
Table of Contents

Introdução	1.1
Capítulo 1 - OnLiners em Shell	1.2
Capítulo 2 - Open Street Maps	1.3
Capítulo 3 - Open Stack: Implantação e uso	1.4
Capítulo 4 - O Projeto Debian quer você!	1.5
Capítulo 5 - Semeando liberdade: como (e onde) o software livre inclui as pessoas	1.6
Capítulo 6 - Programando Hardware: Utilizando Recursos GPIO do Raspberry PI	1.7
Capítulo 7 - Usando o R como calculadora financeira	1.8
Capítulo 8 - Computação Paralela com MPI (Message Passing Interface)	1.9

Software Livre: Experiências

IX Fórum de Tecnologia em Software Livre de Curitiba

O **Fórum de Tecnologia em Software Livre de Curitiba** - [FTSL](#) é principalmente resultado do engajamento da comunidade de software livre. Sua origem, entretanto, deu-se em um cenário no qual as políticas de governo eletrônico incentivaram o uso de software livre por entidades governamentais - e o compartilhamento do conhecimento gerado com a sociedade.

Em 2003 a diretoria do Serviço Federal de Processamento de Dados - [SERPRO](#) - instituiu o Programa Serpro de Software Livre - [PSSL](#) - para disciplinar e incentivar o uso de software livre na empresa. Em 2004 a diretoria do SERPRO instituiu o uso de soluções que utilizassem software livre em seus servidores de rede local e nas suas estações de trabalho. Em 2005 a então superintendência de soluções de desenvolvimento do SERPRO instituiu um projeto de padrões abertos para prospectar e avaliar as novas tecnologias em software livre para o desenvolvimento de soluções em TI. Em 2006 foi instituído o Projeto Nacional de Inclusão Digital do SERPRO, para implantação de Telecentros Comunitários e outros espaços com forte uso de software livre.

Em 2007 foi realizada a primeira edição do **Fórum de Tecnologia em Software Livre**, dentro das instalações da unidade de Curitiba do SERPRO, promovido por funcionários da empresa. A segunda edição ocorreu novamente no SERPRO, em novembro de 2008 e teve a participação do Centro de Computação Científica e Software Livre - [C3SL](#) - da Universidade Federal do Paraná - UFPR. Em maio de 2010 ocorreu a terceira edição, ainda no SERPRO, com a participação do Centro de Competência em Software Livre - [CCSL](#) - da Universidade de São Paulo - USP. A quarta edição se realizou em novembro de 2011 no Departamento de Informática da UFPR. A partir da quinta edição, que ocorreu em outubro de 2013, o evento passou a ocorrer no campus central da Universidade Tecnológica Federal do Paraná - [UTFPR](#). Hoje o evento é realizado por uma organização que combina analistas e técnicos do SERPRO, professores da UTFPR e membros da comunidade de software livre de Curitiba.

Na 8^a edição do evento foram publicados pela primeira vez os anais com os trabalhos apresentados. Na 9^a edição surgiu a ideia de publicar um livro com o conteúdo apresentado no evento. Este livro reúne artigos de palestrantes e instrutores do IX FTSL, que se dispuseram a compartilhar seu conhecimento técnico com a sociedade.

Capítulo 1

OnLiners em Shell

Júlio Cesar Neves

Prefácio

Frase de Um Japonês:

Quanto menor, melhor!

O Americano chamaria de One-Liners

Eu prefiro chamá-lo de Método KISS (Keep It Simple Stupid)

Como isso começou

O Desafio era fazer o menor programa possível para contar a quantidade de arquivos com cada extensão

```
$ ls | cut -sf2 -d. | sort | uniq -c
 2 log
 1 ods
 1 odt
 3 rtf
 1 sxc
 1 sh
 5 sxi
```

Logo depois, na excelente lista Shell-Script do Yahoo (<http://br.groups.yahoo.com/group/shell-script>), aparece a seguinte dúvida:

- Qual a melhor forma de dizer quantas vezes cada vogal aparece em uma frase.

A melhor resposta foi a seguinte:

```
$ echo "uma FrasE muitissimo legAL" |
  tr '[:upper:]' '[:lower:]' |
  grep -o -E '[aeiou]' | sort | uniq -c
3 a
2 e
3 i
1 o
2 u
```

- Autor da resposta: Tiago Barcellos Peczenyj

Substituição de Processos

Você sabe que só se pode passar a saída de um comando para a entrada de outro com um pipe (|), né?

ERRADO, veja isso:

```
$ cat <(ls)
arq1
arq2
arq3
```

Agora veja esta maluquice:

```
$ ls -l >(cat)
l-wx----- 1 jneves jneves 64 2006-05-12 11:34 /dev/fd/63 -> pipe:[15199]
```

l-wx----- é um link simbólico, mas não é 777

/dev/fd/63 é um file descriptor de named pipe

pipe:[15199] aponta para um pipe temporário

E pra que serve esta traquitana? Ahh! Para um monte de coisas, veja:

```
$ ls | while read arq; do let i++; echo $i $arq; done
1 arq1
2 arq2
3 arq3
```

Porém veja só:

```
$ echo $i
$
```

Agora veja com substituição de processos:

```
$ while read arq; do let i++; echo $i $arq; done < <(ls)
1 arq1
2 arq2
3 arq3
$ echo $i
3
```

Então vamos ver um one-liner poderoso usando substituição de processos:

```
$ diff <(ls -l dir) <(ls -l dir.bkp)
```

Ahh! Não tá satisfeito? Então vamos ver de outra forma:

```
$ cmp <(cat dir/*) <(cat dir.bkp/*)
```

Na lista de Shell do Yahoo, um colega mandou a seguinte solicitação: "Eu consigo ver se um arquivo tem uma determinada palavra com o grep -i 'palavra chave' arquivo. Porém eu queria fazer de forma automática, com que todos os arquivos que não contém a palavra chave fossem excluídos. valew"

Vi a oportunidade do one-liner usando substituição de processos e mandei de bate-pronto:

```
$ rm -i $(comm -13 <(grep -li 'palavra chave' *) <(ls))
```

Pensei comigo: "Que maravilha este one-liner! Rebentei!" Não passaram 10 minutos até o Tiago Barcellos Peczenyj, um colega de lista, mandasse essa:

```
$ grep -iL 'palavra chave' *
```

O Comando xargs

O comando `xargs` foi feito para remediar um erro comum nos UNIXES antigos que era “Too many arguments” gerados pelo comando `find`.

Forma caretá:

```
$ find $1 -type f -exec grep -l "$2" {} \;
```

Forma “envenenada”:

```
$ cat grep.r  
#  
# Grep recursivo  
# Pesquisa a cadeia de caracteres definida em $2 a  
# partir do diretório $1  
#  
find $1 -type f -print | xargs grep -l "$2"
```

O cara resolveu remover todos os seus arquivos do diretório corrente. Como ficaria melhor?

Forma 1:

```
$ find . -user cara -maxdepth 1 -exec rm -f {} \;
```

Forma 2:

```
$ ls -1 | grep " cara " | cut -c55- | xargs rm
```

Agora olha esse para remover todos com extensão .txt, mostrando seus nomes:

```
$ find . -type f -name "*.txt" | \  
xargs -i bash -c "echo removendo {}; rm {}"
```

A opção `-i` do `xargs` troca pares de chaves `{}` pela cadeia que está recebendo pelo pipe `()`.

O default do `xargs` é o comando `echo` e a opção `-n` diz a quantidade de argumentos que serão tratados de cada vez.

```
$ seq 5 | xargs -n 2  
1 2  
3 4  
5
```

Olha o one-liner aí, gente!!!

```
$ ls dir  
arq1.bug  
arq1.ok  
...  
arq9.bug  
arq9.ok  
$ ls | xargs -p -n2 diff -c  
diff -c arq1.bug arq1.ok ?....y  
...  
diff -c arq9.bug arq9.ok ?....y
```

A opção **-p** Pergunta se você realmente deseja fazer a operação.

O Comando paste

Para gerar uma saída tabulada:

```
$ ls arq[1-8] | paste -s -d'\t\t\n'  
arq1  
arq2  
arq3  
arq4  
arq5  
arq6  
arq7  
arq8
```

Mas... Dá para encolher um pouco?

Claro que dá, veja só:

```
$ ls arq? | paste - - -  
arq1  
arq2  
arq3  
arq4  
arq5  
arq6  
arq7  
arq8
```

\$ paste sd+ impar | bc 25

Para fazer um fatorial:

```
$ seq 4 | paste -sd\ $\times$   
1 $\times$ 2 $\times$ 3 $\times$ 4  
$ seq 4 | paste -sd\ $\times$  | bc  
24
```

Mas... Dá pra encolher um pouco?

Claro que dá, veja só:

```
$ seq -s\ $\times$  5  
1 $\times$ 2 $\times$ 3 $\times$ 4 $\times$ 5  
$ seq -s\ $\times$  5 | bc  
120
```

Também dá para fazer de outra forma:

```
$ echo {1..6} | tr ' ' +  
1+2+3+4+5+6  
$ echo {1..6} | tr '+' '\*' | bc  
720
```

Ou ainda:

```
$ echo $(( $(echo {1..6} | tr '+' '\*')) )  
720
```

A pergunta não é se dá para fazer em Shell. Pergunte qual é a melhor forma de fazer em Shell!

Sites imperdíveis:

- <http://shellscrip.com.br/>
- <https://jneves.wordpress.com/>
- <http://www.thobias.org>
- <http://br.groups.yahoo.com/group/shell-script>

Capítulo 2

Open Street Maps

Capítulo 3

OpenStack: Implantação e uso

Wilson Horstmeyer Bogado

Apresentação

O [Openstack](#) é um software modular que permite implantar uma infraestrutura de nuvem pública ou privada. O Openstack é utilizado como base por diversas empresas de tecnologia de grande porte.

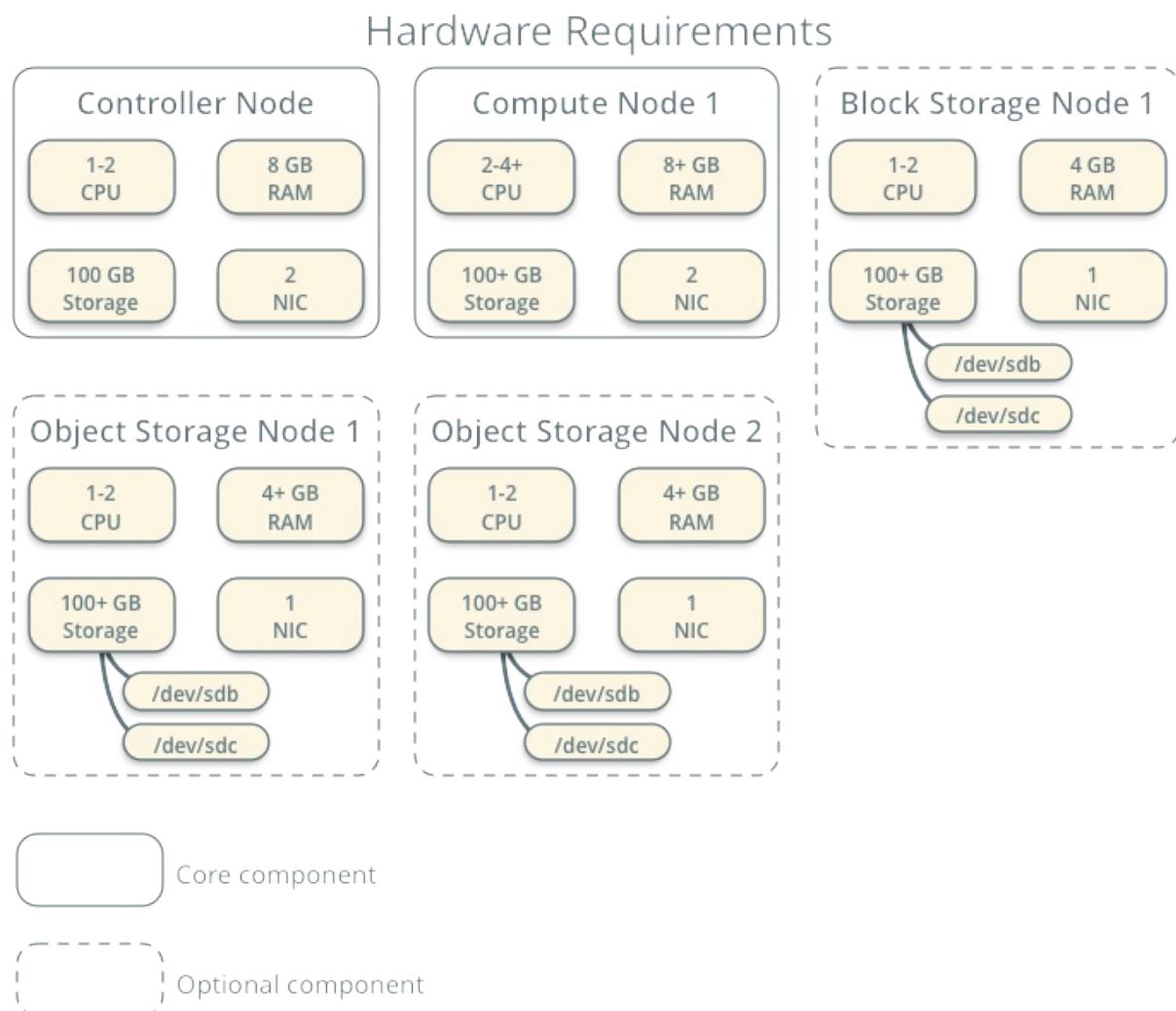
Embora seja um software bastante complexo, por sua característica modular, pode ser implementado em partes e de acordo com a necessidade de cada organização. Instalados os serviços básicos, outros serviços podem ser adicionados a qualquer momento sem interferir com os serviços existentes.

Um completo [tutorial de instalação](#) está disponível para as principais distribuições de Linux. As distribuições cobertas na seção *Deploy OpenStack* são Ubuntu, RHEL/Centos e SUSE.

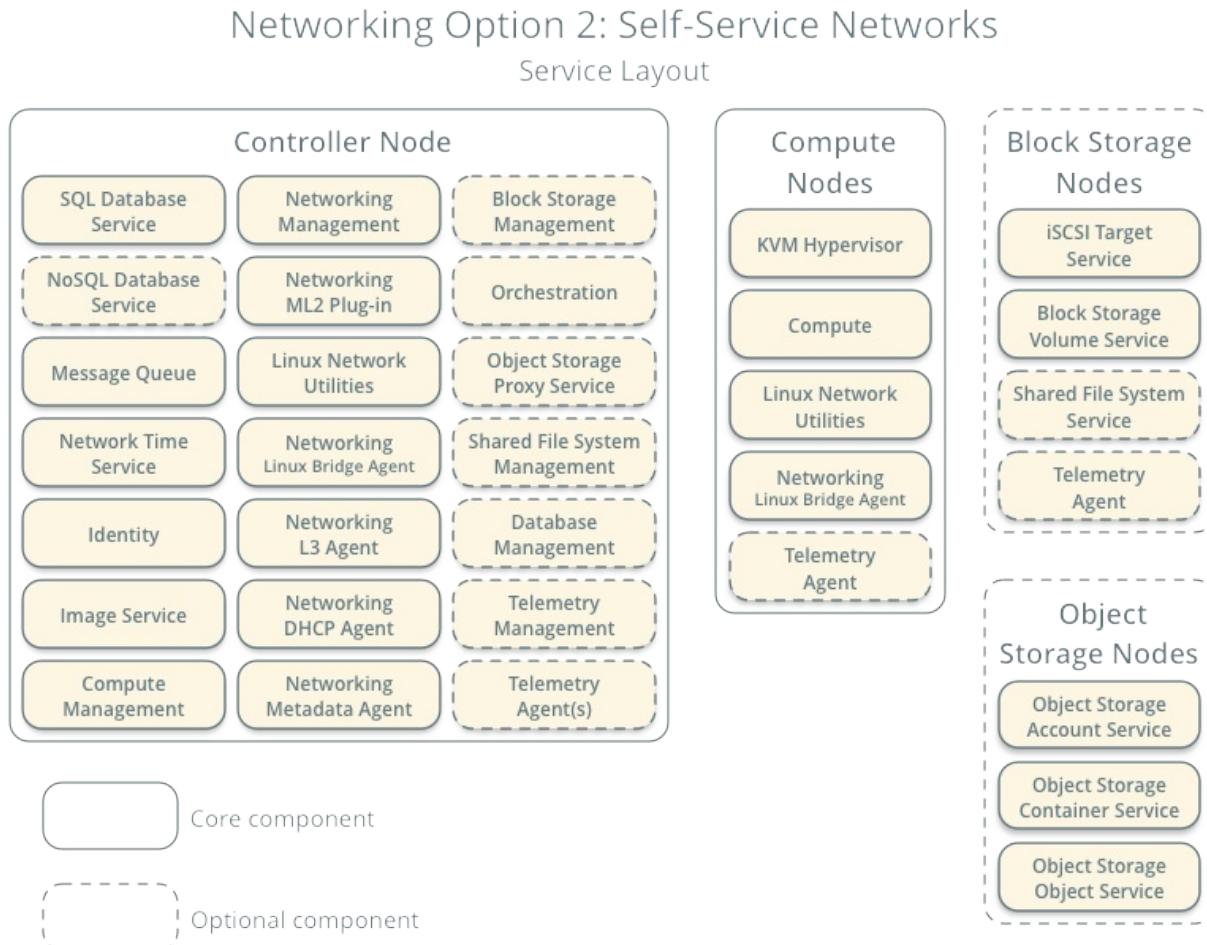
Requisitos de hardware e software

O OpenStack é composto por uma estrutura modular. Embora seja possível realizar uma instalação mínima em uma única máquina (física ou virtual) para fins didáticos, uma instalação de produção requer várias máquinas físicas para distribuir adequadamente os diversos serviços.

A figura abaixo ilustra uma possível arquitetura de hardware:



A figura abaixo ilustra uma possível arquitetura de software:



Nó de controle: Executa os serviços de identidade, imagem, gerenciamento do serviço de computação, gerenciamento do serviço de rede, vários agentes de rede e o console. Além disso este nó abriga serviços de suporte tais como o banco de dados SQL, fila de mensagens e NTP. Necessita de pelo menos duas interfaces de rede.

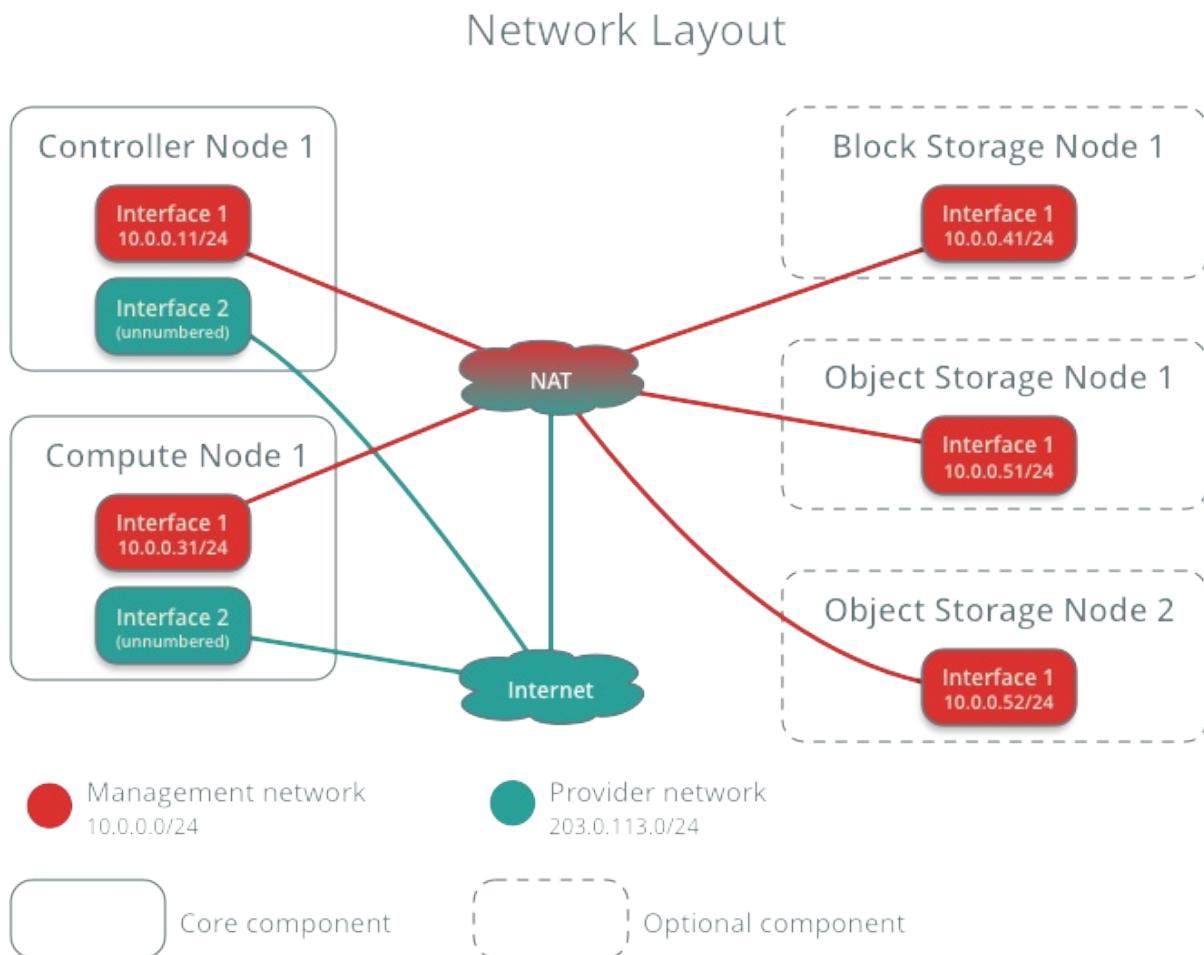
Nó de computação: Executa a parte do hipervisor que opera instâncias. Por padrão, o nó de computação usa o hipervisor KVM. O nó de computação também executa um serviço de rede que conecta instâncias às redes virtuais e provê serviços de firewall para as instâncias através de grupos de segurança.

Nó de armazenamento de blocos: Contém os discos que o serviço de armazenamento provisiona para as instâncias.

Rede física

Uma vez instalado o sistema operacional desejado em cada um dos nós, é necessário configurar as interfaces de rede. Todos os nós requerem acesso à internet para fins administrativos tais como instalação de pacotes, atualizações, DNS e NTP.

A figura abaixo exemplifica uma hipotética configuração de rede física:



Instâncias podem ser conectadas diretamente à rede de gerenciamento (pública) ou à rede virtual (privada).

Serviços do Openstack

Alguns serviços essenciais para operação do Openstack são:

- **Gerenciamento de redes virtuais (neutron)**: Suporte à criação de redes virtuais.
- **Provisionamento de recursos de computação (nova)**: Suporte ao provisionamento de máquinas virtuais e recursos associados.
- **Gerenciamento de imagens (glance)**: Suporte à criação de imagens de sistemas operacionais para criação de máquinas virtuais.
- **Armazenamento em blocos (cinder)**: Alocação de volumes de dados alocados às máquinas virtuais.
- **Console de gerenciamento Web (horizon)**: Interface Web para criação de máquinas virtuais, volumes, redes além de outros serviços opcionais.

Outros serviços podem ser disponibilizados, por exemplo:

- **Máquina física (ironic)**: Suporte ao gerenciamento e provisionamento de máquinas físicas.
- **Orquestração de contêineres (magnum)**: Suporte a motores de orquestração de contêineres tais como Docker Swarm, Kubernetes e Mesos.
- **Banco de dados (trove)**: Suporte a provisionamento de bancos de dados.

- **Gerenciamento de credenciais (barbican)**: Suporte ao armazenamento de dados secretos tais como senhas, chaves criptográficas e certificados digitais.
- **Mensagens (zaqar)**: Suporte ao compartilhamento de informações entre componentes e aplicações distribuídas.
- **Armazenamento de objetos (swift)**: Suporte ao armazenamento e recuperação de objetos.
- **Orquestração (heat)**: Suporte à orquestração de recursos na nuvem.
- **Sistemas de arquivos compartilhados (manila)**: Acesso coordenado a sistemas de arquivos distribuídos.
- **Telemetria e alarmes (aodh)**: Dispara alarmes quando medidas ou eventos coletados excedem regras definidas.
- **Coleta de dados de telemetria (ceilometer)**: Eficientemente monitora dados relacionados aos serviços do OpenStack, coleta eventos e dados de monitoramento enviados por serviços, publica dados coletados em diversos meios.

Provisionamento

O provisionamento de máquinas virtuais e dos recursos associados pode ser feito através de comandos no terminal do nó de controle. O formato geral dos comandos do Openstack é:

```
openstack <serviço> <operação> <argumentos>
```

Por exemplo, para criar um volume bootável de 20GB baseado na imagem teste, com o nome de volume-teste, poderíamos executar:

```
$ openstack volume create --size 20 --image teste --bootable volume-teste
```

Pode-se obter ajuda sobre qualquer comando, total ou parcial com a opção -h. Por exemplo para obter informações sobre a operação do comando volume create poderíamos executar:

```
$ openstack volume create -h
usage: openstack volume create [-h] [-f {json,shell,table,value,yaml}]
                               [-c COLUMN] [--max-width <integer>]
                               [--fit-width] [--print-empty] [--noindent]
                               [--prefix PREFIX] [--size <size>]
                               [--type <volume-type>]
                               [--image <image> | --snapshot <snapshot> | --source <volume> | --source-replicated <re
plicated-volume>]
                               [--description <description>] [--user <user>]
                               [--project <project>]
                               [--availability-zone <availability-zone>]
                               [--consistency-group consistency-group]
                               [--property <key=value>] [--hint <key=value>]
                               [--multi-attach] [--bootable | --non-bootable]
                               [--read-only | --read-write]
                               <name>

Create new volume

positional arguments:
  <name>           Volume name

optional arguments:
  -h, --help        show this help message and exit
  --size <size>     Volume size in GB (Required unless --snapshot or
                    --source or --source-replicated is specified)
  --type <volume-type> Set the type of volume
  --image <image>    Use <image> as source of volume (name or ID)
  --snapshot <snapshot>
                    Use <snapshot> as source of volume (name or ID)
  --source <volume>  Volume to clone (name or ID)
```

```

--source-replicated <replicated-volume>
    Replicated volume to clone (name or ID)
--description <description>
    Volume description
--user <user>      Specify an alternate user (name or ID)
--project <project> Specify an alternate project (name or ID)
--availability-zone <availability-zone>
    Create volume in <availability-zone>
--consistency-group consistency-group>
    Consistency group where the new volume belongs to
--property <key=value>
    Set a property to this volume (repeat option to set
    multiple properties)
--hint <key=value>  Arbitrary scheduler hint key-value pairs to help boot
    an instance (repeat option to set multiple hints)
--multi-attach     Allow volume to be attached more than once (default to
    False)
--bootable         Mark volume as bootable
--non-bootable    Mark volume as non-bootable (default)
--read-only        Set volume to read-only access mode
--read-write       Set volume to read-write access mode (default)

output formatters:
  output formatter options

-f {json,shell,value,yaml}, --format {json,shell,value,yaml}
    the output format, defaults to table
-c COLUMN, --column COLUMN
    specify the column(s) to include, can be repeated

table formatter:
  --max-width <integer>
    Maximum display width, <1 to disable. You can also use
    the CLIFF_MAX_TERM_WIDTH environment variable, but the
    parameter takes precedence.
  --fit-width
    Fit the table to the display width. Implied if --max-
    width greater than 0. Set the environment variable
    CLIFF_FIT_WIDTH=1 to always enable
  --print-empty
    Print empty table if there is no data to show.

json formatter:
  --noindent
    whether to disable indenting the JSON

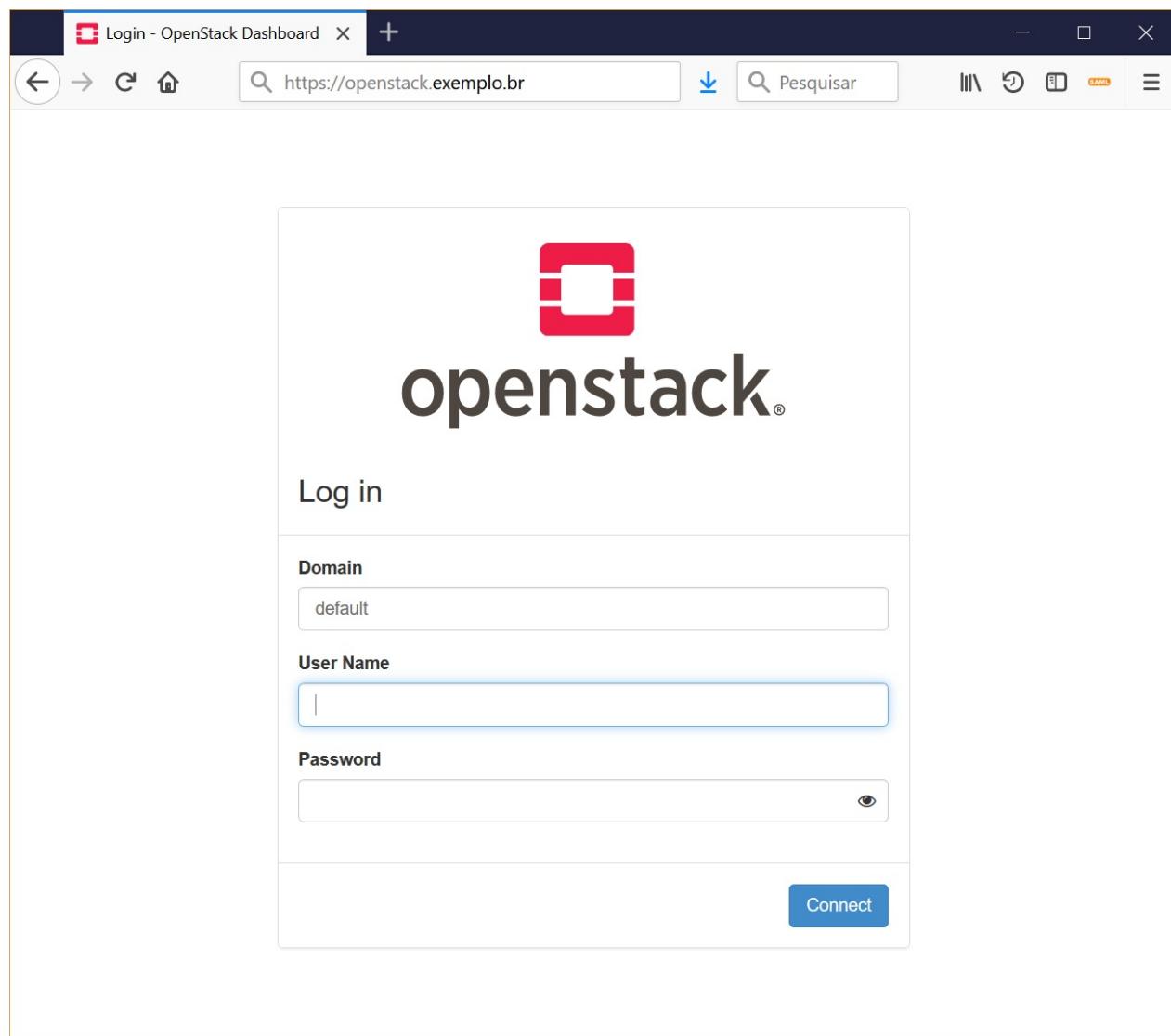
shell formatter:
  a format a UNIX shell can parse (variable="value")

--prefix PREFIX     add a prefix to all variable names

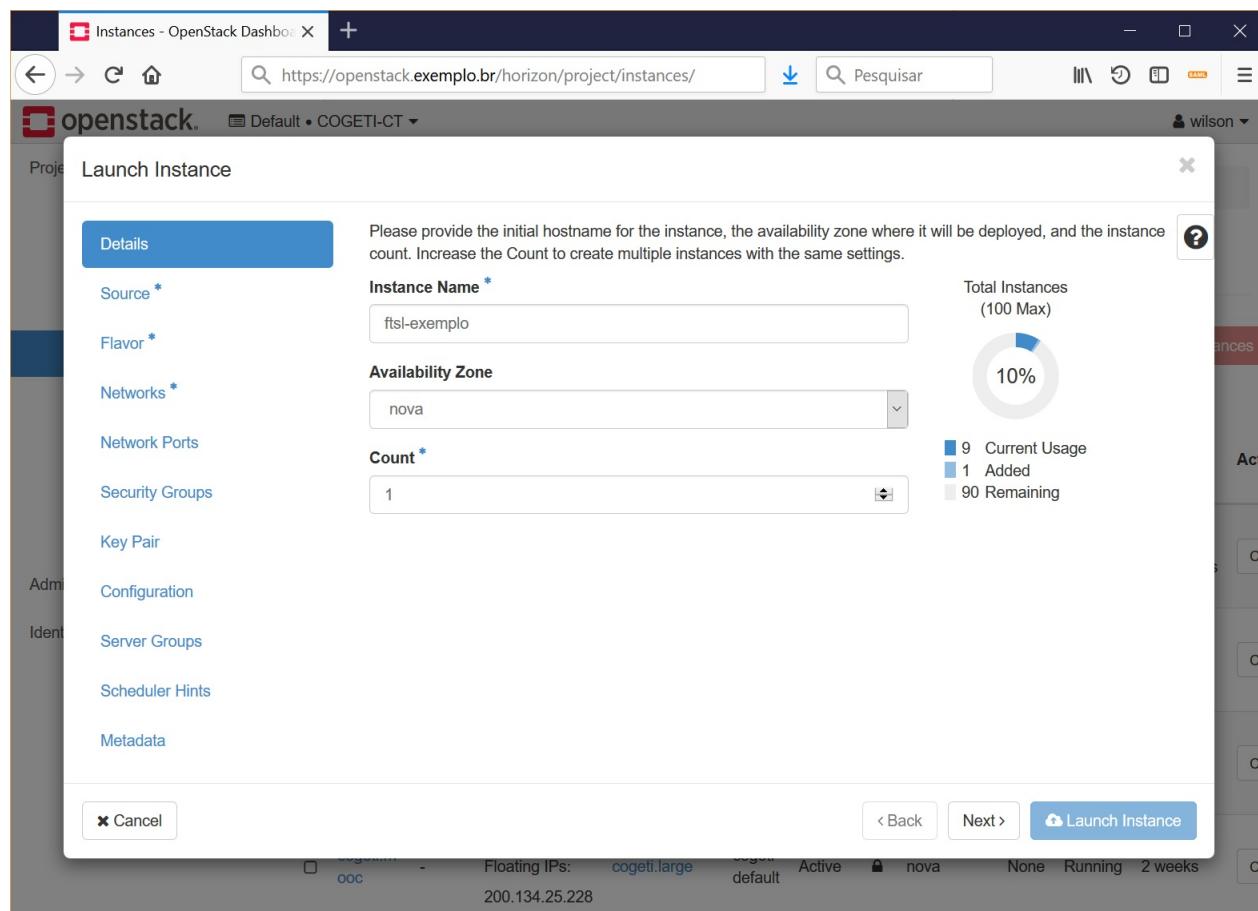
```

Embora os comandos do Openstack sejam relativamente fáceis, com opções bastante intuitivas, em geral usamos o console Web (horizon) para provisionar recursos.

A figura abaixo mostra a página de login do horizon:



Por exemplo, para criar uma nova instância (máquina virtual) no horizon usamos a opção Project / Compute / Instances:



Conclusão

A experiência de implantação do OpenStack, embora complexa e sujeita a certas dificuldades inerentes ao desenvolvimento paralelo de diferentes serviços interdependentes, mostra-se como uma alternativa viável para implantação de infraestruturas de nuvem privada dentro de organizações de médio e grande porte.

Wilson Horstmeyer Bogado é professor da Universidade Tecnológica Federal do Paraná (UTFPR), Campus Curitiba, Departamento Acadêmico de informática. Tem graduação em Engenharia Civil pela UFPR e mestrado em Métodos Numéricos para Engenharia pela UFPR. Foi Coordenador de TI do Campus Curitiba de 2013 a 2017 onde gerenciou a implementação da infraestrutura de virtualização de servidores, inicialmente com VMware. Atualmente colabora com a Coordenadoria de TI no desenvolvimento de software Web com tecnologia Java e na migração da infraestrutura para OpenStack.

Referências

- <https://openstack.org>
- <https://docs.openstack.org/newton/install-guide-ubuntu/>
- <https://docs.openstack.org/pike/install>

Capítulo 4

O Projeto Debian quer você!

Paulo Henrique de Lima Santana

Debian

O Debian é uma organização formada por pessoas de todo o mundo dedicada ao desenvolvimento de Software Livre e a promoção dos ideais da comunidade de Software Livre. O Projeto Debian começou em 1993 com o objetivo de desenvolver um sistema operacional livre para computadores. Traduzido para mais de 30 línguas, e suportando uma enorme variedade de arquiteturas de computadores, o Debian se intitula o sistema operacional universal e é hoje o maior projeto de Software Livre no mundo. Atualmente existem mais de 1.000 desenvolvedores oficiais do Debian e milhares de usuários que contribuem de alguma forma para o projeto.

Site do Projeto Debian: <http://debian.org>

IX Fórum de Tecnologia em Software Livre - FTSL

O Fórum de Tecnologia em Software Livre - FTSL, é um evento anual, realizado em Curitiba - PR, sendo considerado o maior evento gratuito de Software Livre do Brasil, e tem como propósito a disseminação de novas tecnologias baseadas em Software Livre, bem como, a troca de experiências com as comunidades, universidades e empresas públicas e privadas, por meio de palestras, painéis e oficinas.

A 9a edição do FTSL aconteceu de 27 a 29 de setembro de 2017 no Campus central da Universidade Tecnológica Federal do Paraná - UTFPR. No último dia do evento a comunidade Debian promoveu seis palestras na trilha chamada de minievento Debian. Minha palestra com o título **O Projeto Debian quer você!** foi a quarta realizada dentro deste minievento.

Site do FTSL: <http://debian.org>

Site da UTFPR: <http://utfpr.edu.br>

O Projeto Debian quer você!

Nesta palestra considero que você já sabe o que é o Debian e que agora quer saber como contribuir para o Projeto. Portanto, não explico o que é uma distribuição, qual o origem do Debian, como o nome foi criado, o Debian Social Contract, quais os nomes das versões, quais as diferenças entre stable, testing e unstable e entre main, contrib e non-free, etc.

Para ver arquivos de apresentações realizadas pela comunidade Debian, incluindo algumas de introdução com abordagens mais básicas, acesse: <http://debianbrasil.org.br/apresentacoes>

Qualquer pessoa pode contribuir para o Projeto Debian, independente do seu nível de conhecimento. Você pode contribuir de forma anônima, ou se registrando na página Contributors para manter suas contribuições registradas. Para se registrar, acesse: <https://contributors.debian.org>

Depois que você se torna um contribuidor registrado, você pode participar dos processos para se tornar um membro oficial. Existem três tipos de contribuidores oficiais.

- DM - Debian Maintainer (Mantenedor Debian).
- DD - Debian Developer non-uploading (Desenvolvedor Debian sem permissão de upload).
- DD - Debian Developer uploading (Desenvolvedor Debian com permissão de upload).

Para obter mais informações sobre como se tornar um DM e um DD, acesse respectivamente:
<https://wiki.debian.org/DebianMaintainer> e <https://wiki.debian.org/DebianDeveloper>

Você também pode assistir a palestra do João Eriberto Mota Filho sobre este tema realizada no FTSL no link <http://videos.softwarelivre.org/ftsl-2017/como-se-tornar-um-membro-oficial-do-debian-dd-ou-dm.html> e obter o arquivo da apresentação no link <http://www.eriberto.pro.br/palestras/debian-dd.pdf>

Vamos agora ver as formas para contribuir com o Projeto Debian.

0 - Empacotamento de software

O empacotamento de software é a principal forma de contribuir para o Debian, mas exige conhecimento técnico principalmente para usar o terminal. Contribuir com empacotamento permite que você se torne um DD uploading.

O empacotamento não foi abordado nesta palestra porque está disponível no YouTube uma série de vídeos tutoriais gravados João Eriberto Mota Filho com mais de 14 horas mostrando em detalhes como fazer empacotamento. Acesse o site do João Eriberto para ver mais detalhes: <http://debianet.com.br>

1 - Instale e use o Debian

A forma primária para contribuir para o Debian é: instalar e usar o Debian no seu computador ou notebook. Você pode obter uma cópia da imagem oficial do Debian neste link <https://www.debian.org/CD>. Basta você fazer o download, gravar em um DVD ou pendrive e instalar no seu equipamento.

Acesse este link para ver um site mais amigável para fazer o download: <http://get.debian.net>

A principal cópia do repositório do Debian no Brasil, chamado de espelho oficial, fica no Departamento de Informática da Universidade Federal do Paraná (DIInf UFPR) e pode ser acesso neste link: <http://ftp.br.debian.org>

Ao instalar e usar o Debian, você:

- Terá testado o instalador.
- Aumentará a base de usuários.
- Ficará familiarizado com um sistema GNU/Linux.
- Ajudará na divulgação.

2 - Report bugs

Ao encontrar um problema no Debian, envie um bug report (relatório de erros) para que os Desenvolvedores fiquem sabendo e possam corrigi-lo. Acesse este link para saber como relatar um bug no Debian:

<https://www.debian.org/Bugs/Reporting>

Você pode preencher o relatório via terminal com a ferramenta reportbug ou via interface gráfica usando a ferramenta reportbug-ng.

Os bugs no Debian possuem níveis de severidade:

- critical
- grave
- serious
- important
- normal
- minor
- wishlist

Alguns deste bugs são release-critical. Veja mais detalhes neste link: <https://bugs.debian.org/release-critical>

o DD Raphael Hertzog escreveu 7 dicas para enviar relatórios de bugs do Debian úteis e ter o seu problema resolvido (em inglês): <https://raphaelhertzog.com/go/bugreporting>

O Debian possui um Sistema de Acompanhamento de Bugs (BTS - Bug Tracking System) por meio do qual enviamos detalhes de bugs reportados por usuários e desenvolvedores. Cada bug é associado a um número e é mantido no arquivo até que seja marcado como tendo sido trabalhado. Mais detalhes neste link: <https://www.debian.org/Bugs>

Exemplos de bugs abertos no Gimp: <https://bugs.debian.org/cgi-bin/pkgreport.cgi?dist=unstable;package=gimp>

3 - Patches

Se você sabe como resolver um bug, você pode enviar um patch. Os Desenvolvedores irão analisar e se tudo estiver correto, eles aplicarão o seu patch dando o crédito a você pela ajuda, e assim você terá contribuído para o Debian.

4 - Documentação

Você pode produzir documentação para o Debian escrevendo:

- Manuais.
- Tutoriais.
- HOWTOs.
- FAQs.

Veja mais em <https://www.debian.org/doc> e <https://wiki.debian.org/Brasil/Documentos>

Você pode publicar tutoriais ensinando a fazer alguma coisa no Debian, levando em consideração as seguintes ideias:

- Tema técnico ou não técnico.
- Nível básico ou avançado.
- Publicação escrita ou em vídeo.

Alguns sites de brasileiros com dicas e tutoriais:

- <http://eriberto.pro.br/blog>
- <http://blog.silva.eti.br>
- <http://softwarelivre.org/terceiro>
- <http://www.debiandicas.org>
- <http://www.debiandicas.com.br>
- <http://debianmaniaco.blogspot.com.br>

5 - Suporte a outros usuários

Outra forma importante de contribuir para o Debian é ajudando a tirar dúvidas de outros usuários. Existem muitos locais que você pode participar. Os meios oficiais do Projeto Debian são:

- Lista de discussão `debian-user-portuguese`:
 - <https://lists.debian.org/debian-user-portuguese>
- Canal no IRC:
 - Server: `irc.debian.org`
 - Canal: `#debian-br`

Nos últimos anos surgiram outros espaços não oficiais mantidos pela comunidade:

- Fórum
 - <http://www.forumdebian.com.br>
- Grupos no facebook:
 - <https://www.facebook.com/groups/DebianBR>
 - <https://www.facebook.com/groups/5366446847>
- Grupos no telegram:

- <https://t.me/debianbrasil>

6 - Divulgação

Ajude a divulgar o Debian em espaços públicos onde tem pessoas com potencial para se tornarem usuárias do sistema operacional, como por exemplo:

- Sindicatos.
- Escolas.
- Universidades/Faculdades.
- Eventos como congressos, fóruns, encontros, etc.

Utilize seus perfis em redes sociais livres e fechadas para falar sobre Debian, encaminhar publicações de outras pessoas, divulgar os eventos e notícias relacionadas:

- Redes fechadas:
 - Twitter/Facebook/Google+
- Redes livres:
 - GNU Social <https://quitter.se>
 - Pump.io <http://pump.io>
 - Diaspora <https://diaspora.softwarelivre.org>
 - Hubzilla <https://hub.vilarejo.pro.br>

Use produtos com a logomarca Debian. A Comunidade Curitiba Livre vende vários produtos no site <http://loja.curitalivre.org.br> e o lucro obtido é usado para organizar eventos de Software Livre em Curitiba. Nessa loja online você achará:

- Camisetas.
- Canecas.
- Buttons, chaveiros, cordões de crachá, adesivos, etc.

7 - Publicidade

O Projeto Debian tem um time de publicidade (Publicity Team) que elabora textos de notícias e mantém alguns sites relacionados. Esse time é responsável por publicar:

- Debian Press Release <https://www.debian.org/News>
- Debian Project News <https://www.debian.org/News/weekly>
- Debian Bits, the Debian Blog <https://bits.debian.org>
- Debian Micronews <https://micronews.debian.org>

O time de publicidade também mantém os perfis oficiais do Debian em redes sociais livres que você pode seguir:

- Pump.io <https://identi.ca/debian>
- GNU Social <https://quitter.se/debian>

E como você pode contribuir com o time de publicidade? Você pode contribuir escrevendo notícias ou revisando os textos de outras pessoas, mas para isso é imprescindível saber inglês. Mais informações em:

<https://wiki.debian.org/Teams/Publicity>

Você pode ajudar com notícias em português enviando textos para o nosso blog da comunidade brasileira de usuários e desenvolvedores Debian. Obviamente esse blog não é oficial do Projeto Debian. Acesse: <http://debianbrasil.org.br>

8 - Organização de eventos

Todos os anos as comunidades ao redor do mundo organizam nas suas cidades o Debian Day para celebrar o aniversário do Debian que acontece em 16 de agosto (ou no sábado mais próximo). Você pode inscrever a sua cidade para participar no site: <https://wiki.debian.org/DebianDay>

No dia 17 de junho de 2017 foi lançada a nova versão do Debian - versão 9 chamada de Stretch. Veja algumas cidades que organizaram festas de lançamento (release party), inclusive no Brasil: <https://wiki.debian.org/ReleasePartyStretch>

Esses eventos podem ser desde um encontro em um bar ou em uma pizzaria, até um evento com palestras e oficinas relacionadas ao Debian. O importante é promover o encontro da comunidade local para celebrar o Debian.

9 - Produção de material gráfico

Você pode produzir materiais gráficos e disponibilizá-los para que outras pessoas utilizem livremente. Por exemplo:

- Desenhos para camisetas.
- Ícones.
- Logos.
- Mascotes.
- Temas/wallpapers.

Envie seu material para o Collab Debian, site que serve de repositório para as contribuições gráficas:

<http://collab.debian.net>

O tema de cada versão do Debian é escolhido por votação, e qualquer pessoa pode enviar propostas. O tema da versão 6 do Debian, chamada de Squeeze, foi feita pelo brasileiro Valéssio Brito. Ele também criou a logo do Debian Day e é o mantenedor do projeto Collab Debian.

Os temas das duas últimas versões do Debian (8 - Jessie e 9 - Stretch) foram criados por Juliette "Taka" Belin, estudante francesa de programação de computadores e multimídia. O nome do tema da Jessie é Lines e do Stretch é Soft Waves.

Para ver o portfólio da Juliette acesse o seu site pessoal <http://jbelin.com> e a sua galeria no DeviantArt <https://takaju.deviantart.com/gallery>

10 - Tradução

Uma das maiores contribuições que você pode dar ao Debian é ajudar a traduzi-lo do inglês para o português do Brasil. Normalmente os DDs non-uploading brasileiros contribuem intensamente com as traduções. O time brasileiro de tradução (L10n) mantém uma wiki com várias orientações para que você possa aprender e passar a colaborar:

<https://wiki.debian.org/Brasil/Traduzir>

Veja a seguir as áreas que você pode traduzir, com trechos retirados da wiki do time brasileiro de tradução.

DDTP - Debian Description Translation Packages

O DDTP - Debian Description Translation Project (em português Projeto de Tradução das Descrições de Pacotes Debian), é um projeto internacional cujo principal objetivo é traduzir as descrições dos pacotes do Debian, dessa forma os usuários que não leem inglês terão mais facilidade em descobrir quais pacotes são úteis a eles. O DDTP foi implementado pelo alemão Michael Brämer e fornece a infraestrutura necessária para apoiar o processo de tradução e revisão. Veja mais detalhes e informações na página sobre o DDTP: <https://www.debian.org/international/l10n/ddtp>

Durante anos a principal interface com o DDTP foi o e-mail, até que Martijn van Oosterhout desenvolveu o DDTSS - Debian Distributed Translation Server Satellite (em português Satélite do Servidor de Traduções Distribuídas Debian). O DDTSS é uma interface web que interage com o DDTP e seu banco de dados, fornecendo um ciclo de tradução e revisão para os diferentes times de tradutores.

A equipe brasileira foi o segunda a aderir ao DDTP e se mantém entre as mais ativas, graças ao trabalho de muitos voluntários. É muito fácil colaborar com o DDTP já que o processo de tradução e revisão é feito pela web e os textos a serem traduzidos/revisados são normalmente pequenos. Um curto espaço de tempo por dia pode ajudar muito a equipe.

A ferramenta oficial de tradução do DDTP para a equipe brasileira é o DDTSS. Acesse a página da nossa equipe em: http://ddtp2.debian.net/ddtss/index.cgi/pt_BR

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/DDTP>

Debian Installer (D-I) e Guia de Instalação

Composto por dois componentes principais: o próprio instalador e o guia de instalação, requer trabalho continuado e alinhamento das traduções entre os materiais. Atualmente está 100% traduzido.

Felizmente, nossos esforços de tradução estão bastante avançados e atualmente os pacotes apt, dpkg, dselect, aptitude e outros, todos parte do sistema básico Debian, estão traduzidos para o idioma português do Brasil.

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/DI>

Modelos debconf

O projeto consiste na tradução dos arquivos do tipo POT utilizados pelo debconf, o sistema de configuração de pacotes da distribuição Debian, durante a instalação de pacotes no sistema.

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/DebConf>

Páginas man

Este projeto visa a tradução de páginas de manual de pacotes específicos do Debian, tais como debconf, fakeroot, dpkg e kernel-package. Pois já existe na ldp-br um projeto de tradução de páginas de manual mais comuns.

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/ManPages>

Páginas web

Através das páginas web todos podem obter informações a respeito do Projeto Debian e suas diversas iniciativas. Para que as mesmas estejam disponíveis no idioma português do Brasil, uma equipe de tradutores, coordenada através da lista de discussão debian-i10n-portuguese, utiliza um sistema de controle de versão (CVS) onde os arquivos fonte, em formato WebWML (WML), ficam armazenados. Os mesmos são utilizados para geração das páginas em formato HTML durante o processo de reconstrução automática do site, que ocorre periodicamente.

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/WebWML>

As traduções dos arquivos POT e das páginas web WML seguem um ritual de envio de mensagens para a lista de discussão dos tradutores com o campo de assunto contendo as seguintes siglas:

- ITT - Intent To Translate (intenção de traduzir).
- RFR - Request For Review (requisição para revisão).
- LCFC - Last Chance For Comment (última chance para comentários).
- DONE (feito).

Para mais informações, acesse: <https://wiki.debian.org/Brasil/Traduzir/Pseudo-urls>

O status das traduções pode ser visto na seguinte página:

https://i10n.debian.org/coordination/brazilian/pt_BR.by_status.html

Uma boa dica é ver a palestra do Adriano Rafael Gomes no FISL16:

- Vídeo: <http://ur1.ca/qw5f8>
- Arquivo da apresentação: <https://bolicho.arg.eti.br/pub/eventos/2015/fisl16>

Mensagem final

Espero que após ler este documento e acessar os links indicados, você se sinta encorajado(a) a contribuir com o Projeto Debian.

A comunidade brasileira é uma grande usuária de Software Livre, mas precisamos passar de meros usuários(as) para contribuidores(as) de fato. Seja com código ou com qualquer outro tipo de contribuição, todos nós podemos e devemos ajudar os projetos de Software Livre.

Quando você começar a contribuir para o Debian, descobrirá que a comunidade é muito receptiva e calorosa com os(as) novatos(as). O Projeto mantém várias iniciativas para promover a diversidade. Leia mais sobre isso na página a seguir:
<https://www.debian.org/intro/diversity>

Venha fazer parte dessa comunidade incrível que é o Projeto Debian!

Minibiografia do autor

Paulo Henrique de Lima Santana.

Debian Maintainer (DM).

Bacharel em Ciência da Computação pela UFPR - Universidade Federal do Paraná.

Trabalha com administração de redes e sistemas GNU/Linux em Curitiba.

Entusiasta de Software Livre, participa de diversos grupos de atuação como a Comunidade Curitiba Livre (<http://curitibalivre.org.br>), o Grupo Debian Curitiba, e a ASL.Org - Associação Software Livre.Org (<http://asl.org.br>).

Palestrou em edições do FISL - Fórum Internacional Software Livre (<http://fisl.org.br>), da Latinoware - Congresso Latino-americano de Software Livre e Tecnologias Abertas (<http://latinoware.org>) e da Campus Party Brasil (<http://brasil.campus-party.org>).

Coordenou a organização de eventos em Curitiba como algumas edições do FLISOL - Festival Latino-americano de Instalação de Software Livre (<http://flisol.curitibalivre.org.br>), do SFD - Software Freedom Day (<http://sfd.curitibalivre.org.br>), do Circuito Curitibano de Software Livre (<http://circuito.curitibalivre.org.br>), e da MiniDebConf Curitiba (<http://br2016.mini.debconf.org> e <http://br2017.mini.debconf.org>).

Co-coordenou a organização da grade de programação do Fórum Internacional de Software Livre em 2015 (FISL16) e em 2016 (FISL17).

Curador de Software Livre da Campus Party Brasil desde 2013.

Contatos:

E-mail: phls@softwarelivre.org

Site pessoal: <http://phls.com.br>

Perfil no telegram: @phls00

Perfis em redes sociais livres:

- <http://quitter.se/phls>
- <http://identi.ca/phls00>
- <http://diaspora.softwarelivre.org/u/phls>
- <http://hub.vilarejo.pro.br/channel/phls>

Perfis em redes sociais fechadas:

- <http://twitter.com/phls00>
- <http://facebook.com/phls00>

Referências bibliográficas

Site do Debian: <https://www.debian.org>

Wiki do Debian: <https://wiki.debian.org>

Site da Comunidade Debian Brasil: <http://debianbrasil.org.br>

Site do FTSL - Fórum de Tecnologia em Software Livre: <http://ftsl.org.br>

Capítulo 5

Semeando Liberdade

Como (e onde) o software livre inclui as pessoas

Flávio Gomes da Silva Lisboa

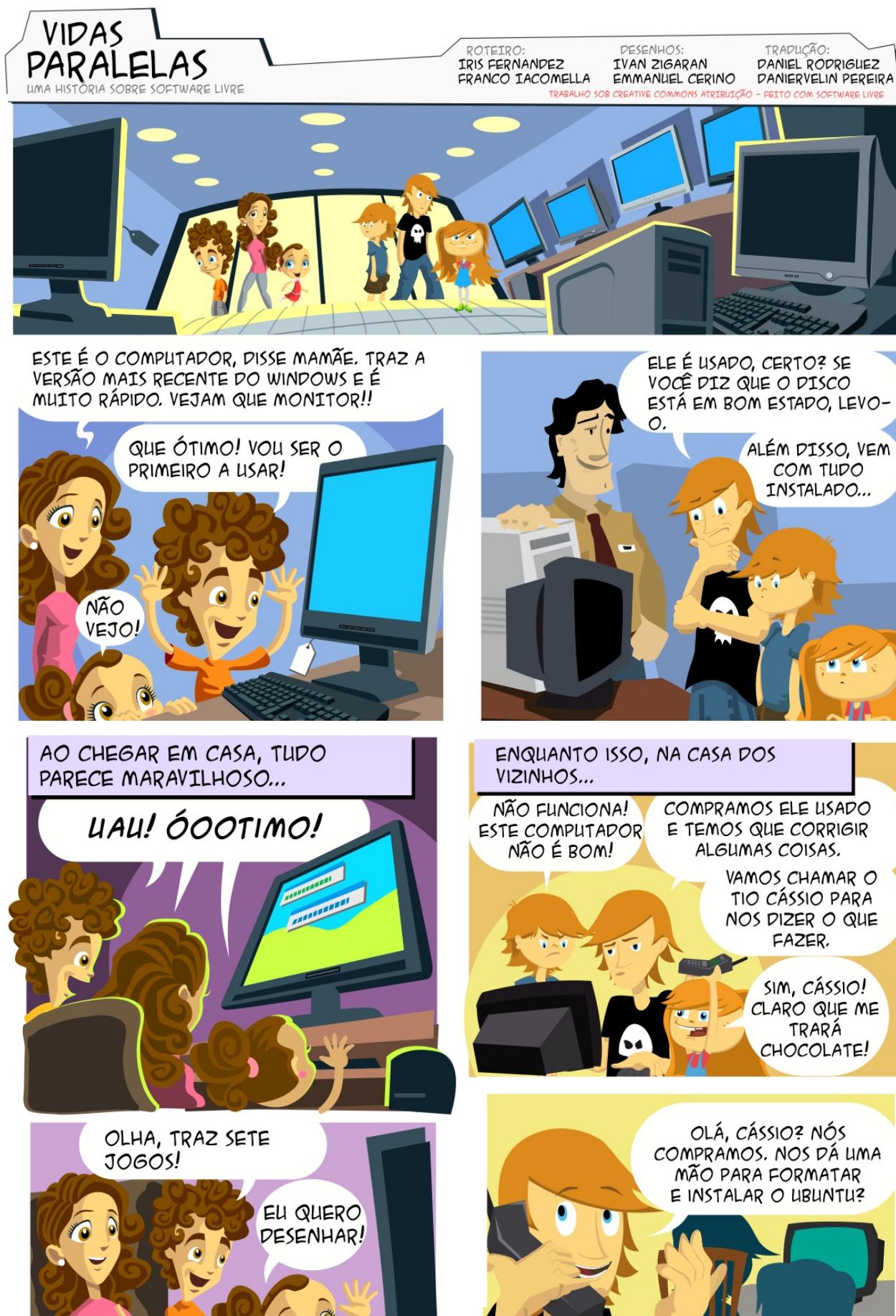
O que é Software Livre

Por “software livre” devemos entender aquele software que respeita a liberdade e senso de comunidade dos usuários. Grosso modo, isso significa que os usuários possuem a liberdade de executar, copiar, distribuir, estudar, mudar e melhorar o software. Assim sendo, “software livre” é uma questão de **liberdade**, não de preço. Um programa é software livre se os usuários possuem as quatro liberdades essenciais:

- A liberdade de **executar** o programa como você desejar, para qualquer propósito (liberdade 0).
- A liberdade de **estudar** como o programa funciona, e adaptá-lo às suas necessidades (liberdade 1). Para tanto, acesso ao código-fonte é um pré-requisito.
- A liberdade de **redistribuir** cópias de modo que você possa ajudar ao próximo (liberdade 2).
- A liberdade de **distribuir** cópias de suas versões modificadas a outros (liberdade 3). Desta forma, você pode dar a toda comunidade a chance de beneficiar de suas mudanças. Para tanto, acesso ao código-fonte é um pré-requisito.

A seguir apresentaremos uma ilustração dos fundamentos do software livre a partir da história de duas famílias que tem experiências diferentes com o uso de computadores. Essa história foi produzida pelo projeto Sembrando Libertad, do ministério de educação da Argentina.







POUCOS DIAS DEPOIS





ESTE MATERIAL PERTENCE AO LIVRO "SOFTWARE LIVRE PARA PEQUENAS PESSOAS" DO PROJETO PLANTIO LIBERDADE

ESTA HISTÓRIA É LIVRE!
VOCÊ PODE COPIAR, TRADUZIR, ADAPTAR E REDISTRIBUÍ-LA COMO QUISER. DISPONÍVEL EM:
WWW.SEMBRANDOLIBERTAD.ORG.AR

O que é Tecnologia Social

Segundo Dagnino (2004, p. 193), **tecnologia social** é (ou deveria ser) a denominação para uma tecnologia:

- Adaptada a pequeno tamanho físico e financeiro;
- Não-discriminatória (patrão-empregado);
- Orientada para o mercado interno de massa;
- Liberadora do potencial e da criatividade do potencial e da criatividade do produtor direto;
- Capaz de viabilizar economicamente os empreendimentos autogestionários e as pequenas empresas.

Para saber mais:

[Programando o Futuro] (<https://www.slideshare.net/programandofturo/tecnologia-social>)

O que é Inclusão Digital

Segundo Mori (2011, p. 40), “as compreensões de ‘inclusão digital’ podem ser aglutinadas em três vertentes:

- a) ‘inclusão digital’ como acesso;
- b) ‘inclusão digital’ como “alfabetização digital”; e
- c) ‘inclusão digital’ como apropriação de tecnologias.”

As três vertentes como verbos:

- Ter
- Usar
- Apropriar-se (dominar, controlar)

Inclusão Digital como Acesso

Segundo Mori (2011, p. 40), “tem como foco a garantia do acesso à infraestrutura de TICs. Uma característica desta abordagem é utilizar como indicador principal de ‘inclusão digital’ a disseminação de bens e serviços relacionados à informática e às telecomunicações.”

Inclusão Digital como Alfabetização Digital

Segundo Mori (2011, p. 40), “a característica principal desta segunda abordagem compreende a infraestrutura tecnológica como algo similar ao lápis e ao papel para quem não é alfabetizado.”

Inclusão Digital como Apropriação de Tecnologias

Segundo Mori (2011, p. 41), “a terceira vertente considera como efetivo objetivo da ‘inclusão digital’ a apropriação das TICs, e não apenas a capacidade de uso básico que a ‘alfabetização digital’ proporciona. Defende que exista não apenas acesso à infraestrutura e ‘alfabetização digital’, mas processos mediante os quais as pessoas sejam capazes de compreender o significado dos meios técnicos e digitais, reinventar seus usos e não se constituir como meros consumidores”.

Exclusão Digital

Segundo Bastos, Kaneko, Napolitano e Reis (2012), “a exclusão digital é uma das muitas formas de manifestação da exclusão social”. Para eles, “ser excluído digitalmente é não ter acesso à informação, conhecimento, opiniões e tecnologias.” Os autores citam uma frase de Pierre Lévy: “Toda nova tecnologia cria seus excluídos”.

Uso da Tecnologia da Informação para Fins Sociais

Inclusão Digital X Inclusão Social

Inclusão social é oferecer oportunidades iguais de acesso a bens e serviços a todos.

Uma das vertentes da inclusão digital é a **garantia do acesso** à infraestrutura de TICs.

Inclusão digital **pode** ser um instrumento para inclusão social.

A Questão

Como (e onde) o software livre inclui as pessoas?

Digitalmente?

Socialmente?

Em qualquer um dos casos, isso é **suficiente**?

Software Livre

“Nós fazemos campanha por essas liberdades porque todo mundo merece. Com essas liberdades, os usuários (tanto individualmente quanto coletivamente) controlam o programa e o que ele faz por eles. Quando os usuários não controlam o programa, o programa controla os usuários. O desenvolvedor controla o programa e, por meio dele, controla os usuários. Esse programa não livre é “proprietário” e, portanto, um instrumento de poder injusto”. (GNU PROJECT)

Soberania Tecnológica

“¿O es que un estado que tenga sus finanzas, salud, educación, ... en manos de corporaciones tecnológicas puede decir que es independiente? ¿o que sus datos están seguros? ¿o que controla su tecnología?”. (RAMÓN)

Referências bibliográficas

BASTOS, Caio Gomide. KANEKO, Flávia Yukimi. NAPOLITANO, Guilherme. REIS, Thiago de Barros. **Inteligência Coletiva e Inclusão Digital**. 2012. Disponível em <https://pt.slideshare.net/thiagodbr/inteligencia-coletiva-e-inclusao-digital-12520004>.

DAGNINO, Renato. A tecnologia social e seus desafios. **Tecnologia social**: uma estratégia para o desenvolvimento. Fundação Banco do Brasil. Rio de Janeiro : 2004

GNU PROJECT. **What is free software?** Disponível em <http://www.gnu.org/philosophy/free-sq.html>. Acesso em 11 out 2017.

MORI, Cristina Kiomi. **Políticas públicas para inclusão digital no Brasil**: aspectos institucionais e efetividade em iniciativas federais de disseminação de telecentros no período 2000-2010. Tese (Doutorado em Política Social) – Instituto de Ciências Humanas. Universidade de Brasília, 2011.

RAMÓN, Ramón. **Soberania Nacional e Independencia Tecnológica**: Soberanía Tecnológica. Disponível em <https://pt.slideshare.net/ramonramonsa/soberania-tecnologica-fisl14>

Capítulo 6

Programando Hardware: Utilizando Recursos GPIO do Raspberry Pi

Fabiano Sardenberg Kuss

Introdução

A eletrônica digital integrada a circuitos micro-controlados ou microprocessados permitiu uma nova forma de criação de componentes e dispositivos especialistas. Com uma grande gama de componentes implementados em circuitos integrados complexos, interfaces de comunicação, sensores diversos embarcadas em placas e popularização de motores de passo e servo motores é possível criar dispositivos capazes de atender a quase todas as ideias de automação sem custos elevados ou necessidade de equipamentos para apoio ao desenvolvimento dos circuitos.

A integração entre dispositivos eletrônicos e interface GPIO é uma opção de deve ser investigada com muito critério pois na maioria das vezes é possível utilizar placas muito mais baratas e menos complexas, tanto do ponto de vista do software necessário para o processamento quanto do próprio hardware SBC, na implementação de automação de sistemas. Vale destacar que apenas sinais digitais podem ser processados por este tipo de interface, o que nem sempre é adequado a sensores que dependem de precisão a partir de dados analógicos.

GPIO

A sigla GPIO significa General-purpose input/output, que pode ser traduzido como entrada e saída de propósito geral, representando um conjunto de pinos capazes de receberem e enviarem sinais para uma funcionalidade genérica. Cada dispositivo pode implementar a pinagem da forma que for mais conveniente para o circuito, inexistindo um padrão de barramento para este tipo de interface, mas a implementação do Raspberry Pi é visto como uma referência para SBCs e neste trabalho a utilizaremos como referência.

Compreendendo a pinagem

Apesar dos SoCs Broadcom oferecerem de 54 portas para interfaces de entrada e saída de propósito geral mas o Raspberry Pi oferece um conjunto de 26 pinos nos modelos A+ e B+, PI 2 e P3 para serem utilizado em interfaces por meio de sinais digitais. Na interface do Raspberry são expostas ao usuário 40 pinos para a integração com dispositivos com a seguinte configuração:

- 8 pinos para negativo (ground)
- 2 pinos de 5V
- 2 pinos de 3V
- 2 pinos para I2C
- 26 pinos para uso geral (GPIO)

Os pinos GPIO são identificados por números que variam de 1 à 26, sem uma organização lógica da disposição dos mesmos. Deve ser evitada a utilização da alimentação da placa em dispositivos que tenham consumo de corrente superior a 500mA. A tensão utilizada nos pinos GPIO é de 3,3V o que pode ser um problema na interface com shield Arduino ou componente eletrônicos que operem com tensão de 5v como é o caso da tecnologia TTL, assim sempre que possível deve-se optar por circuitos integrados CMOS para interação com pinos GPIO do Raspberry Pi.

A interação entre o sistema operacional e o barramento é feito por meio de acesso em endereços reservados de memória, no Linux representado por /dev/mem e acionamento de interrupção. O seguinte fragmento de código exemplifica a interação entre um programa e o envio de sinais digitais (tensão) aos pinos:

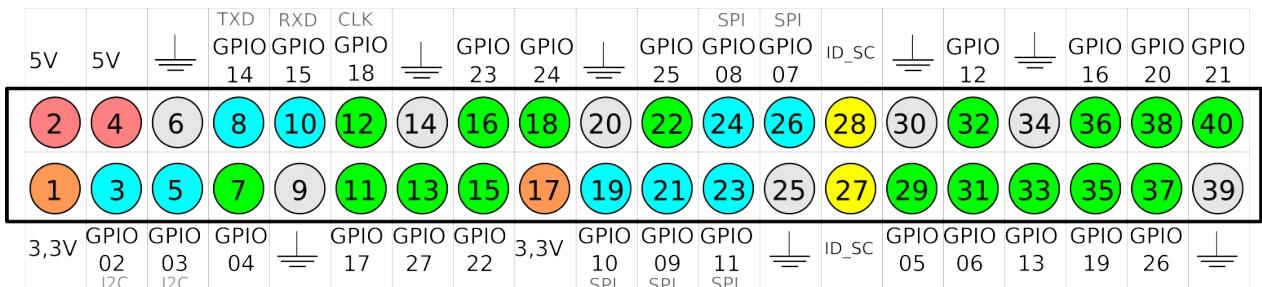


Figura 1: Barramento GPIO

```

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <errno.h>
#include <stdint.h>

static volatile uint32_t * gpio ;

int main ( int argc , char ** argv ) {
    int fd ;
    if ((fd = open ( "/dev/mem" , O_RDWR | O_SYNC ) ) < 0) {
        printf (" Nao foi possivel acessar /dev/mem: %s\n " , strerror(errno)) ;
        return -1;
    }
    gpio = ( uint32_t *)mmap(0, getpagesize(), PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x20200000) ;

    if ((int32_t) gpio < 0) {
        printf("Falha ao acessar: %s\n" , strerror (errno)) ;
        return -1;
    }

    * (gpio + 1) = (*(gpio + 1) & (7 << 21)) | ( 1 << 21) ;
}

```

Programando em Python

Existe suporte para acesso ao barramento GPIO em várias linguagens de programação, em C existe uma biblioteca que permite o desenvolvimento de aplicações de forma semelhante a interface do Arduino utilizando wiringPi.h, mas Python é provavelmente a linguagem mais utilizada para acesso as portas GPIO. As instalações Linux para Raspberry contem as bibliotecas necessárias para o desenvolvimento de programas com interação com outros dispositivos por meio de portas lógicas.

Mesmo sem conhecimentos na linguagem o programador pode rapidamente adquirir as habilidades necessárias para a criação de aplicações que não demandem implementação de algoritmos mais complexos. Foge ao escopo deste trabalho a proposta de ensinar a programar em Python, mas os exemplos utilizados podem ser implementados por pessoas que nunca tiveram contato com a linguagem. No entanto para o melhor aproveitamento do conteúdo serão apresentadas algumas informações básicas e uso do interpretador.

O escopo de uma classe, função, método, condicional ou controle de laço em Python é delimitado por meio de indentação. Funcionalidades podem ser importadas pela inserção de bibliotecas ou scripts utilizando a palavra reservada import. O interpretador Python pode ser acionado basicamente pela chamada de um console interativo ou pela interpretação de um script.

No console do sistema operacional execute o comando python, será exibida uma interface com um prompt onde é possível a inserção de comandos a serem interpretados cada vez que a tecla enter é pressionada ao final de uma instrução completa. Neste trabalho serão apresentadas apenas as estruturas de controle if, for e a manipulação de listas simples. Uma aplicação Oi Mundo pode ser criada conforme o seguinte fragmento de código, utilizando o console interativo:

Utilizando variáveis:

```
oi = "Oi"
mundo = "Mundo"
print oi, mundo
```

Interando com o comando for:

```
oi = "Oi Mundo"
for i in range (0, 10) :
    print oi, i
```

Criando um script para ser utilizado pelo interpretador é possível realizar a passagem de parâmetros por meio da importação da biblioteca sys

```
import sys
print sys.argv [ 1 ]
```

Para executar o script basta executar o interpretador passando o nome do script como primeiro parâmetro e algum valor, numérico ou texto com segundo parâmetro

```
python nome_do_script Oi
```

Python GPIO

A biblioteca RPi.GPIO deve ser importada no inicio do programa para importar as funcionalidades de acesso aos pinos, isto é feito por meio da diretiva import. Existem duas formas de endereçamento de pinos com a biblioteca, utilizando a nomenclatura do Broadcom ou a numeração de contagem dos pinos, esta definição é feita por meio da função setmode e as constantes BCM (Broadcom) e BOARD (sequencial), o seguinte fragmento de código define a contagem pelo posição do pino, para alterar basta substituir GPIO.setmode(GPIO.BCM) por GPIO.setmode(GPIO.BOARD)

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
```

O próximo passo é definir o comportamento da pino, se será utilizado para entrada ou saída de dados, por meio da função setup e das constantes OUT ou IN passados como segundo parâmetro. Sempre que o programa é finalizado é importante restaurar os valores padrão de tensão dos pinos por meio da função cleanup(), caso isso não seja feito os pinos podem ter comportamento inadequado na inicialização e será exibido um alerta informando que a podem ocorrer instabilidades no acesso as portas.

```
GPIO.setup(PIN NUMBER 1, GPIO.OUT)
GPIO.setup(PIN NUMBER 2, GPIO.IN)
GPIO.cleanup()
```

Uma aplicação básica para acender um led ligado no pino

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)
GPIO.output(7, True)
GPIO.cleanup()
```

Equipamentos e componentes necessários

Antes de iniciar a implementação serão feitos breves comentários sobre alguns componentes que são úteis na prototipação de projetos que envolvam sensores e dispositivos integrados ao SBC. Não se pretende aqui nenhum aprofundamento nas questões físicas e eletrônicas relacionadas aos componentes apresentados, apenas fornecer informações básicas sobre a funcionalidade de cada um dos produtos que serão utilizados em neste trabalho.

Cabos

Para fazer a interface entre os pinos e os componentes que receberão o impulso digital é necessário o uso de cabos ou adaptadores para a passagem da corrente elétrica. Como a pinagem do Raspberry utiliza conectores macho uma das pontas do cabo de conexão deverá ser capaz de conectar-se a esta. É possível fazer cabos utilizando fios rígidos mas é mais simples adquirir produtos prontos que estão disponíveis em muitas das lojas de equipamentos eletrônicos ou em sites de venda on line.

Protoboard

Uma protoboard é sempre bem vinda quando se está trabalhando com componentes eletrônicos e no caso de interação entre as portas GPIO do SBC e componentes a fase de protipação quase sempre vai exigir o uso deste tipo de equipamento. O uso de protobards é relativamente intuitivo mas eventualmente o leitor pode não estar familiarizado com este recurso, então esta seção fará uma breve apresentação sobre como instalar energizar, colocar componentes e ligar a fiação. Nem todas as protobards dispõe de barramento para corrente compartilhada ou divisão em setores, mas para quem deseja se aventurar no desenvolvimento de sistemas utilizando interfaces de sinais analógicos é recomendável a aquisição de uma protobord com estas funcionalidades.

Led

Led é uma sigla para *light-emitting diode*, são diodos que produzem um efeito de eletroluminescência quando energizados por uma tensão elétrica. Como um diodo este componente permite passagem de corrente apenas em uma direção logo o sentido da corrente é importante para o funcionamento do led. Este tipo de componente é comercializados com duas pernas de tamanhos diferentes onde a mais longa é o cátodo (+) e a maior anodo(-). Aplicações do tipo oi mundo em circuitos embarcados costuma ser representadas por piscar de leds o que torna este tipo de componente algo básico para programadores que desenvolvem aplicações que utilizam interfaces do tipo GPIO.

Criando uma aplicação IoT

Fazendo um led piscar

A primeira aplicação integrando o barramento GPIO com dispositivos eletrônicos será um projeto para acender e apagar um led de forma intermitente em intervalos de tempo fixo. Para isso inicialmente é necessário que se construa um circuito conforme apresentado na figura a seguir. Neste exemplo foram utilizados os pinos 3 (negativo) e 4 (sinal digital) do barramento do Raspberry. O polo positivo que fará o led acender é o sinal digital de 3,3V enviado no pino de dados (4)

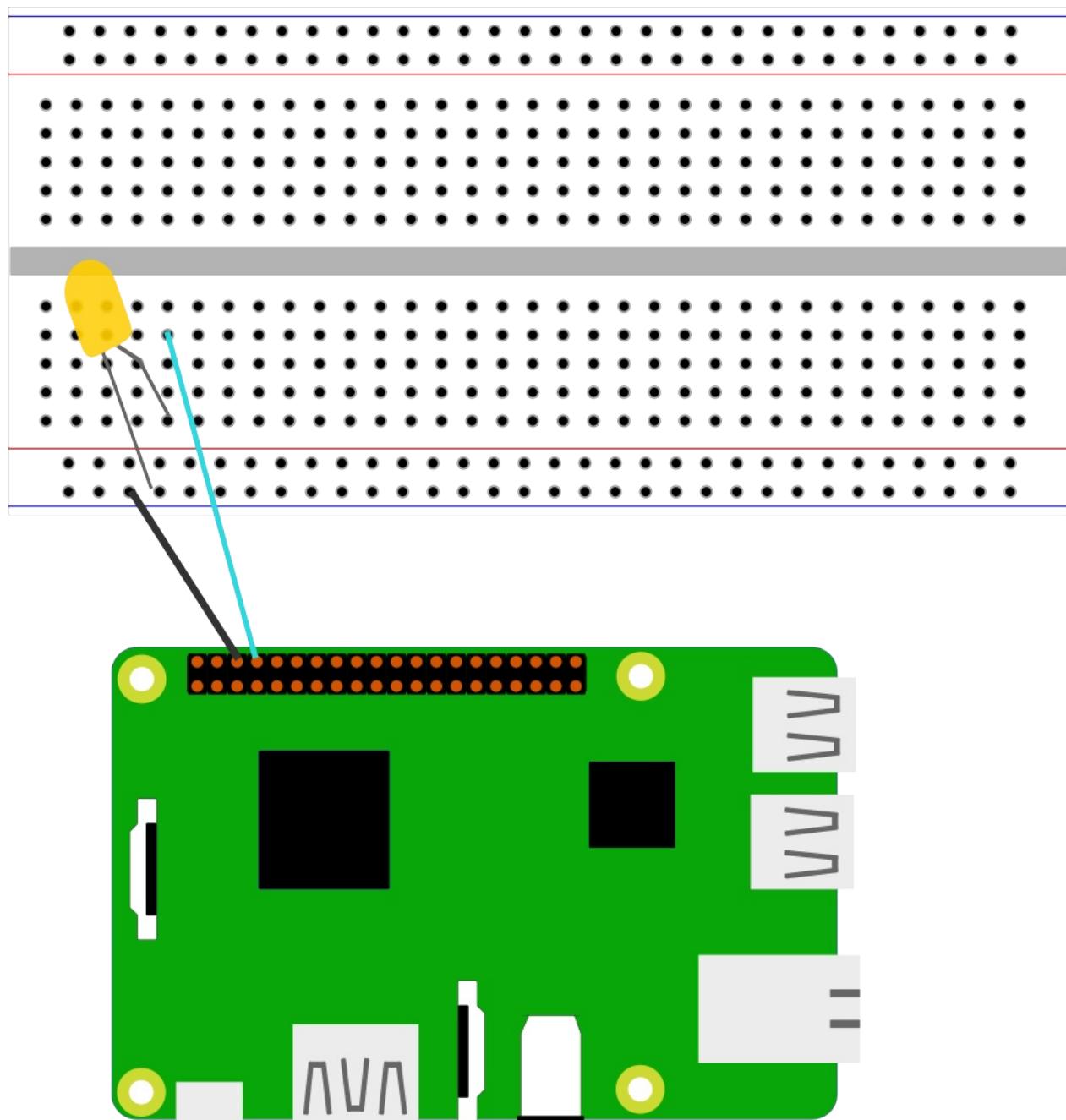


Figura 2: Esquema da ligação eletrônica

O código utilizado para controlar os sinais enviados utiliza a biblioteca time para produzir uma pausa em intervalo de tempo fixo onde o led permanecerá ligado e o tempo em que o mesmo ficará desligado, ou seja o pino 4 estará ou não recebendo energia. O sistema permanecerá ativo até que o usuário cancele a execução do programa pressionando as teclas Ctrl e C (Ctrl+c) simultaneamente. Para garantir a execução da função cleanup() o tratamento de exceções é uma estratégia que pode ser aplicada, sua sintaxe demanda pelo uso de um bloco de código utilizando as palavras reservadas try e except como apresentada no seguinte segmento de código:

```

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)

try:
    while True:
        GPIO.output(7, True)
        time.sleep(1)
        GPIO.output(7, False)
        time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()

```

Recebendo sinais

Ser capaz de receber sinais como entrada é uma tarefa essencial para vários projetos envolvendo controle de dispositivos ou interação com o sistema operacional. A implementação deste tipo de controle utilizando o barramento GPIO integrado ao sistema operacional permite explorar as capacidades do sistema operacional utilizando um computador relativamente potente para uma série de projetos. A leitura de um sinal em determinada porta é semelhante ao controle de interrupções em sistemas operacionais, ou seja, verifica-se se ocorreu alteração de estado de sinal e em caso positivo é acionada uma rotina para tratamento deste evento. No exemplo será utilizada a estratégia de interrupção e continuidade de sinal por meio de cabos mas o cenário ideal seria o uso de um botão do tipo push.

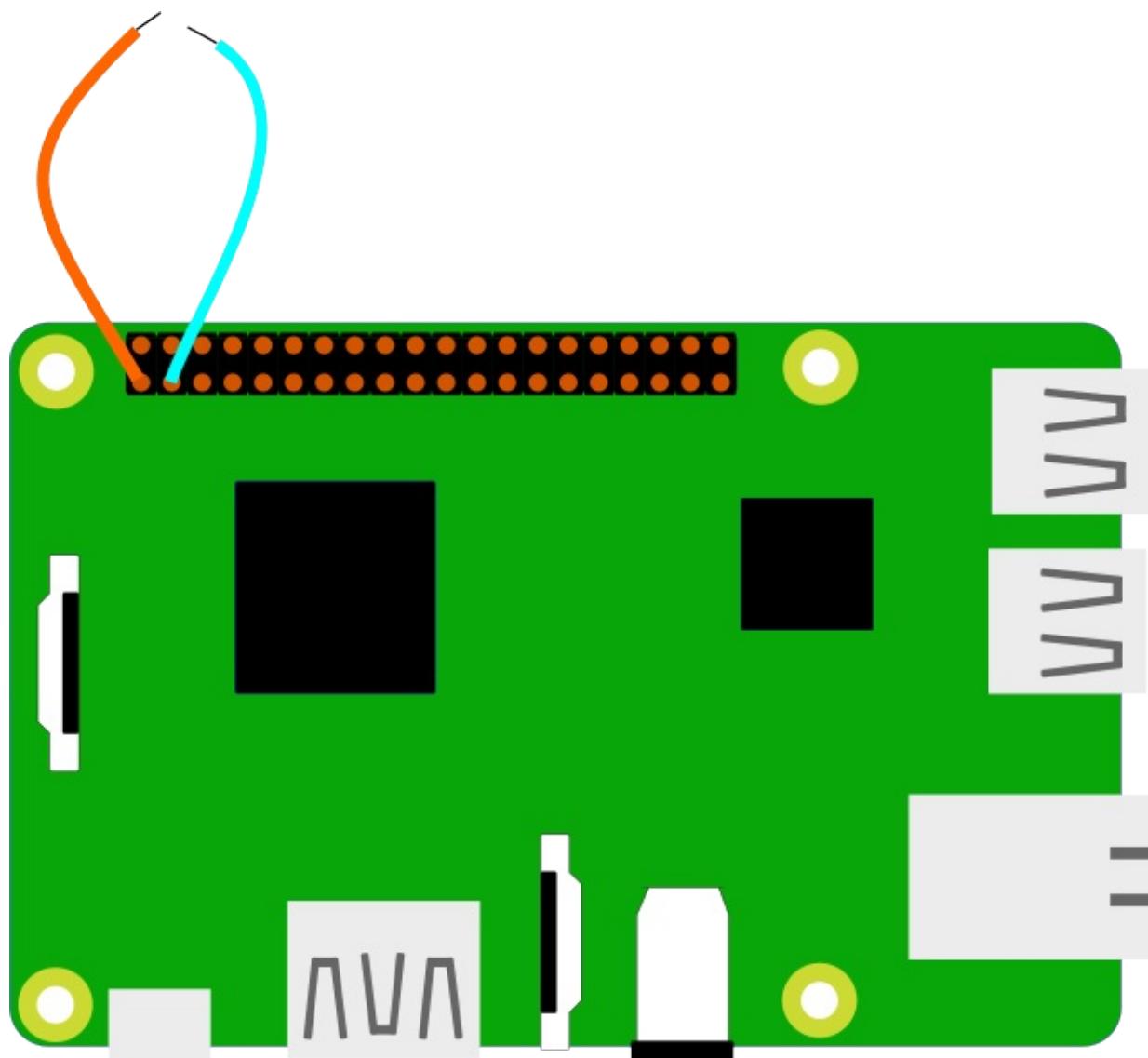


Figura 3: Esquema da ligação eletrônica para entrada de sinal

O exemplo a seguir guarda a posição do pino que será utilizado para controle em uma variável, para aplicações que envolvam várias portas é mais simples ter uma representação significativa sobre o número do pino do que lembrar a funcionalidade implementada baseado no número da porta. Outra diferença importante em relação aos códigos anteriores é uso função sleep() com um número decimal. O valor inteiro para a função sleep() representa intervalos de tempo utilizando segundos mas muitas vezes este valor é muito grande para a implementação então utiliza-se valores decimais que podem representar milissegundos ou microssegundos, no exemplo o valor 0.1 representa intervalo de tempo de 100ms ou 100s/1000s.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
BOTAO = 2
GPIO.setup(BOTAO, GPIO.IN)
try :
    while True :
        status = GPIO.input(BOTAO,True)
        if status == True :
            print "Ligado"
        time.sleep(0.1)
except KeyboardInterrupt :
    GPIO.cleanup ()
```

Integrando com a rede

Os exemplos anteriores poderiam ser facilmente implementados em arquiteturas muito mais simples do que um *single board computer* como por exemplo uma plataforma Arduino, a integração com redes também pode ser feita com placas menores como a NodeMCU ou com dispositivos específicos implementados para atuar com circuitos micro-controlados. A diferença do uso de um dispositivo como o Raspberry Pi é a capacidade de integração com serviços mais sofisticados e capacidade de processamento que permite a exposição de serviços sem a dependência de infra-8 estrutura provida por um computador tradicional mantendo, por exemplo, um banco de dados local.

O código utilizado no exemplo é um pouco mais complexo que os demais pois exige algum conhecimento de programação orientada a objetos e sobre socket. Basicamente o programa informa ao sistema operacional que está aceitando conexões TCP/IP na porta 8080. Ao chegar alguma conexão direcionada a esta porta está será encaminhada ao programa que fará a leitura do conteúdo utilizando o método recv e interpretará o conteúdo HTTP recuperando os valores passados na linha com a informação GET. A configuração do circuito é a mesma apresentada na figura 1.

```

import SimpleHTTPServer
import Socket Server
#Import RPi.GPIO as GPIO

PORT = 8000

class RPiHandler(SocketServer.BaseRequestHandler) :
    def handle (self) :
        self.data = self.request.recv(1024).strip( )
        if self.data[:6] == "GET / ? " :
            param = self.data[6:].split()[0]
            param = param.split("&")
            led = param[0]
            status = param[1]
            if status == "on " :
                print "Liga"
                #status = GPIO.input(led, True)
            else:
                print "Desliga "
                #status = GPIO.input(led, False)
            self.request.sendall(self.data.upper( ) )

#GPIO.setmode(GPIO.BOARD)

httpd = SocketServer.TCPServer((" ", PORT), RPiHandler)

print "Servidor na porta", PORT

try :
    httpd.serve_forever()
except KeyboardInterrupt :
    print "FIM"
    #GPIO.cleanup()

```

Considerações Finais

O uso dos pinos GPIO do Raspberry Pi é uma importante ferramenta na construção de sistemas que tenham interação com dispositivos eletrônicos permitindo processos de automação e controle de maneira simples aproveitando-se do poder de processadores multi-core e as facilidades providas por um sistema operacional robusto com milhares de sistemas aplicativos portados para a plataforma ARM.0.6.

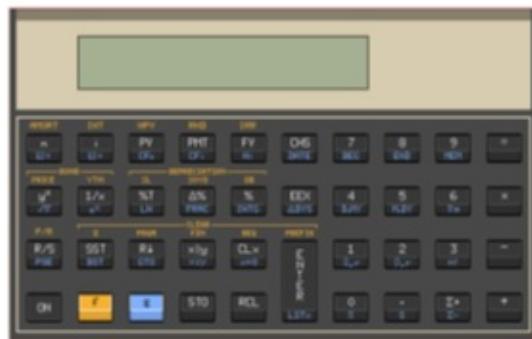
O uso da linguagem Python para o desenvolvimento de aplicações permite agilidade no processo de implementação e facilidade de aprendizagem. Certamente maior qualidade no produto desenvolvido demanda maior domínio da linguagem mas é possível que mesmo sem qualquer conhecimento em Python o programador consiga dar passos significativos na construção de sistemas de interação por meio do envio de sinais digitais.

Apesar de utilizar exemplos simples como acender leds o conjunto de informações apresentadas neste trabalho permite que o leitor utilize estes conhecimentos para a produção de outros sistemas sem a necessidade de muita informação adicional. O fato de saber como enviar e receber informações em uma determinada porta é suficiente para acionamento de dispositivos como motores, painéis, relês, shields, etc.

Capítulo 7

Usando o R como Calculadora Financeira

Jonas Liebl



USANDO O COMO CALCULADORA FINANCEIRA

O que é o Software R

R é um software livre para computação estatística e gráficos. Além de ser amplamente utilizado para cálculos estatísticos no meio acadêmico, vem ganhando notoriedade em mineração de dados, tornando-se uma das ferramentas mais usadas por profissionais conhecidos como “cientistas de dados”. A principal página na Internet através da qual se pode baixar o software e iniciar o seu aprendizado é a <https://www.r-project.org>.

Qual foi a proposta do minicurso ministrado no FTSL 2017

O software R se destaca como uma ferramenta estatística e esse estigma já é suficiente para desestimular o seu uso por profissionais de outras áreas. Ocorre que os seus comandos e a elevada capacidade de processamento podem facilitar a execução de atividades completamente distintas do segmento estatístico, sendo úteis inclusive para uso no ambiente doméstico de uma família. Uma das possibilidades que se mostraram excelentes é no campo da matemática financeira, em especial com sistemas de amortização e planos de capitalização. Além de realizar os cálculos, pode-se emitir planilhas de evolução, também conhecidas como “extratos”, e gravar arquivos muito úteis para inserção como anexos em contratos, por exemplo. Isso tudo é possível combinando-se funções e sintaxes de forma personalizada e gravando scripts para serem executados pelo R sempre que for necessário.

O que é um sistema de amortização

Sistema de amortização é um método de cálculo que tem por objetivo pagar uma dívida ou financiamento no prazo contratado, remunerando-o à taxa pactuada. O mais conhecido é o “Sistema Francês de Amortização”, também chamado de “Tabela Price”. É esse sistema que se encontra residente na famosa calculadora financeira HP 12C. A sua principal característica é que apresenta prestações (ou parcelas) iguais. Alguns conceitos importantes:

PV (present value): valor presente, que corresponde ao valor da dívida ou do financiamento que deverá ser quitado no prazo contratado.

PMT (payment): pagamento, que é o valor da prestação a ser paga em cada período para amortizar a dívida ou financiamento e remunerá-lo pelos juros pactuados.

i (interest): taxa percentual de juros destinada a remunerar a dívida ou financiamento. Deve ser expressa na mesma periodicidade das prestações, ou seja, se forem exigidas prestações mensais, a taxa percentual de juros também deverá ser equivalente a mês.

n (número de períodos): quantidade de prestações para amortizar a dívida ou financiamento.

modo antecipado: ocorre quando a primeira prestação é paga no momento em que a dívida é contraída. É muito comum no comércio, quando se adquire determinado bem e a primeira prestação é dada como entrada no ato da compra.

modo postecipado: a prestação é paga no final do primeiro período, ou seja, se o prazo do financiamento for mensal, esse pagamento acontece um mês após a contratação e os demais, sequencialmente, a cada mês.

modo diferido: o início do pagamento das prestações ocorre após um período de carência. Por exemplo, uma compra no comércio realizada em novembro com primeira prestação a ser paga somente após o carnaval.

Usando o R para cálculos relativos ao Sistema Francês de Amortização

Foram desenvolvidas algumas funcionalidades, consideradas as mais importantes, e gravadas num script denominado CALCULADORA FINANCEIRA.R. Sempre que precisar fazer algum cálculo desse tipo, basta abrir o R e executar Arquivo / Interpretar código fonte R, localizando o script já salvo no computador. O console do R apresentará as orientações básicas, as quais poderão ser recuperadas a qualquer momento através da função help(), ou, mais especificamente quanto ao "Sistema Francês de Amortização", a help.sfa().

Calculando uma prestação

Determinada pessoa pretende obter um empréstimo de R\$ 5.000,00 a ser quitado em 6 prestações mensais de igual valor, a primeira ocorrendo um mês após a contratação. Considerando que a taxa de juros é de 2% ao mês, vamos calcular o valor da prestação. Trata-se de um exemplo típico de cálculo pelo "Sistema Francês de Amortização" no modo postecipado. Conforme orientações fornecidas pelo script, deve-se comandar sfa() informando, entre os parênteses e entre vírgulas, o valor presente (valor do empréstimo), a taxa percentual mensal de juros, a quantidade de prestações mensais e o parâmetro de modo que, no caso do postecipado, é 0 (zero).

```
> sfa(5000, 2, 6, 0)
[1] "***** VALOR DA PRESTAÇÃO: 892.63 *****"
[1] PERÍODO PRESTAÇÃO JUROS AMORTIZAÇÃO SALDO
[1,] 0 0.00 0.00 0.00 5000.00
[2,] 1 892.63 100.00 792.63 4207.37
[3,] 2 892.63 84.15 808.48 3398.89
[4,] 3 892.63 67.98 824.65 2574.24
[5,] 4 892.63 51.48 841.15 1733.09
[6,] 5 892.63 34.66 857.97 875.12
[7,] 6 892.63 17.50 875.13 -0.01
>
```

Note que o script do R não apenas calcula o valor da prestação, como também demonstra a evolução da dívida mês a mês. O saldo final negativo de R\$ 0.01 ocorreu porque os cálculos foram arredondados para duas casas decimais. Sem esse arredondamento, o saldo seria zerado. Deve-se observar que o R adota o ponto como separador de decimais, que é a convenção utilizada em países de cultura inglesa.

Outro benefício desse script é que o R gravou um arquivo denominado “EXTRATO.CSV” na mesma pasta que está definida para realizar os processamentos. Esse arquivo poderá ser aberto e formatado para inserção como anexo em contratos, para demonstrar a evolução da dívida.

Calculando uma taxa de juros

Trata-se de um cálculo muito útil como subsídio para uma decisão. Por exemplo, uma pessoa pretende comprar uma geladeira de R\$ 1.500,00 e a loja oferece duas opções de pagamento. A primeira é dividindo o valor em 3 parcelas mensais “sem juros”, com a primeira sendo paga no ato da compra. A segunda é o pagamento total à vista com um desconto de 10%. Como o comprador possui esse valor num investimento remunerado a 5% ao mês, deseja saber se vale a pena abrir mão do investimento para obter o desconto.

Intuitivamente, algumas pessoas dividem o 10% do desconto pelo número de parcelas e imaginam que a taxa mensal de juros embutida na operação é de 3,33%. Esse cálculo é equivocado porque não considera a antecipação do primeiro pagamento, nem a amortização mensal das parcelas subsequentes. No cálculo financeiro, devemos informar como valor presente aquele que seria pago na opção à vista: R\$ 1.500,00 menos 10% de desconto, que é igual a R\$ 1.350,00. As prestações são aquelas da opção a prazo, ou seja, R\$ 1.500,00 dividido por 3, que corresponde a R\$ 500,00. Finalmente, como o modo é antecipado, o parâmetro a ser utilizado é de -1 (menos um). Essas informações serão comandadas em **isfa()**.

```
> isfa(1350,500,3,-1)
[1] "Quantidade de cálculos: 1002 "
[1] "*****"
[1] " TAXA ESTIMADA: 11.5544 % "
[1] "*****"
[1] " VALOR DA PRESTAÇÃO: 500 "
[1] "*****"
      PERÍODO PRESTAÇÃO JUROS AMORTIZAÇÃO SALDO
[1,]      0        0  0.00      0.00 1350.00
[2,]      0      500  0.00    500.00  850.00
[3,]      1      500 98.21   401.79  448.21
[4,]      2      500 51.79   448.21      0.00
>
```

No presente caso, perder o desconto que seria obtido na compra à vista para pagar a geladeira em 3 vezes “sem juros” representa uma taxa interna de juros na ordem de 11,5544% ao mês. Como a remuneração mensal do investimento é de 5%, ou seja, inferior a essa taxa interna de juros, a melhor opção é a de resgatar os R\$ 1.350,00 do investimento e adquirir a geladeira à vista.

Não existe uma fórmula para calcular a taxa de juros de maneira direta. O seu cálculo é obtido por um processo iterativo. Note que o R informou que realizou 1002 cálculos para chegar ao resultado com quatro casas decimais, mas nós nem notamos isso porque ocorreu em fração de segundo. Se necessitarmos de mais casas decimais, precisaremos editar o script para aumentar o processo iterativo, mas isso só vale a pena quando os valores envolvidos são muito grandes.

Calculando o valor presente

É usual os bancos estipularem um limite de crédito pela capacidade de pagamento em função da renda mensal do tomador. Supondo que essa capacidade está delimitada a 30% da renda de R\$ 5.000,00 de uma pessoa, a prestação total mensal não pode passar de R\$ 1.500,00. Considerando ainda a existência de encargos acessórios no valor de R\$ 100,00 a título de seguro e taxa de administração, a prestação mensal, destinada a amortização e juros, fica limitada a R\$ 1.400,00. Vamos calcular qual seria o valor do empréstimo nessas condições, a ser resgatado em 6 prestações mensais iguais com 3 meses de carência, sendo a taxa de juros de 2,5% ao mês. As informações serão comandadas em **psfa()**.

```
> psfa(1400, 2.5, 6, 3)
[1] "*****"
[1] " VALOR PRESENTE: 7160.78 "
[1] "*****"
[1] " VALOR DA PRESTAÇÃO: 1400 "
[1] "*****"
[1]          PERÍODO  PRESTAÇÃO    JUROS AMORTIZAÇÃO     SALDO
[1,]        0        0   0.00      0.00 7160.78
[2,]        1        0 179.02   -179.02 7339.80
[3,]        2        0 183.50   -183.50 7523.30
[4,]        3        0 188.08   -188.08 7711.38
[5,]        4       1400 192.78  1207.22 6504.16
[6,]        5       1400 162.60  1237.40 5266.76
[7,]        6       1400 131.67  1268.33 3998.43
[8,]        7       1400  99.96  1300.04 2698.39
[9,]        8       1400  67.46  1332.54 1365.85
[10,]       9       1400  34.15  1365.85  0.00
>
```

O valor do empréstimo possível considerando a capacidade de pagamento e as regras do negócio é de R\$ 7.160,78. Como se trata de um empréstimo com carência, durante os três primeiros meses só há incorporação de juros. O pagamento das prestações começa no quarto mês e a sua quitação ocorre no nono.

O que é um plano de capitalização

Plano de capitalização é um método de cálculo que tem por objetivo formar um capital ou obter um montante, a partir de depósitos periódicos consecutivos remunerados a uma determinada taxa. O melhor exemplo de capitalização é um plano de previdência em que, durante muitos anos, é feito um depósito mensal de determinado valor com o objetivo de, a partir de determinada data, receber um benefício mensal a título de aposentadoria.

Usando o R para cálculos relativos a planos de capitalização

Também foram desenvolvidas algumas funcionalidades, tidas como as mais importantes, e gravadas nesse script denominado CALCULADORA FINANCEIRA.R. O console do R apresentará as orientações básicas, as quais poderão ser recuperadas a qualquer momento através da função **help()**, ou, mais especificamente quanto a capitalizações e montantes, a **help.cap()**.

Calculando o valor do depósito mensal

Vamos supor que uma pessoa deseja comprar um notebook de R\$ 3.000,00 daqui a 6 meses e que, para isso, pretende economizar uma parcela fixa do seu salário depositando-a mensalmente numa aplicação que rende 2% ao mês. Faremos uso do comando `cap()`, informando, entre os parênteses e entre vírgulas, o valor do montante desejado, a taxa percentual mensal de juros e a quantidade de depósitos mensais.

```
> cap(3000, 2, 6)
    PERÍODO DEPÓSITO JUROS MONTANTE
[1,]      1 475.58  0.00  475.58
[2,]      2 475.58  9.51  960.67
[3,]      3 475.58 19.21 1455.46
[4,]      4 475.58 29.11 1960.15
[5,]      5 475.58 39.20 2474.93
[6,]      6 475.58 49.50 3000.01
>
```

O valor a ser economizado e depositado mensalmente é de R\$ 475,58. Ao realizar o sexto depósito e computando os juros calculados sobre as parcelas precedentes, o valor de R\$ 3.000,00 estará disponível para adquirir o pretendido notebook.

Calculando a quantidade de depósitos mensais

Imaginando que essa mesma pessoa tenha condições de economizar um pouco mais, elevando o depósito mensal para R\$ 700,00, vamos calcular em quantos meses terá o montante disponível para a compra desejada. O comando a ser utilizado é o `ncap()`.

```
> ncap(3000, 700, 2)
    PERÍODO DEPÓSITO JUROS MONTANTE
[1,]      1 700  0.00  700.00
[2,]      2 700 14.00 1414.00
[3,]      3 700 28.28 2142.28
[4,]      4 700 42.85 2885.13
[5,]      5 700 57.70 3642.83
>
```

Cabe esclarecer que o desenvolvimento do script não prevê períodos fracionários. No exemplo em pauta, são depósitos mensais de igual valor efetuados sempre a cada mês. Percebemos que, no quarto mês, o valor está bem próximo do montante desejado. Com o quinto depósito, o montante já é mais do que suficiente para adquirir o notebook. Se o comprador deseja juntar exatamente os R\$ 3.000,00, poderá ampliar o quarto depósito para o valor de R\$ 814,87 ($3.000,00 - 2.885,13 + 700,00$) ou reduzir o quinto para R\$ 57,17 ($700,00 - 642,83$). Assim, o montante exato será atingido com a parcela 4 ou 5, respectivamente.

Calculando o montante

Essa funcionalidade tem o objetivo de calcular o montante obtido com os depósitos periódicos de igual valor e inclui a possibilidade de interromper a fase de depósitos, continuando apenas com a remuneração de juros sobre os saldos precedentes.

Aproveitando o exemplo anterior de depósitos de R\$ 700,00, vamos interromper a sequência após a quarta parcela e deixar o saldo sendo remunerado por mais 6 meses. O comando é o `mont()`, cujas orientações constam no `help.cap()`.

```
> mont(700,2,4,6)
    PERÍODO DEPÓSITO JUROS MONTANTE
[1,]      1     700  0.00   700.00
[2,]      2     700 14.00  1414.00
[3,]      3     700 28.28  2142.28
[4,]      4     700 42.85  2885.13
[5,]      5       0 57.70  2942.83
[6,]      6       0 58.86  3001.69
[7,]      7       0 60.03  3061.72
[8,]      8       0 61.23  3122.95
[9,]      9       0 62.46  3185.41
[10,]     10      0 63.71  3249.12
>
```

O montante parcial obtido no momento do quarto depósito continuou sendo remunerado a 2% ao mês por mais 6 meses, elevando o valor de R\$ 2.885,13 para R\$ 3.249,12. Percebe-se também que o valor necessário para a compra do notebook seria atingido dois meses após a interrupção dos depósitos.

Considerações finais

O script **CALCULADORA FINANCEIRA.R** poderá ser baixado e usado tanto para os cálculos a que se propõe, quanto para conhecimento das sintaxes utilizadas para produzir esses resultados. Evidentemente, trata-se apenas de uma pequena incursão no mundo da matemática financeira para demonstrar que o R não está restrito à Estatística. As potencialidades do software podem nos ajudar em outras atividades profissionais ou acadêmicas. Como diria Goethe:



Imagen: Goethe em 1828, óleo sobre tela de Stieler.

script CALCULADORA FINANCEIRA.R

O script para efetuar os cálculos mencionados neste capítulo pode ser obtido através do link:

http://www.angelfire.com/un/cde/CALCULADORA_FINANCEIRA.R

A inserção no R se dá através do comando **Arquivo / Novo script**, o qual abre uma janela de edição possibilitando colar o conteúdo previamente copiado da página do link. Na sequência, comanda-se **Arquivo / Salvar como**, selecionando-se a pasta onde se deseja salvá-lo e atribuindo-se o nome de CALCULADORA FINANCEIRA. A extensão R será assumida automaticamente. O comando **Arquivo / Fechar script** retorna ao console.

A execução do script se dá através do comando **Arquivo / Interpretar código fonte R**, selecionando-se a CALCULADORA FINANCEIRA.R. O console mostrará as orientações contidas nas ajudas, as quais poderão ser resgatadas a qualquer tempo por `help()`, `help.sfa()` e `help.cap()`.

Uma questão importante que merece a nossa atenção é a informação que aparece no console assim que ele é aberto: o "R é um software livre e vem sem GARANTIA ALGUMA". Obviamente, essa garantia não existe porque o R é fruto de um ambiente colaborativo e é totalmente gratuito para o usuário. A hipótese de eventual bug não pode ser descartada,

entretanto, dependendo do aplicativo que está sendo processado, é possível estabelecer um controle de qualidade que assegura total confiabilidade. É o caso dos sistemas de amortização. Ao visualizarmos uma planilha de evolução, o saldo ao final do prazo tende a zero. Eventual resíduo de pequena monta é decorrente do arredondamento do cálculo da prestação e dos juros. Já um saldo ao final do prazo incompatível com essa expectativa pode ser decorrente de erros no desenvolvimento do script ou, até mesmo, de um bug no software.

Esse controle de qualidade também é possível em cálculos de capitalização. Se o cálculo tem o objetivo de obter o valor do depósito para compor determinado montante, o processamento estará correto se esse montante for atingido no prazo desejado. Outros tipos de cálculo de capitalização podem ter o controle de qualidade através de “engenharia reversa”. Por exemplo, no exercício em que foram realizados 4 depósitos de R\$ 700,00 e submetidos a mais 6 meses de rendimentos de juros à taxa percentual de 2% ao mês, chegou-se ao montante de R\$ 3.249,12. A “engenharia reversa” desse cálculo é tomar o valor de R\$ 3.249,12 como valor presente e submetê-lo ao Sistema Francês de Amortização com 4 prestações e 6 meses de carência, à taxa percentual de juros de 2% ao mês. O saldo final no décimo mês deve tender a zero. Isso confirma que a rotina para cálculo desse montante está correta.

```
> sfa(3249.12, 2, 4, 6)
[1] "*****"
[1] " VALOR DA PRESTAÇÃO: 960.95 "
[1] "*****"
      PERÍODO PRESTAÇÃO JUROS AMORTIZAÇÃO SALDO
[1,]      0    0.00  0.00      0.00 3249.12
[2,]      1    0.00 64.98   -64.98 3314.10
[3,]      2    0.00 66.28   -66.28 3380.38
[4,]      3    0.00 67.61   -67.61 3447.99
[5,]      4    0.00 68.96   -68.96 3516.95
[6,]      5    0.00 70.34   -70.34 3587.29
[7,]      6    0.00 71.75   -71.75 3659.04
[8,]      7  960.95 73.18   887.77 2771.27
[9,]      8  960.95 55.43   905.52 1865.75
[10,]     9  960.95 37.31   923.64  942.11
[11,]    10  960.95 18.84   942.11    0.00
>
```

Devemos ter sempre em mente que nenhum software elimina o conhecimento humano. Aquilo que a tecnologia automatiza facilitando o nosso trabalho precisa de validação. Esse processo depende do conhecimento acadêmico e profissional na área de conhecimento que ensejou a demanda pelo desenvolvimento do aplicativo.

Capítulo 8

Computação Paralela com MPI (Message Passing Interface)

Sidney Batista Filho

Programação Paralela com MPI

Programação Paralela

Existem várias aplicações computacionais que podem ser divididas em sub-tarefas independentes, que podem ser executadas simultaneamente em máquinas paralelas (AUDE, 1998, p.73). A programação paralela permite que sejam utilizados os diversos módulos de processamentos de máquinas paralelas para se executar tais aplicações.

As máquinas paralelas podem ser divididas em dois grupos: máquinas com memória primária fisicamente compartilhada, conhecidas como multiprocessadores e máquinas com memória distribuída, conhecidas como multicomputadores. Por outro lado, existem dois modelos de programação, que determinam se a memória é logicamente compartilhada ou não. No primeiro modelo, a comunicação entre as tarefas pode ser feita utilizando-se variáveis compartilhadas entre as mesmas. No segundo modelo de programação, a comunicação é feita através de troca de mensagens. Estas duas classificações – memória fisicamente compartilhada ou não e memória logicamente compartilhada ou não – são ortogonais (ANDREW S. TANENBAUM, 1999, p.526).

Existem diversos ambientes de programação paralela que visam auxiliar o desenvolvimento de aplicações que exploram os recursos de máquinas paralelas. Dentre os diversos ambientes de programação paralela, destaca-se o MPI (SNIR, 1996; MPI FORUM, 1995), por ter se tornado um padrão de facto para qualquer tipo de máquina paralela. O MPI (Message Passing Interface) é uma especificação de uma biblioteca para programação paralela usando troca de mensagens. Esta especificação permite que sejam desenvolvidas implementações do MPI para qualquer tipo de máquina paralela e, ao mesmo tempo, que estas implementações possam aproveitar da melhor forma os recursos de cada tipo de máquina.

O Ambiente MPI

Este capítulo apresenta a especificação da interface padrão para troca de mensagens – o MPI –, enumerando seus principais objetivos e comentando sobre suas versões. Em seguida, são descritos os conceitos básicos do MPI, bem como um pequeno exemplo para que se tenha uma familiarização rápida com o ambiente MPI. Finalmente, são descritas as suas principais operações de comunicação ponto-a-ponto e coletiva.

Apresentação do MPI

MPI (Message Passing Interface) é uma biblioteca de funções padrão e portátil, que define a sintaxe e a semântica de um núcleo de rotinas útil para se escrever programas que exploram a existência de múltiplos processadores com troca de mensagem (PETER PACHECO, 1998, p.3). Segundo (SNIR, 1996, p.xi): “O padrão MPI é o resultado de um esforço que envolveu mais de 80 pessoas de 40 organizações, principalmente dos Estados Unidos e Europa. A maioria dos maiores vendedores de computadores concorrentes da época estava envolvida com o MPI, bem como pesquisadores de universidades, de laboratórios do governo e da indústria”. Este grupo, que definiu o MPI, passou a se chamar MPI Forum (MPI-FORUM) e é responsável pela manutenção deste padrão.

Objetivos

Os principais objetivos do MPI são:

- Permitir que sejam desenvolvidas implementações da especificação MPI em máquinas com arquiteturas diferentes. A partir do mesmo código fonte, que utiliza o MPI, pode-se gerar um programa que pode ser executado em multicomputadores com memória distribuída, em multiprocessadores com memória compartilhada, em clusters de estações de trabalho – COWs (Clusters of Workstations) – e em uma combinação destes tipos de máquinas – ambientes heterogêneos – desde que haja uma implementação da biblioteca MPI para a plataforma específica.
- Permitir uma implementação eficiente de comunicação:
 - evitando, quando possível, cópia de memória para memória,
 - permitindo sobreposição de comunicação e computação.
- Prover uma semântica de interface que seja independente de linguagem de programação. Existem implementações de MPI que podem ser usadas por programas escritos nas linguagens C, C++ e Fortran.
- Prover uma interface de comunicação confiável. Falhas de comunicação devem ser tratadas pelo subsistema de comunicação da plataforma.
- Definir uma interface familiar para os usuários dos sistemas de troca de mensagens mais comuns como por exemplo: PVM (GEIST, 1994) e P4 (BUTLER, 1992).
- Projetar uma interface que permita thread-safety, ou seja, que permita que as funções possam ser invocadas de forma segura por múltiplas threads concorrentemente (ROBBINS, 1996, p.23).

Versões

A primeira versão do MPI, chamada MPI-1.0, foi lançada em 1994 pelo MPI Forum. Em 1995, foi disponibilizada outra versão, a MPI-1.1, sobre a qual este trabalho se baseia (MPI FORUM, 1995; GROPP, 1996).

Posteriormente, foi lançada a versão MPI-2, que consiste de extensões à versão anterior. Nestas extensões estão incluídas operações não existentes nas anteriores como, por exemplo: operações para criação e gerenciamento de processos, entrada e saída paralela e operações para acesso remoto à memória (GROPP, 1999, p.4).

MPI-3.1 foi aprovada pelo MPI Forum em junho de 2015.

Conceitos Básicos em MPI

Tarefas e Communicators

Um programa MPI consiste de várias tarefas autônomas, que executam seu próprio fluxo de instruções, podendo ser classificado, portanto, como sendo um programa MIMD (multiple instruction multiple data), segundo a taxonomia de Flynn (FLYNN, 1972). As tarefas se comunicam através de chamadas às primitivas de comunicação do MPI.

Todas as rotinas do MPI relacionadas à comunicação entre tarefas possuem como parâmetro um communicator, que é uma estrutura de dados que representa um conjunto ordenado de tarefas que podem trocar mensagens entre si. Portanto, esta estrutura de dados representa o domínio da comunicação.

Toda tarefa MPI pertence a pelo menos um communicator e possui, para cada communicator, um rank, que é a sua identificação única no mesmo. Os valores dos ranks são números inteiros consecutivos que variam de 0 até o número de tarefas do communicator menos 1.

Todo par (communicator, rank) identifica apenas uma tarefa em um programa MPI.

Comunicação por Passagem de Mensagem

Existem dois tipos de comunicação por passagem de mensagem: a comunicação ponto- a-ponto, que consiste na transmissão de dados entre um par de tarefas, uma delas enviando e a outra recebendo e a coletiva, que é usada para se transmitir dados entre todas as tarefas de um mesmo grupo especificado por um communicator.

A figura que se segue ilustra, de forma simplificada, um exemplo de comunicação ponto-a-ponto em duas etapas. Na primeira etapa, a tarefa emissora coloca a mensagem no seu buffer de envio e a tarefa receptora possui um buffer de recepção cujo conteúdo é inválido neste momento. Na segunda etapa, a mensagem é copiada para o buffer de recepção. Esta cópia pode ser feita de duas formas: diretamente do buffer de envio para o buffer de recepção (representado pela seta contínua), ou indiretamente, sendo copiada para um ou mais buffers temporários antes de ser copiada para o buffer de recepção (representado pelas setas pontilhadas).

Tipos de chamadas MPI

Os seguintes termos são usados para se discutir as rotinas do MPI:

- local: se a conclusão de uma rotina depender somente da tarefa local que está em execução. Tal operação não requer uma comunicação explícita com outra tarefa do usuário.
- não-local: se a conclusão da rotina necessitar da execução de alguma rotina MPI em outra tarefa.
- bloqueante: se a rotina não retorna enquanto não for permitido ao usuário reutilizar os recursos especificados na chamada da mesma. Por exemplo, uma rotina de envio bloqueante não retorna até que o buffer de envio tenha sido copiado para outro lugar, de modo que o emissor fique livre para alterá-lo.
- não-bloqueante: se a rotina puder retornar antes que a operação iniciada pela mesma conclua. Neste caso, não será permitido ao usuário reutilizar os recursos (como buffers) especificados na chamada. Uma chamada a uma rotina não-bloqueante pode iniciar alterações no estado da tarefa chamadora que ocorrem depois que a chamada retorna. Por exemplo, uma chamada não-bloqueante pode iniciar uma operação de recepção, mas a mensagem é realmente recebida após o retorno da rotina.
- coletiva: se todas as tarefas em um grupo de tarefas precisarem invocar a rotina.

Primeiro exemplo

Como primeiro exemplo, é apresentado um programa simples que usa a biblioteca MPI. O objetivo deste programa é fazer com que cada tarefa cujo rank é diferente de 0 envie uma mensagem para a tarefa cujo rank é 0. A tarefa 0, por sua vez, exibe as mensagens recebidas. Todas as tarefas pertencem a um único communicator: MPI_COMM_WORLD.

```
#include <stdio.h>
#include <mpi.h>

#define MAXLEN 100

main(int argc, char** argv) {
    int myRank;
    /* rank da tarefa */
    int n;
    /* numero de tarefas */
    int source;
    /* rank da origem */
    int dest;
    /* rank do destino */
    int tag = 1;
    /* tag para as mensagens */
    char message[MAXLEN];
    /* buffer para a mensagem */
    MPI_Status status;
    /* status de retorno para o receptor */
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myRank);
    MPI_Comm_size(MPI_COMM_WORLD, &n);
}

if(myRank != 0) {
    dest = 0;
    sprintf(message, "Alo! Da tarefa %d!", myRank);
    MPI_Send(message, strlen(message)+1, MPI_CHAR, dest,
    tag, MPI_COMM_WORLD);
} else {
    for(source = 1; source < n; source++) {
        MPI_Recv(message, MAXLEN, MPI_CHAR, source, tag,
        MPI_COMM_WORLD, &status);
        printf("%s\n", message);
    }
}
MPI_Finalize();
```

... trabalho em progresso.

Mini-biografia

Referência Bibliográfica