Revista Eletrônica da FANESE
ISSN 2317-3769

VISÃO GERAL DA ARQUITETURA DE SISTEMAS OPERACIONAIS LINUX COM ÊNFASE EM SISTEMAS MODERNOS

Ícaro C. Andrade Costa
Igor A. Santos Andrade
Tony B. Rodrigues Mota

RESUMO

Sistemas operacionais são sistemas grandes e complexos, muito difíceis de escrever e normalmente possuem um tempo de vida muito longo e evolução gradativa de acordo com o tempo. Fornecem um conjunto de abstrações que facilitam o uso dos sistemas computacionais de forma mais simples para seus usuários, seja o desenvolvedor de aplicativos ou mesmo o usuário final. Este artigo apresenta uma visão da arquitetura e funcionamento dos sistemas operacionais modernos, focado num dos sistemas que mais cresce no mercado, o Linux em sua distribuição Ubuntu.

ABSTRACT

Operating systems are large and complex systems, very difficult to write and usually have a very long life time and gradual evolution according to the time. Provide a set of abstractions that facilitate the use of computer systems in a simpler way for its users, it is the application developer or even the end user. This paper presents an overview of the architecture and operation of modern operating systems, focused on one of the fastest growing systems in the market, Linux, in its Ubuntu distribution.

1. Introdução

O Linux é um sistema operacional desenvolvido por Linus Torvalds baseado em UNIX e foi inspirado no sistema operacional Minix. A partir do trabalho de Linux Torvalds a primeira versão apareceu em 1991, e desde o seu surgimento houveram grandes desenvolvimentos e melhoras. Uma vez que é um software de código livre, é melhorado constantemente pelas contribuições de vários programadores que se identificam com a plataforma e a filosofia.

O Linux é um sistema operacional semelhante ao UNIX (*UNIX-like*) e que segue padrão POSIX (*POSIX-compliant*). Isso quer dizer que ele segue uma lista de padrões estabelecidos por institutos americanos padronizadores que especificam a compatibilidade

icarocarlosandrade@gmail.com - igorantonioandrade@gmail.com - tonybrm007@gmail.com

entre sistemas operacionais. O POSIX define uma interface de programação de aplicativos (*application programming interface - API*) para compatibilização entre variantes do UNIX, mas atualmente seguida pela maioria dos sistemas.

O componente principal do Linux é o seu kernel, e a partir dele existe uma grande variedades de sistemas operacionais, geralmente na forma de distribuições Linux. Uma distribuição Linux é mais simplesmente descrita como uma variedade particular de aplicações e softwares utilitários, empacotados juntamente com o kernel do Linux.

Com o surgimento da distribuição Ubuntu, o Linux ganhou e vem ganhando cada vez mais usuários entre o público mais leigo. Isso acontece graças a própria filosofia do projeto Ubuntu, que consiste em produzir um sistema que seja de fácil utilização por todos, não importa sua nacionalidade ou nível de conhecimento. Segundo pesquisa desenvolvida em 2010 pela Canonical, a empresa responsável pela distribuição, o Ubuntu já alcançava naquela data um número de 12 milhões de usuários.

Este trabalho tem como objetivo tratar das principais características do sistema operacional Linux, mais especificamente a sua distribuição Ubuntu em sua versão 12.04, a última versão LTS lançada até a confecção do presente trabalho. As versões LTS (Long-term supported - Com suporte a longo prazo) são versões mais estáveis do Ubuntu, lançadas a cada 2 anos, e possuem suporte de 3 anos para desktops e 5 anos para servidores.

2. Principais características

Existem diversas características importantes a se considerar em um sistema operacional, e essas podem variar conforme a sua arquitetura e o seu projeto. Seguem as principais a se considerar no sistema operacional Linux, distribuição Ubuntu 12.04.

2.1. Compatibilidade de Microprocessarodes

De acordo com a documentação do Linux, o SO oferece compatibilidade para um vasto número de arquiteturas de microprocessadores. Segue abaixo uma lista com a compatibilidade aos principais microprocessadores, recuperada da documentação do sistema Linux:

Intel: Intel 386SX/DX/SL, 486SX/DX/SL/SX2/DX2/DX4, Pentium, Pentium Pro, Pentium II, Pentium III (versões regulares e Xeon), Pentium 4 e Celeron (incluindo versões móveis de todos os itens acima) são todos suportados.

AMD: AMD 386SX/DX, 486SX/DX/DX2/DX4, K5, K6, K6-2, são todos suportados K6-3 e Athlon (todas as variedades, incluindo MP). Versões mais antigas do K6 devem ser evitados, pois podem ocorrer problemas. Desativando o "cache interno" na configuração da BIOS pode ser uma solução alternativa. Alguns versões mais antigas K6-2 300Mhz podem ter problemas com os chips de sistema. AMD Opteron de 64 bits e processadores Athlon64, bem como o Athlon64 móvel (ou Turion64), também são suportados, em execução ou em modo de 64 bits ou de 32 bits. Para o modo de 32 bits, compilar um kernel para i386, opcionalmente otimizados para Athlons, já que essencialmente estes processadores funcionam parecido com o modo de 32 bits. Para o modo de 64 bits, compilar um kernel para x86_64. Ele ainda vai executar binários de 32 bits, assumindo que todas as bibliotecas apropriadas estão disponíveis. Os processadores antigos NexGen também são suportados. Alguns mais antigos AMD 486DX podem travar em algumas situações especiais. Todos os chips atuais deve funcionar bem e conseguir uma troca de chip para velho CPU não deve ser um problema.

Cyrix: Cyrix 386SX/DX, 486SX/DX, 5x86, 6x86, e MediaGX são todos suportados.

IDT: IDT WinCHIP processadores C6-PSME2006A são suportadas no Linux.

Transmeta: Os Transmeta Crusoe processadores são suportados.

Existe também suporte para outras plataformas que não sejam x86 que seguem como informado na documentação: Alfa, ARM, CRIS (Axis Communications ETRAX 100LX CPU embutida), IA-64, m68k, MIPS, PA-RISC, PowerPC, S/390, SuperH e SPARC.

O Linux também suporta emulação de FPU (Float Point Unit), que é o hardware dedicado a executar operações matemáticas de dados representados em ponto flutuante em um computador. Além disso tem suporte para Multiprocessadores apartir das versões 2.x do kernel.

Segundo notícia divulgada no site *imasters.com.br* em dezembro de 2012, versões futuras do Linux passarão a não suportar processadores Intel 386, considerando que a série de processadores começou a ser lançada em 1985, sendo muito antiga e pouco utilizada hoje. Essa remoção do suporte, segundo a matéria do site, removerá uma grande complexidade do kernel.

Com relação a compatilibidade com diferentes hardwares, a documentação do Ubuntu 12.04 estabelece uma lista de arquiteturas as quais o sistema operacional suporta. Segue tabela presente na documentação:

Tabela 1. Arquiteturas compatíveis com Ubuntu 12.04, segundo documentação

Architecture	Ubuntu Designation	Subarchitecture	Flavor
Intel x86-based	i386		
AMD64 & Intel EM64T	amd64		
ARM	armel	Marvell Dove	dove
		Freescale i.MX51	imx51
		TI OMAP	omap
		Versatile	versatile
IBM/Motorola PowerPC	powerpc	PowerMac	pmac

Ainda segundo a documentação do Ubuntu 12.04, a imagem padrão do kernel para as plataformas i386 e amd64 suporta o multiprocessamento, sendo que esse suporte é desativado caso seja identificado apenas um processador. No entanto, a plataforma powerpc não tem suporte e o kernel trabalha exclusivamente com a primeira CPU, sendo preciso substituir o kernel padrão do Ubuntu para usufruir das vantagens do multiprocessamento.

2.2. Arquitetura

Em seu projeto geral, o Linux lembra qualquer outra implementação tradicional do UNIX sem microkernel. É um sistema multiusuário e multitarefa com um conjunto completo de ferramentas compatíveis com o UNIX. Os detalhes internos do projeto do Linux tem sido muito influenciados pela história de desenvolvimento do UNIX [SILBERSCHATZ et al. 2013].

O sistema é dividido em três partes: o kernel, as bibliotecas do sistema e os utilitários do sistema. O kernel é o responsável por todas as abstrações importantes do sistema

operacional, as bibliotecas definem as funções por meio das quais as aplicações podem interagir com o kernel e os utilitários são programas que executam tarefas de gerenciamento individuais especializadas. Todo o código do kernel do Linux é executado em modalidade privilegiada e não há código de modalidade de usuário embutido no kernel.

O kernel do Linux segue o modelo histórico do UNIX, sendo um kernel monolítico. Segundo Machado e Maia (2013), "a arquitetura monolítica pode ser comparada com uma aplicação formada por vários módulos que são compilados separadamente e depois linkados, formando um grande e único programa executável, onde os módulos podem interagir livremente."

A principal razão dessa escolha de projeto é o seu desempenho. Considerando que todo o código e estruturas de dados são mantidos no mesmo espaço de endereçamento, nenhuma mudança de contexto é preciso para processos chamarem funções do sistema. Isso porém acarreta um problema pois como todos os módulos são executados em um mesmo espaço de endereçamento, um erro em um módulo pode derrubar todo o sistema.

Os diferentes módulos são códigos que trabalham no espaço de endereçamento do kernel, mas não fazem parte do mesmo código. Dessa forma, módulos diferentes podem ser desenvolvidos para atender necessidades de novos drivers e a partir daí podem ser compilados e adicionados ao kernel já em execução. Os módulos do kernel permitem que seja configurado um sistema Linux com um kernel mínimo padrão, sem quaisquer drivers de dispositivos adicionais embutidos.

O kernel do Linux é distribuído sob a licença *GNU General Public License versão 2 (GPLv2)*, que traz quatro liberdades: a de executar o programa para qualquer propósito; a de estudar como o programa funciona e adaptá-lo para as suas necessidades; a de redistribuir cópias; e a de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos de modo que toda a comunidade se beneficie deles. Para que possa utilizar dessas liberdades o usuário tem acesso ao código fonte do Linux gratuitamente.

2.3. Tipos e estrutura do escalonamento de processos

O escalonamento é a atividade de alocar o tempo de uso da CPU a diferentes tarefas do sistema operacional. Considerando que o processador é um recurso muito disputado e deve ser bem aproveitado, os algoritmos de escalonamento devem trabalhar no sentido de fazer o melhor uso desse tempo entre os diferentes processos que concorrem a uma parcela do uso.

A principal tarefa de um escalonador é definir algoritmos para gerenciar o uso do processador pelos diferentes processos. Nesse contexto existem várias questões que devem ser levadas em conta por um escalonador, como dar um bom tempo de resposta; permitir que todos os processos tenham a oportunidade de utilizar o processador, evitando a espera indefinida; impedir que um processo monopolize o uso do processador; e conciliar processos de diferentes prioridades.

Em sistema de tempo real o escalonador deve seguir algoritmos em que os processos permaneçam com o uso do processador até que surja um outro processo com maior prioridade. Em sistemas com tempo compartilhado os processos recebem fatias de tempo do processador (quantum) e são utilizados meios para dividir o tempo de uma forma justa entre os processos.

O escalonador do Linux é de *time-sharing*, onde o tempo do processador é dividido pelos processos. Cada processo ocupará sua parte do tempo e quando o tempo é esgotado um novo processo deve ser selecionado para execução, ocorrendo uma troca de contexto. Podemos dizer então que o escalonador do Linux é preemptivo.

O escalonamento no Linux é feito com base em prioridade, sendo que existem dois

intervalos: um intervalo para processos de tempo real com valores entre 0 e 99, e um valor de ajuste variando de 100 a 140. Esses dois intervalos são mapeados para um esquema de prioridades global em que valores numericamente mais baixos indicam prioridades mais altas [SILBERSCHATZ et al. 2000, 2013].

Os processos com prioridades definidas para tempo real não podem ter suas prioridades alteradas pelo escalonador, já os processos interativos tem a sua prioridade alterada constantemente pelo escalonador. Essa prioridade é calculada em função da natureza do processo, *I/O bound* ou *CPU bound*, sendo que o escalonador Linux privilegia os processos *I/O bound* em relação aos *CPU bound* de forma a oferecer um melhor tempo de resposta às aplicações interativas. Também é considerado no cálculo da prioridade o tempo de execução do processo. Aqueles que já tiveram seu quantum de tempo esgotado são realocados para o final da fila. Vale ressaltar que o escalonador Linux só executa processos de prioridade dinâmica quando não houver processos de tempo real.

2.4. Algoritmos de escalonamento utilizados

O escalonamento no Linux é feito considerando as threads, e não os processos. Sendo assim, é previsto três algoritmos: um algoritmo de compartilhamento de tempo para obtenção de um escalonamento justo e preemptivo, exclusivo para processos de prioridade dinâmica; um algoritmo em fila (FIFO) para processos de tempo real que não sofrem preempção; e um algoritmo de chaveamento circular para processos em tempo real que devem sofrer preempção. Na prática, esse último algoritmo não é usado para processos em tempo real, mas para aqueles que tem prioridade superior aos processos de tempo compartilhado [TANENBAUM 2010].

Diferente dos escalonadores de muitos outros sistemas, o escalonador do Linux atribui à tarefas de prioridade mais alta um *quanta* de tempo mais longo e à tarefas de prioridade mais baixa um *quanta* de tempo mais curto [SILBERSCHATZ et al. 2013]. Isso é feito para processos de tempo compartilhado, em que sua execução depende do tempo restante que tenha do processador.

Conforme um processo é executado até que tenha todo o seu tempo de execução consumido ele é removido para o final da fila até que todos os outros processos tenham tido a oportunidade de utilização da CPU. Um processo pode ter seu processamento interrompido também por questões de acesso a dispositivos de entrada/saída. Nessa situação, considerando que ele não consumiu toda a sua fatia de tempo, ele é realocado para voltar a executar antes daqueles que tenham executado completamente.

É mantido então duas listas para os processos, uma para processos ativos e outra para processos expirados. O escalonador busca sempre na lista de processos ativos aquele com maior prioridade de execução. Qualquer processo que tenha sua execução interrompida por acessos a dispositivos de entrada/saída aguardará pelo evento e voltará a execução. Processo que tenha consumido todo seu tempo de processamento são movidos para a lista de processos expirados e sua prioridade é novamente definida quando voltam para a fila de ativos.

Já nos algoritmos de processo de tempo real não há disputa pelo uso do processador, sendo executado sempre o processo com prioridade mais alta. No método FIFO para processos em tempo de execução o processo é executado até que seu tempo tenha se exaurido ou que tenha sido bloqueado. Já no algoritmo de chaveamento circular para processos em tempo real os processos são executados até que depois de um tempo sofrem preempção e são transferidos para o final da fila.

2.5. Sistema de arquivos

O primeiro sistema de arquivos utilizado no Linux foi o derivado do MINIX, que possuia uma série de restrições como o tamanho máximo permitido para nome de arquivos de 14 caracteres e tamanho máximo de arquivo de 64MB. Esse foi substituído pelo *ext (Extended File System)*, que já permitia arquivos com até 255 caracteres de nome e tamanho de até 2GB, mas trazia o problema de ser mais lento que o original do MINIX.

Em 1992 foi liberado o *ext2 (Second Extended File System)* com o objetivo de corrigir os problemas do ext. Esse possui uma grande influência do UNIX, podendo ser vista no fato de utilizar grupos de blocos, análogos àqueles utilizados pelo seu sistema de arquivos, o FFS. O bloco, que consiste num conjunto de setores (cada setor tem 512 bytes), é a menor unidade de alocação para o ext2. O tamanho pode ser de 1024, 2048 ou 4096 bytes e é definido na formatação. O ext2 possui técnicas para reduzir o movimento da cabeça de leitura da unidade de disco e para minimizar a fragmentação, pré-alocando até oito blocos quando um arquivo é aberto para gravação.

O sistema de arquivos *ext3 (Third Extended File System)* foi criado por Stephen C. Tweedie e passou a ser utilizado no Linux a partir de novembro de 2001, na versão 2.4.15, sendo um dos mais utilizados até hoje pelo Ubuntu e muitas outras distribuições. Nessa versão do sistema de arquivos foi adicionado o *Journaling*, que consiste em manter as modificações que ocorram gravadas em um log de transações. Quando pendentes no sistema, essas transações ficam alocadas num registro até que o sistema operacional confirme que realmente foram concretizadas, sendo assim descartadas. Seu objetivo principalmente é proteger o sistema em casos de falha ou desligamento repentino. Nesses casos, quando o sistema voltar, as transações são retomadas do ponto em que parou.

Em 11 de outubro de 2008 o *ext4 (Fourth Extended File System)* foi adicionado ao código do Linux, em sua versão 2.6.28. Assim como seus antecessores, possui uma série de melhoras, como o suporte a volumes de até 1EB e arquivos com tamanho de até 1TB. Traz melhoras também no Journaling através do uso de checksums, que melhoram a sua confiabilidade. Possui compatibilidade retroativa com as versões do ext3 e ext2 e é o sistema de arquivos mais utilizados nas versões mais recentes do Linux, inclusive o Ubuntu.

Existe ainda o *NFS (Network File System)*, o sistema de arquivos de rede. Foi desenvolvido pela Sun Microsystems e é usado em todos os sistemas Linux modernos para juntar os sistemas de arquivos localizados em computadores separados em um sistema logicamente unificado. Possui uma arquitetura bastante interessante pois permite que um conjunto qualquer de clientes e servidores compartilhe um sistema de arquivos em comum, podendo ser executado em redes locais ou mesmo em redes de longa distância [TANENBAUM 2010].

Existem ainda, diversos outros sistemas de arquivos compatíveis com o Linux, tais como o *XFS*, *ReiserFS* e *VFAT*, que nada mais é do que o sistema *FAT* comumente utilizado no Windows.

2.6. Estrutura de controle e relacionamento com dispositivos de Entrada/Saída

A gerência de dispositivos de entrada/saída (E/S) é uma das principais e mais complexas funções de um sistema operacional. Sua implementação é estruturada através de camadas em um modelo semelhante ao apresentado para o sistema operacional como um todo. As camadas de mais baixo nível escondem características dos dispositivos das camadas superiores, oferecendo uma interface simples e confiável ao usuário e suas aplicações [MACHADO 2007].

Para tratar da diversidade de dispositivos existentes o sistema operacional implementa uma camada, chamada de subcamada de E/S, com a função de isolar a complexidade dos

dispositivos da camada de sistemas de arquivo e da aplicação, permitindo ao sistema operacional ser flexível ao realizar a comunicação dos processos com qualquer tipo de dispositivo.

Para se comunicar com os dispositivos de E/S conectados ao computador o usuário pode utilizar as *rotinas de entrada/saída*, um conjunto de rotinas que faz parte do subsistema de E/S e permite realizar operações sem se preocupar com detalhes do dispositivo que está sendo acessado. As operações de E/S devem ser realizadas através de *system call*, que chamam as rotinas de E/S do núcleo do sistema operacional.

As operações de entrada e saída podem ser *síncronas* e *assíncronas*. As operações síncronas são aquelas em que o processo que realizou a operação fica aguardando no estado de espera até o seu termino. Nas assíncronas o processo não aguarda a finalização, sendo necessário algum mecanismo de sinalização para informar quando for finalizado.

O controle de dispositivos de E/S no Linux é feito integrando os dispositivos ao sistema de arquivos, chamando-os de *arquivos especiais*. Cada dispositivo adicionado ao sistema é associado a um nome, que normalmente se localiza no diretório */dev*. Esses arquivos especiais podem ser acessados da mesma maneira que os demas arquivos, ou seja, os programas podem abrir, ler e escrever da mesma maneira como é feito com os arquivos comuns [TANENBAUM 2010].

O Linux divide todos os dispositivos, ou arquivos especiais, em três categorias: *dispositivos de blocos, os dispositivos de caracteres e os dispositivos de rede* [SILBERSCHATZ et al. 2013].

Os dispositivos de blocos incluem todos os dispositivos que permitem acesso aleatório a blocos de dados de tamanho fixo, incluindo discos rígidos, disquetes, CD-ROMs e memória flash. São usados para armazenar sistemas de arquivos, e seu acesso direto é também permitido para que os programas possam criar e reparar o sistema de arquivos que o dispositivo contém.

Os dispositivos de caracteres incluem a maioria dos outros dispositivos, como mouses e teclados. São assim chamados por terem sua transmissão de dados feita em fluxos de caracteres e normalmente a leitura e escrita nos dispositivos é feita de forma imediata. Enquanto o acesso aos dispositivos de blocos pode ser feito aleatoriamente, os dispositivos de caracteres só são acessados serialmente.

Os dispositivos de rede, diferentemente dos de blocos e de caracteres, não permite transferir dados diretamente, sendo preciso abrir uma conexão com o subsistema de conexão de rede do kernel [SILBERSCHATZ et al. 2013].

Para tratar de aspectos mais específicos como a velocidade de comunicação, tipos de operações realizadas, representação dos dados e outros detalhes dos dispositivos, são utilizados *drivers*. Os drivers costumam ser particulares para cada dispositivo e sua função principal é a de receber comandos abstratos dos subsistema de E/S e traduzi-los para comandos que o controlador possa entender e executar. Além disso, o driver pode realizar outras funções, como a inicialização do dispositivo e seu gerenciamento.

Os drivers são divididos em duas partes, ambas as quais são partes do núcleo do Linux e executam em modo kernel. A metade superior executa no contexto do chamador e faz a interface com o restante do Linux. A metade inferior executa no contexto do núcleo e interage com o dispositivo [TANENBAUM 2010].

Os controladores são componentes de hardware responsáveis por manipular diretamente os dispositivos de E/S. O sistema operacional, mais exatamente o driver, comunica-se com os dispositivos através dos controladores. Em geral, o controlador pode ser uma placa independente conectada a um slot do computador ou implementado na mesma

placa do processador [MACHADO 2007].

2.7. Interação homem-máquina: Interface, eficiência, percepção e apresentação

Nos sistemas Linux é muito comum o uso do interpretador de comando, também conhecido como *shell*, como interface de comunicação do usuário com a funcionalidades do sistema operacional. Existe uma série de interpretadores de comando para Linux, entre eles o *ksh*, *csh*, *bash*, sendo esse último o usado no Ubuntu por padrão.

O Bash é fortemente baseado no shell original do UNIX, o *Bourne*. Seu nome, na verdade, é o acrônimo para *Bourne Again SHell* [TANENBAUM 2010]. Através dele o usuário pode enviar comandos ao sistema operacional para a execução de programas. Essa interação entre o usuário e o shell pode ser de forma interativa, onde são digitados os comandos diretamente no terminal e passados ao interpretador para execução, ou de forma não-interativa, com a criação de arquivos de script que contêm uma série de comando sequenciais para serem executados.

O interpretador de comandos é a interface preferida para os usuários mais avançados e para os programadores por ser rápida e poderosa. Entretanto, cada dia mais aumenta a quantidade de usuários com baixo conhecimento na área que buscam o Linux como uma alternativa gratuita para sistema operacional, e esse usuários não estão familiarizados (nem tem interesse em se familiarizar muitas vezes) com as telas pretas e linhas de comando. Para facilitar e tornar prático o uso dos sistemas operacionais é que foram criadas as interfaces gráficas.

A interface gráfica (*GUI - Graphical User Interface*), cria um ambiente de área de trabalho, uma metáfora já conhecida, com janelas, ícones, pastas, barras de ferramenta e funcionalidades do tipo arrastar e soltar. Um ambiente de área de trabalho completo contém um gerenciador de janelas, que controla a disposição e a arrumação das janelas, e oferece uma interface gráfica consistente [TANENBAUM 2010].

Os ambientes gráficos mais populares para o Linux são o *GNOME (GNU Network Object Model Environment*) e o *KDE (K Desktop Environment)*. O GNOME era utilizado como interface gráfica do Ubuntu até a versão 10.10, quando passou a ser usado como interface alternativa dando lugar ao Unity como ambiente padrão a partir da versão 11.04. O Unity tem como principal objetivo a simplicidade e a integração entre os diversos tipos de dispositivos usados pelo ubuntu, podendo ser usado por mouse, teclado ou mesmo com *touch*. A partir da versão 11.10 o Unity passou a ser a única interface utilizada no Ubuntu, havendo a necessidade de instalação do GNOME caso seja desejado pelo usuário. O KDE pode ser utilizado como interface gráfica no Kubuntu, projeto derivado do Ubuntu.

As interfaces gráficas no Linux em geral são executadas pelo Sistema X Window, conhecido também como X11 ou apenas X, que define a comunicação e exibição de mapas de bits para sistemas UNIX e afins. O servidor X é o principal componente para controle de dispositivos como teclados, mouses e monitores e é o responsável pelo redirecionamento da entrada ou da saída para os programas do cliente. Pode ser iniciado manualmente, a partir de uma linha de comando, mas é comum que seja iniciado durante o processo de inicialização por meio de um gerenciador de tela [TANENBAUM 2010].

3. Vantagens e desvantagens

O Linux como um sistema operacional tem vários aspectos favoráveis ao seu uso. Por ser um projeto de software livre, ele é distribuído gratuitamente em suas diferentes distribuições. Entretanto, o fato de ser um software livre não impede que empresas se especializem em

produzir versões voltadas para usos específicos e vender seus produtos, ou até mesmo trabalhar com suporte na ferramenta. Nesse aspecto várias distribuições são mantidas por empresas que trabalham dando suporte a uma determinada distribuição que desenvolvem. Isso o torna um SO que ao mesmo tempo tem baixo custo, pois pode ser adquirido gratuitamente, como também um software seguro com versões com suporte de grandes empresas.

Outro fator interessante é a segurança. Considerando que é um sistema com um número de usuário bem menor que outros como Windows, ele é pouco visado em relação a ataques de pragas como vírus. Além disso, qualquer comando ou acesso a recurso do sistema é controlado solicitando acesso de "usuário root", dificultando a execução de um vírus no sistema.

O Ubuntu traz vantagens particulares da sua distribuição. Como seu propósito é ser uma distribuição Linux de fácil uso, ele tem grandes desenvolvimentos em relação à interface gráfica. Um outro grande destaque nessa questão da usabilidade é a sua "Central de Programas", que é um ambiente amigável onde o usuário tem uma vasta lista de aplicativos a sua disposição, bastando apenas pesquisar pelo que deseja e socilitar a instalação.

Suas desvantagens estão relacionados a inexistência de suporte a certos drivers e dispositivos. Isso ocorre principalmente por que os desenvolvedores dos determinados dispositivos não produzem um driver para o Linux no momento de lançamento, se fazendo necessário que usuários da comunidade busquem desenvolver.

Outro aspecto importante é que questões de configuração acabam exigindo do usuário um conhecimento profundo. Muitas funcionalidades precisam ser configuradas através de alterações em arquivos e muitas configurações são feitas através de comandos de terminal, exigindo do usuário conhecimento de comandos do shell. No entanto, esse aspecto acaba sendo mais uma característica do sistema operacional, por ser software livre e permitir que essas configurações sejam alteradas, enquanto que em outras plataformas fechadas isso não é possível.

4. Conclusão

Um sistema operacional é um software bastante complexo de ser implementado e mesmo entendido. Possui diversas estruturas e níveis de abstração, e pode ser implementado de variadas formas a depender do interesse dos seus projetistas e do propósito do mesmo. Tem como objetivo principal fornecer uma interface de comunicação dos recursos computacionais com os programas do usuário, exibindo um modelo de computador mais simples e limpo.

Nesse artigo apresentamos apenas uma visão geral sobre sistemas operacionais focando no sistema Linux, sua arquitetura e funcionamento. Uma possibilidade de extensão desse trabalho seria tratar de tópicos não discutidos como o gerenciamento de memória e a segurança no Linux. Poderia ser feito ainda um estudo de outro sistema operacional bastante utilizado, o Windows da Microsoft, com o objetivo de realizar um comparativo.

References

Alberto, M. C. (2013), "Sistemas Operacionais: Conceitos e Mecanismos". Disponível em: <http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/so:so-cap01.pdf>. Acesso em: Março, 2014.

Brockmeier, J. (2010), "Canonical Announces 12 Million Ubuntu Users, Google Makes a Comeback". Disponível em: <http://ostatic.com/blog/canonical-announces-12-million-ubuntu-users-google-makes-a-comeback>. Acesso em: Março, 2014.

Campos, A. (2007), "Compatibilidade Linux: Processadores". Disponível em: <http://br-

linux.org/linux/compatibilidade-linux-processadores>. Acesso em: Março, 2014.

GNU Operation System, "GNU Bash". Disponível em: <https://www.gnu.org/software/bash/bash.html>. Acesso em: Março, 2014.

iMasters (2012), "Linux não será mais compatível com processadores 386". Disponível em:<http://imasters.com.br/noticia/linux-nao-sera-mais-compativel-com-processadores-386/>. Acesso em: Março, 2014.

Jones, M. T. (2007), "Anatomia do Kernel Linux". Disponível em: <https://www.ibm.com/developerworks/br/library/l-linux-kernel/>. Acesso em: Março, 2014.

Jones, M. T. (2009), "Anatomia do Ext4". Disponível em: <http://www.ibm.com/developerworks/br/library/l-anatomy-ext4/>. Acesso em: Março, 2014.

Jones, M. T. (2012), "Visão Geral sobre o Linux, o Sistema Operacional que é uma Plataforma Universal". Disponível em: <http://www.ibm.com/developerworks/br/library/l-linuxuniversal/>. Acesso em: Março, 2014.

Machado, F. B., Maia, L. P. (2007), "Arquitetura de sistemas operacionais", Rio de Janeiro, LTC Editora, 4ª edição.

Machado, F. B., Maia, L. P. (2013), "Arquitetura de sistemas operacionais", Rio de Janeiro, LTC Editora, 5ª edição.

Melo, A. G. S. (2009), "Arquitetura do Kernel Linux". Disponível em: <http://www.ic.unicamp.br/~islene/1s2009-mc514/Kernel_Linux.pdf>. Acesso em: Março, 2014.

Morimoto, C. (2009), "O Ubuntu terá 200 milhões de usuário em 2015?". Disponível em: <http://www.hardware.com.br/noticias/2011-05/ubuntu2015.html>. Acesso em: Março, 2014.

Nemeth, E., Hein, T. R., Snyder, G. (2007), "Manual Completo do Linux: Guia do Administrador", São Paulo, Pearson Prentice Hall, 2ª edição.

Oliveira, R. S., Carissimi, A., Toscani, S. (2010), "Sistemas Operacionais", Porto Alegre, Bookman Companhia Editora LTDA, 11ª edição.

Poirier, D. (2011), "The Second Extended File System". Disponível em: <http://www.nongnu.org/ext2-doc/ext2.html>. Acesso em: Março, 2014.

Ribeiro, U. (2004), "Certificação Linux", Rio de Janeiro, Axcel Books do Brasil Editora.

Silberschatz, A., Galvin, P. B., Gagne, G. (2013), "Fundamentos de sistemas operacionais: princípios básicos", Rio de Janeiro, LTC, 1ª edição.

Silberschatz, A., Galvin, P. B., Gagne, G. (2000), "Sistemas operacionais: conceitos e aplicações", Rio de Janeiro, Campus, 1ª edição.

Sneddon, J. (2011), "Mark Shuttlewordth delivers UDS keynote; sets goal for 200 million Ubuntu user in 4 years". Disponível em: <http://www.omgubuntu.co.uk/2011/05/mark-shuttleworth-delivers-uds-keynote-address-sets-goal-for-200-million-ubuntu-users-in-4-years/>. Acesso em: Março, 2014.

Sneddon, J. (2012), “Linux Marketshare is Rising”. Disponível em: <http://www.omgubuntu.co.uk/2012/01/is-linux-marketshare-on-the-rise-it-seems-so/>. Acesso em: Março, 2014.

Software no governo do Brasil, “Linux 2.6.0: Lançado novo kernel do linux”. Disponível em: <http://www.softwarelivre.gov.br/noticias/KernelLinux/>. Acesso em: Março, 2014.

Tanenbaum, A. S. (2010), “Sistemas operacionais modernos”, São Paulo, Pearson Prentice Hall, 3ª edição.

The Linux Documentation Project, “Linux Hardware Compatibility”. Disponível em: <http://www.tldp.org/HOWTO/Hardware-HOWTO/cpu.html>.

Ubuntu Documentation, “Supported Hardware”. Disponível em: <https://help.ubuntu.com/lts/installation-guide/powerpc/hardware-supported.html>. Acesso em: Março, 2014.

Ubuntu Unity, “Unity”. Disponível em: <https://unity.ubuntu.com/projects/unity/>. Acesso em: Março, 2014.

Viva o Linux, “Sistemas de arquivos para GNU/Linux”. Disponível em: <http://www.vivaolinux.com.br/artigo/Sistemas-de-arquivos-para-GNU-Linux/?pagina=1>. Acesso em: Março, 2014.