The Operating System – A Swiss Army Knife?

Commodity operating systems provide a rich set of features to be prepared for all kinds of applications and contingencies:

- Malicious or erroneous applications
  - preemptive scheduling, address space separation, disk quotas
- Multi-user operation
  - authentication, access validation and auditing
- Multi-threaded and interacting applications
  - Threads, semaphores, pipes, sockets
- Many/large concurrently running applications
  - virtual memory, swapping, working sets
Clearly, the operating system design must be strongly influenced by the type of use for which the machine is intended. Unfortunately it is often the case with 'general purpose machines' that the type of use cannot be easily identified; a common criticism of many systems is that in attempting to be all things to all men they wind up being totalement satisfactory to no-one.


Some applications may require only a subset of services or features that other applications need. These 'less demanding' applications should not be forced to pay for the resources consumed by unneeded features.

Between a Rock and a Hard Place...

Functional and nonfunctional requirements

Application

System Software

Hardware

Functional and nonfunctional properties

High variety of functional and nonfunctional application requirements
High variety of hardware platforms
High per-unit cost pressure

System software has to be tailored for each concrete application

Configurable Software – Software Product Line

Problem Space

Solution Space

Domain Expert

Architecture and Implementation

Configuration

Variant

System User

Specific Problem

Specific Solution

Intentional side

Extensional side

Features and Dependencies

KSS

© dl KSS (VL 7 | SS 16) 7 Summary and Discussion | 7.1 Summary 7–6

Functional and nonfunctional requirements

Tasks
Sockets
File system
Event latency
Safety

Cache size
Coherence
IRQ latency

ISA
IRQ handling
MMU / MPU

High variety of functional and nonfunctional application requirements
High variety of hardware platforms
High per-unit cost pressure

System software has to be tailored for each concrete application

© dl KSS (VL 7 | SS 16) 7 Summary and Discussion | 7.1 Summary 7–7
Software Product Line: Building Blocks

System Software Product Lines

Problem Space Approaches

Solution Space Approaches

Feature Diagrams

Decompositional

Compositional

Generative

Focus: solution space techniques

Implementation Techniques: Classification

Decompositional Approaches

- Text-based filtering (untyped)
- Preprocessors

Compositional Approaches

- Language-based composition mechanisms (typed)
- OOP, AOP, Templates

Generative Approaches

- Metamodel-based generation of components (typed)
- MDD, C++ TMP, generators

Feature vs. Instance-Based Configuration

Not only features, but also object instances are known at compile-time:

- Facilitates optimizations (static arrays instead of linked lists, . . .)
- Advantages wrt. footprint, latency, resilience, . . .
Real-world flight-control application (11 tasks, 3 alarms, 1 ISR)

Results with eCos and ERIKA Enterprise (open source OSEK)

<table>
<thead>
<tr>
<th>eCos</th>
<th>ERIKA</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel code (bytes)</td>
<td>14763</td>
<td>6765</td>
</tr>
<tr>
<td>kernel time (instructions)</td>
<td>88465</td>
<td>46087</td>
</tr>
<tr>
<td>robustness (10⁹ SDCs)</td>
<td>148</td>
<td>18</td>
</tr>
</tbody>
</table>
Hardware-Centric Operating-System Design

Agenda

7.1 Summary
7.2 From Instance- to Interaction Tailoring
7.3 Evaluation und Diskussion
7.4 References

dOSEK: Dependability-Oriented Static Embedded Kernel

An extremely fault-tolerant OSEK implementation

- Dependability by constructive measures
  - Employ standard hardware memory protection
  - Aggressive avoidance of indirections \( \sim \) lots of inlining
  - Arithmetic encoding of the kernel path (scheduler)

Scenario: quadrotor flight-control application

- 11 tasks, 3 alarms, 1 ISR
- 53 syscall invocations

Results (compared to ERIKA enterprise)

- SDC reduction by **5 orders of magnitude**: \( 10^9 \rightarrow 10^4 \) SDCs
- Code size increases by **factor 25**: 8 \( \rightarrow \) 200 KiB
- Syscall latency increases by **factor 4**: 100 \( \rightarrow \) 400 cycles
**dOSEK: Dependability-Oriented Static Embedded Kernel**

An extremely fault-tolerant OSEK implementation

- Dependability by constructive measures
  - Employ standard hardware memory protection
  - Aggressive avoidance of indirections *→* lots of inlining
  - Arithmetic encoding of the kernel path (scheduler)

- Scenario: quadrotor flight-control application
  - 11 tasks, 3 alarms, 1 ISR
  - 53 syscall invocations

- Results (compared to ERIKA enterprise)
  - SDC reduction by 5 orders of magnitude: $10^9 \rightarrow 10^4$ SDCs
  - Code size increases by factor 25: 8 → 200 KiB
  - Syscall latency increases by factor 4: 100 → 400 cycles

  **Culprit:** arithmetically encoded scheduler *→* avoid scheduling!

---

**An OSEK System**

- **Task 1; Priority 4**
  - TASK(Task1) {
    int data = read_data();
    if (data == '0') {
      ActivateTask(Task3);
    } else {
      bb_put(data);
    }
    ChainTask(Task2);
  }

- **Task 2; Priority 5**
  - TASK(Task2) {
    setup_of_device();
    TerminateTask();
  }

- **Task 3; Priority 3**
  - TASK(Task3) {
    parse_message();
    bb_clear_buffer();
    TerminateTask();
  }

---

**An OSEK System: Control-Flow Graphs**

- Task 1; Priority 4
- Task 2; Priority 5
- Task 3; Priority 3
(Partial) Specialization of System Calls

1. **An OSEK System: Global Control-Flow Graph**

2. **(Partial) Specialization of System Calls**

   - Task 1; Priority 4
   - Task 2; Priority 5
   - Task 3; Priority 3

   ```
   data = read_data();
   if (data == '\0')
     bb_put(data);
   SetReady(Task3);
   ChainTask(Task2);
   Idle
   ```

   ```
   ActivateTask(Task3);
   ChainTask(Task2);
   ```

   ```
   bb_clear_buffer();
   TerminateTask();
   ```

   ```
   setup_of_device()
   TerminateTask();
   ```

   ```
   parse_message()
   ```

   ```
   bb_put(data);
   ```

   ```
   SetReady(Task1);
   SetReady(Task2);
   DispatchTo(Task2);
   ```

   ```
   TerminateTask();
   ```

   ```
   SetSuspended(Task1)
   ```

   ```
   SetSuspended(Task2)
   ```

   ```
   DispatchTo(Task2);
   ```

   ```
   TerminateTask();
   ```

   ```
   TerminateTask();
   ```

   ```
   Reschedule()
   ```

   ```
   ```

```

Copyright (c) dl KSS (VL 7 | SS 16) 7 Summary and Discussion | 7.2 From Instance- to Interaction Tailoring

7–18

7–19

7–20

7–21
(Partial) Specialization of System Calls [1]

Task 1; Priority 4
SetReady(Task1);
SetSuspended(Task1);
DispatchTo(Task2);

Task 2; Priority 5
SetReady(Task2);
SetSuspended(Task2);
DispatchTo(Task2);

Task 3; Priority 3
SetReady(Task3);
SetSuspended(Task3);
DispatchTo(Task3);

Idle

SetReady(Task3);
SetSuspended(Task3);
GotoIdle();

dOSEK: Dependability-Oriented Static Embedded Kernel

An extremely fault-tolerant OSEK implementation

- Dependability by constructive measures
  - Employ standard hardware memory protection
  - Aggressive avoidance of indirections ~ lots of inlining
  - Arithmetic encoding of the kernel path (scheduler)

- Scenario: quadrotor flight-control application
  - 11 tasks, 3 alarms, 1 ISR
  - 53 syscall invocations

- Results (compared to ERIKA enterprise)
  - SDC reduction by 5 orders of magnitude: \( 10^9 \rightarrow 10^4 \) SDCs
  - Code size increases by factor 25: \( 8 \rightarrow 200 \) KiB
  - Syscall latency increases by factor 4: \( 100 \rightarrow 400 \) cycles

Further application-specific tailoring pays off!

Instance-Based Tailoring (e.g., based on OIL file)

- Kernel constrained to specified features and system objects.
Interaction-Based Tailoring (e.g., based on GCFG analysis)

- Kernel constrained to specified **features** and **system objects**.
- Further constrained to **actually possible app → kernel interactions**.

**Agenda**

- 7.1 Summary
- 7.2 From Instance- to Interaction Tailoring
- 7.3 Evaluation und Diskussion
- 7.4 References

**Evaluation**

...
Wie gehts weiter? (Bachelor/Master)

SST
7,5
Systemsoftwaretechnik

EZS
5 – 7,5
BS
5 – 7,5
MW
5 – 7,5

EZS2
P: 10
BST
5
KSS
2,5
VS
5 – 7,5

Examsarbeit / Projektarbeit
Seminar

Referenzen


