CHAPTER 10

■ ■ ■

Process API

QUESTIONS AND EXERCISES

1. What is the Process API?

Answer:

The Process API consists of classes and interfaces that let you work with native processes in Java programs. Using the API, you can:

- Create new native processes from Java code
- Get process handles for native processes whether they were created by Java code or by other means
- Destroy running native processes
- Query processes for liveness and their other attributes
- Get the list of child processes and the parent process of a process
- Get the process ID (PID) of native processes
- Get the input, output, and error streams of newly created processes
- Wait for a process to terminate
- Execute a task when a process terminates

The Process API is small. It consists of the classes and interfaces such as `Runtime`, `ProcessBuilder`, `ProcessBuilder.Redirect`, `Process`, `ProcessHandle`, and `ProcessHandle.Info`.

2. What does an instance of the `Runtime` class represent?

Answer:

Every Java application has an instance of the `Runtime` class that lets you query and interact with the runtime environment in which the current Java application is running.

3. How do you get an instance of the `Runtime` class?

Answer:

The `Runtime` class is singleton. You can get its sole instance using the `getRuntime()` static method of this class:

```
// Get the instance of the Runtime
Runtime runtime = Runtime.getRuntime();
```

4. How do you use the `ProcessBuilder` class? What method of this class is used to start a new native process?

Answer:

You create an instance of the `ProcessBuilder` class using one of its constructors. You set the command to start the native process using its `command()` method. You can also set the command in its constructor. Call other methods of the `ProcessBuilder` class to set the attributes for the command.

Once you have set the command and its attributes in a `ProcessBuilder` instance, you can call its `start()` method to start a native process. The following is an example of starting the `java` process with an option `"--version"`:

```
Process p = new ProcessBuilder()
                    .command("java", "--version")
                    .inheritIO()
                    .start();
```

Each time you call the `start()` method, a new native process is created using the already stored properties in the `ProcessBuilder` instance.

5. What does an instance of the `Process` class represent?

An instance of the `Process` class represents a native process started by the JVM using the `Runtime.exec()` method or the `ProcessBuilder.start()` method.

6. What does an instance of the `ProcessHandle` interface represent? How do you obtain a `ProcessHandle` from a `Process`?

Answer:

An instance of the `ProcessHandle` interface represents a native process whether the process was started by the JVM or by any other means.

You need to call the `toHandle()` method of the `Process` to get its `ProcessHandle`:

```
ProcessBuilder pb = /* Get a process builder */;

// Start a new process
Process newProcess = pb.start();

// Get the process handle
ProcessHandle handle = newProcess.toHandle();
```

7. How do you get the handle of the current process representing the running Java program?

Answer:

The `current()` static method of the `ProcessHandle` interface returns the handle of the current process.

```
// Get the handle of the current process
ProcessHandle current = ProcessHandle.current();
```

8. What does an instance of the `ProcessHandle.Info` interface represent?

Answer:

An instance of the `ProcessHandle.Info` represents a detailed snapshot information about a native process.

9. What is the default standard I/O of the new process created by the `start()` method of the `ProcessBuilder` class?

Answer:

The default standard I/O of the new process created by the `start()` method of the `ProcessBuilder` class is piped to the parent process. If you want to access the output, you need read from the appropriate pipe. When the standard I/O of the new process is piped to the parent process, you can use the following methods of the `Process` to get the I/O streams of the new process:

- `OutputStream getOutputStream()`
- `InputStream getInputStream()`
- `InputStream getErrorStream()`

10. Can you terminate the current Java program using the Process API?

Answer:

No. You cannot terminate the current Java program using the Process API. Attempting to do so results in in an `IllegalStateException`. You need to use `System.exit()` method to terminate the current Java program.