In almost any type of database environment—from development to testing to production—database administrators rely heavily on automating tasks. Typical jobs that DBAs automate include the following:

- Shutdown and startup of databases and listeners
- Backups
- Validating the integrity of backups
- Checking for errors
- Removing old trace or log files
- Checking for errant processes
- Checking for abnormal conditions

Automating routine tasks allows DBAs to be much more effective and productive. Automated environments inherently run smoother and more efficiently than manually administered systems. DBA jobs that run automatically from scripts consistently execute the same set of commands each time and therefore are less prone to human error and mistakes. There are two scheduling utilities described in this chapter:

- Oracle Scheduler
- The Linux/Unix `cron` utility

This chapter begins with a section detailing the basic aspects of the Oracle Scheduler utility. This scheduler is available if you have an Oracle database installed. Oracle Scheduler can be used to schedule jobs in a wide variety of configurations.

Also contained in this chapter is a section detailing how to use the Linux/Unix `cron` scheduling tool. In Linux/Unix environments, DBAs often use the `cron` scheduling utility to automatically run jobs. The `cron` utility is ubiquitous and easy to implement and use. If you’re an Oracle DBA, you must be familiar with `cron` because sooner or later, you’ll find yourself in an environment that relies heavily on this tool to automate database jobs.

The last several sections in this chapter show you how to implement several real-world DBA jobs such as performance reporting, monitoring, and operating system file maintenance. You should be able to extend these scripts to meet the automation requirements of your environment.
Automating Jobs with Oracle Scheduler

Oracle Scheduler is a tool that provides you a way of automating the scheduling of jobs. Oracle Scheduler is implemented via the `DBMS_SCHEDULER` internal PL/SQL package. Oracle Scheduler provides you with a sophisticated set of features for scheduling jobs. The following sections of this chapter only cover the basics of using Oracle Scheduler to automate jobs with simple requirements.

**Tip** There are currently nearly 70 procedures and functions available within the `DBMS_SCHEDULER` package. See the Oracle Database PL/SQL Packages and Types Reference guide (available on OTN) for complete details.

Creating and Scheduling a Job

In Listing 21–1, a shell script is created that contains an RMAN backup command. This shell script is named `rmanback.bsh` and is located in the `/orahome/oracle/bin` directory. The shell script also assumes that there is a `/orahome/oracle/bin/log` directory available.

**Listing 21–1. A Shell Script Containing an RMAN Backup Command**

```bash
#!/bin/bash
# source oracle OS variables; see chapter 2 for an example of oraset script
. /var/opt/oracle/oraset  RMDB1
rman target / <<EOF
spool log to '/orahome/oracle/bin/log/rmanback.log'
backup database;
spool log off;
EOF
exit 0
```

In Listing 21–2, you use the `CREATE_JOB` procedure of the `DBMS_SCHEDULER` package to create a job. Run it as SYS (from the SQL*Plus).

**Listing 21–2. Using the CREATE_JOB Procedure**

```sql
BEGIN
DBMS_SCHEDULER.CREATE_JOB(
  job_name => 'RMAN_BACKUP',
  job_type => 'EXECUTABLE',
  job_action => '/orahome/oracle/bin/rmanback.bsh',
  repeat_interval => 'FREQ=DAILY;BYHOUR=14;BYMINUTE=11',
  start_date => to_date('21–OCT-10'),
  job_class => '"DEFAULT_JOB_CLASS"',
  auto_drop => FALSE,
  comments => 'RMAN backup job',
  enabled => TRUE);
END;
/```