Starting Processes in the Future

Scheduling Future Processes Using at:

1. Suppose you need to perform a task on a specific day sometime in the future. However, you know you will be away from the machine on that day. How will you perform the task?
2. You can use the at utility program to execute any non-interactive command at a specified time, as illustrated in the screenshot below:

```
c8:/tmp>at now + 2 days
warning: commands will be executed using /bin/sh

at> # now issue a command
at> mail < /var/log/messages student@localhost

at> # hit CTRL-D to quit
at> <EOT>
job 103 at Sun Jun  6 15:43:00 2021

c8:/tmp># check on scheduled job:
c8:/tmp>atq
103     Sun Jun  6 15:43:00 2021 a coop

c8:/tmp># remove the scheduled job
c8:/tmp>atrm 103

c8:/tmp>#check
c8:/tmp>atq
c8:/tmp>
```

Scheduling Future Processes Using at

cron:

1. **cron** is a time-based scheduling utility program.
2. It can launch routine background jobs at specific times and/or days on an on-going basis.
3. **cron** is driven by a configuration file called /etc/crontab (cron table), which contains the various shell commands that need to be run at the properly scheduled times.
4. Each line of a **crontab** file represents a job, and is composed of a so-called **CRON** expression, followed by a shell command to execute.
5. Typing **crontab -e** will open the crontab editor to edit existing jobs or to create new jobs. Each line of the crontab file will contain 6 fields:

Field	Description	Values
MIN	Minutes	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31
MON	Month field	1-12
DOW	Day Of Week	0-6 (0 = Sunday)
CMD	Command	Any command to be executed

Examples:

1. The entry * * * * * /usr/local/bin/execute/this/script.sh will schedule a job to execute **script.sh** every minute of every hour of every day of the month, and every month and every day in the week.
2. The entry 30 08 10 06 * /home/sysadmin/full-backup will schedule a full-backup at 8.30 a.m., 10-June, irrespective of the day of the week.

sleep:

1. Sometimes, a command or job must be delayed or suspended.
2. Suppose, for example, an application has read and processed the contents of a data file and then needs to save a report on a backup system.
3. If the backup system is currently busy or not available, the application can be made to **sleep** (**wait**) until it can complete its work.
4. **sleep** suspends execution for at least the specified period of time, which can be given as the **number of seconds** (the default), **minutes**, **hours**, or **days**
5. After that time has passed (or an interrupting signal has been received), execution will resume.

Syntax: sleep NUMBER[SUFFIX]...

where SUFFIX may be:

- **s** for seconds (the default)
- **m** for minutes
- **h** for hours
- **d** for days.

6. **sleep** and at are quite different; sleep delays execution for a specific period, while at starts execution at a later time.