

RISO <<Recuperação por Imagem de Sistema Operacional>>

Versão 0.8

Manual do Usuário

Escrito por

Germano Teixeira de Miranda
Gilmar Pereira de Alcântara
Lucas Henrique Ferreira de Melo

Divisão de Gerencia e Operações - DGO
Departamento de Recursos em Informática – DRI
Departamento de Computação - DECOM
Centro Federal de Educação tecnológica de Minas Gerais – CEFET-MG
Fevereiro / 2015

Índice

Prefácio	3
Licença	3
Informações de contato do suporte técnico	3
Introdução	4
Apresentação	4
Requisitos do sistema	5
Usando o RISO.....	5
Instalando o sistema RISO	5
Criando clones.....	7
RISOS	9
Apresentação	9
Executando	9
Iniciar Servidor	10
Criar imagens.....	11
Criar torrents.....	11
Criar live-cd	12
Atualizar.....	12
Créditos	12
Help	13
RISO	13
Apresentação	13
Executando	14
Instalar imagens já existentes	15
Baixar e instalar novas imagens	15
Baixar e instalar imagem de recuperação (LIVE-CD)	18
Atualizar	18
Configurações	18
Créditos	19
Help	19
CÓDIGO COMENTADO	20
Instalação.....	20
Risos.....	26
Riso	46
GITHUB	60
O que é o Git.....	60
O que é o GitHub.....	60
Criando um repositório remotamente.....	60
Trabalhando com Git via terminal.....	60
Instalando o Git.....	60
Clonando repositório.....	61
Informações do repositório.....	61
Atualizando repositório.....	61
Status do repositório.....	61
Adicionando mudanças no repositório.....	61
Commitando arquivo.....	61
Informações de log (commit).....	62
Identificando mudanças no arquivo.....	62
Removendo arquivos.....	62
Revertendo modificações.....	62
Alterações locais para o repositório remoto.....	63
REFÊRENCIAS.....	64

Prefácio

Licença

"THE BEER-WARE LICENSE" (Revision 42) :

<riso@comp.eng.br> wrote this file. As long as you retain this notice you can do whatever you want with this stuff. If we meet some day, and you think this stuff is worth it, you can buy me a beer in return.

Informações de contato do suporte técnico

Telefone: (31) 3319-6755

Telefone: (31) 3319-6704

E-mail: riso@comp.eng.br

Introdução

Motivação

Quando somos responsáveis pela manutenção de um grande numero de computadores, garantir o funcionamento do sistema operacional com todos os softwares necessários é uma tarefa árdua que se torna cada vez mais penosa à medida que o numero de computadores aumenta. Frequentemente nesses ambientes esses computadores possuem hardware semelhante o que facilita muito o nosso trabalho se usarmos softwares como g4u ou partimage que permitem a criação de uma imagem que é distribuída pela rede. O RISO foi desenvolvido com a mesma finalidade, criar uma imagem e distribuí-la pela rede, o seu diferencial é a simplicidade e o protocolo torrent usado para distribuir essa imagem pela rede tornando o processo muito mais rápido. O desenvolvimento do riso foi iniciado por um grupo de estagiários e funcionários do CEFET-MG e como somos fervorosos defensores de uma cultura livre ele é um software livre implementado em scriptshell.

Apresentação

O RISO é um sistema de recuperação de computadores que surgiu com a finalidade de facilitar a administração de laboratórios de computação. Como nesses laboratórios existe um grande numero de computadores por funcionários é muito difícil manter a configuração de todos sincronizada. Com o RISO essa tarefa se torna trivial. O RISO vai permitir a configuração de um computador que servira de modelo para os outros do laboratório, com ele, criaremos uma cópia perfeita de todas as configurações e programas do computador de referência.

O projeto encontra-se atualmente no GitHub, por esse link <https://github.com/decom/RISO> você pode visualiza-lo e baixa-lo no botão Download ZIP. Para baixar SOMENTE O MANUAL, clique em Manual-Riso.pdf e logo depois em View Raw.

Para realizar essa tarefa o RISO cria imagens dos sistemas operacionais existentes no computador de referência e através da rede por protocolo torrent envia essas imagens para os outros computadores do laboratório.

O RISO se divide em duas partes, um servidor que fica instalado no computador de referência (risos) e um cliente que fica instalado nos demais computadores do laboratório (riso). Em ambos os casos o RISO fica instalado em uma partição própria no HD sobre o Ubuntu Server, que deve ter espaço suficiente para armazenar as imagens dos outros sistemas operacionais existentes no computador.

Após a instalação do sistema RISO o HD terá o seguinte formato:



- 1 Uma imagem é uma copia exata de um sistema operacional juntamente com seus programas e configurações, o seu tamanho é igual ao tamanho ocupado pelo sistema operacional e seus programas no HD.

Essa configuração poderá ser alterada conforme necessidade do usuário. Já que esse pode prever o tamanho das partições conforme uso do sistema e também a quantidade de sistemas que serão utilizados. Mas a partição de recuperação é uma exceção, pois tem que comportar metade do HD, caso todas as partições estiverem totalmente ocupadas, teremos imagens que com seus tamanhos somarão o tamanho aproximado do espaço que elas ocupam no disco. Esse tem que ser menor que a partição de recuperação para um funcionamento ideal do sistema.

Uma partição para Windows, uma partição para Linux Desktop, uma partição para a área de troca (swap) e uma partição para o sistema de recuperação. Essa partição de recuperação contém o Ubuntu Server com sistema RISO instalado, RISO Server (risos) no computador de referência e RISO Cliente (riso) nas demais máquinas.

Atualmente o RISO trabalha apenas em computadores com dois sistemas operacionais um Windows e um Linux, ele pode ser modificado para trabalhar com quantos sistemas forem desejados. Existem duas versões uma com um Windows e um Linux e outra com 2 Windows e 1 Linux, isso além da partição de recuperação que será tratada posteriormente.

Os comandos para criar, instalar, baixar e disponibilizar as imagens no servidor, bem como a função de criar torrents, são executadas por comandos simples que usam aplicativos baixados no processo de instalação. O código foi projetado para agilizar o processo e preparar o sistema, evitando o trabalho desgastante de fazer o processo à mão.

Requisitos do sistema

Os requisitos mínimos para a utilização do sistema são:

- Processador Intel® Pentium® (mínimo de 66 MHz 486DX2) , similar ou superior.
- Sistema operacional: Ubuntu Server 9.10 ou superior.
- 16MB de RAM ou superior.
- Placa de rede 100 Mbps ou superior.

Usando o RISO

Instalando o sistema RISO

Escolha um computador para ser o nosso computador de referência e mãos à obra.

A primeira coisa a fazer é fazer o particionamento do HD:

- > Uma partição para o Windows.
- > Uma para o Linux.
- > Uma para a Área de Troca (swap).
- > Uma para a partição de recuperação.

Com o HD formatado:

> Instale o Windows de sua preferência na partição do Windows, já testei com Windows XP e com o Windows 7.

> Instale um Linux Desktop na partição do Linux, eu uso o Ubuntu mas vai funcionar com qualquer distribuição.

> Na partição de recuperação instale o Ubuntu Server 9.10 ou superior. Não se preocupe em arrumar o menu do grub2 agora pois a instalação do riso irá editá-lo.

Para instalar o sistema baixe o arquivo master.zip Ele pode ser baixado pelo terminal com o comando:

```
> wget -c https://github.com/decom/RISO/archive/master.zip
```

Na partição de recuperação onde você instalou o sistema Ubuntu Server descompacte o arquivo com o comando:

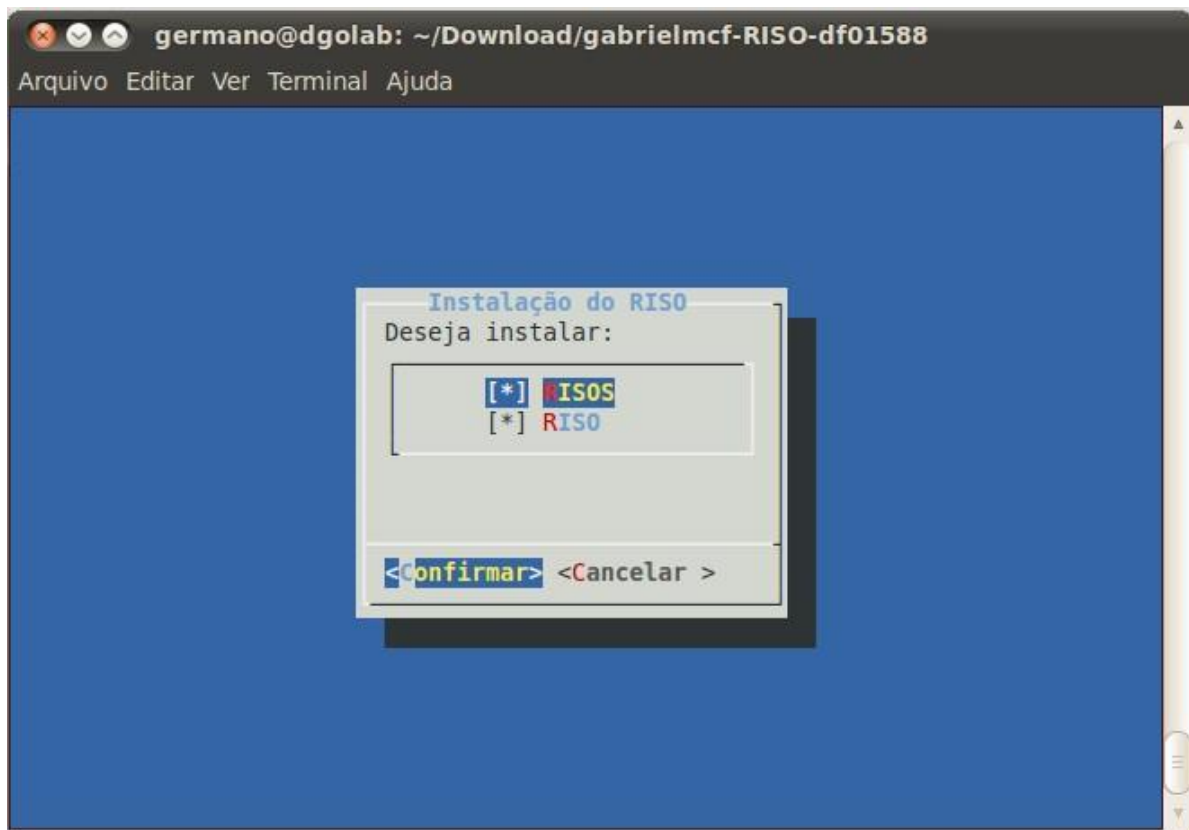
```
> unzip master.zip -d ./
```

Entre na pasta descompactada e instale o sistema.

```
> cd RISO-master
```

> ./install (certifique-se de estar conectado à internet, se o computador não estiver pegando ip automaticamente durante o boot coloque um dhclient no rc.local)

A seguinte tela será mostrada:



Você pode marcar e desmarcar opções com a barra de espaço. Deixe as duas selecionadas e aperte <Confirmar>.

Agora espere até que todas as dependências sejam baixadas e o sistema seja instalado.

... isso pode levar alguns minutos...

Bom, se tudo deu certo você pode abrir o RISO Cliente com o comando:

```
> riso
```

e o RISO Server com o comando:

> risos

Em um sistema limpo é importante verificar se todas as dependências foram instaladas, evitando assim contratempos no uso.

Quando é executado o script de instalação do riso, é criada uma árvore de diretórios (/urs/riso/imagens) e links para o aceso do sistema. Já a instalação do risos, instala uma maior quantidade de dependências, e executa scripts baseados no Remastersys (aplicativo criado para gerar iso do seu sistema atual), juntamente com as especificações de partições, como sistemas de arquivo, partição de cada S.O e especifica um padrão para o tamanho das imagens [0-9] , já que elas não foram criadas ainda.

As especificações das partições são inseridas num arquivo chamado riso.service, a existência desse arquivo que ira definir se o seu computador é o servidor ou não. Nesse arquivo também é escrito o nome do serviço, que é muito importante, pois podem existir vários servidores risos na mesma rede, desde que eles tenham nomes diferentes de serviço. Já a saída que carrega as variáveis é especificada pelo nome do riso. No caso do Decom o nome é _DECOM_RISO._tcp.

Por ultimo o riso verifica se o sistema é Windows 7 pois já que o arquivo de iniciação é binário, o mesmo é simplesmente trocado por um que vem no pacote do riso, (arquivo BCD).

Criando clones

Para começar a criar clones da maquina que acabamos de fazer, precisamos criar um LIVE-CD do RISO. Para fazer isso execute a função 4. Criar LIVE-CD do RISOS e siga os passos at´q que tenha um LIVE-CD do RISO em mãos.

Depois de criar o LIVE-CD você precisa criar as imagens dos sistemas operacionais existentes na maquina de referência. Para isso use a função criar imagens no risos e selecione todas as imagens.

Aqui começamos a usar o maquina que será o nosso clone.

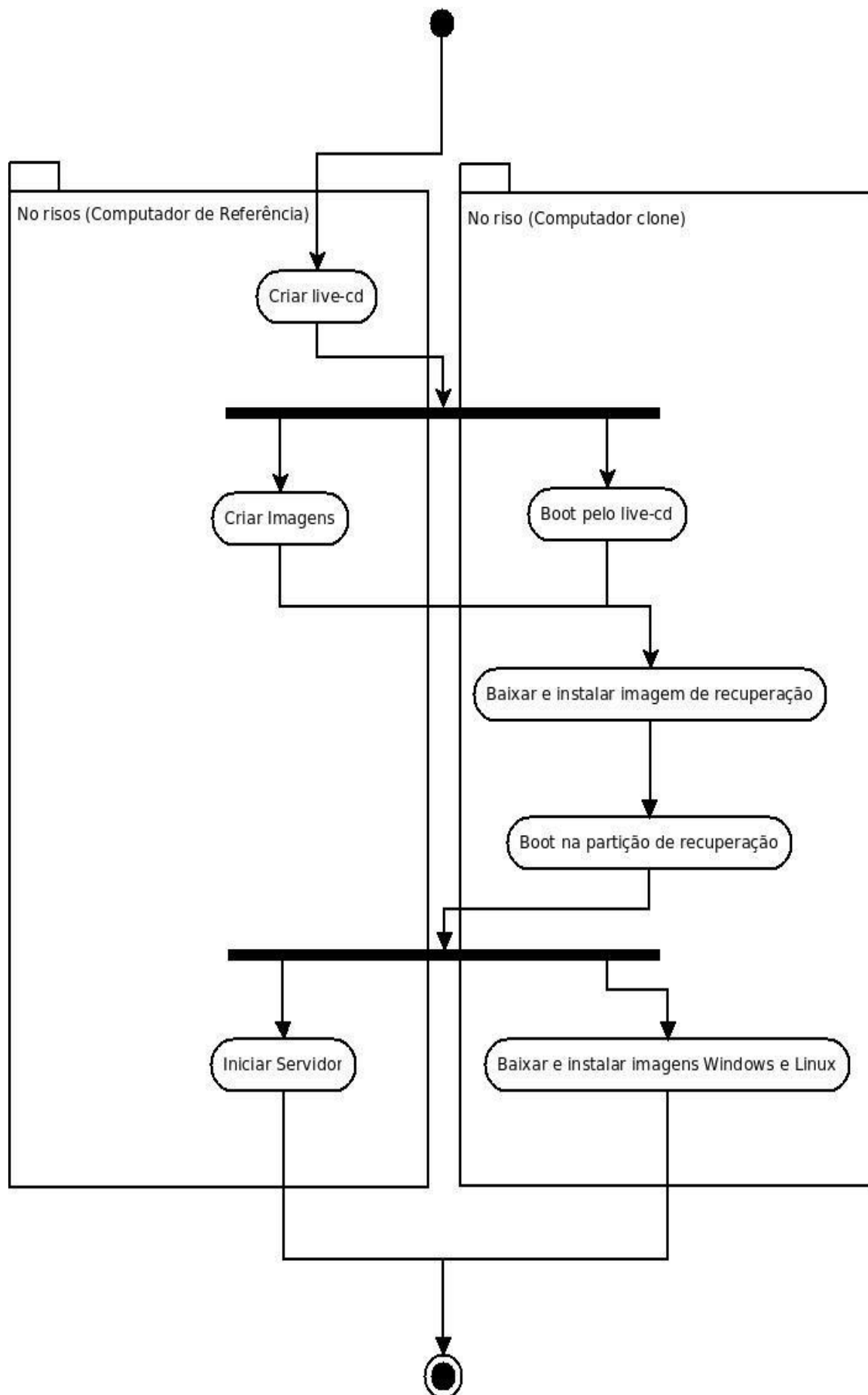
De boot pelo LIVE-CD e execute a função 3. baixar imagem de recuperação. Quando o processo terminar, remova o CD e de boot pelo HD.

Antes de baixar e instalar a imagem do Windows e do Linux precisamos iniciar o servidor com a função 1. Iniciar servidor do risos no computador de referência.

Volte ao computador que será o clone e execute a função 2. Baixar e instalar imagens.

Quando o processo terminar você terá uma cópia perfeita da maquina de referência. Lembre-se de que a instalação da imagem de recuperação, parte que envolve o LIVE-CD, só é necessária na primeira vez em que a maquina é clonada. Nas posteriores clonagens será necessário baixar a imagens do Linux e do Windows.

O diagrama abaixo irá lhe ajudar a ter uma visão geral de como realizar o processo:



Se ao final do processo de instalação da imagem do Windows e do Linux alguns dos sistemas operacionais não estiverem listados no menu de grub, entre novamente na partição de recuperação e execute o comando:

> update-grub

isso fará com que todos os sistemas operacionais sejam identificados corretamente.

Uma segunda alternativa é o uso do Clonezilla. Esse programa permite a clonagem completa do HD. Após concluir a instalação em uma máquina esse pode ser replicado para as outras. Tomando o cuidado para que os clientes (computadores clonados), não tenham o servidor risos.

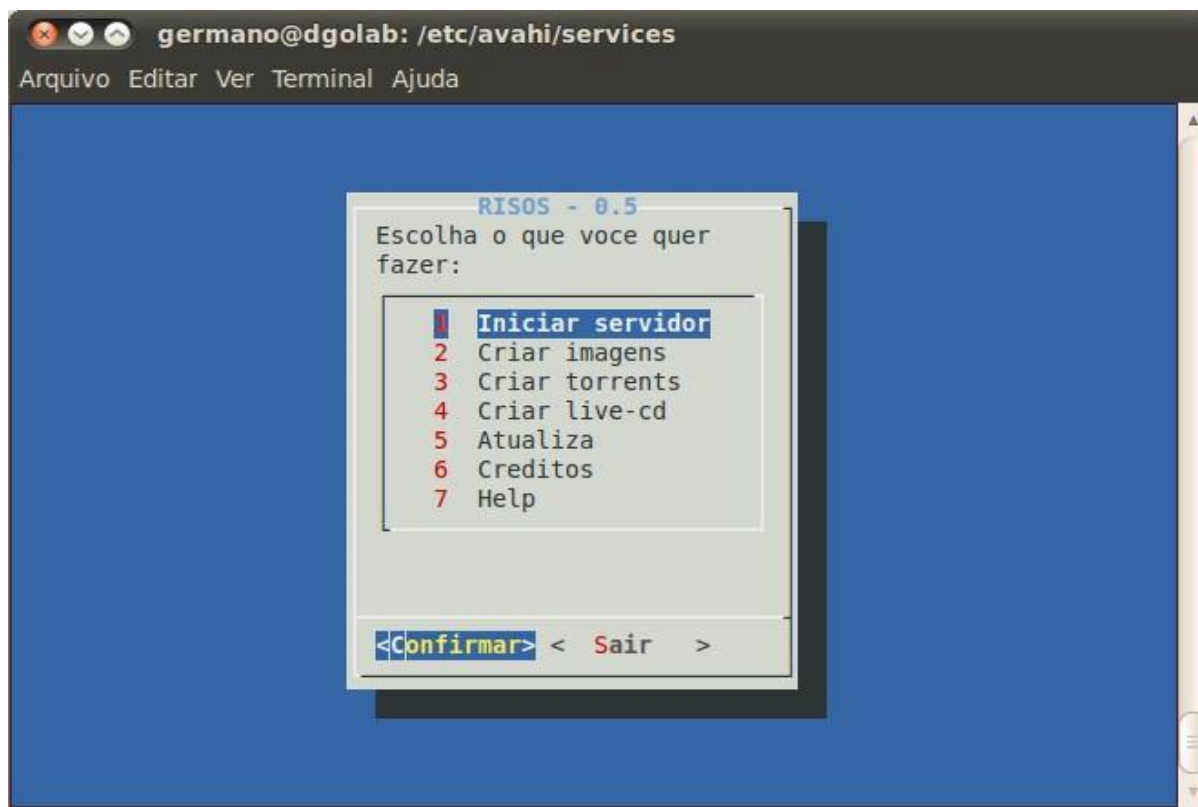
RISOS

Apresentação

Esse é o RISO que fica instalado no computador de referência do qual todos os outros computadores do laboratório serão um espelho, portanto, recomendo que esse computador seja colocado numa sala separada e inacessível aos usuários do laboratório, pois um problema nele será repassado a todos os outros computadores do laboratório.

Executando

Para executar o RISO para servidor abra o terminal e digite “sudo risos”, com isso a seguinte tela será exibida:



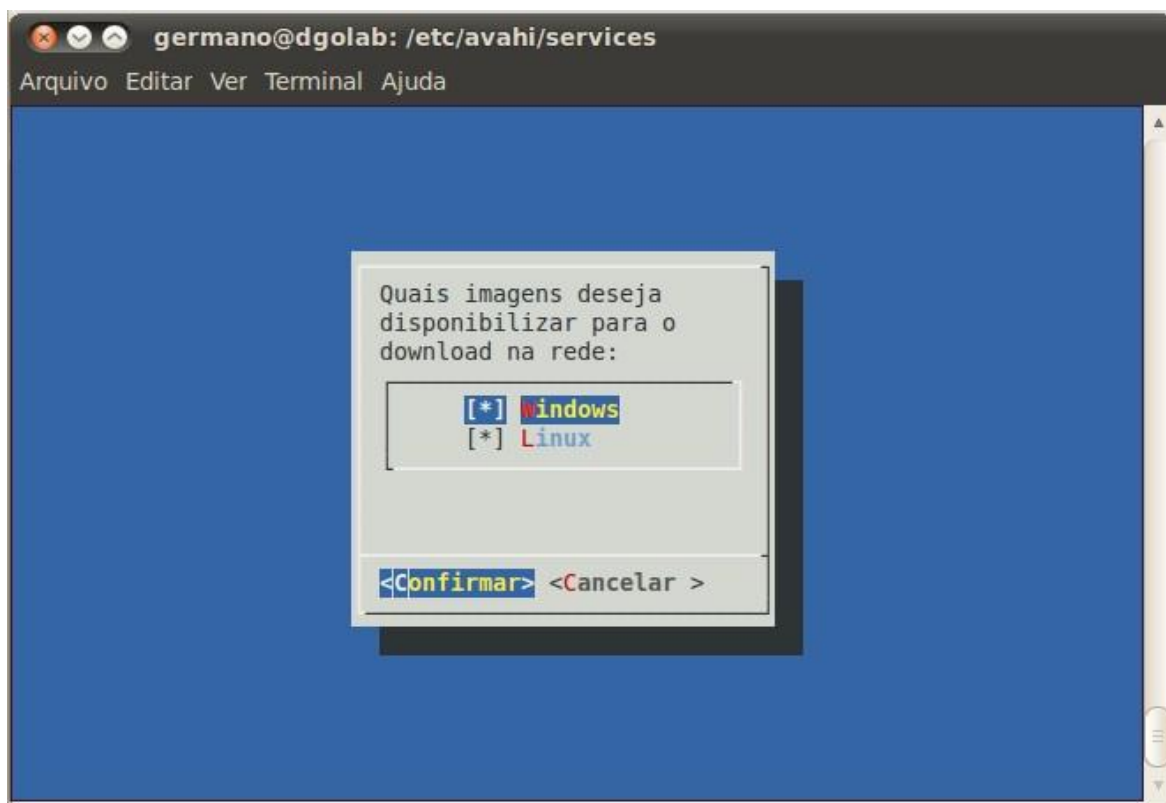
As opções fornecidas por ela são:

1. Iniciar servidor;
2. Criar imagens;
3. Criar torrents;
4. Criar live-cd;
5. Atualizar;
6. Créditos;
7. Help.

Cada uma dessas opções está explicada abaixo.

Iniciar servidor

Nessa opção será exibida a seguinte tela:



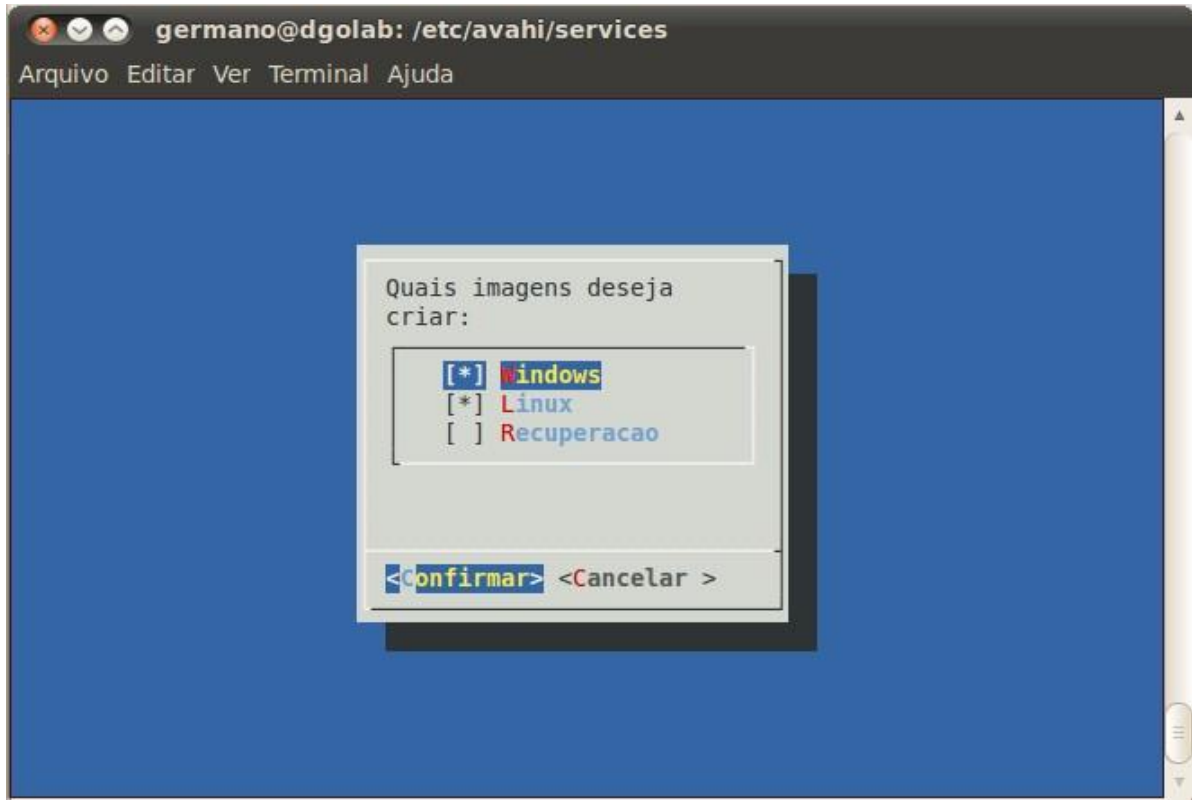
Nessa opção você irá escolher quais imagens deseja disponibilizar na rede se pretende baixar a imagem de ambos os sistemas operacionais deve escolher Windows e Linux se pretende baixar apenas uma delas escolha apenas a opção correspondente, para marcar ou desmarcar uma opção na checklist use “espaço” quando terminar tecle “ENTER”.

Você deve ter percebido que a imagem da partição de recuperação onde o RISO está instalado não está disponível, isso é porque diferente das outras imagens ela não é disponibilizada através do protocolo torrent mas sim via scp, portanto, não precisa que o servidor seja iniciado para ser baixada.

Para parar o servidor e finalizar a distribuição de imagens na rede digite “CTRL + q”.

Criar imagens

Antes de iniciar o servidor para disponibilizar as imagens na rede precisamos criá-las, essa opção nos permitirá fazer isso, as imagens criadas ficarão armazenadas em “/usr/riso/imagens/”, ao escolher essa opção você verá a seguinte tela:



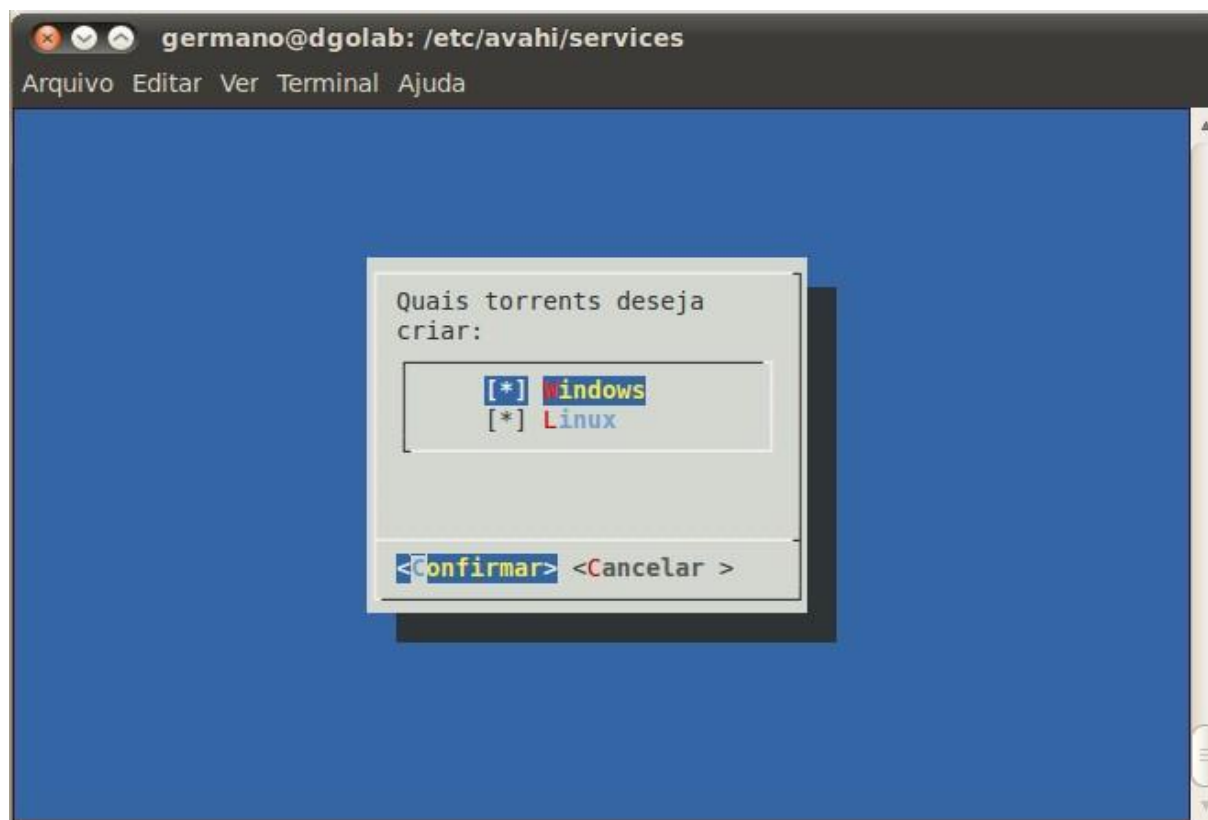
Use “espaço” para marcar ou desmarcar uma opção. O importante ressaltar aqui é que como as imagens do Linux e do Windows ficam armazenadas dentro da partição de recuperação quando criar uma imagem da partição de recuperação todas as imagens existentes serão apagadas. Isso significa que quando criar uma imagem da partição de recuperação terá que criar as imagens do Linux e do Windows novamente. Isso foi feito para deixar a imagem de recuperação com o menor tamanho possível.

Criar torrents

Para disponibilizar as imagens é usado o protocolo torrent. Os arquivos torrent são automaticamente criados quando uma imagem é criada e esses arquivos contêm o endereço IP do computador em que as imagens estão armazenadas. Como o endereço IP do computador de referência pode mudar por algum motivo essa função permite que os arquivos torrent sejam criados sem a necessidade de criar as imagens novamente.

Se o endereço IP do computador de referência for alterado e o arquivo torrent não for recriado as máquinas clientes não conseguiram fazer o download da imagem.

Ao executar essa função a seguinte tela será exibida:



Criar live-cd

Essa opção nos permite criar um LIVE-CD do RISO.

Quando criamos um clone do computador de referência precisamos formatar seu HD para que seu particionamento fique igual ao do computador de referência. Para fazer isso é necessário rodar o RISO a partir de algum outro dispositivo que não o HD, por isso usamos esse CD.

Após executar essa função um arquivo .iso será gerado em “/home/remastersys/riso0.8.iso”. Para gravar essa imagem em um CD insira uma mídia virgem no drive de cd e digite:

```
> cdrecord -v -fs=16M speed=32 dev=1,0,0 -data /home/remastersys/riso0.8.iso
```

o seu dispositivo de dev pode ser outro, para identificar o seu dev corretamente use:

```
> cdrecord -scanbus
```

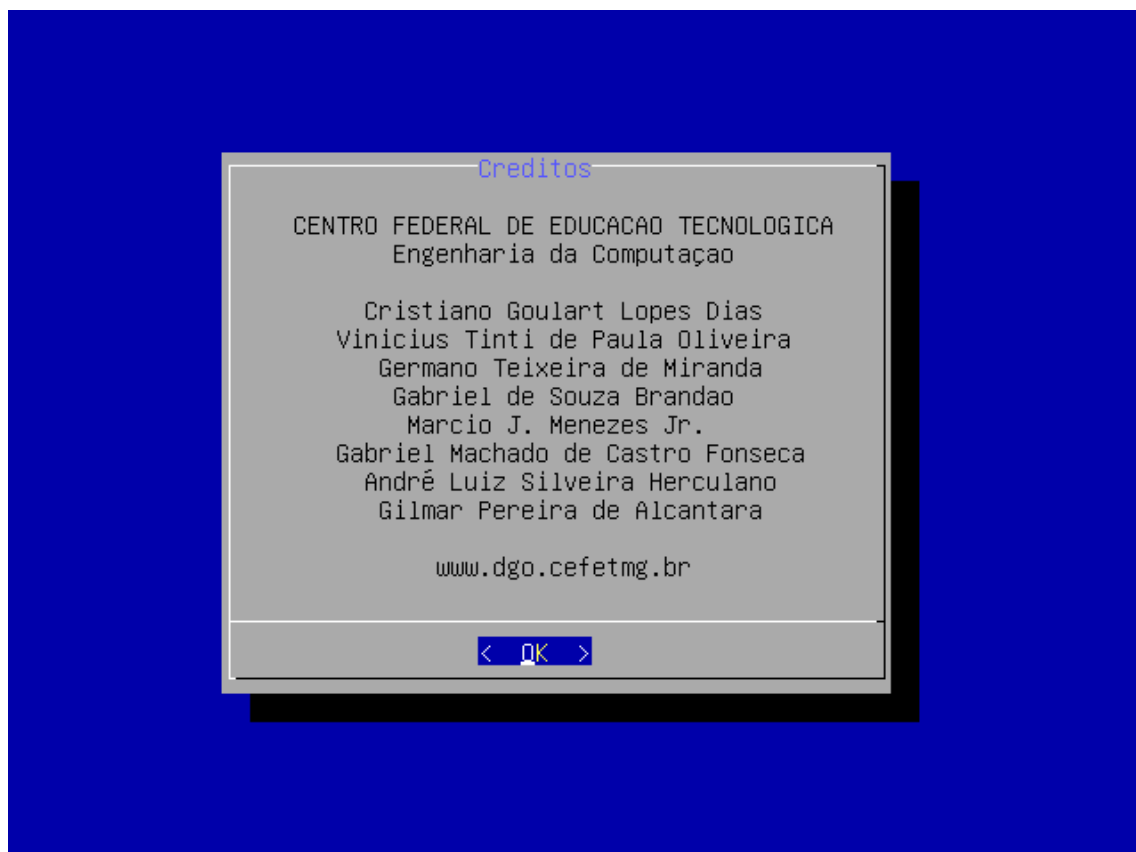
Atualizar

Essa é a opção que permite atualizar o RISO, quando executá-la uma nova versão do RISO será instalada.

As versões do RISO não são retro compatíveis, portanto quando fizer a atualização do RISOS no servidor atualize também o RISO no cliente, o mesmo vale para o caso oposto.

Créditos

Essa opção exibe a seguinte tela:



Esse é um agradecimento às pessoas que ajudaram a desenvolver o sistema RISO, se algo der errado durante a utilização do sistema a culpa é deles.

Help

Diferente do que ocorre normalmente em outros softwares, nessa opção você não verá uma explicação sobre as funções do programa, isso já foi feito por esse documento.

Nessa opção você vai encontrar um e-mail para o qual poderá mandar suas duvidas e sugestões que responderei o mais rápido possível.

Talvez um dia alguma alma caridosa se disponha a fazer um help melhorzinho, até lá é só isso mesmo.

RISO

Apresentação

Esse é o RISO que fica instalado nos demais computadores do laboratório, ele vai ser o responsável por baixar e instalar as imagens criadas e disponibilizadas pelo servidor criando assim uma cópia perfeita do computador de referência.

Após a instalação o código pode ser inicializado e usado. Para a compreensão do código e

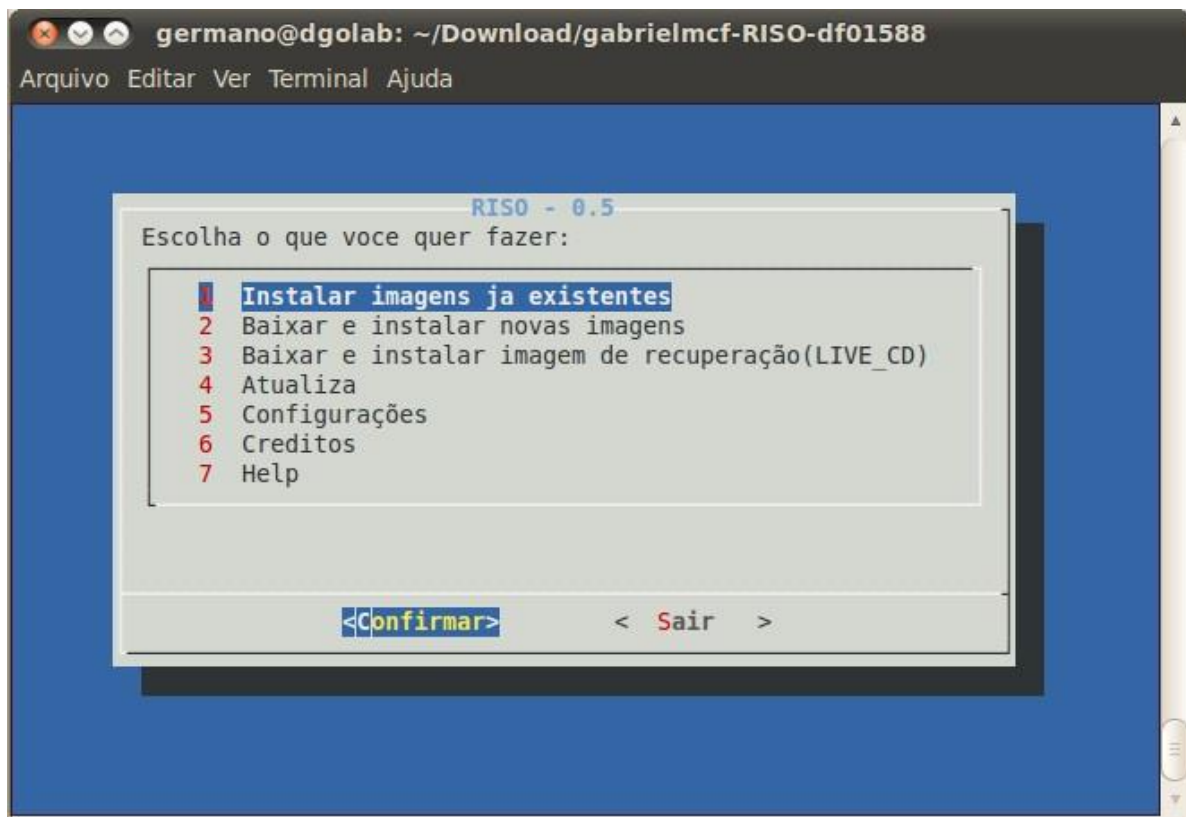
viável entender que Shell-Script é uma linguagem estrutural logo a função é somente executada se já foi implementada, em termos mais coloquiais o código é executado “de baixo para cima”.

O sistema tem como layout o dialog, que abre janelas com opções e mensagens, isso para que se tenha uma interface mais amigável com o usuário, informações podem ser encontradas em <http://aurelio.net/shell/dialog/>

Inicialmente temos um sistema de telas de menus que executam determinadas funções de acordo com a necessidade do usuário.

Executando

Para executar o RISO para cliente abra o terminal e digite “sudo riso”, com isso a seguinte tela será exibida:



As opções fornecidas por ela são:

1. Instalar imagens já existentes;
2. Baixar e instalar novas imagens;
3. Baixar e instalar imagem de recuperação (LIVE-CD);
4. Atualizar;
5. Configurações;
6. Créditos;
7. Help.

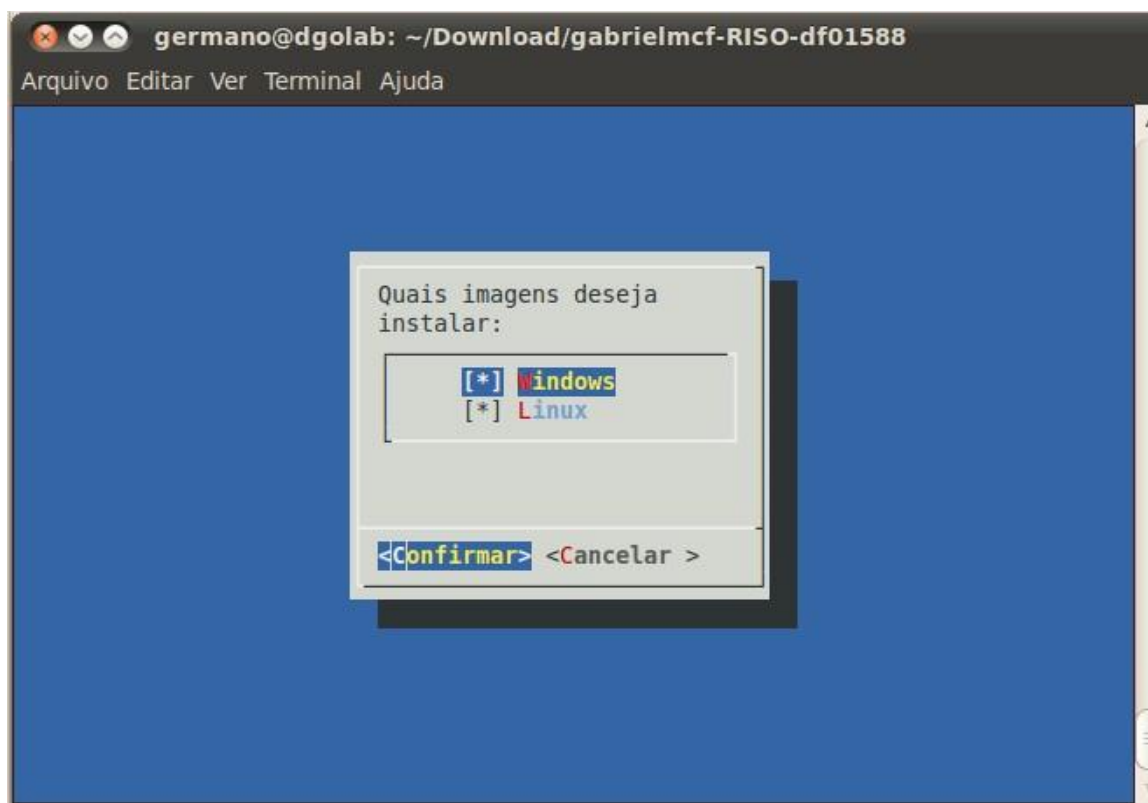
Cada uma dessas opções está explicada abaixo.

Instalar imagens já existentes

A cada vez que o RISO baixa imagens do computador de referência ele as armazena em “/usr/riso/imagens/”, com isso podemos instalá-las sem fazer o download das imagens no computador de referência.

A grande vantagem dessa opção é que ela não baixa as imagens novamente antes de instalá-las, uma mão na roda a resolver problemas decorrentes da utilização como os vírus, um problema constante no Windows.

Essa opção irá exibir a seguinte tela:



Use “espaço” para marcar ou desmarcar uma opção.

O Windows é gerado e restaurado como o ntfsclone, pelo comando :
ntfsclone -r \$img_win -O \$partwindows , onde \$img_win representa o conteúdo da variável img_win que é igual a /usr/riso/imagens/windows.img e \$partwindows representa o conteúdo da variável partwindows que é a partição onde o Windows está instalado /dev/sda1 no nosso caso, mas pode ser qualquer outra partição.

Já, para instalar o Linux, e montada a partição dele, e ela é somente descompactada no local onde foi montada, no nosso caso é /mnt. A imagem do Linux é um tar, e é criada e gerada pela simples compactação e descompactação da partição onde o Linux se encontra instalado.
O comando de instalação é tar -xvf \$img_lin e é feito onde a partição esta montada.

Baixar e instalar novas imagens

É importante ter em mente que todo o funcionamento do riso passa pela saída do avahi, pois a usa

para carregar as variáveis. Partindo do ponto que todas as variáveis estão carregadas corretamente, o programa vai funcionar.

Para baixar as imagens o riso pega os arquivos de windows.img.torrent e linux.tar.torrent via ssh (comando scp), e inicia o rtorrent (programa para distribuição via torrent de arquivos em uma rede), usando os arquivos de torrent como parâmetro. Após isso ele começa baixar, e a imagem do servidor é compartilhada com outros computadores da rede.

Obs: é bom ter em mãos alguns comandos de configuração de rede, pois para que as imagens sejam baixadas com sucesso, os computadores têm que estar conectados entre si e com o servidor risos.

Apos baixar a imagens ele as instala. A princípio ele compara os tamanhos da imagem que estão sendo baixadas com o tamanho da que está no servidor (valor de variável fornecido pela saída do avahi) e carrega uma barra de status. Esse procedimento gerava alguns erros e então foi substituído.

Um laço mantinha o download das imagens, até os tamanhos das imagens (local) e do (servidor) serem do mesmo tamanho, isso foi possível fazendo a execução do rtorrent em background. A alteração consiste em deixar essa execução em primeiro plano, o que nos dá uma interface idêntica a do servidor, que permite o acesso a várias informações como ver o ip do servidor e ter acesso a quais ips estão fazendo download da imagem.

Para que possa fazer a instalação da imagem após o termino do download, foi necessária a criação de um arquivo oculto chamado (.rtorrent.rc) na home do usuário, no caso o root. Nesse é possível setar a execução de um outro, que finaliza o rtorrent caso o dowload seja concluído e a função de instalação possa ser chamada.

Esse arquivo também nos permitiu resolver uma limitação de tamanho da imagem a ser baixada e de memória, pois podemos setar o tamanho máximo que algum arquivo transferido via rtorrent possa ter (no nosso caso as imagens), e setar a memória a ser utilizada.

O conteúdo desse arquivo se encontra abaixo:

[illegible]

A ausência desse, causa problemas, pois se a imagem for muito grande não é possível fazer o download das imagens, isso foi constatado na migração do sistema para Windows 7 que ocupa bem mais memória que o Windows XP. Além de termos uma maneira viável de finalizar o torrent no final do download.

Na parte gráfica de baixar imagens ela exibe essa seguinte tela para o usuário:



Use “espaço” para marcar e desmarcar itens.

Aqui você irá selecionar as imagens que deseja baixar e instalar no computador, é importante lembrar que para baixar uma imagem ela deve antes ter sido previamente disponibilizada pelo RISO do servidor.

O RISO carrega as variáveis usando a saída do aplicativo avahi, essa saída é trabalhada e dividida, para poder obter o conteúdo de cada variável. Como já foi dito acima essa saída está intimamente ligada ao /etc/avahi/services/riso.service que é escrito na instalação do risos, isso no servidor que disponibiliza o conteúdo do arquivo na rede. Logo somente o **servidor pode ter o etc/avahi/services/riso.service**.

Cada computador cliente tem acesso a essa saída, ao ser especificado o nome do serviço, e pode carregar as variáveis, baseado no servidor, essas informações também são redirecionadas para o arquivo /usr/riso/riso.part.

Ao iniciar o riso ele carrega as variáveis com a saída do avahi (ver função carrega_variaveis) e chama o menu para escolher as opções adequadas a funcionalidade desejada no momento. Logo é necessário que o avahi esteja em funcionamento e o computador cliente esteja conectado com o servidor. Já que o avahi disponibiliza serviços pela rede, no caso do riso se resume ao conteúdo de um arquivo.

Baixar e instalar imagem de recuperação (LIVE-CD)

Você deve ter percebido que a partição de recuperação foi ignorada pelas opções Instalar imagens e Baixar e instalar novas imagens, isso é porque o RISO não consegue restaurar uma partição e usá-la ao mesmo tempo.

Para baixar e instalar a imagem de recuperação você terá que executar essa opção a partir de um LIVE-CD do RISO, foi isso que você fez quando criou o clone do computador de referência.

A boa notícia é que essa opção só precisava ser executada uma vez, na primeira vez que a cópia do computador de referência for criada.

Diferente do Windows e do Linux que são baixados via torrent, nesse processo ele não é usado, por motivos óbvios, já que o rtorrent e o riso estão instalados na recuperação e nesse processo irá ser substituída.

Com essa opção o riso reescrever o disco, e altera o sistema de partições, ele é usado no início do uso do riso, mais informações no manual.

Ele baixa a imagem rec.tar, e a iso que será gravada são criadas pelo riso.

A iso se encontra em /home/remastersys e pode ser gravada no Live CD ou DVD (dependendo do tamanho), (obs.: ao gravá-la num pendrive dá erro e não acontece o boot).

Quando é executada essa opção, é baixada uma nova tabela e partição criada quando criamos a recuperação (usa-se sfdisk -d para criá-la e o sfdisk também para escrevê-la), e após isso ocorre a montagem da nova partição da recuperação, baixa e instala como se fosse o Linux (descompactando). Depois desse processo teremos somente a recuperação instalada.

Atualizar

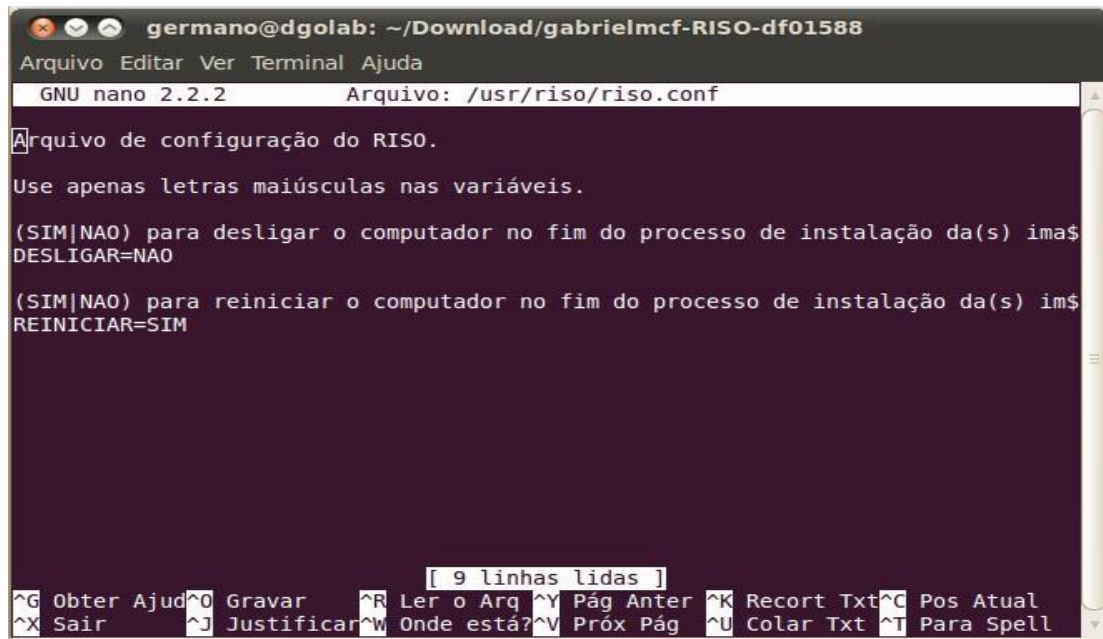
Essa opção é a mesma do RISOS e já foi explicada no tópico Atualizar anterior.

A opção de Atualizar irá baixar uma nova versão do riso num link especificado, no nosso caso no Guit-Hub.

Configurações

Quando o RISO termina de instalar uma imagem ele por padrão reinicia o computador. Aqui você pode alterar o comportamento do computador ao terminar o processo, você pode configurá-lo para desligar ao invés de reiniciar.

Ao selecionar essa opção será exibido o seguinte arquivo:



```
germano@dgolab: ~/Download/gabrielmcf-RISO-df01588
Arquivo Editar Ver Terminal Ajuda
GNU nano 2.2.2 Arquivo: /usr/riso/riso.conf
Arquivo de configuração do RISO.
Use apenas letras maiúsculas nas variáveis.
(SIM|NAO) para desligar o computador no fim do processo de instalação da(s) ima$
DESLIGAR=NAO
(SIM|NAO) para reiniciar o computador no fim do processo de instalação da(s) im$
REINICIAR=SIM
[ 9 linhas lidas ]
^G Obter Ajud ^O Gravar ^R Ler o Arq ^Y Pág Anter ^K Recort Txt ^C Pos Atual
^X Sair ^J Justificar ^W Onde está? ^V Próx Pág ^U Colar Txt ^T Para Spell
```

- 2 LIVE-CD de um sistema operacional e um sistema operacionais que roda direto do CD de boot, sem usar o HD
Mude os valores na frente de cada variável para obter o comportamento desejado.

Créditos

Essa opção exibe a mesma tela do RISOS e já foi explicada no tópico Créditos anterior.

Help

Essa opção também exibe a mesma tela do RISOS e já foi explicada no tópico Help anterior.

CÓDIGO COMENTADO

Abaixo segue o código comentado do RISO, para um maior entendimento é necessário saber o básico de Linux e Shell Script, como dito anteriormente. Nas referências do manual estão alguns links onde você pode se aprofundar mais nessas linguagens. Mas não fique preso somente nas referências, procure apostilas, vídeo aulas etc. Oque for melhor para o desenvolvimento de sua aprendizagem.

Instalação

```
#!/bin/bash

# -----
#=====
#                               |
# Arquivo de instalação do riso e do risos|
#                               |
#=====
# -----

#Essas são as variáveis, nelas estão contidos todos os programas necessários para
o funcionamento correto do riso.
dep_cli="avahi-utils rtorrent screen ntfsprogs ssh"
dep_ser="avahi-utils avahi-daemon bittorrent rtorrent screen ntfsprogs grub2 ssh
coreutils mkisofs genisoimage findutils bash passwd sed squashfs-tools casper
rsync mount eject libdebian-installer4 os-prober ubiquity user-setup discover
laptop-detect syslinux xterm util-linux xresprobe cdrecord"

#Essa função concerta o grub do sistema.
DisableUUID() {

#lê o arquivo grub.cfg, pega todas as linha que tem UUID.
b=`cat /boot/grub/grub.cfg | grep =UUID=`

#separa somente um (EXEMPLO) UUID=33assad3215465213, todos os UUID tem o mesmo
valor. Mas esse UUID é o antigo.
uuidUbuntu=`(cut -d ' ' -f3 | cut -d '=' -f2-3)<<<${b}`

#lista todas as partições com o comando os-prober e pega a partição do linux com o
cut.
partLinux=`os-prober | grep linux | cut -d ':' -f1`

#substitui no arquivo do grub.cfg o UUID antigo (errado) pela partição do linux.
sed "s|${uuidUbuntu}|${partLinux}|g" -i /boot/grub/grub.cfg

}

#Logo após o usuário ser reconhecido como root, ele chama a função menu, que se
encontra logo abaixo. Essa função define quais sistemas serão instalados.
menu() {
```

apt-get install -y dialog #instala o dialog, o -y serve para forçar as instalações e não precisar de verificação.

```
opcao=$( dialog --stdout \
--title 'Instalação do RISO' \
--ok-label 'Confirmar' \
--checklist 'Deseja instalar:' \
0 0 0 \
RISOS '' ON \ #deixa a opção On pré-selecionada.
RISO '' ON )   #deixa a opção On pré-selecionada.

# De acordo com a opção escolhida, executa funções
case $opcao in
  "RISOS") instala_risos;;
  "RISO") instala_riso;;
  "RISOS" "RISO") instala_riso; instala_risos;;
esac
}
```

#Essa função como próprio nome diz, instala o riso. Mais abaixo ela está detalhadamente comentada.

```
instala_riso() {

    echo "Instalando Cliente (RISO)..."

    echo "Baixando dependências..."

    #instala o rtorrent, screen, ntfsplogs e o ssh.
    apt-get install -y $dep_cli

    echo "Criando árvore de diretórios..."

    #cria o diretório riso dentro do diretório usr
    mkdir -p /usr/riso.
    #cria o diretório de imagens dentro do diretório riso.
    mkdir -p /usr/riso/imagens

    echo "Criando arquivo de inicialização..."

    #redireciona o que está entre ' para /usr/bin/riso e se houver algo lá escrito,
    ele sobrescreve.
    echo '#!/bin/bash' > /usr/bin/riso

    #redireciona o que está entre ' para /usr/bin/riso adicionando aquela linha no
    final.
    echo '/usr/riso/riso $@' >> /usr/bin/riso

    #da permissão de execução no /usr/bin/riso. Nesse caso o riso.
    chmod +x /usr/bin/riso

    #Exibe mensagem para o usuário.
    echo "Movendo script..."

    #copia o riso que está no diretório src para o diretório usr/riso/riso.
    cp ./src/riso /usr/riso/riso

    #da permissão de execução no riso.
    chmod +x /usr/riso/riso
}
```

```
#copia o riso.conf para /usr/riso/riso.conf.
cp ./conf/riso.conf /usr/riso/riso.conf

#imprime as duas mensagens finais para o usuário.
echo "Sistema instalado com sucesso."
echo "Tente digitar 'riso' para iniciar."
}
```

#Essa é a função onde o risos é instalado, logo abaixo ela está detalhadamente explicada.

```
instala_risos() {
    echo "Instalando Servidor (RISOS)..."

    #libera o login como root.
    echo "Liberando login como root..."
    echo -n "root:" > /etc/shadow.tmp
    pass=`cat /etc/shadow | grep \`who | awk 'NR==1{print $1}`\` | cut -d':' -f2`
    echo -n "${pass}:" >> /etc/shadow.tmp
    cat /etc/shadow | grep root | cut -d':' -f3,4,5,6,7,8,9 >> /etc/shadow.tmp
    cat /etc/shadow | grep -v root >> /etc/shadow.tmp
    mv /etc/shadow.tmp /etc/shadow

    echo "Baixando dependências..."

    #instala todos os programas contidos na variável $dep_ser.
    apt-get install -y $dep_ser

    echo "Criando árvore de diretórios..."

    #cria o diretório riso dentro da pasta usr.
    mkdir -p /usr/riso
    #cria o diretório imagens dentro de riso
    mkdir -p /usr/riso/imagens

    echo "Criando arquivo de inicialização..."

    #redireciona o que está entre '#' para /usr/bin/risos e se houver algo lá escrito
    ele sobrescreve.
    echo '#!/bin/bash' > /usr/bin/risos

    #redireciona o que está entre '#' para /usr/bin/risos.
    echo '/usr/riso/risos $@' >> /usr/bin/risos

    #da permissão de execução no risos.
    chmod +x /usr/bin/risos

    echo "Movendo script..."

    #copia o risos que está no diretório src para o diretório usr/riso/risos.
    cp ./src/risos /usr/riso/risos

    #da permissão de execução no risos.
    chmod +x /usr/riso/risos

    echo "Gerando chaves rsa..."
}
```

```

#SSH sem senhas e autenticação através de certificados RSA. Para transferência de
arquivos, no caso o torrente, evitando o uso de senhas.
ssh-keygen -t 'rsa' -f '/root/.ssh/id_rsa' -N ''
su -c "cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys2"

echo "Criando variáveis de configuração..."

#Pegando partições do sistema. Exemplo /dev/sdaX

#pega a partição de recuperação do computador.
partrec=`df -T | awk 'NR==2{print $1}'`

#pega a partição do windows do computador.
partwindows=`os-prober | grep Windows | cut -d':' -f1`

#pega a partição do linux do computador.
partlinux=`os-prober | grep linux | cut -d':' -f1`

#pega a partição de swap do computador.
partswap=`blkid | grep swap | grep -v ram | cut -d':' -f1`

#Pegando sistema de arquivo das partições. Exemplo ext e ntfs.

#pega o sistema de arquivos da partição de recuperação.
sa_partrec=`df -T | awk 'NR==2{print $2}'`

#pega o sistema de arquivos da partição do Linux.
sa_partlinux=`blkid ${partlinux} | cut -d'"' -f4`

#pega o sistema de arquivos da partição do Windows.
sa_partwindows=`blkid ${partwindows} | cut -d'"' -f4`

#Prepara grub2

#sed serve para substituir trechos de textos nos arquivos.

#então ele está editando algumas linhas de configuração do grub, que se encontram
no arquivo /etc/default/grub/
sed s/'GRUB_DEFAULT=0'/'GRUB_DEFAULT=1'/g -i /etc/default/grub/
sed s/'GRUB_DISTRIBUTOR=*'/'#GRUB_DISTRIBUTOR=''/g -i /etc/default/grub
echo 'GRUB_DISTRIBUTOR=Recuperacao' >> /etc/default/grub
sed s/'#GRUB_DISABLE_LINUX_UUID=true'/'GRUB_DISABLE_LINUX_UUID=true'/g -i
/etc/default/grub
sed
s/'#GRUB_DISABLE_LINUX_RECOVERY="true"'/'GRUB_DISABLE_LINUX_RECOVERY="true"'/g -i
/etc/default/grub
rm -f /etc/grub.d/20_memtest86+
mv /etc/grub.d/10_linux /etc/grub.d/50_linux
update-grub

#chama a função que desabilita o UUID, ela está explicada no início do código.
DisableUUID

#Cria riso.service, que é o arquivo que servidor. Ele é usado para carregar todas
as variáveis do sistema através do avahi-browser.

```

```

#Tudo que está escrito dentro das " " do echo é redirecionado para o riso.service
que está em /etc/avahi/service/riso.service
echo "Criando arquivo de configuração..."
echo '<?xml version="1.0" standalone="no"?><!--*-nxml-*-->' >
/etc/avahi/services/riso.service
echo '<!DOCTYPE service-group SYSTEM "avahi-service.dtd">' >>
/etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo '<!--riso.service -->' >> /etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo '<!--' >> /etc/avahi/services/riso.service
echo ' Arquivo com as variáveis de configuração do riso.' >>
/etc/avahi/services/riso.service
echo ' Essas variáveis são usadas para efetuar a comunicação do servidor
riso(risos) com os cliente(riso).' >> /etc/avahi/services/riso.service
echo ' Para mais informações sobre o processo ver http://avahi.org' >>
/etc/avahi/services/riso.service
echo '-->' >> /etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo '<service-group>' >> /etc/avahi/services/riso.service
echo ' <name>Servidor RISO</name>' >> /etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' <service>' >> /etc/avahi/services/riso.service
echo ' <!--Nome do serviço-->' >> /etc/avahi/services/riso.service
echo " <type>_RISO._tcp</type>" >> /etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' <!--Campo não é usado-->' >> /etc/avahi/services/riso.service
echo ' <port>1234</port>' >> /etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' <!--Variáveis com sistema de arquivo das partições-->' >>
/etc/avahi/services/riso.service
echo " <txt-record>sa_partrec=${sa_partrec}</txt-record>" >>
/etc/avahi/services/riso.service
echo " <txt-record>sa_partlinux=${sa_partlinux}</txt-record>" >>
/etc/avahi/services/riso.service
echo " <txt-record>sa_partwindows=${sa_partwindows}</txt-record>" >>
/etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' <!--Variáveis com tamanho das imagens geradas no servidor-->' >>
/etc/avahi/services/riso.service
echo ' <txt-record>tamlinux=123456789</txt-record>' >>
/etc/avahi/services/riso.service
echo ' <txt-record>tamwindows=123456789</txt-record>' >>
/etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' <!--Localização dos sistemas operacionais no HD-->' >>
/etc/avahi/services/riso.service
echo " <txt-record>partswap=${partswap}</txt-record>" >>
/etc/avahi/services/riso.service
echo " <txt-record>partrec=${partrec}</txt-record>" >>
/etc/avahi/services/riso.service
echo " <txt-record>partlinux=${partlinux}</txt-record>" >>
/etc/avahi/services/riso.service
echo " <txt-record>partwindows=${partwindows}</txt-record>" >>
/etc/avahi/services/riso.service
echo '' >> /etc/avahi/services/riso.service
echo ' </service>' >> /etc/avahi/services/riso.service
echo '</service-group>' >> /etc/avahi/services/riso.service

```



```

# preinst script for remastersys

#verifica se /remastersys é um diretório.
  if [ -d /remastersys ]; then
#se for um diretório, move o /remastersys para /home.
  mv /remastersys /home
  fi

#verifica se o /etc/remastersys.conf é um arquivo normal.
  if [ -f /etc/remastersys.conf ]; then
#se for, move ele para /etc/remastersys.conf.old.
  mv /etc/remastersys.conf /etc/remastersys.conf.old
  fi

#verifica se o splash.pcx é um arquivo normal.
  if [ -f /etc/remastersys/isolinux/splash.pcx ]; then
#se for, move ele para /etc/remastersys/isolinux/splash.pcx.saved.
  mv /etc/remastersys/isolinux/splash.pcx
/etc/remastersys/isolinux/splash.pcx.saved
  fi

#verifica se o splash.rle é um arquivo normal.
  if [ -f /etc/remastersys/isolinux/splash.rle ]; then
#se for, move ele para /etc/remastersys/isolinux/splash.rle.saved.
  mv /etc/remastersys/isolinux/splash.rle
/etc/remastersys/isolinux/splash.rle.saved
  fi

#verifica se o splash.xpm.gz é um arquivo normal.
  if [ -f /etc/remastersys/grub/splash.xpm.gz ]; then
#se for, move ele para /etc/remastersys/grub/splash.xpm.gz.saved.
  mv /etc/remastersys/grub/splash.xpm.gz
/etc/remastersys/grub/splash.xpm.gz.saved
  fi

#Instalação do remastersys

#mkdir -p serve para não reportar o erro e cria uma pasta dentro da outra. Nesse
caso está criando a o remastersys dentro de etc.
  mkdir -p /etc/remastersys
#copia /grub para /etc/remastersys.
  cp -r ./src/remastersys/remastersys/grub /etc/remastersys
#copia isolinux para /etc/remastersys.
  cp -r ./src/remastersys/remastersys/isolinux /etc/remastersys
#copia preseed para /etc/remasterstys.
  cp -r ./src/remastersys/remastersys/preseed /etc/remastersys
#copia remastersys.conf para /etc/remastersys.conf.
  cp ./src/remastersys/remastersys.conf /etc/remastersys.conf

# postinst script for remastersys
#verifica se o splash.pcx.saved é um arquivo normal.
  if [ -f /etc/remastersys/isolinux/splash.pcx.saved ]; then
#se for, move ele para /etc/remastersys/isolinux/splash.pcx.
  mv /etc/remastersys/isolinux/splash.pcx.saved
/etc/remastersys/isolinux/splash.pcx
  fi

#verifica se o splash.rle.saved é um arquivo normal.

```

```

    if [ -f /etc/remastersys/isolinux/splash.rle.saved ]; then

#se for, move ele para /etc/remastersys/isolinux/splash.rle.
mv /etc/remastersys/isolinux/splash.rle.saved /etc/remastersys/isolinux/splash.rle
    fi

#verifica se o splash.xmp.gz.saved é um arquivo normal.
    if [ -f /etc/remastersys/grub/splash.xpm.gz.saved ]; then
#se for, move ele para /etc/remastersys/isolinux/splash.xpm.gz
    mv /etc/remastersys/grub/splash.xpm.gz.saved
/etc/remastersys/isolinux/splash.xpm.gz
    fi

#verifica se o remastersys.conf.old é um arquivo normal.
    if [ -f /etc/remastersys.conf.old ]; then
#se for, move ele para /etc/remastersys.conf.
    mv /etc/remastersys.conf.old /etc/remastersys.conf
    fi

#Se estiver usando windows 7. Muda arquivo de boot para não usar      mais UUID
e sim um arquivo binário BCD.
    windows7=`os-prober | grep "Windows 7"`
    if [ ! -z "$windows7" ]; then
        echo "Configurando Windows 7..."
        umount /mnt 2> /dev/null
        mount ${partwindows} /mnt
        cp ./conf/BCD /mnt/Boot/BCD
        umount /mnt 2> /dev/null
    fi

    echo "Sistema instalado com sucesso"
    echo "Tente risos para iniciar"
}

```

#Verifica se usuário é o root antes de executar.

#Todo usuário tem um número padrão, sendo que o root o seu número é 0. Então se o número do usuário for igual a 0 ele entra como root.

```

USER=`id -u`
if [ $USER == '0' ]; then
    menu
else
    echo "Só o root pode fazer isso, juvenzinho!"
fi

```

Risos

```
#!/bin/bash
```

```

#Variáveis do script
dir_riso=/usr/riso #diretório do riso
dir_img=$dir_riso/imagens #diretório da imagem
img_win=$dir_img/windows.img #arquivo da imagem do Windows
img_lin=$dir_img/linux.tar #arquivo da imagem do Linux

```

```

img_rec=$dir_img/rec.tar           #arquivo da imagem da recuperação (ubuntu_server)
tor_win=$img_win.torrent           #torrent da imagem do windows
tor_lin=$img_lin.torrent           #torrent da imagem do Linux
riso_service=/etc/avahi/services/riso.service #serviço do servidor de imagem
tab_part=$dir_img/tab_part         #arquivo onde fica a tabela de partição
riso_part=$dir_riso/riso.part
arq_log=$dir_riso/riso.log         #Arquivo de log

#Função de Log
log() {
    [ -e $arq_log ] || touch $arq_log
    echo `date +%F %H:%M:%S` "RISOS: $" >> $arq_log
}

#Carrega variaveis do arquivo riso.service para variáveis do sistema.
carrega_variaveis() {
    avahi_saida=`avahi-browse -rtcp _DECOM_RISO._tcp | grep '='`
    servidor=`cut -d ';' -f8 <<< ${avahi_saida}` #Coloca em uma variável o ip do
servidor.
    if [ -z $servidor ]; then
        dialog \
            --title 'Atencao' \
            --ok-label 'OK' \
            --msgbox 'O servidor RISOS precisa estar conectado à rede.\nVerifique a
conexão e tente novamente.' \
            7 45
        log "ERRO: Servidor não encontrado na rede."
        exit 1
    fi
    partwindows=`(cut -d ';' -f10 | cut -d '"' -f2 | cut -d '=' -f2) <<<
${avahi_saida}` #Partição com Windows.
    partlinux=`(cut -d ';' -f10 | cut -d '"' -f4 | cut -d '=' -f2) <<<
${avahi_saida}` #Partição com Linux.
    partrec=`(cut -d ';' -f10 | cut -d '"' -f6 | cut -d '=' -f2) <<<
${avahi_saida}` #Partição de Recuperação.
    partswap=`(cut -d ';' -f10 | cut -d '"' -f8 | cut -d '=' -f2) <<<
${avahi_saida}` #Partição Swap.

    #Tamanho total das imagens criadas no servidor.
    tamwindows=`(cut -d ';' -f10 | cut -d '"' -f10 | cut -d '=' -f2) <<<
${avahi_saida}` #tamanho padrão 132456789.
    tamlinux=`(cut -d ';' -f10 | cut -d '"' -f12 | cut -d '=' -f2) <<<
${avahi_saida}` #tamanho padrão 123456789.

    #Sistema de arquivo das partições.
    sa_partwindows=`(cut -d ';' -f10 | cut -d '"' -f14 | cut -d '=' -f2) <<<
${avahi_saida}` #sa_partwindows=/dev/sda1
    sa_partlinux=`(cut -d ';' -f10 | cut -d '"' -f16 | cut -d '=' -f2) <<<
${avahi_saida}` #sa_partlinux=/dev/sda2
    sa_partrec=`(cut -d ';' -f10 | cut -d '"' -f18 | cut -d '=' -f2) <<<
${avahi_saida}` #sa_partrec=/dev/sda3

    #Atualiza arquivo de variaveis.
    echo 'partwindows='$partwindows > $riso_part #escreve a partição do windows
/dev/sda1 em $riso_part.
    echo 'partlinux='$partlinux >> $riso_part #escreve a partição do linux
/dev/sda2 em $riso_part.
    echo 'sa_partwindows='$sa_partwindows >> $riso_part #escreve a sa_partwindows
que é ntfs em $riso_part.

```

```

    echo 'sa_partlinux='${sa_partlinux} >> $riso_part #escreve a sa_partlinux que é
ext4.
}

#Inicia tracker para disponibilizar imagens na rede.
inicia_servidor() {
#essa função levanta as 2 imagens a serem baixadas, assim que essas imagens
estiverem prontas, o download poderá ser feito.

    if [ -e ${img_win} -a -e ${img_lin} ]; then #verifica se a imagem do windows e
linux são existentes.
        log "Tracker iniciado para Windows e Linux."
        btttrack --port 6969 --dfile dstate | grep laialadaiasabadanaavemaria &
        cd ${dir_img} #entra no diretório /usr/riso/imagens.
        rtorrent ${tor_lin} ${tor_win}
        log "Tracker finalizado para Windows e Linux."
    else
        dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\n Não existem imagens prontas.' \
        7 40
        log "ERRO: Ao iniciar tracker para Windows e Linux, Não existem imagens
prontas."
        return 2
    fi
    #verificar se a função está levantando as 2 imagens com o Gilmar.
}

#Inicia tracker para disponibilizar imagem do Windows na rede.
inicia_servidor_windows() {
#Essa função vai no diretório de imagens do riso, pega a imagem do windows que foi
criada, verifica se ela é existente.
#Se a imagem for existente, levanta ela na rede e disponibiliza o download para as
maquinas clientes.

    if [ -e ${img_win} ]; then #verifica se a imagem do windows é existente.
        log "Tracker iniciado para Windows."
        btttrack --port 6969 --dfile dstate | grep laialadaiasabadanaavemaria &
        cd ${dir_img} #entra no diretório de imagens /usr/riso/imagens.
        rtorrent ${tor_win}
        log "Tracker finalizado para Windows."
    else
        dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\nVocê ainda não criou a imagem do windows.' \
        7 45
        log "ERRO: Ao iniciar tracker para Windows, Não existe imagem pronta."
        return 2
    fi
}

#Inicia tracker para disponibilizar imagem do Linux na rede.
inicia_servidor_linux() {
#Essa função vai no diretório de imagens do riso, pega a imagem do windows que foi
criada, verifica se ela é existente.

```

#Se a imagem for existente, levanta ela na rede e disponibiliza o download para as maquinas clientes.

```
if [ -e ${img_lin} ]; then #verifica se a imagem do linux é existente.
    log "Tracker iniciado para Linux."
    btrack --port 6969 --dfile dstate | grep laialadaiasabadanaavemaria &
    cd ${dir_img} #entra no diretório de imagens /usr/riso/imagens.
    rtorrent ${tor_lin}
    log "Tracker finalizado para Linux."
else
    dialog \
    --title 'Atenção' \
    --ok-label 'OK' \
    --msgbox '\nVocê ainda não criou a imagem do linux.' \
    7 44
    log "ERRO: Ao iniciar tracker para o Linux, Não existe imagem pronta."
    return 2
fi

}

#Cria imagem do Linux.
cria_linux() {
#Essa função remove a imagem antiga do linux e seu torrent, monta a partição ao
qual o linux está instalado em /mnt
#Logo depois de montado em /mnt ele simplesmente compacta todos os arquivos do
Linux.

log "Criando imagem do Linux."
echo "Removendo imagem antiga do linux..."
rm -f ${img_lin} #remove a antiga imagem do linux.
rm -f ${tor_lin} #remove o torrent do linux.

#Cria arquivo .tar do linux
umount /mnt 2> /dev/null #desmonta a partição.
mount ${partlinux} /mnt #monta a partição do linux /dev/sda2 em /mnt.
cd /mnt #entra no diretório mnt ao qual foi montada a partição do linux.
tar -cvf ${img_lin} * #compacta a imagem do linux.

#Atualiza Sistema de arquivo da partição no riso.service
sa_partlinux=`blkid ${partlinux} | cut -d'"' -f4`
sed "s/sa_partlinux=[^<]*/sa_partlinux=${sa_partlinux}/" < ${riso_service} >
${riso_service}.tmp
mv ${riso_service}.tmp ${riso_service}

umount /mnt 2> /dev/null #desmonta a partição que estava montada em mnt e
mostra a mensagem de imagem criada.
log "Imagem do Linux criada."
}

#Cria arquivo .torrent para imagem do linux
cria_torrent_linux() {
#Essa função

if [ -e $img_lin ]; then #verifica se a imagem do linux é existente.
    log "Criando arquivo torrent do linux."
    #Cria o torrent
    btmakemetafile.bittorrent ${img_lin} http://${servidor}:6969/announce 2>
/dev/null
```

```

        #Escreve o tamanho da imagem no riso.service
        tam=`du ${img_lin} | cut -f1` #pega o tamanho total da imagem.
        sed "s/tamlinux=[0-9]*/tamlinux=${tam}/" < ${riso_service} >
        ${riso_service}.tmp
        mv ${riso_service}.tmp ${riso_service}
        log "Arquivo torrent do Linux criado."
    else
        dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\nVocê ainda não criou a imagem do Linux.' \
        7 44
        log "ERRO: Ao criar arquivo torrent do Linux, não existe imagem pronta."
        return 2
    fi
}

#Cria imagem do Windows.
cria_windows() {
#Essa função desmonta a partição do windows, em seguida cria um clone dessa imagem
o chamando de windows.img
#Ao qual o windows.img é salvo no diretório de imagens do risos /usr/riso/imagens

    log "Criando imagem do Windows."
    echo "Removendo imagem antiga do windows..."
    rm -f ${img_win} #remove a antiga imagem do windows.
    rm -f ${tor_win} #remove o torrent do windows.

    #Cria arquivo .img do windows
    umount ${partwindows} 2> /dev/null #desmonta a partição do windows.
    ntfsclone -s -O ${img_win} ${partwindows} #usa o programa do ntfsclone para
    criar arquivo windows.img.

    #Se não conseguiu criar a imagem
    if [ "$?" -ne "0" ]; then
        dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox 'Não consegui criar uma imagem do Windows.\n
        O sistema operacional pode ainda não ter sido instalado.\n
        A partição ou o sistema de arquivo indicados pelas variáveis\n
        partwindows e sa_partwindows em /etc/avahi/services/riso.service\n
        pode estar incorreta.\n' \
        0 0
        log "ERRO: Ao criar imagem do Windows, ntfsclone falhou ao criar arquivo
        windows.img."
        return 3
    fi
    #Atualiza Sistema de arquivo da partição no riso.service
    sa_partwindows=`blkid ${partwindows} | cut -d'"' -f4`
    sed "s/sa_partwindows=[^<]*/sa_partwindows=${sa_partwindows}/" <
    ${riso_service} > ${riso_service}.tmp
    mv ${riso_service}.tmp ${riso_service}
    log "Imagem do Windows criada."
}

#Cria arquivo .torrent para imagem do linux.

```

```

cria_torrent_windows() {

    #servidor=$( dialog --stdout --inputbox 'Digite o IP do seu Servidor:' 0 0 )
    #Alteração

    if [ -e ${img_win} ]; then #verifica se a imagem do windows é existente
img_win.
        log "Criando arquivo torrent do windows."
        btmakemetafile.bittorrent ${img_win} http://${servidor}:6969/announce 2>
/dev/null

        #Escreve o tamanho da imagem no riso.service
        tam=`du ${img_win} | cut -f1` #pega o tamanho total da imagem do windows.
        sed "s/tamwindows=[0-9]*/tamwindows=${tam}/" < ${riso_service} >
${riso_service}.tmp
        mv ${riso_service}.tmp ${riso_service}
        log "Arquivo torrent do windows criado."
    else
        dgrubialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\nVocê ainda não criou a imagem do windows.' \
        7 45
        log "ERRO: Ao criar torrent do windows, não existe imagem pronta."
        return 2
    fi
}

#Cria imagem do SO com R.I.S.O. instalado.
cria_rec() {
#Para se criar a recuperação é necessário que se remova a imagem do windows e a do
linux, as mesmas são removidas com seus torrents.

    log "Criando imagem de recuperação."
    echo "Removendo imagem antiga do windows..."
    rm -f ${img_win} #remove antiga imagem do windows.
    rm -f ${tor_win} #remove torrent do windows.
    echo "Removendo imagem antiga do Linux..."
    rm -f ${img_lin} #remove antiga imagem do linux.
    rm -f ${tor_lin} #remove o torrent da imagem do linux.
    echo "Removendo imagem antiga do Linux de recuperação..."
    rm -f ${img_rec} #remove antiga imagem de recuperação.
    sfdisk -d /dev/sda > ${tab_part}
    umount -a 2> /dev/null #desmonta a partição.
    cd /
    tar --exclude=home/remastersys --exclude=proc/* --exclude=risos --
exclude=riso.service -cvf ${img_rec} *

    #Atualiza Sistema de arquivo da partição no riso.service
    sa_partrec=`df -T | awk 'NR==2{print $2}'`
    sed "s/sa_partrec=[^<]*/sa_partrec=${sa_partrec}/" < ${riso_service} >
${riso_service}.tmp
    mv ${riso_service}.tmp ${riso_service}
    log "Imagem de recuperação criada."
}

#Atualiza riso.
atualiza() {

    log "Iniciando atualização do script."

```

```

log "Verificando disponibilidade do servidor de atualização."
ping -q -c 1 200.131.37.236 > /dev/null 2>&1
if [ "$?" -eq "0" ]; then
    log "Iniciando download do script novo."
    (wget 200.131.37.236/riso/risos0.5 -O /usr/riso/risos && log "RISO
atualizado com sucesso.") || log "ERRO: Não foi possível baixar o novo script."

    #Mensagem de atualizado com sucesso.
    dialog \
        --ok-label 'OK' \
        --title 'Atenção' \
        --msgbox '\n  RISOS atualizado com sucesso.\n' \
        7 39

    bash /usr/riso/risos
    exit
else
    dialog \
        --ok-label 'OK' \
        --title 'ERRO' \
        --msgbox '\n  Esta versão já é a mais recente.' \
        7 40
    log "ERRO: RISO já é a versão mais nova."
fi
}

#Pessoas que desenvolveram esse material.
creditos() {

    dialog \
        --ok-label 'OK' \
        --title 'Creditos' \
        --msgbox '
CENTRO FEDERAL DE EDUCACAO TECNOLOGICA \n
Engenharia da Computacao \n
Cristiano Goulart Lopes Dias \n
Vinicius Tinti de Paula Oliveira \n
Germano Teixeira de Miranda \n
Gabriel de Souza Brandao \n
Marcio J. Menezes Jr. \n
Gabriel Machado de Castro Fonseca \n
André Luiz Silveira Herculano \n
Gilmar Pereira de Alcantara \n
www.dgo.cefetmg.br \n
'
        0 0
}

#Notificação para imagem criada com sucesso
linux_criado() {

    dialog \
        --ok-label 'OK' \
        --title 'Atenção' \
        --msgbox '\nImagem do Linux criada com sucesso.\n' \
        7 40
}

```



```

#Notificação para imagem criada com sucesso
windows_criado() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\n Imagem do Windows criada com sucesso.\n' \
        7 43
}

#Notificação para imagem criada com sucesso
rec_criado() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\n Imagem de Recuperação criada com sucesso.\n' \
        7 47
}

#Notificação para imagens criadas com sucesso
imagens_criadas() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\n  Imagens criadas com sucesso.\n' \
        7 37
}

#Notificação para torrent criado com sucesso
torrent_linux_criado() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\n  Torrent Linux criado com sucesso.\n' \
        7 42
}

#Notificação para torrent criado com sucesso
torrent_windows_criado() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\n  Torrent Windows criado com sucesso.\n' \
        7 43
}

#Notificação para torrent criado com sucesso
torrents_criados() {

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox '\nTorrents do Windows e Linux criados com sucesso.\n' \
        8 0
}

#Acredite isso é um help.

```

```

helpi() {

    dialog \
    --ok-label 'OK' \
    --title 'Help' \
    --msgbox 'Envie um e-mail com suas duvidas e sugestões \n
para: "germano@comp.eng.br" e responderei o\n
mais rápido possível.' \
    0 0

}

#Cria um livecd do SO com o R.I.S.O instalado.
#Essa função foi criada com base no código do remastersys.
livecd() {

    log "Criando live-cd do riso."

    #Modifica rc.local
    cat /etc/rc.local | grep -v exit > /etc/rc.local.tmp
    echo "rm /etc/avahi/services/riso.service" >> /etc/rc.local.tmp
    echo "rm /usr/riso/risos" >> /etc/rc.local.tmp
    echo "rm /usr/bin/risos" >> /etc/rc.local.tmp
    echo "exit 0" >> /etc/rc.local.tmp
    mv /etc/rc.local.tmp /etc/rc.local
    chmod +x /etc/rc.local

    # load the remastersys.conf file
    . /etc/remastersys.conf

    cdfs $@
    iso $@

    #Restaura rc.local
    cat /etc/rc.local | grep -v riso > /etc/rc.local.tmp
    mv /etc/rc.local.tmp /etc/rc.local
    chmod +x /etc/rc.local

    dialog \
    --ok-label 'OK' \
    --title 'Atenção' \
    --msgbox 'Arquivo de imagem criado com sucesso em:\n
/home/remastersys/riso0.5.iso.\n\n
Para garava-lo em um CD/DVD tente:\n
cdrecord -v -fs=16M speed=32 dev=1,0,0 -data /home/remastersys/riso0.5.iso\n\n
Identifique seu dev com: cdrecord -scanbus' \
    0 0
    log "Live-cd do riso criado."
}

#Função usada para criar o livecd do riso.
#Essa função foi criada com base no código do remastersys.
cdfs () {

    #removing popularity-contest as it causes a problem when installing with ubiquity
    apt-get -y -q remove popularity-contest

    # check whether system is gnome or kde based to load the correct ubiquity frontend
    if [ "`cat /etc/X11/default-display-manager`" = "/usr/bin/kdm" ]; then
    apt-get -y -q install ubiquity-frontend-kde
    apt-get -y -q remove ubiquity-frontend-gtk
    else

```

```

apt-get -y -q install ubiquity-frontend-gtk
apt-get -y -q remove ubiquity-frontend-kde
apt-get -y -q install metacity
fi

# prevent the installer from changing the apt sources.list
if [ ! -f "/usr/share/ubiquity/apt-setup.saved" ]; then
cp /usr/share/ubiquity/apt-setup /usr/share/ubiquity/apt-setup.saved
fi

sleep 1

# Step 3 - Create the CD tree in $WORKDIR/ISOTMP
echo "Checking if the $WORKDIR folder has been created"
if [ -d "$WORKDIR" ]; then
rm -rf $WORKDIR/dummysys/var/*
rm -rf $WORKDIR/dummysys/etc/*
rm -rf $WORKDIR/ISOTMP/isolinux
rm -rf $WORKDIR/ISOTMP/grub
rm -rf $WORKDIR/ISOTMP/.disk

else
mkdir -p $WORKDIR/ISOTMP/casper
mkdir -p $WORKDIR/ISOTMP/preseed
mkdir -p $WORKDIR/dummysys/dev
mkdir -p $WORKDIR/dummysys/etc
mkdir -p $WORKDIR/dummysys/proc
mkdir -p $WORKDIR/dummysys/tmp
mkdir -p $WORKDIR/dummysys/sys
mkdir -p $WORKDIR/dummysys/mnt
mkdir -p $WORKDIR/dummysys/media/cdrom
mkdir -p $WORKDIR/dummysys/var
chmod ug+rxw,o+rwt $WORKDIR/dummysys/tmp

fi

if [ "$CDBOOTTYPE" = "ISOLINUX" ]; then
mkdir -p $WORKDIR/ISOTMP/isolinux
else
mkdir -p $WORKDIR/ISOTMP/boot/grub
fi

mkdir -p $WORKDIR/ISOTMP/.disk

echo "Copying /var and /etc to temp area and excluding extra files"

if [ "$EXCLUDES" != "" ]; then
for addvar in $EXCLUDES ; do
VAREXCLUDES="$VAREXCLUDES --exclude='$addvar' "
done
fi

rsync --exclude='*.log.*' --exclude='*.pid' --exclude='*.bak' --exclude='*.[0-9].gz' --exclude='*.deb' $VAREXCLUDES -a /var/. $WORKDIR/dummysys/var/.
rsync $VAREXCLUDES -a /etc/. $WORKDIR/dummysys/etc/.

rm -rf $WORKDIR/dummysys/etc/X11/xorg.conf*
rm -rf $WORKDIR/dummysys/etc/resolv.conf
rm -rf $WORKDIR/dummysys/etc/hosts
rm -rf $WORKDIR/dummysys/etc/hostname

```

```

rm -rf $WORKDIR/dummysys/etc/timezone
rm -rf $WORKDIR/dummysys/etc/mtab
rm -rf $WORKDIR/dummysys/etc/fstab
rm -rf $WORKDIR/dummysys/etc/udev/rules.d/70-persistent*
rm -rf $WORKDIR/dummysys/etc/cups/ssl/server.crt
rm -rf $WORKDIR/dummysys/etc/cups/ssl/server.key
rm -rf $WORKDIR/dummysys/etc/ssh/ssh_host_rsa_key
rm -rf $WORKDIR/dummysys/etc/ssh/ssh_host_dsa_key.pub
rm -rf $WORKDIR/dummysys/etc/ssh/ssh_host_dsa_key
rm -rf $WORKDIR/dummysys/etc/ssh/ssh_host_rsa_key.pub
ls $WORKDIR/dummysys/var/lib/apt/lists | grep -v ".gpg" | grep -v "lock" | grep -v
"partial" | xargs -i rm $WORKDIR/dummysys/var/lib/apt/lists/{} ;

if [ "$1" = "dist" ]; then

echo " " > $WORKDIR/dummysys/etc/gdm/gdm.conf-custom
rm -rf $WORKDIR/dummysys/etc/group
rm -rf $WORKDIR/dummysys/etc/passwd
rm -rf $WORKDIR/dummysys/etc/shadow
rm -rf $WORKDIR/dummysys/etc/shadow-
rm -rf $WORKDIR/dummysys/etc/gshadow
rm -rf $WORKDIR/dummysys/etc/gshadow-

find $WORKDIR/dummysys/var/run $WORKDIR/dummysys/var/log
$WORKDIR/dummysys/var/mail $WORKDIR/dummysys/var/spool $WORKDIR/dummysys/var/lock
$WORKDIR/dummysys/var/backups $WORKDIR/dummysys/var/tmp
$WORKDIR/dummysys/var/crash -type f -exec rm {} \;

for i in dpkg.log lastlog mail.log syslog auth.log daemon.log faillog lpr.log
mail.warn user.log boot debug mail.err messages wtmp bootstrap.log dmesg kern.log
mail.info
do
    touch $WORKDIR/dummysys/var/log/${i}
done

# remove the non system users on the passwd and group files and copy them to
dummysys

# see if any temp users left over from previous versions of remastersys as the
process has now changed

grep '^[^:]*:[^:]*:[0-9][0-9][0-9]:' /etc/passwd | awk -F ":" '{print
"/usr/sbin/userdel -f",$1}'> $WORKDIR/cleantmpusers
. $WORKDIR/cleantmpusers

grep '^[^:]*:[^:]*:[0-9]:' /etc/passwd >> $WORKDIR/dummysys/etc/passwd
grep '^[^:]*:[^:]*:[0-9][0-9]:' /etc/passwd >> $WORKDIR/dummysys/etc/passwd
grep '^[^:]*:[^:]*:[0-9][0-9][0-9]:' /etc/passwd >> $WORKDIR/dummysys/etc/passwd
grep '^[^:]*:[^:]*:[3-9][0-9][0-9][0-9][0-9]:' /etc/passwd >>
$WORKDIR/dummysys/etc/passwd

grep '^[^:]*:[^:]*:[0-9]:' /etc/group >> $WORKDIR/dummysys/etc/group
grep '^[^:]*:[^:]*:[0-9][0-9]:' /etc/group >> $WORKDIR/dummysys/etc/group
grep '^[^:]*:[^:]*:[0-9][0-9][0-9]:' /etc/group >> $WORKDIR/dummysys/etc/group
grep '^[^:]*:[^:]*:[3-9][0-9][0-9][0-9][0-9]:' /etc/group >>
$WORKDIR/dummysys/etc/group

grep '^[^:]*:[^:]*:[5-9][0-9][0-9]:' /etc/passwd | awk -F ":" '{print $1}'>
$WORKDIR/tmpusers1

```

```

grep '^[^:]*:[^:]*:[1-9][0-9][0-9][0-9]:' /etc/passwd | awk -F ":" '{print $1}'>
$WORKDIR/tmpusers2
grep '^[^:]*:[^:]*:[1-2][0-9][0-9][0-9][0-9]:' /etc/passwd | awk -F ":" '{print
$1}'> $WORKDIR/tmpusers3

cat $WORKDIR/tmpusers1 $WORKDIR/tmpusers2 $WORKDIR/tmpusers3 > $WORKDIR/tmpusers

cat $WORKDIR/tmpusers | while read LINE ;do

echo $LINE | xargs -i sed -e 's/{,}//g' $WORKDIR/dummysys/etc/group >
$WORKDIR/dummysys/etc/group.new1
echo $LINE | xargs -i sed -e 's/{,}//g' $WORKDIR/dummysys/etc/group.new1 >
$WORKDIR/dummysys/etc/group.new2
echo $LINE | xargs -i sed -e 's/{,}//g' $WORKDIR/dummysys/etc/group.new2 >
$WORKDIR/dummysys/etc/group

rm -rf $WORKDIR/dummysys/etc/group.new1 $WORKDIR/dummysys/etc/group.new2

done

fi

# make sure the adduser and autologin functions of casper as set according to the
mode

[ "$1" = "dist" ] && [ ! -d $WORKDIR/dummysys/home ] && mkdir
$WORKDIR/dummysys/home
[ "$1" = "dist" ] && chmod 755 /usr/share/initramfs-tools/scripts/casper-
bottom/*adduser /usr/share/initramfs-tools/scripts/casper-bottom/*autologin
[ "$1" = "backup" ] && [ -d $WORKDIR/dummysys/home ] && rm -rf
$WORKDIR/dummysys/home
[ "$1" = "backup" ] && chmod 644 /usr/share/initramfs-tools/scripts/casper-
bottom/*adduser /usr/share/initramfs-tools/scripts/casper-bottom/*autologin

# copy over some of the necessary stuff for the livecd

if [ -f /etc/remastersys/preseed/custom.seed ]; then
cp /etc/remastersys/preseed/custom.seed $WORKDIR/ISOTMP/preseed/custom.seed
fi

if [ "$CDBOOTTYPE" = "ISOLINUX" ]; then

#BOOT Type is isolinux

cp /boot/memtest86+.bin $WORKDIR/ISOTMP/isolinux/memtest

# check and see if they have a custom isolinux already setup. eg. they copied over
the isolinux folder from their original livecd or made a custom one for their
distro

if [ ! -f /etc/remastersys/customisolinux/isolinux.cfg ]; then

find /usr -name 'isolinux.bin' -exec cp {} $WORKDIR/ISOTMP/isolinux/ \;

# setup isolinux for the livecd

#check if it is 8.04 or higher and if it is, add the direct install option to
isolinux

VERSION=`lsb_release -r | awk '{print $2}' | awk -F "." '{print $1}'`

```

```

sed -e 's/ __LIVECDLABEL__ /'"$LIVECDLABEL"'/g'
/etc/remastersys/isolinux/isolinux.txt > $WORKDIR/ISOTMP/isolinux/isolinux.txt
cp /etc/remastersys/isolinux/isolinux.cfg $WORKDIR/ISOTMP/isolinux/isolinux.cfg

else

cp /etc/remastersys/customisolinux/* $WORKDIR/ISOTMP/isolinux/

fi

else

# Boot Type selected is GRUB

apt-get install -y grub

cp /boot/memtest86+.bin $WORKDIR/ISOTMP/boot/memtest
find /boot -name 'stage2_eltorito' -exec cp {} $WORKDIR/ISOTMP/boot/grub/ \;
find /usr -name 'stage2_eltorito' -exec cp {} $WORKDIR/ISOTMP/boot/grub/ \;
find /lib -name 'stage2_eltorito' -exec cp {} $WORKDIR/ISOTMP/boot/grub/ \;

mv /etc/default/grub /etc/default/grub.tmp
apt-get install -y grub2
mv /etc/default/grub.tmp /etc/default/grub

cp /etc/remastersys/grub/splash.xpm.gz $WORKDIR/ISOTMP/boot/splash.xpm.gz

# setup grub for the livecd

#check if it is 8.04 or higher and if it is, add the direct install option to grub

sed -e 's/ __LIVECDLABEL__ /'"$LIVECDLABEL"'/g' /etc/remastersys/grub/menu.lst >
$WORKDIR/ISOTMP/boot/grub/menu.lst

fi

ARCH=`archdetect | awk -F "/" '{print $1}'`

cat > $WORKDIR/ISOTMP/README.diskdefines <<FOO
#define DISKNAME $LIVECDLABEL
#define TYPE binary
#define TYPEbinary 1
#define ARCH $ARCH
#define ARCH$ARCH 1
#define DISKNUM 1
#define DISKNUM1 1
#define TOTALNUM 0
#define TOTALNUM0 1
FOO
cp $WORKDIR/ISOTMP/README.diskdefines $WORKDIR/ISOTMP/casper/README.diskdefines

sleep 1

# Step 4 - Make the filesystem.manifest and filesystem.manifest-desktop
echo "Creating filesystem.manifest and filesystem.manifest-desktop"

dpkg-query -W --showformat='${Package} ${Version}\n' \

```

```

> $WORKDIR/ISOTMP/casper/filesystem.manifest

cp $WORKDIR/ISOTMP/casper/filesystem.manifest
$WORKDIR/ISOTMP/casper/filesystem.manifest-desktop

sleep 1

# Step 5 - Prepare casper.conf depending on whether this is a backup or dist

if [ "$1" = "backup" ]; then
LIVEUSER=`grep '^[^:]*:[^:]*:1000:' /etc/passwd | awk -F ":" '{ print $1 }'`
fi

echo "# This file should go in /etc/casper.conf" > /etc/casper.conf
echo "# Supported variables are:" >> /etc/casper.conf
echo "# USERNAME, USERFULLNAME, HOST, BUILD_SYSTEM" >> /etc/casper.conf
echo " " >> /etc/casper.conf
echo "export USERNAME=\"$LIVEUSER\"" >> /etc/casper.conf
echo "export USERFULLNAME=\"Live session user\"" >> /etc/casper.conf
echo "export HOST=\"$LIVEUSER\"" >> /etc/casper.conf
echo "export BUILD_SYSTEM=\"Ubuntu\"" >> /etc/casper.conf

cp /etc/casper.conf $WORKDIR/dummysys/etc/

sleep 1

# if the mode is dist then renumber the uid's for any user with a uid greater than
1000
# and make the passwdrestore file so the uid's are restored before the script
finishes
# if this is not done, the livecd user will not be created properly

if [ "$1" = "dist" ]; then

# make sure user-setup-apply is present in case backup mode was last used

if [ -f /usr/lib/ubiquity/user-setup/user-setup-apply.orig ]; then
cp /usr/lib/ubiquity/user-setup/user-setup-apply.orig /usr/lib/ubiquity/user-
setup/user-setup-apply
fi

else

# since this is backup mode, prevent user-setup-apply from running during install
if [ ! -f /usr/lib/ubiquity/user-setup/user-setup-apply.orig ]; then
mv /usr/lib/ubiquity/user-setup/user-setup-apply /usr/lib/ubiquity/user-
setup/user-setup-apply.orig
fi
echo "exit 0"> /usr/lib/ubiquity/user-setup/user-setup-apply
chmod 755 /usr/lib/ubiquity/user-setup/user-setup-apply

# copy the install icon to the sudo users desktop

cp `find /usr -name ubiquity*.desktop` /home/`grep '^[^:]*:[^:]*:1000:'
/etc/passwd | awk -F ":" '{ print $1 }'`/Desktop

fi

```

```

sleep 1

echo "Setting up casper and ubiquity options for $1 mode"

rm -f /usr/share/ubiquity/apt-setup
echo "#do nothing" > /usr/share/ubiquity/apt-setup
chmod 755 /usr/share/ubiquity/apt-setup

# make a new initial ramdisk including the casper scripts
mkinitramfs -o /boot/initrd.img-`uname -r` `uname -r`

echo "Copying your kernel and initrd for the livecd"
cp /boot/vmlinuz-`uname -r` $WORKDIR/ISOTMP/casper/vmlinuz
cp /boot/initrd.img-`uname -r` $WORKDIR/ISOTMP/casper/initrd.gz

# Step 6 - Make filesystem.squashfs

if [ -f $WORKDIR/remastersys.log ]; then
rm -f $WORKDIR/remastersys.log
touch $WORKDIR/remastersys.log
fi

if [ -f $WORKDIR/ISOTMP/casper/filesystem.squashfs ]; then
rm -f $WORKDIR/ISOTMP/casper/filesystem.squashfs
fi

echo "Creating filesystem.squashfs ... this will take a while so be patient"

#check if it is 8.04 or higher and if it is, add the -no-recovery option so it
doesn't include the recovery file and provide higher compression mode

SQUASHFSOPTSHIGH="-no-recovery -always-use-fragments"

echo "Adding stage 1 files/folders that the livecd requires."

# add the blank folders and trimmed down /var to the cd filesystem

mksquashfs $WORKDIR/dummysys/ $WORKDIR/ISOTMP/casper/filesystem.squashfs -no-
duplicates $SQUASHFSOPTSHIGH 2>>$WORKDIR/remastersys.log

sleep 1

echo "Adding stage 2 files/folders that the livecd requires."

# add the rest of the system depending on the mode selected

if [ "$1" = "backup" ]; then
mksquashfs / $WORKDIR/ISOTMP/casper/filesystem.squashfs -no-duplicates
$SQUASHFSOPTSHIGH -e \
.thumbnails \
.cache \
.bash_history \
Cache \
boot/grub \
dev \
etc \
media \
mnt \

```



```

proc \
sys \
tmp \
var \
$WORKDIR $EXCLUDES 2>>$WORKDIR/remastersys.log

else
mksquashfs / $WORKDIR/ISOTMP/casper/filesystem.squashfs -no-duplicates
$SQUASHFSOPTSHIGH -e \
.thumbnails \
.cache \
.bash_history \
Cache \
boot/grub \
dev \
etc \
home \
media \
mnt \
proc \
sys \
tmp \
var \
$WORKDIR $EXCLUDES 2>>$WORKDIR/remastersys.log

fi

sleep 1

#add some stuff the log in case of problems so I can troubleshoot it easier
echo "-----"
>>$WORKDIR/remastersys.log
echo "Mount information" >>$WORKDIR/remastersys.log
mount >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "Casper Script info" >>$WORKDIR/remastersys.log
ls -l /usr/share/initramfs-tools/scripts/casper-bottom/ >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "/etc/remastersys.conf info" >>$WORKDIR/remastersys.log
cat /etc/remastersys.conf >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "/etc/casper.conf info" >>$WORKDIR/remastersys.log
cat /etc/casper.conf >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "/etc/passwd info" >>$WORKDIR/remastersys.log
cat $WORKDIR/dummysys/etc/passwd >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "/etc/group info" >>$WORKDIR/remastersys.log
cat $WORKDIR/dummysys/etc/group >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log
echo "Command-line options = $@" >>$WORKDIR/remastersys.log
echo "-----"
>>$WORKDIR/remastersys.log

```

```

# cleanup the install icons as they aren't needed on the current system

if [ "$1" = "backup" ]; then
rm -rf /home/\`grep '^[^:]*:[^:]*:1000:' /etc/passwd | awk -F ":" '{ print $1
}'`/Desktop/ubiquity*.desktop
fi

sleep 1

#checking the size of the compressed filesystem to ensure it meets the iso9660
spec for a single file
SQUASHFSSIZE=`ls -s $WORKDIR/ISOTMP/casper/filesystem.squashfs | awk -F " "
'{print $1}'`
if [ "$SQUASHFSSIZE" -gt "3999999" ]; then
echo " The compressed filesystem is larger than the iso9660 specification allows
for a single file. You must try to reduce the amount of data you are backing up
and try again."
echo " The compressed filesystem is larger than the iso9660 specification allows
for a single file. You must try to reduce the amount of data you are backing up
and try again.">>$WORKDIR/remastersys.log

exit 1
fi

}

#Gera arquivo iso do livecd do riso.
#Essa função foi criada com base no código do remastersys.
iso (){

    CREATEISO=""`which mkisofs`"
    if [ "$CREATEISO" = "" ]; then
    CREATEISO=""`which genisoimage`"
    fi

    # check to see if the cd filesystem exists

    if [ ! -f "$WORKDIR/ISOTMP/casper/filesystem.squashfs" ]; then
    echo "The cd filesystem is missing. Either there was a problem creating the
compressed filesystem or you are trying to run sudo remastersys dist iso before
sudo remastersys dist cdfs"
    echo "The cd filesystem is missing. Either there was a problem creating the
compressed filesystem or you are trying to run sudo remastersys dist iso before
sudo remastersys dist cdfs" >>$WORKDIR/remastersys.log

    exit 1
    fi

    #checking the size of the compressed filesystem to ensure it meets the iso9660
spec for a single file
    SQUASHFSSIZE=`ls -s $WORKDIR/ISOTMP/casper/filesystem.squashfs | awk -F " "
'{print $1}'`
    if [ "$SQUASHFSSIZE" -gt "3999999" ]; then
    echo " The compressed filesystem is larger than the iso9660 specification
allows for a single file. You must try to reduce the amount of data you are
backing up and try again."

```

```
echo " The compressed filesystem is larger than the iso9660 specification
allows for a single file. You must try to reduce the amount of data you are
backing up and try again.">>$WORKDIR/remastersys.log
```

```
exit 1
fi
```

```
# Step 7 - Make md5sum.txt for the files on the livecd - this is used during
the
```

```
# checking function of the livecd
```

```
echo "Creating md5sum.txt for the livecd/dvd"
```

```
cd $WORKDIR/ISOTMP && find . -type f -print0 | xargs -0 md5sum > md5sum.txt
```

```
if [ "$CDBOOTTYPE" = "ISOLINUX" ]; then
```

```
#isolinux mode
```

```
# remove files that change and cause problems with checking the disk
```

```
sed -e '/isolinux/d' md5sum.txt > md5sum.txt.new
```

```
sed -e '/md5sum/d' md5sum.txt.new > md5sum.txt
```

```
rm -f md5sum.txt.new
```

```
sleep 1
```

```
# Step 8 - Make the ISO file
```

```
echo "Creating $CUSTOMISO in $WORKDIR"
```

```
$CREATEISO \
```

```
-quiet \
```

```
-r \
```

```
-V "$LIVECDLABEL" \
```

```
-cache-inodes \
```

```
-J \
```

```
-l \
```

```
-b isolinux/isolinux.bin \
```

```
-c isolinux/boot.cat \
```

```
-no-emul-boot \
```

```
-boot-load-size 4 \
```

```
-boot-info-table \
```

```
-o $WORKDIR/$CUSTOMISO "$WORKDIR/ISOTMP" 2>>$WORKDIR/remastersys.log
```

```
1>>$WORKDIR/remastersys.log
```

```
else
```

```
#grub mode
```

```
# remove files that change and cause problems with checking the disk
```

```
sed -e '/boot/d' md5sum.txt > md5sum.txt.new
```

```
sed -e '/md5sum/d' md5sum.txt.new > md5sum.txt
```

```
rm -f md5sum.txt.new
```

```
sleep 1
```

```
# Step 8 - Make the ISO file
```

```
echo "Creating $CUSTOMISO in $WORKDIR"
```

```
$CREATEISO \
```

```
-quiet \
```

```
-r \
```

```
-V "$LIVECDLABEL" \
```

```
-cache-inodes \
```

```
-J \
```

```
-l \
```

```

        -b boot/grub/stage2_eltorito      \
        -no-emul-boot                    \
        -boot-load-size 4                \
        -boot-info-table                 \
        -o $WORKDIR/$CUSTOMISO "$WORKDIR/ISOTMP" 2>>$WORKDIR/remastersys.log
1>>$WORKDIR/remastersys.log

```

```
fi
```

create the md5 sum file so the user doesn't have to - this is good so the iso file can later be tested to ensure it hasn't become corrupted

```
echo "Creating $CUSTOMISO.md5 in $WORKDIR"
```

```
md5sum $WORKDIR/$CUSTOMISO > $WORKDIR/$CUSTOMISO.md5
```

```
sleep 1
```

```
echo "$WORKDIR/$CUSTOMISO is ready to be burned or tested in a virtual machine."
```

```
echo " "
```

```
echo "Check the size and if it is larger than 700MB you will need to burn it to a dvd"
```

```
echo " "
```

```
ls -hs $WORKDIR/$CUSTOMISO
```

```
echo " "
```

```
echo "It is recommended to run 'sudo remastersys clean' once you have burned and tested the $CUSTOMISO"
```

```
echo " "
```

```
}
```

#Checklist de seleção.

```
menu_inicia_servidor() {
```

```
    /etc/init.d/avahi-daemon restart #reinicia o serviço do avahi
```

```
    opcao=$( dialog --stdout \
```

```
    --ok-label 'Confirmar' \
```

```
    --checklist 'Quais imagens deseja disponibilizar para o download na rede:' \
```

```
    0 0 0 \
```

```
    Windows '' ON \
```

```
    Linux '' ON )
```

De acordo com a opção escolhida, executa funcoes

```
case $opcao in
```

```
    "Windows") inicia_servidor_windows;;
```

```
    "Linux") inicia_servidor_linux;;
```

```
    "Windows" "Linux") inicia_servidor;;
```

```
esac
```

```
}
```

#Checklist de seleção.

```
menu_criar_imagens() {
```

```
    opcao=$( dialog --stdout \
```

```
    --ok-label 'Confirmar' \
```

```
    --checklist 'Quais imagens deseja criar:' \
```

```
    0 0 0 \
```

```
    Windows '' ON \
```

```
    Linux '' ON \
```

```

Recuperacao '' off )

# De acordo com a opção escolhida, executa funcoes
case $opcao in
    "Windows") cria_windows && cria_torrent_windows && windows_criado;;
    "Linux") cria_linux && cria_torrent_linux && linux_criado;;
    "Recuperacao") cria_rec && rec_criado;;
    "Windows" "Linux") cria_windows && cria_torrent_windows && cria_linux &&
cria_torrent_linux && imagens_criadas;;
    "Windows" "Recuperacao") cria_rec && cria_windows &&
cria_torrent_windows && imagens_criadas;;
    "Linux" "Recuperacao") cria_rec && cria_linux && cria_torrent_linux &&
imagens_criadas;;
    "Windows" "Linux" "Recuperacao") cria_rec && cria_windows &&
cria_torrent_windows && cria_linux && cria_torrent_linux && imagens_criadas;;
esac

}

#Checklist de seleção.
menu_criar_torrents() {

    opcao=$( dialog --stdout \
        --ok-label 'Confirmar' \
        --checklist 'Quais torrents deseja criar:' \
        0 0 0 \
        Windows '' ON \
        Linux '' ON )

    # De acordo com a opção escolhida, executa funcoes
    case $opcao in
        "Windows") cria_torrent_windows && torrent_windows_criado;;
        "Linux") cria_torrent_linux && torrent_linux_criado;;
        "Windows" "Linux") cria_torrent_windows && cria_torrent_linux &&
torrents_criados;;
    esac

}

#Menu principal.
menu() {

    while : ; do
        opcao=$(
            dialog --stdout \
                --ok-label 'Confirmar' \
                --cancel-label 'Sair' \
                --title 'RISOS - 0.5' \
                --menu 'Escolha o que voce quer fazer:' \
                0 0 0 \
                1 'Iniciar servidor' \
                2 'Criar imagens' \
                3 'Criar torrents' \
                4 'Criar live-cd' \
                5 'Atualiza' \
                6 'Creditos' \
                7 'Help' )

        # ESC, sai do programa...
        [ $? -ne 0 ] && break
    done
}

```

```

# De acordo com a opção escolhida, executa função
case $opcao in
    1) menu_inicia_servidor;;
    2) menu_criar_imagens;;
    3) testaip && menu_criar_torrents;;
    4) livedd "backup";;
    5) atualiza;;
    6) credits;;
    7) helpi;;
esac

done

}

#Verifica se usuário é root antes de iniciar o risos.
USER=`id -u`
if [ $USER == '0' ]; then
    log "Iniciando RISOS -----"
    carrega_variaveis && menu
    log "Finalizando RISOS -----"
    clear
else
    echo 'Só o root pode fazer isso, jovenzinho!'
fi

```

Riso

```

#!/bin/bash

#Variáveis do script
dir_riso=/usr/riso
dir_img=$dir_riso/imagens #/usr/riso/imagens
img_win=$dir_img/windows.img #/usr/riso/imagens/windows.img
img_lin=$dir_img/linux.tar #/usr/riso/imagens/linux.tar
img_rec=$dir_img/rec.tar #/usr/riso/imagens/rec.tar
tor_win=$img_win.torrent #/usr/riso/imagens/windows.img.torrent
tor_lin=$img_lin.torrent #/usr/riso/imagens/linux.tar.torrent
tab_part=$dir_img/tab_part #/usr/riso/imagens/tab_part
riso_conf=$dir_riso/riso.conf #/usr/riso/riso.conf
riso_part=$dir_riso/riso.part #/usr/riso/riso.part
arq_log=$dir_riso/riso.log #/usr/riso/riso.log

#desabilita o UUID no Ubuntu
DisableUUID() {
    b=`cat /boot/grubgrub.cfg | grep =UUID=`
    uuidUbuntu=`(cut -d ' ' -f3 | cut -d '=' -f2-3)<<<${b}`
    partLinux=`os-prober | grep linux | cut -d ';' -f1`

    sed "s|${uuidUbuntu}|${partLinux}|g" -i /boot/grub/grub.cfg
}

#Função de Log
log() {
    [ -e $arq_log ] || touch $arq_log
    echo `date +%F %H:%M:%S` "RISO: $" >> $arq_log
}

```

```

#Carrega variaveis disponibilizadas pelo avahi para variáveis do sistema.
carrega_variaveis() {

    avahi_saida=`avahi-browse -rtcp _DECOM_RISO._tcp | grep '='`
    servidor=`(cut -d ';' -f10 | cut -d '"' -f20 | cut -d '=' -f2) <<<`
    ${avahi_saida}` # pega o ip do servidor.
    if [ -z $servidor ]; then
        dialog \
            --title 'Atencao' \
            --ok-label 'OK' \
            --msgbox 'Não existe nenhum servidor RISOS ativo na rede.\nVerifique a
conexão e tente novamente.' \
            0 0
        log "ERRO: Servidor não encontrado na rede."
        exit 1
    fi
    partwindows=`(cut -d ';' -f10 | cut -d '"' -f2 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #Partição com Windows /dev/sda1 consegui fazer funcionar com -f20.
    partlinux=`(cut -d ';' -f10 | cut -d '"' -f4 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #Partição com Linux /dev/sda2 também funcionando com -f20.
    partrec=`(cut -d ';' -f10 | cut -d '"' -f6 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #Partição de Recuperação /dev/sda3 também funcionando com -f20.
    partswap=`(cut -d ';' -f10 | cut -d '"' -f8 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #Partição de Swap '=' também funcionando com -f20.

    #Tamanho total das imagens criadas no servidor.
    tamwindows=`(cut -d ';' -f10 | cut -d '"' -f10 | cut -d '=' -f2) <<<`
    ${avahi_saida}` # $tamwindows=123456789 | é um tamanho padrão, funcionando com -
f20.
    tamlinux=`(cut -d ';' -f10 | cut -d '"' -f12 | cut -d '=' -f2) <<<`
    ${avahi_saida}` # $tamlinux=123456789 | é um tamanho padrão, funcionando com -f20.

    #Sistema de arquivo das partições.
    sa_partwindows=`(cut -d ';' -f10 | cut -d '"' -f14 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #sa_partwindows=ntfs, também funcionando com -f20.
    sa_partlinux=`(cut -d ';' -f10 | cut -d '"' -f16 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #sa_partlinux=ext4, também funcionando com -f20.
    sa_partrec=`(cut -d ';' -f10 | cut -d '"' -f18 | cut -d '=' -f2) <<<`
    ${avahi_saida}` #sa_partrec=ext3, também funcionando com f-20.

    #Atualiza arquivo de variaveis.
    echo 'partwindows=$partwindows > $riso_part #Redireciona (Escreve)
partwindows=/dev/sda1 para o $riso_part
    echo 'partlinux=$partlinux >> $riso_part #Redireciona (Escreve)
partlinux=/dev/sad2 para o $riso_part
    echo 'sa_partwindows=$sa_partwindows >> $riso_part #Redireciona (Escreve)
sa_partwindows=123546789 para o $riso_part
    echo 'sa_partlinux=$sa_partlinux >> $riso_part #Redireciona (Escreve)
sa_partlinux=123456789 para o $riso_part

}

#Instala imagens do Linux e do Windows.
instala() {
    log "Iniciando instalação do Windows e do Linux."
    if [ -e ${img_win} -a -e ${img_lin} ]; then #Verifica se o arquivo img_win e
img_lin são existentes, nesse caso a imagem do Windows e do Linux.

        #Carrega variáveis.

```

```

#Verifica se $partwindows, $sa_partwindows, $partlinux e $sa_partlinux são
nulos e se o dono do arquivo é o usuário atual.
[ -z $partwindows -o -z $sa_partwindows ] && [ -z $partlinux -o -z
$sa_partlinux ] && . $riso_part

#Desmonta partições.
umount $partwindows 2> /dev/null
umount $partlinux 2> /dev/null

#Clona windows.
log "Instalação do Windows iniciada."

(ntfsclone -r $img_win -O $partwindows && log "Fim da instalação do
Windows") || log "ERRO: Windows não foi instalado com sucesso."

#Clona linux.
mkfs.${sa_partlinux} $partlinux
mount $partlinux /mnt #Monta a partição do linux /dev/sda2 em /mnt
cd /mnt #Entra no diretório /mnt
log "Instalação do Linux iniciada."
rm -f boot/grub/grub.cfg #Remove o Antigo Grub.
(tar -xvf $img_lin && log "Fim da instalação do Linux.") || log "ERRO:
Linux não foi instalado com sucesso." #Extrai a imagem do Linux.
cd .. #sai do diretório.
umount /mnt 2> /dev/null #Desmonta a partição.

sleep 3
log "Instalação do Windows e do Linux finalizada."
update-grub #Atualiza o grub.

DisableUUID #A função é jogada aqui para não atrapalhar o grub logo após a
instalação das imagens.

#verifica se desliga computador
desligar=`grep "DESLIGAR" ${riso_conf} | cut -d=' ' -f2` #desligar busca no
$riso_conf o valor de NÃO, sendo assim desligar=NÃO.
if [ "$desligar" = "SIM" ]; then #Se $desligar for igual a sim ele desliga
o computador.
log "Desligando computador."
halt
fi

#verifica se reinicia computador
reiniciar=`grep "REINICIAR" ${riso_conf} | cut -d=' ' -f2` #reiniciar busca
no $riso_conf o valor de SIM, sendo assim reiniciar=SIM.
if [ "$reiniciar" = "SIM" ]; then #Se $reiniciar for igual a sim ele
reinicia o computador, nesse caso ele ira reiniciar.
log "Reiniciando computador."
reboot #Reinicia a maquina.
fi
else #Caso não exista imagens prontas.
dialog \
--title 'Atenção' \
--ok-label 'OK' \
--msgbox '\n Não existem imagens prontas.' \
7 40
log "ERRO: Não existem imagens prontas."
return 5
fi
}

```



```

#Instala imagem do Windows.
instala_windows() {
    log "Iniciando instalação do Windows."
    if [ -e ${img_win} ]; then #Verifica se a imagem do Windows é existente
$img_win.

        #le do arquivo de variaveis
        [ -z $partwindows -o -z $sa_partwindows ] && . $riso_part #-z verifica se
o $partwindows e $sa_partwindows são nulos e o -o verifica se o dono do
        #arquivo é o usuario atual.

        #Desmonta partições
        umount $partwindows 2> /dev/null #desmonta a partição do windows
/dev/sdal.

        #Clona windows
        log "Instalação do Windows iniciada."
        (ntfsclone -r $img_win -O $partwindows && log "Fim da instalação do
Windows") || log "ERRO: Windows não instalou com sucesso."

        log "Instalação do Windows finalizada."
        update-grub #atualiza o grub.

        DisableUUID #A função vai se encaixar em todo código, assim evitando que o
grub seja modificado da forma incorreta.

        #verifica se desliga computador
        desligar=`grep "DESLIGAR" ${riso_conf} | cut -d'=' -f2` #Busca no
$riso_conf o valor de NÃO, sendo assim desligar=NÃO.
        if [ "$desligar" = "SIM" ]; then #Se desligar for igual a SIM ele desliga
o computador.
            log "Desligando computador."
            halt
        fi

        #verifica se reinicia computador
        reiniciar=`grep "REINICIAR" ${riso_conf} | cut -d'=' -f2` #Busca o
$riso_conf o valor de SIM, sendo assim reiniciar=SIM.
        if [ "$reiniciar" = "SIM" ]; then #Se reiniciar for igual a SIM, reinicia
o computador.
            log "Reiniciando computador."
            reboot #Reiniciando a maquina.
        fi
    else #Caso não exista imagens prontas.
        dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\n Você ainda não baixou a imagem do windows.' \
        7 50
        log "ERRO: Imagem do Windows ainda não foi baixada."
        return 5
    fi
}

#Instala imagem do Linux.
instala_linux() {
    log "Iniciando instalação do Linux."
    if [ -e ${img_lin} ]; then #Verifica se o $img_lin é existente.

```

```

#Carrega variáveis
[ -z $partlinux -o -z $sa_partlinux ] && . $riso_part #-z verifica de
$partlinux e $sa_partlinux são nulos e o -o verifica se o dono do arquivo
#é o usuario atual.

#desmonta partições.
umount $partlinux 2> /dev/null #desmonta a partição do linux $partlinux
/dev/sda2.

#clona linux
mkfs.${sa_partlinux} $partlinux
mount $partlinux /mnt #monta /dev/sda2 em /mnt.
cd /mnt #entra no diretório mnt.
log "Instalação do Linux iniciada."
rm -f boot/grub/grub.cfg #remove o antigo grub.
(tar -xvf $img_lin && log "Fim da instalação do Linux.") || log "ERRO:
Linux instalou com sucesso." #extraí a imagem do linux $img_lin.
cd .. #sai do diretório.
umount /mnt 2> /dev/null #desmonta a partição.

sleep 3
log "Instalação do Linux finalizada."
update-grub #atualiza o grub.

#verifica se desliga computador
desligar=`grep "DESLIGAR" ${riso_conf} | cut -d=' ' -f2` #Busca no
$riso_conf o valor de NÃO, sendo assim desligar=NÃO.
if [ "$desligar" = "SIM" ]; then #Se desligar for igual a SIM ele desliga
o computador.
log "Desligando computador."
halt
fi

#verifica se reinicia computador
reiniciar=`grep "REINICIAR" ${riso_conf} | cut -d=' ' -f2` #Busca no
$riso_conf o valor de SIM, sendo assim reiniciar=SIM.
if [ "$reiniciar" = "SIM" ]; then #Se desligar for igual a SIM ele desliga
o computador.
log "Reiniciando computador."
reboot #reinicia o computador.
fi
else #Se não existir imagens prontas.
dialog \
--title 'Atenção' \
--ok-label 'OK' \
--msgbox '\n Você ainda não baixou a imagem do linux.' \
7 50
log "Imagem do Linux ainda não foi baixada."
return 5
fi
}

#Baixa imagens do Linux e do Windows.
baixa_imagens() {
log "Iniciando download do Linux e do Windows."
log "Conectando ao servidor."
echo "Conectando ao servidor..."
carrega_variaveis || return 1

#Vefifica se imagem do windows já é a mais recente

```

```

sizelocwin="0" #cria uma variável no valor de 0.
[ -e ${img_win} ] && sizelocwin=$(du ${img_win} | cut -f1 ) #verifica se
$img_win é existente e coloca o tamanho total dela na variável sizelocwin.
if [ "$sizelocwin" = "$tamwindows" ]; then #se o tamanho total da imagem do
windows $sizelocwin for igual a $tamwindows ele mostra mensagem...
#imagem do Windows no computador já é a mais recente.

dialog \
--title 'Atenção' \
--ok-label 'OK' \
--msgbox '\nA imagem do Windows no computador já é a mais recente.' \
7 50
log "ERRO: Imagem do Windows já é a mais recente."
return 4
fi

#Vefifica se imagem do linux já é a mais recente
sizeloclin="0" #cria uma variável com o valor de 0.
[ -e ${img_lin} ] && sizeloclin=$(du ${img_lin} | cut -f1) #verifica se
$img_win é existente e coloca o tamanho dela na variável sizeloclin.
if [ "$sizeloclin" = "$tamlinux" ]; then #se o tamanho total da imagem do
windows $sizeloclin for igual a $tamwindows ele mostra mensagem...
#imagem do Linux no computador já é a mais recente.

dialog \
--title 'Atenção' \
--ok-label 'OK' \
--msgbox '\n A imagem do Linux no computador já é a mais recente.' \
7 50
log "ERRO: Imagem do Linux já é a mais recente."
return 4
fi

#Deleta imagens antigas.
log "Iniciando remoção da imagem antiga do Windows."
echo "Removendo imagem antiga do Windows..."
rm -f $img_win #remove a imagem antiga do windows.
rm -f $stor_win #remove o torrent antigo do windows.
log "Fim da remoção da imagem antiga do Windows."
log "Inicio da remoção da imagem antiga do Linux."
echo "Removendo imagem antiga do Linux..."
rm -f $img_lin #remove a imagem antiga do linux.
rm -f $stor_lin #remove o torrent antigo do linux.
log "Fim da remoção da imagem antiga do Linux."

#Inicia download das imagens em terminal virtual.
log "Iniciando download dos torrent do Windows."
#baixa o torrent da imagem do windows via ssh.
(scp -o StrictHostKeyChecking=no root@${servidor}:${tor_win} ${tor_win} && log
"Torrent do Windows baixado com sucesso.") || log "ERRO: Torrent do Windows não
foi baixado."
log "Iniciando download do torrent do Linux."
#baixa o torrent da imagem do linux via ssh.
(scp -o StrictHostKeyChecking=no root@${servidor}:${tor_lin} ${tor_lin} && log
"Torrent do Linux baixado com sucesso.") || log "ERRO: Torrent do Linux não foi
baixado."
cd ${dir_img} #entra no diretório $dir_img que é
$dir_img=riso/usr/riso/imagens.
log "Iniciando download das imagens do Windows e do Linux."
screen -d -m rtorrent ${tor_win} ${tor_lin}

#Se usuário cancelar volta para menu.
trap "killall rtorrent; clear; exit" 2 15

```

```

echo "Download do windows finalizado."
sleep 3 # Loop que aguarda o torrent ser concluído.

reset

}

#Baixa imagem do Windows.
baixa_imagem_windows() {
    log "Iniciando download do Windows."
    echo "Conectando ao servidor..."
    carrega_variaveis || return 1 #chama a função carrega as variáveis para
    conectar ao servidor.

    #Verifica se imagem já é a mais recente
    sizelocwin="0" #cria uma variável com o valor 0
    [ -e ${img_win} ] && sizelocwin=$(du ${img_win} | cut -f1) #verifica se a
    imagem do windows é existente $img_win e coloca na variável sizelocwin.
    if [ "$sizelocwin" = "$tamwindows" ]; then #se o tamanho total da imagem do
    windows $sizelocwin for maior que $tamwindows mostra a mensagem...
        #a imagem do windows no computador já é a mais recente.
        dialog \ #caixa de dialogo com o usuário.
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\n A imagem no computador já é a mais recente.' \
        7 50
    log "ERRO: A imagem do Windows é a mais recente."
    return 4
fi

#Deleta imagens antigas.
log "Iniciando remoção da imagem antiga do Windows."
echo "Removendo imagem antiga do Windows..."
rm -f ${img_win} #remove a imagem antiga do windows.
rm -f ${tor_win} #remove o torrent antigo do windows.
log "Remoção da imagem antiga do Windows finalizada."

#Inicia download da imagen em terminal virtual.
log "Inicia o download do torrent do Windows."
#baixa o torrent da imagem do windows via ssh.
(scp -o StrictHostKeyChecking=no root@${servidor}:${tor_win} ${tor_win} && log
"Download do torrent do Windows finalizado.") || log "ERRO: O torrent do Windows
não foi baixado."
cd $dir_img #entra no diretório usr/riso/imagens.
log "Iniciando Download do Windows."
screen -d -m rtorrent $tor_win

#Se usuario cancelar volta para menu.
trap "killall rtorrent; clear; exit" 2 15

echo "Download do Windows finalizado."
sleep 3 #Loop que aguarda o torrent a ser concluído.
reset

}

#Baixa imagem linux.
baixa_imagem_linux() {

```

```

log "Iniciando download do Linux."
echo "Conectando ao servidor..."
carrega_variaveis || return 1 #retorna a função de carrega_variaveis.

#Verifica se imagem já é a mais recente
sizeloclin="0" #cria uma variável no valor de 0.
[ -e ${img_lin} ] && sizeloclin=$(du ${img_lin} | cut -f1) #verifica se a
imagem do linux é exestente e coloca o valor total dela em sizeloclin.
if [ "$sizeloclin" = "$tamlinux" ]; then #se o tamanho total da imagem do
linux $sizeloclin for igual a $tamlinux mostra a mensagen...
    #imagem do linux no computador já é a mais recente.
    dialog \ #caixa de dialogo com o usuário.
    --title 'Atenção' \
    --ok-label 'OK' \
    --msgbox '\n A imagem no computador já é a mais recente.' \
    7 50
log "ERRO: A imagem do Linux já é a mais recente."
return 4
fi

#Deleta imagens antigas.
log "Deletando imagem antiga do Linux."
echo "Removendo imagem antiga do Linux..."
rm -f ${img_lin} #remove a imagem antiga do linux.
rm -f ${tor_lin} #remove o torrent antigo do linux.
log "Imagem antiga do Linux deletada."

#Inicia download da imagem em terminal virtual.
log "Baixando torrent do Linux."
#baixa a imagem do linux via ssh.
(scp -o StrictHostKeyChecking=no root@${servidor}:${tor_lin} ${tor_lin} && log
"Torrent do Linux baixado com sucesso.") || log "ERRO: Erro ao baixar torrent do
Linux."
cd ${dir_img} #entra no diretório usr/riso/imagens.
log "Iniciando download do Linux."
screen -d -m rtorrent ${tor_lin}

#Se usuário cancelar volta para menu.
trap "killall rtorrent; clear; exit" 2 15

echo "Download do Windows finalizado."
sleep 3 # Loop que aguarda o torrent ser concluido.
reset
}

#Instala imagem do SO com o R.I.S.O..
instala_rec() {
log "Iniciando instalação da recuperação."
log "Instalando recuperação."
#Instala imagem
tar -xvf rec.tar #simplesmente extrai a imagem de recuperação com o comando
tar -xvf nome do arquivo, que no caso é o rec.tar.

#instala o grub
log "Instalando o GRUB."
chroot /mnt update-grub #atualiza o grub.
#instala o grub em /dev/sda para reconhecer todos os sistemas operacionais e
todas as partições.

```

```

    (chroot /mnt grub-install /dev/sda && log "GRUB instalado com sucesso.") ||
log "ERRO: GRUB não instalado com sucesso."

#Verifica se a instalação foi concluída com sucesso
if [ -e /mnt/vmlinuz ] #o comando -e verifica se o arquivo é existente.
then
    #mostra a mensagem para o usuário.
    clear
    log "Imagem instalada de recuperação instalada com sucesso."
    log "Reiniciando o computador."
    reboot #reinicia a máquina.
else
    #caso venha acontecer algo de errado, mostra a mensagem de erro para o
usuário.
    log "ERRO: Não foi possível instalar a imagem de recuperação."
    dialog \
        --title 'Atenção' \
        --ok-label 'OK' \
        --msgbox '\n          ...NOSSA!!!, o script falhou!!!...
...reinicie o computador e tente outra vez...' \
        8 50
fi
}

#Baixa imagem do SO com o R.I.S.O...
baixa_rec() {
    log "Iniciando download da imagem de recuperação."
    echo "Conectando ao servidor..."
    carrega_variaveis || return 1 #retorna a função que carrega as variaveis para
efetuar a conexão com servidor.

    #Baixa refaz tabela de partição
    log "Baixando tabela de partições."
    (scp -o StrictHostKeyChecking=no root@${servidor}:${tab_part} ${tab_part} &&
log "Tabela de partições baixada com sucesso") || log "ERRO: Não foi possível
baixar tabela de partições."
    swapoff -a
    log "Montando nova tabela de partições."
    #monta a tabela de partição.
    sfdisk -f /dev/sda < ${tab_part}

    #Desmonta partição
    umount ${partrec} 2> /dev/null #/dev/sda3

    log "Formatando partições e criando swap"
    #Formata partição
    mkfs.${sa_partrec} ${partrec} #ext3 | /dev/sda3.
    mkswap ${partswap} #${partswap} no avahi_saida é $partswap= .
    swapon ${partswap} #${partswap} no avahi_saida é $partswap= .

    #Baixa imagem .tar via scp
    umount /mnt 2> /dev/null #desmonta a partição.
    mount ${partrec} /mnt #monta a partição $partrec que nesse caso é /dev/sda3.
    cd /mnt #entra na partição montada em mnt.
    log "Iniciando o download da imagem de recuperação."
    #baixa a imagem da recuperação via ssh.
    (scp -o StrictHostKeyChecking=no root@${servidor}:${img_rec} ./rec.tar && log
"Download da imagem de recuperação finalizada.") || log "ERRO: Não foi possível
realizar o download da imagem de recuperação."
}

```

```

#Define configurações do riso
configuracoes() {
    dialog \
    --title 'AVISO' \
    #mensagem de alerta enviada para o usuário.
    --yesno '\nAlterar esse arquivo é potencialmente
    perigoso, se não souber o que esta fazendo pare agora.\n\nDeseja
continuar?' \
    0 0
    if [ "$?" -eq "0" ]; then
        log "Iniciando configuração manual do arquivo de configuração."
        nano $riso_conf #abre o arquivo de configuração do riso, $riso_conf com o
editor de texto nano.
        log "Finalizada a edição manual do arquivo de configuração."
    fi
}

#Atualiza R.I.S.O..
atualiza() {
    log "Iniciando atualização do script."
    log "Verificando disponibilidade do servidor de atualização."
    ping -q -c 1 200.131.37.236 > /dev/null 2>&1 #pinga o servidor para verificar
se realmente a disponibilidade.
    if [ "$?" -eq "0" ]; then
        log "Iniciando download do script novo." #wget é como se fosse um link de
download em modo de texto.
        #faz o download do riso pelo ip do site.
        (wget 200.131.37.236/riso/riso0.5 -O /usr/riso/riso && log "RISO
atualizado com sucesso.") || log "ERRO: Não foi possível baixar o novo script."

        #Mensagem de atualizado com sucesso.
        dialog \
        --ok-label 'OK' \
        --title 'Atenção' \
        --msgbox '\n RISO atualizado com sucesso.\n' \
        7 39

        bash /usr/riso/riso
        exit
    else #Mensagem caso a sua versão já for a mais recente.
        dialog \
        --ok-label 'OK' \
        --title 'ERRO' \
        --msgbox '\n Esta versão já é a mais recente'\
        7 40
        log "ERRO: RISO já é a versão mais nova."
    fi
}

helpi() { #função de ajuda (help).
    dialog \
    --ok-label 'OK' \
    --title 'Help' \
    --msgbox '\n
    Envie um e-mail com suas duvidas e sugestões \n
    para: "riso@comp.eng.br" e reponderei o \n
    mais rápido possível.' \
    0 0
}

```

```

#Pessoas que desenvolveram esse material.
creditos() {

    dialog \
    --ok-label 'OK' \
    --title 'Creditos' \
    #exibe nos creditos do programa os nomes dos magnatas que desenvolveram essa
    aplicação.
    --msgbox '
        CENTRO FEDERAL DE EDUCACAO TECNOLOGICA \n
        Engenharia da Computacao \n
        \n
        Cristiano Goulart Lopes Dias \n
        Vinicius Tinti de Paula Oliveira \n
        Germano Teixeira de Miranda \n
        Gabriel de Souza Brandao \n
        Marcio J. Menezes Jr. \n
        Gabriel Machado de Castro Fonseca \n
        André Luiz Silveira Herculano \n
        Gilmar Pereira de Alcântara \n
        \n
        www.dgo.cefetmg.br \n
        \n' \

    0 0
}

```

```

#Checklist de seleção.
menu_instala_imagens() {

    opcao=$( dialog --stdout \
    --ok-label 'Confirmar' \
    --checklist 'Quais imagens deseja instalar:' \
    0 0 0 \
    Windows '' ON \ #deixa os botões pre-selecionados altomaticamente.
    Linux '' ON ) #deixa os botões pre-selecionados altomaticamente.

    # De acordo com a opção escolhida, executa funcoes
    case $opcao in #salva sua escolha ne uma variável, nesse caso $opcao.
        "Windows") instala_windows;;
        "Linux") instala_linux;;
        "Windows" "Linux") instala;;
    esac

}

```

```

#Checklist de seleção.
menu_baixa_imagens() {

    opcao=$( dialog --stdout \
    --ok-label 'Confirmar' \
    --checklist 'Quais imagens deseja baixar:' \
    0 0 0 \
    Windows '' ON \ #deixa os botões pre-selecionados altomaticamente.
    Linux '' ON ) #deixa os botões pre-selecionados altomaticamente.

    # De acordo com a opção escolhida, executa funcoes
    case $opcao in #salva sua escolha ne uma variável, nesse caso $opcao.
        "Windows") baixa_imagem_windows && instala_windows;;
        "Linux") baixa_imagem_linux && instala_linux;;
        "Windows" "Linux") baixa_imagens && instala;;
    esac
}

```



```

esac

}

#Menu principal.
menu() {

    while : ; do
        opcao=$(
            dialog --stdout \
                --ok-label 'Confirmar' \
                --cancel-label 'Sair' \
                --title 'RISO - 0.5' \
                --menu 'Escolha o que voce quer fazer:' \
                    0 0 0 \
                    1 'Instalar imagens ja existentes' \
                    2 'Baixar e instalar novas imagens' \
                    3 'Baixar e instalar imagem de recuperação(LIVE_CD)' \
                    4 'Atualiza' \
                    5 'Configurações' \
                    6 'Creditos' \
                    7 'Help' )

        # ESC, sai do programa...
        [ $? -ne 0 ] && break

        # De acordo com a opção escolhida, executa função
        #Salva a escolha em uma variável, no caso $opcao.
        case $opcao in
            1) menu_instala_imagens;;
            2) testaip && menu_baixa_imagens;;
            3) baixa_rec; instala_rec;;
            4) atualiza;;
            5) configuracoes;;
            6) creditos;;
            7) helpi;;
        esac

    done

}

#Avalia argumentos.
avaliar_args() {

    #Cria vetor com os argumentos recebidos
    #argumento[0] = numero de argumentos passados.
    argumento=( $# $@ )

    #Verifica se algum argumento foi passado
    if [ $# = 0 ]; then
        return
    #Mostra help se requisitado
    elif [ ${argumento[1]} = "--help" ]; then
        echo "Uso: riso [OPÇÃO] [ARQUIVO]..."
        echo "Baixa e/ou instala imagen(s) do(s) sistema(s) operacional(s)"
        echo ""
        echo "    --baixa, baixa as imagens sucessoras"
        echo "    windows --imagem do windows"
        echo "    linux --imagem do linux"
        echo "    rec --imagem do sistema de recuperação"
    fi
}

```

```

echo "
echo "      --instala, instala as imagens sucessoras
echo "      windows --imagem do windows
echo "      linux --imagem do linux
echo "      rec --imagem do sistema de recuperação
echo "
echo "A imagem de recuperação(rec) deve ser baixada e/ou instalada"
echo "a partir de um SO externo ao HD (Ex: cd de boot).
echo "
echo "Exemplo:
echo "      Baixa imagens do linux e do windows e instala a do linux"
echo "      riso --baixa windows linux --instala linux
echo "
echo "Comunicar \"bugs\" para <riso@comp.eng.br>
exit

#Tenta executar
else
    #Verifica syntax
    for i in $(seq 1 ${argumento[0]}); do
        if [ ${argumento[$i]} = "--baixa" ]; then
            until [ $i -eq ${argumento[0]} -o ${argumento[$i]} = "--instala"
]; do

                ((i++))
                case ${argumento[$i]} in
                    "windows") break;;
                    "linux") break;;
                    "rec") break;;
                    *) echo "riso: Syntax ERRO"; echo "Tente \"riso --help\"
para mais informações"; exit;;
                esac
            done
            elif [ ${argumento[$i]} = "--instala" ]; then
                until [ $i -eq ${argumento[0]} -o ${argumento[$i]} = "--baixa" ];
do

                ((i++))
                case ${argumento[$i]} in
                    "windows") break;;
                    "linux") break;;
                    "rec") break;;
                    *) echo "riso: Syntax ERRO"; echo "Tente \"riso --help\"
para mais informações"; exit;;
                esac
            done
            elif [ ${argumento[$i]} = "windows" ]; then
                continue
            elif [ ${argumento[$i]} = "linux" ]; then
                continue
            elif [ ${argumento[$i]} = "rec" ]; then
                continue
            else
                echo "riso: Syntax ERRO"
                echo "Tente \"riso --help\" para mais informações"
                exit
            fi
        done

        #Executa comando
        carrega_variaveis || return 1
        for i in $(seq 1 ${argumento[0]}); do
            if [ ${argumento[$i]} = "--baixa" ]; then

```

```

        until [ $i -eq ${argumento[0]} -o ${argumento[$i]} = "--instala"
]; do
    ((i++))
    case ${argumento[$i]} in
        "windows") baixa_imagem_windows;;
        "linux") baixa_imagem_linux;;
        "rec") baixa_rec;;
    esac
done
elif [ ${argumento[$i]} = "--instala" ]; then
    until [ $i -eq ${argumento[0]} -o ${argumento[$i]} = "--baixa" ];
do
    ((i++))
    case ${argumento[$i]} in
        "windows") instala_windows;;
        "linux") instala_linux;;
        "rec") instala_rec;;
    esac
done
elif [ ${argumento[$i]} = "windows" ]; then
    continue
elif [ ${argumento[$i]} = "linux" ]; then
    continue
elif [ ${argumento[$i]} = "rec" ]; then
    continue
fi
done
fi
reboot
}

#Verifica se usuário é root antes de iniciar o riso.
USER=`id -u`
if [ $USER == '0' ]; then #se o usuário for root, se da inicio ao riso
    log "Iniciando RISO -----"
    avaliar_args $@
    menu
    log "Encerrando RISO -----"
    clear
else
    echo 'Só o root pode fazer isso, jovenzinho!'
fi

```

GitHub

As informações passadas nesse tópico são básicas, mas sendo que através delas você terá a capacidade de manipular seus arquivos no GitHub.

Caso tenha alguma dúvida sobre o Git, recomendamos que assista o vídeo e leia mais sobre o assunto nos links que se encontram na referência desse manual.

O que é o Git?

O Git é um sistema de controle de versão (ou versionamento) distribuído que está sendo cada vez mais usado, pois foca em velocidade e robustez.

Um sistema de controle de versão tem a finalidade de gerenciar as diferentes versões de um documento.

Exemplos de projetos que utilizam o Git: Arch Linux, Android, Debian, Digg, Eclipse, Fedora, JQuery, openSUSE, GNOME, entre outros.

O que é o GitHub?

O GitHub é um serviço de hospedagem distribuído, desenvolvidos em Ruby on Rails para projetos que utilizam o controle de versão Git. Assim, é utilizado como repositório online de códigos fonte para projetos de código aberto.

Podemos encontrar nele informações sobre todos os commits (atualizações) dos projetos que o utilizam, uma rede social que possibilita que outras pessoas acompanhem o desenvolvimento de seu projeto, um recurso para visualizar gráficos de quantas atualizações cada pessoa no projeto está realizando, entre outras novidades.

Criando um repositório remotamente

Para começarmos é muito importante a criação de uma conta no GitHub. Para isso é necessário acessar o site <https://github.com/>

Após ter criado sua conta no GitHub, faça seu login e crie um repositório. Copie o endereço do link que se encontra na parte inferior direita da página. Esse link será muito importante para trabalharmos no Git.

Não entraremos em detalhes sobre como criar uma conta e repositório pois é algo trivial.

Trabalhando com Git via terminal

Instalando o Git

Acesse seu terminal e instale o Git. A instalação é feita com esse comando:

```
sudo apt-get install git-core
```

Clonando o Git

Com o Git instalado em sua máquina, você precisa clonar aquele repositório feito remotamente. Para clonar esse repositório você deve usar os seguintes comandos: **git clone <link>** esse link é aquele copiado anteriormente.

Exemplo: **git clone** <https://github.com/usuario/testes.git>.

Feito isso, o repositório remoto será baixado em sua máquina.

Informações do repositório

Para visualizar as informações do repositório remoto você utiliza o comando:

git remote -v

Logo após feito o comando, ele mostrará o repositório aonde foi feita a busca, repositório de origem (Fetch) e aonde você pode fazer o Push, que é fazer mudanças no repositório.

Atualizando repositório

Com o comando **git pull** você pode baixar a atualização do código. Muitas vezes estão várias pessoas aprimorando o projeto, então é sempre bom você usar esse comando, pois caso algum desenvolvedor estiver feito alguma mudança no projeto você ficara sabendo qual mudança foi feita. O uso desse comando é **extremamente importante**.

Status do repositório

Você também pode pedir para o Git informar qual é o status desse repositório, toda vez que você for fazer alguma manipulação você poderá usar desse recurso do Git. O comando é: **git status**

Adicionado mudanças no arquivo

Sempre que você for modificar um arquivo ou alterado, você deve adicioná-lo. Vamos supor que você abriu um arquivo qualquer e o modificou de alguma forma. Se você der git status ele listara que você tem mudanças a serem feitas.

Para adicionar as mudanças a serem feitas nesse arquivo é usado o comando:

git add <nome do arquivo>

Commitando arquivo

Depois de usar o git add, ele já reconhece o arquivo como conteúdo relevante para o repositório, sendo que você precisa efetivar o commit. Esse commit é um comentário de mudanças do

arquivo, nele você colocará um breve comentário daquilo que você modificou no arquivo.

Para efetivar o commit você usa o seguinte comando: **git commit -m “Descrição do commit”**

O parâmetro **-m** indica que você está enviando uma descrição desse commit.

Informações de log (commit)

Você também pode visualizar o commit no histórico do repositório com esse comando: **git log**

O **git log** irá exibir o id (código hash) completo do commit, autor, data e o que foi commitado.

Junto com o comando **git log** você pode colocar o **-p** para exibir as diferenças de um commit para outro, e você coloca também quais commits irá comparar. Exemplo, colocando o **-2** ele irá comparar os últimos dois commits. Ficando assim o comando:

git log -p -2

Identificando mudanças no arquivo

Você pode usar o comando **git diff** para identificar quais são as mudanças a serem feitas no arquivo. Esse comando só funcionará quando você efetuar alguma modificação, sendo que ainda o arquivo não tenha sido adicionado na staged com o comando **git add**.

Removendo arquivos

Caso você queira remover um arquivo do repositório, você deve usar o comando **git rm <nome do arquivo>** e para efetivar essa remoção você deve o **git commit -m “Descrição da remoção”** Feito isso de um **git push <nome do repositório alterado> master** para mudar o repositório remotamente.

Revertendo modificações

Caso você tenha modificado um arquivo mas queira reverter (voltar a versão anterior). Essa reversão pode ser feita em duas ocasiões, sendo que uma você pode reverter com a versão do staged e a outra você pode reverter a mudança que você fez no staged com o repositório.

Revertendo a modificação com a versão do staged, se usa esse comando:

git checkout <nome do arquivo>

Essa outra reversão é quando você já estiver adicionado o arquivo com **git add**, se usa o comando **git reset HEAD <nome do arquivo>**. Logo depois você deve usar o **git checkout** para cancelar essa mudança e voltar o arquivo ao estado original.

Upando alterações locais para o repositório remoto

Muito bem, após você ter modificado o arquivo, adicionado as mudanças feitas e commitado. Agora está faltando você pegar o arquivo local e colocá-lo remotamente, para o controle de versão do GitHub.

Para isso você vai usar o comando: **git push <nome do repositório alterado> master**

Logo em seguida ele irá pedir o nome de usuário do repositório remoto e a senha. Colocando os dados corretos será feita a alteração no repositório.

REFERÊNCIAS

<http://www.guiafoca.org/>
<http://aurelio.net/shell/canivete/>
<http://linux.die.net/man/1/avahi-browse>
<http://www.remastersys.com/>
<http://www.diolinux.com.br/2014/04/remastersys-no-ubuntu-1404LTS.html>
<https://www.youtube.com/watch?v=yNwh0S0S0bU>
<http://tableless.com.br/iniciando-no-git-parte-1/>
<http://tableless.com.br/iniciando-no-git-parte-2/>