Guia do Bootloader das Galáxias: Noah's Ark Bootloader

Gabriel Campelo, *Bacharelando, UFRR,* Joshua Pereira, *Bacharelando, UFRR*

***Abstract*—This journal explains what is a bootloader and how to create and simulate a simple custom made bootloader.**

***Resumo*—Este artigo explica o que é um bootloader e como criar e simular um bootloader personalizado simples.**

***Index Terms*—Bootloader, Bootsector, Sistema Operacional, LaTeX, paper, template.**

I. Introdução

ESTE artigo tem o objetivo de explicar, de maneira simples, o funcionamento de um bootloader. Será apresentado o que é um processo de boot, onde o bootloader se encaixa nesse processo, e como o bootloader Noah's Ark foi desenvolvido e simulado. Para o desenvolvimento do projeto foram utilizadas as ferramentas Netwide Assembler (NASM) e QEMU.

27 de Junho de 2019

II. O processo de Boot

Existe maneiras diferentes de se considerar o início do processo de boot de um computador, alguns consideram que o processo se inicia quando o kernel é carregado e executado, enquantos outros consideram que se inicia quando ligamos a máquina. Apesar de tal questão não influenciar no processo em si, para fim deste artigo iremos considerar que o processo se inicia ao ligarmos a máquina.

Em geral, podemos dividir o processo de boot em 4 fases consecutivas:

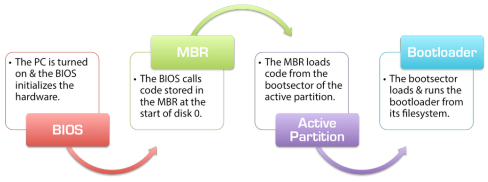

Figura 1. Etapas da Sequência de Boot

A. BIOS

Quando o sistema é ligado, os registradores da CPU são resetados para um valor pré-definido. A CPU então executa a instrução contido no local apontado pelos registradores. Neste local da memória ROM, é garantido que irá ser encontrado a BIOS do sistema, o qual será executado. A BIOS irá realizar um processo denominado Power-On Self Test, o qual irá inicializar os dispositivos de entrada e saída do hardware e, quando possível, realizar um pequeno teste de funcionalidade.

Após este processo, a BIOS irá buscar por um dispositivo de boot nos discos rígidos disponíveis (um pequeno programa geralmente localizado em um endereço específico do disco), e então irá carregá-lo na memória. A BIOS então termina sua execução e entrega o controle para o Master Boot Record.

B. Master Boot Record(MBR)

O MBR é um setor de boot (bootsector) encontrado nos primeiros bytes do disco. Nele, existe uma Tabela de Partições(Partition Table), onde se encontram as partições lógicas do sistema.

No MBR também se encontra o código bootstrap. Este código é responsável por procurar uma partição ativa na Tabela de Partições, checa se as demais partições estão desativas, e carrega na memória o código do bootsector daquela partição (chamado de Volume Boot Record) e o comanda a CPU á executa-lo. Este código de bootstrap costuma ser chamado de Primeiro Estágio do Bootloader.

Por fim, temos também a Boot Signature, que são os últimos dois bytes do MBR que irão indicar se o dispositivo de boot é bootável ou não. Caso estes bits tenham os valores 0x55 e 0xAA, o disco será considerado bootável, caso contrário a BIOS assumirá que ele não é bootável.

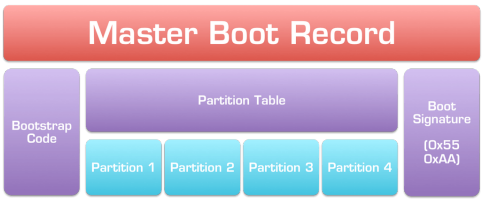

Figura 2. Organização do MBR

C. Partição Ativa

Após ser carregada na memória, a CPU irá executar o código prsente no bootsector da partição, o qual irá realizar uma instrução de desvio de XX bytes à frente, ou para outra secção da partição, e depois executar o que se chama de Segundo Estágio do Bootloader. Esse segundo estágio irá procurar pela partição por um arquivo específico.

Após encontrar tal arquivo, será executado o bootstrap encontrado no bootsector da partição, que é o último estágio do processo de Boot: carregar o kernel do Sistema Operacional e preparar o computador para ele.

Professor Dr. Herbert Rocha, da disciplina de Sistemas Operacionais, pelo apoio e paciência.

Nossos colegas de laboratório, pelo apoio psicológico.

D. O Bootloader

Como vimos anteriormente, o bootloader atua em 2 estágios: primeiro quando a BIOS procura e carrega o código bootstrap do MBR, enquanto o segundo estágio é o processo de busca e inicialização do kernel do sistema operacional.

Nessa última fase, o bootloader irá carregar todos as configurações e base de dados necessários para o kernel do sistema operacional (come headers, módulos, etc.). Opcionalmente, o bootloader poderá apresentar uma opção para selecionar qual sistema operacional o usuário deseja carregar. Após preparar todo o ambiente para o kernel, o bootloader carrega o kernel do sistema operacional, executa o coódigo inicialização do kernel e entrega o controle do computador para o sistema operacional.

A partir deste ponto o sistema operacional inicia e acaba o processo de boot do sistema computacional.

III. O QUE É UM BOOTLOADER

Como foi mencionado anteriormente, o bootloader é um programa que é executado antes de qualquer sistema operacional, responsável pela busca e carregamento do kernel do sistema operacional. Ou seja, é um programa utilizado para inicializar um sistema operacional.

Um bootloader está associado à apenas um S.O., enquanto um S.O. pode ter múltiplos bootloaders, classificados como primário e secundário.

Um bootloader contém dois estágios: o Primeiro Estágio, que procura e carrega na memória uma partição ativa (bootável); e o Segundo Estágio, que procura pelo arquivo de boot do kernel presente na partição, carrega-o na memória para execução e prepara o ambiente para o kernel a ser carregado antes de finalmente executá-lo.

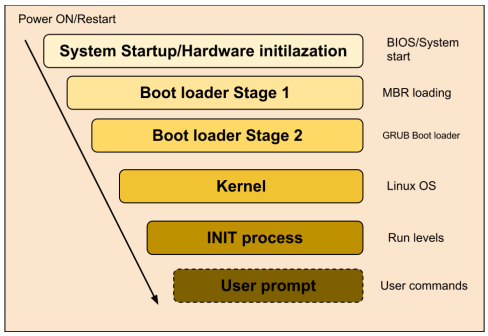

Figura 3. Estágios do Bootloader no processo de Boot de um sistema linux

IV. NOAH'S ARK BOOTLOADER

Para o desenvolvimento do bootloader, foram necessários 1 (um) dia para o estudo do assunto, que inclui entender como é o processo de Boot; 2 (dois) dias para fazer o estudo das ferramentas utilizadas, assim como para entender e desenvolver os programas necessários.

Noah's Ark Bootloader foi desenvolvido nas linguagens Assembly e C. O programa principal, o bootloader, foi desenvolvido em Assembly, enquanto alguns programas que auxiliarão o bootloader foram desenvolvidos em C. Uma boa parte das funções executadas pelo bootloader encontram-se em arquivos separados, deixando no arquivo principal somente as mais importantes. O objetivo deste bootloader é tentar realizar as etapas de busca, carregamento e execução de um kernel linux.

As ferramentas utilizadas para a realização deste projeto foram:

- Netwide Assembler (NASM): para a compilação dos programas e criação dos arquivos binários.
- QEMU: para simular e testar o bootloader.
- Editor de texto genérico.

A. Procedimentos

O primeiro procedimento de nosso bootloader é declarar sua posição na memória, de onde ele será buscado e executado pela BIOS. Em seguida, entrar no modo 16 bits, também conhecido como Modo Real (Real Mode - RM). Nesse modo, nosso computador reconhece apenas 1MB de memória. Aqui iremos armazenar o endereço de memória do kernel e o número do dispositivo de boot, que serão usados posteriormente.

A seguir, iremos atualizar o Stack Pointer do sistema para que ele aponte para um espaço vazio na memória. Esse procedimento é importante pois este espaço vazio será onde o kernel irá se manifestar. Feito isso, agora carregamos o kernel para um local da memória que armazenamos anteriormente, através de um procedimento de leitura de disco, irá procurar pelo kernel na partição, que é o drive de boot anteriormente armazenado, e colocá-lo na memória.

Agora nós devemos desligar as interrupções da BIOS, em preparação para a seguinte etapa: devemos executar um procedimento para entrarmos no Modo Protegido (Protected Mode - PM), no espaço 32 bits. Este modo nos permite que consigamos extender o limite de nossa memória, o que em troca possibilita que executemos códigos maiores. Nesse modo nos iremos finalmente executar o kernel. Nesta etapa, nós fazemos um procedimento que irá chamar a função que inicializa o kernel. Após isso, o kernel começa a executar independentemente. Nós então passamos o controle para o kernel. E assim finaliza o nosso bootloader.

B. Testes

Foram realizados testes primeiramente para entender como utilizar as ferramentas QEMU e NASM. Para tal, foi desenvolvido um pequeno programa que seria reconhecido como um bootloader com o único objetivo de mostrar na tela a senteça "Hello, World!".

Os testes seguintes forma realizados com o objetivo de tentar localizar o kernel linux através do bootloader. Porém, após estudarmos mais afundo sobre esse processo, se demonstrou necessário um conhecimento mais afundo sobre o kernel e como carregá-lo no sistema.

Após varias tentativas, não fomos capazes de configurar o kernel para que este se encontre em uma posição específica da memória. Devido à isso, o Noah's Ark Bootloader não foi finalizado completamente, pois apesar de executarmos todas as fases até o procesos de troca para o Modo Protegido, não foi possível testar se ele consegue executar o kernel.

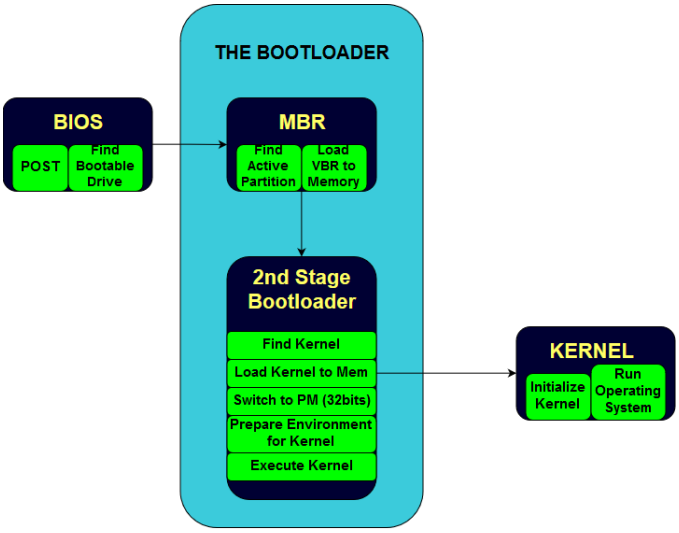

Figura 4. Passos seguidos pelo Bootloader

C. *GitHub*

O GitHub do projeto Noah's Ark Bootloader pode ser encontrado aqui.

V. Conclusão

Nós vimos neste artigo como o boot de um sistema computacional vem a ocorrer, e como o bootloader está envolvido neste processo. Concluímos então como o bootloader é um programa muito importante para o sistema, pois é ele quem inicializa todo o sistema que o usuário vem a conhecer e interagir. Todo sistema computacional necessita de um bootloader, o qual é específico para cada sistema. Existem muitos bootloaders conhecidos, como o GRUB e o LILO.

O desenvolvimento de um bootloader requer conhecimento de várias áreas da computação, principalmente sobre programação em baixo nível e sobre o hardware do computador.

Apesar de todos nossos esforços e tentativas, fomos capazes de fazer com que o Noah's Ark Bootloader chegasse até a etapa de troca para o modo protegido. Porém, não fomos capazes de executar corretamente um kernel linux.

Apêndice A
Leitura adicional - Modo Protegido

Foi mencionado que o bootloader troca de modo durante o seu segundo estágio, indo do Modo Real (16 bits) para o Modo Protegido (32 bits). Essa troca é necessária para que o sistema tenha uma capacidade maior de memória para poder executar o kernel. Aprofundemos um pouco mais sobre o modo protegido e como ele aumenta a capacidade da memória.

O Modo Protegido é um modo operational compatível com CPUs x86. Ele permite que o software do sistema consiga usar funcionabilidades adicionais, dentre elas a virtualização de memória, paginação e capacidade de multi-tarefa. Assim podemos perceber que trocamos para este modo para ter acesso à essas funcionalidades, em especial a virtualização de memória que será usada para aumentar a capacidade da memória de maneira virtual.

Referências

[1] Tanenbaum, Andrew S and Bos, Herbert *Modern Operating Systems* Pearson, 2015.

[2] Eugene Obrezkov, 'How to implement your own 'Hello, World!' boot loader'. [Online]. Disponível: https://blog.ghaiklor.com/2017/10/21/how-to-implement-your-own-hello-world-boot-loader. [Acessado 26-06-2019]

[3] NeoSmart Knowledgebase, 'The BIOS/MBR Boot Process'. [Online]. Disponível: https://neosmart.net/wiki/mbr-boot-process. [Acessado 26-06-2019]