou

```
<style>
    div{background-color: Black;  position: left;}
</style>
```

Utilizamos a primeira forma para uma melhor legibilidade de código, mas a segunda forma também é valida. Além destas formas, o CSS pode ser criado em um arquivo externo com extensão ".css" e ser chamado pela página HTML através do elemento <link rel="stylesheet" type="text/css" href="local/nome_do_arquivo.css">. Este arquivo deve conter somente a estrutura CSS:

```
div{
    background-color: Black;
    position: left;
}
```

5. JavaScript

Do que adianta ter uma página estruturada (HTML) e bonita (CSS) se a mesma é estática? Para resolver este problema se apresenta uma solução, à linguagem JAVASCRIPT.

O Java Script nasceu em 1996 seu criador Brendan Eich o introduziu no navegador netscape 2.0, neste mesmo amo a Microsoft lançou o Jscripte o introduziu em seu navegador. Como haviam duas linguagens distintas para realizar a mesma tarefa,

pode até se imaginar como os programadores da época sofriam para tornar um site dinâmico e que fosse passível de rodar em ambos os navegadores.

O nome oficial do JavaScript é ECMAScript por ter o grupo Europeu ECMA por trás de sua padronização. O ECMAScript ou JavaScript é uma linguagem orientada a objetos ou/e eventos que é responsável por realizar cálculos e manipular objetos entre outros eventos que estamos acostumados a ver nos sites da web.

5.1. Recursos e Estrutura do JavaScript

A linguagem JavaScript é responsável por criar a interação do usuário com a página, fornecer respostas as suas ações, apresentar avisos e bloquear acessos. Sua estrutura é semelhante a do CSS: uma tag do HTML faz um chamado da função JavaScript e esta função é criada no elemento <script> logo após o elemento <style>, ou no fim do código HTML, ou ainda em um arquivo separado. Nas Versões anteriores do HTML ele era o único responsável por validação dos campos de um formulário. Atualmente ele trabalha em conjunto com as novas funcionalidade do HTML5.

O armazenamento de informações através de cookies, sessionSorage ou localSorage são realizados via JavaScript. Buscas e conexões com o banco de dados também são feitos da mesma forma. Estas, dentre outras funcionalidades tão importantes em um site, só podem ocorrer por meio do JavaScript.

Exemplos da Estrutura JavaScript:

```
<script type="text/javascript">
  function onload(){
alert(“Helo Word!”); }
</script>
```

ou em arquivo externo com extensao “.js”:

```
functiononload(){alert(“Helo Word!”);}
```

A função onload() é executada no carregamento inicial da página, o elemento alert(); apresenta em tela um caixa de dialogo para o usuário contendo uma informação ou mensagem.

6. Considerações Finais

No mundo da tecnologia, principalmente no que diz respeito a sistemas para internet, a evolução é exponencial, evidenciando isto nos navegadores do usuário, com aplicações atraentes, rápidas, seguras e modernas.

Para acompanhar este crescimento é necessário que os desenvolvedores explorem novas ferramentas e busquem conhecimento constantemente. Esta obra tem como objetivo mostrar a grande evolução do HTML, CSS e JavaScript, incentivando o desenvolvedor utilizar adequadamente os novos recursos oferecidos por elas, em

conjunto com um sistema operacional gratuito, rápido, seguro e fácil - FreeBSD, o unix gratuito da Universidade Berkeley.

7. Conclusão

A leitura e interpretação do artigo deve proporcionar buscas mais profundas dos temas aqui tratados. Referente aos tópicos sobre web, este artigo não tem por objetivo o ensino da programação, e sim da apresentação da tecnologia.

Os tópicos sobre o sistema UNIX, são tratados com profundidade, ou seja traz uma resposta refinada a cada tema o que torna o estudo mais concentrado, ocorre então que alguns temas não são abordados, pois fogem do foco ou deixariam o artigo muito extenso.

8. Referências

- Arquitetura de Sistemas Operacionais 4ª Edição Francis Berenger Machado Luiz Paulo Maia, 2008.
- Sistemas Operacionais - Vol. 11: Série Livros Didáticos Informática UFRGS Rômulo S. Oliveira, Alexandre S. Carissimi, Simão S. Toscani.
- Introdução ao Sistema Operacional Unix - artigo- Cristiana Munhoz Eugênio, Daniela Regina Barbetti, 1997.
- CSS3. Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS3 – 2012 Mauricio Samy Silva.
- Introdução aos Sistemas Operacionais – Unicamp, Eleri Cardozo, Mauricio F. Magalhaes, 2002 (Artigo).
- HTML5 linguagem de marcação que revolucionou a web – 2011 – Mauricio Samy Silva.
- HTML5 Curso W3C Escritório Brasil – Elcio Ferreira e Diego Eis.
- JavaScript Guia do Programador – 2010 – Mauricio Samy Silva.