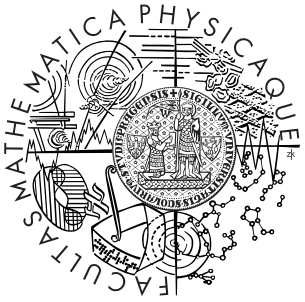
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Mário Mikula

Nástroje pro vývoj XBW

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Jan Lánský

Studijní program: Informatika, Programování

2009

Ďakujem pánovi Mgr. Janovi Lánskemu za odborné vedenie mojej práce a konzultantovi Petrovi Uzlovi za rady a za čas, ktorý mi venovali. Taktiež ďakujem ľudom, ktorí mi pomáhali s úpravami.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 18.5.2009 Mário Mikula

Obsah

Název práce: Nástroje pro vývoj XBW
Autor: Mário Mikula
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Jan Lánský
e-mail vedoucího: Jan.Lansky@mff.cuni.cz

Abstrakt: Predmetom tejto práce sú dva nástroje pre zjednodušenie vývoja kompresného systému XBW. Prvým je knižnica umožnujúca výpis ladiacich informácií, vhodná pre hierarchickú štruktúru modulov XBW. Táto štruktúra je dynamická a môže sa meniť s časom. Druhým je program, kontrolujúci korektnosť fungovania XBW pre jeho rôzne kombinácie parametrov. Parametre pre testovanie sa budú dať zadávať vhodným spôsobom - regulárnymi výrazmi.

Klíčová slova: XBW, ladíci výpisy, testování, komprese

Title: Tools for Development of XBW
Author: Mário Mikula
Department: Department of Software Engineering
Supervisor: Mgr. Jan Lánský
Supervisor's e-mail address: Jan.Lansky@mff.cuni.cz

Abstract: This thesis discusses two tools for simplifying the development process of XBW compression system. The first is a library for writing up debug information suitable for the hierarchical structure of XBW modules. Which is a dynamic structure and can change over time. The second is a program which verifies correctness of XBW functionality based on selected parameter combinations. Parameter combinations for testing are defined in a suitable manner - with regular expressions.

Keywords: XBW, debug printings, testing, compression

Kapitola 1

Úvod

XBW je open-source projekt pre testovanie rôznych bezstratových kompresií a ich aplikácií, vyvíjaný na MFF UK. Vývoj sa zameral na veľké XML súbory, pre ktoré kombinuje metódy kompresie XML a kompresie textu. Na týchto súboroch dosahuje veľmi dobré výsledky. Viac informácií je možné získať na stránkach projektu (http://xbw.sourceforge.net/).

Na vývoji sa stále pracuje a programátorom by v ňom pomohlo niekoľko jednoduchých nástrojov. Dva z nich sú predmetom tejto práce.

Jedným z nich je knižnica pre výpis ladiacich informácií. Táto knižnica by mala poskytovať niekoľko úrovní závažnosti výpisu, výstup na štandardný a chybový výstup. Ďalej výstup do súborov a ich prípadné verzovanie.

XBW sa skladá z hierarchicky usporiadaných programových modulov. Toto usporiadanie by malo byť možné v knižnici definovať a využívať ho pre nastavenia modulov a skupín modulov vo vzájomnom vzťahu. Každý modul by ale mal byť, v prípade potreby, samostatný. Táto štruktúra je navyše dynamická a môže sa za behu meniť. To by malo byť taktiež v knižnici zohľadnené.

Knižnica je určená pre potreby vývoja, takže dôraz nie je kladený na výkon. Malo by ale byť možné jednoducho oddeliť kód knižnice od kódu XBW

a tým vypnúť ladiace výpisy.

Druhým z nástrojov je program pre testovanie správnosti behu XBW. XBW poskytuje niekoľko kompresných metód a ich nastavení. Tie sú zadávané ako parametre. Tester by mal byť schopný nejakým rozumným spôsobom testovať XBW s rôznymi kombináciami týchto parametrov.

Kontrola správnosti spočíva v kompresii testovacieho vstupného súboru pomocou zvolených metód, jeho následná dekompresia a porovnanie s pôvodným nekompresovaným súborom.

Kapitola 2

Analýza

2.1 Prostredie

Výber programovacieho jazyka bol jednoznačný. Je žiadúce, aby XBW, jeho knižnice a pomocné programy boli napísané v jednom jazyku a aby sa dali prekladať súčasne. Vzhľadom na to, že XBW je naprogramované v jazyku C, nebolo nad čím uvažovať.

Dobré by bolo, aby boli použité len štandardné prostriedky jazyka. Od začiatku ale bolo jasné, že sa to nepodarí, kedže sa bude pracovať so súborovým systémom a ten je závislý na operačnom systéme. Vývoj a testovanie XBW prebieha na GNU/Linuxe, takže bol zvolený tento systém.

2.2 Knižnica pre ladiace výpisy

S programovacím jazykom je spojená aj otázka knižnice pre ladiace výpisy. Prečo programovať dalšiu logovaciu knižnicu, keď sú podobné nástroje už naprogramované? Ani z teoretického hľadiska nie je takáto knižnica veľmi zaujímavá, tak prečo nepoužiť nejakú existujúcu?

Väčšina logovacích nástrojov je robená pre vyššie programovacie jazyky

typu C++ a Java. To pre účely XBW nevyhovuje. Ostáva menšia časť a z nej vyzerá nádejne napríklad projekt *Log4c* (http://log4c.sourceforge.net/). To je knižnica pre jazyk C po vzore *Log4j* pre Javu. Je ale príliš univerzálna, určená pre široké použitie a pre naše účely zbytočne komplikovaná. Pre XBW by vyhovovalo niečo menšie a jednoduchšie. Robené špeciálne pre jeho potreby.

XBW potrebuje niekoľko pevne daných úrovní závažnosti logovania. Výstup na konzolu a do súborov. Nezávislý výstup z každého modulu a pridávanie a rušenie modulov za behu programu. Moduly vo vzťahoch rodič - potomok a s tým súvisiace dedenie nastavení. Vypnutie výpisov pomocou preprocesoru jakyza C.

Voľba padla na napísanie takejto jednoduchej logovacej knižnice špeciálne pre XBW a bola pomenovaná *libdbg*.

2.3 Tester

XBW implementuje rôzne metódy kompresie a ich nastavenia, rôzne spôsoby parsovania vstupného súboru a jeho rôzne kódovania. Počet kombinácií týchto parametrov je príliš veľký na to, aby bolo únosné manuálne testovanie a do budúcna sa tento počet bude len zvyšovať. Testovanie by uľahčil nástroj, ktorému by sa zadala množina kombinácií parametrov, automaticky by všetky otestoval a informoval by o výsledku.

2.3.1 Možnosti zadávania kombinácií parametrov

Ako zadávať množinu kombinácií parametrov na testovanie? Počet parametrov pre XBW, určujúcich metódu spracovania a kompresie vstupného súboru, je vo verzii 0.1alpha 10. Každý z nich má od 2 do 10 parametrov. Zďaleka nie všetky tieto parametre môžu byť špecifikované súčasne, pretože sa vzájomne vylučujú, takže reálne sa ich bude používať menej ako 10. Tieto

čísla nie sú veľké, ale počet rôznych možných kombinácií je aj tak príliš veľký na to, aby sa zadávali po jednej.

Ďalšou možnosťou je ku každému parametru určiť argumenty, ktoré sa budú testovať. V každom testovacom cykle sa vyberie jeden argument k príslušnému parametru a tak vznikne kombinácia. Takto sa otestujú všetky kombinácie. Pre súčasnú verziu by bol tento spôsob asi postačujúci, ale keď sa rozšíri množstvo možných parametrov a ich argumentov, tak by mohol byť tento zápis problematický, kvôli dĺžke a neprehľadnosti.

Ponúka sa vylepšenie pomocou regulárnych výrazov. Argumenty pre daný parameter sa budú určovať regulárným výrazom, čo je dostatočne silný mechanizmus pre tieto účely a zároveň je jednoduchý. Nie je problém vybrať len niektoré argumenty (pokrýva predchádzajúcu možnosť), všetky argumenty alebo množinu argumentov s nejakým spoločným prvkom v názve.

Tu sa ale objavuje jeden problém. Voči čomu porovnať regulárny výraz.

Príklad. Chceme otestovať všetky možnosti parseru XBW, tak použijeme "*Parser=.**".

Regulárny výraz *.** zodpovedá všetkým možným reťazcom znakov. Ako tester zistí, ktoré z nich sú platné argumenty pre parameter *Parser*? Zvažované boli dve možnosti. Obe majú svoje výhody a nevýhody.

Prvou z nich je konfiguračný súbor, kde bude určené, čo sú platné parametre XBW a čo sú ich platné argumenty. Regulárne výrazy budú porovnávané voči tomuto súboru. Výhodou je, že je to jednoduché a prehľadné. Keď sa v nejakej budúcej verzii XBW pridajú ďalšie parametre a argumenty, tak bude treba tento súbor upraviť a to je nevýhoda.

Druhou možnosťou je nejakým spôsobom konfiguračný súbor generovať. Napríklad z dokumentácie XBW alebo z jeho pomocného výpisu. Táto možnosť je elegantnejšia, pretože nepotrebuje manuálny zásah do konfiguračného súboru ani do kódu, pri zmene možných parametrov. Silná nevýhoda je ale to, že formát zdroja dát by musel ostať nemenný, aby generovanie korektne

fungovalo.

Argument proti druhej verzii je príliš závažný, takže bola zvolená prvá varianta s manuálnym konfiguračným súborom.

2.3.2 Testovanie a meranie času

Ako realizovať samotné testovanie správnosti behu XBW? Ak všetko korektne funguje, tak sa kompresovaný a následne dekompresovaný súbor zhoduje s pôvodným súborom. A toto sa bude overovať.

Zároveň sa bude merať čas kompresie a dekompresie pomocou unixového nástroja *time*.

Kapitola 3

Užívateľská dokumentácia libdbg

3.1 Kompilácia a linkovanie

Kompilácia a linkovanie sú riadené *Makefile* skriptom pre program *make* a používa sa prekladač *GCC*. Je vyrobená statická knižnica *libdbg.a*. Pre používanie funkcií z knižnice je potrebné do programu includovať hlavičkový súbor *libdbg.h*. Tento súbor vyžaduje ešte súbory *dbg_errors.h* a *dbg_config.h*. Obsah týchto súborov nie je veľký a je možné ho pre zjednodušenie presunúť do *libdbg.h*. Oddelený ostal z dôvodu prehľadnosti.

Pre zapnutie logovania je potrebné mať pri kompilácii definované makro *DEBUG_PRINTS*. Viz Výpis 3.5.6.

Synchronizácia pre vlákna sa zapína definovaním makra *PTHREAD*.

Doxy dokumentáciu je možné vygenerovať pomocou nástroja *doxygen* (http://www.stack.nl/~dimitri/doxygen/) a priložného súboru *Doxyfile*.

3.2 Datové typy

3.2.1 Úrovne závažnosti

Pre úroveň závažnosti výpisu je použitý nasledujúci typ:

```
typedef enum
{
        DBG_NONE,
        DBG_ERROR,
        DBG_WARNING,
        DBG_INFO,
        DBG_DEBUG,
        DBG_TRACE
}
dbg_loglevel_t;
```

Jeho hodnoty sú lineárne usporiadané *DBG_ERROR* < *DBG_WARNING* < ... < *DBG_TRACE*.

Príklad: Modul s nastavenou úrovňou logovania *DBG_INFO* bude príjmať výpisy úrovne *DBG_INFO* a menšie, čiže *DBG_WARNING* a *DBG_ERROR*.

DBG_NONE je špeciálna úroveň, ktorá sa nepoužíva v logovacej funkcii a nastavuje sa modulu, ak z neho nechceme žiadne výpisy.

3.2.2 Flagy

Typy pre flagy. Ich význam je vysvetlený vo funkciách, ktoré s nimi pracujú. Viz Funkcie rozhrania 3.5.

```
dbg_command_flags_t
dbg_module_flags_t
```

3.2.3 Modul

Typ pre modul (modul id):

```
dbg_module_t
```

3.3 Koreňový modul

Koreňový modul je globálna premenná knižnice a slúži ako predok všetkých ostatných modulov. Čiže sa pomocou neho dajú nastavovat ich vlastnosti. Koreňový modul je inicializovaný funkciou *dbg_init()* (viz Inicializácia 3.5.1) a je to jediný modul, ktorý nemá rodiča. Je potrebné ho nastaviť ako rodiča modulu, ktorý v reálnej hierarchii žiadneho rodiča nemá.

```
extern dbg_module_t dbg_root_module;
```

3.4 Návratové kódy

Pre návratové kódy funkcií rozhrania sa používa typ:

```
typedef enum
{
        DBG_SUCCESS,
        DBG_FAILURE,
        DBG_LOW_MEMORY,
        DBG_BAD_FLAGS,
        DBG_INVALID_NULL_PARAMETER,
        DBG_NOT_ALLOWED,
        DBG_LOGFILE_VERSION_TOO_HIGH,
        DBG_UNKNOWN_ERROR,
        DBG_MUTEX_ERROR
}
```

```
dbg_retcode_t;
```

Významy jednotlivých hodnôt sú vysvetlené jednotlivo pre každú funkciu. Viz Funkcie rozhrania 3.5.

3.5 Funkcie rozhrania

3.5.1 Inicializácia

```
dbg_retcode_t dbg_init(void);
```

Inicializácia logu. Touto funkciou je inicializovaný koreňový modul (viz Koreňový modul 3.3) na logovanie do file descriptoru 1 (štandardný výstup) s logovaciou úrovňou *DBG_WARNING*. Je potrebné ju zavolať pred prvým volaním samotnej výpisovej funkcie *dbg_print()* (viz Výpis 3.5.6).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_LOW_MEMORY - Nebolo dostatok volnej operačnej pamäte pre inicializáciu. Log nie je inicializovaný.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.2 Vytvorenie modulu

```
dbg_retcode_t dbg_add_module(const char * name,
        dbg_module_t parent, dbg_module_t * new_module_out);
```

Vytvorenie nového modulu. Parametre sú názov nového modulu, jeho rodič a pointer na miesto v pamäti, kam sa uloží jeho id. Nový modul dedí všetky nastavenia po svojom rodičovi a po úspešnom návrate z funkcie je pripravený na logovanie. Ak modul nemá žiadneho logického predka, tak musí byť ako predok použitý koreňový modul (viz Koreňový modul 3.3).

Nový modul má prednastavenú logovaciu úroveň *DBG_WARNING* a zapnuté logovanie do file descriptoru číslo 1
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Jeden z parametrov *name* alebo *parent* bol *NULL*.

DBG_LOW_MEMORY - Nedostatok operačnej pamäte.

DBG_LOGFILE_VERSION_TOO_HIGH - Verzia výstupného súboru je príliš vysoká. Viz Konfiguračný súbor 3.6.

DBG_UNKNOWN_ERROR - Chyba štandardnej funkcie alebo systemového volania. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.3 Odstránenie modulu

```
dbg_retcode_t dbg_remove_module(dbg_module_t module,
        dbg_command_flags_t flags);
```

Funkcia Odstráni modul a rekurzívne jeho potomkov. Toto správanie môže byť zmenené flagom *DBG_CFLAGS_NOT_RECURSIVE*. Takto je ale možné odstrániť len modul, ktorý nemá potomkov. Odstránený modul už nesmie byť ďalej používaný bez opätovnej reinicializácie funkciou *dbg_add_module()* (viz Vytvorenie modulu 3.5.2).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_FAILURE - Parameter *module* bol koreňový modul. Ten nie je možné odstrániť.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_NOT_ALLOWED - Nerekurzívna varianta dostala na odstránenie modul, ktorý ma nejakých potomkov a to nie je dovolené.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.4 Nastavenie logovacej úrovne

```
dbg_retcode_t dbg_set_loglevel(dbg_module_t module,
        dbg_loglevel_t level, dbg_module_flags_t module_flags,
        dbg_command_flags_t command_flags);
```

Funkcia nastaví logovaciu úroveň (parameter *level*) pre daný modul (parameter module) a rekurzívne pre jeho potomkov. Ak je použitý flag *DBG_CFLAGS_NOT_RECURSIVE* v parametri *command_flags*, loglevel potomkov sa nenastavuje. Parameter *module_flags* je kombinácia flagov *DBG_MFLAGS_FILE* a *DBG_MFLAGS_FD*, ktoré určujú, pre ktoré vý-

stupy bude úroveň nastavená. *DBG_MFLAGS_FILE* znamená súbor, *DBG_MFLAGS_FD* file descriptor. Pre viac informácií o logovacej úrovni viz Úrovne závažnosti 3.2.1.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter *module* bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_get_loglevel(dbg_module_t module,
        dbg_module_flags_t module_flags,
        dbg_loglevel_t * loglevel_out);
```

Funkcia zistí logovaciu úroveň pre daný modul (parameter *module*), daný výstup (parameter *module_flags*) a uloží ho na dané miesto v pamäti (parameter *loglevel_out*). Parameter *module_flags* musí byť buď *DBG_MFLAGS_FILE* (pre súbor) alebo *DBG_MFLAGS_FD* (pre file descriptor).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter *module* bol *NULL*.

DBG_BAD_FLAGS - Zlý flag v parametre *module_flags*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.5 Nastavenia výstupu

```
dbg_retcode_t dbg_set_logfile(dbg_module_t module,
        const char * filename,
        dbg_command_flags_t command_flags);
```

Funkcia nastaví pre daný modul (parameter *module*) výstupný súbor (parameter *filename*) a otvorí ho pre zápis. Výstupný súbor je rekurzívne nastavený aj všetkým potomkom modulu, ak toto chovanie nie je vypnuté pomocou flagu *DBG_CFLAGS_NOT_RECURSIVE* v parametri *command_flags*. Tento parameter akceptuje aj flag *DBG_CFLAGS_INCREMENTAL*, ktorý spôsobí, že ak výstupný súbor existuje, tak bude vytvorený nový s poradovým číslom verzie za názvom a použitý bude tento nový súbor. Ak bol pred úspešným volaním tejto funkcie otvorený nejaký iný výstupný súbor, tento bude zatvorený. *NULL* v parametri *filename* znamená, že modul nebude mať určený výstupný súbor a logovanie do súboru nebude mať žiaden efekt.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter *name* alebo *parent* bol *NULL*.

DBG_LOW_MEMORY - Nedostatok operačnej pamäte.

DBG_LOGFILE_VERSION_TOO_HIGH - Verzia výstupného súboru je príliš vysoká. Viz Konfiguračný súbor 3.6.

DBG_UNKNOWN_ERROR - Chyba štandardnej funkcie alebo systemového volania. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu.

DBG_FAILURE - Funkcia sa pokúsila zatvoriť súbor, ktorý nemal byť otvorený. Datové štruktúry sa dostali do nekonzistentného stavu (pravdepodobne vonkajším zavinením).

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_get_logfile(dbg_module_t module,
        const char ** logfile_out);
```

Funkcia zistí názov logovacieho súboru pre daný modul (parameter *module*) a uloží do do daného miesta v pamäti (parameter *logfile_out*). Ukladá sa len pointer na pole znakov a toto pole sa nesmie meniť. V prípade potreby zmeny tohoto reťazca, tento musí byť prekopírovaný.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_set_logfd(dbg_module_t module,
        int fd, dbg_command_flags_t command_flags);
```

Funkcia nastaví file descriptor pre logovanie (parameter *fd*) pre daný modul (parameter *module*) a rekurzívne pre jeho potomkov. Flag *DBG_CFLAGS_NOT_RECURSIVE* v parametri *command_flags* vypína re-

kurzívne správenie funkcie. File descriptor musí byť otvorený pre zápis. Používa sa na nastavenie výstupu na štandardný (*stdout*) alebo chybový (*stderr*) výstup.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_get_logfd(dbg_module_t module,
        int * logfd_out);
```

Funkcia zistí logovací file descriptor pre daný modul (parameter *module*) a uloží do do daného miesta v pamäti (parameter *logfd_out*).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.6 Výpis

```
dbg_retcode_t dbg_print_function(dbg_module_t module,
        dbg_loglevel_t level, const char * fmt, ...);
```

Funkcia zabezpečujúca výpis na výstup. Parameter *module* je modul, ktorý výpis spôsobil, parameter *level* je úroveň závažnosti výpisu, parameter *fmt* je formátovací reťazec (formát *printf*), za ktorým nasledujú premenné jemu zodpovedajúce.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_UNKNOWN_ERROR - Chyba funkcie *frpintf()* alebo *write()*. Globálna chybová premenná *errno* je nastavená na príšlušnú hodnotu.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
#define dbg_print(module, loglevel, ...) \
        dbg_print_function(module,loglevel, __VA_ARGS__)
```

Toto makro volá funkciu *dbg_print_function*, a to v prípade, že je v C preprocesore definované makro *DEBUG_PRINTS*. Inak vracia *DBG_SUCCESS*. Jednoduchá cesta, ako vypnúť všetky výpisy je pri kompilácii nedefinovať makro *DEBUG_PRINTS*.

3.5.7 Ďalšie funkcie

```
dbg_retcode_t dbg_get_parent(dbg_module_t module,
        dbg_module_t * parent_out);
```

Funkcia zistí rodiča daného modulu (parameter *module*) a uloží ho na danú adresu (parameter *parent_out*).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_get_children(dbg_module_t module,
        dbg_module_t ** children_out,
        int * children_count_out);
```

Funkcia zistí deti daného modulu (parameter *module*) a uloží ich na danú adresu (parameter *children_out*). Na adresu, na ktorú ukazuje parameter *children_count_out*, bude uložený ich počet. Deti sú uložené do pola, ktoré je používané otcovským modulom, preto nie je doporučené toto pole meniť inak ako cez funkcie rozhrania. V opačnom prípade hrozí, že sa datové štruktúry dostanú do nekonzistentného stavu.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_set_module_flags(dbg_module_t module,
        dbg_module_flags_t module_flags,
        dbg_command_flags_t command_flags);
```

Funkcia nastaví flagy pre daný modul (parameter *module*) a rekurzívne pre jeho potomkov na príslušnú hodnotu (parameter *module_flags*). Flag *DBG_CFLAGS_NOT_RECURSIVE* v parametri *command_flags* vypína rekurzívne správenie funkcie.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

```
dbg_retcode_t dbg_get_module_flags(dbg_module_t module,
        dbg_module_flags_t * flags_out);
```

Funkcia zistí flagy daného modulu (parameter *module*) a uloží ich na danú adresu (parameter *flags_out*).
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_INVALID_NULL_PARAMETER - Parameter module bol *NULL*.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.5.8 Ukončenie logovania

```
dbg_retcode_t dbg_terminate(void);
```

Táto funkcia by mala byť zavolaná niekedy po poslednom logovaní. Ukončuje logovanie, zatvára súbory a uvoľnuje pamäť.
Návratové hodnoty:

DBG_SUCCESS - Úspešný návrat.

DBG_FAILURE - Koreňový modul je *NULL*, nie je čo ukončovať.

DBG_UNKNOWN_ERROR - Chyba štandardnej funkcie alebo systemového volania. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu.

DBG_MUTEX_ERROR - Chyba pri používaní mutexu pre synchronizáciu. Globálna chybová premenná *errno* je nastavená na príslušnú hodnotu. Táto chyba sa môže vyskytnúť len pri zapnutej synchronizácii (viz kapilota Kompilácia a linkovanie 3.1).

3.6 Konfiguračný hlavičkový súbor

Súbor *dbg_config.h* obsahuje niekoľko makier, ktoré ovplyvňujú preklad knižnice.

MAX_LOGFILE_VERSION Určuje maximálne číslo verzie, pre verzovaný výstupný súbor.

LOGFILE_VERSION_LENGTH Je počet cifier v tomto čísle, čiže jeho dĺžka.

CHILDREN_ARRAY_SIZE_DEFAULT Udáva počiatočnú veľkosť pola pre deti jedného modulu.

CHILDREN_ARRAY_SIZE_MULTIPLIER Udáva, koľko krát sa zväčší pole detí pri realokácii.

STRING_BUFFER_LENGTH Určuje maximálnu dĺžku logovacieho reťazca.

3.7 Príklad použitia

Súbor *test.c* je malá ukážka použitia *libdbg*. Preloží sa príkazom *make test*.

Kapitola 4

Programátorská dokumentácia *libdbg*

4.1 Prehľad zdrojových súborov

dbg_config.h Konfiguračný hlavičkový súbor. Viz kapitola 3.6.

dbg_errors.h Hlavičkový súbor s návratovými kódmi. Viac o návratových kodóch v kapitole 3.4.

files.c a files.h Programový modul, ktorý sa stará o prácu so súbormi. Podrobnejší popis v kapitole 4.4.

dbglib.h a dbglib.c Hlavné zdrojové kódy knižnice. Rozobraté v kapitolách Makrá a konštanty v *libdbg.c* 4.2 a Datové štruktúry v *libdbg.c* 4.3.

test.c Krátka ukážka použitia *dbglib*.

4.2 Makrá a konštanty v *libdbg.c*

Okrem makier v súbore *dbg_config.h* sú makrá a konštanty ešte v súbore *dbglib.c*. Nasleduje ich popis.

MODULE_NAME_DEFAULT Zmena nie je bezpečná. Je to reťazec, ktorý sa použije pri alokácii nového modulu, v uživateľovi nepristúpnej funkcii. Logicky by mal byť *NULL*, protože modul, novo vytvorený uživateľom, okamžite dostane meno, ktoré mu uživateľ určí (viz Vytvorenie modulu 3.5.2). Toto makro slúži len pre účel pomenovania *NULL* konštanty.

LOGFILE_NAME_DEFAULT Taktiež by sa nemalo meniť. Je to názov súboru, ktorý bude priradený novému modulu. Tento súbor sa ale pri alokácii modulu neotvára, takže by sa datové štruktúry mohli dostať do nekonzistentného stavu.

LOGLEVEL_DEFAULT Predvolený logovací level pre nové moduly.

PARENT_DEFAULT Predvolený rodič nového modulu. Nemá zmysel meniť, pretože uživateľské funkcie nedovolia vytvoriť nový modul bez udania rodiča.

LOGFILE_DEFAULT Predvolený výstupný súbor modulu. Predpokladá sa, že tento súbor je otvorený pre zápis, ak nie je NULL. Nie je doporučené menenie.

LOGFD_DEFAULT Predvolený výstupný file descriptor. Taktiež sa predpokladá, že daný file descriptor je otvorený pre zápis.

MODULE_FLAGS_DEFAULT Predvolené flagy modulu.

4.3 Datové štruktúry v *libdbg.c*

V súbore *libdbg.c* je definovaná štruktúra pre modul:

```
struct dbg_module_struct
{
        char * name;
        dbg_loglevel_t loglevel_file;
        dbg_loglevel_t loglevel_fd;
        dbg_module_t parent;
        dbg_module_t * children;
        int children_count;
        int children_array_size;
        char * logfile_name;
        FILE * logfile;
        int logfd;
        dbg_module_flags_t flags;
};
```

Význam jednotlivých položiek je zrejmý z ich názvov. Taktiež sa tu nachádza deklarácia premennej pre koreňový modul (viz Koreňový modul 3.3), čo je pointer na zmienenú štruktúru.

```
dbg_module_t dbg_root_module;
```

Nové moduly sa alokujú dynamicky a vytvára sa z nich n-árny obojsmerný strom. Koreňom tohoto stromu je koreňový modul (viz Koreňový modul 3.3). Modul si udržuje pointer na svojho rodiča a pole pointerov na svoje deti. Ak je toto pole naplnené, tak sa realokuje na novú veľkosť podľa konštanty *CHILDREN_ARRAY_SIZE_MULTIPLIER* (viz Konfiguračný hlavičkový súbor 3.6). Tento strom sa prechádza sekvenčne a nevyhladáva sa v ňom, takže prvky nie sú nijak usporiadané.

4.4 Práca so súbormi

Viacero modulov môže zapisovať do jedného súboru, používať rovnaký ukazateľ na otvorený súbor. Problém nastane, keď uživateľ jeden z týchto modulov odstráni. Pri odstanovaní modulu sa zatvára súbor, do ktorého modul zapisoval. Ak by sa zatvoril tento zdielaný súbor, tak sa to ostatné moduly nedozvedia a budú sa snažiť zapisovať do zatvoreného súboru, čo skončí chybou.

Problém sa rieši tým, že sa pre každý súbor udržuje počet modulov, ktoré ho majú otvorený. Až keď tento počet klesne na nulu, súbor je zatvorený. Kód k tomuto mechanizmu sa nachádza v súbore *files.c* a verejné funkcie pre prácu s ním v súbore *files.h*.

Počet otvorených súborov bude najviac toľko, koľko je modulov a tých budú rádovo jednotky až desiatky. To pre výkonnosť dnešných počítačov nepredstavuje žiadny problém. Pre jednoduchosť bol zvolený spájaný zoznam indexovaný názvom súboru. Keď príde žiadosť na otvorenie súboru, tak sa tento súbor vyhľadá v zozname. Ak už ho má nejakú modul otvorený, vráti sa ukazateľ na súbor a zvýši sa počet jeho otvorení. Ak súbor ešte otvorený nie je, tak sa otvorí a počet otvorení sa nastaví na jeden.

Obdobne pri zatváraní súboru. Ak je počet jeho otvorení rovný jednej, tento počet sa zníži na nulu a súbor sa zatvorí. Ak je počet otvorení väčší ako jeden, zníži sa o jednotku.

Tento mechanizmus je závislý na ostatnom kóde len v tom, že používa návratové kódy z *dbg_errors.h* (viz Návratové kódy 3.4), takže je možné ho jednoducho použiť aj v iných projektoch.

Kapitola 5

Užívateľská dokumentácia testeru

Tester je konzolová aplikácia, ktorá testuje správnosť fungovania XBW na vstupných testovacích súboroch.

5.1 Kompilácia a linkovanie

Kompilácia a linkovanie sú riadené *Makefile* skriptom pre program *make* a používa sa prekladač *GCC*. Spustiteľný súbor vytvorený programom *make* je nazvaný *tester*.

Doxy dokumentáciu je možné vygenerovať pomocou nástroja *doxygen* a priložného súboru *Doxyfile*.

5.2 Použitie

tester [-i input_file |-p parameters |-A] [-s] [-v] [-b] [-x xbw_path]
[-C xbw_compress_params] [-X xbw_decompress_params] [-c config_file]
[-d diff_command] [-t [-T time_command]] [-F xbw_output_directory]
[[-f directory_with_xbw_input_files] |[xbw_input_file1 ...]]

5.2.1 Predvolené nastavenia

Tester spustený bez parametrov predpokladá:

- Cestu k XBW *xbw/xbw*.
- Vstupné súbory umiestnené v zložke *xbw_input_files*.
- Dočasné súbory ukladá do zložky *xbw_output_temp*.
- Konfiguračný súbor *config.txt*.

Pre viac informácií o predvolených nastaveniach viz Konfiguračný hlavičkový súbor 6.4.

5.2.2 Parametre

-i input_file Vstupný súbor so zápisom parametrov pre testovanie.

-p parameters Zápis parametrov pre testovanie zadaný ako parameter.

-A Otestuje všetky možné kombinácie parametrov. Nedoporúča sa používať, pretože týchto kombinácií je veľa a tester môže bežať dlho.

-s Tichý mód. Vypisujú sa len behové chyby a informácie o prípadnom nesprávnom behu XBW. Nekombinovať s parametrom -v a -b.

-v "Upovídaný" mód. Vypisujú sa niektoré dodatočné informácie. Nekombinovať s parametrom -s.

-b Silnejší "upovídaný" mód. Nekombinovať s parametrom -v.

-r Po kompresii vypíše kompresný pomer výstupného súboru.

-x xbw_path Cesta k spustitelnému súboru XBW.

-C xbw_compress_params Parametre pre XBW, ktoré budú použité pri kompresii.

-X xbw_decompress_params Parametre pre XBW, ktoré budú použité pri dekompresii.

-c config_file Konfiguračný súbor. Viz Konfiguračný súbor 5.3.

-d diff_command Príkaz a jeho parametre, ktorým sa budú porovnávať súbory na zhodu.

-t Zapína meranie času kompresie a dekompresie.

-T time_command Príkaz a jeho parametre, ktorým bude použitý na meranie času.

-F xbw_output_directory Zložka pre umiestnenie výstupých súborov z XBW. Ak neexistuje, bude vytvorená. Vhodné, keď sa púšťa viac inštancií testuru súčasne, aby sa predišlo konfliktom s dočasnými súbormi.

-f directory_with_xbw_input_files Zložka so vstupnými testovacími súbormi pre XBW. Z tejto zložky sa vezmú všetky regulárne súbory.

xbw_input_file1 Argumenty sú vstupné testovacie súbory pre XBW. Testujú sa regulárne súbory.

5.3 Konfiguračný súbor

Konfiguračný súbor obsahuje parametre XBW a ich argumenty. Formát je nasledovný:

```
parameter1:
        argument1.1
        argument1.2
        argument1.3

parameter2:
        argument2.1
        argument2.2
        argument2.3
        argument2.4
```

Parameter musí byť presne ten reťazec, ktorý sa zadavá XBW. Následuje dvojbodka a na novom riadku je tabulátor a argument. Po všetkých argumentoch následuje prázny riadok a ďalší parameter. Príklad:

```
--Parser=:
        off
        xml
        text

--Syl_partition=:
        Middle_Left
        Left
        Right
        Word
        Symbol
```

5.4 Syntax testov

Prvý znak je '#', za ktorým nasleduje testovaný parameter. Ďalej buď znak '#', čo znamená, že parameter nemá žiadne argumenty. Alebo znak '%' a regulárny výraz. Tento regulárny výraz popisuje argumenty daného parametru, ktoré sa budú testovať. Porovnáva sa s argumentami príslušného parametru v konfiguračnom súbore (kapitola 5.3). Príklad:

```
#--Parser=%.*#--Syl_partition=%Left|Right|Word#
```

Takto zadaný test spustí XBW so všetkými kombináciami všetkých argumentov parametru *Parser* a troch daných argumentov parametru *Syl_partition*.

5.5 Výstup

Výstup je nasledujúceho formátu:

```
=============START=============
Processing file input_file1

Compressing:
compression_command1

Decompressing:
decompression_command1

Comparing decompressed file and original:
diff_command1
==============OK!==============
=============START=============
Processing file input_file2

Compressing:
compression_command2

Decompressing:
decompression_command2

Comparing decompressed file and original:
diff_command2
=============FAIL!=============
```

Input_file je vstupný súbor pre XBW. *Compression_command* je kompletný príkaz, pre kompresiu vstupného súboru. *Decompression_command* príkaz

pre dekompresiu. *Diff_command* je príkaz pre porovnanie dekompresovaného súboru s pôvodným. Nasleduje výpis "OK!", ak sa súbory zhodujú a "FAIL!", ak sa nezhodujú, čiže nastala chyba XBW. Príklad:

```
=============START=============
Processing file "xbw_input_files/input1.txt"

Compressing:
xbw/xbw  --BWT=KS --Parser=text --Syl_partition=Word
xbw_input_files/input1.txt xbw_output_temp/input1.txt.xbw

Decompressing:
xbw/xbw -x xbw_output_temp/input1.txt.xbw
xbw_output_temp/input1.txt.xbw.dec

Comparing decompressed file and original:
diff >/dev/null xbw_input_files/input1.txt
xbw_output_temp/input1.txt.xbw.dec
==============OK!==============
```

Do tohoto výpisu sa ešte pridajú výpisy pomocných programov pre meranie času a porovnávanie súborov. Výstup môže byť iný aj v prípade, že je použitý niektorý z parametrov -v, -b alebo -s (viz Parametre 5.2.2). Alebo sa vyskytne nejaká behová chyba (kapitola 5.6).

5.6 Chyby

Chyba môže nastať v samotnom testeri. Vtedy program skončí s upresňujúcou chybovou hláškou. Najbežnejšie chyby sú napríklad špecifikovanie vstupného súboru alebo zložky, ktorá neexistuje alebo sa nedá čítať. Chyba

v syntaxi konfiguračného súboru. Chyba v zadaní niektorého z pomocných programov, takže nie je možné ho spustiť.

Iný druh chýb, sú chyby pomocných programov a programu XBW. Ak nastane takáto chyba, tak sa vypíše výstup programu a ukončí sa testovací proces pre jeden vstupný súbor a jednu kombináciu parametrov, ale tester pokračuje ďalej následujúcou kombináciou parametrov.

Kapitola 6

Programátorská dokumentácia testeru

6.1 Prehľad zdrojových súborov

config.h Konfiguračný hlavičkový súbor. Viz kapitola 6.4.

error_codes.h Návratové kódy funkcií.

global_settings.c a .h Zdrojový súbor s globálnymi premennými a jeho hlavičkový súbor.

par_struct.c a .h Zdrojový súbor s datovými štruktúrami pre ukladanie testovaných parametrov a funkcie pre prácu s nimi.

tester.c Hlavný zdrojový súbor.

verbose.c a .h Funkcie pre výpisy podľa zvolenej úrovne "upovídanosti".

dyn_str/dyn_str.c a .h Datový typ *dynamic_str* a funkcie s ňím pracujúce. Tento typ reprezentuje reťazec znakov, prispôsobujúci sa dĺžke.

6.2 Datové štruktúry a algoritmy

6.2.1 Datové štruktúry

Ako uchovávať testované parametre a ich argumenty v pamäti? Ako sa k nim bude pristupovať a aké operácie sa s nimi budú robiť?

Najprv je potrebné do pamäte načítať konfiguračný súbor. Ten sa číta sekvenčne. Potom sa naparsuje reťazec s parametrami a regulárnymi výrazmi a uloží sa do pamäte. Tento postup je taktiež sekvenčný. Nakoniec sa regulárne výrazy porovnajú s argumentami pre príslušné parametre. Ak máme vyhľadaný parameter, tak porovnávanie jeho argumentov s regulárnym výrazom je sekvenčné. Parametrov budú rádovo jednotky, maximálne desiatky. Do datovej štruktúry sa budú len pridávať, mazanie nie je potrebné.

Zvolená datová štruktúra je dvoj-rozmerný spájaný zoznam. Prvý rozmer je spájaný zoznam prametrov a každý parameter má spájaný zoznam svojich argumentov a pointer na posledný prvok v tomto zozname.

```
struct par_struct {
        char par_name[MAX_PARAMETER_NAME_LENGTH + 1];
        arg_struct_p args;
        arg_struct_p last_arg;
        par_struct_p next;
};

struct arg_struct {
        char value[MAX_ARGUMENT_LENGTH + 1];
        arg_struct_p next;
};
```

6.2.2 Algoritmy

Dvoj-rozmerné spájané zoznamy popísané v minulej kapitole sa v programe vyskutujú dva krát. Každý sa plní v inej fáze behu.

V prvej fáze sa číta konfiguračný súbor a informácie z neho sa ukladajú do prvého zoznamu. V druhej fáze sa parsuje vstupný reťazec. Prečíta sa názov parametru a regulárny výraz popisujúci jeho argumenty. Parameter sa vyhľadá v prvom zozname, jeho argumenty sa porovnajú s prečítaným regulárnym výrazom a tie ktoré mu vyhovujú sa uložia do druhého zoznamu, pod príslušný parameter. Takýmto spôsobom sa naplní druhý zoznam.

Ten sa potom rekurzívne prechádza a skladajú sa všetky kombinácie argumentov pre zadané parametre. Pre každú kombináciu sa zavolá XBW.

Zoznam sa prechádza od prvého prvku, z ktorého sa vybere jeho prvý argument. Rekurzívne sa zavolá funkcia na ďalší prvok v zozname. Po vynorení z rekurzie sa prejde na ďalší argument a opätovne sa volá rekurzia. Rekurzia sa zastaví na poslednom prvku v zozname, so zaznamenaným argumentov od každého prvku a s týmito argumentami sa spustia testy.

6.3 Regulárne výrazy

Pre prácu s regulárnymi výrazmi je použitá GNU knižnica a funkcie z hlavičkového súboru *regex.h*.

6.4 Konfiguračný hlavičkový súbor

Konfiguráčný hlavičkový súbor obsahuje niekoľko predvolených nastavení, ktoré je možné si prispôsobovať. Po zmene je nutná rekompilácia.

XBW_PATH Cesta k binárke XBW.

XBW_COMPRESS_PARAMS Parametre XBW použité pri kompresii.

XBW_DECOMPRESS_PARAMS Parametre XBW použité pri dekompresii.

TIME_COMMAND Príkaz pre meranie času behu XBW a jeho parametre.

DIFF_COMMAND Príkaz pre porovnávanie súborov na zhodu a jeho parametre.

MKDIR_COMMAND Príkaz pre vytvorenie zložky a jeho parametre.

XBW_INPUT_FILES_DIRECTORY Zložka so vstupnými testovacími súbormi pre XBW

XBW_OUTPUT_DIRECTORY Zložka pre ukladanie dočasných súborov.

XBW_COMPRESS_SUFFIX Prípona pre súbory kompresované pomocou XBW.

XBW_DECOMPRESS_SUFFIX Prípona pre dekompresované súbory.

MAX_PARAMETER_NAME_LENGTH Maximálna dĺžka názvu ktoréhokoľvek parametru XBW v konfiguračnom súbore a zadaného do vstupného reťazca. Ak nejaký parameter presiahne túto dĺžku program môže skončiť chybou.

MAX_ARGUMENT_LENGTH Maximálna dĺžka argumentu v konfiguračnom súbore.

CONFIG_FILE Konfiguračný súbor. Viz kapitola 5.3.

Kapitola 7

Záver

Cieľom tejto práce bolo vytvoriť dva praktické nástroje pre uľahčenie vývoja projektu XBW. Myslím, že tento cieľ sa mi, do značnej miery, splniť podarilo. Oba nástroje sú naprogramované podľa požiadavok, dajú sa jednoducho zakomponovať do XBW a používať. V oboch je ale stále čo zlešovať.

Do *libdbg* by sa mohli pridať ďalšie výstupy, ako napríklad databáza, sieťový socket.

Tester by si mohol uchovávať namerané hodnoty časov a kompresných pomerov a nejakým spôsobom ich spracovávať. Vytvárať rôzne štatistiky a vyhodnocovať testy.

Literatura

[1] Šesták, R., Lánský, J: *Compression of Concatenated Web Pages Using XBW*, In: Geffert, V. et al. (Eds.): SOFSEM 2008, LNCS 4910, Springer-Verlag Berlin, Heidelberg, Germany, 2008, pg. 743-754, ISBN 978-3-540-77565-2, ISSN 0302-9743.

[2] Lánský, J., Šesták, R., Uzel, P., Kovalčin, S., Kumičáak, P., Urban, T., Szabó, M.: *XBW - Word-based compression of non-valid XML documents*, http://xbw.sourceforge.net/.

[3] Pokorný, J., Žemlička, M.: *Základy implementace souboru a databází 2. vydání*, Karolinum, 2004.

[4] Herout, P.: *Učebnice jazyka C - 1. díl*, Kopp, 2002, ISBN: 978-80-7232-351-7

[5] Eckel, B.: *Myslíme v jazyku C++*. GRADA Publishing, 2000, ISBN: 80-247-9009-2