Medvědův průvodce po LibUCW

Martin Mareš
mj@ucw.cz

Katedra Aplikované Matematiky
MFF UK Praha

2015

Co je LibUCW

Knihovna pro zpříjemnění života Céčkových programátorů :)

Součásti:

- Generické datové struktury
- Alokátory paměti
- Modulární bufferované I/O
- Parser konfiguračních souborů
- Parser command-line argumentů
- Stringové operace
- Třídění interní i externí
- Hlavní smyčka (abstrakce nad `select()` a spol.)
- Logování
- Transakce a výjimky

Generické datové struktury

- ⟨ucw/slists.h⟩, ⟨ucw/clists.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 (Bloomův filtr)

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 či iostreams, a přitom ohebnější.

Back-endy:

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

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 (zero-copy)

Filtry:

- Konverze znakových sad
- Komprese
- Zřetězení

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 }
```

Parser command-line parametrů

```
#include <ucw/opt.h>

int english, sugar, verbose;
char *tea_name;

struct opt_section options = { OPT_ITEMS {
  OPT_HELP("Usage: teapot [options] tea-name"),
  OPT_HELP("Options:"), OPT_HELP_OPTION,
  OPT_BOOL('e', "english-style", english, 0,
    "\tEnglish style (with milk)"),
  OPT_INT('s', "sugar", sugar, OPT_REQUIRED_VALUE,
    "<spoons>\tAmount of sugar (in teaspoons)"),
  OPT_INC('v', "verbose", verbose, 0,
    "\tVerbose (the more -v, the more verbose)"),
  OPT_STRING(OPT_POSITIONAL(1), NULL, tea_name,
    OPT_REQUIRED, ""),
  OPT_END
} };
```

Stringové operace

Céčko je proslulé nepohodlnými řetězci. Snadná pomoc:

- 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⟩
- Různé drobnosti – `sepsplit()`, `str_has_suffix()`

Sorter

⟨ucw/sorter/array.h⟩ – pole, řádově GB
⟨ucw/sorter/sorter.h⟩ – soubory, řádově TB

- Opět generování kódu pro konkrétní druh dat
- Algoritmy volí podle dat a konfigurace
- V paměti:
  - QuickSort
  - RadixSort (máme-li monotónní hash)
  - Paralelní verze obojího
  - Optimalizace využití cache
- Na disku:
  - Disk není páska
  - MergeSort (typicky 256-cestný)
  - RadixSort
- Unifikace položek se stejným klíčem

Hlavní smyčka

⟨ucw/mainloop.h⟩

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

Logování

⟨ucw/log.h⟩

- Základní použití:

```
msg(L_ERROR, "Cannot drink %s: %m", name)
```

- První parametr kombinuje:
  - Závažnost zprávy: `L_INFO`, `L_WARN`, `L_ERROR` ...
  - Typ zprávy (definován aplikací)
  - Cílový logstream (definován aplikací)
  - Příznaky: `L_SIGHANDLER`
- Logstreamy:
  - Určují cíl (soubor, syslog, ...)
  - Určují formát
  - Mohou filtrovat
  - Mohou přeposílat
  - Mohou omezovat průtok
- Implicitní logstream: stderr bez filtrů

Logování – konfigurace

```
Logging {
    Stream {
        Name            http
        Substream       http-file
        Substream       http-syslog
        Limit { Types remote; Rate 10; Burst 30 }
    }
    Stream {
        Name            http-file
        FileName        /var/log/http-%Y%m%d
        Microseconds    1
    }
    Stream {
        Name            http-syslog
        SyslogFacility daemon
        Levels:remove   debug
    }
}
```

Transakce

⟨ucw/trans.h⟩

- Zobecnění mechanismu výjimek
- Použití:
  - Otevřeme transakci
  - Transakce si pamatuje vytvořené objekty (paměť, deskriptory, pracovní soubory, ...)
  - Neúspěšné ukončení (rollback): uvolnění objektů, výskok z kódu
  - Úspěšné ukončení (commit): označené objekty zůstanou
- Výjimky obsahují:
  - Hierarchický identifikátor (`ucw.fb.read`)
  - Chybovou hlášku
  - Objekt, na kterém chyba nastala
  - Další data závislá na typu
- Podtransakce může výjimku pohltit / předat dál / upravit

Různé

Některé další moduly:

- Manipulace s URL
- Rozdělování zátěže mezi vlákna
- Rychlá komprese (inspirována LZO)
- Kódování: base64, base224
- Kryptografie: MD5, SHA1, HMAC
- Zabezpečení: CRC

Plány do budoucna:

- Další vývoj transakcí
- Datové struktury optimalizované na cache

Závěrem

Výhody:

- Léty ověřeno, že se používá pohodlně
- Mnohem rychlejší než srovnatelné knihovny v C i C++
- Využívá nové syscally (`epoll`, `CLOCK_MONOTONIC` atd.)

Nevýhody:

- Nekompletní dokumentace
- Ne úplně stabilní API (občas nekompatibilní změny)
- Vyžaduje rozšíření GCC (asi by šlo i ICC/clang)

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

Patche (a do dokumentace obzvlášť) vítány :)