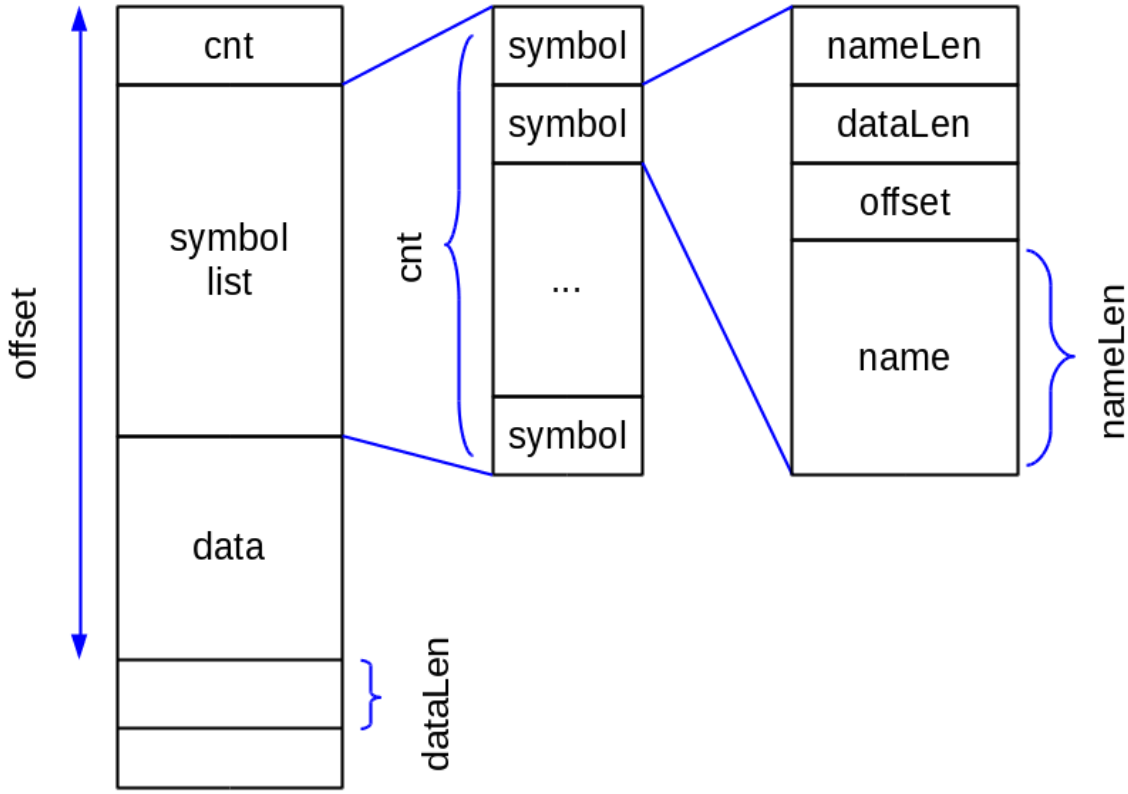
Úkolem je vytvořit C/C++ funkci, která dokáže fungovat jako jednoduchý linker. Základní informace o fungování linkeru byly zmíněné v poslední přednášce a proseminářích PA1, pokud jste tyto informace zoptimalizovali, doporučujeme si je před řešením této úlohy obnovit.
V této úloze uvažujeme velmi zjednodušený problém linkování. Omezení jsou následující:

- na vstupu jsou vždy 2 soubory,
- uvažujeme pouze exportované symboly, neuvažujeme import symbolů,
- neuvažujeme segmenty,
- výstupní formát je shodný jako vstupní formát.

Vstupní soubor k linkování obsahuje symboly. Každý symbol je identifikován svým jménem. Jména jsou unikátní, pokud se vyskytne duplicita v rámci jednoho souboru nebo pokud se stejné jméno symbolu objeví v obou linkovaných vstupních souborech, je to považované za chybu. V souboru ukládáme odděleně informaci o délce jména symbolu, jméno nemá explicitně omezenou délku, může obsahovat libovolné bajty. Ke každému symbolu se váže nějaký blok dat (například zkompilovaný kód).
Struktura souboru je zachycena na obrázku:



- soubor začíná celým číslem (4B) cnt, toto číslo udává počet symbolů v souboru,
- za tímto počtem je seznam jednotlivých symbolů,
- za seznamem symbolů jsou data, která odpovídají jednotlivým symbolům.
- Symbol je definován pomocí 3 celých čísel (každé má velikost 4B) a jména:
    - nameLen udává počet bajtů jména symbolu,
    - dataLen udává velikost (počet bajtů) datového bloku, který přísluší symbolu,
    - offset udává pozici datového bloku v souboru (počet bajtů od počátku souboru).
    - za touto trojicí celých čísel následují bajty, která tvoří jméno symbolu. Počet těchto bajtů (tedy délka jména symbolu) je daná hodnotou nameLen.
- Pořadí symbolů v seznamu není nijak omezeno.
- Bloky dat v datové části jsou odkazované jednotlivými symboly. Obsah není pro implementaci zajímavý, tato data je pouze potřeba beze změn přenést do výsledného souboru.
- Pořadí bloků v datové části není nijak omezeno (pouze je potřeba, aby souhlasily odkazy z jednotlivých symbolů).

Požadovaná funkce má následující rozhraní:

```
bool linkFiles ( const char * srcName1,
                 const char * srcName2,
                 const char * dstName );
```

srcName1, srcName2
jsou ASCIIZ řetězce se jmény zdrojových souborů. Funkce tyto soubory může číst, nesmí jej ale modifikovat.
dstName
je ASCIIZ řetězec se jménem cílového souboru (spojení vstupů). Funkce tento soubor vytváří, uloží do něj symboly a jím příslušná data z obou vstupních souborů. Data ze vstupních souborů nestačí pouze zkopírovat, je potřeba i správně přepočítat odkazy.
návratová hodnota
true pro úspěch, false pro neúspěch. Za neúspěch považujte:

- chybu při práci se soubory (nelze číst, zapsat, neexistuje, ...),
- chybný formát vstupního souboru (nesprávný obsah hlavičky, nedostatek dat, neplatný odkaz na data symbolu,...),
- duplicita ve jménech symbolů.

Poznámky:

- Pečlivě ošetřujte souborové operace. Testovací prostředí úmyslně testuje Vaši implementaci pro soubory neexistující, nečitelné nebo soubory s nesprávným datovým obsahem.
- Jména souborů jsou řetězce, které nemusíte nijak kontrolovat. Názvy souborů ani přípony nejsou nijak omezené, podstatné je pouze, zda lze soubory daného jména otevřít a číst/zapisovat.
- Při implementaci lze použít C i C++ rozhraní pro práci se soubory, volba je na Vás.
- V přiloženém archivu najdete sadu testovacích souborů a jim odpovídající výstupy. Dále v přiloženém zdrojovém souboru naleznete ukázkovou funkci main, která spouští konverze na ukázkových souborech. Do tohoto zdrojového souboru můžete vložit Vaši implementaci, testovat ji a odevzdat ji na Progtest. Pokud chcete upravený zdrojový soubor odevzdávat, zachovejte v něm bloky podmíněného překladu.
- Předpokládáme, že soubory jsou vždy kódované v kódování little-endian (stejný jako HW, na kterém běží program).
- Rozdělte řešení do více funkcí.
- Pro zpracování hodnot se známou velikostí (například 4 bajty rozměru obrázku v hlavičce) se hodí datové typy s garantovanou velikostí (stdint.h, např. datové typy int32_t, uint32_t).
- Úloha obsahuje bonusové testy. Tyto testy zkouší časovou a paměťovou efektivitu Vaší implementace. Na vstupu jsou dlouhé soubory s mnoha symboly, max. doba běhu a velikost dostupné paměti je významně omezena.
- Ukázkový formát souborů je navržený jako velmi zjednodušený pouze pro tuto úlohu. Skutečný formát .obj souborů je jiný.