Real-time operating system in Java

Qinghua Lu

University of Wollongong

Follow this and additional works at: https://ro.uow.edu.au/theses

Recommended Citation


Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Real-time Operating System in Java

A thesis submitted in fulfilment of the requirements for the award of the degree

Master of Computer Science -Research

from

UNIVERSITY OF WOLLONGONG

by

Qinghua Lu

School of Computer Science & Software Engineering

August, 2007
Dedicated to
My Parents,
Lu Changyou and Luo Xiue!
The following papers were written as part of this research.


Declaration

I, Qinghua Lu, declare that this thesis, submitted in fulfilment of the requirements for the award of Master of Computer Science -Research, in the School of Computer Science & Software Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Qinghua Lu
31 August 2007
# Table of Contents

List of Abbreviations................................................................................................................. 8  
Abstract .................................................................................................................................... 9  
Acknowledgement................................................................................................................... 11  
Chapter 1 Introduction........................................................................................................... 12  
  1.1 Background and Motivation ..................................................................................... 13  
  1.2 Objectives .................................................................................................................. 14  
  1.3 Outline of the Thesis ................................................................................................. 15  
Chapter 2 Real-time Operating Systems.............................................................................. 16  
  2.1 What is a RTOS? ....................................................................................................... 16  
  2.2 Basic Concepts of the RTOS.................................................................................... 16  
    2.2.1 Tasks ................................................................................................................... 16  
    2.2.2 Design Architecture ........................................................................................... 18  
    2.2.3 Scheduling .......................................................................................................... 18  
    2.2.4 Polling and Interrupts ........................................................................................ 20  
    2.2.5 Timer ................................................................................................................... 20  
    2.2.6 Threads and Events ............................................................................................ 21  
    2.2.7 Inter-process Communication ........................................................................... 22  
    2.2.8 Memory Management ........................................................................................ 22  
    2.2.9 Testing and Performance measurement............................................................ 22  
    2.2.10 Networking ....................................................................................................... 23  
Chapter 3 RTOSes in a Safe Language.................................................................................. 24  
  3.1 The Language Requirements of RTOSes ................................................................ 24  
  3.2 What is a Safe Language? ......................................................................................... 25  
  3.3 Low-level Languages ............................................................................................... 26  
  3.4 High-level Languages .............................................................................................. 27  
    3.4.1 The C Programming Language ......................................................................... 27  
    3.4.2 The Oberon-2 Programming Language ............................................................ 28  
    3.4.3 The Java Programming Language .................................................................... 29  
    3.4.4 Issues with Using Java ....................................................................................... 31  
  3.5 Low-level Issues of Developing an OS in a Safe High-level Language ..................... 32  
  3.6 Examples of OSes Developed in a Safe Language .................................................... 33  
    3.6.1 XO/2 ................................................................................................................... 33  
    3.6.2 JX Operating System ......................................................................................... 35  
    3.6.3 Singularity Operating System .......................................................................... 36  
Chapter 4 Design of JARTOS............................................................................................... 38  
  4.1 Real-time Design Issues ............................................................................................ 38  
    4.1.1 Interrupts ............................................................................................................. 38  
    4.1.2 Scheduling .......................................................................................................... 39  
    4.1.3 Inter-process Communication ........................................................................... 39
4.4 User Process Design..................................................................................................47
  4.4.1 Process Method Structure..................................................................................48
  4.4.2 The Life of a Process .......................................................................................50
  4.4.3 Process Timing..................................................................................................51
  4.4.4 Events ..............................................................................................................51
  4.4.5 Inter-process Communication ..........................................................................54

4.5 Tables Design..........................................................................................................56
  4.5.1 Process Control Block ......................................................................................56
  4.5.2 Configuration Constants..................................................................................57
  4.5.3 OS Table ............................................................................................................57
  4.5.4 Process Table ....................................................................................................58
  4.5.5 Scheduler Table................................................................................................58
  4.5.6 Event Table.......................................................................................................59
  4.5.7 Memory Table..................................................................................................59
  4.5.8 Message Table..................................................................................................59
  4.5.9 Circular Buffer Table.......................................................................................60
  4.5.10 Common Data Table.......................................................................................60

4.6 OS Methods ............................................................................................................60
  4.6.1 Timer Interrupt Handler ....................................................................................60
  4.6.2 Performance Probe...........................................................................................61
  4.6.3 Enable Interrupt.................................................................................................62

4.7 OS Processes ............................................................................................................62
  4.7.1 Start Application Process ................................................................................62
  4.7.2 Stop Application Process ................................................................................63
  4.7.3 Terminate Process.............................................................................................63
  4.7.4 Idle Process.......................................................................................................63
  4.7.5 Timer Process....................................................................................................63
  4.7.6 Event Monitor Process.....................................................................................64
  4.7.7 Message Monitor Process................................................................................64
  4.7.8 Garbage Collector Process ...............................................................................65
  4.7.9 Performance Analysis Process .......................................................................65
  4.7.10 Timeout Report Process..................................................................................65

4.8 O.S. Supervisor calls...............................................................................................65
  4.8.1 Get Message.....................................................................................................65
  4.8.2 Send Message..................................................................................................66
  4.8.3 Receive Message...............................................................................................66
  4.8.4 Release Message...............................................................................................67
  4.8.5 Add to Circular Buffer.....................................................................................67
  4.8.6 Remove from Circular Buffer .........................................................................68
  4.8.7 Wait..................................................................................................................68
  4.8.8 Wait Event.......................................................................................................68
  4.8.9 Others..............................................................................................................69
4.9 Library of Event Handlers ................................................................. 69
Chapter 5 Code Design of JARTOS ...................................................... 70
5.1 TINI Architecture .......................................................................... 70
5.2 Overview of Code Design ............................................................. 74
5.3 Can Java Implement the Design of JARTOS? .............................. 76
5.4 Low-level Issues ........................................................................... 77
5.5 Design of Testing .......................................................................... 79
  5.5.1 Test Harness ........................................................................... 79
  5.5.2 Test Application ..................................................................... 80
  5.5.3 Assertions ............................................................................. 80
Chapter 6 Code Implementation ........................................................... 82
6.1 Classes .......................................................................................... 82
  6.1.1 OS Tables ............................................................................. 82
  6.1.2 Processes ............................................................................. 83
6.2 Passing Object by Reference ......................................................... 84
6.3 Scheduling Processes ................................................................. 85
6.4 Low-level Issues ........................................................................... 85
6.3 Test Harnesses ............................................................................ 85
Chapter 7 Performance Measurement .................................................. 87
7.1 Performance Measurement of Java Instructions Running on TINI ...... 87
  7.1.1 Testing the getHundredth() .................................................... 88
  7.1.2 Testing the WHILE loop ....................................................... 88
  7.1.3 Testing the System.out.println() ........................................... 92
7.2 Impact of JARTOS on Performance of Java instructions .......... 94
  7.2.1 Testing the WHILE loop ....................................................... 94
  7.2.3 Testing the System.out.println() ........................................... 97
7.3 Performance Measurement of JARTOS ........................................... 98
  7.3.1 Testing Clock Simulation ....................................................... 98
  7.3.2 Testing the Timer Interrupt Handler ....................................... 99
  7.3.3 Testing the Process Overhead Time ....................................... 100
  7.3.4 Testing the Flow of Control of JARTOS ............................... 100
  7.3.5 How Long should Clock be? ................................................. 105
  7.3.6 Reliability Testing of JARTOS ............................................... 106
Chapter 8 Conclusion and Future Work ................................................. 110
8.1 Future Work ................................................................................. 111
  8.1.1 Network ............................................................................... 112
  8.1.2 Sun SPOT ............................................................................ 112
Bibliography ...................................................................................... 115
Appendix A System Library ................................................................. 122
Appendix B Processes Provided with OS .......................................... 144
Appendix C OS Kernel ........................................................................ 148
Appendix D Test Applications ............................................................ 153
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>EDF</td>
<td>Earliest-deadline-first</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>Mac OS</td>
<td>The Macintosh Operating System</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>the latest version of the Mac OS</td>
</tr>
<tr>
<td>MMU</td>
<td>Memory Management Unit</td>
</tr>
<tr>
<td>msec</td>
<td>millisecond</td>
</tr>
<tr>
<td>msg.</td>
<td>message</td>
</tr>
<tr>
<td>no.</td>
<td>number</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>proc.</td>
<td>process</td>
</tr>
<tr>
<td>PC</td>
<td>Program Counter</td>
</tr>
<tr>
<td>rti</td>
<td>return from interrupt</td>
</tr>
<tr>
<td>RM</td>
<td>Rate-Monotonic</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-time Operating System</td>
</tr>
<tr>
<td>RTOSes</td>
<td>Real-time Operating Systems</td>
</tr>
<tr>
<td>Sun SPOT</td>
<td>Sun Small Programmable Object Technology</td>
</tr>
<tr>
<td>TCB</td>
<td>Task Control Block</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TNI</td>
<td>Tiny InterNet Interface</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
</tbody>
</table>
Real-time operating systems (RTOSes) are required to run for years, and never fail, without human intervention. Safety is the primary concern for RTOSes because they usually control physical equipment. One strand of real-time operating system (RTOS) research is looking at the question: can developing an RTOS in a safe language result in a system that an errant process can’t crash? Choosing a good programming language can significantly improve the safety of the RTOS. In this thesis, we examine the advantages and associated problems of writing RTOSes in a safe language, namely Java.

We design an RTOS named JARTOS that schedules processes on a micro-controller called TINI. The code of the JARTOS system is mainly written in Java, since Java provides both static and dynamic safety. The Java compiler handles potentially unsafe operations rather than the programmer. Also, Java includes run-time support to catch and handle run-time errors.

JARTOS is designed to be a time-sharing system, where cooperative multiprocessing is used to schedule real-time processes. JARTOS switches processes on a timer interrupt. Each process is required to execute quickly and then give up the processor. Otherwise it will be timed out. To implement a timeout, JARTOS supports a timer interrupt that regularly updates a clock and checks for timeouts. To keep the number of interrupts to a minimum, input/output is done using polling where possible. Also, interrupts code is designed to be transparent to the processes. An interrupt handler sets flags and values, and then returns to the process it interrupted.

In the context of achieving real-time performance, we look at the issues of implementing our system design in Java. We introduce how we used Java constructs to implement the design of JARTOS, and how we solved the low-level issues.
RTOSes have to guarantee that real-time processes execute within specified time deadlines. Loss of synchronization can occur when deadlines are not met. Timing problems are often very difficult to find. In JARTOS, we designed a set of performance measurements to investigate timing problems. These performance measurements are carefully designed to provide the right information at minimal cost in performance. Performance of TINI and JARTOS are measured and discussed.
First, I would like to express my deepest appreciation to my supervisor, A./Prof. Phillip McKerrow, for his guidance, patience, support and enthusiasm during the process of this thesis.

I would like to thank my co-supervisor, Dr. Zhi Quan Zhou, for guidance, patience and help during my study.

Thanks also to my colleagues Sherine Antoun and Li Chen for their help and friendship.

From the bottom of my heart, I would like to express my special thanks to my parents, Lu Changyou and Luo Xiue, for their endless love, encouragement and support during my study and all of my life.

Finally, special thanks to my fiancé, Qiao Shuaichang, for his endless love, helpful advice. Without his support, the work in this thesis would not have been possible.