FHO | FUNDAÇÃO HERMÍNIO OMETTO - PARADIGMAS DE PROGRAMAÇÃO

LEANDRO APARECIDO DE SOUZA RA 104808
PAULA THAMYRES DA SILVA RA 103146
RAFAEL LUIZ FEMINA RA 83820
WELLINGTON RODRIGO FRANCO RA 68646
DANILLO MYKAEL RA 1022987
HIGOR GABRIEL DE FREITAS RA:104481

LINGUAGEM DE PROGRAMAÇÃO RUST

ARARAS - SP
2019

RESUMO

Este relatório descreve alguns tópicos de uma linguagem de programação conhecida como RUST sendo a mesma contém multi paradigmas de programação, sendo funcional, procedural e orientada a objetos, descrevemos e contextualizando algumas de suas vantagens que esta tecnologia pode oferecer para qualquer desenvolvedor, apresentaremos também um pouco da plataforma utilizada para os desenvolvimentos nesta linguagem.

SUMÁRIO

INTRODUÇÃO

Rust é uma linguagem de programação bem recente. A versão 1.0 que promete a estabilidade sintática e semântica da linguagem no futuro foi liberada agora em maio de 2015 . Começado como projeto particular do funcionário da Mozilla Graydon Hoare e depois apoiado pela Mozilla para ajudar no desenvolvimento de uma nova browser engine "Servo" Rust foi anunciada pela primeira vez em 2010. A ideia principal que levou a criação de Rust é a ideia de uma linguagem de programação para o desenvolvimento de sistemas (de baixo nível) que tenha um modelo de memória seguro e que inclua também elementos modernos como por exemplo pattern matching, tipos algébricos, funções de ordem maior (closures) e genéricos. Os princípios gerais que guiam o desenvolvimento da linguagem são: não deixar a segurança de memória ser comprometida, abstrações não devem custar nada no tempo de execução, praticidade é a chave. A linguagem foi desenvolvida visando substituir linguagens de programação de baixo nível como por exemplo as linguagens C e C++ nas respectivas áreas de aplicação. Um dos principais problemas de programação com C/C++ é a da segurança de memória. Os erros mais comuns são relacionados a ponteiros que apontam para regiões inválidas de memória (use after free, index out of bounds), vazamento de memória (memory leaking - memória não mais usada mas erradamente não liberada). Por isso uma das mais importantes características de Rust são as promessas de segurança de memória que ela dá. Essa característica é especialmente importante para a programação concorrente e paralela porque ajuda a evitar problemas típicos e intrínsecos desses princípios de programação. O conceito fundamental de

ownership que torna isso possível. Além disso a linguagem oferece vários recursos normalmente não encontrados em linguagens para a programação de baixo nível.

HISTÓRICO DA LINGUAGEM

Rust (Ferrugem OU oxidação em inglês) é uma linguagem de programação desenvolvida sob a óptica de código aberto; sendo voltada às ações de concorrência (sem disputa de dados), velocidade, abstração (sem overhead) e segurança de memória (sem coletor de lixo), é uma linguagem que atende a alguns paradigmas de programação, como: programação genérica, programação funcional (declarativa), programação imperativa (estruturada e orientada a objetos) e programação concorrente.

Os programas escritos em Rust têm desempenho similar ao de programas escritos em linguagens de baixo nível e operam com abstrações encontradas em linguagens de alto nível (BALBAERT, 2015, p. 3). Rust é unia linguagem de sistemas, o que significa que é voltada a aplicações “pesadas” para programadores que seguem uma linha de produção hardcore, como o desenvolvimento de sistemas operacionais, mecanismos de simulação para aplicações de realidade virtual, engine de jogos, entre nutras (MOZ://A. 2017).

Na linguagem de programação Rust, os paradigmas de programação suportados se misturam e interligam-se de forma natural. Graças a sua engenharia de desenvolvimento, a linguagem permite escrever programas ambiciosos cm alto nível, com desempenho similar, encontrados em programas escritos em assembly de forma rápida.

Quando o programador comete um deslize, o compilador informa o que está incorreto ou necessita de acertos para que o programa seja escrito de forma adequada. A maior parte das verificações de segurança de acesso e gerenciamento de memória é realizada em tempo de compilação, garantindo alto grau de

desempenho em tempo de execução de um programa (RUST, 2017). Dessa forma, situações permitidas em outras linguagens em Rust são verificadas na compilação, e, se houver algo incorreto, o compilador indica o que deve ser corrigido para que o código esteja dentro dos parâmetros operacionais necessários. As poucas ações não erificadas na compilação resultam cm algum tipo de erro dc execução que, quando interrompem o programa, pode ser corrigido mudando-se a postura de programação. Os erros em Rust são chamados de pânico.

Rust é uma Linguagem de programação que começou a ser desenvolvida cm 2006 pelo canadense Graydon Hoane, um funcionário da Mozilla Research. que a produziu com influência da linguagem OCamL, utilizada em aplicações industriais com suporte à programação funcional, programação imperativa e programação orientada a objetos. Em 2009, a Mozilla interessou-se pela linguagem e passou a ser o seu primeiro investidor. Com a participação da M07.illa, a linguagem foi porrada ara o compilador LLVM, que se trata de uma coleção de tecnologias de compiladores com ferramentas modulares e reutilizáveis, e lançou Rust oficialmente cm 2010 (DAN_ATILIO, 2015, GOMES. 2010 & MAFRA. 2014). Desde então, a linguagem vem sendo desenvolvida com a participação de Utfl extenso Lime de desenvolvedores, além, é claro, de Graydon Hoane (https://w.rust-lang.org/pt-BR/team.htm1.).

A linguagem Rust tem uma sintaxe semelhante à sintaxe encontrada nas linguagens de programação C e C++, com um certo tempero de estilo da linguagem Ada. Esta linguagem é extremamente versátil e permite escrever programas das mais diversas finalidades, como drivers e sistemas operacionais, além de outras possibilidades já apontadas.

Embora seja uma linguagem recente, completando oito anos de vida em 2018. Já ganhou grandes adeptos, destacando-se a própria empresa Mozilla no desenvolvimento do programa Servo e de partes importantes do programa dc navegação Firefox. Rust vem sendo utilizada também em projetos como OpenDNS, Dropbox. Skylight, Autumn, Tesscl. Chef Habitat, Neon. Xi editor, como destaca Tavares (2017), além do sistema operacional Redox OS, totalmente escrito cm Rust (https://w.redox os.org/).

A linguagem Rust não tem, ainda, uma documentação de padrões ao estilo ISO/IEC, como ocorre com outras linguagens já consagradas, como C, C++, Pascal, entre outras.

Neste sentido, é mantido um conjunto de documentos a partir do endereço https://doc.rust-tang.org/stabl..e/reference/tntroductton.html., OS quais devem ser considerados por desenvolvedores de software.

Com Rust é possível desenvolver programas menos propensos a bugs e vulnerabilidades de segurança no acesso à memória. A linguagem inclui recursos poderosos, como abstrações de custo zero, gerenciamento seguro de memória, concorrência e muito mais (MOZ://A, 2017). Pode ser usada de forma imperativa ou declarativa, a critério do programador.

Uma curiosidade sobre a linguagem Rust é sua mascote, além do logotipo oficial (catraca de bicicleta com a letra R). Depois de o sistema operacional Linux usar um pinguim e um antigo gerenciador de banco de dados chamado FoxPro usar urna raposa como mascote, vários projetos da área da computação começaram a considerar o uso de animais para seus produtos como mascotes. No caso da linguagem Rust, sua mascote, no oficial, é o Ferris, um simpático caranguejo que pode ser conferido no site Rustocean no endereço: http://www.rustacean.net/.

CARACTERÍSTICAS DA LINGUAGEM RUST

Rust possui muitos recursos que o tornam útil, mas os desenvolvedores e suas necessidades diferem. Eu cubro cinco dos principais conceitos que tornam o aprendizado do Rust valioso e mostro essas ideias na origem do Rust.

Código reutilizável via módulos

O Rust permite organizar o código de uma maneira que promove sua reutilização. Você alcança este nível de organização usando módulos, que contêm funções, estruturas e até mesmo outros módulos que você pode tornar públicos (ou seja, expor aos usuários do módulo) ou privados (ou seja, usar somente dentro do módulo e não pelos usuários do módulo, pelo menos não diretamente). O módulo organiza o código como um pacote que outros usuários podem usar.

Você usa três palavras-chave para criar módulos, usar módulos e modificar a visibilidade dos elementos nos módulos.

A palavra-chave mod cria um novo módulo

A palavra-chave use permite usar o módulo (expor as definições no escopo para usá-las)

A palavra-chave pub torna elementos do módulo públicos (caso contrário, eles são privados).

Os módulos permitem coletar a funcionalidade de maneira pública ou privada, mas também é possível associar métodos a objetos usando a palavra-chave impl .

Verificações de segurança para código mais limpo

O compilador Rust aplica garantias de segurança da memória e outras verificações que tornam a linguagem de programação segura (diferente de C, que pode ser inseguro). Portanto, em Rust, você nunca precisará se preocupar com

ponteiros pendentes ou com o uso de um objeto após ele ter sido liberado. Essas coisas fazem parte da linguagem Rust central. Mas, em áreas como desenvolvimento integrado, é importante fazer coisas como colocar uma estrutura em um endereço que represente um conjunto de registros de hardware.

O Rust inclui uma palavra-chave unsafe com a qual é possível desativar verificações que geralmente resultariam em um erro de compilação. Conforme mostrado na Listagem 3, a palavra-chave unsafe permite declarar um bloco inseguro. Nesse exemplo, eu declaro uma variável imutável x e, em seguida, um ponteiro para essa variável chamado raw. Então, para remover a referência de raw (que nesse caso imprimiria 1 no console), eu uso a palavra-chave unsafe para permitir essa operação, que de outra forma seria sinalizada na compilação.

É possível aplicar a palavra-chave unsafe em funções e em blocos de código dentro de uma função do Rust. A palavra-chave é mais comum na gravação de ligações em funções não Rust. Esse recurso torna o Rust útil para coisas como desenvolvimento de sistema operacional ou programação integrada (bare-metal).

Melhor manipulação de erros

Erros acontecem, independentemente da linguagem de programação que você usa. No Rust, os erros encontram-se em dois campos: erros irrecuperáveis (o pior tipo) e erros recuperáveis (o tipo não tão ruim).

Suporte para tipos de dados complexos (coleções)

A biblioteca padrão do Rust inclui várias estruturas de dados populares e úteis que podem ser usadas em seu desenvolvimento, inclusive quatro tipos de estruturas de dados: sequências, mapas, conjuntos e um tipo diverso.

Para sequências, é possível usar o tipo vetor (Vec), que eu usei no exemplo de encadeamento. Esse tipo fornece uma matriz dinamicamente redimensionável e é útil para coletar dados para processamento posterior. A estrutura VecDeque é semelhante ao Vec, mas é possível inseri-la em ambas as extremidades da

sequência. A estrutura LinkedList também é semelhante ao Vec, mas com ela é possível dividir e anexar listas.

Para mapas, você tem as estruturas HashMap e BTreeMap. Use a estrutura HashMap para criar pares chave-valor e você poderá referenciar elementos por suas chaves (para recuperar o valor). A BTreeMap é semelhante à HashMap, mas ela pode classificar as chaves e você pode iterar facilmente todas as entradas.

Para conjuntos, você tem as estruturas HashSet e BTreeSet (que você observará após as estruturas de mapas). Essas estruturas são úteis quando você não possui valores (apenas chaves) e chama novamente as chaves que foram inseridas.

A estrutura diversa é atualmente a BinaryHeap. Essa estrutura implementa uma fila de prioridade com um heap binário.

AMBIENTES DE DESENVOLVIMENTO DISPONÍVEIS

Rust possui uma implementação do Language Server Protocol, o RLS, que fornece autocompletar e refatoração independente do editor de texto ou ambiente de desenvolvimento integrado. Os seguintes editores dão suporte ao RLS de forma nativa ou através de extensões/plugins

- Visual Studio Code - fornecido pela extensão oficial;
- Atom - fornecido pelo pacote oficial ide-rust;
- Sublime Text - fornecido pelo pacote oficial Rust Enhanced;
- Eclipse - fornecido pelo plugin Corrosion;
- GNOME Builder - suporte nativo;
- O IntelliJ IDEA fornece suporte a Rust sem o uso do RLS, através do plugin IntelliJ Rus;

Instalação e teste da linguagem Rust (tirado do livro Primeiros passos com a linguagem Rust Por José Augusto N. G. Manzano)

Para desenvolver programas de computador em determinada Linguagem de programação, é necessário instalar tal linguagem no computador em uso. No C3SO da linguagem Rust, há algumas formas, segundo sua documentação, de efetuar sua instalação, sendo a mais comum feita por meio da ferramenta de instalação rustup, que gerencia de forma consistente as diversas versões lançadas da linguagem dentro do seu ciclo de lançamento para diversas plataformas computacionais. Para mais informações sobre o gerenciador rustup. consulte o endereço https : //gi..thub.com/rust-1.ang-nursery/rustup. rs/blob/naster/README .md.

Todas as ferramentas de operação da linguagem Rust são instaladas no diretório %IJS[RPROFILE%\.cargo\btn, onde ficam os programas rustup. cargo e rustc. Por isso é comum que esse diretório seja manualmente configurado junto variável PATH caso o processo de instalação não surta este efeito automaticamente, o que pode ocorrer por causa da grande variedade de configuraçòes e plataformas em que a linguagem pode ser usada. Para saber se o processo de instalação foi bem—sucedido, execute, no prompt de comando do terminal, modo texto do sistema operacional, a instrução rustc - - version. Sc esta apresentar falha na chamada do programa, talvez seja necessário configurar manual mente a variável PATH,

No caso da instalação do programa rustup para o sistema operacional Windows, é necessário ter previamente instaladas as ferramentas de compilação Microsoft C++ do Visual Studio edição 2013 ou posterior. Caso não tenha esse recurso instalado, é necessário fazê-lo primeiro. Para isso, acesse, por meio do programa de navegação, o endereço http://landtnghub.vtsuatstudto.com/vi,suat-cpp-bufld-toots, acione o botão Download Visual C++ Build Tools 2015, copie o arquivo e execute-o por meio de um duplo clique.

Para instalar o programa rostup nos sistemas operacionais Linux ou Mac OS é necessário estar com urna janela de terminal aberta e informar a instrução curl. https://sh.rustup.rs -sSf I sh para fazer o download do programa. Provavelmente será necessário informar sua senha de usuário. A instalação estará concluida quando for apresentada no terminal a mensagem “Rust is installed now. Great!’ (DHINAKARAN, 2017, p. 9; KALHLAVIRTA, V., 2017, p. 8). Para mais detalhes sobre a instalação Linux ou Mac, acesse o endereço https://doc.rust lang.org/book/second-edi.tion/chOl-O1-tnstaflati.on.htnl..

Depois de instalar as ferramentas de compilação C++ da Microsoft, acione a instalação do programa rustup-tntt.exe. que é realizada em urna tela do modo console (modo texto ou modo prompt). Informe a opção 1 (um), acione a tecla <Enter> e aguarde o processo de instalação ser encerrado, o que ocorrerá quando for apresentada a mensagem Press the Fnter key to conti.nue. Acione a tecla

<Enter>. Após a instalação, abra um terminal texto no sistema operacional em uso e execute a instrução rustc - -ver si.on, que, se tudo estiver correto, apresentará uma mensagem semelhante à informação rustc 1.24.0 (4d90ac38c 2018-02-12) indicando que a linguagem pode ser usada em seu sistema.

O programa rustup é o programa de instalação da linguagem Rust, o qual faz ocontrole de versões a ser utilizadas, acompanhado dos programas rustc. que SO OS compiladores da Linguagem, e do programa cargo, que permite gerenciar as dependências de pacotes (conjuntos de bibliotecas) a ser usados, possibilitando compilar tudo o que é necessário para um programa ser criado por meio de pacotes externos que são obtidos do repositório oficial chamado creates.to.

Como já é tradição em várias linguagens de programação, o primeiro programa a ser escrito em uma nova linguagem é o programa “Alo, Mundo!” (sem acentuação). Para isso, é adequado que se crie no sistema em uso um diretório OU uma pasta para a gravação dos projetos dos programas-torne. Sugere-se então a criação da pasta ou do diretório estudo. Para o programa em questão, crie a pasta ou o diretório alo.

Após criar a pasta ou o diretório de projeto ato dentro da pasta ou diretório estudo. faça seu acesso via linha de comando de acordo com as regras do uso do sistema operacional em operação. Abra um editor de texto e escreva o código seguinte:

```
fn mamo {
printtn!(’AIo, Mundo!);
```

Grave na sequência dentro de estudo/ato o código de programa com o nome main.rs (não use outro nome), e na sequência execute a instrução na linha dc comando:

```
rustc nain.rs
```

Em seguida, se não houver nenhum erro de codificação. o programa poderá ser executado, Na linha de comando do sistema operacional Windows, escreva a

instrução main (ou .\main) e acione <Enter>. Nos sistemas operacionais padrão POSIX, como OS sistemas Linux e Mac, escreva o comando . /main e acione <Enter>. Observe a apresentação da mensagem “Alo, Mundo “.

Instalando Rust no Microsoft Windows

Para Windows, existe um pacote padrão MSI que deve instalar tudo o que for necessário. Ele pode ser encontrado no endereço: https://www.rust-lang.org/pt-BR/downloads.html.

Considerações sobre o Windows

No Windows, o Rust requer adicionalmente as ferramentas de construção C++ para o Visual Studio 2013 ou mais recente. A maneira mais fácil de adquirir as ferramentas de construção é instalando o Microsoft Visual C++ Build Tools 2017 que fornece apenas as ferramentas de construção do Visual C++. Como alternativa, é possível instalar o Visual Studio 2017, o Visual Studio 2015 ou o Visual Studio 2013 e, durante a instalação, selecionar ferramentas C++.

Para obter informações adicionais sobre como configurar o Rust no Windows, consulte a Documentação do rustup específica do Windows.

INSTALAÇÃO DO RUST

INSTALANDO NO LINUX OU MAC

Se você está em um ambiente Linux ou Mac, tudo o que você precisa é abrir um terminal e digitar o seguinte comando:

```
$ curl https://sh.rustup.rs -sSf | sh
```

Este comando vai baixar um script e iniciar a instalação. Talvez seja solicitado que você digite sua senha. Se tudo ocorrer bem, a mensagem abaixo vai aparecer:

```
Rust is installed now. Great!
```

Claro, se você não aprova o uso do `curl | sh`, você pode baixar o *script*, inspecioná-lo e executá-lo da maneira que achar melhor.

O *script* de instalação já adiciona automaticamente o Rust à variável PATH do seu sistema logo após o seu próximo *login*. Se você quiser usar o Rust imediatamente, execute o seguinte comando no seu terminal:

```
$ source $HOME/.cargo/env
```

Outra opção é adicionar a linha abaixo no seu `~/.bash_profile`:

```
$ export PATH="$HOME/.cargo/bin:$PATH"
```

INSTALANDO NO WINDOWS

No Windows, visite o *site* https://rustup.rs e siga as instruções para baixar o arquivo rustup-init.exe. Execute este arquivo e siga as demais instruções que aparecerem na sua tela.

Se você está usando um *shell* diferente, talvez você poderá usar os mesmos comandos que os usuários de Linux ou Mac usam. Se algum comando não funcionar, consulte a documentação referente ao *shell* que você está utilizando.

CRIANDO UM PROGRAMA OLÁ, MUNDO

Agora que Rust já está instalado, vamos escrever nosso primeiro programa. Quando aprendemos uma nova linguagem de programação, é tradicional escrever um pequeno programa que imprime "Olá, mundo!" (*"Hello, world!"*) na tela, e é exatamente isso que vamos fazer nesta seção.

Nota: Este livro assume que o leitor tem um pouco de familiaridade com a linha de comando. Rust não exige que você use um determinado editor ou IDE, ou seja, você está livre para usar o que bem entender para escrever seu código Rust.

CRIANDO UM DIRETÓRIO DE PROJETO

Primeiramente, crie uma pasta para colocar o seu código Rust. O Rust não se importa onde você vai armazenar o seu código, mas neste livro, nós sugerimos criar um diretório chamado *projetos* e armazenar todos os seus projetos ali. Abra o seu terminal e digite os seguintes comandos:

Linux e Mac:

```
$ mkdir ~/projetos
$ cd ~/projetos
$ mkdir ola_mundo
$ cd ola_mundo
```

Windows CMD:

```
> mkdir %USERPROFILE%\projetos
> cd %USERPROFILE%\projetos
> mkdir ola_mundo
> cd ola_mundo
```

Windows PowerShell:

```
> mkdir $env:USERPROFILE\projetos
> cd $env:USERPROFILE\projetos
> mkdir ola_mundo
> cd ola_mundo
```

ESCREVENDO E EXECUTANDO UM PROGRAMA EM RUST

Crie um novo arquivo *main.rs*. Arquivos relacionados à linguagem Rust sempre terminam com a extensão *.rs*. Se o nome do seu arquivo tem mais de uma palavra, use um *underscore* (_) para separá-los. Por exemplo, você deve nomear o seu arquivo *ola_mundo.rs* em vez de *olamundo.rs*.
Agora abra o arquivo *main.rs* que você acabou de criar e digite o seguinte código:

Arquivo: main.rs

```
fn main() {
    println!("Olá, mundo!");
}
```

Salve o arquivo e volte ao seu terminal. No Linux ou OSX, digite os seguintes comandos:

```
$ rustc main.rs
```

```
$ ./main
Olá, mundo!
```

Para executar o seu programa no Windows, digite `.\main.exe` em vez de `./main`. Independente do seu sistema operacional, você deverá ver a mensagem `Olá, mundo!` no seu terminal. Se você chegou até aqui, parabéns! Você escreveu o seu primeiro programa em Rust. Isso faz de você um programador Rust!

ANATOMIA DE UM PROGRAMA EM RUST

Agora vamos ver o que aconteceu com o seu programa "Olá, mundo!" em detalhes. Aqui está a primeira peça do quebra-cabeça:

```
fn main() {

}
```

Estas linhas definem uma *função* em Rust. A função `main` é especial: é a primeira coisa que é executada em cada programa escrito em Rust. A primeira linha diz: "Estou declarando uma função chamada `main` que não contém nenhum parâmetro e que não retorna nada." Se existissem parâmetros, eles estariam dentro dos parênteses, `(` e `)`.

Também repare que o corpo da função está envolvido por duas chaves, `{` e `}`. Rust requer essas chaves no começo e no fim do corpo de cada função. Considera-se boa prática colocar a chave inicial na mesma linha da declaração da função, com um espaço entre elas.

Dentro da função `main`:

```
    println!("Olá, mundo!");
```

Esta linha faz todo o trabalho nesse pequeno programa: imprime um texto na tela. Existem alguns detalhes a se notar aqui. O primeiro é que o estilo de indentação do Rust usa quatro espaços, e não um *tab*.

A segunda parte importante é o `println!`. Este comando está chamando uma *macro*, que é a forma de se fazer metaprogramação em Rust. Se estivéssemos chamando uma função, ficaria assim: `println` (sem o `!`). Vamos discutir *macros* em Rust com mais detalhes no Apêndice D, mas por agora, você só precisa saber que quando usamos um `!`, significa que estamos chamando uma *macro* em vez de uma função.

Em seguida vem `"Olá, mundo!"`, que é uma *string*. Nós passamos esta *string* como um argumento para a *macro* `println!`, que por sua vez imprime a *string* na tela. Fácil!

A linha termina com um ponto e vírgula (`;`). O `;` indica que esta expressão acabou, e que a próxima está pronta para começar. A maioria das linhas de código em Rust terminam com um `;`.

CONSIDERAÇÕES FINAIS

A linguagem Rust une vários conceitos modernos com o conceito de linguagens de programação em baixo nível. A especificidade e a vantagem são as garantias de segurança de memória viabilizadas pelo princípio de propriedade (ownership). Por ser uma linguagem muito nova ainda faltam algumas caraterísticas nas bibliotecas básicas sobretudo na área de paralelização. Supomos que essa falta será suprida brevemente porque já existe uma comunidade significativa bem como uma empresa claramente interessada no desenvolvimento de Rust.

REFERÊNCIAS

[HAA12] G. Hoare e A. Abel Avram. Interview on rust. http://www.infoq.com/news/2012/08/Interview-Rust, 2012. Último acesso: 06/2015.

[Hoa] G. Hoare. stage1/rustc builds. https://mail.mozilla.org/pipermail/rust-dev/2011-April/000330.html. Último acesso: 06/2015.

[HPC+13] E. Holk, M. Pathirage, A. Chauhan, A. Lumsdaine e N.D. Matsakis. Gpu programming in rust: Implementing high-level abstractions in a systems-level language. Em Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International, páginas 315–324, May 2013.

Principais conceitos em Rust (tirado do site https://www.ibm.com/developerworks/br/library/os-developers-know-rust/index.html)