CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO
JONATHAN WOLFF ANDRADE

LOJAS BERLANDA: DESENVOLVIMENTO DE SOFTWARE APLICADO AO DEBIAN LINUX, LINUX HELPER SHELL SCRIPT E DIAGNOSTIC WEB HELPER SHELL

LAGES
2013

JONATHAN WOLFF ANDRADE

LOJAS BERLANDA: DESENVOLVIMENTO DE SOFTWARE APLICADO AO DEBIAN LINUX, LINUX HELPER SHELL SCRIPT E DIAGNOSTIC WEB HELPER SHELL

Trabalho apresentado ao Centro Universitário UNIFACVEST, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Msc. Márcio José Sembay
Co-orientador: Prof. Msc. Cassandro Albino Devenz

LAGES

2013

JONATHAN WOLFF ANDRADE

LOJAS BERLANDA: DESENVOLVIMENTO DE SOFTWARE APLICADO AO DEBIAN LINUX, LINUX HELPER SHELL SCRIPT E DIAGNOSTIC WEB HELPER SHELL

Trabalho apresentado ao Centro Universitário UNIFACVEST, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Msc. Márcio José Sembay
Co-orientador: Prof. Msc. Cassandro Albino Devenz

Lages, SC _____/_____/2013. Nota _____ _____

(data de aprovação) **(assinatura do orientador do trabalho)**

_____

(coordenador do curso de graduação, nome e assinatura)

LAGES

2013

AGRADECIMENTOS

Aos meus pais: Marinez Wolff, falecido pai João Farias de Andrade, padrasto Jonil França Rocha; a minha namorada Eduarda Clarissa da Silva. Pelo apoio em todos os sentidos para tornar meu objetivo possível, de graduar-me neste curso.

Aos Professores Márcio José Sembay e Cassandro Albino Devens pela sua sábia didática. Criando um ambiente favorável para o desenvolvimento deste projeto.

Aos meus colegas de aula, pelo companheirismo demonstrado ao passar destes quatro anos. Espero que todos façamos algo para contribuir com o avanço do mundo da Computação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AT&T – *American Telephone and Telegraph.*

BASH – *Bourne-Again Shell.*

ERP – *Enterprise Resource Planning. Ou Sistema integrado de Gestão Empresarial.*

CSS – *Cascading Style Sheets.*

CP – *Comando de cópia no GNU/LINUX.*

FSF – *Free software foundation.*

HTTP – *Protocolo padrão de navegação na Web.*

HUG – *Filosofia GNU/LINUX amigável*.

IP – *Internet Protocol*.

KISS – *Keep it simple stupid, filosofia GNU/LINUX robusta*.

OS – *Operating System*.

PHP – *Personal Home Page ou Hypertext Preprocessor*

RAM – *Randomic Aritimetic Memory.*

RM – *Comando para remover no GNU/LINUX.*

SH – *Shell.*

S.O – *Abreviatura de Sistema Operacional.*

TCP – *Transmission Control Protocol.*

TEF – *Transferência eletrônica de fundos (“cartões de crédito”).*

VOL – *Viva o Linux. Website da comunidade Linux brasileira*.

WWW – *World Wide Web.*

RESUMO

Este trabalho apresenta duas ferramentas desenvolvidas para facilitar a utilização do sistema operacional Linux e mostrar suas vantagens e comparação a outros sistemas operacionais. A ideia inicial de desenvolvimento foi oriunda do período em que o próprio Autor trabalhou nas Lojas Berlanda. Visa amenizar a complexidade, mostrando como é a interação entre usuário e sistema através de comandos com técnicas e didáticas diferenciadas, desenvolvidas em código aberto. Proporciona interfaces distintas, intuitivas e robustas, possibilitando futuras customizações para quem delas fizer uso. Mostra que mesmo neste contexto de facilitação e liberdade, é necessário conhecer a estrutura do sistema e a sintaxe dos comandos, neste caso homologadas para distribuições oriundas do Debian Linux. Possui recursos de acesso remoto via web browser controlado (garantindo segurança e qualidade de serviço no diagnóstico de possíveis problemas em sistemas operacionais Linux), além de acesso a informações do núcleo do sistema em ambas interfaces.

Palavras chaves: Código Aberto, Linux, Didático.

ABSTRACT

This paper presents two tools developed to facilitate the use of Linux operating system and show its advantages when compared to other operating systems. The initial idea of this development had its origins during the author's period working at Lojas Berlanda. Aims to alleviate its complexity, showing how occurs the interaction between user and system through different teaching techniques and commands, developed in open source. Provides separate interfaces, both intuitive and robust, enabling future customizations for those who make use of them. Shows that even in this context of facilitation and freedom, it is necessary to know the system structure and syntax of the commands, in this case approved for distributions derived from Debian Linux. It features remote access via controlled web browser (ensuring safety and good service in the diagnosis of potential problems on Linux), plus access to the core information system on both interfaces.

Keywords: Open Source, Linux, Didatic.

I. INTRODUÇÃO

1.1 Apresentação

Desde sua criação, o sistema Linux sempre foi considerado um sistema operacional de difícil utilização por pessoas com pouco conhecimento técnico sobre computação. Esta crença fez com que o sistema fosse adotado muito mais em servidores de empresas do que em computadores pessoais de usuários domésticos, onde o reinado dos sistemas proprietários se estabeleceu (GOMES JUNIOR, 2007).

Por mais evoluída que esteja a ciência da computação, a utilização de um Sistema Operacional personalizável e robusto ainda pode ser um desafio. Todos os tipos de usuários, em sua maioria, ficaram conformados e limitados pelo *S.O* atualmente mais difundido: o Windows. Ao mesmo tempo existe uma comunidade de usuários de Software Livre contribuindo com o avanço de Sistemas Operacionais concorrentes e de código aberto, estes derivados da plataforma do Unix, conhecidos como distribuições Linux.

Os motivos para ingressar neste ambiente ou comunidade, variam de usuário para usuário, mesmo sabendo das vantagens de utilizá-lo, muitos são desmotivados por sua complexidade.

> Se isto pode ser dito, Ken Thompson foi o inventor do Unix, então Linus Torvalds, um calouro da Universidade de Helsinki, inventou o Linux. Ele agora é famoso por ter postado em 25 de agosto de 1991 em seu Blog pessoal "Olá à todos! Estou criando um sistema operacional (free)", o que selou seu destino para sempre. As similaridades entre Thompson e Torvalds são curiosas. Uma pode ser que Thompson escreveu seu programa "Viagem no Espaço" apenas por diversão. Torvalds, em seu deslumbre com o Minix, outra distribuição do *S.O* Unix, achou tudo isto interessante de mais para criar uma versão do popular interpretador de comandos Unix, o *bash*, rodando em seu sistema operacional de “brincadeira”. Novamente, esta "brincadeira" desencadeou e alterou aspectos fundamentais para o desenvolvimento da indústria de softwares. (GANCARZ, 2003, p. 5, tradução nossa).

Hipoteticamente softwares que se encaixam neste contexto, possuem uma qualidade de serviço superior, em contrapartida são de difícil utilização, principalmente para usuários não habituados com as linhas de comando no terminal, presentes em todas as distribuições Linux. Isto pode ser interpretado como um problema para a disseminação do *S.O*, uma barreira que transforma o interesse e a busca de conhecimento, em um comodismo entre os usuários avançados e entusiastas.

Um longo caminho histórico, repleto de antecedentes, foi traçado até chegarmos ao Kernel Linux. É muito importante entender todo esse caminho. Muitos fatos que conhecemos hoje em dia foram causados por episódios antigos. A maior lição que se extrai de tudo isso é a capacidade do homem. A vontade de fazer suplanta obstáculos. Ken Thompson aprendeu isso. Linus Torvalds também. Sem falar em Richard Stallman, que conseguiu difundir uma filosofia pelo mundo inteiro (MOTA FILHO, 1996).

Interagir com o Kernel do sistema através de comandos, mesmo que seja para uma tarefa simples de mover um arquivo de uma diretório para outro, é logicamente, mais trabalhoso do que arrastar com o mouse. Consequentemente traz mais experiência ao usuário, forçando-o a aprimorar seus conhecimentos, antes de realizar determinada tarefa; uma disciplina que ao ser observada do ponto de vista de um usuário que nunca experimentou o Linux, será considerada equivocadamente como desnecessária.

Durante vários anos, os usuários de computadores pessoais acreditavam que a utilização de Linux estava reservada a administradores de redes e servidores. De fato, durante um bom tempo, a grande preocupação dos desenvolvedores era somente com as funcionalidades de servidor oferecidas pelo Linux. Felizmente, um grande esforço tem sido realizado pela comunidade de software livre em geral para tornar o Linux uma alternativa bastante atrativa aos usuários de desktops do mundo todo (GOMES JUNIOR, 2007).

Todo um estudo se torna requisito para compreender historicamente, filosoficamente e tecnicamente a estrutura deste *S.O* e seus recursos. Quanto mais dinâmico, robusto e didático for o aprendizado, mais consciente será o usuário.

1.2 Justificativa

Porque é robusto e estável por natureza, distribuições Linux são a escolha de milhões atualmente. Mas muitos não sabem que este é um projeto de software livre, uma parte do objetivo Linux, é em geral um fenômeno de contracultura: desenhado pela pessoa que o produziu e publicado ao contrário das noções de propriedade, intelectual que vem dominando a cultura “mainstream” há tempos. Enquanto muitos programadores focam seus esforços no desenvolvimento de programas comerciais que você não pode alterar, compartilhar ou examinar (somente adquirir e "rodar" em seu sistema), o Linux e outras soluções livres são produto resultante de muitos usuários que bravamente publicaram e compartilharam suas

pesquisas e trabalho abertamente para todos, para o benefício de todos (STUTZ, 2001, tradução nossa).

Este trabalho foi desenvolvido buscando suprir, mesmo que modestamente esta necessidade de facilitação da utilização do Linux pelo terminal e aprendizado dos comandos. Utilizando um shell script, de comandos *bash* na interface background (LINUX HELPER SHELL SCRIPT) de forma mais robusta e abrangente, contado com recursos que variam de muito simples à avançados. Este possui uma opção que configura uma interface mais limitada, porém mais didática e amigável para rodar em web browsers, que foca no breve diagnóstico de possíveis problemas na estrutura do *S.O* utilizando programação orientada a objetos em *PHP* (DIAGNOSTIC WEB HELPER SHELL), possibilitando o acesso remoto e seguro a várias informações úteis. Ou seja, uma forma alternativa de incentivo e demonstração dos recursos deste *S.O*, obviamente, ambos com código aberto e documentado. Possibilitando aos utilizadores customizá-los e aprimorá-los de acordo com sua necessidade.

Necessidades semelhantes são relatadas em livros relacionados, o que também justifica estas afirmações.

Quanto mais aprendia sobre os comandos e suas opções, mais conseguia automatizar tarefas rotineiras do servidor, fazendo pequenos scripts. Códigos simples, com poucos comandos, mas que poupavam tempo e garantiam a padronização na execução. Ainda tenho alguns deles aqui no meu $HOME, vejamos: mudar o endereço *IP* da máquina, remover usuário, instalação do MRTG, gerar arquivo de configuração do DHCPD. Em alguns meses já eram dezenas de scripts para as mais diversas tarefas (JARGAS, 2008).

1.3 Importância

Está diretamente ligada a facilidade proporcionada no resultado final e na possibilidade de livre alteração do código. Assim como é no comportamento do ser humano, está a busca pelo conforto, nada mais agradável do que utilizar um software poderoso, com segurança e facilidade. Além de modestamente impulsionar novos usuários e entusiastas ao GNU/Linux, shell script ou *PHP*.

O cérebro mamífero ou sistema límbico é uma área do cérebro responsável pelos sentimentos, a ativação desta área justifica a incessante fuga da dor e dificuldade presente no comportamento humano, o que pode ser visto também no apego do ser humano com a

tecnologia, o comodismo em aceitar a ideia de uma máquina utilizando um conjunto de hardware e software traz um resultado mais satisfatório ao realizar determinada tarefa. São comportamentos justificados pela utilização e ativação do sistema límbico.

> As demais formações anatômicas que tradicionalmente integravam o rinencéfalo são hoje estudadas como parte do chamado sistema límbico. Este pode, pois, ser conceituado como um sistema relacionado fundamentalmente com a regulação dos processos emocionais e do sistema nervoso autônomo constituído pelo lobo límbico e pelas estruturas subcorticais a ele relacionadas.
> (MACHADO, 2001, p. 277).

Se o objetivo da tecnologia é facilitar tarefas que antes eram massivas e desnecessariamente feitas manualmente, o que reflete diretamente na produção, seja em uma empresa ou instituição acadêmica. Porque não utilizar a mesma ideologia no desenvolvimento de softwares? Quanto mais amigável a interface, mais público ela obviamente atingirá, este alvo pode ser designado aos usuários domésticos e intermediários, o que não se aplica aos usuários mais experientes, que mesmo sem ter um contato prévio, provavelmente ficarão encantados com os recursos do Linux, pois a utilização pode se tornar simples para usuários avançados que conheçam outros *S.O* a fundo.

Com o Linux, você é livre para apagar o que quiser no disco e executar qualquer software nele (como em liberdade, não só em gratuidade). Como alternativa, você pode rodar o Linux a partir de um LiveCD (ignorando o conteúdo do seu computador, sem alterá-los) ou instalar o Linux para dual boot com o Windows ou MAC *OS* X, ou o sistema que você escolher. Conclui-se que com o Linux você é livre para escolher o que fazer com o seu computador. Em poucos anos, o Linux avançou, deixou de ser considerado um sistema operacional especialista tornando-se mainstream (convencional). Pré-compilados e configurados, os sistemas Linux conseguem ser instalados sem conhecimentos avançados. Versões do Linux rodam em todos os tipos de dispositivos, de PCs para portáteis, consoles de jogos (assim como o PlayStation 3) e supercomputadores em Marte. Em suma, o Linux se tornou um sistema que quase todo mundo consegue executar em qualquer lugar. (NEGUS, 2010, tradução nossa).

Estas afirmações se tornaram realidade, um exemplo é o novo produto da Canonical, empresa responsável pela distribuição Ubuntu. O Ubuntu Phone *OS*, lançado em 2013. Roda em smartphones e tablets, já é compatível com os dispositivos: Samsung Galaxy Nexus, LG

Nexus 4, Google Nexus 10, Google Nexus 7. Obviamente buscando preservar características do GNU/Linux, com a beleza do Ubuntu.

Figura 1 – Ubuntu Phone
Fonte: < http://www.extremetech.com/wp-content/uploads/2013/02/ubuntu-tv-pc-smartphone-tablet.jpg>

Autores de livros sobre shell script geralmente relatam suas experiências, para justificar e demonstrar a importância de seu trabalho, pois para um leigo pode parecer chato e desnecessário, porém isto é extremamente importante.

> A rotina era sempre a mesma: o Arnaldo sentava-se e demonstrava como fazer determinada tarefa, digitando os comandos, para que eu visse e aprendesse. Porém, era como acompanhar uma corrida de Fórmula-1 no meio da reta de chegada. Ele digitava rápido, falava rápido e com seu vasto conhecimento, transmitia muito mais informações do que minha mente leiga era capaz de absorver naqueles poucos minutos. Quando ele voltava para sua mesa, então começava meu lento processo de assimilação. Primeiro, eu respirava fundo, sabendo que as próximas horas iriam fatigar os neurônios. Com aquele semblante murcho de quem não sabe nem por onde começar, eu dava um history para ver quais eram todos aqueles comandos estranhos que ele havia digitado. E eram muitos.
> (JARGAS, 2008, p. 19).

Compreender este contexto, resulta em entender um dos motivos que faz do Windows, um sistema comercial e de código fechado, ser mais popular do que o Linux, um sistema sem custos e de código aberto.

1.4 Objetivo do Trabalho

1.4.1 Objetivo Geral

É a disponibilização de duas ferramentas, desenvolvidas em filosofias diferentes, com recursos diferentes. Interligadas propositalmente para forçar a utilização do terminal e posteriormente facilita-la e melhorar a visualização através da interface web.

O objetivo inicial era apenas comprimir vários scripts em um único arquivo, para disponibilizar recursos semelhantes a um Painel de Controle ou Configurações do Sistema.

Didaticamente mostrar como interagir com o Kernel do *S.O* Linux através de comandos padrões das distribuições.

1.4.2 Objetivo Específico

- Compartilhar conhecimento.
- Documentar toda a programação, de forma mais intuitiva e acessível.
- Ser compatível com qualquer distribuição derivada do Debian.
- Feita para rodar em distribuições server e desktop.
- Ser acessível remotamente de qualquer *S.O* ou dispositivo.
- Ser robusta com finalidade didática. Ou seja, *KISS* com objetivo *HUG*.
- A interface shell script foi totalmente estruturada, pensando na mobilidade dos arquivos, e aproveitando a velocidade de interpretação dos scripts.
- Compatibilidade é uma das metas, mesmo que esteja sendo executado em uma distribuição não derivada do Debian, ao mínimo parcialmente o script irá funcionar.
- A interface web é orientada à objetos, aproveitando os recursos da linguagem de programação *PHP*, explorando a função “shell_exec”.
- Disponibilizar o acesso remoto de qualquer dispositivo ligado à rede, utilizando tecnologia responsiva é capaz de funcionar em computadores, tablets e celulares.
- Ter vantagens se comparada a ferramentas já existentes, como o acesso “ssh” e o “vnc”.

1.5 Metodologia

1.5.1 Estudo de Caso

Para o desenvolvimento do trabalho foi aproveitado a documentação feita durante o período de 2010 à 2013, enquanto o próprio autor trabalhou no T.I das Lojas Berlanda, onde havia a necessidade de utilizar muitos dos comandos existentes neste trabalho. Na época com a inexistência destas ferramentas muito tempo era perdido explicando o que era, quando utilizar e como utilizar determinado comando para obter um resultado necessário. Também serviu como fonte de inspiração, consulta e ajuda os constantes acessos ao fórum do website Viva o Linux.

O trabalho foi dividido em três etapas, citadas:

1) Levantamento dos comandos utilizados na empresa durante o Suporte: ERP, TEF e Redes;

2) Seleção de comandos e estruturação da interface background;

3) Codificação em Shell Script e *PHP*;

Importante é destacar que a mesma necessidade de automatizar os comandos para agilizar o suporte pode ser aplicado à qualquer empresa com computadores Linux, e também em qualquer outro caso, até mesmo em uma rede doméstica de 2 computadores, teria sua utilidade.

Por vezes o caso aparece-nos pela frente, e sentimo-nos obrigados a tomá-lo como objeto de estudo. Isso acontece quando um professor decide estudar um aluno em dificuldades, quando sentimos curiosidade por determinados procedimentos, ou quando decidimos avaliar um programa (STAKE, 2005, tradução nossa).

1.5.2 Estudo Bibliográfico

Para a realização deste trabalho foram pesquisados em livros de programação Shell Script e *PHP*. Livros sobre a Filosofia Unix e os sistemas operacionais Linux. Levou-se em consideração ergonomia, robustez, segurança e didática para que os futuros utilizadores possam desfrutar dos recursos e inteirações disponibilizadas.

1.5.3 Cronograma

O seguinte cronograma foi utilizado para o desenvolvimento deste trabalho.

Atividades	2013							
	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Determinação do tema								
Revisão Bibliográfica								
Testes								
Codificação								
Desenvolvimento do TCC								
Entrega do TCC a Banca Avaliadora								
Apresentação do TCC a Banca Avaliadora								

Tabela 1. Cronograma do TCC.

1.5.4 Estrutura do trabalho

Para o desenvolvimento deste trabalho foram cumpridas as seguintes etapas: pesquisa de material bibliográfico, revisão bibliográfica, implementação do software e testes.

Foram feitas pesquisas em livros para escrever este trabalho, a programação e desenvolvimento foi feita com pesquisas em web sites como o Viva o Linux, entre outros.

Na primeira fase foi realizado um levantamento bibliográfico, coletando os dados e informações necessários para o início do trabalho. Na revisão bibliográfica fundamentou-se teoricamente e proporcionou embasamento teórico, justificando os tópicos abordados no TCC e o padrão de processo adotado para o desenvolvimento do sistema.

Para implementação e testes do sistema foi utilizado a linguagem Shell Script, com o interpretador de comandos *bash*, homologada em distribuições oriundas do Debian. Foi verificado problemas de compatibilidades no Debian, Ubuntu e Linux Mint.

Hardware utilizado:

- Notebook para implementação e documentação;
- Processador Intel Core™ i5-3230M, 2,60GHz;
- 4 GB de memória *RAM*;

Software utilizado:

- Sistema Operacional Ubuntu 13.04, Raring Ringtail;
- Eclipse;

II. FUNDAMENTAÇÃO TEÓRICA

2.1 História do Unix e GNU/Linux

Muitas características presentes nos *S.O* derivados do Unix são resultantes do curioso percurso que definiu sua base tão robustamente, sendo desnecessário substitui-la com o passar do tempo.

A ideia de software livre de Richard Stallman (formado em Física pela Universidade de Harvard - EUA) nasceu quando ele ainda era integrante de um grupo de hackers do laboratório de inteligência artificial do MIT. Stallman utilizava uma impressora para colocar no papel os códigos de programação que escrevia e resolvia quaisquer problemas eventuais com o equipamento, pois era conhecedor de seus códigos. Quando a impressora foi substituída, Stallman pediu ao fabricante do novo equipamento os códigos-

fontes e não recebeu resposta positiva. Assim, iniciou uma busca por tornar acessíveis os códigos guardados secretamente pelos fabricantes. Surgiu então a idéia de software livre e da Free Software Foundation (Fundação do software livre, comumente referenciada como *FSF*). Em meados de 1984, Richard Stallman iniciou seus trabalhos em uma parte do projeto GNU, pretendendo criar um grupo de livre compartilhamento de software, pois acreditava que se ele gostasse de um programa, precisava compartilhar o mesmo com outras pessoas que também gostavam dele. O projeto GNU tinha como objetivo a criação de um sistema operacional que fosse compatível com o padrão Unix. Este fato ia de encontro aos direitos autorais do Unix, motivo pelo qual a *FSF* criou uma licença denominada GPL (General Public Licence – Licença Pública Geral). A licença tinha como base a livre distribuição do software, o direito ao estudo, à modificação e ao aperfeiçoamento por quaisquer pessoas, sem que fosse exigido pagamento de licenças. Um novo sistema operacional estaria sendo criado pelo projeto GNU, seguindo os requisitos da GPL (GOMES JUNIOR, 2007).

Mike Gancarz, descreve em seu livro Linux and the Unix Philosophy sobre a NIH Sydrome (Síndrome, do não ser convidado aqui) que explica os fundamentos de um Software baseado em outro, desambiguando o paradoxo existente entre softwares com finalidades Científicas (no livro citadas como softwares de "aperfeiçoamento e extensão") e cópias sem fundamentos, melhor detalhada respectivamente.

> A Síndrome de "não ser convidado aqui" é caracterizada pela decisão de descartar tudo o que outro desenvolvedor já criou com a intenção de demonstrar uma solução superior. Isto demonstra um pequeno ato de egoísmo, interesse em preservar o melhor do trabalho alheio, usando-o como uma base para um novo resultado. Isto não é apenas se auto-ajudar, isto desperdiça um tempo precioso na reescrita, tempo que poderia ser mais bem gasto se estivesse criando outras soluções. Na pior das hipóteses, a nova solução é muitas vezes meramente melhor ou pouco diferente, e este é o problema.
> Ocasionalmente, a nova solução é melhor porque o desenvolvedor viu o trabalho do desenvolvedor original e era apto para melhorá-lo, aproveitando o que é bom e descartando o resto. Isto é o caso de aperfeiçoamento e extensão, não a NIH síndrome.
> (GANCARZ, 2003, pág. 2, tradução nossa).

Tendo ciência de outros fatores históricos, não só o desenvolvimento do Unix mas de vários Softwares que foram criados através da colaboração de vários programadores, em

forma de comunidade, aproveitando e aprimorando códigos já existentes sem ter ciência do objetivo inicial de determinada ideia. Este contexto também pode ser aplicado ao desenvolvimento do Unix, do Linux, da Linguagem de programação C e até de Jogos como DOOM que foi desenvolvido inteiramente na linguagem C.

> As pessoas ficam maravilhadas com a portabilidade do Unix, mas nem sempre foi assim. Thompson escreveu o código original na linguagem Assembly, em 1972 ele reescreveu com uma linguagem chamada "B", isto só se tornou evidente, devido o avanço do hardware que anteriormente não estava disponível. Outro membro do AT&T's dos Laboratório Bell, Dennis Ritchie, fez extensões e modificações na B em 1973, aproveitando isto na linguagem C, amada e desprezada por programadores ao redor do mundo.
> (GANCARZ, 2003, pág. 3, tradução nossa).

2.2 Computadores pessoais e o GNU/Linux

De 1981 (ano de lançamento do primeiro computador pessoal, pela IBM) a 1984, foram comercializados mais de 250 mil computadores pessoais. As projeções de analistas pontavam para aproximadamente 80 milhões de computadores pessoais até finais do século XX, número que foi absurdamente maior, chegando a 500 milhões de PCs (Personal Computers – Computadores Pessoais) vendidos até o ano 2000. No início da década de 90, o desenvolvimento de aplicações gráficas facilitou o uso dos PCs (também chamados desktops) por parte de quaisquer pessoas e permitiu a sua introdução nos lares (GOMES JUNIOR, 2007).

Com o passar dos anos o Linux se tornou mais presente nos PCs de todos os tipos de usuários no mundo! Este número só tende a crescer, com o aumento dos preços dos *S.O* concorrentes e a constante evolução das distribuições Linux. Algumas ainda tem seus foco voltado para servidores, como o Red Hat e o Debian, ambas começaram e continuam voltadas a este nicho de mercado; o que não impossibilitou distribuições oriundas destas, serem focadas em Desktops, como o Fedora e Linux Mint respectivamente.

2.3 A importância do GNU/Linux na Internet

Na década de 80, o Unix passou a ser comercializado. Neste mesmo período a ARPA (Advanced Research Project Agency - Agência de Projeto de Pesquisa Avançada), foi designada pelo Departamento de Defesa dos EUA para criar uma rede de computadores resistente a ataques militares. Batizada de ARPANET, utilizava o protocolo *TCP/IP* para troca de dados, este era montado em Unix. Com os constantes avanços, a ARPANET tornou-se a Internet.

Com a Internet, outros *S.O* foram introduzidos e utilizados para diversas funcionalidades, substituindo parcialmente o império do Unix no ambiente web.

Os sistemas Unix e seus derivados são confiáveis e muito famosos por sua segurança, e atualmente ainda são muito utilizados, principalmente em servidores que rodem algum serviço de rede que exija estes requisitos, como por exemplo: e-mail, websites, gateway para sistemas de *ERP*, *TEF*, etc. Imagine tentar realizar uma compra com um cartão de crédito, e ao invés de "transação aprovada", receber mensagens como: "desculpe, nosso servidor Windows está com vírus e fazendo as atualizações automáticas, volte mais tarde!".

2.4 Desvantagens do WINDOWS

O Linux é um sistema muito robusto se comparado ao Windows. No Windows utilizando o executar ou acessando através do atalho "Windows + R", para abrir as Configurações do Sistema.

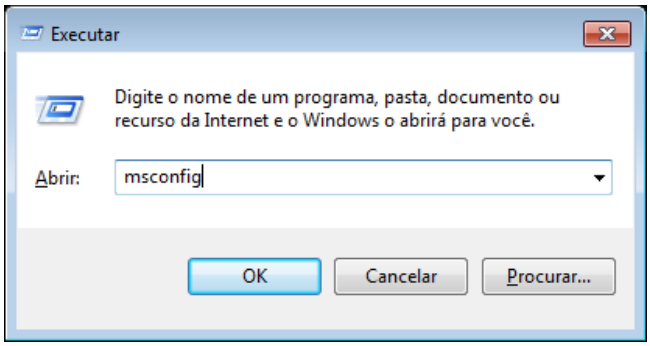

Figura 2 – Executar no Windows
Fonte: Próprio Autor

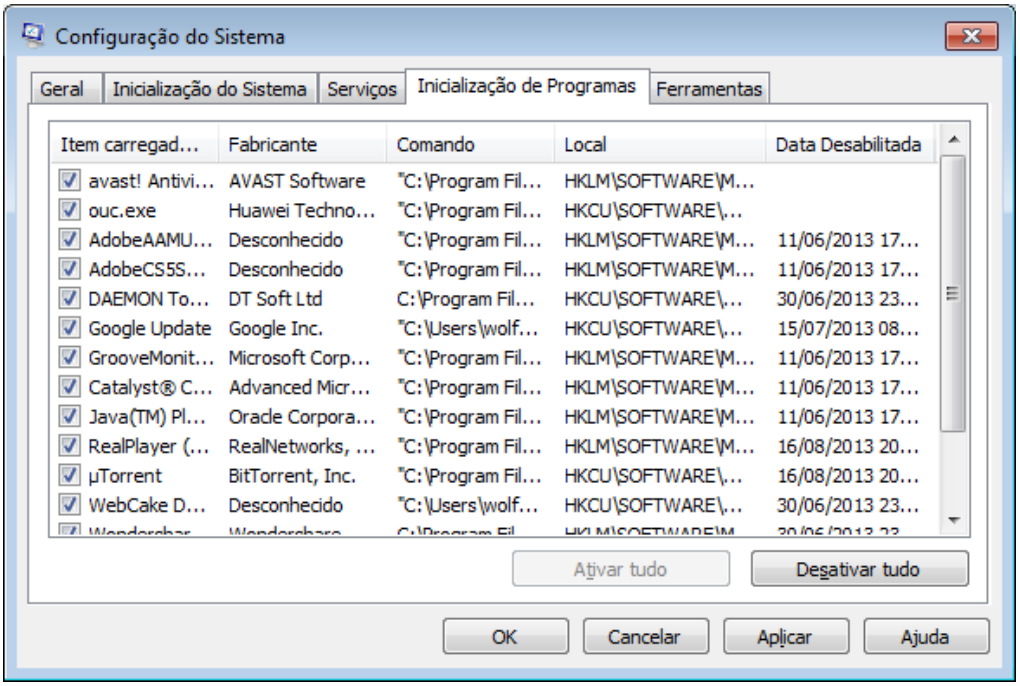

Figura 3 – Tela do msconfig
Fonte: Próprio Autor

Neste exemplo, na aba "Inicialização de Programas" pode ser definido quais programas irão ser executadas ao iniciar o *S.O*, observa-se a quantidade de programas habilitados, muitos deles desnecessariamente, o que afeta diretamente no desempenho do processador e uso de memória *RAM*.

O início do mercado de navegadores ou browsers se deu a empresa Netscape, que com uma pequena equipe teve um papel muito importante no desenvolvimento da Internet e da Ciência da Computação, porém conforme é relatado no filme "A Guerra dos Navegadores", a Microsoft, empresa de Bill Gates, iniciou várias perseguições até achar uma solução para acabar com o modesto navegador de licença comercial da Netscape, utilizando o famoso Internet Explorer. De início a ideia não obteve sucesso, pois o navegador também tinha licença comercial e um desempenho não satisfatório, porém com a jogada de distribuir seu navegador instalado como "brinde" ao adquirir o Windows, a Microsoft dominou o mercado dos navegadores.

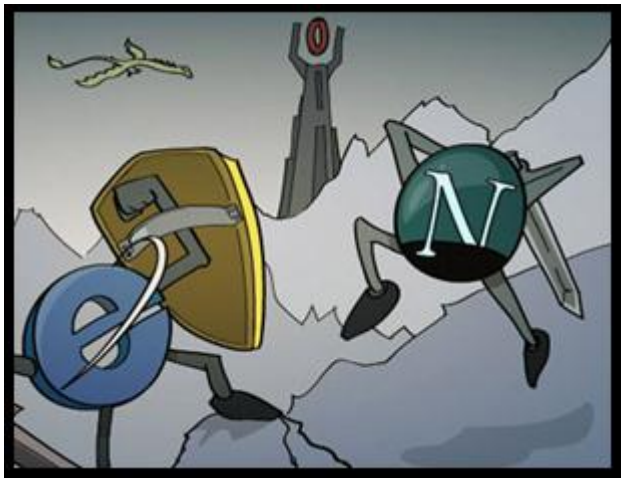

Figura 4 – Arte que representa o conflito entre as duas empresas e seus navegadores.
Fonte: <http://thecssninja.com/talks/remote_debugging/assets/ie-vs-netscape.jpg3>

Mesmo com fatos como este, o Windows continua sendo o *S.O* mais utilizado no mundo, um simples exemplo disto é sua própria tecla presente na maioria dos teclados fabricados atualmente, inclusive Notebooks, Netbooks e os novos Ultrabooks.

Figura 5 – Tecla Windows
Fonte: <http://imguol.com/2013/01/25/tecla-windows-1359134593299_956x500.jpg>

A mais recente façanha da Microsoft para se manter no topo deste nicho de mercado foi a criação do recurso de UEFI, gravando o número serial do computador/*S.O* em hardware. UEFI (Unified Extensible Firmware Interface) é uma interface de firmware padrão para PCs, projetada para substituir o BIOS (basic input/output system). Ela promete aumentar o desempenho de carregamento do sistema (boot), e está presente na maioria dos novos notebooks que vem com o Windows 8 de fábrica. Até ai tudo bem, porém este recurso possui uma característica peculiar: impedir qualquer tipo de boot no sistema, a não ser o do próprio Windows 8! Isto mesmo, sem boot pelo pendrive, sem Dual Boot. Então, as únicas formas de

instalar *S.O* concorrentes simultaneamente em um computador com UEFI, é considerada ilegal.

O sistema operacional comercial mais estável e renomado atualmente é o MAC *OS*, que também foi desenvolvido em cima do Kernel Unix.

2.5 Trabalhar com a Linha de Comando

Todo sistema computacional requer um componente de interface humana. Para a administração de sistemas Linux, normalmente se usa uma interface de texto. O sistema apresenta ao administrador um prompt, que na sua forma mais simples é um único caractere, como $ ou #. O prompt significa que o sistema está pronto para aceitar comandos digitados nele, os quais normalmente ocupam uma ou mais linhas de texto. Essa interface é chamada genericamente de linha de comando. É responsabilidade de um programa chamado shell fornecer o prompt de comando e interpretar os comandos. O shell fornece uma camada de interface entre o kernel do Linux e o usuário final, o que é a origem de seu nome. O shell original para sistemas Unix foi escrito por Steve Bourne e se chamava simplesmente *sh*. O Shell padrão do Linux é o *bash*, o Bourne-Again Shell, que é uma variante GNU de *sh* (HEADER, 2012).

Assim como no exército, um soldado responde “senhor, sim senhor” e fica submisso a ordens de seu superior, e as executa sem questionar. No computador o prompt é o contexto do soldado em estar pronto, o kernel é o objetivo, o usuário final é o superior. O shell é a conexão entre a ação e a inércia resultante das ordens do superior, para atingir determinado objetivo no kernel. Caso o comando seja informado incorretamente, o shell nem o entrega ao kernel, pois sabe que o mesmo não irá executar a tarefa, portanto o primeiro passo para ver este contexto funcionando, a tornar o ambiente favorável para o trabalho, é o aprendizado de conhecer os comandos, suas formas de inteiração para posteriormente se comunicar com o kernel ou sistema.

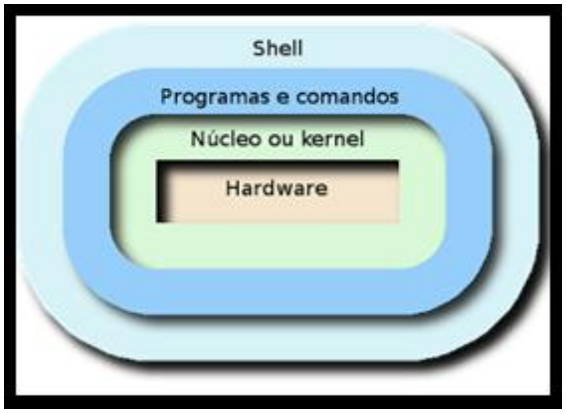

Figura 6 – Estrutura do GNU/Linux
Fonte: <http://1.bp.blogspot.com/_0mtan84bpiI/Sp__eXvGWSI/AAAAAAAAAOU/bzmSiL9RxdU/s320/grafico.png>

Os comandos possuem uma sintaxe relativamente intuitiva, mas é necessário dominar o inglês para situar-se facilmente, por exemplo, o "ls", comando de listagem, abreviação da palavra "list" em inglês. Pode ser utilizado neste mesmo contexto, para listar todos os arquivos dentro de um diretório, este diretório e todo seu conteúdo está gravado no HD, ou seja, o usuário, através do prompt executa o comando de listagem, o *bash* interpreta e envia para o shell, que interpreta e entrega ao kernel, que faz a leitura no HD (Hard Disk ou Disco Rígido) e disponibiliza a informação no caminho contrário (comunicando usuário e máquina).

2.6 PHP e Bootstrap

Figura 7 – PHP
Fonte: < http://www.revista.espiritolivre.org/wp-content/uploads/2013/03/19-03-2013_php-logo.jpg>

Escolhida para o desenvolvimento da interface Web (Diagnostic web helper shell), assim como o Shell Script, é uma linguagem de script, portanto permite a execução de comandos, porém orientada a objetos, e não somente estruturada. No contexto do trabalho, é usada de para acesso externo ao Linux, inserindo comandos remotamente com a função "shell_exec": específica para a necessidade de acesso limitado, sem comprometer a segurança do computador hospedeiro. Tornando possível o acesso por qualquer dispositivo que possua

navegador, utilizando para isto, a tecnologia responsiva do framework de *CSS* Twitter Bootstrap, além de outros recursos presentes em sua folha de estilos. É importante ressaltar, que assim como Linux e o Shell Script, o Bootstrap e *PHP* também são free, portanto ferramentas não pagas que permitem livre alteração de código.

Figura 8 – Bootstrap
Fonte: < http://m5designstudio.com/wp-content/uploads/2013/04/bootstrap_responsive_layout.png>

2.7 Shell Script

O shell é um poderoso ambiente de programação capaz de automatizar quase qualquer coisa que vocês puder imaginar no seu sistema Linux. O shell é também a sua interface interativa com o seu sistema. Quando você inicia um shell, ele executa algumas tarefas automáticas para ficar pronto para o uso e, então, apresenta um prompt de comando. O prompt de comando lhe diz que o shell está pronto para aceitar comandos a partir do seu dispositivo de entrada padrão, que geralmente é o teclado (HEADER, 2012).

São junções de comandos de forma intuitiva, cada comando Linux é uma abreviação da palavra em Inglês com igual significado, para executar determinada orientação ao *S.O*, semelhante ao DOS. Porém vários outros fatores tornam o shell muito mais complexo e poderoso.

Cerca de 80 % dos códigos contidos em ambas as ferramentas deste trabalho de conclusão de curso, são em shell script. O desenvolvimento foi muito demorado e sua codificação começou em 2010, por se tratar de uma sintaxe complexa. Simples se comparada a linguagens como Java que é multiplataforma e exige muito tempo de estudo. Porém possibilita uma inteiração mais rápida e eficaz com o sistema operacional Linux, recursos que nenhuma outra linguagem possibilita.

Seu início foi baseado na criação de scripts separados e sem ligação, com o tempo houve a necessidade de fundi-los de forma a criar uma ferramenta, com várias funcionalidades.

> Alguns scripts cresceram além do seu objetivo inicial, ficando maiores e mais complicados. Cada vez era mais difícil entrar o lugar certo para fazer as alterações, tomando o cuidado de não estragar seu funcionamento. Outros técnicos também participavam do processo de manutenção, adicionando funcionalidades e corrigindo bugs, então era comum olhar um treco novo do código que demorava um bom tempo até entender o que ele fazia e por que estava ali. Era preciso amadurecer. Mais do que isso, era preciso fazer um trabalho mais profissional.
> (JARGAS, 2008, p. 20).

Com o avanço das funcionalidades das ferramentas, houve a necessidade de refinação da codificação, trazendo novos desafios organizacionais e funcionais. Felizmente e também infelizmente, uma linha de comando errada não impede o programa de ser compilado, diferente da maioria das linguagens de programação. Como diz o nome é um script, então se a lógica for crescente, um erro na primeira linha pode comprometer o funcionamento das demais, mas não impede a compilação e execução.

Por se tratar de uma programação estruturada, o nível organizacional não é tão complexo como na orientação à objetos. Mas fatores como o uso de pipes, caracteres especiais, parâmetros específicos de cada comando, entre outros elementos, tornam possível executar indeterminadas tarefas em uma única linha de comando.

> Os scripts codificados com pressa e sem muito cuidado com o alinhamento e a escolha de nomes de variáveis estavam se tornando um pesadelo de manutenção. Muito tempo era perdido em análise, até realmente saber onde e o que alterar. Os scripts de “cinco minutinhos” precisavam evoluir para programas de verdade. Cabeçalhos informativos, código comentado, alinhamento de blocos, espaçamento, nomes descritivos para variáveis e funções, registro de mudanças, versionamento. Estes eram alguns dos componentes necessários para que os scripts confusos fossem transformados em programas de manutenção facilitada, poupando nosso tempo e, consequentemente, dinheiro.
> (JARGAS, 2008, p. 21).

2.8 Segurança

Este é um fator que o GNU/Linux disponibiliza em alta qualidade de serviço por padrão, devido seu sistema de arquivos, seu sistema de permissões, e a própria estrutura do shell, tornam seus *S.O* robustos e seguros por natureza. Se você instala um destes, obviamente não há necessidade de um antivírus, pois a estrutura não permite que programas mal intencionados, classificados como vírus, tão famosos no Windows, de se alojar, auto executar e replicar automaticamente. Resumindo, se o usuário opta por qualquer distribuição Linux, qualquer distribuição MAC ou Unix. Não é possível pegar vírus!

Porém, isto não impede que um script malicioso seja executado incorretamente, existem várias combinações de comandos que dão resultados semelhantes ao de um vírus, exemplificar uma função que causa um estouro de memória temporária, ou seja, enche sua *RAM* até o *S.O* travar, facilita o entendimento. Exemplo: **:(){ :|:& };:**

Estranho, mas é verdade, este monte de "smiles" é uma função que ao ser executada por um script, faz o que foi citado acima. O que não impede dela ser adicionada em qualquer parte de determinado script... Obviamente, os scripts deste trabalho não são maliciosos, portanto não há risco nenhum de usa-los.

Este item aborda uma das preocupações deste trabalho, na interface em shell script, o nível de manipulação feita pela ferramenta é maior que na interface web, pois na interface background é necessário alterar permissões, remover arquivos, copiar arquivos, executar scripts. Ou seja, um nível quase máximo de permissões é necessário para realizar determinadas tarefas, através do usuário de root ou execuções de comandos por "sudo".

Já na interface web, um novo desafio foi enfrentado: disponibilizar uma forma de acesso remoto que seja segura, e tenha alguma vantagem se comparada as ferramentas existentes como VNC e SSH. Apenas com acessos de execução e visualização, é possível obter informações do kernel, software e hardware do computador hospedeiro. Para isto foram bloqueados os comandos de cópia(*CP*), movimentação(*MV*) e remoção(*RM*) de arquivos, então nesta interface o usuário pode literalmente "ver tudo, mas não alterar nada".

Segue, na Figura 9, o modal que é apresentado utilizando recursos de JavaScript, *CSS*, e *PHP*, apresentada quando a função de segurança é ativada ao reconhecer que o comando inserido possuí as strings respectiva aos comandos bloqueados e logo bloqueia a execução, portando o comando não apresenta nenhuma modificação no prompt, pois foi bloqueado antes de sua execução, o mesmo acontece a função que interpreta a conexão entre o *PHP* e o

Kernel(núcleo do sistema) do Linux. Neste caso exemplificado, o comando executado removeria todos os arquivos na pasta hospedada, ou seja, na /var/www/shell onde está o Index.php, Error.php, dentre todos os outros. O mesmo poderia ser feito para qualquer outra pasta com permissões no computador hospedado, ou seja, ser crackeado(“hackeado”). Esta função e as alterações de segurança feitas na instalação desta ferramenta, garantem a segurança para todos os tipos de conexões.

Figura 9 – Modal de bloqueio
Fonte: Próprio Autor

III. PROJETO

As duas ferramentas aqui desenvolvidas foram criadas e testadas no *S.O* Debian Linux e distribuições baseadas no mesmo, como o Ubuntu. A utilização das mesmas é possível em outras distribuições, porém não há garantia de bom funcionamento.

Figura 10 – Logo da distribuição Debian
Fonte: <https://wiki.videolan.org/images/Debian-logo.jpg>

A hospedagem da interface web, se utilizada juntamente com a interface background deve seguir as mesmas regras. Porém a interface web foi feita para funcionar de forma independe, portanto é possível sua instalação/hospedagem e utilização em qualquer plataforma, inclusive em Windows.

3.1 Interfaces do Linux helper shell script

Aqui serão mostradas as interfaces em background da ferramenta em shell script, desenvolvida para funcionar em um único arquivo de script totalmente estruturado, que funciona basicamente como interface entre usuário e kernel, executando os comandos desejados dentro da sintaxe, de forma automática. Automatizando determinadas tarefas de forma robusta e eficaz. Através da seleção de opções em cada menu, a ferramenta realiza a respectiva funcionalidade automaticamente.

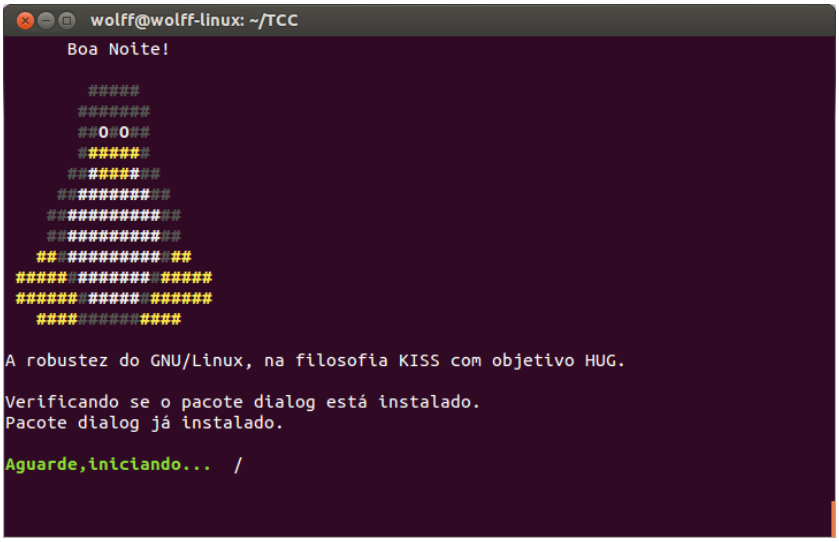

Figura 11 – Tela de boas vindas
Fonte: Próprio Autor

A figura 11 apresenta a tela exibida através do terminal GNU/Linux, a interface é 100 % background, portanto em modo texto. Nela o script automaticamente verifica qual é a hora e faz a saudação para o usuário, verifica se o pacote "dialog" está instalado, e se necessário instala o mesmo automaticamente, pois ele é necessário para apresentar algumas telas.

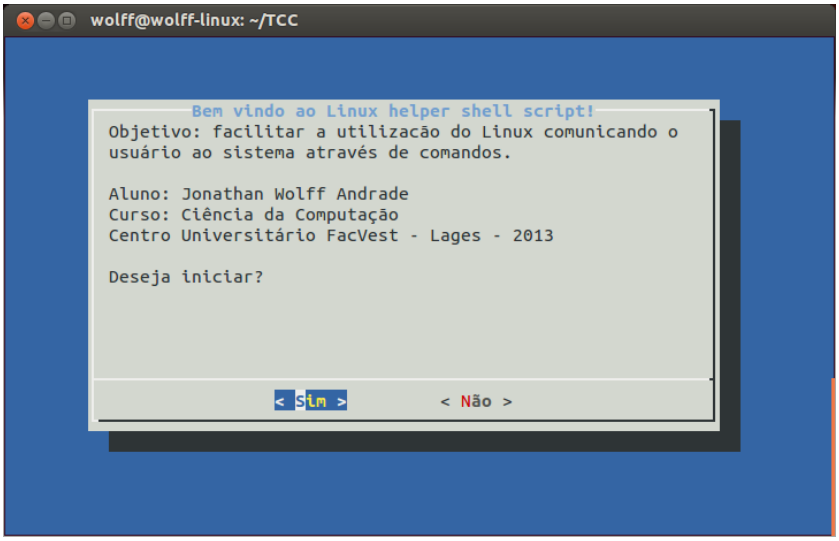

Figura 12 – Tela de apresentação
Fonte: Próprio Autor

A figura 12 mostra a tela de apresentação, nela o pacote "dialog" já está sendo executado e mostrando informações sobre o trabalho acadêmico. Há opção de continuar a execução ou encerrar.

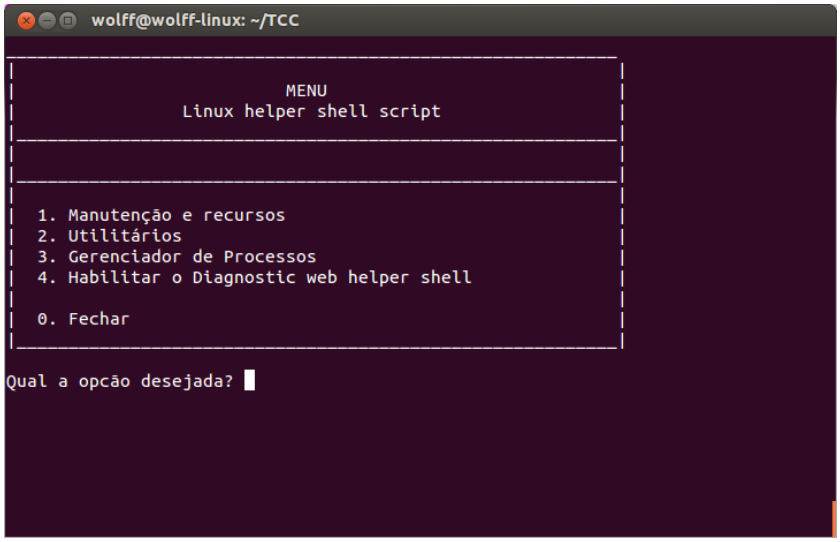

Figura 13 – Tela de menu principal
Fonte: Próprio Autor

A figura 13 apresenta a tela do menu principal da ferramenta, nela é possível optar por entrar em um dos Menus secundários, ou habilitar o Diagnostic web helper shell. Neste e nos outros menus sempre há a opção 0, apenas neste caso ela encerra a execução. E nas demais redireciona para o menu anterior.

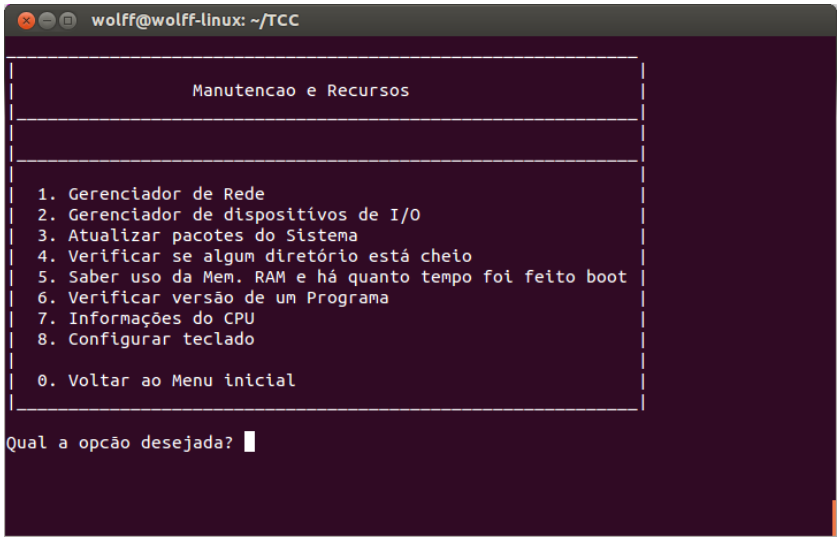

Figura 14 – Tela do menu Manutenção e Recursos
Fonte: Próprio Autor

A figura 14 apresenta a tela do menu secundário de Manutenção e Recursos, nela há um atalho para o Gerenciador de Rede, Gerenciador de dispositivos de I/O, dentre outras funcionalidades como verificar o Load Average e informações do processador.

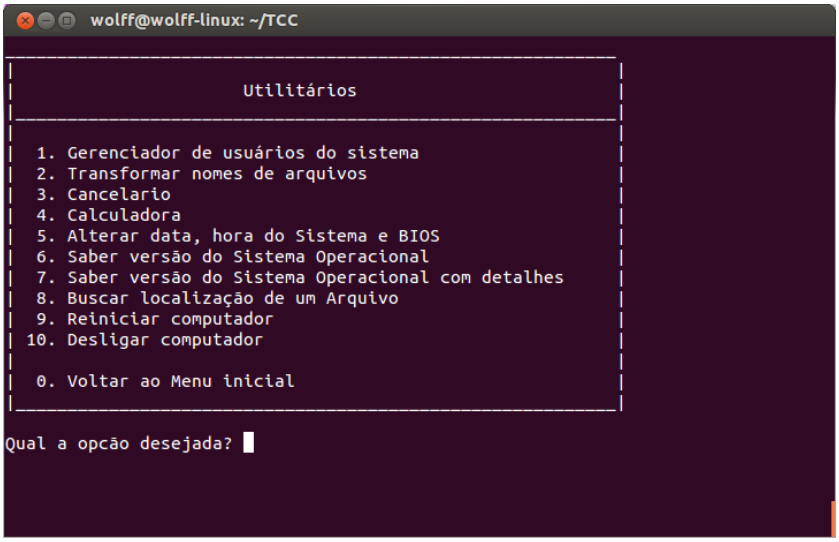

Figura 15 – Tela do menu Utilitários
Fonte: Próprio Autor

A figura 15 apresenta a tela do menu secundário de Utilitários, nela há um atalho para o Gerenciador de usuários do sistema, dentre outras funcionalidades de miscelânea, como saber a versão do *S.O* com detalhes, reiniciar ou desligar o computador, dentre outras.

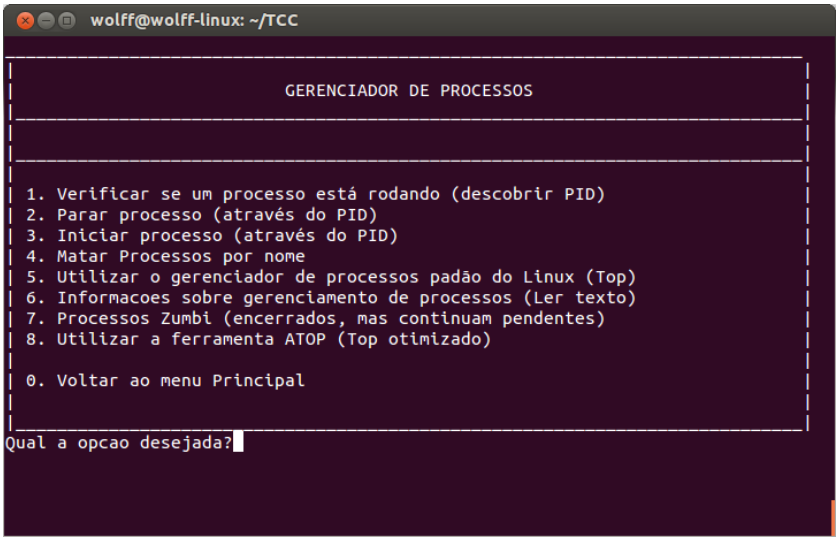

Figura 16 – Tela do menu Gerenciador de processos
Fonte: Próprio Autor

A figura 16 apresenta a tela do menu do Gerenciador de processos, várias opções semelhantes ao Gerenciador de Processos do Windows, é possível matar, parar e descobrir informações de processos/programas que estão sendo executados. Também há um link para as ferramentas Top e Atop, que fazem as mesmas funções de forma mais complexa e poderosa.

Este é o único menu que possuí uma documentação embutida, a opção 6 possibilita a visualizações de informações sobre este menu, sobre os processos do Linux, dentre outros dados importantes e relacionados.

Figura 17 – Tela do Gerenciador de dispositivos de rede
Fonte: Próprio Autor

A figura 17 apresenta a tela do menu de Gerenciador de dispositivos de Rede, é um dos mais importantes e poderosos da ferramenta, possibilita configuração das interfaces de rede, visualização de informações, correções, monitoramento, dentre outras funcionalidades. É importante ressaltar a opção 11 de Tunelamento por SSH através do TSOCKS, pois trata-se de um método pouco conhecido, que vale a pena ser testado.

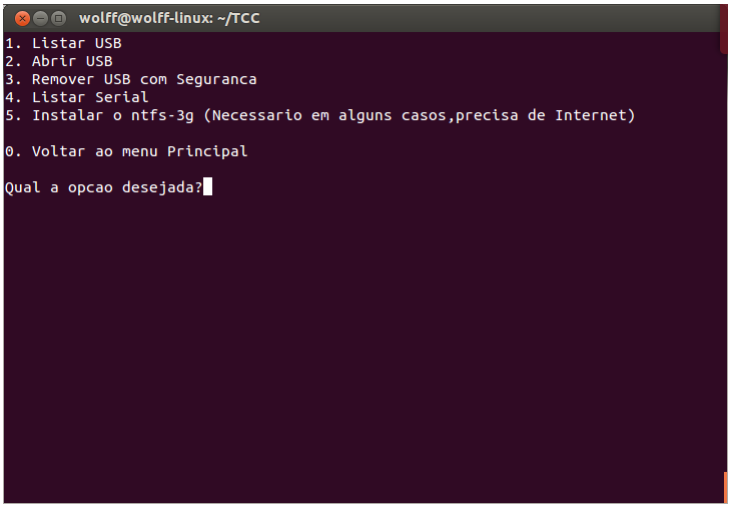

Figura 18 – Tela de dispositivos de I/O
Fonte: Próprio Autor

A figura 18 apresenta a tela de funcionalidades de dispositivos de I/O, ou seja, permite visualizar portas seriais, montar imagens de pen drives e manipula-las para armazenamento flash. Também a opção de instalação e configuração do ntfs-3g, pacote necessário em alguns casos.

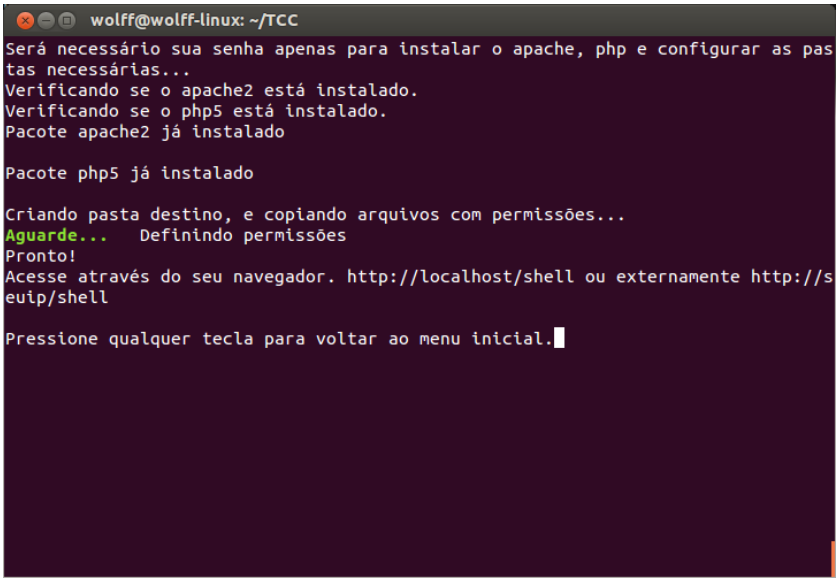

Figura 19 – Tela de configuração automática do Diagnostic web helper shell
Fonte: Próprio Autor

Ao selecionar a 4 opção do menu principal, o shell script automaticamente configura o Diagnostic web helper shell, sendo necessário informar a senha, apenas para instalar os pacotes necessários: apache2 e php5, sendo necessário estar conectado a internet pois os pacotes são instalados via “apt-get”, além de definir as permissões e proprietário dos diretórios utilizados com o comando “chmod” e “chown” respectivamente. Caso algum dos pacotes já se encontra instalado no sistema, o script pula os passos de instalação.

Após completar os passos, é orientado ao usuário acessar o resultado no navegador.

3.2 Interfaces do Diagnostic web helper shell

É a interface web do trabalho, focada em disponibilizar um breve diagnóstico do sistema Linux, com páginas desenvolvidas para serem utilizadas de maneira didática. É automaticamente configurada através do script, porém não necessariamente dependente do mesmo para funcionar. A codificação foi feita em *PHP*, possibilitando acesso remoto e visualização de informações do Linux externamente, de forma segura e robusta, com a função “shell_exec”, específica para este tipo de interação.

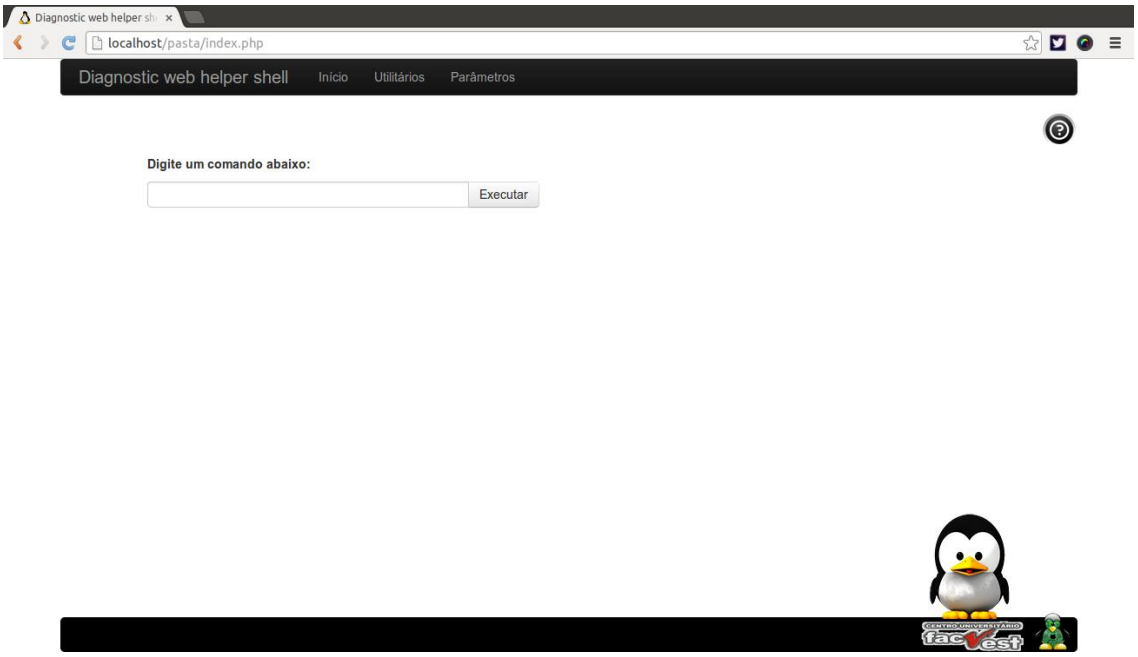

Figura 20 – Tela inicial
Fonte: Próprio Autor

A figura 20 apresenta a tela Inicial, como pode ser observado, ela possuí poucos elementos, tornando agradável o acesso.

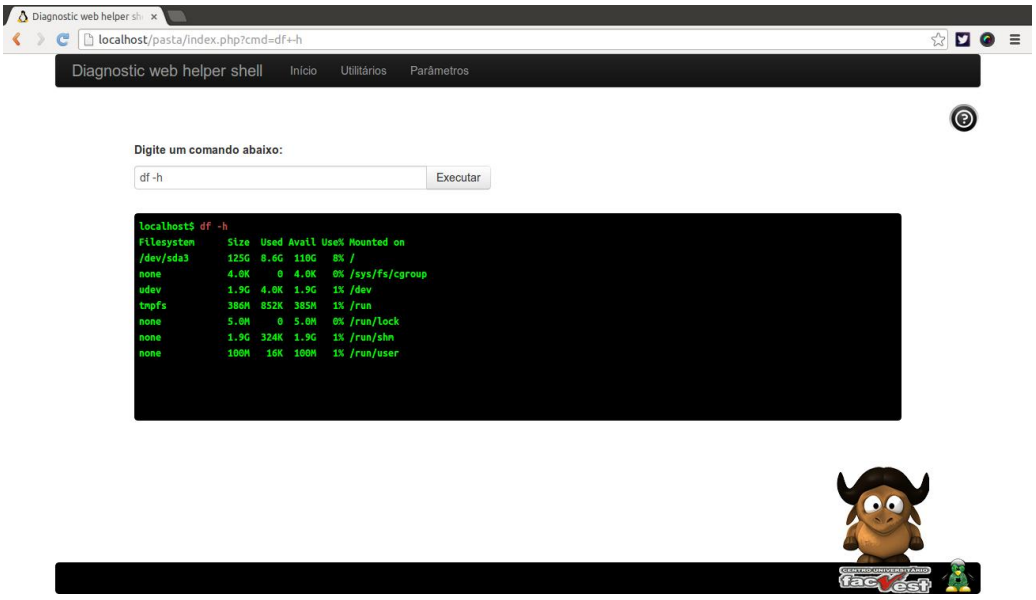

Figura 21 – Tela inicial com terminal ativado
Fonte: Próprio Autor

A figura 21 apresenta a tela Inicial, mesma da figura 20, porém com o terminal ativado, ou seja, um comando foi executado e o resultado exibido em formato de prompt. Qualquer comando pode ser executado nela, porém somente os liberados terão efeito.

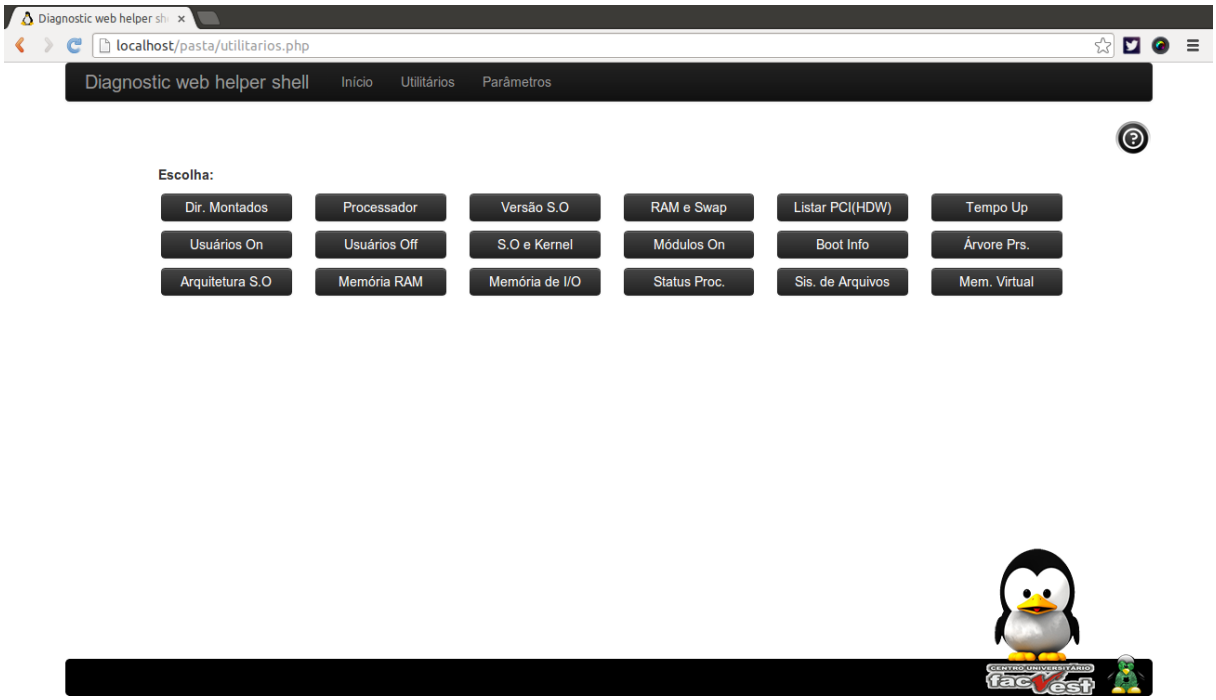

Figura 22 – Página Utilitários
Fonte: Próprio Autor

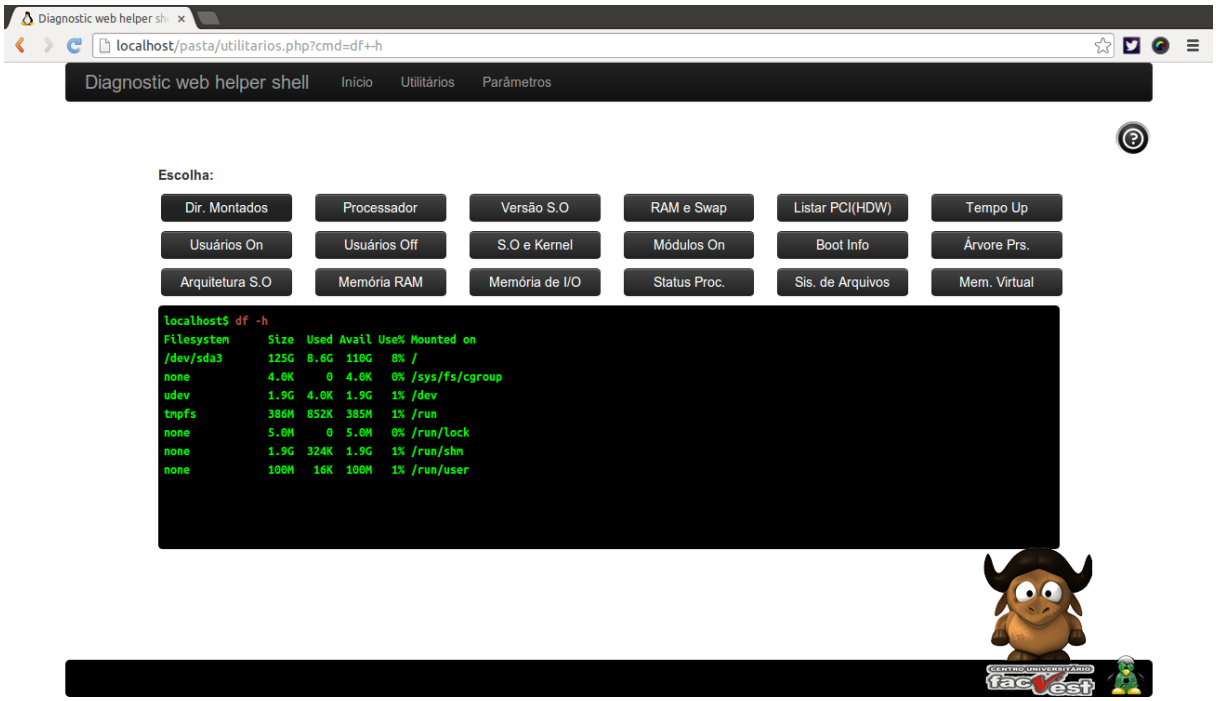

Figura 23 – Página Utilitários com terminal ativado
Fonte: Próprio Autor

A figura 22 apresenta a página Utilitários, como pode ser observado, ela possuí vários botões que executam várias comandos prontos, ao deixar o mouse sobre é exibido um texto complementar explicando a funcionalidade. Ao clicar em qualquer um deles, o resultado é exibido em formato de prompt conforme a figura 23.

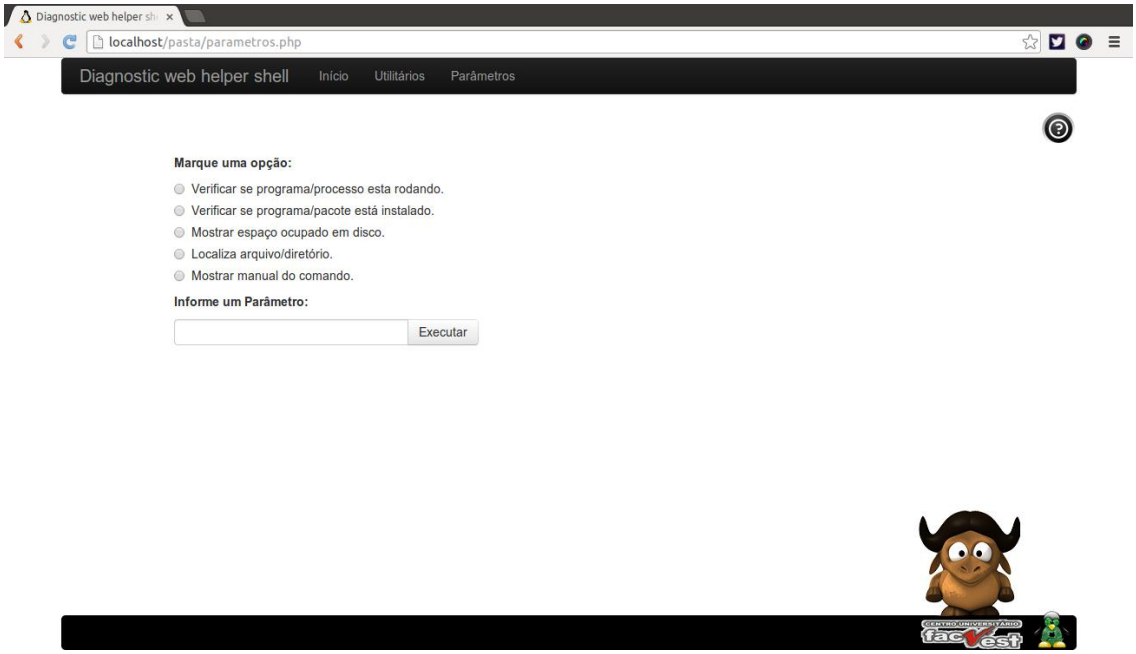

Figura 24 – Página Parâmetros
Fonte: Próprio Autor

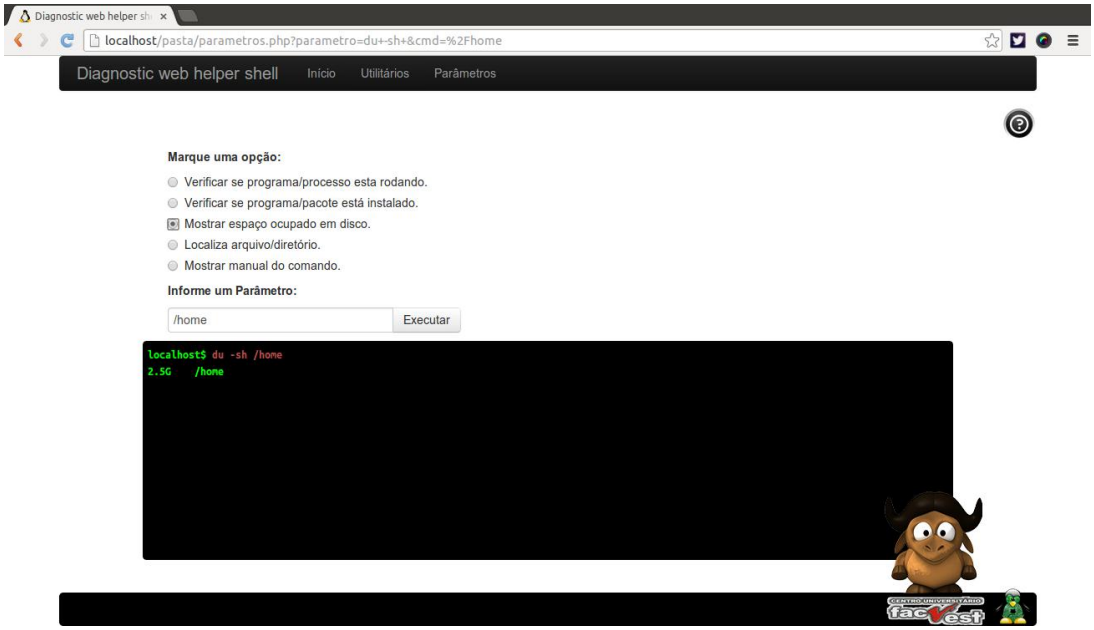

Figura 25 – Página Parâmetros com terminal ativado
Fonte: Próprio Autor

A figura 24 apresenta a página Parâmetros, como pode ser observado, ela exige a seleção de um dos radio buttons, que são as opções a serem passadas juntamente com o parâmetro, que é informado no campo abaixo, ao executar automaticamente ambas as informações são capturadas e transformadas para a sintaxe de comandos. Assim como nas outras telas, o resultado é exibido em formato de prompt conforme a figura 25.

Figura 26 – Recursos responsivos
Fonte: Próprio Autor

A figura 26 apresenta os recursos responsivos, derivados do Bootstrap, possibilitam conforme as imagens, acesso a ferramenta web de qualquer resolução de tela ou navegador, esta mesma figura faz parte da página de Tutorial, que é ativada ao clicar no ícone de interrogação, presente em todas as telas, representado na Figura 27. O mesmo direciona para a página citada, que ensina como utilizar cada recurso da ferramenta, de forma bem acessível e didática.

Figura 27 – Botão hover de acesso a página de Tutorial
Fonte: Próprio Autor

3.3 Caso de Uso

Aplicadas a UML, as duas interfaces possuem a mesma representação estática:

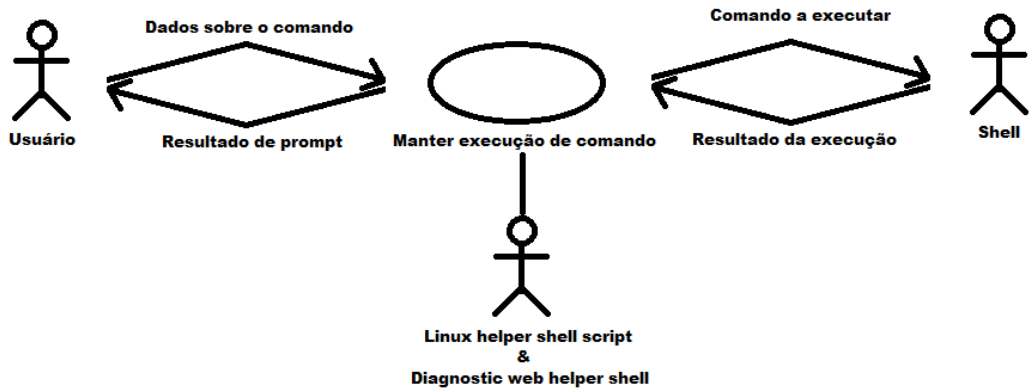

Figura 28 – Diagrama de caso de uso
Fonte: Próprio Autor

As interações também podem ser observadas em sequência para facilitar o entendimento, o diagrama é valido para as duas Ferramentas.

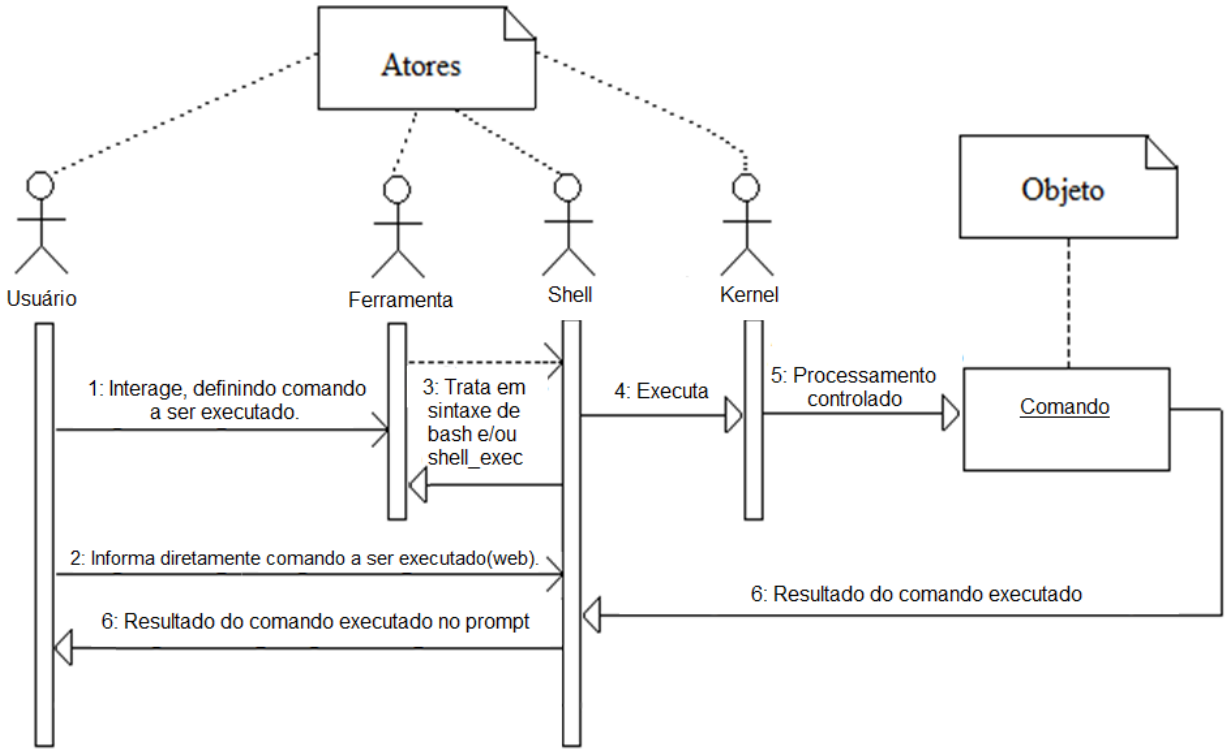

Figura 29 – Diagrama de sequência
Fonte: Próprio Autor

IV. CONSIDERAÇÕES FINAIS

Todos os recursos utilizados, e o resultado final das duas ferramentas criadas neste trabalho, ambos são de código aberto, com finalidades não comerciais, feitos para serem customizados por seus usuários livremente. É de vontade do Autor preservar estas prioridades, para contribuir com o desenvolvimento da Ciência da Computação aplicada neste contexto.

As ferramentas foram desenvolvidas para disponibilizar uma forma alternativa e didática de interagir com o Kernel dos *S.O* Linux através de comandos, busca diminuir mesmo que modestamente, o paradoxo existente entre as filosofias de desenvolvimento *HUG* e *KISS*, pois unidas podem disseminar ainda mais o GNU/Linux, e separadas só atrasam a conclusão deste objetivo. Assim como é desde o início da Ciência e da Tecnologia, a busca por conhecimento para obter facilidade, é uma das metas a serem mantidas por usuários que customizarem as ferramentas e o trabalho como um todo.

V. REFERÊNCIAS BIBLIOGRÁFICAS

NEGUS, Christopher. **Linux Bible 2010 edition**. Indianapolis: Wiley Publishing Inc., 2010.

GANCARZ, Mike. **Linux and the Unix Philosophy**. S.l: Digital Press, 2003.

FORD, Neal. **The Productive Programmer**. Sebastopol: O'Reilly Media, Inc., 2008.

STUTZ, Michael. **The Debian Linux Cookbook: Tips and Techniques for Everyday Use**. Berkeley: Group West, 2001.

SCHRODER, Carla. **Linux Networking Cookbook**. Sebastopol: O'Reilly Media, Inc., 2008.

HEADER, Adam; SCHNEITER, Stephen; PESSANHA, Bruno; STANGER, James. **Certificação Linux LPI: rápido e prático: nível 1: exames 101 e 102**. Rio de Janeiro: Alta Books, 2012.

GOMES JÚNIOR, Antônio. **A computação pessoal e o Sistema Operacional Linux**. Guariba: [s.n.], 2007.

JARGAS, Aurélio Marinho. **Shell script profissional**. São Paulo: Novatec, 2008.

MOTA FILHO, João Eriberto. **Descobrindo o Linux**. São Paulo: Novatec, 2007.

DALL'OGLIO, Pablo. **PHP: Programando com Orientação a Objetos**. São Paulo: Novatec, 2007.

WELLING, Luke; THOMSON, Laura. **PHP e MySQL desenvolvimento Web**. Rio de Janeiro: Elsevier, 2005.

MACHADO, Ângelo. **Neuroanatomia Funcional**. São Paulo: Atheneu, 2001.

STAKE, Robert. **Investigación con estudio de casos**. Madrid: Morata, 2005.

VI. ANEXOS

Código fonte do Diagnostic Web Helper Shell

index.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$funcoes = new Funcoes();
$display = new Display();
?>
<html>
<?php
$display->display_header();
?>
<body>
	<?php
	$display->display_menu();
	?>
	<div class="row">
		<!--offset1 da um span a esquerda, alinhagem... -->
		<div class="span10 offset1">
			<h5>Digite um comando abaixo:</h5>
			<form action="index.php" method="get">
				<div class="input-append">
					<input class="span4" id="appendedInputButton"
type="text"
						name="cmd" value="">
					<button class="btn"
type="submit">Executar</button>
				</div>
			</form>
			<?php
			if (!$funcoes->seguranca($_GET['cmd']) )
				$funcoes->erro();
			else{
```

```
                        //? esta conferindo se é verdadeiro e : se for
falso
                        //? Se foi setado, concatena com cmd(parametro+cmd)
: caso contrário mostra apenas cmd(cmd)
                        $_GET['cmd'] =
(isset($_GET['parametro']))?$_GET['parametro'] . $_GET['cmd']:$_GET['cmd']
;


                        //Mostra cmd
                        if ($_GET['cmd'] != ""){
                              echo "<pre class=\"linux\">";
                              echo
gethostbyaddr($_SERVER['REMOTE_ADDR'])."$ ";
                              echo "<span class='text-error'>" .
      $_GET['cmd'] . "</span><br>";
                              $cmd = shell_exec($_GET['cmd']);
                              echo $cmd;
                              echo "</pre>";
                        }
                  }
                  ?>


            </div>
      </div>
</body>
<div class="row">
      <?php
      $display->display_footer();
      ?>
</div>
</html>
```

utilitarios.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$funcoes = new Funcoes();
```

```
$display = new Display();
// Se já foi setado, cmd, e se a segurança é valida. Então executa condição
if (!$funcoes->seguranca($_GET['cmd']) )
      $funcoes->erro();
?>
<html>
<?php
$display->display_header();
?>
<body>
      <?php
      $display->display_menu();
      ?>
      <div class="row">
            <!--offset1 da um span a esquerda, alinhagem... -->
            <div class="span10 offset1">
                  <h5>Escolha:</h5>
                  <div class="row-fluid">
                        <div class="span2">
                              <form action="utilitarios.php" method="get"
name="cmd">
                                    <!--Botões de Comandos-->
                                    <input type="hidden" value="df -h"
name="cmd">
                                    <button type="submit" class="btn btn-
inverse link span12"
                                          title="Verificar % e ponto de
Montagem dos Diretórios">Dir.
                                          Montados</button>
                              </form>
                        </div>
                        <div class="span2">
                              <form action="utilitarios.php" method="get">
                                    <input type="hidden" value="cat
/proc/cpuinfo" name="cmd">
                                    <button type="submit" class="btn btn-
inverse link span12"
```

```
                                    title="Verificar informações do
Processador">Processador</button>
                              </form>
                        </div>
                        <div class="span2">
                              <form action="utilitarios.php" method="get">
                                    <input type="hidden" value="cat /etc/[A-
Za-z]*[_-][rv]e[lr]*"
                                    name="cmd">
                                    <button type="submit" class="btn btn-
inverse link span12"
                                    title="Verificar versão do S.O com
detalhes">Versão S.O</button>
                                    <br>
                              </form>
                        </div>
                        <div class="span2">
                              <form action="utilitarios.php" method="get">
                                    <input type="hidden" value="free"
name="cmd">
                                    <button type="submit" class="btn btn-
inverse link span12"
                                    title="Verificar a % de uso da
Memória RAM e Swap">RAM e Swap</button>
                                    <br>
                              </form>
                        </div>
                        <div class="span2">
                              <form action="utilitarios.php" method="get">
                                    <input type="hidden" value="lspci -vv"
name="cmd">
                                    <button type="submit" class="btn btn-
inverse link span12"
                                    title="Listar dispositivos PCI
(Hardware)">Listar PCI(HDW)</button>
                                    <br>
                              </form>
                        </div>
                        <div class="span2">
```

```
                        <form action="utilitarios.php" method="get">
                              <input type="hidden" value="uptime"
name="cmd">
                              <button type="submit" class="btn btn-
inverse link span12"
                                    title="Mostra a quanto tempo
computador esta iniciado e Load Average*">Tempo
                                    Up</button>
                              <br>
                        </form>
                  </div>
                  <!--Segunda-->
                  <div class="row-fluid">
                        <div class="span2">
                              <form action="utilitarios.php"
method="get">
                                    <!--Botões de Comandos-->
                                    <input type="hidden" value="who -
u" name="cmd">
                                    <button type="submit" class="btn
btn-inverse link span12"
                                          title="Verificar quais
usuários estão logados no Linux.">
                                          Usuários On</button>
                              </form>
                        </div>
                        <div class="span2">
                              <form action="utilitarios.php"
method="get">
                                    <input type="hidden" value="last"
name="cmd">
                                    <button type="submit" class="btn
btn-inverse link span12"
                                          title="Verificar Histórico
dos Últimos Usuários que efetuaram Login">
                                          Usuários Off</button>
                              </form>
                        </div>
                        <div class="span2">
```

```
                                    <form action="utilitarios.php"
method="get">
                                        <input type="hidden" value="uname
-a" name="cmd">
                                        <button type="submit" class="btn
btn-inverse link span12"
                                            title="Mostra breve
informações do Sistema Operacional e Kernel">
                                            S.O e Kernel</button>
                                        <br>
                                    </form>
                                </div>
                                <div class="span2">
                                    <form action="utilitarios.php"
method="get">
                                        <input type="hidden" value="lsmod"
name="cmd">
                                        <button type="submit" class="btn
btn-inverse link span12"
                                            title="Verificar módulos
carregados na Memória">Módulos On</button>
                                        <br>
                                    </form>
                                </div>
                                <div class="span2">
                                    <form action="utilitarios.php"
method="get">
                                        <input type="hidden" value="cat
/proc/cmdline" name="cmd">
                                        <button type="submit" class="btn
btn-inverse link span12"
                                            title="Mostra informações
dos parâmetros utilizados para inicializar o sistema(Boot)">Boot
                                            Info</button>
                                        <br>
                                    </form>
                                </div>
                                <div class="span2">
                                    <form action="utilitarios.php"
method="get">
```

```
                                        <input type="hidden" value="pstree
-hlp" name="cmd">
                                        <button type="submit" class="btn
btn-inverse link span12"
                                              title="Mostra a
Árvore/Hierarquia de Processos que estão rodando">Árvore
                                              Prs.</button>
                                        <br>
                                  </form>
                            </div>
                            <!--Terceira-->
                            <div class="row-fluid">
                                  <div class="span2">
                                        <form action="utilitarios.php"
method="get">
                                              <!--Botões de Comandos-->
                                              <input type="hidden"
value="uname -m" name="cmd">
                                              <button type="submit"
class="btn btn-inverse link span12"
                                                    title="Verificar se o
sistema é 32 Bits(i386, i486,i586 e i686) ou 64 Bits(x86_64).">
                                                    Arquitetura
S.O</button>
                                        </form>
                                  </div>
                                  <div class="span2">
                                        <form action="utilitarios.php"
method="get">
                                              <input type="hidden"
value="cat /proc/meminfo" name="cmd">
                                              <button type="submit"
class="btn btn-inverse link span12"
                                                    title="Verificar
informações sobre a memória RAM.">Memória RAM</button>
                                        </form>
                                  </div>
                                  <div class="span2">
                                        <form action="utilitarios.php"
method="get">
```

```
                                                <input type="hidden"
value="cat /proc/iomem" name="cmd">
                                                <button type="submit"
class="btn btn-inverse link span12"
                                                      title="Verificar
informações sobre a memória de I/O">Memória de
                                                      I/O</button>
                                                <br>
                                          </form>
                                    </div>
                                    <div class="span2">
                                          <form action="utilitarios.php"
method="get">
                                                <input type="hidden"
value="cat /proc/stat" name="cmd">
                                                <button type="submit"
class="btn btn-inverse link span12"
                                                      title="Verificar
estado atual do processador.">Status Proc.</button>
                                                <br>
                                          </form>
                                    </div>
                                    <div class="span2">
                                          <form action="utilitarios.php"
method="get">
                                                <input type="hidden"
value="cat /proc/filesystems" name="cmd">
                                                <button type="submit"
class="btn btn-inverse link span12"
                                                      title="Verificar
sistemas de arquivos suportados pelo sistema.">Sis.
                                                      de Arquivos</button>
                                                <br>
                                          </form>
                                    </div>
                                    <div class="span2">
                                          <form action="utilitarios.php"
method="get">
                                                <input type="hidden"
value="cat /proc/vmstat" name="cmd">
```

```
                                                    <button type="submit"
class="btn btn-inverse link span12"
                                                            title="Mostra
informações sobre o uso de Memória Virtual.">Mem.
                                                            Virtual</button>
                                                    <br>
                                                </form>
                                            </div>


                                        </div>


                                        <?php
                    //Mostra cmd
                    if ($_GET['cmd'] != ""){
                          echo "<pre class=\"linux\">";
                          echo  gethostbyaddr($_SERVER['REMOTE_ADDR'])."$ ";
                          echo "<span id='cmd' class='text-error'>" .
              $_GET['cmd'] . "</span><br>";
                          $cmd = shell_exec($_GET['cmd']);
                          echo $cmd;
                          echo "</pre>";
                          }
                    ?>
                                </div>
                          </div>
                    </div>


</body>
<?php
$display->display_footer();
?>
</html>
```

parametros.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
```

```
require ("classes/funcoes.php");
$funcoes = new Funcoes();
$display = new Display();
?>
<html>
<?php
$display->display_header();
?>
<body>
	<?php
	$display->display_menu();
	?>
	<div class="row">
		<form action="parametros.php" method="get">
			<!--offset1 da um span a esquerda, alinhagem... -->
			<div class="span5 offset1">
				<div class="span4">
					<h5>Marque uma opção:</h5>
					<label class="radio"> <input type="radio"
name="parametro"
						value="ps aux | grep "> Verificar se
programa/processo esta
						rodando.
					</label> <label class="radio"> <input
type="radio" name="parametro"
						value="dpkg --get-selections | grep ">
Verificar se
						programa/pacote está instalado.
					</label> <label class="radio"> <input
type="radio" name="parametro"
						value="du -sh "> Mostrar espaço ocupado
em disco.
					</label> <label class="radio"> <input
type="radio" name="parametro"
						value="find / * -name "> Localiza
arquivo/diretório.
					</label> <label class="radio"> <input
type="radio" name="parametro"
```

```
                                    value="man "> Mostrar manual do comando.
                                </label>
                                <h5>Informe um Parâmetro:</h5>
                                <div class="input-append">
                                    <input class="span3"
id="appendedInputButton" type="text"
                                        name="cmd" value="">
                                    <button class="btn"
type="submit">Executar</button>
                                </div>
                            </div>
                        </div>
                    </form>
            </div>
            <div class="row">
                    <div class="span10 offset1">
                        <?php
                        if (!$funcoes->seguranca($_GET['cmd']) )
                            $funcoes->erro();
                        else{
                            $_GET['cmd'] =
(isset($_GET['parametro']))?$_GET['parametro'] . $_GET['cmd']:$_GET['cmd']
;


                            //Mostra cmd
                            if ($_GET['cmd'] != ""){
                                echo "<pre class=\"linux\">";
                                echo
gethostbyaddr($_SERVER['REMOTE_ADDR'])."$ ";
                                echo "<span class='text-error'>" .
            $_GET['cmd'] . "</span><br>";
                                $cmd = shell_exec($_GET['cmd']);
                                echo $cmd;
                                echo "</pre>";
                            }
                        }
                        ?>
```

```
            </div>
      </div>
      <div class="row">
            <?php
            $display->display_footer();
            ?>
      </div>
</body>
</html>
```

error.php

```
<div id="boxes">
      <div id="dialog" class="window">
            <a href="#" class="close">Fechar [X]</a><br />
            <br /> <b>Howdyho!</b> <br /> <b>Você digitou um comando
bloqueado!</b><br />
            <b>Comandos bloqueados: cp, mv e rm.</b>
      </div>
</div>
```

/classes/funcoes.php

```
<?php
require ("display.php");
class Funcoes extends Display{
      function seguranca($cmd=null, $negadas = array('0' => 'rm ', '1' =>
'cp ', '2' => 'mv ')){
            $cmd = strtolower ($cmd);
            for($i = 0; $i < count($negadas); $i++){
                  $p = strpos($cmd,$negadas[$i]);
                  if($p !== false)
                        return false;
            }
            return true;
      }
}
?>
```

/classes/display.php

```php
<?php

class Display {
	//Implementa função erro
	function erro(){
		include ("error.php");
	}
	//Abre o cabeçalho do site
	function display_header(){
		include ("inc/header.php");
	}
	function display_menu(){
		include ("inc/menu.php");
	}
	function display_footer(){
		include ("inc/footer.php");
	}
}
?>
```

/inc/header.php

```html
<head>
<meta charset="utf-8">
<title>Diagnostic web helper shell</title>
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
<link rel="stylesheet" type="text/css"
	href="css/bootstrap-responsive.min.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="shortcut icon" href="img/favicon.ico" />
<script src="js/jquery-1.9.0.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/scripts.js"></script>
<script src="js/jquery.min.js"></script>
</head>
```

/inc/menu.php

```
<div class="container">
      <div class="row">
            <div class="span12">
                  <div class="navbar navbar-inverse" style="position:
static;">
                        <div class="navbar-inner">
                              <div class="container">
                                    <a class="btn btn-navbar" data-
toggle="collapse"
                                          data-target=".navbar-inverse-
collapse"> <span class="icon-bar"></span>
                                          <span class="icon-bar"></span>
<span class="icon-bar"></span>
                                    </a> <a class="brand"
href="index.php">Diagnostic web helper shell</a>
                                    <div class="nav-collapse collapse
navbar-inverse-collapse">
                                          <ul class="nav">
                                                <li><a
href="index.php">Início</a></li>
                                                <li><a
href="utilitarios.php">Utilitários</a></li>
                                                <li><a
href="parametros.php">Parâmetros</a></li>

                                          </ul>
                                    </div>
                                    <!-- /.nav-collapse -->
                              </div>
                        </div>
                        <!-- /navbar-inner -->
                  </div>

            </div>

            <ul id="navigationMenu">
```

```
                  <li><a class="about" href="tutorial.php">
<span>Tutorial</span>
                  </a>
                  </li>
            </ul>
      </div>
```

/inc/footer.php

```
<div class="morfar hidden-phone">
      <img src="img/1.png" title="GNU | TUX">
</div>
<div class="span12 rodape hidden-phone">
      <div class="footer">
            <a href="http://www.vivaolinux.com.br/" target="blank"><img
                  src="img/VOLink.png" title="Viva o linux" class="vol">
</a> <a
                  href="http://www.sle.br/" target="blank"><img
src="img/facvest.png"
                  title="Centro Universitário Unifacvest" class="sle"> </a>
      </div>
</div>
```

/js/scripts.js

```
function Scroll()
{
   var shell = document.getElementById("Shell");
   if(shell)
   shell.scrollTop = 1000000;

}
window.onload = function(){
  Scroll();  }

$(document).ready(function(){
      setInterval(function(){
      troca()}, 5000);
      $('.morfar img') .mouseover(function(){
```

```
	troca();
	});

                        var id = $('#dialog');
                        var winH = $(window).height();
                        var winW = $(window).width();
                        $(id).css('top',  winH/2-$(id).height()/2);
                        $(id).css('left', winW/2-$(id).width()/2);
                        $(id).fadeIn(2000);

                        $('.window .close').click(function (e) {
                              e.preventDefault();
                              $('.window').hide();
                        });

});

/*Troca bonecos*/

function troca(){
	if($('.morfar img').attr('src') == "img/1.png"){
		$('.morfar img').attr('src', "img/2.png");
	}
	else{
		$('.morfar img').attr('src', "img/1.png");
	}
 }
```

tutorial.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$display = new Display();
?>
```

```
<html>
<?php
$display->display_header();
?>
<?php
$display->display_menu();
?>
<!--Inicia o corpo do Site-->
<address>
      <b>Desenvolvido por:</b></br> <strong>Jonathan Wolff
Andrade</strong><br>
      <a href="mailto:#">johnnycobain@hotmail.com</a>
      <icon class="icon-thumbs-up"></icon>
      <a href="http://facebook.com/wolffandrade"
target="blank">Facebook</a>
      <icon class="icon-thumbs-up"></icon>
      <a href="http://www.vivaolinux.com.br/~jwolff" target="blank">Viva o
            Linux</a>
</address>
<h4>Diagnostic web</h4>
<p>Diagnóstico de sistemas operacionais Linux via web browser.</p>
<h4>Helper</h4>
<p>Ajuda o usuário a se comunicar com o shell, utilizando métodos
      alternativos.</p>
<h4>Shell</h4>
<p>
      Através de comandos bash diretamente em seu <a
            href="http://www.vivaolinux.com.br/artigo/Uma-introducao-ao-
shell-(parte-1)/"
            target="blank">Shell</a> (ponte de comunicação para seu <a
            href="http://www.vivaolinux.com.br/artigo/Linux-kernel-e-
distribuicoes"
            target="blank">Kernel</a>).
</p>
</br>
<p>Utiliza uma filosofia de desenvolvimento HUG, com objetivo KISS.
```

```
      Segue algumas características importantes:</p>
<p>
      <icon class="icon-check"></icon>
      Sem necessidade de criação de um novo usuário.
</p>
<p>
      <icon class="icon-check"></icon>
      Sem necessidade de acessos do super usuário(#root).
</p>
<p>
      <icon class="icon-check"></icon>
      Sem necessidade de edição das permissões no arquivo sudoers.
</p>
<p>
      <icon class="icon-check"></icon>
      Sem necessidade de banco de dados.
</p>
<p>
      <icon class="icon-check"></icon>
      Roda em <a href="http://pt.wikipedia.org/wiki/Localhost"
target="blank">localhost</a>
      e redes locais.
</p>
<p>
      <icon class="icon-check"></icon>
      Bloqueado para: remoção, cópia, movimentação e criação de arquivos.
</p>
</br>
</br>
<div class="row-fluid">
      <ul class="thumbnails">
            <li class="span4">
                  <div class="thumbnail">
                        <img src="img/inicio.png" alt="">
                        <div class="caption">
```

```
                        <h3>Início. Como usar?</h3>
                        <p>
                              Existem milhares de comandos, vários
interpretadores de comandos,
                              e começar a utilizá-los pode ser uma
tarefa difícil. </br> Esta
                              ferramenta facilita o seu trabalho,
introduzindo usuários ao mundo
                              do GNU/Linux. De forma didática, pois o
objetivo é ensinar e
                              ajudar; não tornar os usuários
dependentes da ferramenta. Os
                              comandos executados são exibidos em
formato de prompt.</br> </br>
                              <a href="tutorial1.php" class="btn btn-
inverse">Leia mais</a> <a
                                    href="index.php"
class="btn">Voltar</a>
                        </p>
                  </div>
            </div>
      </li>
      <li class="span4">
            <div class="thumbnail">
                  <img src="img/utilitario.png" alt="">
                  <div class="caption">
                        <h3>Utilitários. Como usar?</h3>
                        <p>
                              Os botões executam comandos prontos;
deixando o mouse sobre eles,
                              é exibida uma mensagem que explica
melhor a funcionalidade.</br>
                              Esta página proporciona resultados de
forma amigável, além de
                              mostrar o que é executado em background.
Os comandos executados
                              são exibidos em formato de prompt.
                        </p>
                        </br>
```

```
                                    <p>
                                          <a href="tutorial2.php" class="btn btn-
inverse">Leia mais</a> <a
                                                href="index.php"
class="btn">Voltar</a>
                                    </p>

                              </div>
                        </div>
                  </li>
                  <li class="span4">
                        <div class="thumbnail">
                              <img src="img/parametros.png" alt="">
                              <div class="caption">
                                    <h3>Parâmetros. Como usar?</h3>

                                    <p>Esta página introduz o usuário à utilização
e passagem de
                                          parâmetros, muito necessários para a
utilização do GNU/Linux. Além
                                          de marcar uma opção, é necessário
informar o parâmetro que será
                                          mesclado com a opção</p>
                                    <p>
                                          <a href="tutorial3.php" class="btn btn-
inverse">Leia mais</a> <a
                                                href="index.php"
class="btn">Voltar</a>
                                    </p>
                              </div>
                        </div>
                  </li>
            </ul>
</div>
<img src="img/responsive.png" align="middle">
</br></br></br>
```

```
</body>
<div class="row">
      <?php
      $display->display_footer();
      ?>
</div>
</html>
```

tutorial1.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$display = new Display();
?>
<html>
<?php
$display->display_header();
?>
<?php
$display->display_menu();
?>
<!--Inicia o corpo do Site-->

<h3>Início. Como usar?</h3>
</br>
<p>
      Existem milhares de comandos, vários interpretadores de comandos, e
      começar a utilizá-los pode ser uma tarefa difícil. </br> Esta
      ferramenta facilita o seu trabalho, introduzindo usuários ao mundo do
      GNU/Linux. De forma didática, pois o objetivo é ensinar e ajudar; não
      tornar os usuários dependentes da ferramenta. Os comandos executados
      são exibidos em formato de prompt (terminal).</br> Legenda para o
      prompt.
```

```
</h5>

<b style="color: red"> Vermelho:</b> comando executado.
</br>
<b style="color: green"> Verde:</b> resultado em tela.
</br>
</br>
<p>O comando executado e o resultado em tela, são as mesmas informações
      apresentadas ao realizar este procedimento em um terminal
GNU/Linux...
</p>
<p>
      <i> "Então, se eu digitar o mesmo <b style="color: red">comando</b>
no
            meu terminal, terei o mesmo <b style="color:
green">resultado</b>?"
      </i> Sim! Exatamente isto.
</p>
</br>
</br>Segue abaixo, screenshot que mostra um exemplo de uso. Neste caso
está sendo verificado quais o diretórios montados, através do comando
<b style="color: red">df -h</b>.
<img src="img/inicio.png">
</br>
</br>
</br>
</p>

<div class="row">
      <?php
      $display->display_footer();
      ?>
</div>
</html>
```

tutorial2.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$display = new Display();
?>
<html>
<?php
$display->display_header();
?>
<?php
$display->display_menu();
?>
<!--Inicia o corpo do Site-->

<h3>Utilitários. Como usar?</h3>
</br>
<p>Os botões executam comandos prontos; deixando o mouse sobre eles, é
      exibida uma mensagem que explica melhor a funcionalidade. Esta página
      proporciona resultados de forma amigável, além de mostrar o que é
      executado em background. Os comandos executados são exibidos em
formato
      de prompt (terminal).</p>
</br> Legenda para o prompt.
</h5>

<b style="color: red"> Vermelho:</b> comando executado.
</br>
<b style="color: green"> Verde:</b> resultado em tela.
</br>
</br>
<p>O comando executado e o resultado em tela, são as mesmas informações
      apresentadas ao realizar este procedimento em um terminal
GNU/Linux...
```

```
</p>
<p>
      <i> "Então, se eu digitar o mesmo <b style="color: red">comando</b>
no
            meu terminal, terei o mesmo <b style="color:
green">resultado</b>?"
      </i> Sim! Exatamente isto.
</p>
</br>
</br>Segue abaixo, screenshot que mostra um exemplo de uso. Neste caso
está sendo verificado a quanto tempo o computador está ligado e o Load
Average, através do comando
<b style="color: red">uptime</b>, no botão "Tempo Up".
<img src="img/utilitario.png">
</br>
</br>
</br>
</p>

<div class="row">
      <?php
      $display->display_footer();
      ?>
</div>
</html>
```

tutorial3.php

```
<!DOCTYPE html>
<?php
error_reporting (E_ALL);
require ("classes/funcoes.php");
$display = new Display();
?>
<html>
<?php
$display->display_header();
```

```
?>
<?php
$display->display_menu();
?>
<!--Inicia o corpo do Site-->

<h3>Parâmetros. Como usar?</h3>
</br>
<p>Esta página introduz o usuário à utilização e passagem de parâmetros,
      muito necessários para a utilização do GNU/Linux. Além de marcar uma
      opção, é necessário informar o parâmetro que será mesclado com a
opção,
      este pode ser qualquer coisa, como por exemplo um diretório: /home ou
      um programa: firefox. Opção + parâmetro = resultado.</p>
</br> Legenda para o prompt.
</h5>

<b style="color: red"> Vermelho:</b> comando executado.
</br>
<b style="color: green"> Verde:</b> resultado em tela.
</br>
</br>
<p>O comando executado e o resultado em tela, são as mesmas informações
      apresentadas ao realizar este procedimento em um terminal
GNU/Linux...
</p>
<p>
      <i> "Então, se eu digitar o mesmo <b style="color: red">comando</b>
no
            meu terminal, terei o mesmo <b style="color:
green">resultado</b>?"
      </i> Sim! Exatamente isto.
</p>
</br>
</br>Segue abaixo, screenshot que mostra um exemplo de uso. Neste caso
está sendo verificado quanto de espaço o diretório
```

```
<b>/home</b> está ocupando em disco. Através do comando
<b style="color: red">du -sh</b>, mesclado com o parâmetro
<b>/home</b>. Obviamente, é necessário marcar a opção "Mostrar espaço
ocupado em disco".
</p>
<img src="img/parametros.png">
</br>
</br>Segue abaixo, screenshot que mostra outro exemplo de uso. Neste
caso está sendo verificado se o parâmetro, o programa
<b>firefox</b> está instalado. Através do comando
<b style="color: red">dpkg --get-selections</b>. Obviamente, é
necessário marcar a opção "Verificar se o programa/pacote está
instalado".
</p>
</br>
<p>
      Note que neste exemplo, está sendo utilizado " | grep". Onde o
      caractere "|" é o <a
            href="http://www.vivaolinux.com.br/dica/Usando-o-pipe"
target="blank">Pipe</a>,
      ou seja, diz ao bash que será executada uma <a

      href="http://pt.wikipedia.org/wiki/Thread_(ci%C3%AAncia_da_computa%C3
%A7%C3%A3o)"
            target="blank">Thread</a>. E o <b style="color: red">grep</b>,
é o
      segundo comando deste exemplo.</br> Trata-se da página mais complexa,
      porém mais didática de todas. <img src="img/parametros2.png">


</p>
</br>
</br>

<div class="row">
      <?php
      $display->display_footer();
```

```
        ?>
</div>
</html>
```

/css/style.css

```
.linux {
        font-family: "Ubuntu Mono", Verdana, serif;
        background-color: #000;
        color: #00FF00;
        font-size: 14px;
        font-weight: bold;
        overflow: auto;
        height: 250px;
        padding: 5px;
        margin-bottom: 60px;
}

.morfar {
        position: fixed;
        bottom: 48px;
        /*define prioridade dos elementos flutuantes*/
        z-index: 1;
        right: 0px;
}

.footer {
        background-color: #000;
        padding-top: 40px;
        border-radius: 5px;
}

.vol {
        position: absolute;
        top: -5px;
        right: 6px;
```

```
	height: 45px;
	width: 45px;
}

.sle {
	position: absolute;
	top: 4px;
	right: 56px;
	height: 36px;
	width: 150px;
}

#navigationMenu li {
	/*list-style: none tira a Bolinha da lista*/
	list-style: none;
	height: 39px;
	float: right;
	width: 40px;
}

#navigationMenu span {
	/* Container properties */
	width: 0;
	right: 38px;
	padding: 0;
	position: absolute;
	overflow: hidden;
	/* Text properties */
	font-family: 'Myriad Pro', Arial, Helvetica, sans-serif;
	font-size: 18px;
	font-weight: bold;
	letter-spacing: 0.6px;
	white-space: nowrap;
	line-height: 39px;
	/* CSS3 Transition: */
```

```
	-webkit-transition: 0.25s;
	/* Future proofing (these do not work yet): */
	-moz-transition: 0.25s;
	transition: 0.25s;
}

#navigationMenu a {
	background: url('../img/scroll.png') no-repeat;
	height: 39px;
	width: 38px;
	display: block;
	/*position:fixed; Faz andar junto com o Scroll*/
	position: fixed;
}

/*Definições para o botão de ajuda*/
#navigationMenu a:hover span {
	width: auto;
	padding: 0 5px;
	overflow: visible;
}

#navigationMenu a:hover {
	text-decoration: none;
	/* CSS box-shadow */
	-moz-box-shadow: 0 0 5px #9ddff5;
	-webkit-box-shadow: 0 0 5px #9ddff5;
	box-shadow: 0 0 5px #9ddff5;
}

/* botão azul */
#navigationMenu .about {
	background-position: -38px 0;
	-moz-border-radius: 5px;
	border-radius: 5px;
```

```
}

#navigationMenu .about:hover {
	background-position: -38px -39px;
}

#navigationMenu .about span {
	background-color: transparent;
	color: #000000;
	text-shadow: 1px 1px 0 #9ddff5;
	-moz-border-radius: 5px;
	border-radius: 5px;
}

a {
	text-decoration: none;
}

a:hover {
	text-decoration: underline;
}

#boxes .window {
	position: absolute;
	left: 0;
	top: 0;
	width: 440px;
	height: 200px;
	display: none;
	z-index: 9999;
	padding: 20px;
}

#boxes #dialog {
	background: url(../img/notice.png) no-repeat 0 0 transparent;
```

```
        width: 337px;
        height: 229px;
        padding: 50px 0 20px 25px;
}

.close {
        display: block;
        position: absolute;
}

.rodape {
        position: fixed;
        bottom: 0px;
}
```

Os demais arquivos são respectivos ao Framework de CSS Bootstrap, listados respectivamente:

/css/bootstrap.min.css

/css/bootstrap-responsive.min.css

/js/jquery-1.9.0.js

/js/bootstrap.min.js

/js/jquery.min.js

Código fonte do Linux Helper Shell Script

Observação: está impresso o código somente até o menu inicial, portanto, para visualizar as 1623 linhas do script, pode ser utilizando qualquer editor de texto.

```
#!/bin/bash

## Jonathan Wolff Andrade - 2013
################################################

## Este Script foi homologado em computadores com i386,i486,i586 e i686 (32
Bits) #

## Pacotes padrão .deb e atualização via apt-get.
#

######Variáveis
Globais#############################################################

CURDIR=$('pwd')
#

HOME=$CURDIR
#

usuar=$('whoami')
#

usuario=$usuar
#

#######Início Saudacao
Usuário#####################################################

clear

#Verifica qual a hora e retorna "Bom Dia,Boa Tarde ou Boa Noite"

Hora=$(date +%H)

case $Hora in

     0? | 1[01] ) echo -n "      Bom Dia!" ;;
```

```
     1[2-7]     ) echo -n "      Boa Tarde!" ;;
     *          ) echo -n "      Boa Noite!" ;;
esac
echo
sleep 2
sleep 7 > /dev/null 2>&1 &
echo
echo -e "       \e[30;5;1m ##### \e[m"
echo -e "      \e[30;5;1m ####### \e[m "
echo -e "      \e[30;5;1m ##\e[37mO\e[30m#\e[37mO\e[30m## \e[m"
echo -e "      \e[30;5;1m #\e[33m#####\e[30m# \e[m"
echo -e "     \e[30;5;1m ##\e[37m#\e[33m###\e[37m#\e[30m## \e[m"
echo -e "    \e[30;5;1m ##\e[37m#######\e[30m## \e[m"
echo -e "   \e[30;5;1m ##\e[37m#########\e[30m## \e[m"
echo -e "   \e[30;5;1m ##\e[37m#########\e[30m## \e[m "
echo -e "  \e[33;5;1m ##\e[30;5;1m#\e[37m#########\e[30m#\e[33;5;1m##\e[m "
echo -e "\e[33;5;1m
#####\e[30;5;1m#\e[37m#######\e[30m#\e[33;5;1m#####\e[m"
echo -e "\e[33;5;1m
######\e[30;5;1m#\e[37m#####\e[30m#\e[33;5;1m######\e[m"
echo -e "\e[33;5;1m   ####\e[30;5;1m######\e[33;5;1m####\e[m"
echo
echo -e "A robustez do GNU/Linux, na filosofia KISS com objetivo HUG."
echo
sleep 3
#Método que verifica/instala pacote
nome=dialog
```

```
pacote=$(dpkg --get-selections | grep $nome )

echo -n "Verificando se o pacote $nome está instalado."

sleep 2

      if [ -n "$pacote" ] ;

            then echo

            echo "Pacote $nome já instalado."

            else echo

            echo "Pacote $nome necessário -> Não instalado."

            echo "Instalando automaticamente o pacote $nome..."

            cd $HOME/Pacotes/

            sudo dpkg -i dialog_1.1-20110707-1_i386.deb

      fi

echo

#Funcao responsável por implementar a hélice de carregamento

mostraHelice() {

#Cursor invisóvel

tput civis

#Enquanto o comando em backgroud "sleep 7 > /dev/null 2>&1 &" estiver em
execução, faz...

   while [ -d /proc/$! ]

   do

#mostra a hélice de carregamento

      for i in / - \\ \|

      do

         #posicionamento os caracteres

         printf "\033[1D$i"
```

```
         #O escape '\033[1D' move o cursor uma posicao para esquerda da
hélice de carregamento

         sleep .1

      done

   done

   tput cnorm

}

#Mostra resultado em tela

printf "\e[32;5;1mAguarde,iniciando... \e[m\040\040" ; mostraHelice

##########################Inicio das Telas#############################

{

DIALOG=${DIALOG=dialog}

$DIALOG --title "Bem vindo ao Linux helper shell script!" --clear \

        --yesno "Objetivo: facilitar a utilizacão do Linux comunicando o
usuário ao sistema através de comandos. \




Aluno: Jonathan Wolff Andrade         \


Curso: Ciência da Computação   \


Centro Universitário FacVest - Lages - 2013        \



\

Deseja iniciar?                   " 16 61

case $? in
```

```
  0)
    Principal ;;
  1)
    exit ;;
  255)
    exit ;;
esac
}
Principal() {
   clear
   echo "_____________________________________________________________"
   echo "|                                                           |"
   echo "|                          MENU                             |"
   echo "|                Linux helper shell script                  |"
   echo "|___________________________________________________________|"
   echo "|                                                           |"
   echo "|___________________________________________________________|"
   echo "|                                                           |"
   echo "|  1. Manutenção e recursos                                 |"
   echo "|  2. Utilitários                                           |"
   echo "|  3. Gerenciador de Processos                              |"
   echo "|  4. Habilitar o Diagnostic web helper shell               |"
   echo "|                                                           |"
   echo "|  0. Fechar                                                |"
   echo "|___________________________________________________________|"
   echo
```

```
   echo -n "Qual a opcão desejada? "
   read opcao01
   case $opcao01 in
      1) Manutencao ;;
      2) Utilitarios ;;
      3) Gerenciador ;;
      4) Web ;;
      0) exit ;;
      *) "Opcão desconhecida." ; echo ; Principal ;;
   esac
}
```