Trabalho VINAC - Programação 2
Autor: Matheus Gastal Magalhães - GRR20244620

Lista de arquivos:
-Lz.h - Header do LZ que faz a compressão e descompressão de arquivos.
-Lz.c - Arquivo C que executa compressão e descompressão, disponibilizado pelos professores.
funções.h - Header contendo protótipo de funções que serão chamadas na main, além da struct diretório e struct membro.
funções.c - Arquivo C contendo funções insere_sem_compressão, insere_comprimido, move_membro, remove_arquivo, lista_informacoes e extrai_arquivos. Funções implementadas para as funções -ip, -ic, -m, -r, -c e -x. Além disso, possui funções auxiliares que ajudam na implementação e clareza do código das funções principais, são essas: acha_iguais, imprime_info, get_tamanho, conta_membros_no_archive, transfere_info e mover.
main.c - Implementação do main para receber argumentos e definir quais funções chamar, essa função foi basicamente toda feita em sala pelo professor David Menotti, portanto, ela é bem similar ao que o professor passou, só alterei algumas impressões durante os processos para deixar mais amigável para o usuário.

Funções Aux:

int acha_iguais(int i, char **arquivos, struct diretorio dir):
Verifica se o arquivo na posição i da lista arquivos já existe no diretório dir. Retorna o índice do membro existente ou -1 se não existir. Evita duplicação e permite substituição.

void imprime_info(struct membro mb):
Imprime as informações de um membro do archive no formato amigável (nome, UID, tamanhos, data e offset). Usado na listagem dos arquivos.

long int get_tamanho(FILE *f):
Retorna o tamanho em bytes de um arquivo aberto (FILE*). É usado para determinar tanto o tamanho de membros quanto do próprio archive.

int conta_membros_no_archive(const char *archive):
Lê o final do archive para descobrir quantos membros estão armazenados, baseando-se na contagem gravada junto ao diretório. Retorna esse número ou -1 em caso de erro.

void transfere_info(struct membro *mb, char *nome_arquivo, int index, long int tam_original, long int tam_disco, long int loc):
Preenche os campos de uma struct membro com as informações básicas do arquivo: nome, UID, tamanhos, data de modificação, offset e ordem. Usado ao inserir novos membros no diretório.

void mover(FILE *archive, long origem, long destino, long tamanho):
Move um trecho de dados dentro do arquivo archive do offset origem para o destino, copiando tamanho bytes. É essencial para reorganizar dados de membros em substituições com tamanhos diferentes.

Funções principais:

insere_sem_compressao(char *archive, char **arquivos, int num):
Essa função é responsável por inserir arquivos no archive sem aplicar compressão. Primeiramente, ela tenta abrir o archive em modo leitura/escrita (rb+). Caso ele não exista, é criado em modo escrita/leitura (wb+). Em seguida, a função verifica se já há membros existentes no archive. Caso haja, o diretório é carregado da parte final do arquivo e a função calcula o maior offset presente, que indica onde os novos dados podem começar a ser escritos. Em seguida, o diretório é realocado para comportar os novos membros. Para cada arquivo da lista arquivos, a função abre e lê o conteúdo completo em um buffer. Se o arquivo já existia no archive (identificado via acha_iguais), a função verifica se o tamanho mudou. Se não mudou, sobrescreve diretamente. Caso o novo tamanho seja diferente, os dados seguintes são movidos (para frente ou para trás) usando a função mover e os offsets dos demais membros são atualizados. Por fim, o novo conteúdo é escrito e o diretório atualizado. Se o arquivo não existia, o conteúdo é simplesmente adicionado ao final, e as informações do novo membro são preenchidas com transfere_info. Ao final, o diretório completo é regravado na parte final do archive.

lista_informacoes(char *archiver):
Esta função lista todos os membros presentes no archive. Ela começa abrindo o arquivo em modo leitura e verificando a quantidade de membros com conta_membros_no_archive. Caso o archive esteja vazio, exibe uma mensagem. Em seguida, carrega o diretório e utiliza a função imprime_info para exibir os dados de cada membro.

move_membro(char *archive, char *nome_mover, char *nome_target):
A função move_membro permite reorganizar a ordem dos arquivos dentro do archive. Ela abre o archive, carrega o diretório e identifica os índices do membro a ser movido e do membro de destino (target). Em seguida, remove o membro da posição original e o insere logo após o target (ou no início, se target for NULL), atualizando os campos ord. Após essa reorganização lógica, ela cria um arquivo temporário e reorganiza fisicamente os dados dos membros, copiando em nova ordem e atualizando a localização do membro. Por fim, o conteúdo do temporário é escrito de volta no archive e o diretório atualizado com os novos valores.

insere compactado(char *archive, char **arquivos, int num):
Esta função insere arquivos no archive utilizando compressão LZ sempre que o tamanho após a compressão for menor do que o tamanho original. Após abrir ou criar o archive, ela carrega o diretório existente e calcula o maior offset usado. Para cada arquivo, a função lê o conteúdo original e tenta comprimir usando LZ_Compress. Se o resultado for menor que o original, o arquivo é armazenado comprimido. Caso contrário, o arquivo é armazenado sem compressão. Se o arquivo já existir no archive, os dados são substituídos. Caso o tamanho seja diferente do anterior, os dados subsequentes são movidos usando a função mover, e os offsets atualizados para manter a integridade. A lógica de substituição é a mesma da versão sem compressão, aproveitando a estrutura do diretório e a função mover para reorganizar os dados corretamente. Para novos arquivos, os dados são adicionados ao final e as informações são preenchidas com transfere_info. Ao final, o diretório atualizado é escrito na parte final do arquivo archive.

remove_arquivos(char *archive, char **arquivos, int num):
Essa função remove os arquivos especificados do archive. Ela carrega o diretório, filtra os membros a serem mantidos e copia seus dados para um arquivo temporário. Durante essa cópia, atualiza os offsets e ordens dos arquivos restantes. Após isso, sobrescreve completamente o archive com os dados reorganizados e regrava o diretório. Essa abordagem garante que o espaço dos arquivos removidos seja efetivamente liberado do archive.

extrai_arquivos(char *archive, char **arquivos, int num)
A função extrai_arquivos permite restaurar os arquivos originais a partir do archive. Se nenhum nome for especificado, extrai todos os membros. Para cada arquivo, a função verifica se está armazenado comprimido (comparando tam_disco e tam_original). Se estiver comprimido, aplica LZ_Uncompress. Em seguida, escreve os dados em um novo arquivo no disco, com o mesmo nome original.

Dificuldades encontradas durante o desenvolvimento do projeto:
Honestamente, foi um dos projetos mais desafiadores que já desenvolvi, não pela dificuldade de entender o que fazer, mas pela falta de contato com as funções usadas para manipulação de arquivos. Começar a desenvolver foi muito desafiador, no entanto, após a inserção sem compressão, tudo se tornou mais familiar. Sobre dificuldades técnicas encontradas, encontrei principalmente na função inserir_sem_compressão, devido aos casos que precisavam ser tratados, inserção repetida, com tamanho maior ou menor, primeira inserção. Essas condições em termos de entendimento não eram difíceis, porém traduzir a ideia geral para código se mostrou desafiador.

Bugs conhecidos:
Não há verificação para nomes maiores que 1024 caracteres no campo nome. Arquivos com nomes longos podem causar overflow de buffer ou comportamento indefinido.

Agradecimentos e conclusão:
Agradeço ao professor Diego Addan pelo apoio prestado durante todo desenvolvimento do trabalho. Esse trabalho foi bem desafiador e interessante, em um primeiro momento, sua usabilidade era um pouco abstrata para mim, porém, durante o desenvolvimento as coisas foram ficando mais claras e mais tranquilas, agradeço aos meus colegas que desbravaram essas dificuldades comigo, discutindo ideias e soluções para os problemas encontrados.