Real Time Operating Systems

Nadine Buchegger
FH Wiener Neustadt
Oberpetersdorf, Austria
nadine.buchegger@fhwn.ac.at

Benjamin Moser
FH Wiener Neustadt
Bad Fischau, Austria
benjamin.moser@fhwn.ac.at

***Abstract*— Im Folgenden wird auf das Thema der Real-Time Operating Systems eingegangen. Es werden allgemeine Informationen erörtert und darauf eingegangen, wie sie sich von allgemeinen OS unterscheiden, wie sie zu unterteilen sind und sie funktionieren. Ebenso wird auf Vor- und Nachteile sowie auch auf Beispiele von RTOS eingegangen.**

Keywords*— *OS, RTOS, UNIX, Hard-Firm-Soft-Real Time, Funktionsweise von RTOS, Scheduler, Function Library, Fast Dispatch Latency, Memory Management, Vor- und Nachteile

I. EINLEITUNG

Ein Betriebssystem (OS) gehört zur Software und ist für das Bereitstellen der grundlegendsten Funktionen des Computers verantwortlich. Beim Booten des Computers wird das OS in den RAM geladen und wird als Schnittstelle zwischen Programmen und Hardware genutzt.

Bei Unix handelt es sich um Multiuser-Betriebssystemen, die heutzutage zu den einflussreichsten, aber auch zu den weitverbreitesten Betriebssystemen zählen. Das gesamte Dateisystem ist hierarchisch strukturiert bzw. organisiert, wobei die Verzeichnisse dem Root-Verzeichnis untergeordnet sind. (Unix verfolgt den Grundsatz “Everything is a file.”.)

Die Echtzeit, sprich die “Real-Time”, hat zu bedeuten, dass Aktionen, Aufgaben usw. ohne Unterbrechungen innerhalb kürzester Zeit abgearbeitet werden. Das heißt, dass innerhalb eines exakt definierten Zeitraumes (maximal) auf eine Aufgabe bzw. Ereignis reagiert wird und genau diese Zeit ist die Echtzeit.

II. VERSCHIEDENE KATEGORIEN DER ECHTZEITSYSTEME

Die Real Time Operating Systems lassen sich zusätzlich in verschiedene Kategorien einteilen und zwar: Echtzeit-Unix, Echtzeit-Kernel, Echtzeit-Betriebssystemerweiterungen und Echtzeitbetriebssysteme.

Echtzeit-Unix: Das Echtzeit-Unix verbindet die Vorteile eines Echtzeitbetriebssystem mit den Vorteilen eines Unix Betriebssystems. Zusätzlich ist es mit dem Unix System V von AT&T kompatibel und hauptsächlich wird es für Prozessleitsysteme verwendet.

Echtzeit-Kernel: Ein Echtzeitkernel ist ein UNIX kompatibler Mikro-Kernel mit Speicherverwaltung, Scheduler und Schnittstellen auf der Basis von TCP/IP. Dieser Kernel ist meist optimal an Anforderungen angepasst und beinhaltet gut optimierten Code für unterschiedliche Plattformen.

Echtzeitbetriebssystemerweiterungen: Diese Erweiterungen erweitern ein MS-Dos System, um die Funktionen eines Echtzeitbetriebssystems. Dies beinhaltet auch eine Bibliothek zur Einhaltung der Echtzeitbedingungen.

Echtzeitbetriebssysteme: Diese Art der Betriebssysteme ist sehr effizient und sehr flexibel konfigurierbar. Außerdem sind diese an UNIX orientiert.

III. WICHTIGE BEGRIFFE FÜR REAL TIME SYSTEMS

Determinismus/deterministisch: Eine Anwendung, wird als deterministisch bezeichnet, wenn diese auf einem Echtzeitbetriebssystem läuft und wenn ihr Timing in einer bestimmten Fehlertoleranz sichergestellt werden kann.
Jitter: Wenn ein Fehler bei der Planung des Timings stattfindet und dieser Fehler sich auf alle weiteren Iterationen eines Programms oder auf Programmcode außerhalb einer Schleife auswirkt, wird dies Jitter genannt. Echtzeitbetriebssysteme sind im Normalfall dafür optimiert, so wenig wie möglich Jitter zuzulassen. Ohne Jitter braucht ein Task jedes Mal dieselbe Zeit bei der Ausführung, jedoch kann es mit Jitter zu Verzögerungen im System kommen.
Response Time/Execution Time: Die Response Time ist das Zeitintervall, zwischen dem Anfang der Ausführung eines Jobs und der Beendigung des Jobs. Die Execution Time ist das Zeitintervall eines Jobs, indem er aktiv auf die Prozessorressourcen zugreift. Diese beiden Zeitspannen werden oftmals verwechselt.
Prozesse: Prozesse, oder in den Windowsbetriebssystemen auch Tasks genannt, ist eine im Ablauf befindliche Instanz eines

Computerprogramms. Der Prozess beinhaltet das Programm samt Daten und der Prozesskontext. Diese werden über den Prozess Scheduler gemanagt.

IV. Allgemeines zu Real Time Operating Systems

Real-time Operating Systems (RTOS) sind schnelle Betriebssysteme für sogenannte Echtzeitanwendungen wie zum Beispiel Antiblockiersysteme, Multimedia-Systeme oder Herzschrittmacher. Diese garantieren die Verarbeitung der Daten in einer kurzen festgelegten Zeit, das heißt ein Real-Time Operating System ist zeitgebunden, dabei sind Zeitbeschränkungen für Aufgaben festgelegt.

Der Unterschied zwischen „normalen“ Betriebssystemen und Real-Time Operating Systems ist daher die Reaktionszeit auf Ereignisse. Bei Betriebssystemen wird nicht im Vorhinein festgelegt, wann jede Aufgabe abgeschlossen sein muss. Ein Real-Time Operating System hat eine schnelle Reaktion auf Ereignisse.

RTOS basieren auf einer aufgabenbasierten Entwicklung, daher ist das Testen nach Aufgabenbereichen möglich und es gibt wenig Abhängigkeiten zwischen den Modulen. Bei ereignisgesteuerten Real-time Operating Systems gibt es keine (und wenn nur sehr geringe) Zeitverschwendungen bei der Abarbeitung von Aufgaben, da jede Aufgabe eine bestimmte Zeitspanne zugeordnet bekommt. Außerdem erfolgt die Planung in RTOS prioritätsorientiert. Demnach erhält jede Aufgabe eine individuelle Priorität. Je höher die Priorität, desto schneller wird die Antwort auf ein Ereignis geliefert.

Die Hauptaufgaben von Real-Time Operating Systems:

- Verwaltung des Prozessors und anderer Ressourcen, um Anforderungen einer Echtzeitanwendung zu erfüllen
- Synchronisation von Ereignissen
- Reaktion auf Ereignisse

Geschichtlich fängt die Entstehung der RTOS mit den Betriebssystemen an. Der erste Vorläufer eines Betriebssystems entstand mit dem Rechner der zweiten Generation in den 50er bzw. 60er Jahren, da für den ersten Rechner noch kein OS benötigt wurde. Erst ab dem Jahre 1965 entstand ein Großrechnerbetriebssystem wie z.B. UNIX. Erst dann kamen die ersten Themen mit Multitasking bzw. Multithreading zum Einsatz und es entstanden die ersten Betriebssysteme. Erst nach dieser Entwicklung kamen die nächsten Schritte, die zur Entwicklung von RTOS und anderen Systemen führten.

V. funktionsweise der Real Time Operating Systems

Im Gegensatz zu einem „general computing device“, wo externe Peripheriegeräte mit diesem verbunden sind und außerhalb zu sehen sind, hat ein RTOS (z.B ein embedded system) fast alle Peripheriegeräte innerhalb des Moduls selbst. (z.B System on Chip). Ein Real Time Operating System ist ein System, welches hohe Priorität auf die pünktliche Abarbeitung von Tasks und Senden von Diensten legt. Ein Echtzeitbetriebssystem ist dazu bestimmt, die verschiedenen Hardwareressourcen eines PC’s zu verwalten und die Anwendungen zu hosten, die auf dem Gerät laufen sollen. Außerdem kann das OS mit der analogen Außenwelt kommunizieren und auch antworten. Ein Echtzeitbetriebssystem ist so konzepiert, dass es mehrere verschiedene Anwendungen in exakter zeitlicher Abstimmung ausführen kann. Die ist vor allem nützlich für Mess- und Industriesystemen, wo eine Verzögerung ein Sicherheitsrisiko darstellen könnte. Wenn es um die zeitliche Abwicklung der Prozesse geht gibt es einige Arten, diese abzuwickeln. Entweder die Prozesse werden zu einem festen Zeitpunkt abgewickelt, was zum Beispiel bei zeitgesteuerten Programmen wichtig sein kann. Außerdem können die Prozesse zu einem festgelegten Zeitpunkt mit Toleranzen abgearbeitet werden, was bei Programmen, wo das Timing der einzelnen Jobs nicht so große Auswirkung hat nützlich sein. Des Weiteren kann ein Prozess zum spätesten Zeitpunkt abgewickelt werden, was nützlich sein kann, wenn der Prozess auf einen anderen Prozess warten muss. Außerdem kann ein Prozess auch zum frühesten Zeitpunkt abgearbeitet werden, wenn dieser Prozess zum Beispiel notwendig ist, damit das Programm läuft.

VI. Verschiedene Arten der Real Time Operating Systems

Real Time Operating Systems unterscheidet man in drei unterschiedliche Arten: Hard Real Time, Firm Real Time und Soft Real Time.

Hard Real Time Systems sind Systeme, die Prozesse zu einer gewissen Zeit starten, um diese dann zur gewissen Deadline abzuarbeiten. Hauptsächlich werden diese Art der Systeme im Gesundheitswesen und bei Flugzeigen eingesetzt.

Firm Real Time Systems sind ähnlich zu den Hard Real Time Systems, jedoch hat das Nichtschaffen einer Deadline nicht so einen großen Einfluss auf das System wie beim Hard Real Time System, kann allerdings ebenfalls die Qualität des Produkts beeinflussen. Einsatzgebiete für diese Art der Systeme sind verschiedene Multimedia Anwendungen.

Die letzte Art der Real Time Operating System sind die Soft Real Time Operating Systems. Es gibt zwar Deadlines in dieser Art, jedoch werden Verzögerungen akzeptiert jedoch rechnet das System bereits mit einer Verzögerung. Diese Art der Systeme werden hauptsächlich für Transaktionssysteme oder Virtual Reality Anwendungen verwendet.

VII. Komponenten eines Real Time Operating Systems

Die wichtigsten Komponenten eines RTOS sind der Scheduler, Function Library, Fast Dispatch Latency bzw. Context Switch Time, User-defined objects and classes, Memory Management und Symmetric Multiprocessing. Im Folgenden werden diese etwas genauer erläutert:

Scheduler

Im Allgemeinen wird mit dem Scheduler angegeben, in welcher Reihenfolge die einzelnen Aufgaben, die das RTOS erledigen muss, ausgeführt werden können in Abhängigkeit der Priorität.

Function Library

Die Function Library dient als Schnittstelle, über die der Anwendungscode mit dem Kernel verbunden wird. Dabei werden die Anforderungen mithilfe dieser Function Library an den Kernel gesendet, damit die Anwendung auch die gewünschten Ergebnisse liefert.

Fast Dispatch Latency/Context Switch Time

Die Fast Dispatch Latency gibt die Zeit ab dem Zeitpunkt an,

1. zu dem das Echtzeitbetriebssystem feststellt, dass eine Aufgabe/ein Task vollständig ausgeführt und beendet ist
2. bis ein neuer Thread (muss ausführbar sein) gestartet wird oder
3. eine neue Aufgabe/Task gestartet wird und es führt dazu, dass Aufgaben, die als wichtiger eingestuft sind, das Ausführen der derzeitigen Aufgabe verhindern.

Context Switch Time ist die Zeit des Wechsels von einem zum anderen Thread, wobei der Kontext von dem einem Thread gespeichert wird und durch den des neuen ersetzt.
In einem RTOS ist es wichtig, dass diese Context Switch Time minimal und determiniert bleibt.

User-defined data objects and classes

Das RTOS-System basiert auf Programmiersprachen wie C oder C++ mit Datenstrukturen, die entsprechend ihrer jeweiligen Funktionsweise organisiert werden sollten. Es werden Objektmengen definiert, mit denen das RTOS die Anwendung steuert.

Memory Management

Das Memory Management dient dazu, um jedem Programm bzw. Objekt einen Speicher zuzuordnen/zuzuweisen. Das ist wichtig, da in einem RTOS der Speicher nicht ein-/ ausgelagert werden kann (das würde sonst nicht determinieren).

Symmetric Multiprocessing SMP

Hierbei geht es darum, dass ein Real-Time Operating System mehrere Aufgaben/Threads verarbeiten und trennen kann, damit diese auf mehreren Kernen ausgeführt werden können. Dabei ermöglicht man die parallele Verarbeitung

und Ausführung von Code (= Symmetric Multiprocessing).

VIII. Vor- und Nachteile eines Real-Time Operating Systems

Vorteile, die sich unter anderem aus den Eigenschaften ableiten lassen, sind beispielsweise die prioritätsbasierte Planung und das einfache Testen des RTOS durch die aufgabenbasierte Entwicklung, da unabhängig an unterschiedlichen Bereichen gearbeitet werden kann. Dadurch entsteht die Wartbarkeit und die Erweiterbarkeit des RTOS, die für die geringen Abhängigkeiten zwischen den Modulen sorgen. Durch die ereignisgesteuerten RTOS besteht eine hohe Effizienz und es wird demnach keine Verarbeitungszeit verschwendet.

Zu den Nachteilen kann man zählen, dass nur das Ausführen von minimalen Aufgaben gleichzeitig möglich ist. Genauso wie Aufgaben mit niedrigen Prioritäten, da diese dann auch länger brauchen als Aufgaben mit höheren Prioritäten.

IX. Beispiele für Real Time Operating Systems

Eines der größten Beispiele für ein Real Time Operating System ist LynxOS. Bei Lynx Operating System handelt es sich um ein Echtzeitbetriebssystem der kalifornischen Firma Lynx Software Technologies, welches hauptsächlich auf dem Markt der „embedded systems“ eingesetzt wird. Neben Anwendungsfällen inder Medizin oder der Telekommunikation wird vorwiegend LynxOS bei kritischen Anwendungen in Militär, Luft- und Raumfahrtsystemen eingesetzt.

Ein weiteres großes Beispiel für ein Echtzeitbetriebssystem ist „ChorusOS“. Das Betriebssystem wurde unter anderem von Sun Microsystems entwickelt, was seit 2010 in Oracle eingegliedert worden ist. Das Bauen von offenen und skalierbaren Betriebssystemen ist durch das Design und die Technologie hinter ChorusOS gegeben. Das Architekturmuster der Software bildet eine Schnittmenge zwischen Verteilten und Adaptiven Systemen. Diese Schnittstelle nennt das Betriebssystem Kommunikation.

Noch ein großes Beispiel für ein Echtzeitbetriebssystem ist DRYOS. Es wurde entwickelt von dem Kamerahersteller Canon und wurde 2007 erstmals veröffentlicht. Wie man vom Hersteller vermuten kann wurde es für Digitalkameras und Camcorders eingesetzt. Das Betriebssystem ersetzte das ältere VxWorld von Wind River Systems, was auf älteren Kameras von Canon lief.

X. Zusammenfassung des Themas Real Time Operating Systems

Echtzeitsysteme beschreiben Systeme, welche einen Job unter bestimmten Zeitbedingungen zu erledigen haben. In dieser Gruppe von Systemen befindet sich das zu behandelnde Real Time Operating System oder auch Echtzeitbetriebssystem genannt. Die ersten Betriebssysteme entstanden in den 50er bis 60er-Jahren, wobei das Thema Real Time Operating System beziehungsweise Multitasking am Computer erst später ein Thema wurde. Der eigentliche Unterschied zwischen einem Betriebssystem wie Windows oder Mac OS und einem Echtzeitbetriebssystem wie Lynx OS besteht darin, dass ein normales Betriebssystem nicht festgelegt hat, wann welcher Job abgearbeitet werden soll. Genau dies ist der Vorteil bei einem Real Time Operating System, da dieser eine Zeitplan hat, welcher je nach Art des Betriebssystems sehr tolerant oder eben sehr strikt sein kann. Dieser Zeitplan ist jedoch nicht nur auf ein Programm oder einen Prozessorkern bestimmt, sondern wird verwendet, um mehrere Programme möglichst effizient und nach dem Zeitplan auszuführen.
Ein Real Time Operating System hat einige sehr wichtige und bedeutsame Komponenten. Dazu zählen der Scheduler, die Function Library, Fast Dispatch Latency/Context Switch Time, User-defined data objects, Memory Management und Symmetric Multiprocessing. Die wichtigste ist der Scheduler, der wie schon erläutert, die Reihenfolge der Aufgaben (den Zeitplan) festlegt. Des Weiteren ist die Context Switch Time sowie auch das Memory Management sehr wichtig. Die Context Switch Time ist die Zeitspanne für den Wechsel eines Threads, die minimal bleiben soll. Den Objekten wird immer ein Speicher zugewiesen – diese Aufgabe wird vom Memory Management übernommen.