CENTRO UNIVERSITÁRIO FACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

ESTUDO DE CASO SOBRE SOFTWARE LIVRE E MODO DEBIAN DE PRODUÇÃO

ADAIR WOLFF

LAGES (SC), DEZEMBRO DE 2012.

CENTRO UNIVERSITÁRIO FACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

ESTUDO DE CASO SOBRE SOFTWARE LIVRE E MODO DEBIAN DE PRODUÇÃO

ÁREA(s)
Estudo de caso / Sistema Operacional

Adair Wolff

Projeto apresentado á Banca Examinadora do trabalho de conclusão do Curso de Ciência da Computação para análise e aprovação.

Lages (SC), Dezembro de 2012.

ESTUDO DE CASO SOBRE SOFTWARE LIVRE E MODO DEBIAN DE PRODUÇÃO

Adair Wolff

Este projeto foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, do curso de Ciência da Computação do Centro Universitário Facvest.

.

_____

Profº Márcio José Sembay. Msc
Coordenador do Curso

Banca Examinadora:

_____

Profº Márcio José Sembay. Msc
Orientador

Lages (SC), Dezembro de 2012.

EQUIPE TÉCNICA

Acadêmico(a)
Adair Wolff

Professor Orientador
Márcio José Sembay. Msc

Professor Co-orientador
Cassandro Devens. Esp

Coordenador do Curso
Márcio José Sembay. Msc

DEDICATÓRIA

Dedico este trabalho aos meus pais Sebastião e Eloir, que sempre me incentivaram a buscar os meus objetivos, mesmo quando eles pareciam distantes; a estudar e sempre acreditar que ser competente é o mais importante.
Dedico também a minha irmã Adriana, que também sempre me incentivou, compartilhou e ajudou com as suas experiências, pois estando em um curso diferente do meu me ajudou a ver problemas e questões da informática sob outro ponto de vista.

AGRADECIMENTOS

Agradeço a todos que contribuíram de alguma forma em ter me permitido cursar uma faculdade, sonho tão difícil de realizar para tantas pessoas.

Agradeço aos meus professores por repassarem seus conhecimentos, pela paciência na hora de nos ensinar e principalmente pela dedicação que alguns tiveram nos momentos extraclasse, mostrando que o trabalho do professor não acaba quando ao final da aula mas sim que o trabalho do professor vai além da sala de aula, e agradeço aos colegas de sala, por compartilharem de suas experiências, e aos meus amigos que me incentivaram e auxiliaram de tantas formas para que eu chegasse até aqui.

SUMÁRIO

LISTA DE ABREVIATURAS

FSF – Fundaçao de Software Livre

CPD – Centro de Processamento de Dados

DNS – Domain name System

ID – Identificação Digital

GPL – General Public Licence

IP – Internet Protocol

ISO – International Organization for Standardization

POP3 – Post Office Protocol version 3

HTTP – Hypertext transport protocol

FTP – Protocolo de transferência de arquivos

SO – Sistema Operacional

TCC – Trabalho de Conclusão de Curso

TCP – Transmission Control Protocol

SARG – Squid Analisys Reports Generator

ACL – Lista de Controle de Acessos

GCC – Gnu Compiler Collection

IRC – Internet Relay Chat

GNU – Gnu is Not Unix

KDE – K Desktop Environment.

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O software livre garante ao usuário a liberdade de ter acesso ao código-fonte do software por ele utilizado, de realizar ele mesmo adaptações no software conforme a sua realidade e permite ainda que esse software alterado possa ser redistribuído para outros usuários que compartilham de tal necessidade.

Atender aos diferentes interesses dos usuários de softwares num mercado globalizado cada vez mais dependente de informática tem se mostrado uma tarefa bastante desafiadora até mesmo para grandes empresas mundialmente famosas; o modo de produção do software livre Linux Debian tem se mostrado bastante eficiente diante de tal desafio, alcançando um crescimento rápido e com resultados satisfatórios, gerando inovações úteis ao usuário e se adaptando continuamente aos novos modelos de engenharia de software que se apresentam no mercado fazendo com que até mesmo grandes empresas de software incorporem pelo menos parte do seu modo de produção com o objetivo de atingir melhores resultados e ajudar na inclusão digital dos estudantes das escolas publicas.

ABSTRACT

The Free Software assure to the user the freedom about have access to the source code from the software used by himself, makes himself adaptations in the software as his reality and allows that the changed code can be redistributed to the others users who share this same need.

Attend all the distinct insterests from the software users into a globalized market even more dependent of informatic has showed a hard task even to the bigger enterprises mundially famous; the free software Linux Debian production mode has showed eficient enough facing this type of challenge, reaching a fast growing with good results, bringing up usefull inovations to the user and continuously adapting himself to the new shapes of software engineering that appears in the market making even the biggers softwares enterprises incorporate at least one part from his production mode with the objective of gain better results and help in the digital inclusion of the pupils from the public school.

I – INTRODUÇÃO

1. APRESENTAÇÃO

A informática tem se mostrado um fator modificador do mundo, e tomado um lugar importante na sociedade, suas alterações tendem a facilitar a vida de vários usuários em diferentes países, e para atender as necessidades dessa grande demanda de usuários com interesses bastante diversificados um padrão de produção vem se destacando e vem influenciando até mesmo os demais modos de produção proprietários, esse modo de produção de softwares livres vem promovendo uma grande revolução na indústria da produção de software.

O estudo de caso tem o interesse de afirmar o lugar do software livre dentre os meios de produção de software, ressaltar suas qualidades e mostrar de que forma ele esta aprimorando e influenciando não apenas com seus métodos, mas também com sua filosofia, o modo de produção de softwares em todo o mundo.

Além de promover um estudo sobre o funcionamento do Linux e as etapas do modo de produção dos softwares livres por todo o mundo, foi mantido uma preocupação em exemplificar de maneira pratica como personalizar de maneira fácil e rápida seu sistema operacional Linux, de forma que ele atenda as necessidades do usuário sem ser necessária a preocupação com a aquisição de softwares de terceiros, sendo necessário apenas um estudo aprofundado no modo de funcionamento do sistema e na pratica das alterações da programação dos arquivos e scripts que compõem o sistema operacional Linux.

2. JUSTIFICATIVA

Levando em consideração o grande crescimento nos últimos anos dos sistemas direcionados aos usuários, e percebendo a grande influencia do modo de produção dos softwares livres até mesmo sobre as empresas de modo de produção proprietário, apresentando uma alternativa rápida, inovadora, competitiva e efetiva na criação de softwares cada vez mais presentes no cotidiano dos usuários, auxiliando a sociedade em diversos aspectos.

3. IMPORTÂNCIA

A importância do estudo se compreende em fornecer um estudo de caso a respeito da importância do software livre na sociedade, como agente modificador não apenas no espaço em que atua, mas também como uma ideologia que vem influenciando os demais modos de produção de uma forma que os torne mais eficientes, e provando sua importância na participação na produção de softwares cada vez mais competitivos e eficientes.

4. OBJETIVOS DO TRABALHO

4.1 OBJETIVO GERAL

O trabalho tem como objetivo apresentar um estudo de caso sobre o sistema de produção da versão do sistema operacional Linux Debian, exemplificando o modo de produção de software livre, apresentando-o como alternativa viável para investimentos em estudo e aplicação da produção de software no mercado atual e que permite que sobre o mesmo sejam implementados projetos do governo que visam melhorar a qualidade de vida da população.

4.2 OBJETIVOS ESPECÍFICOS

Os objetivos Específicos deste trabalho são:

- Mostrar como funciona o modo de produção de software livre;
- Mostrar a importância do software livre na sociedade;
- Apresentar a criação de scripts que alterem e personalizem o Linux;
- Modificar a estrutura dos arquivos Linux de modo a ajustá-lo as necessidades do usuário de forma rápida e muitas vezes inacessível ao usuário de um sistema proprietário.

5. METODOLOGIA

A figura 1 demonstra as etapas necessárias para o desenvolvimento do TCC (Trabalho de Conclusão de Curso).

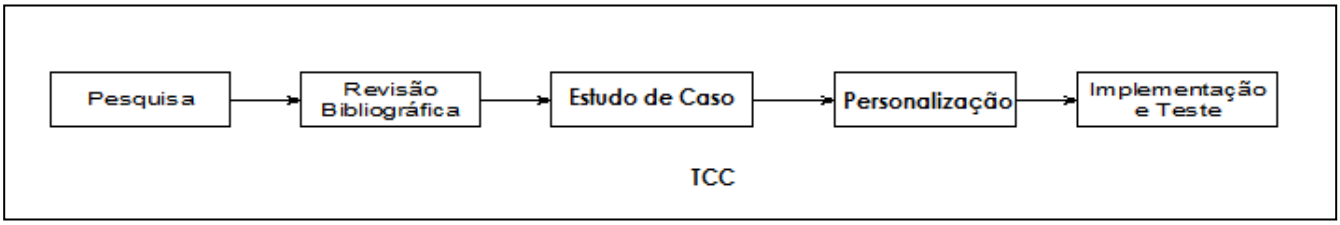

Figura 01. Etapas do desenvolvimento do TCC

Para alcançar os objetivos do TCC, indicado na figura 1, inicia-se pelo levantamento de todo material bibliográfico necessário para fundamentar os conceitos e justificativas que serão abrangidos durante todo o trabalho. Este material é proveniente, basicamente de livros e de páginas Web.

Durante a revisão bibliográfica, baseando-se em todo material pesquisado anteriormente, serão efetuadas todas as fundamentações teóricas do trabalho, desde conceituação básica até a conclusão final do mesmo. Todo esse processo será acompanhado pela coordenação do TCC e principalmente pelo professor orientador.

Na fase de desenvolvimento do estudo de caso foi documentado todos os passos que compõem o modo de produção da distribuição Debian Linux.

Na fase de personalização do sistema foram implementadas algumas alterações dentro da distribuição Linux Debian que não seriam possíveis de forma tão pratica dentro de um sistema operacional proprietário, adaptando o sistema operacional instalado de acordo com as necessidades do usuário sem que fosse preciso recorrer a outros aplicativos, apenas estudando e conhecendo o modo de funcionamento do Software Livre.

Na fase de implementação e teste foram adicionadas as alterações nos scripts que compõem o sistema operacional Linux Debian e ainda criados alguns Shell scripts para que o resultado fosse alcançado e a personalização do sistema atingisse níveis bastante satisfatórios.

6. CRONOGRAMA

A metodologia apresentada na tabela 01 demonstra os passos para o desenvolvimento do TCC.

Primeiramente, foi feito uma longa pesquisa de material teórico para início do trabalho, após isso foi realizado a confecção e entrega da apresentação da pré-proposta. Após a aprovação da pré-proposta pela coordenação, será iniciado o desenvolvimento da revisão bibliográfica.

Posteriormente, será feita o estudo de caso do sistema à ser estudado para posterior entrega e defesa do TCC, a banca examinadora, apresentando o trabalho.

Atividade	Julho de 2012	Agosto de 2012	Setembro de 2012	Outubro de 2012	Novembro de 2012
Pesquisa	x	x			
Confecçao e apresentaçao da pré-proposta		x			
Elaboraçao e revisão bibliográfica		x	x	x	
Desenvolvimento do estudo de caso			x	x	
Entrega do TCC à banca avaliadora					x

Tabela 01: Cronograma do TCC.

II – REVISÃO BIBLIOGRÁFICA

1. DEFINIÇÃO DE SOFTWARE LIVRE

O software livre surgiu não somente como uma forma de produzir software de forma rápida e de fácil evolução, mas sim com toda uma ideologia agregada, a mente por trás disso tudo se trata de Richard Stallman, fundador da Free Software Foundation.

De forma semelhante ao software de código aberto, o software livre é qualquer programa de computador cujo código-fonte esta disponível a qualquer usuário para permitir o uso, cópia, estudo do código-fonte e redistribuição, seu ponto principal é se opuser ao software proprietário e seu modo de produção fechada, muito usada dentro das empresas grandes fabricantes de software, mas esse modo de produção proprietário vem perdendo espaço para o modo de produção livre nos últimos anos, fazendo com que as empresas de tecnologia mesclem os dois modos de produção com o objetivo de alcançar maior evolução e contentamento de seus usuários dentro do mundo globalizado.

> Longe de ser decididamente pós-moderno, o ciberespaço pode surgir como uma espécie de materialização técnica dos ideais modernos. Em particular, a evolução contemporânea da informática constitui uma impressionante realização do objetivo marxista de apropriação dos meios de produção pelos próprios produtores. (LEVY, 1997, p. 23)

O modo de produção de software livre vem sido inserido ao modo de produção tradicional de modo a trazer melhores resultados do que o modo antigo de produção, hoje dá para observar grandes empresas às quais antigamente pregavam contra o software livre adaptando seu modo de produção de forma que pareça pelo menos um pouco com a ideologia de Richard Stallman que foi tão odiada pelas grandes empresas proprietárias no passado.

O modo de produção proprietário havia predominado o mercado de produção de software em meados dos anos 90, desde o sistema operacional básico ao funcionamento até os programas básicos para edição de texto, planilhas etc. tornando qualquer usuário de computadores em qualquer lugar do mundo altamente dependentes desse tipo de monopólio, mas com o passar do tempo esse tipo de produção proprietário não se mostrou competente o suficiente para atender as necessidades de tantos usuários ao redor do mundo, sendo produzidos softwares que desejavam a desejar, tanto em desempenho quanto em

recursos, para que os softwares voltassem a atender às necessidades do maior numero de usuários foi necessário ressuscitar um modo de produção antigo o qual foi bastante útil nos primórdios da informática, chamado de ecossistema de colaboração, que se trata do modo como às universidades abriam e distribuíam o código-fonte entre as outras universidades dessa forma alcançando resultados de forma mais rápida e muito mais satisfatória do que de maneira proprietária.

> O que caracteriza a verdadeira revolução tecnológica não é a centralidade de conhecimentos e informação, mas a aplicação desses conhecimentos e dessa informação para a geração de conhecimentos e de dispositivos de processamento e proliferação da informação, em um ciclo de constante realimentação cumulativo entre inovação e uso, esse determinado ciclo evolutivo da realimentação entre a introdução de uma nova tecnologia, seus usos e seus desenvolvimentos em novos domínios, torna-se muito mais rápido no novo paradigma tecnológico e conseqüentemente a difusão desse conhecimento amplifica seu poder de forma infinita, à medida que os usuários apropriam-se inteiramente delas e a redefinem, vindo a afirmar que as novas tecnologias da informação não são simplesmente ferramentas a serem aplicadas, mas processos a serem definidos com o tempo e a colaboração contínua ao redor do mundo. (CASTELLS, 2010, p. 320)

A Free Software Foundation considera um software como livre quando atende aos quatro tipos de liberdade para os usuários. As quatro liberdades são: Liberdade para executar o programa, liberdade par estudar como o programa funciona através de seu código-fonte, liberdade de redistribuir cópias do programa e a liberdade de modificar e redistribuir suas alterações de modo de que toda a comunidade se beneficie.

Para que essas quatro liberdades sejam alcançadas os softwares tem que ser distribuídos com o código fonte inclusos, e que também não sejam colocadas restrições para que os usuários alterem e redistribuam o mesmo. Para que isso seja possível, as licenças são importantes, a mais famosa se trata da General Public Licence (GPL), à qual rege a maioria dos softwares livres conhecidos, dentre eles o Linux e boa parte dos softwares que vem inclusos nele.

É garantido, portanto a alteração do software pelo usuário de forma que suas necessidades sejam atendidas, e o uso das mesmas alterações em quantas máquinas for necessário aos usuários, e se for redistribuída a cópia com as alterações essa cópia deve conter também o código-fonte para que a mesma atenda as quatro liberdades e continue sendo de software livre.

A GPL é a licença que judicialmente garante as quatro liberdades aos usuários e desenvolvedores de software livre, fazendo com que um aplicativo somente possa ser chamado assim quando atender plenamente todas as liberdades descritas pela GPL, permitindo assim ao usuário

redistribuir copias para qualquer um, em qualquer lugar, sem que seja necessário pagar, e nem ao menos pedir por essa permissão, e ao realizar alterações no código fonte o usuário pode usar normalmente tanto para o desenvolvimento do seu trabalho ou lazer, sem nem mesmo mencionar que essas alterações existem, essa é a liberdade para utilizar e reescrever o código conforme a necessidade do usuário.

Já a liberdade de redistribuir as cópias alteradas ou não do software livre deve incluir tanto o programa em si quanto seu código-fonte, dando as mesmas liberdades que foram recebidas pelo usuário para aqueles que adquirirem a copia, garantindo assim a liberdade de todos.

Alguns exemplos de software livres conhecidos são o Sistema Operacional Linux, e a grande maioria dos aplicativos que rodam sobre ele, o mensageiro instantâneo Amsn, o criador de DVDs K3B, o Wine que simula uma plataforma Windows dentro de um sistema operacional Linux, dentre muitos outros exemplos, alem de softwares, projetos e licenças baseados não diretamente na GPL mas que contem bastante de sua ideologia.

1.2 SURGIMENTO DO SOFTWARE LIVRE

Com base no manifesto GNU publicado por Richard Stallman, foi estabelecido as bases para um projeto posteriormente chamado de Free Software Foundation (FSF) e tinha a pretensão de criar um sistema livre que seria chamado GNU (Gnu is not Unix), o qual seria baseado no sistema operacional unix.

> O movimento do aplicativo livre (freeware em inglês) surgiu em 1984 de uma constatação: não era possível fazer funcionar um computador sem antes instalar um sistema operacional obtido com uma licença restritiva. Essencialmente da Microsoft na época, mas também da Apple. Richard Stallman, pesquisador do Massachusetts Institute of Technology (MIT), funda então a Free Software Foundation, para apoiar o movimento iniciado por ele mesmo. (PISANI, 2010, p. 64)

Nesse mesmo ano Stallman popularizou o conceito de copyleft, um mecanismo legal para proteger a distribuição e modificação de códigos-fonte e assim constituir as diretrizes de redistribuição de software livre. E a primeira licença do GPL foi liberada para o software GNU Emacs, um editor de textos o qual esta evoluindo até hoje e vem nativo em muitas distribuições Linux.

Richard Stallman é primeiramente um ativista político, por isso suas bases para o software livre podem se estender hoje muito além da produção de software, e tem influenciado não somente as grandes empresas de produção de software mas tambem a industria da musica e entretenimento,

incluindo até a web 2.0 que possui raízes muito fortes com o conceito de software livre criado por Stallman.

1.3 DIFERENÇAS ENTRE SOFTWARE LIVRE E DE CÓDIGO ABERTO

Segundo Pisani (1999), o aplicativo livre – que pode circular livremente, o que não significa que seja gratuito – e o open source – aplicativo cujo programa-fonte é aberto – são dois movimentos muito próximos, freqüentemente confundidos, mas cuja historia não é a mesma. Apesar de suas filosofias serem parecidas e o resultado, para o usuário final, ser o mesmo.

Alguns exemplos de software de código aberto são o browser Mozilla Firefox, A IDE Eclipse, e as linguagens PHP, Perl, Ruby e Python, os sistemas operacionais FreeBSD e o Android, o pacote de aplicativos para escritório OpenOffice.org nomeado aqui no Brasil como BrOffice, o servidor web Apache, o compactador de arquivos 7-Zip dentre muitos outros exemplos.

O software livre se diferencia do software de código aberto, pois suas ideologias são um pouco diferentes, o software de código aberto surgiu de forma diferente do software livre, surgindo em 1998, em Palo Alto, com a decisão da empresa Netscape de publicar os códigos-fonte da ultima versão de seu navegador em janeiro de 1998 e nessa época foi proposto o termo open source para caracterizar essa forma de desenvolvimento, mas também de forma de trabalho muito mais aberta e colaborativa, a abertura do código-fonte do Netscape Navigator deu origem a vários outros browsers muito usados hoje em dia, como o Mozilla Firefox.

> Um aplicativo open source não é um programa sem licença. É um programa cuja licença é aberta, com redistribuição autorizada, e cujo código-fonte esta disponível, permitindo trabalhos derivados, já o software livre garante por meio da GPL o direito de cópia, distribuição e alteração do software. (PISANI, 2010, p. 79)

Basicamente o software de código aberto nada mais é do que um software cujo código fonte esta disponibilizado, seja pela internet ou por meio de um CD, DVD ou outro tipo de mídia secundária, para fins de estudo ou curiosidade, garantindo por meio da Open Source Iniciative o direito de que qualquer pessoa pode ter acesso ao código do programa mas que ainda assim as condições de uso e distribuição do software original são determinadas pelo desenvolvedor que disponibilizou o código.

Sendo assim muitos aplicativos que são de código aberto não são necessariamente softwares livres, já que para atender as exigências da Free Software Foundation e da GPL é necessário principalmente o cumprimento das quatro liberdades básicas, de execução para qualquer fim, de estudar, redistribuir e liberar seus aperfeiçoamentos e seu código fonte.

2. LICENÇA GPL E SEUS DERIVADOS

A GPL é a licença mais importante desde o surgimento do projeto GNU, a General Public Licence foi idealizada por Richard Stallman em 1989, sendo a licença que rege a maior parte dos softwares livres, em parte por sua adoção pelo sistema operacional GNU/Linux, dentre outros exemplos como o software base da Wikipedia.

Para que o software livre possa ser redistribuído é regra que a GPL tem que estar contida na documentação do software livre quando o mesmo for distribuído, porém a mesma tem que ser cedida em sua língua original de criação, o inglês isso quer dizer que não pode ser traduzida para outra língua, pelo menos não de forma oficial para que não possa ser prejudicada pela tradução errônea de algum de seus termos, sendo assim a GPL pode ser encontrada em inglês em meio a documentação dos softwares livres quando adquiridos em forma de pacotes, como na versão brasileira Linux Mandriva, e muitas vezes disponibilizada pelo fabricante pela internet.

> A idéia principal do software livre e de código aberto não é a de construir um sistema gratuito. O código aberto permite a cada um contribuir com melhorias e distribuir os produtos assim transformados, o sucesso de suas adaptações se fará de acordo com a adesão dessas alterações, se elas forem realmente uteis aos demais usuários, fazendo com que o software alcance um crescimento de forma natural. Uma distribuição pode ser gratuita ou paga, mas deve respeitar o código aberto. (PISANI, 2010, p. 150)

Um dos heróis desse movimento é Linus Torvalds, que propôs em 1991 um dos mais importantes núcleos do sistema de exploração desenvolvido pela comunidade do programa livre, o Sistema Operacional Linux, o qual introduziu as bases para o modo de produção de software livre do jeito que é executado hoje, esse foi a principal qualidade de Linus Torvalds ao criar o Linux, pois com o novo núcleo do sistema operacional surgiu também uma proposta de melhor aproveitamento da colaboração, à nível mundial, possibilitando com isso não apenas o crescimento do Linux mas de todos os demais softwares livres conhecidos.

2.1 CREATIVE COMMONS

A Creative Commons é uma das organizações que trabalham com licenças, inspirada na GPL que garante os direitos do projeto GNU/Linux, e dos softwares livres de forma geral, se trata de uma organização não governamental sem fins lucrativos que esta localizada na cidade de São Francisco na Califórnia, nos Estados Unidos, e foi fundada por Lawrence Lessig.

O interessante das licenças regidas pela Creative Commons, é que ela não atende somente o campo de softwares, mas sim de todo processo criativo, com diferentes tipos de derivados, através de suas licenças que permitem a copia e o compartilhamento com menos restrições, sendo determinada pelo tipo de licença adquirida e de como o usuário quer ver distribuída sua licença.

As licenças Creative Commons foram idealizadas para permitir a padronização de declarações de vontade no tocante ao licenciamento e distribuição de conteúdos culturais em geral, tais como texto, musicas, imagens, filmes, dentre outros processos criativos cujos detentores dos direitos autorais queiram disponibilizar a licença sob alguma regência das licenças da Creative Commons.

As licenças Creative Commons têm antecedentes na Open Publication License (OPL), na GNU General Public Licence (GPL) e na GNU Free Documentation License (GFDL), todas licenças que tem por objetivo proteger os interesses de quem tem objetiva uma participação maior no desenvolvimento de suas obras intelectuais ou maior distribuição das mesmas. Diferenciando-se das licenças citadas anteriormente por não incluírem necessariamente dentre os direitos disponibilizados ao publico, conforme o tipo de licença adquirido, a possibilidade de manipulação do conteúdo por meio do código aberto, podendo garantir unicamente a livre manipulação, a distribuição, compartilhamento e replicação destes conteúdos protegidos pela Creative Commons. Existem seis grandes licenças da Creative Commons, as quais cuidam de diferentes interesses entre seus adquirentes, Attribution (CC-BY), Attribution Share Alike (CC-BY-SA), Attribution No Derivatives (CC-BY-ND), Attribution Non-Commercial (CC-BY-NC), Attribution Non-Commercial Share Alike (CC-BY-NC-SA), Attribution Non-Commercial No Derivatives (CC-BY-NC-ND), licenças as quais tratam de diferentes tipos de utilização da propriedade intelectual protegida pelas mesmas, atuando em vários campos alem da produção de software, como por exemplo das musicas, literatura, artes de uma forma geral, mostrando como o espírito colaborativo é mais abrangente do que apenas a produção de softwares.

3. PAPEL DA COLABORAÇÃO NA INFORMÁTICA

Embora o software livre e de código aberto pareçam conceitos relativamente novos, suas origens estão situadas nos primórdios da informática, conceitos estes tão essenciais que sem os quais boa parte da tecnologia que sustenta a era da informação atualmente, como o Unix ou o Tcp/ip não teria surgido sem um sistema de colaboração entre diversos programadores situados em locais distantes um do outro redistribuindo seus códigos em busca de maior evolução.

A base colaborativa faz parte da evolução da informática e é por meio dela que se tornou possível a evolução tão rápida e de forma eficiente de determinadas tecnologias, segundo Pisani:

> Os primórdios da informática foram marcados por duas grandes empresas, a IBM e a Digital Equipment Corp (DEC), e como suas receitas provinham em sua maioria da venda de equipamentos físicos, os programas eram tratados como uma ferramenta para a organização e o funcionamento do sistema, devido a escassez de memória primária nos equipamentos de hardware nessa época os recursos para criar essas ferramentas com um bom aproveitamento eram raros e o potencial do computador era profundamente afetado se os software fosse produzido de forma medíocre e certamente afetaria de forma drástica o desempenho do equipamento, com isso no inicio dos anos 1970 trocava-se facilmente esse código entre as universidades e entre programadores, afinal sua propriedade intelectual não era ainda claramente definida, e com isso se tinha maior eficiência na programação dos equipamentos antigos das grandes empresas. (PISANI, 2010, p. 246)

Grandes softwares essenciais na historia da informática surtiram a partir de um sistema de colaboração e liberação de código entre as universidades em meados dos anos 1970, sistemas operacionais como o Unix, linguagens de programação que servem de base para a maioria das linguagens de programação mais usadas hoje em dia como a linguagem C, e protocolos como o TCP/IP só puderam surgir de forma que supra os propósitos para os quais foram feitos de forma eficiente graças a colaboração de diversas universidades e empresas desenvolvedoras de tecnologia, naquela época os conceitos de patentes ainda não estavam completamente definidos e as empresas não ambicionavam ganhar dinheiro apenas com software, o que fazia com que sua propriedade intelectual não fosse fechada, e ao ser repassado o código-fonte para outros projetos poderia receber novas implementações e tomar proporções muitas vezes diferentes do inicial, porem efetuando de forma eficiente suas funções graças a colaboração não apenas de uma equipe fechada, mas de diversos colaboradores ao redor do mundo.

> O paradigma da tecnologia da informação não evolui para seu fechamento como um sistema, mas rumo a abertura como uma rede de acessos múltiplos. É forte e impositivo

> em sua materialidade, mas editável e aberto em seu desenvolvimento histórico, deixando claro que a informática desde o seu inicio dependeu da ampla colaboração descentralizada, e para aproveitar melhor seu potencial atualmente é preciso bases colaborativas sólidas e que permitam a liberdade do estudo e junção de idéias não apenas como aprendizado, mas com objetivo de atingir um determinado propósito da forma mais eficiente, rápida e bem construída. (CASTELLS, 2010, p. 190)

Sem essa abertura talvez a informática não tivesse evoluído de forma tão rápida, sem que os softwares criados por certos indivíduos tivessem sido reavaliados por outros usuários e somados a outras ideias com a intenção de atender de forma cada vez mais eficiente as necessidades das empresas solicitantes, projetos governamentais e boa parte das empresas e da sociedade que precisava de aplicativos cada vez mais eficientes.

3.1 UNIX

Sem a base colaborativa nos primórdios da informática talvez uma boa parte dos bons softwares que hoje fazem parte da historia da informática nem existissem, pelo menos não da forma eficiente pelas quais são conhecidas, e evolução natural das idéias de forma a serem recebidas, incrementadas e retornarem ao campo da discussão de forma a serem novamente reavaliadas e se necessário modificadas com o propósito de suprir determinada carência.

O sistema operacional Unix é um ótimo exemplo de como a colaboração pode gerar frutos funcionais bastante avançados mesmo para sua época em um espaço curto de tempo, vindo a atingir resultados eficientes os quais outros sistemas proprietários normalmente levariam anos para atingir.

> Ken Thompson que trabalhava nos laboratórios Bell para a AT&T escreveu o código fonte de um novo sistema operacional para rodar um jogo chamado Space Travel, e à partir desse ponto com a ajuda de Dennis Ritchie eles reescreveram o jogo e o sistema operacional para rodar em um computador pouco usado na empresa chamado Digital Equipment Corporation PDP-7. Para adequar o novo sistema operacional às suas necessidades, Thompson programou um sistema de arquivos que havia sido projetado por ele anteriormente e alguns utilitários simples de arquivo como o comando cp, mv dentre outros e um interpretador de comandos o qual seria denominado posteriormente de “Shell”. O Unix recebeu esse nome porque a equipe de Thompson estava trabalhando no desenvolvimento de um sistema operacional o qual era chamado de Multics, e como que para estabelecer um contraponto entre o projeto antigo e o novo sistema desenvolvido o mesmo recebeu o nome de Unix. (TAYLOR, 1997, p. 219)

Não demorou muito para que a AT&T percebesse o potencial de seu novo sistema operacional Unix o qual foi escrito inicialmente em linguagem de programação Assembly, mas devido

aos novos investimentos da AT&T sobre o promissor sistema operacional multiusuário sobre o qual estavam trabalhando no momento, foi justificada a aquisição de novos computadores e com isso surgiu o projeto da criação de uma nova linguagem de programação, a qual foi baseada na linguagem chamada “linguagem B”, e em homenagem a essa a nova linguagem acabou sendo chamada de “linguagem C” e sobre a mesma todo o sistema operacional Unix foi reescrito.

Na década de 1970, a AT&T foi proibida de vender o sistema Unix, e esperando bons resultados os laboratórios Bell distribuíram o seu promissor sistema operacional para faculdades e universidades, essas instituições também compravam o poderoso e barato sistema de computação PDP-11 sobre o qual o novo sistema Unix foi reescrito em linguagem C, formando assim uma combinação perfeita, em pouco tempo o Unix era o sistema operacional escolhido para o desenvolvimento de software e pesquisa. Logo grandes empresas se envolveram diretamente no processo de desenvolvimento principalmente a Hewlett-Packard, a Sun Microsystems e a Digital Equipment Corporation, e o Unix estava disponível também para PowerPc, Apple e IBM.

Com isso o Unix de hoje não é um produto de uma dupla de inspirados programadores dos laboratórios Bell. Mas com o investimento, pesquisa e contribuição sobre seu código o sistema pôde agregar inúmeras funções muito úteis e inovadoras para sua época e ajudando com adições significativas ao sistema, propiciando para que o mesmo se transformasse na lenda que é hoje. E não apenas o sistema operacional Unix cresceu com a participação das empresas e universidades que estudavam e contribuíam com seu código, mas também a linguagem C e o protocolo TCP/IP os quais estavam agregados ao Unix.

Como foi mostrado o sistema Unix juntamente com a linguagem C e o protocolo TCP/IP alcançaram seu status de peças importantes na historia da informática graças à contribuição de varias universidades dos Estados Unidos, e com cada implementação bem sucedida somada a outra foi dando origem a um sistema operacional cada vez mais eficiente e que atendia a finalidades distintas de forma bastante razoável e com ferramentas e protocolos que evoluíam junto de forma cada vez mais rápida.

4. ATUAÇÃO DO SOFTWARE LIVRE NA SOCIEDADE

De acordo com Dubner (1993), A informação é a moeda da internet. Como veículo, a internet é incrivelmente eficiente na transferência de informação das mãos de quem a possui para as mãos de quem não a possui. A internet conseguiu o que nenhum advogado especialista em direitos do consumidor conseguiria: diminuir de forma drástica o abismo entre os especialistas e o público. Isso veio a contribuir

de muitas formas com o modo de produção dos softwares livres, agilizando dessa forma o processo de aquisição do produto, teste e também ao modo reportar uma falha fazendo com que a mesma tenha a possibilidade de ser corrigida de forma mais rápida.

> As comunidades de software livre estão ocupando o seu lugar ao lado dos mercados abertos e das empresas hierárquicas como uma importante alternativa de estratégia competitiva e organização do trabalho. Elas potencializam as motivações humanas básicas para transformar o trabalho que antigamente seria considerado não-remunerativo em substancial valor econômico (..) Isso inclui acesso à capacidade computacional e a aplicativos, transparências, globalização, democratização do conhecimento e das competências, e crescente complexidade dos sistemas. (TAPSCOTT, 2007, p. 121)

Tanto o software livre quanto a sua ideologia estão inseridos em meio a tecnologia da informação dentro da sociedade, sendo bastante improvável que algum cidadão comum não esbarre com ela em meio ao seu cotidiano, é bastante comum haver caixas eletrônicos rodando sobre alguma distribuição Linux, ou trabalhar em algum lugar cujo banco de dados rode sobre o Mysql, o qual é regido sob a licença GPL, ou mesmo acessar o site da sua instituição de ensino, local de trabalho, banco ou loja on-line a qual o servidor web rode sobre o servidor apache, o qual esta presente em grande parte dos servidores web espalhados pelo mundo e também esta regido sob a licença GPL.

Um ponto em que o software livre se destaca bastante é sobre o desenvolvimento de novos projetos, sobre os quais os desenvolvedores buscam uma base sobre a qual implementar melhorias no projeto, e continuar a partir do mesmo visando atingir determinado resultado, um dos casos mais eficientes o qual utiliza o sistema operacional Linux é a urna eletrônica brasileira, a qual utiliza um núcleo de sistema operacional Linux sobre o qual é rodada, fazendo com isso que grande parte dos eleitores brasileiros tenham utilizado o sistema operacional do pingüim pelo menos uma vez a cada eleição.

Figura 02: Inicialização da urna eletrônica.

Fonte: Hardware. 2012.

O software livre pode vir a favorecer não apenas os estudantes interessados em evoluir seu nível de programação, mas também vem a ajudar dentro da comunidade em um âmbito geral, auxiliando na tentativa das pequenas e médias empresas se tornarem mais competitivas num mercado cada vez mais dependente de tecnologia.

> Essas empresas de pequeno ou médio porte estão adotando avidamente o código aberto não porque são entusiastas desse modelo", diz Polese, "mas porque podem obter uma funcionalidade incrível por uma fração do custo". De repente, ferramentas criadas para que grandes empresas gerenciem equipes de vendas, clientes, conteúdo, dados e recursos estão ao alcance de legiões de pequenas e médias empresas. Agora, elas têm uma oportunidade de alcançar ou superar os níveis de eficiência e eficácia gerencial que eram vistos apenas em grandes companhias. (Tapscott, 2007, p.113)

Com os recursos disponibilizados em forma de software livre um micro ou pequeno empresário já pode inserir recursos tecnológicos em seu empreendimento, já podendo ter um site rodando sobre o Apache, um banco de dados com recursos rodando sobre o Mysql, um servidor Linux rápido e eficiente conectando os computadores de seu estabelecimento e podendo ainda integrar computadores

rodando Windows e Linux com um servidor Samba, com um investimento relativamente baixo do que seria se recursos semelhantes a esse tivessem que ser adquiridos de uma empresa que fabrica softwares proprietários, aumentando a competitividade no mercado e permitindo aos pequenos empresários acompanhar a tecnologia e competir em pé de igualdade com empresas de maior porte.

4.1 INCLUSÃO DIGITAL

Em meio à era da informação o domínio da tecnologia se torna crucial não apenas no desenvolvimento dos países emergentes, mas também para a inclusão dos menos abastados no mercado de trabalho, o qual está cada vez mais concorrido e ao mesmo tempo cada vez mais dependente das novas tecnologias, fazendo com que o domínio de computadores e conhecimento sobre a internet se torne imprescindíveis para se conseguir um emprego nos dias de hoje.

Segundo o que foi pensado por Castells (2010), O fato de países e regiões apresentarem diferenças quanto ao momento oportuno de dotarem seu povo do acesso ao poder da tecnologia representa fonte crucial de desigualdade em nossa sociedade, ao analisar a sociedade em rede, Castells traz uma perspectiva na qual a posse do domínio da tecnologia se torna imprescindível para que uma maior igualdade social possa ocorrer, e sem ela cria-se um grande abismo quase que intransponível entre quem domina a tecnologia e quem não tem acesso a esse tipo de evolução.

Sem os softwares livres os projetos de inclusão digital se tornariam demasiado caros e atingiriam muito menos pessoas que realmente necessitam desse tipo de ajuda, o software torna possível ao governo investir muito menos dinheiro para desenvolver toda uma suíte que envolve sistema operacional e aplicativos em sua maioria geridos pela GPL, com o objetivo de distribuir de distribuí-lo nas escolas publicas, inserindo com isso os estudantes e até mesmo alguns professores da escola publica no uso dessa nova tecnologia tanto para operação quanto para aprender com os novos meios de acesso a informação os quais são cruciais na gestão de um negocio próprio ou para a inserção no cada vez mais concorrido e dependente de tecnologia mercado de trabalho.

Um bom exemplo desse tipo de procedimento é o Sistema Operacional denominado Linux Educacional, baseado no Linux Debian o qual conta com a ajuda do governo federal e visa desenvolver um sistema operacional o qual já esta sendo utilizado em muitas escolas estaduais pelo Brasil inteiro. Sendo que o Linux Educacional se trata de uma versão Linux baseada no Linux Debian e implementada

com a interface gráfica KDE, contando também com o pacote para escritório chamado BrOffice, o qual traz toda uma suíte de aplicativos para o uso dentro de escritórios, e com isso podendo ser implantado em escolas de modo a familiarizar as muitas crianças que não tem acesso à computadores em suas residências e lhes ensinar como usar essa nova tecnologia muito valiosa para entrar no mercado de trabalho hoje em dia, como editores de textos, planilhas, pesquisas na internet, etc.

> O universal sem totalidade não foge a regra da exclusão. Apenas não se trata demais de adesão ao sentido, mas sim de conexão. O excluído está desconectado. Não participa da densidade relacional e cognitiva das comunidades virtuais e da inteligência coletiva. A cibercultura reúne de forma caótica todas as formas de heresias. Mistura os cidadãos com os bárbaros, os pretensos ignorantes e os sábios. Contrariamente as separações do universal clássico, suas fronteiras são imprecisas, moveis e provisórias. Mas a desqualificação dos excluídos não deixa por isso de ser terrível. (LEVY, 1997, p. 68)

Todos esses projetos visam diminuir esse abismo social que poderia muitas vezes inibir os cidadãos mais carentes de concorrer em vagas dentro do mercado de trabalho contra outros indivíduos que fazem parte de classes mais abastadas, e nenhum desses projetos seria possível sem a existência tanto da GPL que rege os softwares livres, quanto de softwares de código aberto que compõem todo o pacote sobre os quais o governo investe e toda uma equipe trabalha antes de chegar ao usuário final e atender as necessidades apresentadas pelo cliente.

4.2 EXEMPLOS BEM-SUCEDIDOS DE SOFTWARE LIVRE

Dentre os softwares livres existem muitos casos bastante conhecidos de todos nós como o browser Mozilla Firefox, o qual é fruto da liberação do código do navegador Netscape, dando origem a um dos maiores concorrentes hoje do Internet Explorer da Microsoft, já sendo bastante usado por grande parte dos usuários de internet do mundo inteiro, outro lugar onde é utilizado o software livre e grande parte dos usuários nem se dá conta disso é no software base da Wikipedia, Wiki, o qual permite aos vários usuários que possam atualizar as informações que ficarão disponíveis na enciclopédia livre para pesquisas futuras, sendo que alem de rodar sobre um software livre, a Wikipédia compartilha da mesma filosofia tão divulgada por Richard Stallman, a de que qualquer um pode participar da criação de conteúdo, alterá-lo da maneira que achar correto e lucrativo para os demais membros da comunidade.

Esse tipo de filosofia do software livre influenciou as bases para a Web 2.0, ou seja, mostrou que o usuário criando produtos para que os demais usuários pudessem consumir seria uma boa idéia, se

mostrando bastante eficiente em redes sociais como o Orkut, Facebook, sites de conteúdo em vídeo como o youtube, onde qualquer um pode disponibilizar o vídeo de um casamento dando a todos que tem o interesse de asssistir aquele determinado conteúdo vê-lo em qualquer lugar do mundo, esse tipo de visão veio a influenciar até mesmo a criação de aplicativos para web como o Google maps, onde qualquer um pode por seu endereço ou de seu comercio de modo acessível a todos, fazendo com que todas esses projetos tivessem o objetivo do usuário ao mesmo tempo que utiliza o aplicativo acaba criando conteúdo, ou seja, alterando o que esta sendo visto da forma que mais lhe favorece, aumentando com isso o conteúdo disponível, e também a circulação de interessados nesse tipo de conteúdo.

Outros exemplos estão bastante diversificados em boa parte das aplicações necessárias na informática, desde aplicativos gráficos como o Blender, descompactadores como o 7-zip, pacotes para escritório como o Libre Office e o Open Office, e até mesmo linguagens de programação completas como o Ruby, Lua, Python, Perl, incluindo IDEs como o Netbeans e o Eclipse, enfim uma gama bastante completa de softwares para todos os tipos de interesses, o que vem a ajudar não somente o usuário que tem em mãos uma ferramenta funcional livre e acessível, como também ao estudante programador que tem acesso ao código fonte de vários tipos de ferramentas e soluções para programação, possibilitando expandir com isso o seu aprendizado.

5. INTERAÇÃO ENTRE EMPRESA PRIVADA E SOFTWARE LIVRE

As grandes empresas de produção de em meados dos anos 1990 atingiram um mercado globalizado e ficou cada vez mais claro que seu modo de produção não estava atendendo de uma maneira eficaz as necessidades de seus usuários ao redor do mundo, o modelo de produção em que apenas uma empresa e um grupo fechado de colaboradores tentam produzir um sistema operacional ou aplicativo que vá atuar no mundo todo recebeu certo grau de rejeição e demasiado numero de criticas, o que incentiva os usuários a buscar novas alternativas, o software livre é uma dessas alternativas.

> Em um e-mail direcionado a Bill Gates, Andrew Shapiro, membro do centro para internet e sociedade da faculdade de direito de Harvard, exprime uma opinião que certamente ocorreu a maioria dos observadores previdentes das fusões e dos esquemas de sinergia modernos “Se toda idéia dessa revolução é dotar as pessoas de poder, Bill, por que você esta trancando o mercado e restringindo as opções? Sinergizando seu caminho de um negócio para outro a cada mês? Essa contradição representa uma traição muito maior do que o costumeiro discurso duplo da publicidade que estamos acostumados à ouvir. O que

> esta sendo traído é nada menos que as promessas centrais da era da informação: as promessas de opções, interatividade e maior liberdade. (KLEIN, 2008, p. 247)

Com esse alto índice de reclamações de seus produtos na década de 1990, as grandes empresas produtoras de softwares proprietários começaram a reavaliar seu modo de produção, com isso tomaram emprestado certos métodos utilizados inicialmente na produção de softwares livres, tais métodos como a liberação de parte do código, aquisição de novas alterações e a realização de testes a ser feita por usuários ao redor do mundo todo, e não apenas dentro da empresa tornaram os novos produtos de empresas como a Microsoft, Google, Oracle dentre outras empresas mais bem aceitos pelo usuário final.

> Os novos instrumentos da tecnologia da informação deveriam servir prioritariamente para valorizar a cultura, as competências, os recursos e os projetos locais, para ajudar as pessoas a participar de coletivos de ajuda mútua, de grupos de aprendizagem cooperativa etc. em outras palavras, na perspectiva da cibercultura assim como nas abordagens mais clássicas, as políticas voluntaristas de luta contra a desigualdade e a exclusão devem visar o ganho em autonomia das pessoas ou grupos envolvidos, ao contrario de torná-las dependentes como era feito por algumas grandes empresas. Devem, em contrapartida, evitar o surgimento de novas dependências provocadas pelo consumo de informações ou de serviços de comunicação concebidos e produzidos em uma óptica puramente comercial ou imperial e que tem como efeito, muitas vezes, desqualificar os saberes e as competências tradicionais dos grupos sociais e das regiões desfavorecidas. (LEVY, 1999, p. 257)

Dessa forma para atender melhor aos vários e distantes mercados dependentes de informática ao redor do mundo, principalmente os países emergentes que estão consumindo uma grande demanda de tecnologia na busca por maior eficiência na realização de processos, trouxe a tona a necessidade não apenas de produzir softwares de forma fechada em suas matrizes, mas sim de buscar uma forma mais eficiente de atender tamanha demanda, como o movimento de software livre tem se mostrado bastante eficiente na evolução de idéias e no suprimento de bons softwares com uma evolução bastante rápida tanto de idéias novas quanto de soluções satisfatórias aos seus muitos usuários, as grandes empresas tem se inspirado em bons costumes gerados a partir do modo de produção de software livres.

As grandes empresas de tecnologia estão aderindo pelo menos em parte ao software livre por ver em seus métodos uma eficiência comprovada, não de modo que o software livre venha a substituir o software proprietário, mas sim a propor que a interação entre esses dois modos de produção se forem bem explorados pode trazer ótimos resultados para ambas as comunidades, como tem se mostrado no

investimento em software livre por grandes empresas como a IBM, Oracle, HP e Google as quais só vieram a evoluir ao agregar o resultado de seus investimentos em software livre ao seu produto final.

Segundo Petroski (1998) ao comentar sobre a evolução de objetos de nosso cotidiano como talheres, no passado, pessoas inteligentes que hoje poderíamos chamar de engenheiros observaram que algumas coisas já existentes não conseguiam funcionar tão bem como se poderia imaginar em principio. Ao se concentrar nas deficiências os inovadores alteraram esses objetos para remover as suas imperfeições, produzindo assim artefatos novos aperfeiçoados, diferentes inovadores em locais diversos trabalharam sobre um mesmo problema básico, se concentraram em diferentes falhas em momentos distintos e a partir disso, herdamos objetos que atendem nossas necessidades.

Isso vem a reforçar a idéia de que para criar um produto que venha a ser usado em grande parte do mundo, o processo de evolução exige contribuições de vários envolvidos, não apenas uma equipe em um determinado local isolado do globo, mas sim usuários que percebem uma deficiência no produto e com os recursos corretos podem incrementar melhorias por conta própria, a fim de criar um produto cada vez melhor, e em constante e rápida evolução, se isso se aplica a produtos simples, leva a crer que durante a criação de produtos complexos como um aplicativo ou um sistema operacional seja necessária a mesma contribuição de idéias visando buscar a excelência na criação do produto que chega ao usuário final.

III ESTUDO DE CASO

1 DESCRIÇÃO DO ESTUDO DE CASO

No decorrer deste estudo de caso será avaliado o modo de produção da distribuição do sistema operacional Linux Debian e como o mesmo esta sendo usado pelas escolas publicas para auxiliar no desenvolvimento da inclusão digital nas escolas publicas estaduais, o modo de produção Debian foi escolhido por ser um dos modos de produção mais eficientes, mantido em sua grande maioria por voluntários, e que consegue alcançar resultados bastante relevantes no mercado atualmente, exemplificando como o modo de produção do software livre pode vir a compor de forma eficaz parte do modo de produção dos softwares proprietários como vem sendo hoje em dia.

1.1 APLICAÇÃO DO ESTUDO DE CASO

A aplicação do estudo de caso se trata de apresentar como o Linux Debian com seu modo de produção ágil foi capaz de produzir um sistema robusto e funcional o qual auxilia na inclusão digital das escolas publicas estaduais com a distribuição Linux Educacional, a qual se trata de uma distribuição derivada do Linux Debian, com interface gráfica KDE e que traz uma suíte de aplicativos educacionais que se aplicam no cotidiano dos alunos.

O Linux Educacional se trata de uma distribuição desenvolvida com a parceria do Centro de Experimentação em Tecnologia Educacional (CETE) e do Ministério da Educação (MEC) e a Universidade Federal do Paraná (UFPR), sua versão mais atual se trata da versão 4.0, possui uma suíte bastante completa de aplicativos que visam estimular o aprendizado e acesso aos portais do MEC.

1.2 LINUX

Segundo Tanenbaum (1999), um sistema operacional se trata de um software que controla todos os recursos dos computadores, e fornece a base sobre a qual os programas e aplicativos são escritos, dentre várias outras funções de nível mais técnico, Richard Stallman, fundador da Free Software Foundation, que ao tentar fundar um projeto que reúne e dá condições para que os projetos de software

sejam liberados e estudados de forma livre pelo usuário, seria de vital importância a aquisição e programação de um sistema operacional de licença GPL.

1.3 DISTRIBUIÇÃO DEBIAN

Segundo Oliveira (2010), o Linux é um núcleo de sistema, mas os rumos da distribuição divergiram em certo ponto, dando origens a diversos tipos de distribuição Linux, a qual traz diversos aplicativos e programas de sistema, em sua grande maioria oriunda da colaboração de vários programadores ao redor do mundo que se identificaram com um projeto, e a partir disso decidiram colaborar com ele.

Com a evolução do Debian, se tornou mais claro que a imagem que havia se consolidado quanto ao fato de os sistemas de núcleo GNU/Linux serem manipulados e compreendidos apenas por hackers ou grandes peritos em informática esta em grande parte superada. Isso muito se deve ao surgimento, primeiramente, de interfaces gráficas livres amigáveis como por exemplo o KDE, o Gnome e a interface XFCE, baseada em uma das mais antigas interfaces chamada anteriormente de X, e em segundo lugar, ao aparecimento de diversas versões baseadas em distribuições Linux, as quais conseguem atingir facilidades para determinados grupos de usuários, popularizando cada vez mais software livre.

2. VISÃO GERAL DO SISTEMA

Para exemplificar melhor o modo de produção de softwares livre, foi efetuado o estudo do modo de produção da distribuição Linux Debian, a qual conta atualmente com milhares de desenvolvedores espalhados em todo o mundo, a comunidade se destaca por ter um grande numero de pessoas trabalhando ativamente no software. Além disso, faz-se notável sua organização bem definida, dotada de normas muito claras e precisas.

Figura 03: Logotipo da Comunidade Debian

Fonte: Comunidade Debian Brasil, 2012.

Oficialmente fundada em 16 de agosto de 1993 pelo estadunidense Ian Murdock, que juntou as três letras inicias do nome de sua mulher, Debra, com seu primeiro nome a fim de formar o título da distribuição. Inicialmente com apoio do projeto GNU e da Free Software Foundation a distribuição teve inicio contando com um pequeno numero de hackers e hoje conta com um grande numero de colaboradores espalhado em todo o mundo, e para estimular essas pessoas na elaboração de um sistema de alto padrão, o projeto Debian tem – além de documentos com políticas de organização, comunicação conduta, etc., bastante comuns em outras comunidades, alem de um modo de produção bastante eficiente, e que devido a sua organização e documentação bem estruturada vem alcançando diversos bons resultados e dando origem a outros projetos bem sucedidos como o Linux ubuntu, kurumin, etc.

Um dos pontos fortes da distribuição Debian é seu sistema de gestão de pacotes, chamado APT, o qual permite atualizações relativamente fáceis a partir de versões realmente antigas, facilitando muito a vida do usuário comum tornando a instalação de softwares dentro sua distribuição muito mais fáceis, pois se for necessário baixar um outro pacote complementar, isso é feito automaticamente exigindo apenas algumas confirmações do usuário e não mais que o usuário baixe uma determinada biblioteca que estava faltando e recomeçar a instalação do zero, tem se apresentado uma solução muito mais viável do que soluções de instalação de softwares apresentadas pela distribuição Red Hat que precisa que um pacote RPM que contem uma série enorme de pacotes que o poderão vir a ser necessários ou não, trazendo por efeito colateral um acúmulo de pacotes desnecessários dentro do

sistema. O utilitário de gerenciamento de pacotes APT baixa automaticamente e somente os softwares e bibliotecas necessários para a instalação do programa determinado pelo comando dentro do Shell.

Exemplos de uso do gerenciador de pacotes APT:

apt-get install squid = comando que determina que o gerenciador de pacotes APT busque dentro de seus repositórios na internet os pacotes necessários para a instalação do software Squid.

O Debian se destaca pela sua ampla documentação e apoio de seus colaboradores facilitando a entrada de novos usuários que buscam aderir ao software livre de alguma forma, tendo o ciclo de desenvolvimento de suas versões marcado por três fases, as versões "unstable", ainda instáveis, "testing", versões lançadas para que os usuários interessados da comunidade possam baixar e testar as inovações da nova versão, procedimento parecido com o que algumas empresas adotam hoje antes de lançar uma nova versão de seu software, e as versões "stable", versões estáveis que já foram testadas e possuem um grau de confiabilidade maior, são as versões oficiais lançadas no site da comunidade, na tabela abaixo estão listadas os codinomes e respectivas datas do lançamentos das versões.

Versões	Codinomes	Datas em que foram lançadas como "Stable"
6.0	Squeeze	6 de fevereiro de 2011
5.0	Lenny	15 de fevereiro de 2009
4.0	Etch	8 de abril de 2007
3.1	Sarge	6 de julho de 2005
3.0	Woody	19 de julho de 2002
2.2	Potato	15 de agosto de 2000
2.1	Slink	9 de março de 1999
2.0	Hamm	24 de julho de 1997
1.3	Bo	2 de junho de 1997
1.2	Rex	1996
1.1	Buzz	1996

Tabela 02: Versões, codinomes e datas em que se tornaram estáveis.

Fonte: Comunidade Debian Brasil, 2012.

.

O líder oficial do projeto Linux Debian foi passando primeiramente de seu fundador Ian Murdock para os lideres posteriores com o tempo para que os mesmos efetuassem uma melhor gerencia do projeto de acordo com seu tempo de dedicação visando atingir melhores resultados, o atual líder do projeto da comunidade Debian é Stefano Zacchiroli que tomou posse da liderança do projeto em abril de 2010.

No Brasil, a comunidade Debian se faz representativa não tanto pelo numero de desenvolvedores ou de pessoas que trabalham mais ativamente no software final, tal numero, aliás, é considerado baixo quando comparado à quantidade de desenvolvedores europeus e norte-americanos. No entanto, o numero de entusiastas, evangelizadores e outros vários colaboradores elevam a comunidade Debian brasileira a um posto muito significativo no mundo do software livre, incluindo versões como o Kalango Linux, Kurumin, Knoppix, Ubuntu os quais tinham uma comunidade bastante forte no Brasil e sendo todos baseados no Linux Debian.

2.1 CICLO DE FUNCIONAMENTO DO MODO DE PRODUÇÃO

Segundo Maxwell (2000), uma das razões para o Linux ser grande se trata do fato de ter sido desenvolvido por muitas mentes, com Linus Torvalds dando o pontapé inicial, e seu modo de produção sendo aderido por várias partes do mundo, dando origem ao sistema operacional de maior evolução da era da informática e ao modo de produção dos softwares livres de uma forma geral.

Nesta etapa será descrito o ciclo de funcionamento interno do modo de produção da distribuição Linux Debian, mostrando como a colaboração não somente dos programadores mas também dos usuário geram as feedbacks necessários e produzem um resultado bastante eficiente, e como é possível participar e melhorar o software livre através de seu modo de produção que esta sendo cada vez mais copiado em parte pelas demais empresas de tecnologia que compõem o cenário mundial de produção de tecnologia, software dentre outros produtos.

Ao contrário do que é imaginado à primeira vista, o universo dos softwares livres está longe de ser homogêneo. Em torno de um objeto em comum – uma determinada versão do software livre – encontram-se vários colaboradores que considerando graus de afinidade ou determinado ideário, dispõem-se em comunidades especificas e nessas tais comunidades, embora muito parecidas em diversos pontos, detêm diversos atributos que nelas são muito próprios, e dentre todas as distribuições

Linux, a que melhor expressa o modo de produção do software livre é a distribuição Debian, vindo por meio dela, exemplificar esse tipo de modo de produção o qual tem atingido resultados eficientes e servido de exemplo para melhorar os resultados da produção de software mundial em diferentes seguimentos.

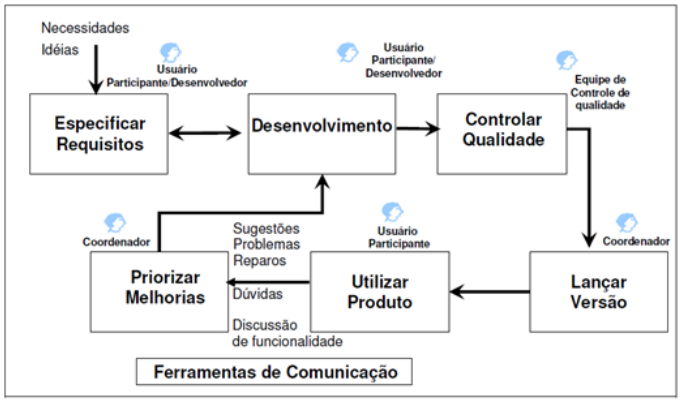

Figura 04: Visão Geral do Sistema.

Fonte: Softwarelivre.Org. 2012.

Dentre os aspectos comuns das distribuições Linux, vale ressaltar o fato de todas se valerem de um modelo aberto e colaborativo, dinâmico e flexível, calcado na espontaneidade e na voluntariedade. Fazendo-se presente a cultura meritocrática e a produção entre pares, elementos-chave da cultura hacker. Os muitos voluntários motivados por uma busca pelo conhecimento, pela possibilidade de se relacionar e pela vontade de serem reconhecidos em função de seus feitos, expressam seus desejos de melhoria da comunidade e se emprenham em tarefas nas quais sejam mais hábeis e que mais lhes dêem prazer, o que traz a tendência de gerar um resultado muito mais eficiente do que o padrão utilizado na produção dos softwares proprietários.

Segundo Hunger (2009), as distribuições Linux Slackware e Debian são as que melhor exemplificam o modo de produção de software livre, nas quais pesam apenas a força das intensas atividades de seus voluntários.

O modo de produção do software livre Linux Debian é um dos casos em que sem a participação de nenhuma empresa em tempo integral, vem atingindo um bom resultado na criação de um sistema operacional bastante útil, o qual esta tendo uma demanda bastante grande, abrangendo uma grande parcela dos usuários, e também dando origem a versões Linux derivadas do Debian as quais são bastante famosas, o que vem somente a provar que o seu modo de produção é bastante eficiente tanto

em produzir inovações, quanto em produzir um software o qual pode ser trabalhado de forma eficiente, graças a sua documentação bastante completa.

3. PADRONIZAÇÃO LINUX

Nesta etapa serão mostrados os padrões que devem ser seguidos pelos programadores de versões Linux que almejam criar um software que funcione de acordo com a padronização internacional, e que futuramente possa ser implementado sem medo. O que atrairá um maior número de colaboradores e conseqüentemente de usuários.

3.1 LSB

O Free Standard Group é uma organização sem fins lucrativos que promove os padrões de código-fonte aberto, em particular, a iniciativa de padronização do Linux. Hoje em dia várias empresas de diferentes ramos vêm colaborando com essa iniciativa, como por exemplo, HP, IBM, Novell (Suse), Debian, Mandriva, Red Hat, etc. com o objetivo de tornar mais fácil para os desenvolvedores de software independentes a criação de um produto de software para o Linux, criando uma base de padronização sobre a qual pode ser desenvolvida. Os itens canalizados por meio do processo de padronização incluem muitos aspectos.

De acordo com Nemeth (2009), os padrões tratam das interfaces de bibliotecas compartilhadas comuns, arquivos de configuração, comandos de sistema, um formato comum de pacote e das diretrizes de instalação, uma API para interfaces de sistema, padrões de hierarquia dos sistemas de arquivos e alguns aplicativos que devem estar contidos dentro da distribuição escolhida.

Graças a esse tipo de padronização é possível nivelar os softwares trabalhados dentro do Linux, sendo possível criar um programa que rode tanto em uma distribuição Linux, quanto na outra, por isso é possível ver o Browser Mozilla Firefox em várias distribuições Linux.

4. ETAPAS DA PRODUÇÃO

Para produzir um produto de qualidade é necessário um modo de produção bastante eficiente e com regras e hierarquia bem estabelecidas, regras essas que vem sendo aperfeiçoadas com o passar do tempo, e que vem mostrando uma evolução e adaptação bastante efetiva e alcançando resultados bastante satisfatórios, as etapas da produção são bem organizadas e rígidas e vem gerando um bom resultado.

Com um modo de produção bem documentado, como na comunidade Debian pode ser encontrado a documentação de grande parte dos softwares produzidos para essa distribuição e suas mudanças, com tarefas bem distribuídas, e efetuado os respectivos testes de forma bastante eficiente, até mesmo durante a analise de requisitos, já que a idéia de um determinado software vem da percepção de um usuário da comunidade que percebeu determinada necessidade de software durante a utilização do mesmo no seu dia-a-dia.

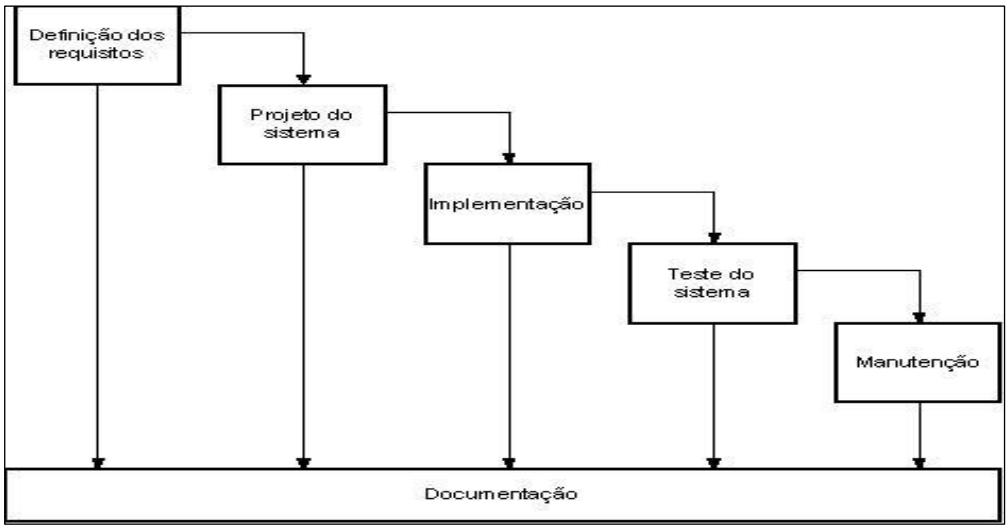

Figura 05: Modelo em cascata.

Fonte: Voat. 2012.

O modo de produção do software livre em grande parte atende aos requisitos dos modelos em cascata, e também do modelo em V, que se trata de um modelo bastante utilizado nas empresas hoje em dia e que envolve o teste desde o momento da analise de requisitos, até o final de seu projeto, sendo testado de forma bastante freqüente, e se assemelha bastante ao modo de produção do software livre e o lançamento de suas releases para que as suas funcionalidades sejam analisadas com bastante freqüência pelos demais usuários que baixarem o programa e se dispuserem a avaliar suas funcionalidade, fazendo com que o modo de produção do software livre atenda de forma bastante eficiente os requisitos tanto do modo de produção em cascata com sua documentação bem feita como do modo de produção em V e seus testes feitos com bastante freqüência, gerando resultados bastante satisfatórios, tanto do ponto de vista do usuário final quanto das empresas que tem se juntado a comunidade produção do software livre.

O modelo de produção em V determina testes exaustivos ao aplicativo desde o momento de sua analise de requisitos visando o teste de aceitação, e como se trata de uma idéia que muitas vezes parte de necessidades de usuários percebidas em seu uso cotidiano do sistema, já tem uma grande possibilidade de atender eficientemente ao teste de aceitação dos demais usuários.

Como se trata de um software que vai ser disponibilizado para que outros programadores ao redor do mundo possam analisar a sua estruturação normalmente é bem realizada, tornando de fácil entendimento para que possa ser realizada a documentação e futuras avaliações visando facilitar os testes futuros do sistema, durante a própria codificação o desenvolvedor responsável vai testar as partes unitárias do software, para ter certeza que o mesmo esteja funcionando corretamente. E ao ser liberado uma release para avaliação pela comunidade os diferentes tipos de participantes da comunidade tendem a avaliar o funcionamento do software em diferentes situações e com os mais variados tipos de hardware atendendo ao teste de integração e de sistema de forma bastante eficiente.

No Debian, nada do que não esteja completamente finalizado, estável e funcionando perfeitamente bem pode entrar no sistema final disponibilizado aos usuários – e isso, muitas vezes implica em abri mão de uma versão mais atual para dar lugar a algo mais antigo, porém mais estável, com o objetivo de não comprometer o resultado final.

4.1 ENTRANDO NA COMUNIDADE

Muitos usuários iniciantes no mundo do software livre costumeiramente crêem que, para ajudar determinada distribuição ou comunidade, é necessário fazer parte ativamente dos projetos, estar em constante contato com seus líderes, angariar várias responsabilidades e realizar trabalhos regularmente. Por certo isso tudo se faz de grande relevância, principalmente quando se almeja alcançar cargos de maior relevância no projeto Debian, mas as comunidades deixam claro que uma atitude colaborativa exige muito menos; a principio, a melhor forma de ajudar é usando a distribuição e testando o sistema, o que é amplamente salientado nos sites de vários projetos, podendo ainda participar do projeto ajudando a escrever a documentação, traduzi-la para mais de um idioma, etc.

Para se cadastrar na comunidade Debian é preciso apenas se cadastrar em um dos sites da comunidade “http://wiki.debianbrasil.org/”, e adquirir o sistema em um de seus muitos servidores de download, podendo também adquirir os códigos fontes por meio destes servidores, seja por meio de download direto, download via torrent, e diversos mirrors espalhados por todo o mundo.

Como é mostrada na figura abaixo, a comunidade Debian conta com colaboradores em diversas partes do mundo, em sua maioria na Europa e Estados Unidos, e possui uma colaboração cada vez mais crescente no Brasil.

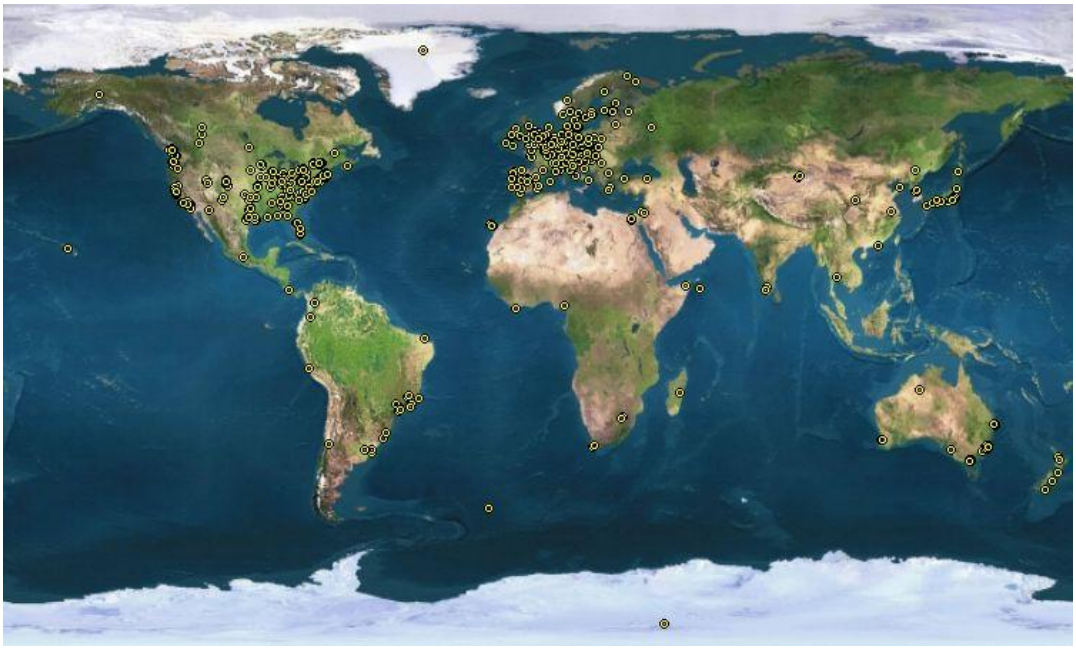

Figura 06: Comunidade de Desenvolvedores Debian.

Fonte: Debian.org. 2012.

Muitos colaboradores estão envolvidos praticamente durante todo o tempo com suas distribuições. E, de um modo geral, observa-se que a quantidade de tempo despendida é muito variável entre os membros, e isso se dá, entre outras razões, pelas diferentes posturas que se assumem, ou seja, pelos diferentes níveis de comprometimento – e nesse aspecto, também se notaram muitas diferenças, desde usuários que não fizeram questão de repassar as respostas que obtiveram em fóruns até colaboradores que passaram horas respondendo a e-mails pessoais com dúvidas de usuários.

4.2 AQUISIÇÃO E ALTERAÇÃO DO CÓDIGO FONTE

O código-fonte de diversos softwares livres utilizados tanto na comunidade Debian quanto nas mais diversificadas distribuições podem ser adquiridos nos mais diferentes sites onde possuem mirrors que disponibilizam para download tanto o programa quanto seu código-fonte normalmente disponíveis no formato compactado tar.gz ou tar.bz, o qual após ser descompactado com o seguinte comando:

tar -xzvf exemplo.tar.gz

Podendo conforme a necessidade do usuário vir a ser alterado e remontado no código com os comandos Linux ./configure, seguido do comando make ou make install, e finalmente sendo conferido se a instalação foi bem sucedida com o comando checkinstall.

4.3 FUNÇÕES E HIERARQUIA

Uma distribuição nada mais é do que um agrupamento de desenvolvedores voluntários ou não que concordaram em implementar melhorias para determinado projeto, sendo composto não somente por programadores, mas possui a participação de analistas de teste, suporte, documentadores, tradutores, etc. Tais agrupamentos dizem muito a respeito da comunidade a qual pertencem, pois sempre atraem pessoas que estão de acordo com a ideologia da distribuição, e geralmente coincidem em objetivos.

Os preceitos de liberdade apregoados pelos membros das mais variadas comunidades de software livre não se traduzem, de maneira alguma, em uma forma irrestritamente anárquica de organização. De maneira geral, as comunidades estão envolvidas em relações de poder e fundamentadas na cultura meritocrática e a liderança que nelas se estabelece estão sujeita a avaliações pessoais por parte dos colaboradores – afinal, estamos falando de voluntários. Caso um líder tome uma atitude que nçao corresponde aos anseios e vontades da comunidade, os colaboradores podem resolver contribuir com outro projeto ou não mais contribuir.

A estrutura geral consiste em Oficiais (se tratando dos lideres, comitê técnico e secretários), colaboradores ligados à distribuição (relacionados a pacotes individuais, repositórios FTP para downloads dos pacotes, gerencia dos lançamentos das versões, também chamados de releases, documentação etc.), à publicidade (imprensa, eventos, parcerias, marketing, blogs e redes sociais, etc.)

e à infra estrutura (suporte a idiomas, traduções, acompanhamento dos bugs, suporte, equipe de segurança, sites da web que possuem fóruns, FAQs, etc.), e Distribuições personalizadas (Debian Jr para crianças, Debian Lex para escritórios legais, Debian-Med, para pesquisa médica e Debian Acessibility, para pessoas com deficiência).

Quanto ao exercício do poder da comunidade, é estabelecida a seguinte hierarquia, o comitê técnico e seu presidente, geralmente apontado por meio de uma resolução geral ou votação, os desenvolvedores, os líderes dos projetos, desenvolvedores individuais que trabalham em uma tarefa particular, delegados apontados pelos líderes para atender de tarefas especificas e o secretário o qual é responsável entre outras coisas, por recolher os votos entre os desenvolvedores e colaboradores, e resolver disputas, dessa forma evidenciando uma estrutura fundamentada não apenas na meritocracia, mas na pisticracia (Raymond,2007), isto é, na relação de confiança entre os pares.

Com a hierarquia baseada na meritocracia, mostra que para que o usuário tenha cargo de maior responsabilidade na comunidade ele deve ter um grande envolvimento nos trabalhos para conseguir atingir determinados postos, trazendo com isso uma maior valorização dos mesmos, aumentando o interesse nos mesmos, e trazendo cada vez melhores resultados para o aperfeiçoamento da distribuição.

Os conselhos são compostos, evidentemente, apenas por colaboradores. No caso do Debian, existe uma estrutura de organização muito precisa, o projeto apesar de ser presidido por um comitê técnico, secretários e delegados, mas o grau máximo de poder esta nas mãos dos desenvolvedores, que podem exercê-lo por meio de uma Resolução Geral ou pelas próprias eleições internas.

Em nível inferior ao dos conselhos, existem os líderes específicos de cada subprojeto, que coordenam aquilo que se poderiam classificar como cada célula das atividades básicas da tradução, documentação, grupos regionais, divulgação, etc. A esses líderes cabe, alem de manter regularmente a realização dos trabalhos, assegurar a atividade das listas, cumprir os prazos determinados (isto é, quando o projeto se compromete a estipular datas para a disponibilizar uma nova versão como é o caso do Linux Ubuntu, por exemplo.) e com isso incentivar novos colaboradores.

Na maioria dos casos, esses lideres, sejam os que estão em níveis máximo ou os que tem atividades mais locais, chegaram a seus atuais postos em função das próprias competências, o atual

líder do projeto da comunidade Debian é Stefano Zacchiroli que tomou posse da liderança do projeto em abril de 2010,

Se determinado líder não conseguir desempenhar suas funções da melhor maneira possível (seja por falta de tempo, por excesso de trabalho ou por perda de identificação com o trabalho proposto), o individuo deve passar seu cargo aos cuidados de quem efetivamente possa fazê-lo com o devido interesse e a devida dedicação, o nome do líder do projeto ou responsável por determinada parte do projeto esta contido no rodapé da pagina web da comunidade onde determinada tarefa é solicitada para os colaboradores que tiverem interesse.

.

4.4 ANÁLISE DE REQUISITOS

O coordenador na maioria das vezes é responsável por selecionar e priorizar as melhorias sugeridas pelos usuários e decidir quais serão inseridas no próximo release, para medir a aceitação dos demais usuários e da comunidade em geral, e que possa também ser analisado e marcado seus bugs a serem corrigidos. Dentro da área de Engenharia de Software a obtenção de requisitos é uma subárea chamada de engenharia de requisitos, nela são estudados os processos de definição e construção dos requisitos do software. Nos softwares comerciais geralmente existe uma preocupação intensa na especificação formal dos requisitos, isto é, de que o software atenda as necessidades e os desejos explícitos do cliente. O que no caso do software livre se torna um pouco difícil, com o software partindo de uma iniciativa muitas vezes pessoal outras vezes partindo de uma observação de uma necessidade, o que diferencia o processo de analise de requisitos dentro do desenvolvimento de um software livre, mas em grande parte dos projetos open source os requisitos são iniciados pelo próprio autor que observa determinada necessidade no mercado ou para o usuário domestico que poderá vir a aderir esse conceito com o tempo.

Com os requisitos de software podem se originar de seus desenvolvedores: esta forma de obtenção de requisitos é a mais comum nos projetos open source. Como o desenvolvedor é também um usuário de seu sistema, os requisitos começam inicialmente por atender as suas necessidades, para que depois seja lançado o release, e por meio de interação com os outros usuários possa alem disso atender as necessidades exigidas por outros usuários que participarem da comunidade ou solicitarem por meio de algum dos meios de comunicação.

Os requisitos podem originar-se dos usuários do software: à medida que os demais usuários vão utilizando-o, eles vão dando um retorno, feedback, aos desenvolvedores, propondo mudanças e melhorias no software como um todo. Obviamente quanto maior a quantidade de usuários ativos, maior será o retorno obtido, desse ponto mostra-se a importância da obtenção de cada vez mais usuários interessados no desenvolvimento do software livre. Segundo Raymond (1999), em seu ensaio bastante famoso na comunidade de software livre chamado de “catedral e o Bazar”, usuários são ótimas coisas para se ter e não somente porque eles demonstram que você esta satisfazendo uma necessidade, mas sim que você criou algo de maneira correta e ainda por cima pode evoluir seu software de maneira rápida e eficaz, com isso os usuários quando cultivados de maneira adequada, podem também se tornar co-desenvolvedores através de suas idéias e participação voluntaria na comunidade mesmo sem saber programar.

A implementação de padrões estabelecidos pode gerar também requisitos a partir da adaptação do software aos requisitos algumas vezes exigidos legalmente para a boa utilização do software, como aplicativos de nota fiscal eletrônica, ou esses padrões podem seguir recursos observados em aplicativos que já estão no mercado e com seu determinado formato já atingindo um bom resultado, levando seus desenvolvedores a adaptar requisitos antes observados em programas já existentes.

Os requisitos podem originar-se da implementação de padrões de mercado, seguindo a metodologia de empresas de mercado, os requisitos do sistema tendem a seguir alguns padrões de implementação de software comerciais, como aplicativos de escritório, ERPs, SGBDs, etc.

Podendo surgir principalmente da necessidade de construir protótipos para fins de aprendizado tanto dos desenvolvedores quanto dos usuários, e a partir disso criar uma opção nova e eficiente que possa vir a fazer parte dos padrões futuros, um exemplo disso são os recursos de aba que surgiram no browser Mozilla Firefox, e passaram a fazer parte de boa parte dos browsers que estão no mercado atualmente.

Sendo do acúmulo constante do maior numero de usuários possíveis a cada comunidade faz com que esse número quanto maior seja vem a influenciar na força do resultado final, tanto a aceitação no mercado como também a interação não apenas demonstre o sucesso do software livre mas também moldam os rumos futuros da distribuição, o numero de usuários participantes da comunidade influenciam não somente na analise de requisitos mas sim em todas as fases da produção do sistema,

graças a contribuição do nicho de seguidores da comunidade que sugerem e determinam boa parte das melhorias a serem implementadas conforme as necessidades daquela determinada quantidade de usuários.

4.5 DESENVOLVIMENTO

A fase de desenvolvimento compreende a codificação do projeto. Nesta fase características importantes são incorporadas ao código, de acordo com algumas decisões propostas pelos dirigentes da comunidade, de modo a alcançar os determinados objetivos visados, fazendo parte de todas as demais etapas do projeto, tanto para corrigir os bugs apontados durante a analise de qualidade ou sugeridos pelos usuários mesmo após o lançamento da release, e feitos assim o refatoramento do código. A fase de desenvolvimento ocorre de maneira pouco formal e o tempo de desenvolvimento pode variar de projeto para projeto, sempre tentando respeitar os tempos determinados para o lançamento das releases, propostos pela política de qualidade Debian.

Logo após ou até mesmo juntamente com o processo de analise dos requisitos já se inicia o processo de desenvolvimento do software, o qual visa a programação do que o software deve fazer e de que maneira esse software vai fazê-lo, sempre tentando realizar a programação da forma mais organizada e bem planejada o possível, para que quando alguma nova qualidade for inserida mesmo que por outro programador essa transição possa acontecer de maneira fácil para o colaborador.

Os desenvolvedores alem de programar da melhor maneira possível o software a ser lançado junto com a distribuição sempre que possível devem tentar documentar o que esta sendo feito e as alterações que estão acontecendo, sendo assim a etapa de desenvolvimento, uma parte central para o lançamento de alguma distribuição, já que tudo gira em torno desse processo, desde a etapa de planejamento ate a parte de documentação precisa de uma ajuda muito grande do programador que esta desenvolvendo aquela determinada parte do software.

4.6 GERENCIAMENTO DE RELEASES PARA TESTES

Releases são os lançamentos das versões dentro do mercado ou no caso dentro da comunidade de desenvolvimento por meio da internet para que a mesma possa ser avaliada, testada e

reportando e documentando os defeitos nela encontradas, ou seja, permitindo que os demais membros da comunidade possam trabalhar sobre o código feito por determinado desenvolvedor.

> Para tratar os usuários como co-desenvolvedores na maneira mais eficaz possível é preciso liberar releases com certa freqüência, e ouvir as sugestões de seus fregueses, no caso os usuários, liberando o código cedo e freqüentemente. À Linus Torvalds por volta de 1991 não era estranho liberar uma nova release do kernel mais de uma vez no mesmo dia na internet para utilização e avaliação de seus colaboradores. (RAYMOND, 1999, p. 12)

Isso vem a reforçar a idéia de que para uma distribuição ser bem sucedida o lançamento de releases tem que ser freqüente, na comunidade Debian é lançada uma versão denominada estável, sendo que a ultima versão estável lançada pela comunidade é a Debian GNU/Linux 6.0.6 a qual esta disponibilizada no próprio site da comunidade, podendo ser selecionada por seu tipo de processador, em que tipo de arquitetura roda, etc. e no mesmo repositório estão disponíveis também as versões testing (para testar) e unstable (já denominada instável) as quais servem para que os demais membros da comunidade e os interessados possam analisar se existe algum erro e reportar o mesmo para o suporte da comunidade da forma mais rápida possível.

> O gerenciamento das releases é o processo de planejar, compilar, testar e implantar uma versão de distribuição de hardware ou de software, bem como o controle de versionamento e a sua armazenagem, e isso mostra que a forma mais comum de lançamento de versões nos projetos de software livre é o “congelamento” do código, criando uma versão oficial e outras versões para teste, as quais não foram testadas como o software lançado oficialmente, dando ao usuário a opção de fazer o download da versão oficial ou de um release instável para efetuar os devidos testes na versão escolhida. (BECTA, 2004, p. 431)

À partir do congelamento da versão, nenhuma funcionalidade é adicionada ao código base, apenas os bugs são corrigidos para o lançamento de uma versão estável, os erros encontrados nas versões “unstable” e “testing” são reportados aos desenvolvedores de forma que possam ser corrigidos pelos desenvolvedores.

Visando sempre que o software à ser lançado deve sempre apresentar funcionalidades significativas para que possa atrair a atenção dos voluntários, a distribuição deve ser lançada no momento certo, e com uma determinada freqüência de modo a estimular o interesse do colaborador da comunidade. Ficando também a cargo do gerenciador de releases o empacotamento da release a ser lançada e o congelamento da versão mais funcional o possível do sistema de acordo com os outros

membros da comunidade, e a liberação freqüente das releases visando alcançar os melhores resultados possíveis.

Um teste é uma verificação feita sobre um código ou fragmento de código para garantir que uma determinada entrada produza sempre, uma saída esperada, esse teste pode ser efetuado pelos membros da comunidade mesmo os que não tem experiência com desenvolvimento de software, pois a própria utilização do software liberado para testes pode ser vista como um teste, esse tipo de procedimento vem a aumentar de forma bastante eficiente a engenharia de software da comunidade debian, lembrando em muito o modelo de desenvolvimento em V, o qual inclui os testes desde o primeiro momento até a implementação e liberação oficial do projeto, esse tipo de procedimento vem sendo adotado até mesmo pela Microsoft a qual liberou para downloads a versão para testes de seu Windows 8, permitindo que o mesmo fosse avaliado por vários olhos ao redor do mundo o que vem a produzir um feedback bastante eficiente sobre os diferentes tipos de bugs a serem encontrados graças a imensa gama de usuários interessados no projeto.

Não existe ninguém melhor do que o usuário para encontrar um bug no sistema, já que o mesmo vai por o sistema a prova dentro das mais diversificadas realidades, fazendo com que se houver um erro no sistema, o mesmo vai aparecer de forma mais rápida em meio a realidade do usuário, tarefa que as vezes se torna muitas vezes intangível para o testador de uma empresa de software proprietário.

O próprio fato da versão mesmo instável já ser liberada para o usuário desde que seja devidamente avisado que não se trata de um release oficial, pode ser visto como o processo de testes de software, já que os usuários da comunidade mesmo sendo leigos poderão auxiliar com sugestões de melhoria nos aplicativos, e os mais entendidos de programação poderão até mesmo contribuir com ideias e até mesmo alterar o código da forma que acharem que trará melhor resultado.

4.7 COMUNICAÇÃO ENTRE OS ENVOLVIDOS

Devido ao fato de que os desenvolvedores de projetos de software livre dificilmente se encontram pessoalmente, uma série de ferramentas de comunicação fica responsável pela comunicação dentre elas estão aplicativos como listas de discussão, fóruns, blogs, e-mails, zines, etc.

Lista de discussão (mailing lists): trata-se de um mecanismo que roda na plataforma de correio eletrônico comum. Os participantes enviam mensagens uns aos outros contendo dúvidas, opiniões, esclarecimentos. É o modo mais simples de comunicação, e também o mais usado dentre boa

parte da comunidade, já que exige apenas um endereço de e-mail, de modo geral, existem listas de discussão para Ada subprojeto (documentação, tradução, desenvolvimento, zines, etc.). Tais listas são, na maior parte dos casos, abertas ao publico, várias outras comunidades utilizam esse tipo de comunicação através das listas de discussão.

Fóruns: Ambiente mais propício à solução de duvidas e troca de informação, bastante acessível a todos, fácil de pesquisar e procurar informações dentro deste meio de comunicação, as mensagens não circulam no ambiente de correio eletrônico, mas por meio de postagens nos sites. Recomenda-se que, antes de lançar qualquer pergunta em um fórum, o usuário preferencialmente pode fazer uma busca a fim de verificar se a questão já foi esclarecida. Nenhuma comunidade prescinde dos fóruns.

De acordo com Pisani (2010), o IRC (Internet Relay Chat): é um protocolo de comunicação que já foi bastante usado na internet, tanto para conversas quanto para troca de arquivos, permitindo a conversa em grupo ou privada. O IRC foi escrito por um finlandês chamado Jarkko Oikarinen em 1988 e com a abertura da internet para o grande publico nos anos 1990 grandes redes de IRC permitiam que qualquer um que fosse assinante de um provedor de acesso pudesse se conectar a tais redes, chegando a se tornar o principal meio de bate-papo na internet no final dos anos 1990.

Cada servidor possui canais, isto é, salas em que se pode conversar abertamente, ou em particular, várias comunidades Linux mantém canais ativos no Freenode. Como mostra a figura abaixo de uma conversa n IRC da comunidade Ubuntu.

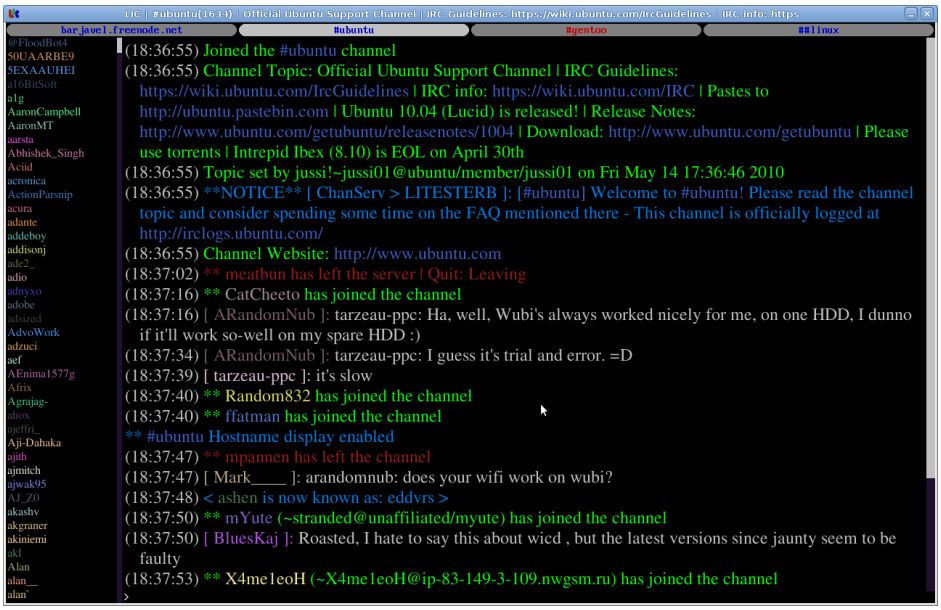

Figura 07: Screen de uma conversa na comunidade Ubuntu

Fonte: Ubuntuforums. 2012.

Mesmo com a ascensão dos mensageiros instantâneos ainda é possível encontrar o protocolo IRC ainda com um grande numero de usuários no Quakenet, IRCnet, Freenode e também no Linux Ubuntu, baseado em Debian presta suporte aos seus usuários por um cliente IRC que já vem nativo em seu sistema.

Blogs (Web Logs): Blogs com dicas sobre o sistema operacional, e com tutoriais sobre soluções de problemas estão presentes em boa parte da internet, possuindo o poder de levar respostas e comentários aos usuários comuns, facilitando seu acesso e especialização no sistema operacional de forma rápida de eficiente, os usuários mais representativos comunicam seus feitos ou os rumos tomados pela distribuição em seus blogs pessoais, os quais muitas vezes são hospedados nos servidores da própria distribuição, criando com isso um atalho simplificado entre os colaboradores da comunidade.

E-mail: Simples e-mails são capazes de simplificar a comunicação entre os desenvolvedores da comunidade, agilizando com isso o processo de troca de informações, sendo até mesmo recomendado que se tire duvidas em blogs, fóruns etc. para que mais usuários tenham acesso as respostas, mas em alguns casos é possível que se tire duvidas por meio de e-mail pessoal, agilizando com isso a comunicação entre os colaboradores.

Zines: São espécies de revistas eletrônicas dedicadas a comunidade interna, isto é realizado pelos próprios membros da comunidade, contendo na maioria dos casos artigos técnicos com resolução de problemas ou contendo, na maioria dos casos, artigos técnicos com resolução de problemas ou dicas para maximizar certas características do sistema. As comunidades brasileiras Slackware e Debian possuem sua própria zine assim como outras comunidades como o Fedora Linux, contando com noticias, novidades, tutoriais, textos explicativos, referencias à comunidade, informações sobre jogos, perguntas e respostas, etc.

FAQs (frequently asked questions, perguntas feitas frequentemente): São parte do projeto de manter uma certa lista de perguntas feitas mais frequentemente pelo usuário da distribuição, já que com isso se traz a ideia de que se foi a questão foi levantada por mais de um usuário, é bastante provável que mais usuários tenham o mesmo tipo de duvida, e que possam achar a resposta em sua pagina de FAQs.

Os mecanismos citados acima tratam da comunicação interna, isto é, realizada pelos próprios membros da comunidade. Contudo, nada impede que alguém que não faça parte das atividades de sua distribuição participe ativa ou passivamente, uma vez que os fóruns, as listas de discussões, os blogs e as zines são abertos à leitura de qualquer internauta. Dessa maneira, pode-se dizer que esses meios também se configuram em um canal de comunicação tanto interno quanto externo, para facilitar o esclarecimento de duvidas dos usuários iniciantes e até mesmo dos usuários que possuem uma maior experiência.

4.8 ANÁLISE DA QUALIDADE

A função da equipe da qualidade é identificar problemas no projeto em um âmbito geral. Dentre as atividades que se destacam estão a revisão e auditoria do código-fonte (Peer review), onde os revisores do código juntamente com a pessoa responsável por registrar os resultados da auditoria do código, sem envolver diretamente a participação do autor. Os testes feitos durante a analise de qualidade tem que ocorrer de uma forma freqüente e tem como resultado um produto de qualidade. O conceito de qualidade e um pouco difícil de definir.

> O número de definições de qualidade pode ser tão extenso quanto o número de pessoas que existem para defini-la, mas de uma maneira geral a qualidade pode ser definida por

> satisfazer a necessidade do cliente, funcionar corretamente dentro da função proposta e se possível tem por objetivo superar as expectativas dos clientes os quais fazem uso de determinado software. (LUCINDA, 1999. p. 134)

Como um controle de qualidade é efetuado também testes automatizados feitos pelos desenvolvedores. Além disso, é preciso existir uma geração de relatórios formais dos testes feitos pelos desenvolvedores. Um processo de desenvolvimento de qualidade freqüentemente tem como resultado um produto de qualidade que atenda suas funções propostas inicialmente da forma mais rápida e eficiente o possível, suas funções sejam executadas de acordo com seus requisitos explícitos e implícitos e que o código-fonte esteja legível, bem organizado, estruturado de uma forma correta e bem documentado.

Ficam a cargo do coordenador analista de qualidade algumas atividades auxiliares as quais contribuem para com a qualidade do produto final, dentre elas destacam-se o estabelecimento de datas para o lançamento da distribuição, através de um cronograma por ele estabelecido conforme o andamento das contribuições para com a versão Debian, ficando a cargo dele também a análise da documentação e pedido de contribuição com determinada parte da distribuição onde ninguém esta trabalhando em determinado momento, isso normalmente é feito no próprio site da comunidade onde possui uma pagina onde é disponibilizado os links das partes do projeto onde falta ajuda para o término de determinada tarefa, ou ate mesmo de alguma tradução ou atualização de documentações de partes do projeto Debian, assim garantindo uma maior qualidade no produto final e com isso a satisfação do usuário.

4.9 SUPORTE E MANUTENÇÃO

É ponto comum que o modo de produção da maioria dos softwares livres prefira deixar que a iniciativa de querer participar venha do usuário e, após saber do interesse do indivíduo, fornece todo o apoio necessário para tirar suas dúvidas e direcioná-lo à área de seu interesse, respondendo suas dúvidas nos fóruns, mantendo sempre disponível uma pagina com a documentação, e ainda disponibilizando uma pagina com as FAQs para auxiliar da maneira mais rápida o usuário em suas dúvidas.

A fase de manutenção é essencial para o projeto de software, pois caracteriza um processo constante de alterações. As alterações originalmente são sugeridas pelos usuários do sistema através do

informe de bugs e sugestões de melhorias no projeto, para dessa forma atingir um maior contentamento por parte dos usuários, devido a facilidade para que se tenha uma comunicação constante na comunicação entre as partes, entre o cliente e a comunidade desenvolvedora, alcançando com isso idéias novas de acordo com a necessidade do usuário para o qual o produto final é direcionado desde o inicio do projeto, e alcançando resultados os quais uma empresa de software proprietário levaria anos para alcançar, com idéias inovadoras e bons resultados graças a participação descentralizada dos usuários e da comunidade.

As sugestões e avisos de bugs são avaliados posteriormente pelos membros da comunidade, e a solução proposta é examinada pelos demais membros da equipe de coordenação, a qual seria a equipe fixa e os colaboradores ativos do projeto, para que todos possam avaliar e aprovar a solução submetida para que posteriormente ela possa ser implementada no projeto.

O retorno dado pelos usuários é um ciclo constante, onde os usuários encontram os erros, submetem os mesmos a testes que podem ser efetuados com diferentes combinações de hardware, e até mesmo em situações bastante distintas, fazendo com que o resultado dos testes utilizados tenha um melhor resultado após serem reportados novamente aos desenvolvedores, os quais são corrigidos e posteriormente lançados na próxima versão do sistema, o numero de usuários na comunidade se torna importante nessa parte do projeto, pois quanto maior o numero de usuários de um determinado projeto maior é a eficiência desta seqüência, o que nos leva ao fato de que a disponibilização adequada de um projeto pode influenciar muito em seu sucesso ou desgaste da distribuição.

4.10 DOCUMENTAÇÃO E TRADUÇÃO

O Debian possui uma ótima documentação que esta disponível na internet, e ainda em sua comunidades tem oportunidades para que sejam feitas a documentação dos demais softwares que estão presentes no sistema.

No modo de produção Debian, foi verificado uma postura bastante eficiente com relação a criação e atualização de sua documentação, fazendo com que o Debian se destaque por sua vasta e bastante detalhada documentação em português abordando vários aspectos do sistema operacional, além dos sites da distribuição e da comunidade, tornando difícil com que algum usuário se sinta perdido no Debian, isso sem falar no fato de que muitos dos representantes brasileiros da comunidade

estão bastante presentes em redes sociais e no IRC freqüentemente disponibilizando conversas que ajudem o usuário.

Muitos usuários que se interessam por ajudar, e não possuem uma grande experiência com programação ou até mesmo não saibam programar, pode ajudar a comunidade tanto contribuindo na documentação quanto traduzindo os conteúdos para a língua portuguesa. Em suma, a resposta mais comum obtida quando se indagou algum membro das comunidades de “software livre” “sobre o que é preciso para se entrar e participar da comunidade”, a resposta foi boa vontade, embora haja diferenças entre as distribuições e cada uma delas siga suas próprias filosofias e preceitos, assinala-se que o espírito da comunidade de software livre esta presente em todas as distribuições, e não somente na comunidade Debian.

5 RESULTADOS DO DEBIAN

5.1 SISTEMAS DEBIAN-LIKE

Não se tratando apenas das versões Debian-like, mas de uma forma geral nas distribuições Linux, as mesmas tem se revelado a cada nova versão, trabalhos de excelência técnica, fato que é comprovado por sua ampla aceitação em instituições educacionais, organizações governamentais e não-governamentais e no próprio mercado, isso sem falar nos servidores os quais usam essas distribuições de núcleo GNU/Linux por motivos de confiar em suas ferramentas e desempenho para empreendimentos em que se nota necessário maior eficiência.

Sistemas Debian-like são bastante usados no Brasil, e mundialmente famosas em parte graças ao Linux Ubuntu a qual é baseada no Debian, e são facilmente identificáveis por causa do uso de um aplicativo chamado APT (Advanced Packanging Tool), um gerenciador de pacotes que faz o download não apenas do aplicativo solicitado mas como o seu próprio nome já especifica, ele tambem baixa todos os pacotes e dependências dos aplicativos, facilitando bastante o trabalho do usuário.

5.2 ABRANGÊNCIA DO DEBIAN

A distribuição do software livre Linux Debian é uma das mais usadas no mundo e além da própria distribuição Debian estão disponíveis também versões baseadas no Debian, também conhecidas como versões Debian-like, todas elas bastante amigáveis para utilização tanto dos usuários novatos

quanto dos usuários mais exigentes pois todas apresentam resultados bastante eficientes, os exemplos se tratam desde versões mais antigas como o knoppix, kalango, e kurumin Linux, quanto de versões que estão fazendo bastante sucesso atualmente como o Linux Backtrack, versão Linux voltada para realizar testes de segurança, Xandros, Foxy Linux, Linux Educacional utilizado em todas as escolas estaduais de Santa Catarina, visando facilitar a inclusão digital dos alunos da rede publica de ensino, Linux Mint, e todas as versões baseadas no hoje famoso Linux Ubuntu como o Linux Kubuntu, Xubuntu, Nbuntu, etc.

4. PERSONALIZAÇÃO DO SISTEMA

1 POSSIBILIDADES DO SOFTWARE LIVRE

O Sistema Operacional Linux Debian assim como muitas outras distribuições derivadas do Unix possuem uma série de opções de personalização dos recursos do sistema, permitindo ao usuário o acesso ao código-fonte garantindo com isso que ele conheça a maneira como foi programado os aplicativos, bibliotecas e recursos que compõem o sistema, incluindo também os recursos nativos de edição e criação de softwares que permitam ao sistema uma maior personalização de forma a possibilitar ao usuário a maior interação para com o sistema o qual esta usando, e se essas mesmas alterações forem disponibilizadas para o uso de outros usuários, a GPL garante que com a mesma seja publicado o código-fonte de forma a possibilitar aos futuros usuários os mesmos direitos a todos que partilham do desejo de utilizar um software livre.

Dentre os aplicativos que estão incluídos no Sistema Operacional Linux Debian se destacam o compilador GCC, o qual permite que programas em linguagem de programação C sejam criados pelo usuário conforme a sua necessidade, um editor de textos simples o qual é aberto pelo terminal, normalmente se tratando do "Vi", o qual permite a edição, manipulação e visualização dos códigos-fontes disponibilizados no Linux, possuindo também um aplicativo interpretador de comandos chamado Shell, o qual executa os comandos enviados pelo usuário ou pelo administrador, podendo ter vários interpretadores de comando como por exemplo o c-shell, Shell, e dentre eles se destaca o Bash (Born Again Shell) que vem em grande parte das distribuições Linux, os quais não apenas executam os comandos enviados pelos usuários, mas também executam diversos comandos criados em lotes através de scripts de Shell que são inseridos na máquina possibilitando uma maior personalização do sistema.

1.1 SHELL

O Shell é um programa interpretador de instruções, que facilita em muito o acesso do usuário as tarefas do sistema operacional, o qual surgiu no Unix. O sistema Unix foi composto da seguinte forma, com o hardware sendo acessado pelo kernel, e esse sendo acessado por determinados programas ou comandos, e esses comandos sendo acessados pelo Shell, assim sendo o Shell se trata de uma camada que permite ao usuário entrar em contato com o sistema operacional e realizar seus

comandos, porem o Shell permite acesso quase ilimitado ao sistema Linux, tanto para edição, análise, e execução de comandos na montagem do acesso ao hardware, facilitando em muito a vida dos usuários que sabem como utilizá-lo.

O Shell disponibiliza ainda a possibilidade de criar a execução de vários comandos por meio de um script também conhecido por shellscript, os quais vem a facilitar a vida do administrador de redes e usuários automatizando tarefas e até mesmo facilitando tarefas bastante sofisticadas.

> O Shell analisa a linha e identifica, separados por espaços em branco, o nome do programa (comando), pesquisando a sua existência seguindo o caminho padrão; identifica se possui opções e quais são elas, seus redirecionamentos e suas variáveis. Quando o programa identificado existe, o Shell analisa as permissões dos arquivos envolvidos, dando uma mensagem de erro, caso o operador não esteja credenciado a executar esta tarefa. (NEVES, 2000. p. 7)

A versão Shell mais utilizada e que vem nativa em várias distribuições Linux é o bash (Bourne again Shell), mas existem mais versões com diferentes possibilidades, como exemplo podem ser dados o Bourne Shell (era a versão padrão de muitas distribuições Unix), Korn Shell e o C Shell.

1.2 GCC

O compilador gcc (GNU Compiler Collection) é um conjunto de compiladores de linguagens de programação produzido pelo projeto GNU e distribuído pela FSF (Free Software Foundation) sendo um software livre o qual já vem nativo em varias distribuições Linux, Solaris e FreeBsd. Inicialmente tinha suporte somente à linguagem de programação C, mais tarde ganhando suporte a algumas outras como o Java, C++, Fortran, etc. mas o mais largamente utilizado ainda é o C.

O compilador GCC foi escrito em 1987 pelo próprio Richard Stallman para servir de compilador do projeto GNU, e vem sofrendo um bom numero de atualizações e pode ser executado em vários tipos diferentes de hardware, rodando em produtos Apple, fazendo com que o que for compilado no GCC em uma maquina rodando Linux tenha grande possibilidade de ser rodado em uma máquina de tecnologia Apple, ou demais tecnologias, fazendo com que esse compilador multiplataforma seja útil em muitos sistemas e não apenas no sistema operacional Linux, fazendo com que seja uma boa opção para estudos.

1.3 EDITORES DE TEXTO

Editores de texto simples são muito importantes para a composição de uma distribuição Linux, pois é por meio deles que se torna possível tanto a visualização quanto a edição do código que vem disponível dentro do software livre, os editores de textos mais conhecidos do mundo Linux são o Vi, que na maioria das vezes já vem nativo em boa parte das distribuições Linux, e é cobrado sua operação durante os testes de aprovação da certificação para profissional Linux LPI-101 e LPI-102, outro editor de textos bem conhecido é o emacs, o qual existe desde os primeiros anos do software livre, e hoje já estão disponibilizados versões mais amigáveis de edição de texto como o Gedit, Kedit, Tomboy, etc.

1.4 BACKUP E SEGURANÇA

Para exemplificar uma das muitas funcionalidades do software livre, foi mostrada a criação de um script em Shell que vem a auxiliar na criação de backups dentro da rede utilizando somente os recursos que já vêm inclusos na distribuição Debian-like Linux Ubuntu, vindo a facilitar à vida do usuário que conhece o funcionamento do sistema, fazendo com que ao usuário perceber a necessidade de uma ferramenta de backup não seja necessário a aquisição de nenhum software adicional, podendo ele mesmo criar um utilitário de backup, nesse caso utilizando o Shellscript.

Primeiramente foi criado dentro de um diretório onde todos os demais usuários do sistema pudessem ter acesso, sendo escolhido o diretório /home e criado dentro dele um novo arquivo chamado backup, e dentro dele foi adicionado os comandos:

```
#! /bin/bash
cd /home/adair ; cp =r * //192.168.0.13/home/backup;
echo $? > /home/adair/Área\ de\ Trabalho/respostabackup.txt
```

E após isso foi salvo e mudado as permissões de acesso do script backup com o comando:

```
chmod 777 backup
```

Sendo finalmente adicionada a tarefa dentro do utilitário crontab, o qual é responsável por agendar tarefas dentro do Linux. Para isso foi acessado o arquivo crontab dentro de /etc e editado o arquivo crontab com um editor de textos simples.

```
00 18 *  *  * root  /home/backup
```

As alterações acima foram feitas com a intenção de agendar a execução do script de backup que copiará os arquivos do diretório /home/adair para outro host da rede todo o dia as zero horas e 18 minutos, foi escolhido um horário preferencial no qual a rede não esta sendo usada para que não haja congestionamento da mesma. E finalmente foi reiniciado o serviço do cron no servidor dentro do diretório /etc/init.d com o comando:

```
./cron restart
```

1.5 PROXY SQUID

O Squid se trata de um software livre que roda sobre diversas distribuições Linux e que atua como um servidor de Proxy dentro da rede e também efetuar algumas outras tarefas, como geração de logs, etc. O termo Proxy vem da palavra em inglês que significa procuração, fazendo com que um servidor de Proxy se trate de um software que tem a procuração de um ou mais hosts para buscar na internet uma informação solicitada pelos hosts que compõem a rede, cabe a esse serviço avaliar se a solicitação do host para a internet e valida ou não, incluindo avaliar se a mesma solicitação trabalha com protocolos, portas ou endereços IP ou MAC permitidos.

O Proxy ainda traz por vantagem atuar como um grande tipo de memória cache usada para armazenar trafego de determinados tipos de protocolos como o FTP e HTTP, já que não há sentido em armazenar acesso a e-mails, conversas por ICQ, IRC, POP3, etc, já que os mesmos tendem a ser atualizados com uma certa freqüência. Ele atua fazendo com que um site uma vez aberto fica armazenado na memória do servidor Proxy diminuindo o trafego de informações utilizado pela banda da internet e agilizando o carregamento das paginas e aumentando a segurança limitando a atuação de vírus e acessos indevidos ao sistema, podendo filtrar os acessos tanto por endereço ip da

maquina host, quanto por MAC address da placa de rede, por protocolos, portas, nomes de domínio, etc.

Para efetuar a instalação do software Squid e o Sistema Operacional utilizado for Linux Debian ou alguma versão Debian-like, como por exemplo o Ubuntu, o mesmo pode ser feito através do comando:

apt-get install squid

O gerenciador de pacotes irá baixar e instalar os pacotes necessários para o bom funcionamento do Squid, e criar um diretório localizado em /etc/squid, o qual contem arquivos como o mime.conf, squid.conf.default (para o caso de que seja necessário restaurar as configurações padrão do arquivo squid.conf) e squid.conf, criando tambem o diretório var/log/squid o qual conterá três arquivos chamados store.log, cache.log e access.log, os quais se tratam de arquivos para armazenar os logs de entrada e saída do Squid, e sobre os quais será detalhado o funcionamento no decorrer mais adiante.

Para que seja possível editar o arquivo squid.conf e realizar as alterações necessárias o usuário pode usar um editor de textos como por exemplo o Vi, e editar as partes que achar necessário alterar, ou é possível que seja zerado o conteúdo do arquivo squid.conf através do comando:

echo –n > /etc/squid/squid.conf

O arquivo de configuração principal que compõe o Squid se chama squid.conf e esta localizado no dentro do diretório etc, no subdiretório squid, o arquivo original é dividido em seções, como por exemplo, Network Options, apenas com o intuito de melhorar a visualização e compreensão, mas na verdade não tem exatamente um efeito prático, podendo-se misturar os comandos nas seções, conforme a necessidade do utilizador.

Para abrir ou editar o arquivo squid.conf é necessário escolher um editor de texto como por exemplo o Vi, que já vem nativo em muitas distribuições Linux ou escolher um editor de textos que melhor agradar ao usuário, como no exemplo abaixo.

gedit /etc/squid/squid.conf

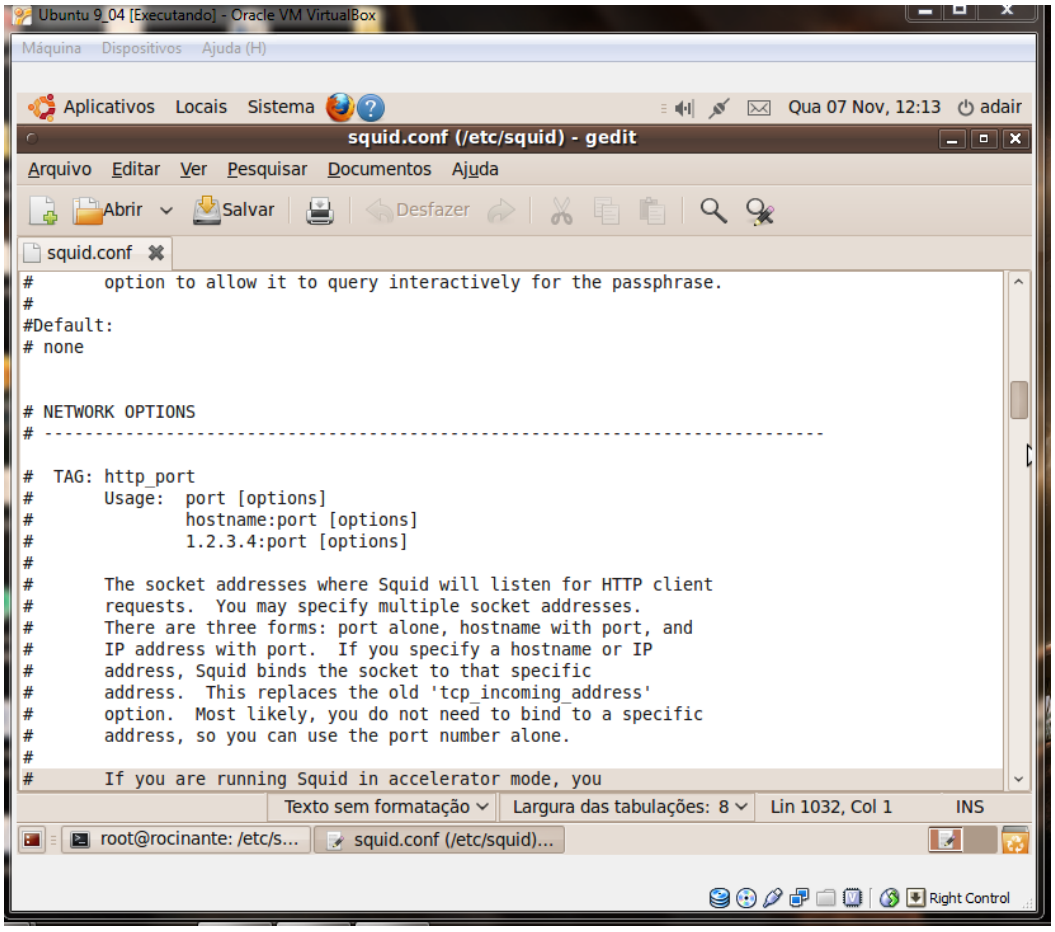

Figura 08: Squid.conf dividido em seções.

O Squid permite determinadas alterações já pré-configuradas em seu Proxy, livrando o administrador de redes do trabalho de lembrar o comando inteiro, oferecendo uma ajuda em suas configurações pré estabelecidas, bastando encontrar a palavra que distingue o determinado protocolo e retirar o caracter "#" o qual serve para comentar determinada linha, então se quiser que aquela determinada função entre em ação é preciso apenas retirar o caractere de comentário.

Por exemplo se quisermos habilitar ou desabilitar o protocolo icp e sua determinada porta, é preciso apenas retirar o comentário da linha correspondente para que a porta 3130 funcione conforme o padrão, podemos mudar o numero da porta ou até mesmo mudar o valor para o valor 0, fazendo com que a porta esteja realmente desabilitada, o mesmo pode ser feito com uma série de outros protocolos, permitindo sua configuração apenas acrescentando uma linha de comando ao arquivo de configuração.

icp_port 3130

Para determinar detalhes a respeito do próprio Squid é possível editando as linhas comentadas no arquivo default, por exemplo o administrador de redes tem acesso a detalhes importantes do sistema, como a possibilidade de especificar o tamanho da cachê utilizada, ou do tamanho dos objetos a serem armazenados pelo Proxy apenas acrescentando ou editando no arquivo a linha:

maximum_object_size_in_memory 8 KB

À qual na versão utilizada do Squid estava determinava o tamanho dos objetos a serem armazenados na memória do Proxy como tendo no máximo 8 KB mas podendo ser expandido conforme a capacidade do servidor, hoje com bastante memória secundária disponível é possível determinar tamanhos de objetos maiores, conforme a sua necessidade, como no exemplo a seguir:

maximum_object_size_in_memory 16 MB

Essa opção pode vir a diminuir bastante o trafego dentro da rede já que armazena arquivos que ocupariam um tempo maior de download desse tipo de informações, mas deve ser usada com cuidado para não sobrecarregar o servidor já que são vistos como objetos os arquivos que são baixados junto com a pagina web, como arquivos em HTML, imagens GIF, JPG, BMP, etc.

O tamanho de cachê dedicado ao armazenamento de endereços ip pode também ser determinado utilizando as opções:

ipcache_size 1024

ipcache_low 90

ipcache_high 95

As opções acima determinam o tamanho da memória dedicada ao armazenamento de endereços ip, como padrão já vem com 1024 KB, mas o que pode vir a ser expandido, e também o numero mínimo (low) e máximo (high) de ips armazenados na memória.

O administrador de redes tem também a possibilidade de determinar o diretório onde vai ser armazenado os arquivos de cache, por padrão já vem configurado que seja armazenado como determinado na linha abaixo:

cache_dir ufs /var/spool/squid 100 16 256

O que determina pela opção cache_dir que o arquivo seja armazenado no diretório “/var/spool/squid”, com o tipo de sistema de alocação já configurado para ser do tipo “ufs”, a quantidade máxima de espaço a ser usada nesse diretório pode ser determinada aqui em MegaBytes, a qual vem com o valor “100”, ou seja, é permitido que o diretório ocupe até no máximo 100 MB de

espaço em disco, o que pode ser expandido alterando o valor para um valor correspondente ao que seja possível sem prejudicar o bom funcionamento do sistema, determinando até 16 diretórios, com até 256 subdiretórios em cada um deles o que pode ser expandido facilmente.

O Squid permite gravar os logs de todos os pedidos solicitados a ele, e onde serão salvos esses determinados logs podem também ser configurados procurando a opção cachê_log, a qual por dafault já salva em /var/log/squid em um arquivo chamado cache.log, podendo haver mais dois tipos de arquivos de log chamados de store.log e access.log. os quais podem ter sua localização configuradas inserindo as seguintes linhas no arquivo squid.conf:

```
cache_log /var/log/squid/cache.log
access_log /var/log/squid/access.log squid
cache_store_log /var/log/squid/store.log
```

Esses acessos aos logs visam permitir que o administrador de redes possa posteriormente analisar as entradas, saídas e pacotes recusados que trafegaram na rede conforme seu interesse e sua necessidade, ou até mesmo redirecionar outro diretório de maior confiabilidade.

Após realizar a configuração do arquivo squid.conf é preciso iniciá-lo a partir do diretório /etc/rc.d/init.d e utilizando o comando:

```
# ./squid start
```

E para interromper o squid basta digitar:

```
# ./squid stop
```

Para abrir ou editar o arquivo squid.conf é necessário escolher um editor de texto como por exemplo o Vi, que já vem nativo em muitas distribuições Linux ou escolher um editor de textos que melhor agradar ao usuário, como no exemplo abaixo.

```
# >>>>>>>>>>>>>>>>>>>>Exemplo Básico de configurações iniciais.
http_port 3128
cache_mem 8 MB
maximum_object_size 4096 KB
cache_dir ufs /var/cache/squid 100 16 256
cache_access_log /var/log/squid/access.log
cache_log /var/log/squid/cache.log
cache_store_log /var/log/squid/store.log
```

e nas linhas abaixo pode-se continuar a configurar mais detalhadamente o squid acrescentando comandos direcionados a diferentes serviços a serem executados por ele, como tornar um serviço de Proxy transparente, como no exemplo abaixo:

```
# >>>>>>>>>>>>> proxy transparente

httpd_access_host virtual
httpd_access_port 80
httpd_access_with_proxy on
httpd_access_uses_hosts_header on
```

Os comandos acima ao serem adicionados ao squid fazem com que o usuário dos hosts da rede não tenha a necessidade de conectar o seu browser ao Proxy cada vez que formatarem a maquina, as linhas acima direciona as requisições de HTTP destinadas a porta 80 passe primeiro para o servidor Proxy antes de seguir para a internet.

1.5.1 ACL (ACCESS CONTROL LIST)

Os comandos que determinam que o administrador de redes possa coordenar de forma eficiente o fluxo da rede dentro do Squid são as ACLs (access control lists – listas de controle de acesso), são por meio delas que é possível controlar o fluxo de usuários que podem ou não ter acesso a determinado conteúdo da internet, quais protocolos podem trafegar, quais portas estão abertas e para quais usuários, ou endereços IP é permitido acesso à internet.

> Sem dúvida as ACLs são a parte mais importante do squid.conf, pois com o uso das mesmas bem configuradas e planejadas, é possível não só manter seus usuários sob controle, mas também melhorar o desempenho e facilitar a administração. O conceito de ACL é muito útil, por nos permitir trabalhar com níveis de acesso baseados em diversas informações. As ACLs permitem especificar endereços de origem ou destino, domínios, horários, usuários, portas ou métodos de conexão ao Proxy, que servirão de base para permitir ou negar o acesso baseando-se em conjuntos dessas ACLs.(LUNARDI, 2005. P. 30)

As listas acl estão dispostas da seguinte maneira:

acl Nome Tipo Obj1 Obj2 ...

O comando acl no inicio da linha demonstra que o que virá a seguir se trata de comandos para configuração de listas de controle de acesso, em seguida é declarado o nome da lista, seguido pelo seu tipo, os quais podem ser do tipo “src”, que trata do endereço de origem da máquina solicitante, “dst” que se refere ao destino do pacote, “time”, que determina a data/hora da solicitação, “port”, que determina o numero da porta solicitada, “proto” a qual especifica o tipo de protocolo que esta trafegando na rede. Seguido pelos objetos propriamente ditos que tendem a combinar com o seu tipo.

As configurações mais básicas de ACLs são mostradas abaixo, não se tratando de todo o Proxy Squid, mas sim de suas principais regras:

```
# >>>>>>>>>>>>>>>>>> ACLs gerais
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.1/8
acl SSL_ports port 443          # https
acl SSL_ports port 563          # snews
acl SSL_ports port 873          # rsync
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
```

```
acl Safe_ports port 631         # cups
acl Safe_ports port 873         # rsync
acl Safe_ports port 901         # SWAT
acl purge method PURGE
acl CONNECT method CONNECT
```

Sendo que a primeira parte trata das configurações da rede em geral, não exatamente da rede interna atendida pelo administrador de redes, definindo na primeira linha todos os hosts da rede como o nome all.

A segunda linha que trata do acl denominada manager dá acesso a um determinado protocolo no caso o protocolo cache_object, um tipo de protocolo só presente no Squid o qual retorna para o servidor informações de como o servidor está configurado (poderia ser o FTP ou o HTTP), dependendo das necessidades da rede a ser implementada.

A acl localhost determina como o endereço solicitante com o endereço ip 127.0.0.1 com o nome localhost, e a seguinte acl to_localhost determina todo pacote que for destinado a esta acl vai ser destinada a rede 127.0.0.0/8.

A acl ssl_ports associa as portas numero 443 e 563 como portas em que se podem fazer uma conexão segura, assim como as demais portas abaixo estão associadas ao conjunto de acls Safe_ports, as quais virão a ser úteis nos comandos seguintes.

No exemplo abaixo será incluído regras que impeçam que os hosts de endereço ip 192.168.0.66 e 192.168.0.67 tenham acesso à internet, e para isso além de ser feito uso das acls declaradas acima será preciso ainda declarar novas acls:

```
#  >>>>>>>>>>>>>>>>>> Minhas ACLs
acl network src 192.168.0.0/24
acl proibir_ip src 192.168.0.66 192.168.0.67
```

As linhas acima declaram como network a minha rede constituída dos endereços da rede 192.168.0.0 e a lista de controle de acesso abaixo separa as requisições que sejam provenientes dos endereços ip 192.168.0.66 e 192.168.0.67 em uma nova lista chamada proibir_ip.

```
# >>>>>>>>>>>>>>>>>> Diretivas HTTP_access
http_access allow manager localhost
http_access deny manager
```

As linhas acima permitem que somente o localhost tenha acesso ao protocolo cache_object, e que a seguir nega a todos os demais hosts da rede acesso ao protocolo determinado na acl denominada “manager”.

```
http_access deny!Safe_ports
```

Ainda nas diretivas do HTTP_access estão os comandos que farão uso das diretivas que foram declaradas anteriormente, a primeira linha apresentada trata de negar acesso a todas as portas diferentes daquelas declaradas na lista de controle de acesso denominada Safe_ports. Ou seja, foi criado um conjunto de números de portas seguras denominado Safe_ports e inserido determinados valores nesse conjunto, e com o comando acima foi negado permissão a todas as portas que não fizessem parte desse determinado conjunto.

```
http_access deny CONNECT!SSL_ports
```

O comando acima nega qualquer conexão a qual não seja feita pelas portas declaradas anteriormente na lista de controle de acesso denominada SSL_ports, fazendo com que uma conexão remota por telnet, usando a porta 23 não seja possível, nem mesmo uma conexão por ssh via porta 22, já que as únicas portas com permissão de conexão são as portas 443 e 563.

```
http_access deny proibir_ip
http_access allow network
http_access deny all
```

As três linhas de comando acima trabalham em conjunto, fazendo com que sua ordem seja muito importante para o bom funcionamento da rede, a primeira linha nega acesso à internet aos dois endereços IP declarados dentro da acl proibir_ip, e na linha seguinte dá acesso à internet na

minha rede 192.168.0.0 já eliminando os dois endereços ip contidos em proibir_ip, e na ultima linha ele nega acesso a todos os outros endereços ips que não fazem parte da acl denominada network.

O Squid permite que sejam proibidos os acessos a determinados sites, esse tipo de recurso era visto em redes mais antigas, as quais impediam que o usuário entrasse no site do Orkut, por exemplo. Esse recurso pode ser realizado criando um arquivo de texto no diretório /etc/squid denominado “acesso_proibido.txt”, e dentro desse arquivo podem ser digitados os sites os quais terão o acesso negado, como “www.uol.com.br” e acrescentar a seguinte linha ao arquivo squid.conf.

```
# >>>>>>>>>>>> Minhas ACLs
acl proibir_sites dstdomain "/etc/squid/acesso_proibido.txt"
```

A primeira linha acima determinou que o texto com as URLs que estejam contidos no arquivo acesso_proibido.txt seja adicionado à lista de controle de acesso chamada proibir_sites. E a maneira como essas determinadas URLs serão tratadas pelo Squid será determinado na linha adicionada posteriormente.

```
#  >>>>>>>>>>>>>>>>>> Diretivas HTTP_access
http_access deny proibir_sites
```

Esse comando delimita o acesso aos sites contidos no arquivo, porém não impede completamente o acesso aos mesmos, já que ao tentar acessar a URL “www.uol.com.br” o usuário terá seu acesso negado, porém se acessar o mesmo site pela URL “www.uol.com” ele terá o acesso permitido.

Para inibir esse tipo de falha existe uma outra forma de restringir o acesso a determinados sites, o qual consiste em restringir os acessos às URLs por palavras, por exemplo, se segundo a política de determinada empresa for determinado que não pode ser acessado o site de redes sociais como o facebook, orkut ou youtube nos hosts da empresa, para impedir de forma eficiente que os usuários tenham acesso a esse tipo de sites e a sites relacionados à palavra “sexo” por exemplo, poderá ser restringido esse tipo de sites primeiramente criando um arquivo no diretório /etc/squid chamado palavras.txt e acrescentando nele as palavras, sexo, facebook, orkut e youtube e acrescentar a linha seguinte ao arquivo squid.conf.

acl proibir_palavras url_regex -i "/etc/squid/palavras.txt"

À qual acrescenta uma acl denominada proibir_palavras que vai retirar o seu conteúdo do que estiver contido no arquivo palavras.txt. Posteriormente acrescentando a regra abaixo no squid.conf, à qual nega acesso aos endereços de URLs as quais contenham as palavras determinadas no arquivo palavras.txt.

http_access deny proibir_palavras

Porém dentro de determinadas empresas existem políticas de exceções que determinam que sites sobre sexo não podem ser acessados, porém sites relacionados a educação sexual a URL "sexoesaúde" podem ser acessados, ou outros sites que contenham as palavras determinadas anteriormente mas com contextos diferentes, como por exemplo abrir uma exceção para o site "perolasdoorkut". Criou-se primeiramente um arquivo de texto no diretório squid chamado liberado.txt o qual contém as palavras com as exceções a serem tratadas.

sexoesaude
peroladoorkut

Posteriormente foi acrescentado as seguintes linhas ao arquivo squid.conf.

acl liberado url_regex -i "/etc/squid/liberado.txt"

criando uma nova acl chamada "liberado" e acrescentando a ela o conteúdo do arquivo "liberado.txt". E acrescentar as regras do squid a linha abaixo.

http_access allow liberado

Com o acréscimo da linha acima foi permitido acesso à sites como “www.peroladoorkut.com” e negado acesso à sites como “www.orkut.com”, dando mais eficiência ao funcionamento da rede.

Após acrescentar as regras descritas acima ao arquivo squid.conf as novas regras das ACLs ficaram dispostas conforme mostrada abaixo:

```
# >>>>>>>>>>>> Regras das minhas ACLs
httpd_access deny proibir_ip
http_access allow liberado
http_access deny bloqueado
http_access allow network
http_access deny all
```

1.6 ANÁLISE DE LOGS

Dentro do diretório /var/log/squid está localizado três diferentes arquivos de log, sendo eles o access.log, store.log e cache.log, dentre os quais o mais importante deles é o arquivo access.log, que traz informações mais relevantes sobre o trafego de rede.

O arquivo access.log é o arquivo que contem todas as informações de acesso dos usuários, mostrando informações do trafego de dados na rede gerados pelos usuários, tanto sobre os pacotes aceitos na rede quanto os pacotes que foram negados, contem informações sobre o tamanho do trafego gerado na rede, quais maquinas ou usuários utilizaram mais banda de rede, quais fizeram mais downloads, tudo para ajudar o administrador de redes a controlar melhor os acessos ao serviço de rede dentro da empresa na qual trabalha.

A análise desses arquivos de log podem ser feitas manualmente abrindo os arquivos com editores de texto simples como o Vi, Gedit, Notepad++, etc. porém esse tipo de leitura pode gerar um excesso de informações bastante confusa até mesmo para usuários avançados, e em um acesso remoto via ssh pode se tornar inviável muitas vezes sobrecarregar a rede chamando todos os dados do arquivo de log, que muitas vezes pode ser bastante volumoso, tornando muitas vezes mais viável o uso do comando tail para vizualizar apenas as 10 ultimas linhas do arquivo de log.

tail –f /var/log/squid/access.log

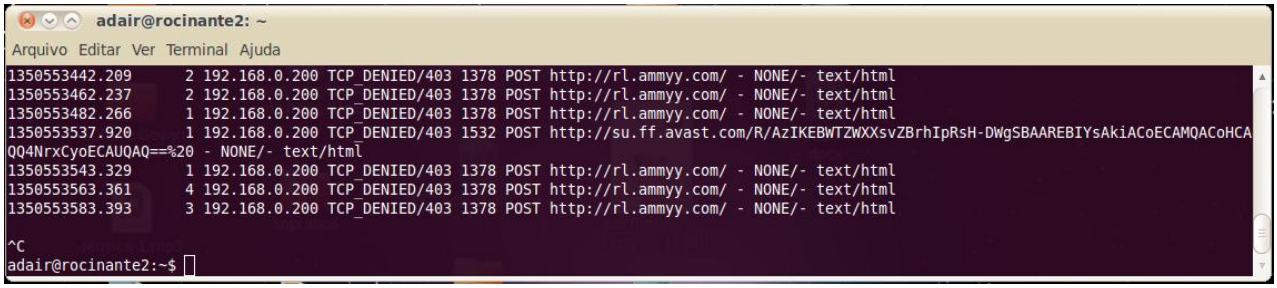

Figura 09: comando tail sobre o arquivo access.log

Tais resultados mesmo que simplificados podem se tornar inviáveis para que o usuário comum tenha entendimento sobre os resultados apresentados na tela, para tornar mais fácil para o usuário leigo entender tais resultados existem programas como o SARG (antigo Sqmgrlog), e o Calamaris para filtrar e apresentar de forma mais amigável os resultados, facilitando durante uma auditoria, reunião, ou até mesmo uma apresentação para o setor administrativo que não possui um grande entendimento de analise de trafego na rede.

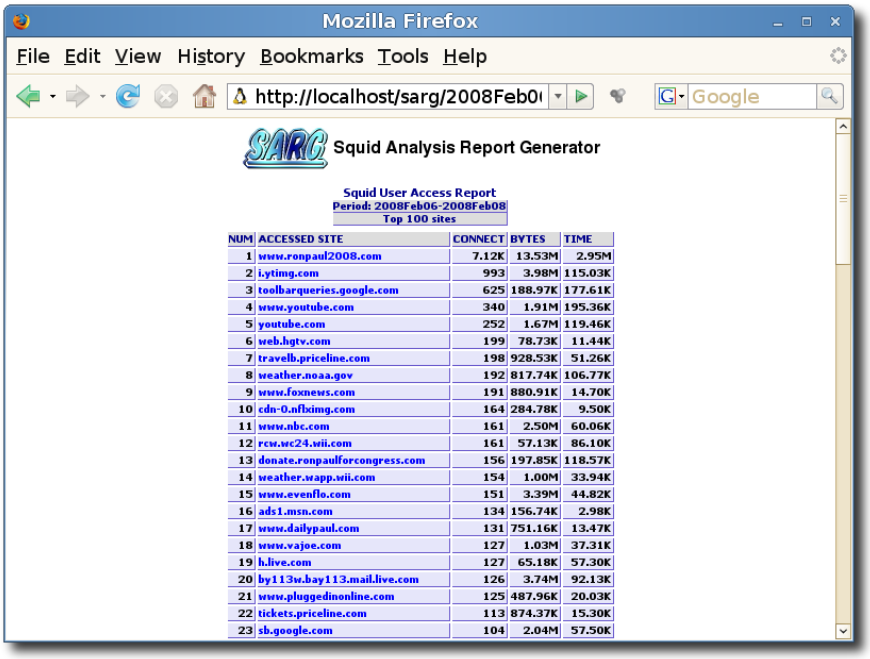

Figura 10: Tela para análise de logs do Sarg.

Fonte: Pcc-services. 2012.

5. RESULTADO DA APLICAÇÃO

Como resultado do estudo de caso é trazido como exemplo o caso da IBM e da distribuição baseada em Debian denominada Linux Educacional a qual é utilizada em todas as escolas estaduais de Santa Catarina com o objetivo de facilitar a inclusão digital nas diversas camadas sociais que freqüentam as escolas da rede publica estadual.

O caso da IBM se caracteriza por ser uma empresa que utiliza de meios nativos do modo de produção Debian para agregar resultados mais eficientes na produção de seus softwares. Hoje se tratando de uma das empresas que mais incentivam o software livre e utilizam do mesmo para construir aplicativos para seus produtos.

> A IBM era uma candidata improvável a campeã de compartilhamento e líder do mundo de código aberto, afinal, estamos falando da Big Blue – a empresa que se tornou enorme construindo e vendendo tudo o que fosse de propriedade exclusiva. Durante décadas, criou softwares que só funcionavam nos computadores IBM. Azar de quem quisesse usá-los no hardware de outro fornecedor. A IBM chamava isso de “controle de conta”. (TAPSCOTT, 2007. p.101)

A IBM estava passando por uma reformulação de seu modo de produção de maneira que a mesma viesse a produzir bons resultados com um investimento menor, buscando uma alternativa ao seu sistema operacional OS/2 que trouxesse melhores resultados e que não precisasse de uma força de produção tão grande quanto outras empresas produtoras de sistemas operacionais. Permitindo a IBM estimar uma economia de cerca de 1 bilhão de dólares em investimento anualmente no que lhe custaria desenvolver um sistema operacional próprio que trouxesse funcionalidades semelhantes as do Linux, o que tornou viável economicamente adotar e incentivar o software livre e de código aberto, possibilitando com essa união que a IBM oferecesse produtos mais competitivos do que suas concorrentes.

> A IBM decidiu se envolver com o código aberto, mas não logo de cara com o Linux. A empresa se uniu ao grupo Apache, uma equipe de programadores que havia desenvolvido um software pra websites. O Apache já tinha cerca de metade do mercado de servidores web, e o novo produto da IBM, o Domino, tinha menos de 1%. A IBM não tinha muito a perde. (TAPSCOTT, 2007. p.102)

Essa parceria trouxe a tona o software para servidor web mais utilizado do mundo, o Apache, que tem grande parte dos servidores web rodando sobre o mesmo. Mostrando como a parceria entre empresa privada e comunidades de desenvolvimento do software livre podem interagir e gerar resultados satisfatórios para ambas as partes.

O outro exemplo de aplicação de resultado do modo de produção do software livre Debian é a distribuição Linux Educacional ncentivada pelo governo e que visa criar uma versão de sistema operacional que possibilite a redução de custos na implantação de laboratórios de informática nas escolas publicas facilitando com isso a inclusão digital aos alunos que estudam na rede pública de ensino e façam com que os mesmos sejam mais competitivos no mercado de trabalho.

1 PARECER FINAL

Ao decorrer da pesquisa sobre o modo de produção Debian foi mostrado como o software livre alcança resultados bastante satisfatórios os quais vem a se adaptar de forma eficiente as necessidades do meio no qual esta inserido, e por contar com a colaboração de desenvolvedores e colaboradores ao redor de todo o mundo, vem atingindo resultados satisfatórios e trazendo aplicativos capazes de suprir diferentes necessidades, tanto para que sirva de exemplo em algumas maneiras de trabalhar em grandes empresas como, por exemplo, a IBM quanto para projetos governamentais que visam facilitar a inclusão digital nas escolas publicas estaduais com o projeto da distribuição Linux educacional a qual é utilizada por vários alunos da rede pública de ensino inserindo os mesmos nas atividades ligadas a informática as quais são cruciais para a inserção do aluno no cada vez mais concorrido mercado de trabalho, conforme o documento apresentado nos anexos 1 e 2.

IV. CONCLUSÃO

O Linux tem se mostrado útil em setores da sociedade bastante distintos, tanto rodando em servidores de empresas de grande porte quanto em projetos que visam diminuir a desigualdade social, um desses projetos é o Linux Educacional que visa aumentar o contato dos alunos de escolas publicas com o uso da informática no seu cotidiano e à medida que as grandes empresas de software aderem ao modo de produção do software livre, mostra-se uma tendência de que as demais empresas no ramo informática mesmo de menor porte venham a seguir esse exemplo, mais uma vez mostrando que o software livre se apresenta não como uma ameaça ao mercado de produção de softwares proprietários, mas sim como uma alternativa capaz de gerar bons resultados desde que se saiba interagir de forma eficiente com a mesma.

V. ANEXO I

1.

Declaração

A Escola de Educação Básica Joao Paulo I possui um laboratório de informática, o qual é composto por 10 computadores, sendo que todas as máquinas utilizam como sistema operacional a distribuição Linux Educacional, a qual é derivada da distribuição Debian, e possui como ambiente gráfico a interface KDE.

Maria Helena Trochoniak dos Santos
Diretor Escola
Ato nº 526 de 14/03/2011
Matrícula: 310841-4-04

Diretor(a)

22/11/12

Data

2.

Declaração

A Escola de Educação Básica Nossa Senhora dos Prazeres possui um laboratório de informática, o qual é composto por 17 computadores, sendo que todas as máquinas utilizam como sistema operacional a distribuição Linux Educacional, a qual é derivada da distribuição Debian, e possui como ambiente gráfico a interface KDE.

Tiago Stor da Silva — Responsável

22/11/2012 — Data

REFERÊNCIAS BIBLIOGRÁFICAS

PISANI. Francis; PIOTET. Dominique. **Como a web transforma o mundo: a alquimia das multidões**. São Paulo. Ed. Senac, 2010.

TAPSCOTT. Don; WILLIAMS. D. Anthony. Wikinomics: **Como a colaboração em massa pode mudar seu negócio.** Rio de Janeiro. Ed. Nova Fronteira, 2007.

CASTELLS. Manuel. **A sociedade em rede.** São Paulo. Ed. Paz e Terra S/A, 2003.

CASTELLS. Manuel. **A galáxia Internet: reflexões sobre a internet**. Rio de Janeiro. Ed. Jorge Zahar, 2003.

LÉVY. Pierre**. Cibercultura**. São Paulo. Ed. 34 Ltda, 1999.

NEVES. Julio Cesar**. Linux: programando Shell**. Rio de Janeiro. Ed. Brasport, 2010.

GATES. Bill; RINEARSON. Peter**. A estrada do futuro**. São Paulo. Ed. Schwarcz, 1995.

DUBNER. J. Stephen ; LEVITT. Steven. Freakomics: **O lado oculto de tudo que nos afeta.** Rio de Janeiro. Ed. Elsevier, 2005.

VIDE. A. David; MALSEED. Mark. **Google: A história do negócio de mídia e tecnologia de maior sucesso dos nossos tempos.** Rio de Janeiro. Ed. Rocco, 2007.

PETROSKI. Henry. **A evolução das coisas úteis.** Rio de Janeiro. Ed. Jorge Zahar Ltda, 2007.

RESENDE. André. **Mundo Enquadrado: O lugar dos símbolos nas coisas reais.** São Paulo. Ed. Altana, 2005.

NAUGLE, Mathew G. **Guia Ilustrado do TCP/IP**. São Paulo. Ed. Berkeley Brasil, 2001.

TANEMBAUM, Andrew S. **Sistemas Operacionais Modernos**. São Paulo. Ed. LTC – Livros Técnicos Pedagógicos, 2004.

MAXWELL, Scott. **Kernel do Linux**. São Paulo. Ed. Makron Books, 2000.

TOSCANI, Simão Sirineo; CARISSIMI, Alexandre da Silva; OLIVEIRA, Rômulo Silva de. **Sistemas Operacionais.** Porto Alegre. Ed. ABDR, 2010.

NEMETH. Evi**. Manual Completo do Linux**. São Paulo. Ed. Pearson Prentice Hall, 2009.

HUNGER. Steve**. Debian GNU/Linux Bible**. Hoboken. Ed. Wiley Publishing Inc, 2001.

LUCINDA. Marco Antônio**. Qualidade: Fundamentos e práticas para cursos de graduação**. Rio de Janeiro. Ed. Brasport, 2010.

KLEIN. Naomi**. Sem logo**. São Paulo. Ed. Record, 2003.

TAYLOR. Dave**. Aprenda Unix em 24 horas**. Indianapolis. Ed. Sams, 2001.

LUNARDI. Marco Agisander**. Squid:Prático e didático**. Rio de Janeiro. Ed. Ciência Moderna, 2005.

RAYMOND. Eric Steven**. A catedral e o bazar**. Redwood. Ed. O'Reilly and Associates, 1999.

BECTA ICT Advice (2004), FITS Release**.** www.becta.org.uk/tsas/docs/fits_release.pdf. (janeiro de 2004).

PARECER FINAL

Reservado para a Coordenação do Curso:

Parecer:_____
_____
_____
_____
_____
_____

Data:_____/_____/_____

_____
Profº. Márcio José Sembay Msc.
Coordenador TCC Ciência da Computação