Solution approach in migration from RTOS to Linux in development of embedded system

NTT DATA MSE CORPORATION
May 27, 2013
Who am I?

Hiroto Imamura

NTT DATA MSE CORPRATION
Solution Business OHQ
Automotive Business Department

- Has been working for Linux system development of mobile phone, and provided professional works in various technical areas such as performance optimization, security, architecture design, and system debugging.

- Recently working as a consultant in Linux system development for IVI system utilizing the expertise.
Changes in circumstances surrounding an automotive area

Explosion of software steps by additional functionalities from market needs
Growing necessity of use of Linux in automotive industry

Shift to high-value added development

Less cost of common area + Base platform for high-value area (Network, Web system)

High-value Area
Commoditization area
Platform

High-value Area
Commoditization area
Platform

To high-value development

Common Characteristic

Linux
Make good use of Linux

<table>
<thead>
<tr>
<th>Tools and Framework</th>
<th>Speed of evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESF</td>
<td>OpenGL</td>
</tr>
<tr>
<td>Android</td>
<td>Bluetooth</td>
</tr>
<tr>
<td>Eclipse</td>
<td>Java</td>
</tr>
<tr>
<td>Tizen</td>
<td>NFC</td>
</tr>
<tr>
<td>CodeSourcery</td>
<td>WiFi</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>GPS</td>
</tr>
<tr>
<td>Git</td>
<td>Felica</td>
</tr>
<tr>
<td>CentOS</td>
<td>IPV6</td>
</tr>
<tr>
<td></td>
<td>HDMI</td>
</tr>
</tbody>
</table>
Network connection of Automotive systems

Cloud data Use
- weather information service
- driving information system
- Intelligent Transport System

Smartphone Use
- Infotainment
- Meter
- Audio/Video

ECU Information
- Engine/Gear/Brake
- Steering wheel/Mirror/Battery
## Differences between RTOS and Linux

<table>
<thead>
<tr>
<th>Item</th>
<th>RTOS</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution elements</td>
<td>Task</td>
<td>Process/Thread</td>
</tr>
<tr>
<td>Priority/Scheduler</td>
<td>Static</td>
<td>Dynamic/TSS</td>
</tr>
<tr>
<td>Memory Manager</td>
<td>Linear Space</td>
<td>MMU</td>
</tr>
<tr>
<td>Versatility</td>
<td>System specific</td>
<td>General purpose</td>
</tr>
<tr>
<td>License</td>
<td>Proprietary</td>
<td>Free</td>
</tr>
<tr>
<td>Middleware</td>
<td>Additional</td>
<td>Included</td>
</tr>
<tr>
<td>Application Development</td>
<td>Need to develop libraries and framework</td>
<td>Can use variety of libraries and framework</td>
</tr>
<tr>
<td>BSP development and porting</td>
<td>Developers prepare</td>
<td>Chip vendor prepare</td>
</tr>
</tbody>
</table>
Supposed challenges in migration to Linux

Challenges in migration from RTOS to Linux in development of embedded system

Requirement  →  Design  →  Coding → Evaluation

Design to realize target performance within limited resource.

Way of performance enhancement

Efficient debug method to solve problems of freeze/reset/optimization etc.

How to integrate large scale software?

How to realize secure system?
Need solutions to migrate from RTOS to Linux

RTOS → Linux
Many challenges!

Solution

Various open source software, tools...
What do I choose?
How do I use?

System architectural design, system debug...
Very difficult!
## Suggestion to automotive area

<table>
<thead>
<tr>
<th>Theme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural design</td>
<td>Architectural design in development using embedded Linux. (Priority Design, Memory Usage Design)</td>
</tr>
<tr>
<td>Efficient debug method</td>
<td>Critical problem solving method in evaluation phase in development using embedded Linux. (RAM Dump mechanism, Process Tracing and Profiling)</td>
</tr>
<tr>
<td>Boot time optimization</td>
<td>Approach method to boot time optimization. (Profiling, Prelinking, Lazy Loading)</td>
</tr>
<tr>
<td>Software build management</td>
<td>Large scale software build management in functionally distributed development.</td>
</tr>
<tr>
<td>Security</td>
<td>Method to realize secure system that guard resources using embedded Linux.</td>
</tr>
</tbody>
</table>
Need appropriate architectural design for Linux to avoid unintentional behavior.
Quality of architectural design leads to product satisfaction and cost down of development.
Efficient debug method - challenge

Reset/freeze

Reset

Freeze

Freeze...

RAM knows everything

Process’s behavior

System executes, but...

Boot time 30 sec

Response time for user operation 1.2 sec

Lack of frame during video play

Need efficient debugging method for critical problem solving
Efficient debug method - approach

Reset/Freeze

RAM Dump mechanism

RAM dump obtain (Target)
- reset
- freeze

Kernel panic
- Process defect
- Freeze

System Reboot

RAM Dump to SD or USB

RAM dump obtain (PC)

Dump data analysis

Address conversion
- Via SD/USB

Analyze linear address space by address conversion.
Address space conversion and data formatting.
Store register set data of bad process and additional info.

Process’s behavior

Process Tracing and Profiling

Point

ftrace
bootchart

Visualizing system behavior makes efficient problem analysis
Boot time optimization - challenge

**RTOS**

- Kernel boot
- Memory initialize
- Driver A Set up
  - Searching devices
  - Initialization
- Driver B Set up
  - Searching devices
- Driver C Set up
- Setting A
- Setting B
- Ex) Show up back monitor

**Linux**

- Kernel boot
- Memory Initialize
- Driver A Set up
- Driver B Set up
- Driver C Set up
- Driver D Set up
  - Searching devices
- Dynamic Library Loading
- Setting A
- Setting B
- Ex) Show up back monitor
- Loading time of Libraries

Versatility of Linux makes unnecessary procedure

Large system size makes long initializing time
Boot time optimization - approach

1. System profiling and tuning.
2. Delete unnecessary procedure.
3. Reducing library loading time before application start.

Analysis bottle neck before starting target application, and improve...
Annual developed software steps in RTOS and Linux (our experience)

<table>
<thead>
<tr>
<th>RTOS system</th>
<th>Linux system</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000ks</td>
<td>20000ks</td>
</tr>
<tr>
<td>30000ks</td>
<td></td>
</tr>
</tbody>
</table>

More than 3 times of RTOS system

Collapsed software development

Waterfall type distributed development

A company developers
B company developers
C company developers

Check in
Release
Manual Build

Miscommunication
Mistake in confirmation
Mistake of check in
Mistake in SCM tool operation

Failure rate: 89%

Too large to analyze integration failure
Build Full-time Team

Waterfall type distributed development

A company developers
B company developers
C company developers

Build Team

Full automatic build

Automatic Smoke test

Analysis of build result for each pattern

Feedback of trouble

Point

At build error, recovery build ran. Checked results and put trouble module out of mainstream.

Failure rate: 0.5%

Run automatically several build pattern, and specify trouble module.

Pick up test case considering requirement and run automatically.

When trouble in integration
- Feedback information of error, and avoid developer to make mistake.
- Make recovery build not to stop development.

Improved software integration process makes efficient large scale software development
Security - challenge

Security control by normal access control of Linux (Discretionary Access Control)

- Owner set Read/Write/Execute permission to files
- Owners’ access control is not applied to administrator
- Administrator can access to any resource to guard

Linux kernel

Administrator is not controlled by DAC.

Discretionary Access Control

Owner
Read/Write permitted

The Others
Read/Write prohibited

Resources

Personal information
Vehicle Identification Number
Frame Number

Access permitted
Access denied

Importance of security is growing as automotive is connected to open network
MAC is effective for security control of resources.

- Define access policy.
- Access policy is judged at each access request.
- Administrator’s access is under control of MAC.

Linux kernel

Mandatory Access Control (MAC)

Policy judgment

Access request

Access denied

Access permitted

administrator

owner

other

Resources

Personal information

Vehicle Identification Number

Frame Number
Automotive-SPICE for Linux/OSS

Linux 2.4  Version UP  Linux 2.6

Pilot project

Linux 2.4

Display
Sound
Telephony

Porting

Linux 2.6

- Porting Guide
- API differential information

Product development

SPICE

requirements
Specification
Design

System Test

Verification
Software Test

Manufacture

Linux 2.6
Linux can be platform for creating service