Table of Contents

About this Guide i

1. Purpose and scope i
2. Feedback i
3. Copyrights, Licenses and Trademarks ii
4. Acknowledgements ii

I. Operating system basics i

1. Real-time and embedded operating systems 1
    - 1.1. OS responsibilities 1
    - 1.2. Trade-offs 3
    - 1.3. Time 5
    - 1.4. Embedded OS 9
    - 1.5. Operating system standards 11
    - 1.6. Linux for real-time and embedded 14
2. Task management and scheduling 17
    - 2.1. Processes and threads 17
    - 2.2. POSIX thread management 18
    - 2.3. Linux tasks and tasklets 20
    - 2.4. Scheduling 21
    - 2.5. Priority-based scheduling 22
    - 2.6. Priority spaces 24
    - 2.7. Linux scheduler 24
    - 2.8. Linux real-time scheduling 25
3. Interrupts 27
    - 3.1. Introduction 27
    - 3.2. Interrupt hardware 27
    - 3.3. Interrupt software 29
    - 3.4. ISR, DSR and ASR 32
4. IPC: synchronization 35
    - 4.1. IPC terminology 35
    - 4.2. Race conditions and critical sections 37
    - 4.3. Signals 39
    - 4.4. Exceptions 40
    - 4.5. Atomic operations 41
    - 4.6. Semaphore, mutex, spinlock, read/write lock, barrier 42
    - 4.7. Condition variable for synchronization within mutex 50
    - 4.8. Priority inversion 53
    - 4.9. Priority inheritance and priority ceiling 54
    - 4.10. Lock-free synchronization for data exchange 56
5. IPC: Data exchange 57
    - 5.1. Shared memory 57
    - 5.2. FIFOs 58
    - 5.3. Messages and mailboxes 58
    - 5.4. Circular buffers 60
    - 5.5. Swinging buffers 61
    - 5.6. Remote Procedure Calls 61

6. Memory management 63
- 6.1. Terminology 63
- 6.2. Shared memory in Linux 64

7. Real-time device drivers....68
- 7.1. Mechanism and policy....68
- 7.2. Device drivers in UNIX....68
- 7.3. Complex device drivers 69
- 7.4. Comedi 70
- 7.5. Real-time serial line....72
- 7.6. Real-time parallel port....72
- 7.7. Real-time networking 72

8. RTAI: the features 74
- 8.1. Overview 74
- 8.2. Task management and scheduling 74
- 8.3. Interrupts and traps....83
- 8.4. IPC: synchronization 84
- 8.5. IPC: data exchange. 88
- 8.6. Memory management....97
- 8.7. Real-time device drivers 97
- 8.8. `/proc` interface 97
- 8.9. RTAI loadable modules 97
- 8.10. Specific features....98

9. Linux-based real-time and embedded operating systems 100
- 9.1. Introduction 100
- 9.2. RTLinux: Real-Time Linux 100
- 9.3. RTAI: the Real-Time Application Interface 102
- 9.4. uCLinux....103
- 9.5. Etlinux 103

10. Non-Linux real-time operating systems....104
- 10.1. The Adeos nano-kernel....104
- 10.2. eCos 104
- 10.3. RT-EMS 105
- 10.4. FIASCO and DROPS 105
- 10.5. Real-time micro-kernel....105
- 10.6. KISS Realtime Kernel 105

II. RTOS implementation....107

11. RTAI: the implementation....108
- 11.1. The RTAI source tree....108
- 11.2. Hardware abstraction layer....110
- 11.3. Linux compatibility layer 115
- 11.4. RTOS core 116
- 11.5. LX/RT....118
- 11.6. Making your own extensions to LX/RT 125
- 11.7. Module implementations....125

12. C++ and real-time 126
- 12.1. C and C++ 126
- 12.2. C++ in the Linux RTOSs....127

13. Cross compilation, debugging and tracing 129
13.1. Cross development 129
13.2. Debugging 129
13.3. Linux Trace Toolkit 129
III. Design 131
14. Design principles 132
14.1. Structure and functionality 132
14.2. Loose coupling 133
14.3. Mediator 134
14.4. Components 134
14.5. Architecture 136
15. Patterns and Frameworks 137
15.1. Definitions 137
15.2. Monitor 137
15.3. Producer-Consumer 144
15.4. Events 147
15.5. State Machines 150
15.6. Execution Engine 152
15.7. Distributed IPC 153
15.8. Transactions 153
16. Design example: “control” 154
16.1. What is control? 154
16.2. Functional components 155
16.3. Infrastructural components 156
16.4. Design 157
16.5. Implementation 158
IV. Tips and tricks 160
17. Tips and tricks 161
17.1. Tasks 161
17.2. Signals 161
17.3. Condition variables 162
17.4. Locks 162
17.5. Interrupts 163
17.6. Memory 163
17.7. Design 164
17.8. Programming 164
Bibliography 166