 在Oracle 8i中,往往会出现要在存储过程中运行操作系统命令的情况.一般来说,利用Oracle Enterprise Manager设定作业时可以达到这个目的.但是由于OEM在设定作业缺乏灵活性,设定的作业的参数是固定的.在实际应用当中往往需要在SQL语句当中运行需要随时运行操作系统命令. Oracle 8i没有直接运行OS命令的语句,我们可以利用DBMS_PIPE程序包实现这一要求. DBMS_PIPE通过创建管道,可以让至少两个进程进行通信.Oracle的管道与操作系统的管道在概念上有相同的地方,但是在实现机制不同. 下面介绍实现具体步骤: 1、创建一个程序包,姑且起名叫DAEMON,SQL语句如下: /*创建daemon程序包*/ CREATE OR REPLACE PACKAGE BODY daemon AS /*execute_system是实现运行os命令的函数*/ FUNCTION execute_system(command VARCHAR2, timeout NUMBER DEFAULT 10) RETURN NUMBER IS status NUMBER. result VARCHAR2(20). command_code NUMBER. pipe_name VARCHAR2(30). BEGIN pipe_name := DBMS_PIPE.UNIQUE_SESSION_NAME. DBMS_PIPE.PACK_MESSAGE(SYSTEM). DBMS_PIPE.PACK_MESSAGE(pipe_name). DBMS_PIPE.PACK_MESSAGE(command). /*向daemon管道发送表示命令的字符*/ status := DBMS_PIPE.SEND_MESSAGE(daemon, timeout). IF status 0 THEN RAISE_APPLICATION_ERROR(-20010, Execute_system:

Error while sending. Status = || status). END IF. status := DBMS_PIPE.RECEIVE_MESSAGE(pipe_name, timeout). IF status 0 THEN RAISE_APPLICATION_ERROR(-20011, Execute_system: Error while receiving. Status = || status). END IF. /*获取返回结果*/ DBMS_PIPE.UNPACK_MESSAGE(result). IF result done THEN RAISE_APPLICATION_ERROR(-20012, Execute_system: Done not received.). END IF. DBMS_PIPE.UNPACK_MESSAGE(command_code). DBMS_OUTPUT.PUT_LINE(System command executed. result = || command_code). RETURN command_code. END execute_system. /*stop是让daemon停止*/ PROCEDURE stop(timeout NUMBER DEFAULT 10) IS status NUMBER. BEGIN DBMS_PIPE.PACK_MESSAGE(STOP). status := DBMS_PIPE.SEND_MESSAGE(daemon, timeout). IF status 0 THEN RAISE_APPLICATION_ERROR(-20030, stop: error while sending. status = || status). END IF. END stop. END daemon. 通过Sql*Plus运行以上语句,将为当前用户创建daemon程序包. 2、创建在OS上运行的守护进程,监听由上面的daemon程序包发来的要求执行os命令的语句.以下Pro*C的代码,必须由pro*c先进行预编译. #include #include EXEC SQL INCLUDE SQLCA. EXEC SQL BEGIN DECLARE SECTION. char *uid = "scott/tiger"./*在这个地方改为你自己访问的用户,密码,服务名*/ int status. VARCHAR command[20]. VARCHAR value[2000]. VARCHAR return_name[30]. EXEC SQL END DECLARE SECTION. void connect_error() { char msg_buffer[512]. int msg_length. int buffer_size = 512. EXEC SQL

WHENEVER SQLERROR CONTINUE. sqlglm(msg_buffer, amp.msg_length). printf("Daemon error while connecting:\n"). printf("%.*s\n", msg_length, msg_buffer). printf("Daemon quitting.\n"). exit(1). } void sql_error() { char msg_buffer[512]. int msg_length. int buffer_size = 512. EXEC SQL WHENEVER SQLERROR CONTINUE. sqlglm(msg_buffer, amp.msg_length). printf("Daemon error while executing:\n"). printf("%.*s\n", msg_length, msg_buffer). printf("Daemon continuing.\n"). } main() { EXEC SQL WHENEVER SQLERROR DO connect_error(). EXEC SQL CONNECT :uid. printf("Daemon connected.\n"). EXEC SQL WHENEVER SQLERROR DO sql_error(). printf("Daemon waiting...\n"). while (1) { EXEC SQL EXECUTE BEGIN /*接收deamon发来的字符*/ :status := DBMS_PIPE.RECEIVE_MESSAGE(daemon). IF :status = 0 THEN /*取出字符*/ DBMS_PIPE.UNPACK_MESSAGE(:command). END IF. END. END-EXEC. IF (status == 0) { command.arr[command.len] = \0. /*如果是stop,该进程就退出*/ IF (!strcmp((char *) command.arr, "STOP")) { printf("Daemon exiting.\n"). break. } ELSE IF (!strcmp((char *) command.arr, "SYSTEM")) { EXEC SQL EXECUTE BEGIN DBMS_PIPE.UNPACK_MESSAGE(:return_name). DBMS_PIPE.UNPACK_MESSAGE(:value). END. END-EXEC. value.arr[value.len] = \0. printf("Will execute system command %s\n", value.arr). /*运行os命令*/ status = system(value.arr). EXEC SQL EXECUTE BEGIN DBMS_PIPE.PACK_MESSAGE(done). DBMS_PIPE.PACK_MESSAGE(:status). :status :=

DBMS_PIPE.SEND_MESSAGE(:return_name). END.
END-EXEC. IF (status) { printf ("Daemon error while responding to system command."). printf(" status: %d\n", status). } } ELSE { printf ("Daemon error: invalid command %s received.\n", command.arr). } } ELSE { printf("Daemon error while waiting for signal."). printf(" status = %d\n", status). } } EXEC SQL COMMIT WORK RELEASE. exit(0). } 以上代码起名为daemon.pc,用proc预编译: proc iname=daemon.pc userid=用户名/密码@服务名 sqlcheck=semantics 得到daemon.c,在用c进行编译,注意在NT上要把orasql8.lib加上,否则编译通过,连接没法通过. 3、在服务器上运行daemon.exe 4、在sqlplus运行测试语句: SQL> variable rv number SQL> execute :rv := DAEMON.EXECUTE_SYSTEM(ls -la). PL/SQL 过程已成功完成。 SQL> execute :rv := DAEMON.EXECUTE_SYSTEM(dir). PL/SQL 过程已成功完成。 SQL>