

Building graphic-rich and better performing
native applications



Pro
Android C++
with the **NDK**

Onur Cinar



Apress®

Pro Android C++ with the NDK



Onur Cinar

Apress®

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4302-4827-9

ISBN-13 (electronic): 978-1-4302-4828-6

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

The images of the Android Robot (01 / Android Robot) are reproduced from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License. Android and all Android and Google-based marks are trademarks or registered trademarks of Google, Inc., in the U.S. and other countries. Apress Media, L.L.C. is not affiliated with Google, Inc., and this book was written without endorsement from Google, Inc.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editor: Steve Anglin

Technical Reviewer: Grant Allen

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Brigid Duffy

Copy Editor: Mary Behr

Composer: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Dedicated to my son Deren, my wife Sema, and my parents, Zekiye and Dogan, for their love, continuous support, and always encouraging me to pursue my dreams.

I could not have done this without all of you.

—Onur Cinar

Contents at a Glance

About the Author

About the Technical Reviewer

Preface

■ **Chapter 1: Getting Started with C++ on Android**

■ **Chapter 2: Exploring the Android NDK**

■ **Chapter 3: Communicating with Native Code using JNI**

■ **Chapter 4: Auto-Generate JNI Code Using SWIG**

■ **Chapter 5: Logging, Debugging, and Troubleshooting**

■ **Chapter 6: Bionic API Primer**

■ **Chapter 7: Native Threads**

■ **Chapter 8: POSIX Socket API: Connection-Oriented Communication**

■ **Chapter 9: POSIX Socket API: Connectionless Communication**

■ **Chapter 10: POSIX Socket API: Local Communication**

■ **Chapter 11: C++ Support**

■ **Chapter 12: Native Graphics API**

■ **Chapter 13: Native Sound API**

■ **Chapter 14: Profiling and NEON Optimization**

Index

Contents

About the Author

About the Technical Reviewer

Preface

■ **Chapter 1: Getting Started with C++ on Android**

Microsoft Windows

Downloading and Installing the Java Development Kit on Windows

Downloading and Installing the Apache ANT on Windows

Downloading and Installing the Android SDK on Windows

Downloading and Installing the Cygwin on Windows

Downloading and Installing the Android NDK on Windows

Downloading and Installing the Eclipse on Windows

Apple Mac OS X

Installing Xcode on Mac

Validating the Java Development Kit on Mac

Validating the Apache ANT on Mac

Validating the GNU Make

Downloading and Installing the Android SDK on Mac

Downloading and Installing the Android NDK on Mac

Downloading and Installing the Eclipse on Mac

Ubuntu Linux

Checking the GNU C Library Version

Enabling the 32-Bit Support on 64-Bit Systems

Downloading and Installing the Java Development Kit on Linux

Downloading and Installing the Apache ANT on Linux

Downloading and Installing the GNU Make on Linux

Downloading and Installing the Android SDK on Linux

Downloading and Installing the Android NDK on Linux

Downloading and Installing the Eclipse on Linux

Downloading and Installing the ADT

Installing the Android Platform Packages

Configuring the Emulator

Summary

■ Chapter 2: Exploring the Android NDK

Components Provided with the Android NDK

Structure of the Android NDK

Starting with an Example

- Specifying the Android NDK Location

- Importing the Sample Project

- Adding Native Support to Project

- Running the Project

- Building from the Command Line

- Examining the Structure of an Android NDK Project

Build System

- Android.mk

- Application.mk

Using the NDK-Build Script

Troubleshooting Build System Problems

Summary

■ Chapter 3: Communicating with Native Code using JNI

What is JNI?

Starting with an Example

- Declaration of Native Methods

- Loading the Shared Libraries

- Implementing the Native Methods

Data Types

- Primitive Types

- Reference Types

Operations on Reference Types

- String Operations

- Array Operations

- NIO Operations

- Accessing Fields

- Calling Methods

- Field and Method Descriptors

Exception Handling

Catching Exceptions

Throwing Exceptions

Local and Global References

Local References

Global References

Weak Global References

Threading

Synchronization

Native Threads

Summary

■ Chapter 4: Auto-Generate JNI Code Using SWIG

What is SWIG?

Installation

Installing on Windows

Installing on Mac OS X

Installing on Ubuntu Linux

Experimenting with SWIG Through an Example

Interface File

Invoking SWIG from Command Line

Integrating SWIG into Android Build Process

Updating the Activity

Executing the Application

Exploring Generated Code

Wrapping C Code

Global Variables

Constants

Read-Only Variables

Enumerations

Structures

Pointers

Wrapping C++ Code

Pointers, References, and Values

Default Arguments

Overloaded Functions

Exception Handling

Memory Management

Calling Java from Native Code

Asynchronous Communication

Enabling Directors

Enabling RTTI

Overriding the Callback Method

Updating the HelloJni Activity

Summary

■ Chapter 5: Logging, Debugging, and Troubleshooting

Logging

Framework

Native Logging APIs

Controlled Logging

Console Logging

Debugging

Prerequisites

Debug Session Setup

Setting up the Example for Debugging

Starting the Debugger

Troubleshooting

Stack Trace Analysis

Extended Checking of JNI

Memory Issues

Strace

Summary

■ Chapter 6: Bionic API Primer

Reviewing Standard Libraries

Yet Another C Library?

Binary Compatibility

What is Provided?

What is Missing?

Memory Management

Memory Allocation

Dynamic Memory Management for C

Dynamic Memory Management for C++

Standard File I/O

Standard Streams

Using the Stream I/O

Opening Streams

Writing to Streams

Reading from Streams

Seeking Position

Checking Errors

Closing Streams

Interacting with Processes

Executing a Shell Command

Communicating with the Child Process

System Configuration

Getting a System Property Value by Name

Getting a System Property by Name

Users and Groups

Getting the Application User and Group IDs

Getting the Application User Name

Inter-Process Communication

Summary

■ Chapter 7: Native Threads

Creating the Threads Example Project

Creating the Android Project

Adding the Native Support

Declaring the String Resources

Creating a Simple User Interface

Implementing the Main Activity

Generating the C/C++ Header File

Implementing the Native Functions

Updating the Android.mk Build Script

Java Threads

Updating the Example Application to use Java Threads

Executing the Java Threads Example

Pros and Cons of using Java Threads for Native Code

POSIX Threads

Using POSIX Threads in Native Code

Creating Threads using pthread_create

Updating the Example Application to use POSIX Threads

Executing the POSIX Threads Example

Return Result from POSIX Threads

Updating the Native Code to Use pthread_join

Synchronizing POSIX Threads

Synchronizing POSIX Threads using Mutexes

Synchronizing POSIX Threads Using Semaphores

Priority and Scheduling Strategy for POSIX Threads

POSIX Thread Scheduling Strategy

POSIX Thread Priority

Summary

■ Chapter 8: POSIX Socket API: Connection-Oriented Communication

Echo Socket Example Application

Echo Android Application Project

Abstract Echo Activity

Echo Application String Resources

Native Echo Module

Connection-Oriented Communication through TCP Sockets

Echo Server Activity Layout

Echo Server Activity

Implementing the Native TCP Server

Echo Client Activity Layout

Echo Client Activity

Implementing the Native TCP Client

Updating the Android Manifest

Running the TCP Sockets Example

Summary

■ **Chapter 9: POSIX Socket API: Connectionless Communication**

Adding Native UDP Server Method to Echo Server Activity

Implementing the Native UDP Server

New UDP Socket: socket

Receive Datagram from Socket: recvfrom

Send Datagram to Socket: sendto

Native UDP Server Method

Adding Native UDP Client Method to Echo Client Activity

Implementing the Native UDP Client

Native UDP Client Method

Running the UDP Sockets Example

Interconnecting the Emulators for UDP

Starting the Echo UDP Client

Summary

■ **Chapter 10: POSIX Socket API: Local Communication**

Echo Local Activity Layout

Echo Local Activity

Implementing the Native Local Socket Server

New Local Socket: socket

Bind Local Socket to Name: bind

Accept on Local Socket: accept

Native Local Socket Server

Adding Local Echo Activity to Manifest

Running the Local Sockets Example

Asynchronous I/O

Summary

■ **Chapter 11: C++ Support**

Supported C++ Runtimes

GAbi++ C++ Runtime

STLport C++ Runtime

GNU STL C++ Runtime

Specifying the C++ Runtime

Static vs. Shared Runtimes

C++ Exception Support

C++ RTTI Support

C++ Standard Library Primer

- Containers

- Iterators

- Algorithms

Thread Safety of C++ Runtime

C++ Runtime Debug Mode

- GNU STL Debug Mode

- STLport Debug Mode

Summary

■ Chapter 12: Native Graphics API

Availability of Native Graphics API

Creating an AVI Video Player

- Make AVILib a NDK Import Module

- Create the AVI Player Android Application

- Create the AVI Player Main Activity

- Creating the Abstract Player Activity

Rendering using JNI Graphics API

- Enabling the JNI Graphics API

- Using the JNI Graphics API

- Updating AVI Player with Bitmap Renderer

- Running the AVI Player with Bitmap Renderer

Rendering Using OpenGL ES

- Using the OpenGL ES API

- Enabling OpenGL ES 1.x API

- Enabling OpenGL ES 2.0 API

- Updating AVI Player with OpenGL ES Renderer

Rendering Using Native Window API

- Enabling the Native Window API

- Using the Native Window API

Summary

■ Chapter 13: Native Sound API

Using the OpenSL ES API

- Compatibility with the OpenSL ES Standard

- Audio Permissions

Creating the WAVE Audio Player

- Make WAVELib a NDK Import Module

- Create the WAVE Player Android Application

- Creating the WAVE Player Main Activity

- Implementing WAVE Audio Playback

Running the WAVE Audio Player

Summary

■ Chapter 14: Profiling and NEON Optimization

GNU Profiler for Measuring Performance

- Installing the Android NDK Profiler

- Enabling the Android NDK Profiler

- Analyzing gmon.out using GNU Profiler

Optimization using ARM NEON Intrinsics

- Overview of ARM NEON Technology

- Adding a Brightness Filter to AVI Player

- Enabling the Android NDK Profiler for AVI Player

- Profiling the AVI Player

- Optimizing the Brightness Filter using NEON Intrinsics

Automatic Vectorization

- Enabling Automatic Vectorization

- Troubleshooting Automatic Vectorization

Summary

Index

About the Author



Onur Cinar has over 17 years of experience in design, development, and management of large scale complex software projects, primarily in mobile and telecommunication space. His expertise spans VoIP, video communication, mobile applications, grid computing, and networking technologies on diverse platforms. He has been actively working with Android platform since its beginning. He is the author of the book *Android Apps with Eclipse* from Apress. He has a Bachelor of Science degree in Computer Science from Drexel University in Philadelphia, PA, United States. He is currently working at Skype division of Microsoft as the Sr. Product Engineering Manager for the Skype client on Android platform.

About the Technical Reviewer



Grant Allen has worked in the IT field for over 20 years as a CTO, enterprise architect, and database architect. Grant's roles have covered private enterprise, academia, and the government sector around the world, specializing in global-scale systems design, development, and performance. He is a frequent speaker at industry and academic conferences, on topics ranging from data mining to compliance, and technologies such as databases (DB2, Oracle, SQL Server, and MySQL), content management, collaboration, disruptive innovation, and mobile ecosystems like Android.

His first Android application was a task list to remind him to finish all his other unfinished Android projects.

Grant works for Google, and in his spare time is completing a PhD on building innovative high-technology environments.

Grant is the author of *Beginning DB2: From Novice to Professional* (Apress, 2008), and lead author of *Oracle SQL Recipes: A Problem-Solution Approach* (Apress, 2010) and *The Definitive Guide to SQLite, 2nd Edition* (Apress, 2010).

Preface

Android is one of the major players in mobile phone market, and continuously growing its market share. It is the first complete, open, and free mobile platform that is enabling endless opportunities for mobile application developers.

Although the official programming language for the Android platform is Java, the application developers are not limited to using only the Java technology.

Android allows application developers to implement parts of their application using native-code languages such as C and C++ through the Android Native Development Kit (NDK). In this book, you will learn how to use the Android NDK to implement performance-critical portions of your Android applications using native-code languages.

Android C++ with the NDK provides a detailed overview of native application development, available native APIs, the troubleshooting techniques, including the step by step instructions and screenshots to help Android developers to quickly get up to speed on developing native application.

What You Will Learn

This book includes the following:

- Installing the Android native development environment on major operating systems.
- Using the Eclipse IDE to develop native code.
- Connecting native code to Java world using Java Native Interface (JNI).
- Auto-generating the JNI code using SWIG.
- Developing multithreaded native apps using the POSIX and Java threads.
- Developing networking native apps using POSIX sockets.
- Debug native code through logging, GDB, and Eclipse Debugger.
- Analyzing memory issues through Valgrind.
- Measuring application performance through GProf.
- Optimizing native code through SIMD/NEON.

Downloading the Code

The source code for this book is available to readers at www.apress.com.

Contacting the Author

Readers can contact the author through author's Android C++ with the NDK site at <http://www.zdo.com/android-c++-with-the-ndk> to ask questions.

Getting Started with C++ on Android

Needless to say, exploring and practicing are the best methods for learning. Having a fully functional development environment ready at the very beginning of this book will enable you to explore and experiment with the material while working through the chapters. The Android C++ development environment is mainly formed by the following components:

- Android Software Development Kit (SDK)
- Android Native Development Kit (NDK)
- Android Development Tools (ADT) Plug-In for Eclipse
- Java Development Kit (JDK)
- Apache ANT Build System
- GNU Make Build System
- Eclipse IDE

This chapter will provide step-by-step instructions for setting up the proper Android C++ development environment. Android development tools are provided for the major operating systems:

- Microsoft Windows
- Apple Mac OS X
- Linux

Since the requirements and the installation procedure vary depending on the operating system, the following sections will walk you through the steps for setting up the Android C++ development environment based on the operating system. You can skip over the ones that don't apply to you.

Microsoft Windows

Android development tools require Windows XP (32-bit only), Vista, or Windows 7. In this section, you will be downloading and installing the following components :

- Java JDK 6
- Apache ANT Build System
- Android SDK
- Cygwin
- Android NDK
- Eclipse IDE

Downloading and Installing the Java Development Kit on Windows

Android development tools require Java Development Kit (JDK) version 6 in order to run. Java Runtime Edition (JRE) itself is not sufficient. Java JDK 6 needs to be installed prior installing the Android development tools.

Note Android development tools only support Java compiler compliance level 5 or 6. Although the later versions of JDK can be configured to comply with those levels, using JDK 6 is much simpler and less prone to errors.

Multiple JDK flavors are supported by Android development tools, such as IBM JDK, Open JDK, and Oracle JDK (formerly known as Sun JDK). In this book, it is assumed that Oracle JDK will be used since it supports a broader range of platforms.

In order to download Oracle JDK, navigate to www.oracle.com/technetwork/java/javase/downloads/index.html and follow these steps:

1. Click the JDK 6 download button, as shown in [Figure 1-1](#). At the time of this writing the latest version of Oracle JDK 6 is Update 33.



Figure 1-1 . Oracle JDK 6 Download button

2. Clicking the Oracle JDK 6 Download button takes you to a page listing the Oracle JDK 6 installation packages for supported platforms.
3. Check "Accept License Agreement" and download the installation package for Windows x86, as shown in [Figure 1-2](#).

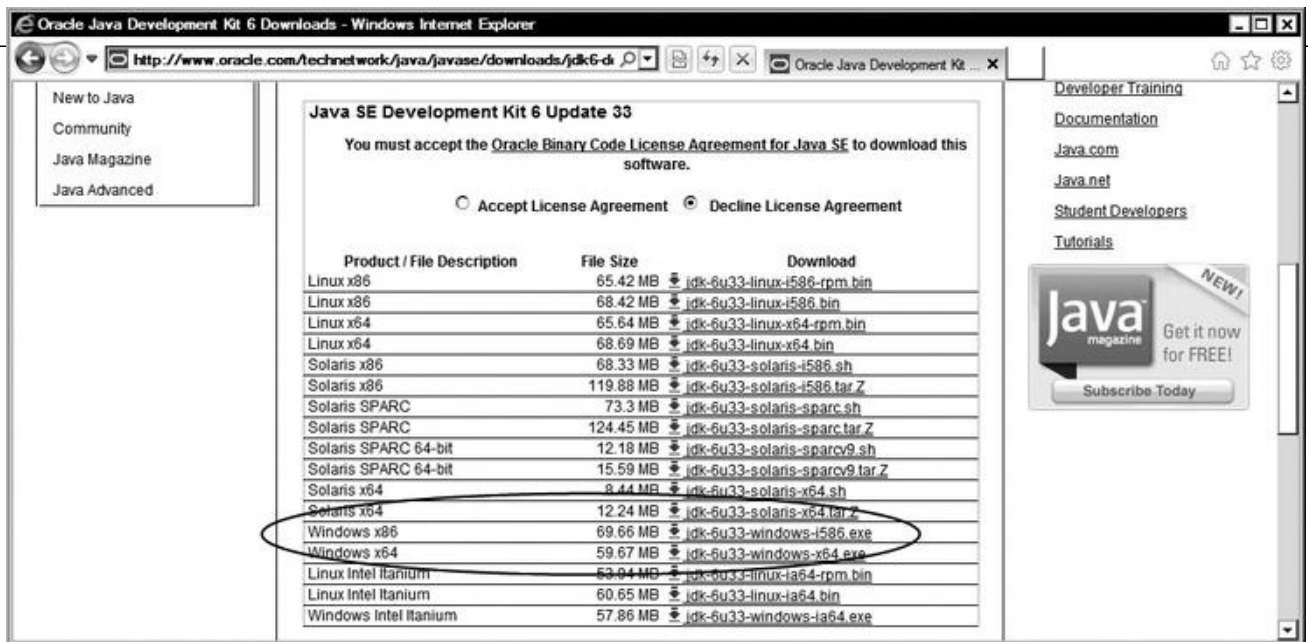


Figure 1-2 . Download Oracle JDK 6 for Windows x86

Now you can install. The Oracle JDK 6 installation package for Windows comes with a graphical installation wizard. The installation wizard will guide you through the process of installing JDK. The installation wizard will first install the JDK, and then the JRE. During the installation process, the wizard will ask for the destination directories, as well as the components to be installed. You can continue with the default values here. Make a note of the installation directory for the JDK part, shown in Figure 1-3.

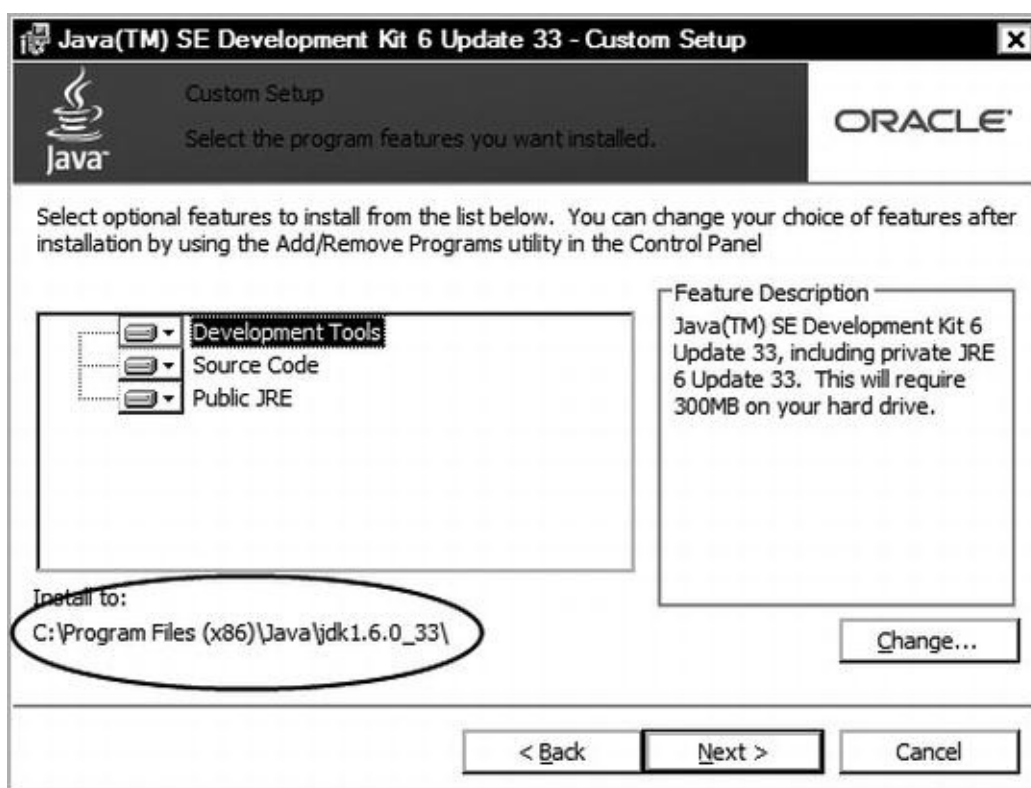


Figure 1-3 . Oracle JDK 6 installation directory

The JDK will be ready to use upon completion of the installation process. The installation wizard does not automatically add the Java binary directory into the system executable search

path, also known as the PATH variable. This needs to be done manually as the last step of the JDK installation.

1. Choose Control Panel from the Start button menu.
2. Click the System icon to launch the System Properties dialog.
3. Switch to the Advanced tab and click the Environment Variables button, as shown in [Figure 1-4](#).

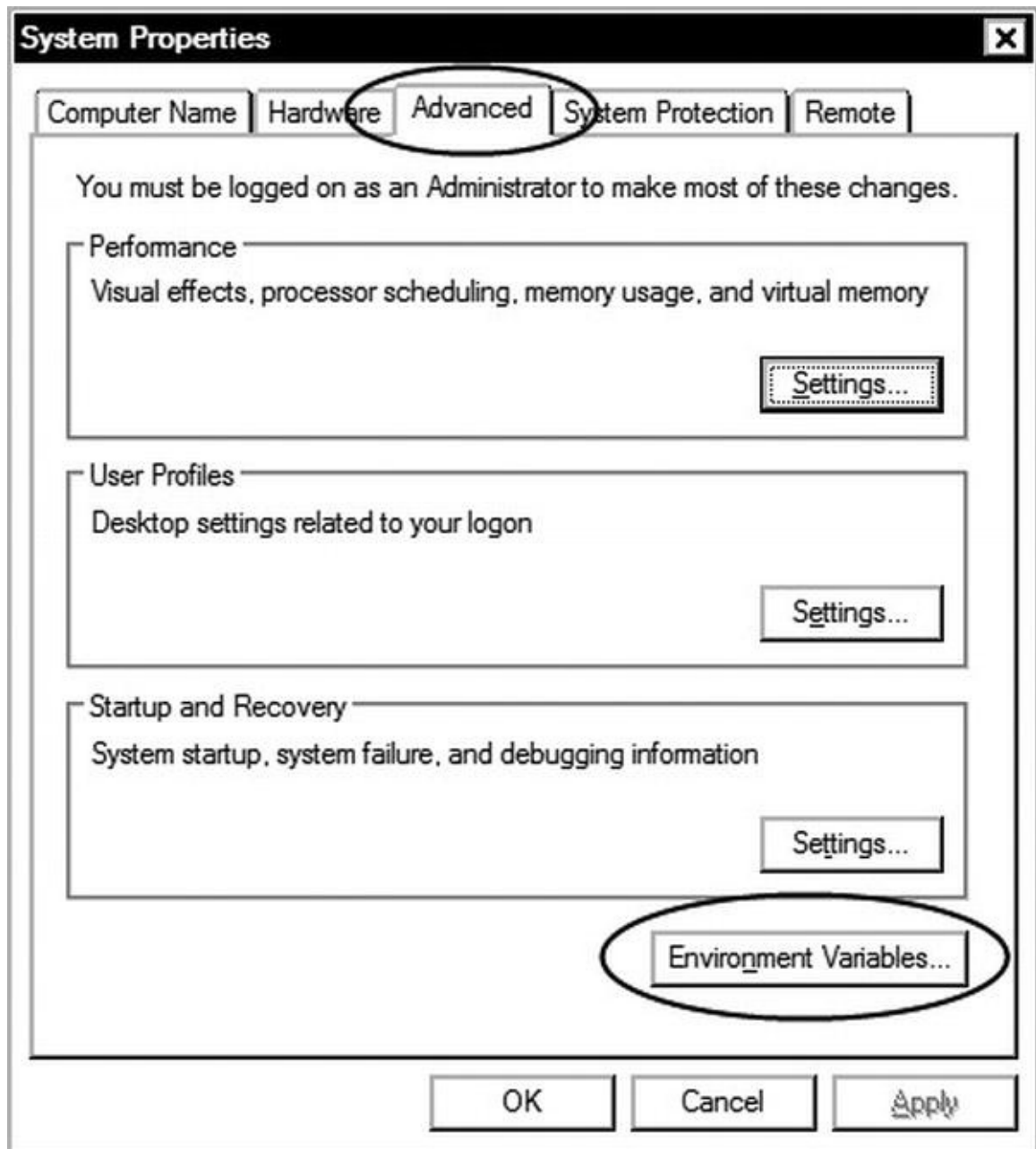


Figure 1-4 . System Properties dialog

4. Clicking the Environment Variables button will launch the Environment Variables dialog. The dialog is separated into two parts: the top one is for the user and the bottom is for the system.
5. Click the New button in the system variables section to define a new environment variable, as shown in [Figure 1-5](#).

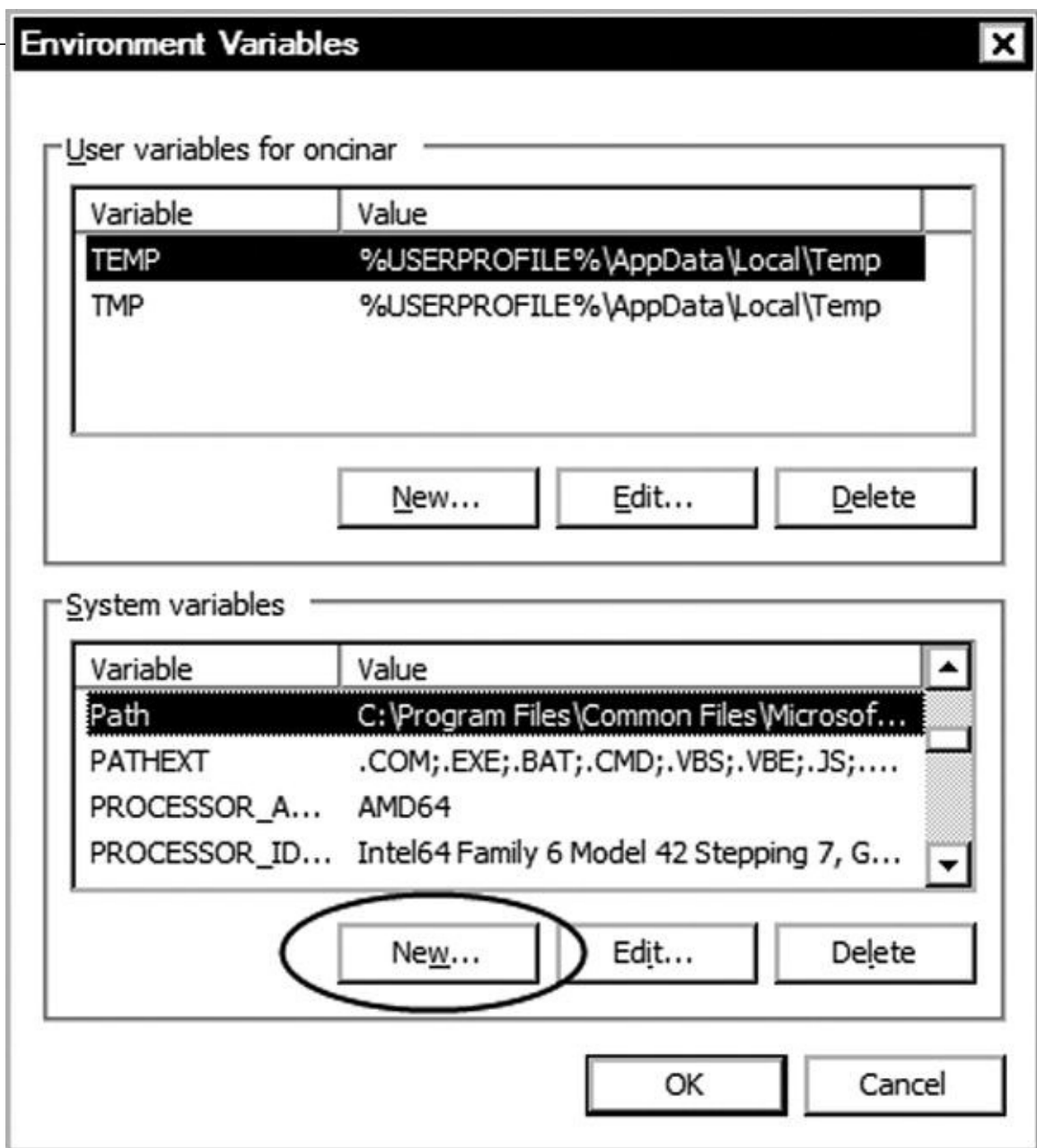


Figure 1-5 . Environment Variables dialog

- Set the variable name to `JAVA_HOME` and the variable value to the Oracle JDK installation directory that you noted during the Oracle JDK installation, as shown in Figure 1-6.
- Click OK button to save the environment variable.

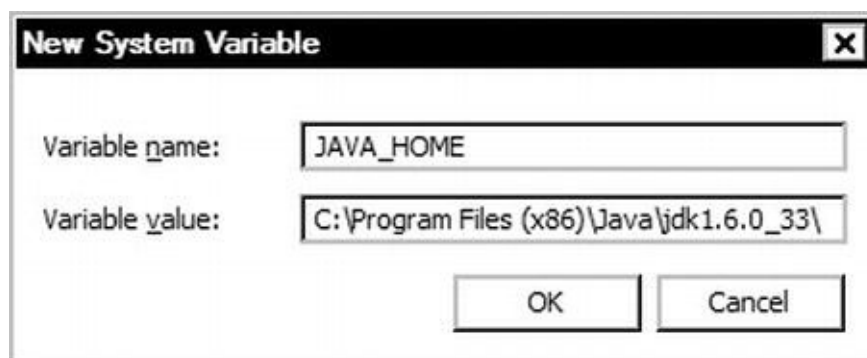


Figure 1-6 . New `JAVA_HOME` environment variable

- From the list of system variables, double-click the PATH variable and append `;%JAVA_HOME%\bin` to the variable value, as shown in [Figure 1-7](#).

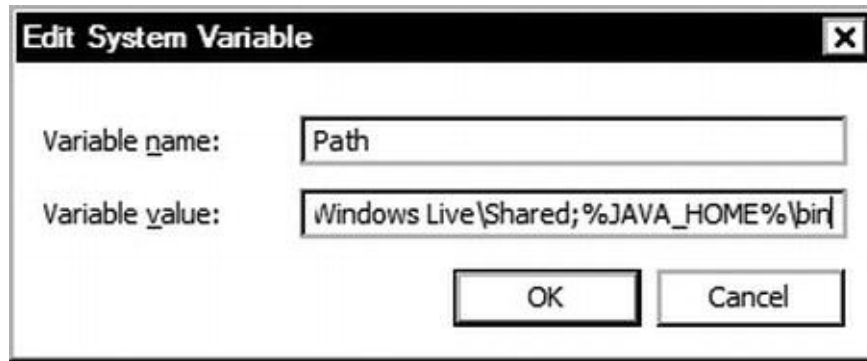


Figure 1-7 . Appending Oracle JDK binary path to system PATH variable

The Oracle JDK is now part of the system executable search path and it is easily reachable. In order to validate the installation, open a command prompt window by choosing **Start** ► **Accessories** ► **Command Prompt**. Using the command prompt, execute `javac -version`. If the installation was successful, you will see the Oracle JDK version number, as shown in [Figure 1-8](#).



Figure 1-8 . Validating Oracle JDK installation

Downloading and Installing the Apache ANT on Windows

Apache ANT is a command-line build tool that whose mission is to drive any type of process that can be described in terms of targets and tasks. Android development tools require Apache ANT version 1.8 or later for the build process to function. At the time of this writing, the latest version of Apache ANT is 1.8.4.

In order to download Apache ANT, navigate to <http://ant.apache.org/bindownload.cgi> and download the installation package in ZIP format, as shown in [Figure 1-9](#). Then follow these steps:

- [**When Magoo Flew: The Rise and Fall of Animation Studio UPA pdf, azw \(kindle\)**](#)
- [Insecure Gulf: The End of Certainty and the Transition to the Post-Oil Era online](#)
- [**download The Sherlockian online**](#)
- [click The 15-Second Handstand Beginner's Guide](#)
- [Solaris \(BFI Film Classics\) pdf, azw \(kindle\), epub, doc, mobi](#)

- <http://drmurphreesnewsletters.com/library/When-Magoo-Flew--The-Rise-and-Fall-of-Animation-Studio-UPA.pdf>
- <http://metromekanik.com/ebooks/Insecure-Gulf--The-End-of-Certainty-and-the-Transition-to-the-Post-Oil-Era.pdf>
- <http://www.mmastyles.com/books/The-Sherlockian.pdf>
- <http://www.freightunlocked.co.uk/lib/The-15-Second-Handstand-Beginner-s-Guide.pdf>
- <http://www.satilik-kopek.com/library/Marie-Claire--ZA---June-2015-.pdf>