

IJC: DU2

Jazyk C

DU2

22.3.2022

Domácí úkol č.2

Termín odevzdání: 19.4.2022

(Max. 15 bodů)

1) (max 5b)

a) V jazyku C napište program "tail.c", který ze zadaného vstupního souboru vytiskne posledních 10 řádků. Není-li zadán vstupní soubor, čte ze stdin. Je-li programu zadán parametr -n číslo, bude se tisknout tolik posledních řádků, kolik je zadáno parametrem 'číslo' (bez znaménka). Případná chybová hlášení tiskněte do stderr. Příklady:

```
tail soubor
tail -n 20 <soubor
```

[Poznámka: výsledky by měly být +-stejně jako u POSIX tail]

Je povolen implementační limit na délku řádku (např. 4095 znaků), v případě prvního překročení mezi hlase chybu na stderr (řádně otestujte) a pokračujte se zkrácenými řádky (zbytek řádku přeskočit/ignorovat).

2) (max 10b)

Přepište následující C++ program do jazyka ISO C

```
// wordcount-.cc
// Použijte: g++ -O2
// Příklad použití STL kontejneru unordered_map<>
// Program počítá četnost slov ve vstupním textu,
// slovo je cokoli oddělené "bílým znakem"

#include <string>
#include <iostream>
#include <unordered_map>

int main() {
    using namespace std;
    unordered_map<string,int> m; // asociativní pole
    // mapuje klíč/string na hodnotu/int
    string word;
    while (cin >> word) // čtení slova (jako scanf "%s", ale bezpečné)
        m[word]++;      // počítání výskytů slova (zvýší hodnotu pro
                        // zadaný klíč/slovo pokud záznam existuje,
                        // jinak vytvoří nový záznam s hodnotou 0 a
                        // tu operace ++ zvýší na 1)

    for (auto &mi: m) // pro všechny prvky kontejneru m
        cout << mi.first << "\t" << mi.second << "\n";
        // klíč/slovo      hodnota/počet
    // prvky kontejneru typu "map" jsou dvojice (klíč,hodnota)
}
```

Výstupy programů musí být pro stejný vstup stejné (kromě pořadí a příliš dlouhých slov).

Výsledný program se musí jmenovat "wordcount.c".

Implementujte tabulku s rozptýlenými položkami (hash table) - viz dále.

Veškeré operace s tabulkou budou v samostatné knihovně (vytvořte statickou i dynamickou/sdílenou verzi). V knihovně musí být prakticky každá funkce ve zvláštním modulu -- to například umožní případnou výměnu `htab_hash_function()` ve vašem staticky sestaveném programu. (V dynamicky sestaveném programu je to možné vždy.) Vyzkoušejte si to: definujte svoji verzi `htab_hash_function()` v programu s podmíněným překladem -- použijte `#ifdef HASHTEST`.

Knihovna s tabulkou se musí jmenovat "libhtab.a" (na Windows je možné i "htab.lib") pro statickou variantu, "libhtab.so" (na Windows je možné i "htab.dll") pro sdílenou variantu a rozhraní "htab.h".

Podmínky:

- Implementace musí být dynamická (malloc/free) a musíte zvládnout správu paměti v C (použijte valgrind nebo jiný podobný nástroj).
- Vhodná rozptylovací funkce pro řetězce je podle literatury (<http://www.cse.yorku.ca/~oz/hash.html> - varianta sdbm):

```
size_t htab_hash_function(const char *str) {
    uint32_t h=0;      // musí mít 32 bitů
    const unsigned char *p;
    for(p=(const unsigned char*)str; *p!='\0'; p++)
        h = 65599*h + *p;
    return h;
}
```

její výsledek modulo `arr_size` určuje index do tabulky:
`index = (htab_hash_function("mystring") % arr_size);`
 Zkuste použít i jiné podobné funkce a porovnejte efektivitu.

- Tabulka je (pro knihovnu privátní) struktura obsahující pole seznamů, jeho velikost a počet položek tabulky v následujícím pořadí:

```
+-----+
| size   | // aktuální počet záznamů [(key,data),next]
+-----+
| arr_size | // velikost pole ukazatelů (počet položek)
+-----+
| arr_ptr | // ukazatel na dynamicky alokované pole ukazatelů
+-----+
|
| v
+----+
|ptr|-->[(key,data),next]-->[(key,data),next]-->[(key,data),next]--|
+----+
|ptr|--|
+----+
|ptr|-->[(key,data),next]-->[(key,data),next]--|
+----+
|ptr|--|
+----+
```

(V obrázku platí velikost `.arr_size==4` a počet položek `.size==5`.)
 Položka `.arr_size` je velikost dynamicky alokovaného pole ukazatelů.
 Paměť pro strukturu se dynamicky alokuje a ukazatel na ni se používá pro identifikaci tabulky. V programu zvolte vhodnou minimální (počáteční) velikost pole a v komentáři zdůvodněte vaše rozhodnutí.
 Rozhraní knihovny obsahuje jen neúplnou deklaraci struktury, definice je uživateli knihovny skryta (jde o formu zapouzdření - "encapsulation").

- Napište funkce podle následujícího hlavičkového souboru (API):

```
=====
// htab.h -- rozhraní knihovny htab (řešení IJC-DU2)
// Licence: žádná (Public domain)

// následující řádky zabrání násobnému vložení:
#ifndef __HTAB_H__
```

```

#define __HTAB_H__

#include <string.h>    // size_t
#include <stdbool.h>   // bool

// Tabulka:
struct htab;    // neúplná deklarace struktury - uživatel nevidí obsah
typedef struct htab htab_t;    // typedef podle zadání

// Typy:
typedef const char * htab_key_t;    // typ klíče
typedef int htab_value_t;    // typ hodnoty

// Dvojice dat v tabulce:
typedef struct htab_pair {
    htab_key_t    key;    // klíč
    htab_value_t    value;    // asociovaná hodnota
} htab_pair_t;    // typedef podle zadání

// Rozptylovací (hash) funkce (stejná pro všechny tabulky v programu)
// Pokud si v programu definujete stejnou funkci, použije se ta vaše.
size_t htab_hash_function(htab_key_t str);

// Funkce pro práci s tabulkou:
htab_t *htab_init(size_t n);    // konstruktor tabulky
size_t htab_size(const htab_t * t);    // počet záznamů v tabulce
size_t htab_bucket_count(const htab_t * t);    // velikost pole
void htab_resize(htab_t *t, size_t newn);    // změna velikosti pole
// (umožňuje rezervaci místa)

htab_pair_t * htab_find(htab_t * t, htab_key_t key);    // hledání
htab_pair_t * htab_lookup_add(htab_t * t, htab_key_t key);

bool htab_erase(htab_t * t, htab_key_t key);    // ruší zadaný záznam

// for_each: projde všechny záznamy a zavolá na ně funkci f
// Pozor: f nesmí měnit klíč .key ani přidávat/rušit položky
void htab_for_each(const htab_t * t, void (*f)(htab_pair_t *data));

void htab_clear(htab_t * t);    // ruší všechny záznamy
void htab_free(htab_t * t);    // destruktory tabulky

#endif // __HTAB_H__
=====

```

Hlavičkový soubor můžete celý převzít (je "Public domain").

- Stručný popis základních funkcí:

t=htab_init(num)	konstruktor: vytvoření a inicializace tabulky num = počet prvků pole (.arr_size)
size_t s=htab_size(t)	vrátí počet prvků tabulky (.size)
size_t n=htab_bucket_count(t)	vrátí počet prvků pole (.arr_size)
htab_resize(t,newnum)	Změní velikost alokovaného pole a přesune položky z původních seznamů. Pokud alokace pole selže, nemění nic.
ptr=htab_find(t,key)	vyhledávání - viz dále
ptr=htab_lookup_add(t,key)	vyhledávání+přidání - viz dále
b=htab_erase(t,key)	zrušení záznamu se zadaným klíčem (úspěch:true) Když průměrná délka seznamů klesne pod vámi definovaný limit AVG_LEN_MIN provede operaci htab_resize na poloviční velikost.
htab_for_each(t,funkce)	projde všechny záznamy, na každý zavolá funkci

(pozor na možné změny tabulky!)

htab_clear(t) zrušení všech položek, tabulka zůstane prázdná

htab_free(t) destruktor: zrušení tabulky (volá htab_clear())

kde t,t2 je ukazatel na tabulku (typu htab_t *),
 b je typu bool,
 ptr je ukazatel na záznam (položku tabulky {klíč,hodnota}),

- Vhodně zvolte typy parametrů funkcí (včetně použití const).
- Záznam [(key,value),next] je typu
 struct htab_item
 a obsahuje položky:
 next ... ukazatel na další záznam
 struct htab_pair ... veřejná struktura s položkami:
 key ukazatel na dynamicky alokovaný řetězec,
 value ... asociovaná data = počet výskytů
 Tento záznam je definován v privátním hlavičkovém souboru pro všechny
 moduly tabulky a není dostupný při použití knihovny ("Opaque data type").
 Uživatel používá ukazatel na vnořenou strukturu htab_pair_t.
- Funkce
 htab_pair_t *htab_find(htab_t *t, htab_key_t key);
 V tabulce t vyhledá záznam odpovídající řetězci key a
 - pokud jej nalezne, vrátí ukazatel na záznam
 - pokud nenalezne, vrátí NULL
- Funkce
 htab_pair_t htab_lookup_add(htab_t *t, htab_key_t key);
 V tabulce t vyhledá záznam odpovídající řetězci key a
 - pokud jej nalezne, vrátí ukazatel na záznam
 - pokud nenalezne, automaticky přidá záznam a vrátí ukazatel
 Když průměrná délka seznamů přesáhne vámi definovaný limit
 AVG_LEN_MAX provede operaci htab_resize na dvojnásobnou velikost.
 Poznámka: Dobře promyslete chování této funkce k parametru key.
 Poznámka: podobně se chová C++ operator[] pro std::unordered_map
- Když htab_init nebo htab_lookup_add nemohou alokovat paměť,
 vrací NULL (a uživatel musí testovat výsledek těchto operací)
 Poznámka: C++ na to používá výjimky ("exceptions").

Napište funkci

```
int read_word(char *s, int max, FILE *f);
```

která čte jedno slovo ze souboru f do zadaného pole znaků
 a vrátí délku slova (z delších slov načte prvních max-1 znaků,
 a zbytek přeskočí). Funkce vrátí EOF, pokud je konec souboru.
 Umístěte ji do zvláštního modulu "io.c" (nepatří do knihovny).
 Poznámka: Slovo je souvislá posloupnost znaků oddělená isspace znaky.

Omezení: řešení v C bude tisknout jinak uspořádaný výstup
 a je povoleno použít implementační limit na maximální
 délku slova (např. 127 znaků), delší slova se ZKRÁTÍ a program
 při prvním delším slovu vytiskne varování na stderr (max 1 varování).

Poznámka: Vhodný soubor pro testování je například seznam slov
 v souboru /usr/share/dict/words
 nebo texty z <http://www.gutenberg.org/>
 případně výsledek příkazu: "seq 1000000 2000000|shuf"

(10b)

Použijte implicitní lokalizaci (= nevolat setlocale()). Zamyslete se nad tím,
 jaké problémy by přineslo použití UTF-8 při zapnuté lokalizaci s tímto
 dnes běžně používaným kódováním.

Napište soubor Makefile tak, aby příkaz make vytvořil programy "tail", "wordcount", "wordcount-dynamic" a knihovny "libhtab.a", "libhtab.so" (nebo "htab.dll" atd.).

Program "wordcount" musí být staticky sestaven s knihovnou "libhtab.a".

Program "wordcount-dynamic" musí být sestaven s knihovnou "libhtab.so". Tento program otestujte se stejnými vstupy jako u staticky sestavené verze.

Vyzkoušejte si různé hodnoty `AVG_LEN_{MIN|MAX}` a odhadněte jejich rozumnou velikost pro optimální výkonnostní charakteristiky (`htab_resize` se nesmí provádět příliš často). Zamyslete se, jak se zhorší průměrná výpočetní náročnost (složitost) operace vložení dalšího záznamu vzhledem k nutnosti provádět `htab_resize`.

Porovnejte efektivitu obou (C i C++) implementací (viz např. příkaz `time`) a zamyslete se nad výsledky (pozor na vliv vyrovnávacích pamětí atd.) Také si zkuste překlád s optimalizací i bez ní (`-O2`, `-O0`) a porovnejte efektivitu pro vhodný vstup.

Poznámky:

- pro testy `wordcount-dynamic` na linuxu budete potřebovat nastavit `LD_LIBRARY_PATH="."` (viz "man ld.so" a odpovídající přednáška)
- Čtete pokyny pro vypracování domácích úkolů (viz dále)

Obecné pokyny pro vypracování domácích úkolů

- * Pro úkoly v jazyce C používejte ISO C11 (soubory *.c)
Pro úkoly v jazyce C++ používejte ISO C++17 (soubory *.cc)
Použití nepřenositelných konstrukcí v programech není dovoleno.
- * Úkoly zkontrolujte překladačem například takto:

```
gcc -std=c11 -pedantic -Wall -Wextra priklad1.c
g++ -std=c++17 -pedantic -Wall priklad.cc
```

Místo gcc můžete použít i jiný překladač - podle vašeho prostředí.
V souvislosti s tím napište do poznámky na začátku souboru jméno a verzi překladače, kterým byl program přeložen (implicitní je GCC `g++ --version` na počítači merlin).
- * Programy pište, pokud je to možné, do jednoho zdrojového souboru. Dodržujte předepsaná jména souborů.
- * Na začátek každého souboru napište poznámku, která bude obsahovat jméno, fakultu, označení příkladu a datum.
- * Úkoly je nutné zabalit programem zip takto:

```
zip xnovak99.zip *.c *.cc *.h Makefile
```

Jméno `xnovak99` nahradíte vlastním. Formát souboru bude ZIP. Archiv neobsahuje adresáře. Každý si zkontroluje obsah ZIP archivu jeho rozbalením v prázdném adresáři a napsáním "make".
- * Posílejte pouze nezbytně nutné soubory -- ne *.EXE !
- * Řešení se odevzdává elektronicky v IS FIT
- * Úkoly neodevzdané v termínu (podle WIS) budou za 0 bodů.
- * Opsané úkoly budou hodnoceny 0 bodů pro všechny zúčastněné a to bez výjimky (+ bonus v podobě návštěvy u disciplinární komise).

Poslední modifikace: 22. March 2022

Pokud naleznete na této stránce chybu, oznamte to dopisem na adresu peringer AT fit.vutbr.cz