Trabalho 1 - Sistemas Operacionais

Pedro Luiz da Costa Silva
Ciência da Computação
FACOM - UFMS
2022.1

1 Introdução

Este trabalho tem como objetivo a criação de um código em C capaz de, a partir de um bootblock e um kernel fornecidos, gerar uma imagem de Sistema Operacional pronta para ser colocada em um disco bootável. A imagem gerada foi testada usando o emulador Bochs.

2 Decisões de Implementação

- Uso de `fread` para leitura dos cabeçalhos ELF e de programa;
- Uso de buffers (unsigned char) para armazenamento temporário dos dados de bootfile e kernelfile;
- Uso das mesmas assinaturas de funções no arquivo fornecido `buildimage.c` (apesar de nem todas funções usarem todos os argumentos).

3 Funcionalidades não implementadas

Todas funcionalidades pedidas foram implementadas.

4 Problemas enfrentados na implementação

- **Problemas de padding:** tentei escrever na imagem sem fazer padding, ou seja, não levava em conta o tamanho dos setores do floppy. Com isso, a imagem criada não era bootável. Resolvido fazendo o padding apropriado nos setores;
- **Falta de assinatura:** sem a assinatura de bootloader no final do primeiro setor da imagem, o disco não era reconhecido como bootável. Resolvido inserindo a assinatura como em um MBR;
- **Problemas com fwrite:** tentei escrever do bootfile para o imagefile sem usar um buffer, e assim o fwrite interpretava o ponteiro de bootfile como um buffer (e os dados vinham errados). Resolvindo implementando um buffer;
- **Descobrir como dizer ao bootloader quantos setores ler do kernel:** Carregar o bootloader no primeiro setor e o kernel nos setores seguintes não era suficiente para fazer a imagem ser bootável, pois o bootloader precisava saber quantos setores o kernel ocupava.
  Para resolver isso, rodei um hexdump da imagem gerada pelo `buildimage.given`. Vi que a partir do terceiro byte havia um inteiro 9, o que não havia no `.text` de bootloader. Como 9 é o número de setores gerados pelo arquivo `kernel.s`, deduzi que o bootloader usava essa posição (terceiro byte) para saber quantos setores ler do kernel.
  Inserindo o inteiro correspondente ao número de setores do kernel a partir do terceiro byte, a imagem passou a ser bootável.

5 Observações

- Uso do executável gerado: `./buildimage [--extended] <bootblock> <kernel>`