Medvědův průvodce po LibUCW

Martin Mareš
mj@ucw.cz

Katedra Aplikované Matematiky
MFF UK Praha

2011

Co je LibUCW

Knihovna pro zpříjemnění života Céčkových programátorů :)

Součásti:

- Generické datové struktury
- Alokátory paměti
- Modulární bufferované I/O
- Parser konfiguračních souborů
- Stringové operace
- Třídění interní i externí
- Hlavní smyčka (abstrakce nad `select()` a spol.)
- Drobnosti: makra `MIN`, `MAX`, `COMPARE`, `ARRAY_SIZE`

Generické datové struktury

- ⟨ucw/slist.h⟩, ⟨ucw/clist.h⟩ – spojové seznamy
- ⟨ucw/gary.h⟩ – rostoucí pole
- ⟨ucw/heap.h⟩ – binární halda
- ⟨ucw/binheap.h⟩ – binomiální halda
- ⟨ucw/redblack.h⟩ – červeno-černé stromy
- ⟨ucw/hashtable.h⟩ – hashovací tabulky
- ⟨ucw/trie.h⟩ – trie (množiny řetězců)
- ⟨ucw/bitsig.h⟩ – pravděpodobnostní množiny

Generické hashovací tabulky

```
struct pair {
  int key;
  int data;
};
#define HASH_NODE struct pair    // Typ
#define HASH_PREFIX(x) pair_##x // Jméno
#define HASH_KEY_ATOMIC key      // Klič
#define HASH_ATOMIC_TYPE int
#define HASH_ZERO_FILL           // Alokace
#define HASH_WANT_LOOKUP         // Operace
#define HASH_WANT_REMOVE
#include <ucw/hashtable.h>
```

Vznikne:

`pair_init()`, `pair_cleanup()`,
`pair_lookup()`, `pair_remove()`

Alokátory paměti

Alokátory paměti:

- Wrappery: `xmalloc()`, `xfree()`, `xrealloc()`
- Memory pooly ⟨ucw/mempool.h⟩
  - Postupná alokace, jednorázové uvolnění
  - Návrat do uschovaného stavu (zásobník)
  - Natahovací buffer pro postupnou konstrukci (třeba řetězců)
- Element pooly ⟨ucw/eltpool.h⟩
  - Rychlý alokátor pro objekty fixní velikosti

Bufferované I/O – back-endy

⟨ucw/fastbuf.h⟩ – modulární systém pro bufferované I/O

Výrazně rychlejší než stdio, a přitom ohebnější.

Back-endy:

- Soubory
- Direct I/O
- Paměťově mapované soubory
- Data v paměti
- Souvislé bloky paměti (i mempooly)
- Atomické soubory

Bufferované I/O – front-endy

Front-endy:

- Přístup po bajtech či blocích – `bgetc()`, `bputc()`, `bread()`, `bwrite()`, `bbcopy()`
- Řetězcové – `bgets()`, `bgets_mp()`, `bprintf()`
- Unicode – `bget_utf8()`
- Binární – `bgetw()`, `bgetl_be()`
- Přímý přístup ke kusům bufferu

Filtry:

- Konverze znakových sad
- Komprese

Parser konfiguračních souborů – definice

```
int nr1, t1;
char *str1;
clist secs;

static struct cf_section cf_top = {
  CF_INIT(init_top),
  CF_ITEMS {
    CF_INT("NumGnomes", &nr1),
    CF_STRING("YourName", &str1),
    CF_PARSER("WakeUp", &t1, time_parser, -1),
    CF_LIST("Gnome", &secs, &cf_sec_1),
    CF_END
  }
};
```

Parser konfiguračních souborů – soubor

```
# Your name
YourName                A. U. Thor

# Expected number of gnomes in your garden
NumGnomes               4k

# Time when you usually wake up
# (no de-gnoming before that moment)
WakeUp                  8:00

# List of known gnomes
Gnome { Name Grumpy ; Color green }
Gnome { Name Smurfy ; Color blue }
Include cf/common-gnomes
Gnome:edit { Name Wheezy } { Color Gray }
```

Stringové operace

Céčko je proslulé nepohodlnými řetězci. Snadná pomoc:

- Různé šikovné funkce – např. `sepsplit()`
- Funkce alokující výsledek v mempoolu ⟨ucw/mempool.h⟩
  `mp_printf()`, `mp_strcat()`
- Funkce alokující pomocí `alloca()` ⟨ucw/stkstring.h⟩
  `stk_printf()`, `stk_strcat()`
- Spolehlivý parser čísel ⟨ucw/strtonum.h⟩
- Vyhledávací automaty ⟨ucw/kmp.h⟩

Hlavní smyčka

⟨ucw/mainloop.h⟩

- Sledování file-deskriptorů – `file_add()`
- Časovače – `timer_add()`
- Sledování procesů – `process_add()`
- Synchronní doručování signálů – `signal_add()`
- Blokové I/O – `block_io_add()`

Různé

- Logger
- Rychlá komprese
- Token Bucket Filter
- Manipulace s URL
- Rozdělování zátěže mezi vlákna
- Ve vývoji: Transakce a výjimky
- … a mnoho dalšího

Závěrem

Výhody:

- Léty ověřeno, že se používá pohodlně.
- Mnohem rychlejší než srovnatelné knihovny v C i C++.

Nevýhody:

- Nekompletní dokumentace
- Ne úplně stabilní API (občas nekompatibilní změny)
- Nemá čistý namespace (to je i trochu výhoda)

Ke stažení na `http://www.ucw.cz/libucw/` (Git)

Patche (a do dokumentace obzvlášť) vítány :)