Symbian OS v9.x 临时服务器模板

Hamish Willee, Andrew Thoelke, Adrian Taylor

由 Symbian Developer Network 出版

版本 1.0 – 2007 年 8 月

1 绪论

Symbian OS 大量使用了客户-服务器模型。临时服务器（transient server）是仅当需要是才存在的服务器：它在首次有客户端试图连接时启动，在最后一个客户端会话关闭一段时间后关闭。

用于启动和关闭服务器的代码是样板化的。本文及相关的代码提供了一个用于 Symbian OS v9 临时服务器的模板，它在不同于客户端的进程中启动服务器；注意，本文不介绍与客户端运行在同一进程中的服务器启动代码。

很多 Symbian OS 服务器都是临时的——减少占用内存比缩短服务器启动时间更重要，尤其在服务器提供的服务并非经常需要时。

临时服务器在 Symbian OS v9.x 中非常重要，Symbian OS v9.x 中的平台安全性意味着有时需要将访问权限赋予进程边界外的服务，例如，实现功能需求比客户端高的服务时。

2 服务器的功能

服务器由两部分组成：

t-client.dll：这是一个客户端 DLL（包含客户端 API）；

t-server.exe：服务器可执行文件。

客户端 API（t-client.h 中定义的 RMySession）提供了用于完成如下任务的方法：连接到服务器、同步发送消息、异步接收消息及取消未完成的异步接收消息请求。

```
class RMySession : public RSessionBase
    {
public:
    IMPORT_C TInt Connect();
    IMPORT_C TInt Send(const TDesC& aMessage);
    IMPORT_C void Receive(TRequestStatus& aStatus,TDes& aMessage);
    IMPORT_C void CancelReceive();
    };
```

从临时服务器的角度看，最有趣的方法是 Connect( )。这个方法首先尝试连接到服务器并创建一个会话。如果失败（如由于服务器不存在），这个函数将尝试启动服务器，然后再创建会话。

```
EXPORT_C TInt RMySession::Connect()
//
// Connect to the server, attempting to start it if necessary
//
    {
    TInt retry=2;
    for (;;)
        {
        TInt r=CreateSession(KMyServerName,TVersion(0,0,0),1);
        if (r!=KErrNotFound && r!=KErrServerTerminated)
            return r;
        if (--retry==0)
            return r;
        r=StartServer();
        if (r!=KErrNone && r!=KErrAlreadyExists)
            return r;
        }
    }
```

在服务器端，将调用 CMyServer::NewSessionL( )以创建服务器端会话对象。在这个例子中，这是其全部功能；而在真实的服务器中，通常需要检查服务器的版本与客户端期望的版本是否一致，还需要客户端是否具备使用该服务器所需的功能。

服务器维护一个会话计数。结束会话时，将调用 CMyServre::DropSession( ) 函数将该计数减 1，当计数减为 0 时，将启动关闭定时器。如果在此期间创建了另一个会话，关闭定时器将重置；否则，当定时器到期后将关闭服务器。

3 模板代码包含的内容

该模板由一个客户-服务器接口和一个控制台测试可执行文件组成，后者用于测试运行情况。要编译测试代码和示例代码，可进入命令行窗口，切换到目录\Transient\，然后执行常规编译调用命令：

```
bldmake bldfiles
abld build [gcce] | [armv5]
```

3.1.1 3.1 服务器文件

服务器文件存储在目录\Transient\Server\中，下表描述了每个服务器文件。

目录	文件	描述
src\	client.cpp	客户端 DLL 的源代码。
	server.cpp	服务器的源代码。
inc\	server.h	服务器定义。
	clientserver.h	客户端与服务器共享的定义。
	t-client.h	客户端 API，被导出到/epoc32/include/。
group\	t- server.mmp	服务器 exe 的项目文件。
	t-client.mmp	客户端 DLL 的项目文件。
sis\	server _gcce.pkg	S60 Server PKG 文件——导入 GCCE 二进制代码。
	server _armv5.pkg	S60 ServerPKG 文件——导入 ARMv5 二进制代码。

3.1.2 3.2 测试控制台文件

测试代码存储在目录\Transient\Test_ConsoleExe\中。测试代码是控制台可执行文件 T-Test.exe，它加载 t-test.dll，后者调用服务器的每个客户端 API。

下表描述了测试代码中的每个文件。

目录	文件	描述
src\	test.cpp	测试 exe 的源代码。
	testclient.cpp	测试 dll 的源代码。
inc\	plugin.h	测试 DLL 的定义。
	testclient.h	测试 DLL 的定义。
group\	t-test.mmp	测试 exe 的项目文件。
	t-testc.mmp	测试 DLL 的项目文件。
sis\	TestConsoleExe_gcce.pkg	S60 测试 PKG 文件——导入 GCCE 二进制代码。
	TestConsoleExe_armv5.pkg	S60 测试 PKG 文件——导入 ARMv5 二进制代码。

3.1.3 3.3 在设备上安装

子目录\sis\中的 PKG 文件用于编译服务器 exe 和测试 exe 的安装文件，服务器 exe 和测试 exe 可在 S60 3rd Edition 和 UIQ3 设备上运行。

PKG 文件指定相对路径，因此需要像FAQ-1113: How do I avoid using absolute paths in my package file?描述的那样使用 makesis，如 makesis -d%EPOCROOT% server_gcce.pkg。

可能还需要对文件进行签名，这取决于使用的是什么设备。

4 测试代码的作用

测试代码进行三项主要测试：

4.1.1 验证接口

它测试正常使用接口的情况，包括在服务器尚未启动时连接到服务器、发送和接收消息、创建另一个会话（服务器已启动）。最后测试当两个会话都关闭时临时服务器是否终止。

4.1.2 同时启动

该测试验证当两个会话同时创建时启动代码是否能够成功。

4.1.3 关闭时启动

该测试验证服务器已关闭或正在关闭时启动代码能够正常运行，例如，试图在释放资源后重启已经不运行的进程或服务器。

5 如何运行测试代码

控制台 exe 不是 GUI 应用程序，因此不会在 S60 或 UIQ 的 UI shell 中显示出来。但可以使用如下代码在另一个（可见的）应用程序中运行它：

```
RProcess testexe;
_LIT(KTestCodeFileName,"\\sys\\bin\\t-test.exe");
_LIT(KNullDesc,"");
TInt err= testexe.Create(KTestCodeFileName, KNullDesc);
if (err==KErrNone)
    {
    TRequestStatus stat;
    testexe.Rendezvous(stat);
    if (stat!=KRequestPending)
        testexe.Kill(0); // abort startup
    else
        testexe.Resume(); // logon OK - start the server
    User::WaitForRequest(stat);// wait for start or death
    testexe.Close();
    }
```

这段测试代码在 Nokia E61 手机中能够成功运行。使用 DevCert 对 SIS 文件进行了签名，它包含一个基本功能。没有在 UIQ3 设备中运行这段代码，但应该能够安装并正常运行。

6 如何扩展该模板

6.1.1 6.1 平台安全性

当前，客户端 DLL 和服务器 EXE 都没有功能。根据该模板创建自己的服务器时，需要赋予该 EXE 使用受保护的 API 所需的全部功能；需要赋予客户端 DLL 所有可能客户的所有功能。

如果还需要控制对服务器的访问，则应使用服务器基类CPolicyServre，而不是CServer2。这样，实现安全服务器或确保已有服务器的安全将比较简单。

也可直接使用 CServer2 来确保 API 的安全，例如，下面的代码片段演示了如何使用 _LIT_SECURITY_POLICY_S0 检查客户端的 SID：

```
// Check that SID matches required application.
// Use security policy check so behaviour obeys global security policy.
 static _LIT_SECURITY_POLICY_S0(mySidPolicy, KRequiredSecureId);
r = mySidPolicy().CheckPolicy(aMessage);
if (!r)
    {
    User::Leave(KErrPermissionDenied);
    }
```

当前，服务器名称没有受到保护，这意味着其他可通过宣称自己是 t-server，从而将其伪装成 t-server 服务器。如果客户端可能传递敏感信息给服务器，客户端必须能够确保加载的服务器是正确的。为此，可给服务器指定一个受保护的名称——以!打头的名称。如果这样做，将需要请求 ProtServ 功能。有关服务器名称，后面还有更详细的解释。

6.1.2 6.2 UID

这个示例使用示例范围（example range）中的 UID/SID；对于商业应用程序，需要使用 www.symbiansigned.com分配的 UID。

6.1.3 6.3 客户-服务器的问题

下面介绍将该模板作为服务器设计的一部分时可能修改的几个地方。

6.1.3.1 6.3.1 客户端 API

客户端 API、进程间通信（Inter-Process Communication，IPC）和服务器端实现取决于服务器需要完成的任务。很多服务器采用与这个示例中的 API 类似的方式在客户端和服务器之间复制数据。

读者应根据具体情况对文件、类和方法重命名。

6.1.3.2 6.3.2 消息槽数量

RMySession::Connect() 调用 RSessionBase::CreateSession() 来指定消息槽数量 KServerDefaultMessageSlots。消息槽数量是服务器设计中的一个重要参数，因为它定义了客户端能够同时向服务器发出的未处理请求数量。在这个例子中，只有一个异步函数，因此只能有一个未处理的请求，所以只需要一个消息槽。

最大的消息槽数为 255。如果 aAsyncMessageSlots==-1，则表明会话应使用全局可用消息池中的消息。

6.1.3.3 6.3.3 重试次数

RMySession::Connect( )首先尝试建立到服务器的会话，如果失败则尝试重启服务器（服务器可能不存在，也可能已终止运行）。这个过程将一直重复，直到会话创建成功、服务器因意想不到或无法恢复的错误而启动失败或达到指定重试的次数（KServerRetryCount）。

从理论上说，KServerRetryCount（client.cpp）可以无限大；当前使用的值 2 看起来能够消除 99%与启动/关闭相关的问题，但如果出现特殊的情况，可能需要修改这个值。

6.1.3.4 6.3.4 服务器名称

需要编辑 clientserver.h 中定义的服务器名称（KMyServerName），可能还要使服务器名称成为受保护的，这在前面介绍过。

可根据设计需求（如 IPC 函数枚举），修改 clientserver.h 中的其他所有值。

7 如何报告错误或提出有关模板代码的问题

要提出有关本模板代码的问题，最佳的地方是 Symbian 的feedback.api新闻组。

返回到 Developer Library

要阅读 Symbian Developer Network 上最新的文章吗？