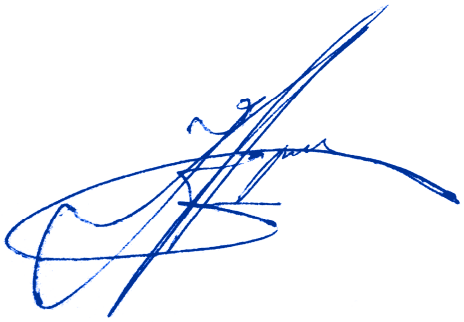
Прототип анализатора XML представлений программ

Нархов Константин Георгиевич

На правах рукописи



Москва. Декабрь 2006 г.

Содержание

1 Описание программы генерации исходного кода на языке Си на основании документа XML

1.1 Общие сведения

Программа генерации исходного кода на языке Си на основании документа XML называется «parser.pl». Программа написана на языке Perl в среде «K Development v2.1». Для её выполнения требуется ЭВМ IBM PC с предустановленной ОС типа UNIX (или MS Windows) и интерпретатором языка Perl. Техническая документация написана с использованием программных продуктов Microsoft Word XP и Corel Draw 12.

1.2 Функциональное назначение

Программа предназначена для анализа специализированных XML документов, описывающих СИ-программу с помощью тэгов, и генерации на основании данного анализа непосредственно кода, готового для компиляции. Данная программа имеет следующие функциональные возможности:

- чтение XML документов и сохранение исходных текстов СИ-программ;
- анализ XML документов в соответствии со специализированным набором тэгов на предмет выявления базовых логических узлов СИ-программы (for, while, …);
- построение дерева базовых логических узлов СИ-программы;
- генерация кода СИ-программы на основании результатов анализа XML-документа
- масштабирование за счет подключения внешних модулей (библиотек) с целью увеличения числа «узнаваемых» базовых логических узлов СИ-программы и тэгов XML;

Управление программой осуществляется посредством консоли (терминала) в ОС типа UNIX или через эмулятор консоли (программа cmd) в ОС MS Windows.

1.3 Описание логической структуры

Логическая структура программы представлена на следующей схеме:

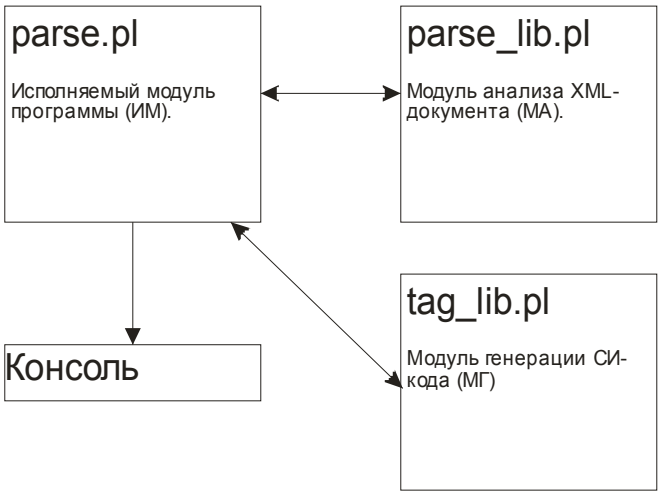


Рисунок 1 – Структурная схема программы

– Исполняемый модуль программы (ИМ) – процесс, управляющий программой. Данный процесс осуществляет разбор XML-документа и построение логического дерева программы. По этому дереву генерируется код СИ-программы. Узлами дерева являются теги <compound-statement></compound-statement>, листьями – теги-утверждения, не имеющие дополнительных вложений <compound-statement></compound-statement>;

– Модуль анализа XML-документа (МА) является хранилищем функций, используемых при построении логического дерева программы ИМ;

– Модуль генерации СИ-кода программы (МГ) является хранилищем функций, используемых ИМ для генерации СИ-кода программы. Обязательными входными параметрами всех функций МГ являются название текущего узлового тега (или тега-утверждения, не имеющего дополнительных вложений <compound-statement></compound-statement>) и порядковый номер строки, содержащей этот тег, в XML-документе. На основании этих данных, а также контекстной зависимости рассматриваемого тега, производится дополнительный разбор XML. По результатам разбора генерируется СИ-код.

1.3.1 Общий алгоритм программы

Список обозначений:

1. glob_level – глобальный текущий уровень. При построении дерева данная переменная обозначает смещение относительно корневых узлов дерева. Корневыми узлами являются теги translation-unit, header-name, declaration, function definition и проч. Изменение этой переменной происходит только после построения всех узлов текущего глобального уровня.
2. rel_level – относительный уровень. Переменная обозначает смещение относительно корневых узлов дерева при построение листьев дерева, т.е. «тупиковых» элементов, не имеющих ссылок на более «глубокие» уровни.
3. mas_strok – массив строк, содержащихся внутри заданных тегов. В программе это либо <function-definition></ function-definition >, либо <compound-statement></compound-statement>.

Общий алгоритм программы приведен на рисунке 2.

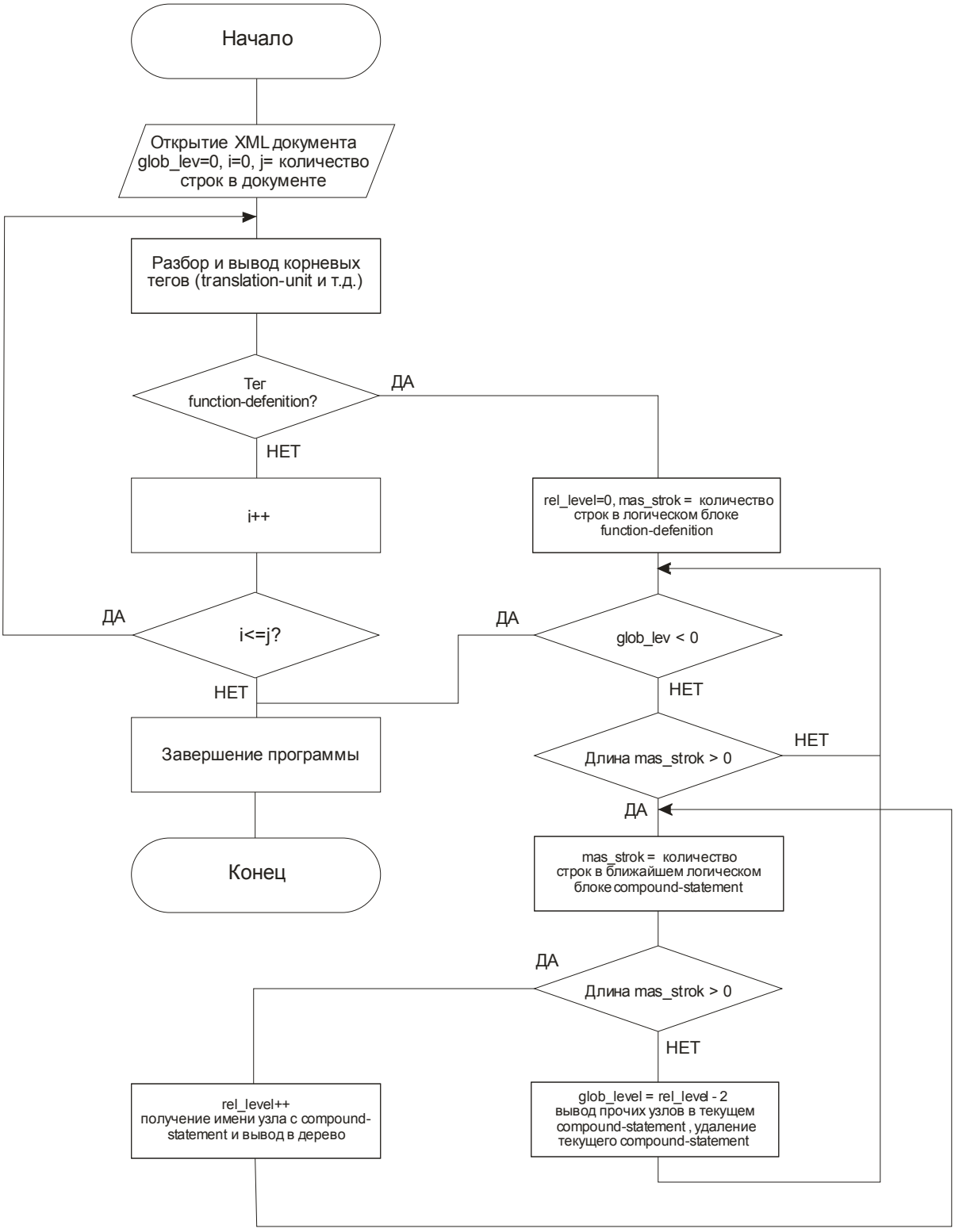


Рисунок 2 – общий алгоритм работы программы

1.4 Используемые технические средства

Для написания исходных текстов и запуска программы «parser.pl» использовалась инструментальная ЭВМ следующей конфигурации:

- Intel Pentium 4, 2.4 ГГц;
- 256 Mb ОЗУ;
- 80 Gb жесткий диск;
- ОС Red Hat Linux 7.3 и ОС Windows XP Professional SP2;
- Интерпретатор ActivePerl 5.8.0.806 под ОС Windows и интерпретатор Perl 5.6.1 под ОС Red Hat Linux 7.3;

1.5 Вызов и загрузка

Программа запускается с консоли инструментальной ЭВМ. Под ОС Windows для этого необходимо выполнить следующие действия:

- Вы меню «Пуск» выбрать подпункт «Выполнить» и запустить эмулятор консоли **`cmd`**;
- Перейти в папку с программой «parser.pl» командой **`cd c:\usr\work\XML\`** (путь может отличаться от приведенного).
- Запустить программу «parser.pl» командой **`perl parser.pl OsThreadUsr.xml`**, где OsThreadUsr.xml – XML файл, на основании которого будет сгенерирован СИ-код.
- Если необходимо просмотреть логическое дерево программы тэгов XML-файла следует вызвать программу с ключом **`/tree`**

1.6 Входные данные

Программа имеет всего два входных параметра: имя XML-документа и флаг вывода логического дерева программы. Имя XML-документа может быть указано как в относительном виде, так и в абсолютном. Иными словами запуск будет успешен и в случае **`perl parser.pl c:\usr\work\XML\OsThreadUsr.xml`**, и в случае **`perl parser.pl OsThreadUsr.xml`** (при условии, что программа запускается из каталога **`c:\usr\work\XML\`**). Параметр вывода логического дерева программы **`/tree`** является необязательным и при его отсутствии производится генерация СИ-кода. В случае присутствия этого параметра генерация СИ-кода сопровождается выводом логического дерева программы на консоль.

1.7 Выходные данные

Выходными данными являются:

- Файл с СИ-кодом, готовым для компиляции. Этот файл помещается в папку, откуда была запущена программа.
- Информация, которая выводится оператору на консоль. В случае вызова программы с флагом **`/tree`** на консоль будет выведено логическое дерево программы. В любом другом случае на консоль выводится сгенерированный СИ-код.

Следует заметить, что генерация СИ-кода программы не зависит от флага **`/tree`** и производится в любом случае.

2 Тестирование и отладка

2.1 Конфигурирование ОС для работы программы

Конфигурирование ОС для работы программы сводится к установке дистрибутива интерпретатора Perl.

2.2.1 ОС типа UNIX

Для установки Perl на ОС типа UNIX следует выполнить следующие действия (на примере) ОС Red Hat Linux 7.3:

- Скачать свежий дистрибутив (например с http://cpan.org/ports/index.html);
- Распаковать в какую-либо папку на инструментальном ЭВМ;
- Выполнить последовательность команд: **`perl MakeFile.pl; make; make test; make install;`**

Следует обратить внимание на то, что большинство дистрибутивов ОС типа Unix поставляются со встроенным интерпретатором Perl. Проверить наличие Perl в ОС типа Unix можно командой **`perl -v`**. При наличии интерпретатора на экран будет выдана версия установленного дистрибутива Perl.

2.2.2 ОС типа Windows

Для установки Perl на ОС типа Windows следует выполнить следующие действия (на примере) ОС Windows XP Professional:

- Скачать свежий дистрибутив ActivePerl с сайта производителя (http://www.ActiveState.com/ActivePerl/);
- Запустить инсталлятор и следовать подсказкам мастера установки до завершения установки;

Рекомендуется установить Perl в папку **`usr`** на тот же диск, где располагается ОС Windows (обычно **`c:\usr`**).

2.2 Тестирование программы

Программа тестировалась на разборе файла **`osTstThread.xml`** (XML код приведен в приложении). Запуск осуществлялся как с флагом **`/tree`**, так и без него.

2.2.2 Запуск без флага /tree

Для запуска следует выполнить последовательность команд, описанную в пункте 1.5. Непосредственно программа будет вызываться командой: **`perl parser.pl osTstThread.xml`**. Сгенерированный СИ-код будет помещен в файл **`common.c`** (см. приложение), а также выведен на консоль (см. Рисунок 3). Запуск под ОС типа Unix аналогичен, за исключением написания пути к рабочей папке: **`cd /usr/work/XML/`**.

2.2.3 Запуск без флага /tree

Для запуска следует выполнить последовательность команд, описанную в пункте 1.5 Непосредственно программа будет вызываться командой: **`perl parser.pl osTstThread.xml /tree`**. Сгенерированный СИ-код будет помещен в файл **`common.c`** (см. приложение), а также выведен на консоль (см. Рисунок 4). Запуск под ОС типа Unix аналогичен, за исключением написания пути к рабочей папке: **`cd /usr/work/XML/`**.

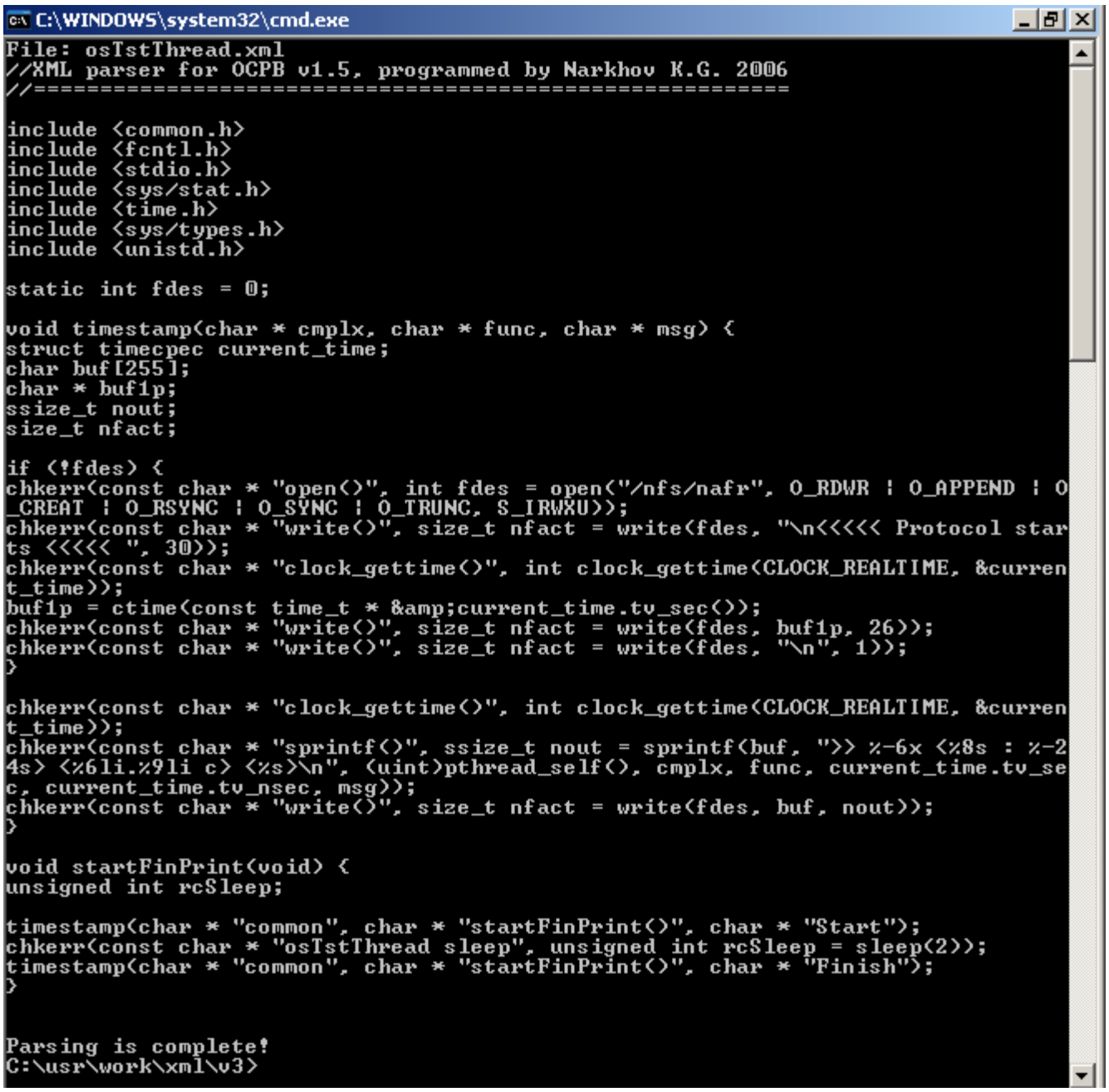


Рисунок 3 – Результат выполнения perl parser.pl osTstThread.xml.

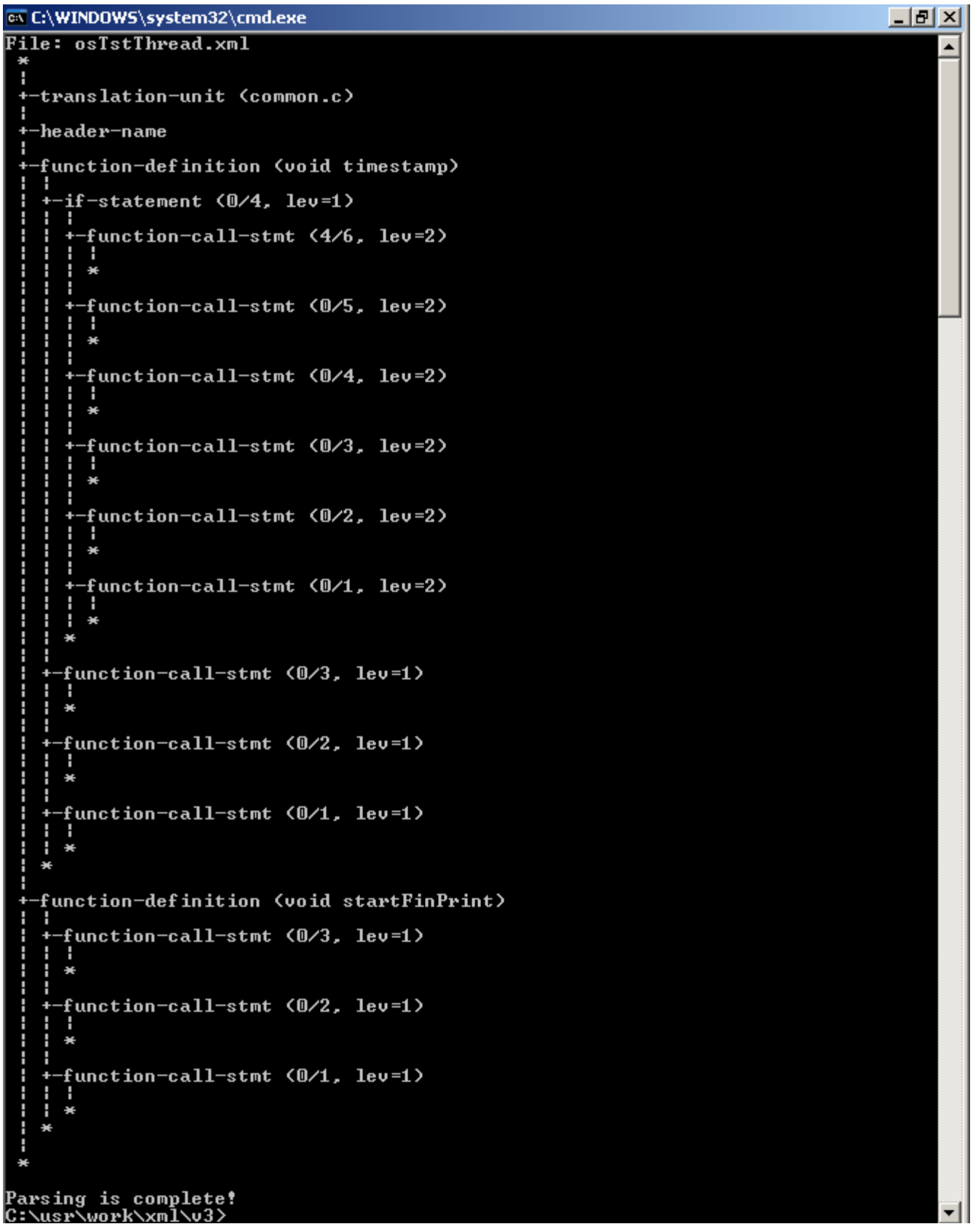


Рисунок 3 – Результат выполнения perl parser.pl osTstThread.xml /tree

Приложение А

XML-документ **osTstThread.xml**:

```
<translation-unit file-name="common.c" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:/E:/Work2006/RUP/XML-schema/C-objects.xsd">
    <header-name header-type="sys" path=""> common.h</header-name>
    <declaration>
        <var-declaration type="static int">
            <identifier> fdes</identifier>
            <assign/>
            <expression>
                <token> 0</token>
            </expression>
            <semicolon/>
        </var-declaration>
    </declaration>
    <function-definition function-name="timestamp" function-type="void">
        <open-parenthesis/>
        <parameter parameter-type="char *"> cmplx</parameter>
        <comma/>
        <parameter parameter-type="char *"> func</parameter>
        <comma/>
        <parameter parameter-type="char *">msg</parameter>
        <close-parenthesis/>
        <compound-statement>
            <declaration>
                <var-declaration type="struct timecpec">
                    <identifier> current_time</identifier>
                    <semicolon/>
                </var-declaration>
                <var-declaration type="char">
                    <identifier> buf[255]</identifier>
                    <semicolon/>
                </var-declaration>
                <var-declaration type="char *">
                    <identifier> buf1p</identifier>
                    <semicolon/>
                </var-declaration>
                <var-declaration type="ssize_t">
                    <identifier> nout</identifier>
                    <semicolon/>
                </var-declaration>
                <var-declaration type="size_t">
                    <identifier> nfact</identifier>
                    <semicolon/>
                </var-declaration>
            </declaration>
            <if-statement>
                <open-parenthesis/>
                <expression>
                    <token>!fdes</token>
                </expression>
                <close-parenthesis/>
                <compound-statement>
                    <function-call-stmt>
                        <header-name header-type="sys" path="sys/">types.h</header-name>
                        <header-name header-type="sys">sys/stat.h</header-name>
                        <header-name header-type="sys">fcntl.h</header-name>
                        <function-call function-name="chkerr" function-type="void">
                            <open-parenthesis/>
                            <parameter parameter-type="const char *">"open()"</parameter>
                            <comma/>
                            <parameter parameter-type="int">fdes = open("/nfs/nafr", O_RDWR |
                                O_APPEND | O_CREAT | O_RSYNC | O_SYNC | O_TRUNC,
S_IRWXU)</parameter>
                            <close-parenthesis/>
                        </function-call>
                        <semicolon/>
                    </function-call-stmt>
                    <function-call-stmt>
                        <header-name header-type="sys">unistd.h</header-name>
                        <function-call function-name="chkerr" function-type="void">
                            <open-parenthesis/>
                            <parameter parameter-type="const char *">"write()"</parameter>
                            <comma/>
                            <parameter parameter-type="size_t">nfact = write(fdes,
                                "\n&lt;&lt;&lt;&lt;&lt; Protocol starts
```

```
                            &amp;lt;&amp;lt;&amp;lt;&amp;lt;&amp;lt; ", 30)</parameter>
                        <close-parenthesis/>
                    </function-call>
                    <semicolon/>
                </function-call-stmt>
                <function-call-stmt>
                    <header-name header-type="sys">time.h</header-name>
                    <function-call function-name="chkerr" function-type="void">
                        <open-parenthesis/>
                        <parameter parameter-type="const char
*">"clock_gettime()"</parameter>
                        <comma/>
                        <parameter parameter-type="int">clock_gettime(CLOCK_REALTIME,
                            &current_time)</parameter>
                        <close-parenthesis/>
                    </function-call>
                    <semicolon/>
                </function-call-stmt>
                <function-call-stmt>
                    <header-name header-type="sys">time.h</header-name>
                    <var type="char *">
                        <identifier>buf1p</identifier>
                    </var>
                    <assign/>
                    <function-call function-name="ctime" function-type="char *">
                        <open-parenthesis/>
                        <parameter parameter-type="const time_t *"
                            >&amp;amp;current_time.tv_sec()</parameter>
                        <close-parenthesis/>
                    </function-call>
                    <semicolon/>
                </function-call-stmt>
                <function-call-stmt>
                    <header-name header-type="sys">unistd.h</header-name>
                    <function-call function-name="chkerr" function-type="void">
                        <open-parenthesis/>
                        <parameter parameter-type="const char *">"write()"</parameter>
                        <comma/>
                        <parameter parameter-type="size_t">nfact = write(fdes, buf1p,
26)</parameter>
                        <close-parenthesis/>
                    </function-call>
                    <semicolon/>
                </function-call-stmt>
                <function-call-stmt>
                    <header-name header-type="sys">unistd.h</header-name>
                    <function-call function-name="chkerr" function-type="void">
                        <open-parenthesis/>
                        <parameter parameter-type="const char *">"write()"</parameter>
                        <comma/>
                        <parameter parameter-type="size_t">nfact = write(fdes, "\n",
1)</parameter>
                        <close-parenthesis/>
                    </function-call>
                    <semicolon/>
                </function-call-stmt>
            </compound-statement>
        </if-statement>
        <function-call-stmt>
            <header-name header-type="sys">time.h</header-name>
            <function-call function-name="chkerr" function-type="void">
                <open-parenthesis/>
                <parameter parameter-type="const char *">"clock_gettime()"</parameter>
                <comma/>
                <parameter parameter-type="int">clock_gettime(CLOCK_REALTIME,
&current_time)</parameter>
                <close-parenthesis/>
            </function-call>
            <semicolon/>
        </function-call-stmt>
        <function-call-stmt>
            <header-name header-type="sys">stdio.h</header-name>
            <function-call function-name="chkerr" function-type="void">
                <open-parenthesis/>
                <parameter parameter-type="const char *">"sprintf()"</parameter>
                <comma/>
                <parameter parameter-type="ssize_t">nout = sprintf(buf, "&gt;&gt; %-6x
                    &lt;%8s : %-24s&gt; &lt;%6li.%9li c&gt;
                    &lt;%s&gt;\n", (uint)pthread_self(), cmplx, func,
                    current_time.tv_sec, current_time.tv_nsec, msg)</parameter>
```

```
                    <close-parenthesis/>
                </function-call>
                <semicolon/>
            </function-call-stmt>
            <function-call-stmt>
                <header-name header-type="sys">unistd.h</header-name>
                <function-call function-name="chkerr" function-type="void">
                    <open-parenthesis/>
                    <parameter parameter-type="const char *">"write()"</parameter>
                    <comma/>
                    <parameter parameter-type="size_t">nfact = write(fdes, buf, nout)</parameter>
                    <close-parenthesis/>
                </function-call>
                <semicolon/>
            </function-call-stmt>
        </compound-statement>
    </function-definition>
    <function-definition function-name="startFinPrint" function-type="void">
        <open-parenthesis/>
        <parameter parameter-type="void"/>
        <close-parenthesis/>
        <compound-statement>
            <declaration>
                <var-declaration type="unsigned int">
                    <identifier>rcSleep</identifier>
                    <semicolon/>
                </var-declaration>
            </declaration>
            <function-call-stmt>
                <function-call function-name="timestamp" function-type="void">
                    <open-parenthesis/>
                    <parameter parameter-type="char *">"common"</parameter>
                    <comma/>
                    <parameter parameter-type="char *">"startFinPrint()"</parameter>
                    <comma/>
                    <parameter parameter-type="char *">"Start"</parameter>
                    <close-parenthesis/>
                </function-call>
                <semicolon/>
            </function-call-stmt>
            <function-call-stmt>
                <header-name header-type="sys">unistd.h</header-name>
                <function-call function-name="chkerr" function-type="void">
                    <open-parenthesis/>
                    <parameter parameter-type="const char *">"osTstThread sleep"</parameter>
                    <comma/>
                    <parameter parameter-type="unsigned int">rcSleep = sleep(2)</parameter>
                    <close-parenthesis/>
                </function-call>
                <semicolon/>
            </function-call-stmt>
            <function-call-stmt>
                <function-call function-name="timestamp" function-type="void">
                    <open-parenthesis/>
                    <parameter parameter-type="char *">"common"</parameter>
                    <comma/>
                    <parameter parameter-type="char *">"startFinPrint()"</parameter>
                    <comma/>
                    <parameter parameter-type="char *">"Finish"</parameter>
                    <close-parenthesis/>
                </function-call>
                <semicolon/>
            </function-call-stmt>
        </compound-statement>
    </function-definition>
</translation-unit>
```

Приложение Б

Сгенерированный файл с исходным СИ-кодом **common.c**:

```
//XML parser for OCPB v1.5, programmed by Narkhov K.G. 2006
//=========================================================

include <common.h>
include <fcntl.h>
include <stdio.h>
include <sys/stat.h>
include <time.h>
include <sys/types.h>
include <unistd.h>

static int fdes = 0;

void timestamp(char * cmplx, char * func, char * msg) {
struct timecpec current_time;
char buf[255];
char * buf1p;
ssize_t nout;
size_t nfact;

if (!fdes) {
chkerr(const char * "open()", int fdes = open("/nfs/nafr", O_RDWR | O_APPEND | O_CREAT | O_RSYNC
| O_SYNC | O_TRUNC, S_IRWXU));
chkerr(const char * "write()", size_t nfact = write(fdes, "\n<<<<< Protocol starts <<<<< ", 30));
chkerr(const char * "clock_gettime()", int clock_gettime(CLOCK_REALTIME, &current_time));
buf1p = ctime(const time_t * &amp;current_time.tv_sec());
chkerr(const char * "write()", size_t nfact = write(fdes, buf1p, 26));
chkerr(const char * "write()", size_t nfact = write(fdes, "\n", 1));
}

chkerr(const char * "clock_gettime()", int clock_gettime(CLOCK_REALTIME, &current_time));
chkerr(const char * "sprintf()", ssize_t nout = sprintf(buf, ">> %-6x <%8s : %-24s> <%6li.%9li c>
<%s>\n", (uint)pthread_self(), cmplx, func, current_time.tv_sec, current_time.tv_nsec, msg));
chkerr(const char * "write()", size_t nfact = write(fdes, buf, nout));
}

void startFinPrint(void) {
unsigned int rcSleep;

timestamp(char * "common", char * "startFinPrint()", char * "Start");
chkerr(const char * "osTstThread sleep", unsigned int rcSleep = sleep(2));
timestamp(char * "common", char * "startFinPrint()", char * "Finish");
}
```