# Relatório de Trabalho 1 - Sistemas Operacionais (2022/1)

Aluno: Edmar Caixeta Filho (2019.1905.030-2)
Professor: Ronaldo Alves Ferreira

## 1. Introdução

A descrição do trabalho pedia que fosse implementado um script para criação de uma imagem de execução que simularia o processo de carregamento do sistema operacional pela BIOS. Dessa forma, o script deveria ler dois dados arquivos ELF, *bootblock* e *kernel*, extrair informações desses arquivos e depois gerar um arquivo final *image*.

## 2. Decisões de Implementação

Acerca das decisões de implementação, foi-se utilizado types comuns aos arquivos ELF de entrada, como por exemplo: *Elf32_Half,* que por de baixo dos panos utiliza-se de inteiros não sinalizados de 16 bits (*uint16_t*). Essa decisão de implementação foi tomada ao tentar uniformizar o projeto e evitar erros de truncamento ao utilizar o tipo inteiro (*int*). As poucas mensagens de erro foram utilizadas seguindo o padrão de microcontroladoras, como por exemplo *ESP32*, onde se é sinalizado entre colchetes o tipo do erro e posteriormente uma mensagem de identificação, por exemplo: *[ERRO] Program Header not found*. E foi implementada uma mensagem de sucesso, caso o processo ocorra de forma completa e sem erros. Os comentários tecidos ao longo do código, foram realizados seguindo a língua inglesa devido ao código inicial fornecido já estar nessa língua. O nome de variáveis foi decidido focando um código conciso e claro, evitando assim abreviações dentro do bom senso e sendo bem direto o intuito daquela variável. O tratamento para a flag –*extended* originalmente era feito na posição de índice 1 do vetor de argumentos argv, mas posteriormente foi passado para o índice 3 pois a linguagem C não tem tratamentos básicos para flags opcionais.

## 3. Dificuldades Encontradas

O simulador bochs não teve serventia alguma, carece de documentação e informações públicas, dessa forma, mesmo que houvesse tentativas de seu uso, não foi bem sucedido. Outra dificuldade encontrada, foi como a descrição do trabalho foi disponibilizada, ficou um pouco confuso o que deveria ser realizado. Mas a maior dificuldade para a realização do trabalho é a documentação extensa e em alguns momentos ambígua, por exemplo: Na

página 2-3, ao dizer sobre PT_LOAD, diz que os bytes extras deveriam ser carregados com zero, mas não fica claro quais bytes extras seriam e onde estariam localizados. Outra parte ambígua, diz que cabeçalhos de programas são opcionais, mas caso não aparecessem, como o formato deveria ser estruturado? A leitura do arquivo ELF deveria seguir lendo as Sections e apenas ignorando a falta desse cabeçalho? Um outro grande empecilho é a volatilidade da linguagem C, parece mágica, mas em alguns momentos ela funciona e de repente dá um erro de segmentação, entendo seu propósito mas pessoalmente acho difícil. Outra dificuldade encontrada foi na elaboração de um makefile.

# 4. Compilação

Para compilar o código fonte, basta inserir na shell:
make

# 5. Extra

O código do trabalho foi versionado através da ferramenta Git e pode ser encontrado no repositório remoto (https://github.com/EdmarCaixeta/buildimage).
Embora tenha sido solicitado para que não fossem enviados arquivos compilados, foi enviado o arquivo compilado já fornecido bootblock pois ele é essencial para o funcionamento do programa.