An example of where this type of environment tracing can come in handy is in tracing problems with the apt-get package manager. If you use an HTTP proxy to connect to the Internet, you may need to set environment variables `http_proxy=host:port`. Let's say you forgot to set the environment variables in a script, the `apt-get` command will not select the proxy and hence, it returns an error. Then, you can actually look at an environment variable and track the issue.

We may need some applications to be run automatically with scheduling tools, such as `cron`. But it might be dependent on some environment variables. Suppose, if we want to open a GUI-windowed application at a given time. We schedule it using `crontab` at a specified time, but this will not work:

```
00 10 * * * /usr/bin/windowapp
```

It is because a windowed application always depends on the `DISPLAY` environment variable. To figure out the environment variables needed, run `windowapp` manually, and then run `ps -C windowapp -eo cmd e`.

Find out the environment variables and prefix them before a command name appears in `crontab` as follows:

```
00 10 * * * DISPLAY=:0 /usr/bin/windowapp
```

Here, `DISPLAY=:0` was obtained from the `ps` output.

About which, whereis, file, whatis, and load average

There are a few commands which are useful for exploring other commands and such. Let's discuss them:

- `which`: The `which` command is used to find the location of a command. We type commands in the terminal without knowing the location where the executable file is stored.
- When we type a command, the terminal looks for the command in a set of locations and executes the executable file if found at the location. These sets of locations are specified using an environment variable `PATH`. For example:

  ```
  $ echo $PATH
  /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
  ```

- We can export `PATH` and can add our own locations to be searched when command names are typed. For example, to add `/home/slynux/bin` to `PATH`, use the following command:

  ```
  $ export PATH=$PATH:/home/slynux/bin
  # /home/slynux/bin is added to PATH
  ```

- The `which` command outputs the location of the command given as the argument. For example:

```
$ which ls
/bin/ls
```

- `whereis`: `whereis` is similar to the which command. But it not only returns the path of the command, it will also print the location of the man page, if available, and also the path of the source code for the command if available. For example:

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

- `file`: The `file` command is an interesting and frequently-used command. It is used for determining the file type:

```
$ file FILENAME
```

- This will print the details of the file regarding its file type.
- An example is as follows:

```
$ file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.15,
stripped
```

- `whatis`: The `whatis` command outputs a one-line description of the command given as the argument. It parses information from the man page. For example:

```
$ whatis ls
ls (1)               - list directory contents
```

apropos

Sometimes, we need to search if some command related to a word exists. Then, we can search the man pages for strings in the command. For this we can use:

```
apropos COMMAND
```

- `load average`: `load average` is an important parameter for the total load on the running system. It specifies the average of the total number of runnable processes on the system. It is specified by three values. The first value indicates the average in one minute, the second indicates the average in five minutes, and the third indicates the average in 15 minutes.