INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
NORTE DE MINAS GERAIS

Trabalho prático de Software Básico

Objetivo do trabalho:

O trabalho consiste na implementação das funções ***fopen, fclose,, fprintf*** *e* ***fscanf*** da biblioteca **libC** utilizando a linguagem de montagem (*Assembly*) da arquitetura de conjunto de instrução **x86_64**.

O tipo de dado ponto flutuante não foi abordado em aula, assim não será cobrado na implementação do trabalho.

As únicas variantes do tipo inteiro válidas como parâmetros formais das funções são "%c", "%ld e %s".

Problema:

A biblioteca padrão do C (conhecida como **libc**) é uma biblioteca de rotinas padronizada da linguagem de programação C que contém operações comuns como **tratamento de entrada/saída**, operações matemáticas e cadeia de caracteres.

A entrada e saída de dados são os meios de comunicação do programa com o ambiente externo, a forma com que o programa recebe os dados a serem processados de um dispositivo externo e devolve a este a resposta.

- A entrada pode ser feita pelo **teclado**, **mouse**, **arquivos de texto**, **arquivos de binário** — através do redirecionamento da entrada padrão, por exemplo — e de outros dispositivos físicos de entrada.
- A saída pode ser feita no **monitor**, **na impressora**, **arquivos texto**, **arquivos binário, na rede** — através do redirecionamento da saída padrão.

O **controle** da entrada e saída (E/S) de dados dos dispositivos é uma das funções principais de um **sistema operacional**. Para promover o compartilhamento seguro do uso dos recursos, **não é permitido aos processos o acesso direto aos dispositivos de entrada e saída**. Assim, cabe ao SO oferecer serviços (**chamadas de sistema** - **syscall**) que permitam ler e escrever dados.

A interação dos programas com o SO para o acesso aos dispositivos pode ocorrer enviando e recebendo **bytes** de/para **dispositivos de caractere**, ou realizando operações de arquivos em **dispositivos de bloco**

Entrada e Saída padrão

Processos comumente solicitam dados de entrada que devem ser fornecidos pelos usuários. Também é comum a exibição de dados (mensagens) para os usuários durante a execução de um programa.

Para isso, no Unix, há o conceito de arquivos padrão para entrada, saída e saída de erros, chamados, respectivamente, de STDIN, STDOUT e STDERR, esses 3 arquivos estão comumente associados ao **terminal** (***teclado ou monitor***), ou à janela em que o programa foi iniciado.

Entrada e Saída Arquivo

As informações que os programas utilizam são perdidas quando eles são finalizados ou quando o computador é desligado.

Isso porque as variáveis de um programa ficam armazenadas na memória primária, que é volátil, isto é, perde seu conteúdo.

Quando você não quer perder as informações de seu programa, tendo-as a mão para a sua próxima execução, você deve guardá-las em um ARQUIVO.

Os arquivos são estruturas especiais que ficam armazenadas na memória secundária do computador (pen drive, disco rígido, SSD...). Servem para guardar as informações enquanto um programa não está em execução, pois elas não são voláteis.

Tipos de arquivos

Em C trabalhamos com dois tipos de arquivos:

1) Arquivo texto: Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto.

Exemplos de arquivos texto: documentos de texto, código fonte C, páginas XHTML.

2) **Arquivo binário** é uma sequência de *bits* que obedece a regras do programa que o gerou.

Abertura de arquivos

Para trabalhar com um arquivo, a primeira operação necessária é abrir este arquivo. Sintaxe de abertura de arquivo:
<ponteiro*file**> = fopen("nome do arquivo","tipo de abertura");**

A função **fopen** recebe como parâmetros o nome do arquivo a ser aberto e o tipo de abertura a ser realizado. Após de aberto, realizamos as operações necessárias e fechamos o arquivo. Para fechar o arquivo usamos a função fclose. Sintaxe de fechamento de arquivo
fclose<ponteiro*file**>;**

A função **fprintf()** funciona como a função printf(). A diferença é que a saída de fprintf() é um arquivo e não a tela do computador. Sintaxe:
int fprintf (FILE *fp,char *str,...);

A função fscanf() funciona como a função scanf(). A diferença é que fscanf() lê de um arquivo e não do teclado do computador. Sintaxe:
Int fscanf (FILE *fp,char *str,...);

O trabalho consiste na implementação das funções ***fopen, fclose, `fprintf` e fscanf*** da biblioteca **libC** da biblioteca **libC** utilizando a linguagem de montagem (*Assembly*) da arquitetura de conjunto de instrução **x86_64**.

A Implementação das funcões ***fopen, fclose, fprintf e fscanf*** deve seguir o padrão de parâmetros formais e associação de registradores. Para cada função será necessário associar uma área da pilha, chamada de registro de ativação.
funcaoX:

pushq %rbp

```
movq  %rsp, %rbp

# corpo da funcao

movq ??, %rax       #quantidade de caracteres exibidos
popq   %rbp
ret
```

Dúvidas: esclarecimentos podem ser solicitados através de e-mail, *classroom* ou *whatsapp*.

O trabalho deve ser implementando em **grupo de três alunos**. Não é permitido discutir os aspectos do algoritmo em *Assembly* e soluções adotadas com outros colegas, e é terminantemente proibido compartilhar código ou trecho de código.

```
if( trabalho == plágio )
{
  nota = 0.000000; /* para quem fornece o código e para quem copiou*/
}
else
{
  nota = 30 – erros;
}
```

Regra para entrega do Trabalho:

- Data da entrega: 15/12/2023

- **Por e-mail:** caribe.souza@ifnmg.edu.br
- **Assunto**: SB: Trabalho I
- **Corpo do E-mail**: nome completo dos componentes
- **Anexo**: código-fonte (zip):

  Nome do código fonte: SB.s

  Nome do arquivo.zip: deve ser composto por **TRAB_SB_** e pelos nomes dos alunos, por exemplo: **TRAB_SB**_JoaoSouza_Ana_Silva***.zip***