OBJETIVO: Exercitar a manipulação de arquivos/dados binários através da leitura de um arquivo ZIP.

QUESTÃO ÚNICA

Assim como todos os padrões de arquivos binários, o arquivo ZIP segue um padrão bem estabelecido que, se seguido, permite acessar os dados dos seus arquivos compactados e, até mesmo, descompactá-los. Mais especificamente, um arquivo ZIP possui o seguinte formato:

Cab. Arq. 1	Arquivo 1 (dados compactados)	Cab. Arq. 2	Arquivo 2 (dados compactados)	Cab. Arq. 3	Arquivo 3 (dados compactados)	...	Cab. Arq. *n*	Arquivo *n* (dados compactados)	*Central Directory*	Fim

Neste trabalho, nós não iremos descompactar os arquivos dentro do ZIP, mas apenas imprimir as informações deles. Tais informações estão especificadas no cabeçalho local de cada arquivo (cinza claro na figura acima). Este cabeçalho, segue o padrão definido ao lado (fonte: wikipedia).

Deste cabeçalho, você deverá imprimir (para cada arquivo), o nome dele (*file name*), o tamanho compactado (*compressed size*) e o tamanho descompactado (*uncompressed size*).

Para "navegar" entre os arquivos dentro do ZIP, você precisará das informações do tamanho do arquivo compactado (*compressed size*), do tamanho do nome do arquivo (*file name length – n*) e do tamanho do campo extra (*extra field length – m*). Desta forma, se o seu ponteiro aponta para o início de um arquivo no ZIP, para ir para o próximo arquivo basta somar este ponteiro com *30 + n + m + compressed size,* onde 30 é o tamanho do cabeçalho (bytes) até o tamanho do nome do arquivo (some os números de bytes ao lado), *n* é o tamanho do nome do arquivo, *m* é o tamanho do campo extra e "*compressed size*" é o tamanho do arquivo compactado. Fazendo isso, o seu ponteiro irá apontar para o início do próximo arquivo.

Local File Header (Cabeçalho do Arquivo)

Byte Offset	Num. Bytes	Descrição
0	4	Local file header signature (0x04034b50)
4	2	Version needed to extract (minimum)
6	2	General purpose bit flag
8	2	Compression method
10	2	File last modification time
12	2	File last modification date
14	4	CRC-32
18	4	Compressed size
22	4	Uncompressed size
26	2	File name length (*n*)
28	2	Extra field length (*m*)
30	*n*	File name
30+*n*	*m*	Extra field

Mas como pegar essas informações do cabeçalho do arquivo e usá-las em C? Simples, basta criar uma estrutura (*struct*) que reflita as informações do cabeçalho e fazer um *cast* do ponteiro. Vamos começar pela estrutura …

```
struct zip_file_hdr {
    int signature;
    short version;
    short flags;
    short compression;
    short mod_time;
    short mod_date;
    int crc;
    int compressed_size;
    int uncompressed_size;
    short name_length;
    short extra_field_length;
} __attribute__ ((packed));
```

Note que que nos campos que possuem 4 bytes (32 bits), usamos o tipo "int", e nos campos que possuem 2 bytes (16 bits), usamos o "short". Note também que não definimos os campos "File Name" e "Extra Field", pois eles possuem tamanhos variáveis.

Por último, note que usamos o modificador "`__attribute__ ((packed))`" no final da estrutura. Isso é necessário para evitar que o compilador modifique a estrutura na memória para fazer alinhamentos com o objetivo de melhorar a performance (isso iria incluir mais bytes na estrutura e gerar resultados errados).

Com base nesta estrutura, basta você alocar memória para ela (`malloc`) e escrever as informações do arquivo ZIP na memória alocada (usando o `fread`). A partir daí, você pode acessar os dados da estrutura normalmente. Exemplo:

```
FILE *zip_file = fopen(argv[1], "rb");
struct zip_file_hdr *file_hdr = malloc(sizeof(struct zip_file_hdr));
fread(file_hdr, sizeof(struct zip_file_hdr), 1, zip_file);
```

A partir deste código, você poderá acessar o file_hdr usando os campos da estrutura zip_file_hdr:

```
printf("  --> Assinatura: %.8x", file_hdr->signature); // Imprime hexadecimal
printf("  --> Tamanho Compactado: %d\n", file_hdr->compressed_size);
```

Sua missão, é abrir um arquivo ZIP, cujo nome será passado pela linha de comando (argv[1]), e listar o nome, tamanho compactado e tamanho descompactado de todos os arquivos dentro dele.

Exemplo de Execução:

```
$ ./read_zip teste.zip
Arquivo 1 ..
  --> Nome do Arquivo: grumpy-cat.jpeg
  --> Tamanho Compactado: 298342
  --> Tamanho Descompactado: 298418
Arquivo 2 ..
  --> Nome do Arquivo: skyrim_environment_1920x1200.jpg
  --> Tamanho Compactado: 1032602
  --> Tamanho Descompactado: 1033308
```

Atenção:

Como existem algumas variações na compactação dos arquivos e para não entrarmos em detalhes mais avançados do formato, você pode usar o arquivo abaixo para testar seu código (a saída será como no exemplo acima).

Arquivo de teste: http://bit.ly/lpa-lab6-zip

Dicas (algoritmo):

1 Crie a estrutura como mostrado;
2 Abra o arquivo ZIP (fopen – ver exemplo);
3 Aloque uma memória para a estrutura criada (malloc – ver exemplo);
4 Enquanto não for o fim do arquivo ZIP (feof)
  4.1 Leia do arquivo a quantidade de bytes da estrutura e salve na memória alocada (fread – ver exemplo). Note que ao fazer isso, os dados do arquivo ZIP irão para a estrutura alocada e o "ponteiro interno do arquivo" andará devido ao fread e irá parar logo antes do nome do arquivo compactado ("`File name`", na tabela anterior), pois ele não faz parte do *struct*.
  4.2 Verifique se a assinatura do cabeçalho é 0x04034b50 (cabeçalho de um arquivo – ver tabela e exemplo). Se sim:
    4.2.1 Aloque memória para o nome do arquivo (malloc de uma string) de acordo com o tamanho do nome do arquivo, disponível nos dados da estrutura lida anteriormente. Inclua um caractere a mais para incluir o final de string do C.
    4.2.2 Leia do arquivo ZIP o nome do arquivo (fread) de acordo com o tamanho do nome do arquivo, disponível nos dados da estrutura lida anteriormente
    4.2.3 Adicione o caractere \0 ao final da string com o nome do arquivo
    4.2.4 Imprima o nome do arquivo (printf)
    4.2.5 Imprima o tamanho compactado (printf – ver exemplo)
    4.2.6 Imprima o tamanho descompactado (printf)
    4.2.7 Libere a memória da string alocada (free)
    4.2.8 Faça o ponteiro do arquivo ZIP apontar para o próximo arquivo dentro dele:

```
fseek(zip_file, file_hdr->extra_field_length +
                                    file_hdr->compressed_size, SEEK_CUR);
```

  4.3 Se a assinatura não for 0x04034b50, saia.
5 Libere a memória da estrutura alocada
6 Feche o arquivo ZIP

ENTREGA DO LABORATÓRIO

Envie, até 20/05/2021 às 23:59, o código-fonte para horacio@icomp.ufam.edu.br com o assunto "Entrega do 6o Laboratório de LPA".