УТВЕРЖДЕН

643.ИАИА.00001-01 33 01-ЛУ

ШАБЛОН ПРОГРАММЫ-СЕРВЕРА ДЛЯ *QNX6*

Руководство программиста

643.ИАИА.00001-01 33 01

Листов 12

2018

Литера

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В документе описана методика написания программ на основе шаблона. В приложении приведен законченный пример программы, реализующий практически все варианты обмена сообщениями.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЕ ПРИМЕНЕНИЯ ПРОГРАММЫ

Шаблон предназначен для разработчиков программного обеспечения для *QNX6*. Является основой для написания компонентов программных комплексов. Самостоятельного применения не имеет.

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

Шаблон является основой написания программ на *QNX6*. Он содержит все необходимое для работы программы как сервера. Включена обработка сообщений от операционной системы и некоторых часто используемых команд. Выработано соглашение об именовании. Имеются механизмы расширения функциональности путем добавления и путем наследования.

Совокупность свойств шаблона позволяет ускорить написание программ, повысить их надежность, сделать их однотипными, что дает лучшее понимание и упрощает сопровождение.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

Описана работа с шаблоном на языке Си. Для версии на C++ действия аналогичны.

3.1. Переименование шаблона

Для подготовки шаблона к работе его необходимо переименовать в соответствии с именем программы.

За основу можно использовать шаблон *server*, но он включает в себя достаточный объем текста, не относящийся к прикладной программе. Путем наследования получен шаблон *program*. Рекомендуется использовать именно его.

Для переименования шаблона требуется выполнить следующие действия:

1) создать директорию с именем программы;
2) переписать в нее файлы из директории *program*;
3) в именах файлов заменить *program* на имя программы;
4) в файлах, включая *Makefile*, заменить все вхождения *program* на имя программы с учетом регистра;
5) в файлах заменить все вхождения префикса *pr* на уникальный префикс программы.

Теперь программа подготовлена к расширению функциональности.

3.2. Расширение путем добавления

3.2.1. Добавление обработки команды

Для добавления обработки команды надо выполнить следующие действия:

1) добавить тип команды в соответствующее перечисление;
2) если команда имеет параметры, добавить описание структуры и включить ее в объединение структур принимаемых сообщений;
3) если команда имеет ответное сообщение, добавить описание структуры и включить ее в объединение структур ответных сообщений;
4) в функции *ProcessMsg()* добавить вариант с типом команды, вызвать фактический обработчик;

5) если команда имеет принимаемое или ответное сообщение, применить запись (извлечение) параметров согласно подразделу 3.4;
6) если команда имеет ответное сообщение, задать значение переменной *psize*;
7) если обработка предполагает отложенный ответ, передать в фактический обработчик переменную *rcvid*, при этом обработчик должен возвращать значение (-1);
8) написать фактический обработчик;
9) добавить в библиотеку соответствующую функцию-обертку.

3.2.2. Добавление обработки импульса

Для добавления обработки импульса надо выполнить следующие действия:

1) добавить код импульса в соответствующее перечисление;
2) в функции *ProcessPulse()* добавить вариант с кодом импульса, вызвать фактический обработчик;
3) если требуется, передать в фактический обработчик значение импульса;
4) написать фактический обработчик;
5) добавить в библиотеку соответствующую функцию-обертку, если импульс поступает от других программ.

3.3. Расширение путем наследования

1) выполнить переименование шаблона согласно подразделу 3.1;
2) в функциях *ProcessMsg()* и *ProcessPulse()* в варианте *default* заменить префикс *se* на префикс родительской программы;
3) в файле <*name*>*_srr.h* задать номер уровня наследования в начальных значениях типов команд и кодов импульса.
4) в файл <*name*>*_lib.h* включить файл <*parent*>*_lib.h*;
5) добавить подключение библиотеки в файле *Makefile*.

Программа готова к расширению путем добавления.

3.4. Запись (извлечение) параметров

Принимаемые и ответные сообщения передаются в виде структур. Для удобства обращения к программе создается библиотека, содержащая функции-обертки над

операциями обмена сообщениями. Эти функции имеют параметры, которые упаковываются в структуру передаваемого сообщения и заполняются из ответного.

Удобно, чтобы фактические обработчики имели параметры, совпадающие с параметрами функций из интерфейсной библиотеки. Для этого в функциях *ProcessMsg()* и *ProcessPulse()* в вариантах типов команд путем фигурных скобок создается составной оператор. В нем объявляются переменные с типами указателей на принимаемую и ответную структуру, инициализируемые значениями принимаемого и ответного сообщений.

При вызове фактических обработчиков, используя созданные указатели, в качестве параметров подставляются поля структур.

Таким образом, сигнатуры функций в интерфейсной библиотеке и фактических обработчиков совпадают, кроме дескриптора соединения. В реализации на C++ совпадение полное.

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Принимаемые и ответные сообщения полностью описываются в файле *<name>_srr.h*.

Шаблон *server* обрабатывает две команды и один импульс.

Команды описаны в таблице 1, импульсы в таблице 2.

Таблица 1 – Команды шаблона *server*

Тип команды	Принимаемое сообщение	Ответное сообщение	Действия
se_t_quit	*se_r_cmd_t*	отсутствует	Завершение работы
se_t_evdlvr	*se_r_evdlvr_t*	отсутствует	Событие для последующего возвращения

Таблица 2 – Импульсы шаблона *server*

Код импульса	Значение импульса	Действия
se_c_quit	Не используется	Завершение работы

Шаблон *program* получен путем наследования шаблона *server*. Собственных сообщений не имеет, их следует добавлять по необходимости.

5. СООБЩЕНИЯ

На этапе инициализации программы выполняются некоторые проверки. В случае невыполнения условий выводятся сообщения в файл *stderr* (см. таблицу 1), работа прекращается.

Таблица 1 – Сообщения при инициализации

Сообщение	Причина
`Callback is not implemented`	Не все внешние вызовы инициализированы
`Can not allocate memory for a message`	Невозможно выделить память для входных или выходных сообщений
`Can not attach the name`	Ошибка при регистрации имени
`Can not establish a connection to self`	Ошибка при соединении с каналом

Во время работы выполняются некоторые проверки. В случае невыполнения условий выводятся сообщения в файл *stderr* (см. таблицу 2), работа продолжается.

Таблица 2 – Сообщения при работе

Сообщение	Причина
`MsgReceive error`	Ошибка при приеме сообщения
`MsgReply error`	Ошибка при ответе сообщения
`unsupported command type`	Тип поступившей команды не поддерживается
`unsupported pulse code`	Код поступившего импульса не поддерживается

Если в качестве обработчика сигналов задана функция *SignalHandler()* из файла *tools*, то при получении сигнала выдается сообщение "`caught signal. Program terminated`", работа прекращается.

ПРИЛОЖЕНИЕ

ПРИМЕР ИСПОЛЬЗОВАНИЯ ШАБЛОНА - ПРОГРАММА *SAMPLE*

В качестве примера использования шаблона *program* была разработана программа *sample*.

В ней продемонстрированы практически все возможные методы обмена сообщениями, а именно:

- команда без данных;
- передача данных серверу;
- запрос данных от сервера;
- передача данных в двух направлениях;
- отложенный ответ;
- двусторонняя передача данных с использованием векторов;
- посылка импульса самой программе;
- обработка импульса;
- использование таймера.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопровод. докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					

Изм. *Подпись* *Дата*