PIMR 1.0.0: Руководство администратора
Версия 1.0.0
www.gradsoft.kiev.ua

October 24, 2002

Contents

1 Общие сведения

1.1 Состав и назначение пакета

Пакет PIMR ("репозитарий реализаций", далее просто "репозитарий") предназначен для повышения доступности CORBA-сервисов ("реализаций"). Кроме того репозитарий может использоваться для упрощения конфигурирования при использовании нескольких взаимосвязанных CORBA-сервисов вместо name-service. Функции репозитария состоят в отслеживании доступности сервисов и перенаправлении запросов работающему сервису. С этой целью им поддерживается реестр сервисов. Репозитарий предназначен для работы под управлением одной из Unix-подобных ОС или Windows NT. Написан на C++ с использованием GradC++ToolBox.

1.2 Где взять

PIMR всегда доступен в CVS (модуль "COS/PIMR").

1.3 Авторская группа

Пакет PIMR разработан компанией GradSoft, домашняя страничка GradSoft http://www.gradsoft.com.ua

2 Инсталляция

2.1 Общие сведения

Пакет PIMR распространяется в исходных кодах, его установка включает следующие действия:

1. Компиляция - компиляция исходного кода в окружении пользователя.
2. Установка - копирование исполняемого модуля и других файлов в указанные пользователем каталоги.
3. Настройка (см. дальше).

В процессе работы при помощи команды make с нужными опциями (см. ниже) Вы можете:

1. откомпилировать пакет;
2. инсталлировать предварительно откомпилированный пакет;
3. удалить объекты, созданные в процессе компиляции;
4. удалить переписанные файлы

2.2 Необходимое ПО

1. Компилятор C++ :
   - Unix: gcc
   - Windows NT: Microsoft Visual C версии 6.0 или выше
2. Утилита make:
   - Unix: Необходим gnu make
   - WindowsNT: nmake из поставки MSVC++
3. Библиотека GradC++Toolbox 1.5.0 и выше
4. CORBA ORB:
   - Unix: ORBacus 4.0.1 или выше
   - Windows NT: ORBacus 4.0.1 или выше
5. Tcl интерпритатор (используется для интерпретирования кофигурационного файла):
   - Unix: Tcl-8.3.x или выше.
   - Windows NT: ActiveState ActiveTcl 8.3.4.2 или выше

2.3 Порядок инсталляции под UNIX

1. Убедиться, что необходимое ПО установленно и работатет.
2. Развернуть архив pimr-1.0.0.tar.gz либо pimr-1.0.0.zip в избранном Вами каталоге (в дальнейшем этот каталог мы будем называть "корневым каталогом проекта", и обозначать `<project_root>`).
3. Перейти в этот каталог.
4. Запустить configure командой `./configure` с нужными опциями (список опций выводтися при запуске `./configure` с опцией `--help`; в частности, можно задать опцию `--prefix=<smth>` для назначения каталога инсталляции)
5. Запустить компиляцию, с помощью команды `gmake`
6. Перейти в root режим с помощью команды `su`
7. Запустить инсталляцию с помощью команды `gmake install`
8. Для деинсталяции пакета можно воспользоваться командой gmake uninstall

2.4 Порядок инсталляции под Windows NT

1. Убедиться, что необходимое ПО установленно и работатет. В настоящее время для работы с MSVC требуется, чтобы:
   (a) переменные среды пользователя INCLUDE и LIB были определены и содержали пути к включаемым файлам и библиотекам MSVC соответственно.
   (b) пути к утилитам **nmake**, **cl**, **link** были прописаны в переменной среды пользователя `PATH`.
2. Развернуть архив pimr-1.0.0.tar.gz либо pimr-1.0.0.zip в избранном Вами каталоге (назовём его `<project_root>`). Отредактировать файл **environment.nt** в подкаталоге `<project_root>\config\Win32`. При редактировании установить значения следующих nmake-переменных:
   - PROJECT_DIR - это `<project_root>`.
   - INSTALL_DIR (каталог инсталляции) - исполняемые файлы будут размещены в подкаталоге **bin**, заголовочные файлы - в подкаталоге **include**, idl-описания - в подкаталоге **idl**, файлы конфигурации по умолчанию - в директории **conf** данного каталога.
   - GRADCTOOLBOX_INSTALL_DIR - директория с установленным GradC++Toolbox. Необходимо чтобы этот компонент был собран для того-же ORB, что и собираемый PIMR.
   - ORB (условное имя ORB) - эта переменная должна принимать значения ORBacus, либо OmniORB, либо TAO в зависимости от марки Вашего ORB. В зависимости от этой переменной должны быть сконфигурированы
     - ACE_ROOT - корневой каталог ACE (в том случае, если Вы используете TAO)
     - TAO_ROOT - корневой каталог TAO (в том случае, если Вы используете TAO, и TAO находится вне дерева каталогов ACE)
     - OMNI_ROOT - корневой каталог omniORB (в том случае, если Вы используете omniORB)
     - OOC_ROOT - корневой каталог ORBacus (в том случае, если Вы используете ORBacus)
   - TCL_DIR - директория с установленным TCL.
3. Для компиляции пакета перейти в каталог `<project_root>` и воспользоваться командой `make` (`make build`).
4. Для инсталляции пакета перейти в каталог `<project_root>` и воспользоваться командой `make install`.
5. Для удаления инсталлированных файлов перейти в каталог `<project_root>` и воспользоваться командой `make uninstall`.

6. Для для удаления файов, созданных в процессе компиляции перейти в каталог `<project_root>` и воспользоваться командой `make clean`.

3 Настройка

Перед использованием PIMR его нужно настроить. Процесс настойки PIMR состоит в настройке конфигурационного файла и использовании аргументов командной строки. Справку по аргументам командной строки можно получить запустив PIMR с опцией `--help`, допускается использование следующих опций:

- `--help`

  Отображает встроенную справку по опциям командной строки.

- `--config имя_файла`

  Позволяет загружать аргументы командной строки из файла.

- `--with-naming инициальный_объект`

  Установка инициальных объектов.

- `--ior-stdout`

  Объектная ссылка на репозитарий печатается на стандартном выводе.

- `--ior-file-PIMR имя_файла`

  Если эта опция указана в командной строке, репозиторий попытается поместить в этот файл IOR-ссылку на сервис репозитория.

И непосредственно опции репозитария:

- `--configfile конфигурационный_файл`

  Указывает репозиторию использовать этот конфигурационный файл. Если эта директива не указана репозиторий попытается использовать файл `pimr.cfg` в текущей директории. Не нужно путать эту опцию с опцией `--config`.

- `--logfile лог_файл`

  Указывает репозиторию использовать этот файл для записи отчётов. Если эта директива не указана репозиторий попытается использовать файл `pimr.log` в текущей директории. О лог-файле будет сказано в разделе "использование".

Можно также указать опции для используемого вами объектного брокера.

Например, репозитарий, использующий ORBACUS и запущеный с такими опциями:

```
pimr.exe
   --configfile conf.pimr --logfile log.pimr
   -OAport 12345 -ORBtrace_connections 2
   --ior-stdout
```

прочитает конфигурацию репозитория из файла "conf.pimr"; будет вести лог в файле "log.pimr"; орб репозитария будет использовать порт 12345 и выдавать тестовые сообщения; при запуске репозитарий выведет свой IOR.

Файл конфигурации репозитория является программой на языке TCL. Для настройки PIMR в язык добавлено несколько команд. Это:

- `service {имя сервиса}`
  `--ior {ior ссылка}`
  `--repositoryid {idl интерфейс}`
  `[--restart {команда перезапуска}]`
  `[--callback {ior ссылка}]`

  Информирует репозитарий о сервисе и его параметрах. Это необходимо для того, чтобы рапозиторий мог перенаправлять запросы к сервису и отслеживать его состояние. Установка `--restart` указывает что сервис нужно перезапускать если он упадёт, этот аргумент не обязательный.

- `forwardfreq {число}`

  Указывает репозитарию инициировать проверку состояния сервисов при перенаправлении не реже, чем каждые N секунд.

- `checkfreq {число}`

  Указывает репозитарию инициировать проверку доступности сервисов не реже, чем каждые N секунд.

- `redirecttimeout {число}`

  Устанавливает время ожидания при перенаправлении запроса в секундах.

- `obsoletetimeout {число}`

  Указывает репозитарию перепроверять доступность сервиса по истечении N секунд после предидущей проверки.

- `restarttimeout {число}`

  Устанавливает время ожидания перезапуска сервиса в секундах. Если сервис находится в состоянии перезагрузки дольше - он помечается как сбойный и больше не перезапускается.

Рассмотрим для примера такой конфигурационный файл:

```
#
# This is TCL demo config file for PIMR
#
set thisHost "127.0.0.1"
```

```
set thisPort 11000

service  HelloService \
    --ior "corbaloc::${thisHost}:${thisPort}/HelloService" \
    --repositoryid "IDL:Demo/HelloWorlder:1.0" \

#    --restart "startHelloWorlder.bat" \  This lines are commented
#    --host "${thisHost}:${thisPort}" \   Note that ${thisHost}
#    --callback "" \                      will be computed by TCL.

forwardfreq 1
checkfreq 10
obsoletetimeout 15
restarttimeout 30
redirecttimeout 60
```

Этот конфигурационный файл сообщает информацию об одном сервисе с именем HelloService. Репозиторий будет искать его на локальной машине на порту 11000 под именем HelloService. Репозитарий не будет пытаться запускать сервис, если тот остановится так как не указана команда запуска.

Желательно выбирать параметры репозитория с соблюдением правил: obsoletetimeout + checkfreq + restarttimeout < redirecttimeout

4 Тестирование

4.1 Общие сведения

Для проверки работоспособности откомпилированного пакета предоставляются тестовые примеры. Для того, чтобы было, что запускать, надо сначала откомпилировать репозиторий и тестовые примеры. Откомпилировать тестовые примеры под Windows можно из каталога `<project_root>` выполнив команду:

`nmake /f makefile.nt buildtests`

Чтобы удалить скомпилированное нужно выполнить:

`nmake /f makefile.nt cleantests`

Под Unix-подобной системой воспользуетесь ???.

Теперь передите в каталог `<project_root>/demo/` и осмотритесь. Прочитайте "readme"-файлы в этой директории и поддиректориях.

В поддиректориях `<project_root>/demo/server*/` размешены тестовые CORBA-сервисы. Для упрощения запуска там же созданы файлы `run.bat` Описания интерфейсов примеров размещены в файлах `<project_root>/demo/*.idl`.

Директория `<project_root>/demo/client/` содержит прграмму-клиент для этих сервисов. Там же расположены файлы `run_straight.bat` и `run_via_repository.bat` предназначенные для запуска клиента соответственно напрямую и через репозитарий.

Директория `<project_root>/demo/repository/` сожержит конфигурации репозитория для демо-примеров и скрипты запуска `repository*.bat`.

Скрипты настроены на использование ip-адреса 127.0.0.1 и портов 11000 (11001), 11111 сервисом и репозитарием соответственно. Поэтому перед запуском убедитесь что данные порты свободны для использования.

4.2 Порядок тестирования под Windows

- Откомпилируйте репозиторий и тестовые примеры.
- Проверка клиента и сервиса

  Запустите клиента. Клиент должен завершиться с ошибкой, говорящей что сервис недоступен.

  Запустите программу-сервис HelloWorld выполнив `run.bat` в директории `<project_root>/demo/server0/` .

  Затем не останавливая сервис выполните программу-клиент с помощью скрипта `run_straight.bat` из директории `<project_root>/demo/client/` . Клиент должен выполниться без ошибок а в окне сервиса должна появиться надпись "Hello, world".
- Тест 1 (тест способности репозитария перенаправлять запрос)
  - Запустите репозиторий с помощью `repository.bat` но не запускайте сервис. Убедитесь что в репозитарии сервис HelloWorld считается остановленым.
  - Запустите клиента с помощью `run_via_repository.bat`. Клиент должен получить исключение `OBJECT_NOT_EXIST`.
  - Запустите сервис HelloWorld выполнив `run.bat` в директории `<project_root>/demo/server0/` .
  - Подождите (не больше минуты) пока запуск сервиса не будет обнаружен репозиторием.
  - Запустите клиента с помощью `run_via_repository.bat`. Клиент должен выполниться без ошибок а в окне сервиса должна появиться строка "Hello, world".
  - Закройте сервис и сразу же запустите клиента. Клиент должен получить исключение `OBJECT_NOT_EXIST.`
  - Остановите репозитарий.
- Тест 2 (тест способности репозитария возобновлять работу сервиса)
  - Запустите репозиторий с помощью `repository1.bat` но не запускайте сервис. Подождите (не больше минуты) - репозитарий должен запустить сервис с помощью скрипта `restart.bat`.
  - Запустите клиента с помощью `run_via_repository.bat`. Клиент должен выполниться без ошибок а в окне сервиса должна появиться строка "Hello, world".

- Закройте сервис и сразу же запустите клиента. Не больше чем через минуту репозитарий должен запустить сервис а клиент - завершить запрос без ошибок.
- Остановите репозитарий.

- Тест 3 (тест способности репозитария воспринимать сообщещние "activated")
  - Запустите сервис HelloWorld выполнив `run.bat` в директории `<project_root>/demo/server0/`.
  - Запустите клиента с помощью `run_straight.bat`. Клиент должен выполниться без ошибок а в окне сервиса должна появиться строка "Hello, world".
  - Запустите репозиторий с помощью `repository.bat`.
  - Подождите (не больше минуты) пока сервис не будет обнаружен репозиторием.
  - Остановите сервис и запустите его из директории `<project_root>/demo/server1/`.
  - Подождите (не больше минуты) пока сервис снова не будет обнаружен репозитарием.
  - Запустите клиента с помощью `run_via_repository.bat`. Клиент должен выполниться без ошибок а в окне сервиса должна появиться строка "Hello, world".
  - Запустите клиента с помощью `run_straight.bat`. Клиент должен выполниться с ошибкой, говорящей что сервис недоступен (потому что он работает на другом порту).
  - Остановите репозитарий.
-

5 Использование

Для того чтобы репозитарий отслеживал сервис его нужно зарегистрировать в нём. Это возможно либо из конфигурационного файла, либо с помощью интерфейса RepositoryAdmin. Тогда сервис будет доступен по ссылке:

`corbaloc::хост_репозитария:порт_репозитария/имя_необходимого_сервиса`

Клиент направляет заявку используя этот корбалок. Заявка автоматически перенаправляется объекту-сервису.

Доступ к репозитарию проводится с помощью ссылки:

`corbaloc::хост_репозитария:порт_репозитария/Repository`

Внимание! Не используйте имя "Repository" при регистрации своих сервисов. Это имя используется для дуступа к репозитарию. При попытке зарегистрировать сервис с таким именем будет возвращено исключение PIMR::RepositoryAdmin::NotAviable.

В процессе работы репозитарий выдаёт сообщения в лог-файл. По определению это `./pimr.log`, но расположение этого файла может быть изменено с помощью аргумента командной строки `--configfile лог_файл`. Для того чтобы изменить степень деталиазции сообщений в файле PIMRPostConfig.h установите значения соответствующих переменных:

```
#define LOG_DEBUG_ENABLE   false
#define LOG_INFO_ENABLE    true
#define LOG_WARNING_ENABLE true
#define LOG_ERROR_ENABLE   true
#define LOG_FATAL_ENABLE   true
```

и перекомпилируйте репозитарий.

6 История документа

15.08.2002 - Создана предварительная версия.