Operating System

- Operating system is the software which is used as interface between
  user and hardware.
- It provides uniform access to hardware for the user.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner
- Ex:- Windows, Linux, Solaris etc
Real time system

- Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced.

- In such a type of system result must be obtained within the limited time constraints.

- If result is not obtained within limited time then result may be incorrect or no meaning of that result.
Typical Real-time System
Example of a Real Time System

Consider a real-time system comprised of three motors and three switches.

The switches have two positions, ON and OFF.

The switches must be scanned at about 10 times per second,

and the motors turned on or off as appropriate.
void main(void)
{
    int i;
    while(1)
    {
        for (i = 0; i < 3; i++ )
        {
            if (switchChanged(i))
                changeMotor(i);
        }
    }
}
Example of a Real Time System

```c
void main(void)
{
    while(1)
    {
        if (OneTenthSecondIsUp)
            for (i = 0; i < 3; i++)
                if (switchChanged(i))
                    changeMotor(i);
        OneTenthSecondIsUp = 0;
    }
}
```

S1                  S2 S3
M1                  M2 M3

---

EmbeddedCraft
crafting of intelligent systems
Adding one sensor

Let a pressure gage must be checked every 50 milliseconds.

A valve opened if the pressure is greater than 100 psi.

Once opened, the valve must be closed after the pressure drops below 90 psi.
Example

```c
if (FiftyMsIsUp)
{
    switch (valveState)
    {
        case CLOSED:
            if (pressure() > 100)
            {
                openValve();
                valveState = OPEN;
            }
            break;
        case OPEN:
            if (pressure() < 90)
            {
                closeValve();
                valveState = CLOSED;
            }
    }
    FiftyMsIsUp = 0;
}
```
Let us add datagrams (data packets)

Assume that the system is connected to a network

and that incoming datagrams must be processed.

This could be handled by adding yet another function call to the loop.

checkDatagrams();
Let us handle datagrams (data packets)

if the function `checkDataGrams` is not called at a sufficient rate, datagrams can be lost.

In order to avoid this, a queue must be created so that when the interrupt service routine for incoming datagrams is entered, the datagram is placed into a queue for processing by the function `checkDatagrams`.
Final code for our system...

```c
Void main()
{
    While(1)
    {
        checkMotorSwitches();
        checkPressure();
        checkDatagrams();
    }
}
```

![Diagram with switches and motors]
Final code for our system
drawback…

- There are no priorities.
- Busy waiting should not be there.
- More responsibility (e.g., timer management) is put on the programmer.
- User has to manage the queue.
- Priority to the task is not given.
RTOS Solution...

Three tasks

First Task

```c
void checkMotorSwitches(void)
{
  while (TRUE) {
    pause(100L);
    for (i = 0; i < 3; i++) {
      if (switchChanged(i)) changeMotor(i);
    }
  }
}
```

```c
void checkPressure(void)
{
  while (TRUE) {
    pause(50L);
    if (pressure() > 100) {
      closeValve();
      while (TRUE) {
        pause(50L);
        if (pressure < 90) {
          openValve();
          break;
        }
      }
    }
  }
}
```

Second Task

```c
void checkDatagrams(void)
{
  typeMsg * msg;
  while (TRUE) {
    msg = waitMsg(DATA_GRAM);
    processDataGram(msg);
    freeMsg(msg);
  }
}
```
RTOS Solution...

Main Code
----------------------
#include <RTOS.h>
Void main()
{
    InitRTOS();
    Createtask (checkmotorswitches (),1);
    Createtask (checkPressure(),2 );
    Createtask (checkDatagrams(),3 );
    StartScheduler();
}
Advantages

1. Busy waiting is eliminated.
2. Timer management is no longer a concern of the programmer.
3. \textit{checkPressure} does not have to retain a state variable.
4. Queue development and management is no longer a concern of the programmer.

5. Kernel API
   1. waitMSG()
   2. Pause()
   3. freeMSG()
   4. Createtask()
   5. StartScheduler()
# OS Used in Embedded System

## Non Real Time Embedded OS
- Embedded Linux (kernel 2.4.x) (www.embedded-linux.org)
- Windows Embedded
- Salvo

## Real Time OS
- Integrity
  - www.ghs.com
- QNX Neutrino (QNX Software Ltd.) (www.qnx.com)
- VxWorks (Wind River) (www.windriver.com)
- µC/OS-II (Jean J. Labross) (http://www.micrium.com/)
- Linux (Kernel 2.6.x) (www.linyx.org)

## Handheld/ Mobile OS
- Symbian OS (www.symbian.com)
- Windows CE (http://windowsce.com/)
- handhelds.org
Micrium's vision is to provide the highest-quality embedded software components in the industry, in the form of engineer-friendly source code with unsurpassed documentation and customer support. Micrium's products consistently deliver on that vision to shorten time-to-market throughout all product development cycles.
INTEGRITY Real-Time Operating System

INTEGRITY® is a secure, royalty-free Real-Time Operating System intended for use in embedded systems that require maximum reliability. INTEGRITY represents the most advanced RTOS technology on the market today.

Without the burden of compatibility with 1980s vintage products, INTEGRITY was designed from the ground up. It employs the latest in RTOS technology and achieves unprecedented levels of reliability, availability, and security for a broad range of real-time applications.

INTEGRITY uses hardware memory protection to isolate and protect itself and user tasks from incorrect operation caused by accidental errors or malicious tampering. Its object-oriented design allows strict access control and verification of the security and integrity of data, communications, individual components, and the system as a whole. Its strict adherence to provable resource requirements allows an embedded system designer to guarantee resource availability. Unlike other memory protected operating systems,
What is handhelds.org?

Our goal is to encourage and facilitate the creation of open source software for use on handheld and wearable computers. We welcome participation and sponsorship by individuals, groups and companies seeking to further this goal. (About handhelds.org)

Annual Server Maintenance and Upgrades Aug 13th thru the 18th

Friday, August 10 2007 @ 07:44 PM EDT
Contributed by: france
Views: 413

I will be in Corvallis, OR on Monday Aug 13th thru Aug 18th, to perform maintenance and upgrades on all the servers hosted at OSUDSL. I believe and hope that everything will fall over to the 2ndary servers, so that no services will be down during this time.

--George France

CEO, Handhelds.org, Inc 501(c)3

LinuxWorld San Francisco 2007 Report

Friday, August 10 2007 @ 05:35 PM EDT
Contributed by: france
Views: 521

It was great to meet the Handhelds.org users and developers that stopped by our booth.

Opie(TM) 1.2.3 was big hit. Many potential users were ecstatic that there is now support for the Palm Tungsten T/T2/T3/T5/C/E/E2, LifeDrive, TX, Zire 71/72, Treo 600/650/680/700w/700p/750/755p and Foleo handheld devices.
Introducing Symbian OS v9.5

The global smartphone market has never been so exciting. With over 110 million Symbian smartphones shipped, high smartphone growth in developing markets, and increasing mass market requirements, Symbian’s addressable market is broadening across segments and regions.

Symbian OS v9.5, the latest evolution of Symbian OS, delivers over 70 new features for high-performance, more powerful smartphones at mass market costs: a truly scalable operating system for the global market.

Lower hardware cost
- High-performance smartphones to run on feature phone hardware
- 20-30% lower RAM usage achievable

Faster time-to-market

Related content
Symbian announces Symbian OS v9.5
Symbian has announced the launch of Symbian OS v9.5, the latest evolution of the world’s leading operating system for smartphones. Symbian OS v9.5 brings high performance features designed for richer consumer and enterprise experiences as well as significant savings to phone build costs and time to market, delivering a truly scalable mobile operating system for the global market.

More

Symbian OS v9.5 product sheet
Symbian OS is the advanced, open operating system licensed by the world’s leading mobile phone manufacturers. It is designed for the specific requirements of advanced 2.5G and 3G mobile phones. Symbian OS combines the power of an integrated applications environment with mobile telephony, bringing advanced data services to the mass market.
**Welcome**

Salvo™ is the first Real-Time Operating System (RTOS) designed expressly for very-low-cost embedded systems with severely limited program and data memory. With Salvo, you can quickly create low-cost, smart and sophisticated embedded products. Pumpkin™ has currently certified Salvo for use with:

- 8051 family and its derivatives
- ARM® ARM7TDMI® and Cortex™ M3
- Atmel® AVR® and MegaAVR™
- Motorola M68HC11
- TI's MSP430 Ultra-Low Power Microcontroller
- Microchip PIC12|14000|16|18 PICmicro® MCUs
- Microchip PIC24 MCUs and dsPIC® DSCs
- Microchip PIC32™ MCUs
- TI's TMS320C2000 DSPs

Below is an actual email from a Salvo user who runs Salvo 4 in a C++ environment on the Atmel AT91SAM7 (ARM7 TDMI) target using the IAR C/C++ Embedded Workbench for ARM toolset:

> My project using Salvo Pro is coming along great, almost ready to release v1.0. I just wanted to drop you an email telling you how much easier my (engineering) life is with Salvo! I have been able to add significant complexity without convoluting the structure, and the resulting C++ code is clean, fast and robust.
# Open Source RTOS

<table>
<thead>
<tr>
<th>1. ucLinux</th>
<th><a href="http://www.uclinux.org">www.uclinux.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Symbian</td>
<td><a href="http://www.symbian.org">www.symbian.org</a></td>
</tr>
<tr>
<td>3. Linux Kernel 2.6.x</td>
<td><a href="http://www.kernel.org">www.kernel.org</a></td>
</tr>
<tr>
<td>5. NemuetOS</td>
<td><a href="http://www.menuetos.net/">http://www.menuetos.net/</a></td>
</tr>
<tr>
<td>6. FreeRTOS</td>
<td><a href="http://www.freertos.org">www.freertos.org</a></td>
</tr>
<tr>
<td>7. RTEM</td>
<td><a href="http://www.rtems.com/">http://www.rtems.com/</a></td>
</tr>
<tr>
<td>8. Solaris</td>
<td><a href="http://www.sun.com">www.sun.com</a></td>
</tr>
<tr>
<td>9. etc</td>
<td></td>
</tr>
</tbody>
</table>
Home Page

Introduction

eCos is an open source, royalty-free, real-time operating system intended for embedded applications. The highly configurable nature of eCos allows the operating system to be customised to precise application requirements, delivering the best possible run-time performance and an optimised hardware resource footprint. A thriving user community has grown up around the operating system ensuring on-going technical innovation and wide platform support.

For further information concerning eCos, please refer to the about eCos page. eCos is not related to the Linux operating system.

eCos news

July 11, 2006  Host tools

eCosCentric has generated new unsupported snapshot builds of the eCos host tools that work around an exception handling issue with Cygwin 1.5.19 and 1.5.20. Download and installation instructions are available.

April 20, 2005  I²C infrastructure

eCosCentric has contributed I²C (Inter-Integrated Circuit) bus infrastructure from their eCosPro offering to eCos. The code and documentation are only available from the CVS repository at this time.
Welcome to the RTEMS home page!

RTEMS is the Real-Time Operating System for Multiprocessor Systems. It is a full featured RTOS that supports a variety of open API and interface standards.

Major decisions about RTEMS are made by the Steering Committee, guided by the Mission Statement.

We encourage everyone to contribute changes and help testing RTEMS, and we provide access to our development sources with anonymous CVS and snapshots.

We strive to provide regular, high quality releases, which we want to work well on a wide range of embedded targets cross development from a variety of hosts including GNU/Linux, FreeBSD, Cygwin, and Solaris.

Active development (mainline): will become 4.9 (obtain from CVS)
Active release branches:
- RTEMS 4.8 (check out from CVS using the tag rtems-4-8-branch).
- RTEMS 4.7 (check out from CVS using the tag rtems-4-7-branch) (latest is 4.7.1)
- RTEMS 4.6 (check out from CVS using the tag rtems-4-6-branch) (last was 4.6.6)

Upcoming Events

RTEMS Classes in 2008

We now have dates for RTEMS classes in 2008. To express interest in other dates in Huntsville or to host a class at your company, please contact Joel Sherrill (joel.sherrill AT OARcorp DOT com). If interested in attending a class scheduling a company specific class in Europe, please contact Thomas Doerfler (Thomas.Doerfler AT embedded-brains DOT de). Pictures from previous classes are online [here](#).
FreeRTOS™ is a portable, open source, mini Real Time Kernel - a free to download and royalty free RTOS that can be used in commercial applications.

Ports exist for many different processor architectures and development tools. Each official port includes a pre-configured example application demonstrating the kernel features, expediting learning, and permitting 'out of the box' development.

Free support is provided by an active user community. Commercial support along with a full
Solaris Operating System

Innovation Matters: Choosing the right OS can save your business time and money.

Get It Now. FREE »

Overview | Features | Tech Specs | Perspectives | Support | Get It

At a Glance | What's New | General FAQs

The Solaris Operating System—supported on over 900 x86 and SPARC platforms—delivers the performance, stability and security your users and customers demand. With more applications available than for any other open operating system, one OS can span your entire enterprise: the Web tier, the data warehouse, and the most demanding technical compute applications.

» Get the Software » Get Trained » Get Support » Move to Solaris 10

Use Solaris 10 for:

Average Customer Rating

[Feedback]
The Linux/Microcontroller project is a port of Linux to systems without a Memory Management Unit (MMU).

Pronounced "you-see-linux", the name uClinux comes from combining the greek letter "mu" and the english capital "C". "Mu" stands for "micro", and the "C" is for "controller". uClinux first ported to the Motorola MC68328: DragonBall Integrated Microprocessor. The first target system to successfully boot is the PalmPilot using a TRG SuperPilot Board with a custom boot-loader created specifically for our Linux/PalmPilot port.

July 2007
Greg Ungerer has been posting patches against the dist for those wishing to follow the mid-release updates. The Patches can be found at the following link: http://www.uclinux.org/pub/uClinux/dist/patches/
Feed back on these patches can be posted to the uClinux-dev mailing list. If you wish to subscribe to the mailing list you can do it here https://mailman.uclinux.org/mailman/listinfo/uclinux-dev/.

July 2007
The current uClinux-dist release is dated January 30, 2007. Here is a quick links to the tar.gz and tar.bz2 packages:

- http://www.uclinux.org/uClinux/dist/uClinux-dist-20070130.tar.gz
Real time Operating system

- Real time operating systems are used as OS in real time system.
- In RTOS tasks are completed in given time constraints.
- RTOS is a multitasking system where multiple tasks run concurrently
  - system shifts from task to task
  - must remember key registers of each task
    (this is called context of task)
Characteristics of Real time Operating system

• Single purpose

• Small size

• Inexpensively mass-produced

• Specific timing requirements
Example of RTOS

- INTEGRITY, VelOSity and µvelOSity From Green hills co
  www.ghs.com

- RTLinux and VxWorks From Windriver
  www.windriver.com
  -µC/OS-II from Micrium
    www.micrium.com

- PowerPac from IAR
  www.iar.com
Real time Operating system categories

Two types

**Soft RTOS**
SOFT real-time system, tasks are performed by the system as fast as possible, but the tasks don't have to finish by specific times

**Hard RTOS**
In HARD real-time systems, tasks have to be performed not only correctly but on time
Real time System Pitfalls -1: Patriot Missile

During 1991 Gulf war, an Iraqi Scud Missile hit an American army barrack killing many soldiers. Actually an American patriot missile could not track and intercept the scud missile.

**Reason:**
A software bug in patriotic missile

When the system turned on then it measure the time in 100 ms. Intervals. This value is multiplied by 10 to obtain second. The calculation was performed in 24 bit floating register. So, value of 1/10 th second is truncated in 24 bits where as exact value is 0.00011 00110 01100 11001 1001100…

So, if value was truncated then en error occurred. (3.4 ms per hour) The patriot missile was switched on for about 100 hours, so the accumulated error was 0.34 second. Scud travels at speed of 1,676 mps, i.e. more then half a kilometer in 0.34 second. So, patriot could not intercept the scud.
Real time System Pitfalls -2: Lockheed Martin

In 1998 Lockheed Martin Titan 4 booster carrying a $1 billion LockMart Vortex Class Spy satellite pitched sideways and exploded 40 seconds after liftoff cape Canaveral, Fla.

Reason:
Fried wiring that had not been inspected. The guidance system remain without power for fraction of second.
Real time System Pitfalls -3: Mars Orbiter

One of the mars orbiter probe crashed into the planet in 1999. It did turn out that engineers who build the Mars Climate orbiter had provided a data table in “proud force” rather then Newton's, the metric measure of force. $125 Million Dollar lost

Reason:
NASA flight controllers at jet propulsion laboratory in pasadena calif., had used the faulty table for their navigation calculations during the long trip from Earth to Mars.
NASA's metric confusion caused Mars orbiter loss

September 30, 1999

(CNN) -- NASA lost a $125 million Mars orbiter because one engineering team used metric units while another used English units for a key spacecraft operation, according to a review finding released Thursday.

For that reason, information failed to transfer between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Colorado and the mission navigation team in California. Lockheed Martin built the spacecraft.

"People sometimes make errors," said Edward Weiler, NASA's Associate Administrator for Space Science in a written statement.

"The problem here was not the error, it was the failure of NASA's systems engineering, and the checks and balances in our processes to detect the error. That's why we lost the spacecraft."
Real time System Pitfalls - 4: The Ariane 5 satellite launch rocket

Rocket self destructed in 4 June -1996. Exactly after 40 second of lift off at an attitude of 3700 meters, the launcher exploded and became a ball of fire. Cost: $500 Million USD

**Reason:**
Bad floating-point exception handling.
A 64 bit floating no is converted into 16 bit signed value.
During one calculation the converted value was more then of 16 bit.

This error is know as “**500 million dollar software error**”
End of Part 1