16.480/552

Embedded Operating Systems
### Introduction

A computer system consists of:

- hardware
- system programs
- application programs

<table>
<thead>
<tr>
<th>Banking system</th>
<th>Airline reservation</th>
<th>Web browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilers</td>
<td>Editors</td>
<td>Command interpreter</td>
</tr>
<tr>
<td></td>
<td>Operating system</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Machine language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microarchitecture</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Physical devices</td>
<td></td>
</tr>
</tbody>
</table>

- Application programs
- System programs
- Hardware
What is an Operating System

• **It is an extended machine**
  – Hides the messy details which must be performed
  – Presents user with a virtual machine, easier to use

• **It is a resource manager**
  – Each program gets time with the resource
  – Each program gets space on the resource
History of Operating Systems (1)

Early batch system
- bring cards to 1401
- read cards to tape
- put tape on 7094 which does computing
- put tape on 1401 which prints output
History of Operating Systems (2)

• First generation 1945 - 1955
  – vacuum tubes, plug boards
• Second generation 1955 - 1965
  – transistors, batch systems
• Third generation 1965 – 1980
  – ICs and multiprogramming
• Fourth generation 1980 – present
  – personal computers
History of Operating Systems (3)

- Structure of a typical FMS job – 2\textsuperscript{nd} generation
History of Operating Systems (4)

• Multiprogramming system
  – three jobs in memory – 3rd generation
The Operating System Zoo

- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Real-time operating systems
- Embedded operating systems
- Smart card operating systems
Computer Hardware Review (1)

- Components of a simple personal computer
Computer Hardware Review (2)

(a) A three-stage pipeline
(b) A superscalar CPU
Computer Hardware Review (3)

Typical access time

<table>
<thead>
<tr>
<th>Access Time</th>
<th>Typical Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 nsec</td>
<td>&lt;1 KB</td>
</tr>
<tr>
<td>2 nsec</td>
<td>1 MB</td>
</tr>
<tr>
<td>10 nsec</td>
<td>64-512 MB</td>
</tr>
<tr>
<td>10 msec</td>
<td>5-50 GB</td>
</tr>
<tr>
<td>100 sec</td>
<td>20-100 GB</td>
</tr>
</tbody>
</table>

- **Typical memory hierarchy**
  - numbers shown are rough approximations
Computer Hardware Review (4)

Structure of a disk drive
Computer Hardware Review (5)

One base-limit pair and two base-limit pairs
(a) Steps in starting an I/O device and getting interrupt
(b) How the CPU is interrupted
Structure of a large Pentium system
• A process tree
  – A created two child processes, B and C
  – B created three child processes, D, E, and F
Operating System Concepts (2)

(a) A potential deadlock. (b) an actual deadlock.
File system for a university department
• **Before mounting,**
  - files on floppy are inaccessible

• **After mounting floppy on b,**
  - files on floppy are part of file hierarchy
Two processes connected by a pipe
Steps in Making a System Call

There are 11 steps in making the system call `read (fd, buffer, nbytes)`
Some System Calls For Process Management

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pid = fork()</code></td>
<td>Create a child process identical to the parent</td>
</tr>
<tr>
<td><code>pid = waitpid(pid, &amp;statloc, options)</code></td>
<td>Wait for a child to terminate</td>
</tr>
<tr>
<td><code>s = execve(name, argv, environp)</code></td>
<td>Replace a process’ core image</td>
</tr>
<tr>
<td><code>exit(status)</code></td>
<td>Terminate process execution and return status</td>
</tr>
</tbody>
</table>
Some System Calls For File Management

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fd = open(file, how, ...)</code></td>
<td>Open a file for reading, writing or both</td>
</tr>
<tr>
<td><code>s = close(fd)</code></td>
<td>Close an open file</td>
</tr>
<tr>
<td><code>n = read(fd, buffer, nbytes)</code></td>
<td>Read data from a file into a buffer</td>
</tr>
<tr>
<td><code>n = write(fd, buffer, nbytes)</code></td>
<td>Write data from a buffer into a file</td>
</tr>
<tr>
<td><code>position = lseek(fd, offset, whence)</code></td>
<td>Move the file pointer</td>
</tr>
<tr>
<td><code>s = stat(name, &amp;buf)</code></td>
<td>Get a file’s status information</td>
</tr>
</tbody>
</table>
### Some System Calls For Directory Management

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s = mkdir(name, mode)</code></td>
<td>Create a new directory</td>
</tr>
<tr>
<td><code>s = rmdir(name)</code></td>
<td>Remove an empty directory</td>
</tr>
<tr>
<td><code>s = link(name1, name2)</code></td>
<td>Create a new entry, name2, pointing to name1</td>
</tr>
<tr>
<td><code>s = unlink(name)</code></td>
<td>Remove a directory entry</td>
</tr>
<tr>
<td><code>s = mount(special, name, flag)</code></td>
<td>Mount a file system</td>
</tr>
<tr>
<td><code>s = umount(special)</code></td>
<td>Unmount a file system</td>
</tr>
</tbody>
</table>
### Some System Calls For Miscellaneous Tasks

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s = chdir(dirname)</code></td>
<td>Change the working directory</td>
</tr>
<tr>
<td><code>s = chmod(name, mode)</code></td>
<td>Change a file’s protection bits</td>
</tr>
<tr>
<td><code>s = kill(pid, signal)</code></td>
<td>Send a signal to a process</td>
</tr>
<tr>
<td><code>seconds = time(&amp;seconds)</code></td>
<td>Get the elapsed time since Jan. 1, 1970</td>
</tr>
</tbody>
</table>
System Calls (1)

- A stripped down shell:

```c
while (TRUE) {
    type_prompt( );
    read_command (command, parameters)
    /* repeat forever */
    /* display prompt */
    /* input from terminal */

    if (fork() != 0) {
        /* Parent code */
        waitpid( -1, &status, 0);
        /* fork off child process */
        /* wait for child to exit */
    } else {
        /* Child code */
        execve (command, parameters, 0);
        /* execute command */
    }
}
```
System Calls (2)

- Processes have three segments: text, data, stack
System Calls (3)

(a) Two directories before linking /usr/jim/memo to ast's directory

(b) The same directories after linking
System Calls (4)

(a) File system before the mount
(b) File system after the mount
## System Calls (5)

<table>
<thead>
<tr>
<th>UNIX</th>
<th>Win32</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fork</td>
<td>CreateProcess</td>
<td>Create a new process</td>
</tr>
<tr>
<td>waitpid</td>
<td>WaitForSingleObject</td>
<td>Can wait for a process to exit</td>
</tr>
<tr>
<td>execve</td>
<td>(none)</td>
<td>CreateProcess = fork + execve</td>
</tr>
<tr>
<td>exit</td>
<td>ExitProcess</td>
<td>Terminate execution</td>
</tr>
<tr>
<td>open</td>
<td>CreateFile</td>
<td>Create a file or open an existing file</td>
</tr>
<tr>
<td>close</td>
<td>CloseHandle</td>
<td>Close a file</td>
</tr>
<tr>
<td>read</td>
<td>ReadFile</td>
<td>Read data from a file</td>
</tr>
<tr>
<td>write</td>
<td>WriteFile</td>
<td>Write data to a file</td>
</tr>
<tr>
<td>lseek</td>
<td>SetFilePointer</td>
<td>Move the file pointer</td>
</tr>
<tr>
<td>stat</td>
<td>GetFileAttributesEx</td>
<td>Get various file attributes</td>
</tr>
<tr>
<td>mkdir</td>
<td>CreateDirectory</td>
<td>Create a new directory</td>
</tr>
<tr>
<td>rmdir</td>
<td>RemoveDirectory</td>
<td>Remove an empty directory</td>
</tr>
<tr>
<td>link</td>
<td>(none)</td>
<td>Win32 does not support links</td>
</tr>
<tr>
<td>unlink</td>
<td>DeleteFile</td>
<td>Destroy an existing file</td>
</tr>
<tr>
<td>mount</td>
<td>(none)</td>
<td>Win32 does not support mount</td>
</tr>
<tr>
<td>umount</td>
<td>(none)</td>
<td>Win32 does not support mount</td>
</tr>
<tr>
<td>chdir</td>
<td>SetCurrentDirectory</td>
<td>Change the current working directory</td>
</tr>
<tr>
<td>chmod</td>
<td>(none)</td>
<td>Win32 does not support security (although NT does)</td>
</tr>
<tr>
<td>kill</td>
<td>(none)</td>
<td>Win32 does not support signals</td>
</tr>
<tr>
<td>time</td>
<td>GetLocalTime</td>
<td>Get the current time</td>
</tr>
</tbody>
</table>

### Some Win32 API calls
Embedded OS

• What is an embedded Operating System
• Who are the players
• Linux as an embedded OS
• Tools and development
• Applications and products
• The embedded OS market
• Embedded OS Resources
OS Flavors

• **Desktop**
  - Windows (9X, XP Home, XP/2000 Pro)
  - Mac

• **Server**
  - Windows (XP/2000 Server & Advanced Server)
  - Unix Varieties

• **Embedded**
  - Many
What is an Embedded OS?

- An "embedded system" is any computer system or computing device that performs a dedicated function or is designed for use with a specific embedded software application.

- Embedded systems may use a ROM-based operating system or they may use a disk-based system, like a PC. But an embedded system is not usable as a commercially viable substitute for general purpose computers or devices.
What makes a good Embedded OS?

- Modular
- Scalable
- Configurable
- Small footprint
- CPU support
- Device drivers
- etc, etc, etc...
What is Real Time?

“A real time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred.”

- Donald Gillies
What is Real Time?

“Real time in operating systems:
The ability of the operating system to provide a required level of service in a bounded response time.”

- POSIX Standard 1003.1
Hard vs. Soft Real Time

- **Hard**
  - *guaranteed worst-case* response times
  - absolutely, positively, first time every time

- **Soft**
  - Kinda, sorta, usually
What makes a good RTOS?

- Multi-threaded and pre-emptible
- Thread priority has to exist because no deadline driven OS exists
- Must support predictable thread synchronization mechanisms
- A system of priority inheritance must exist
Who are the Embedded OS players?

• **Wind River Systems**
  – VxWorks
  – pSOS

• **QNX Software Systems**
  – QNX

• **Green Hills Software**
  – Integrity
Who are the Embedded OS players?

- **Mentor Graphics**
  - VRTX
- **Palm Computing**
  - PalmOS
- **Symbian**
  - SymbianOS
Microsoft

• Embedded NT/XP
  – “Real-time” control
• Windows CE (CE.NET)
  – Internet devices
• Pocket PC 2002
  – Handheld PC’s and PDA’s
Commercial Embedded Linux

- AMIRIX Embedded Linux
  - derived from Debian
- Coollogic Coollinux
  - combines Linux and Java for Internet apps
- Coventive Xlinux
  - kernel can be as small as 143KB
- Esfia RedBlue Linux
  - 400K, designed for wireless apps
Commercial Embedded Linux

- **KYZO Pizza Box Linux**
  - SAMBA based file, print, CD server
- **Lineo Embedix**
  - supports real time and high availability apps
- **LynuxWorks BlueCat**
  - general purpose embedded solution
- **MontaVista Linux**
  - general purpose embedded solution
Commercial Embedded Linux

- **Neoware NeoLinux**
  - Red Hat derived for information appliances
- **PalmPalm Tynux**
  - Internet appliance and multimedia
- **Red Hat Embedded Linux**
  - general purpose embedded solution
- **RedSonic Red-Ice Linux**
  - runs from DiskonChip or CompactFlash
Commercial Embedded Linux

- RidgeRun DSP Linux
  - for multimedia, wireless, RT on DSP
- TimeSys Linux GPL
  - low latency enhanced kernel
- Tuxia TASTE
  - distro targets Internet appliances
- Vital Systems vLinux
  - for ARM based embedded apps
Open Source Embedded Linux

- **Embedded Debian Project**
  - convert Debian to an embedded OS
- **ETLinux**
  - for PC104 SBC’s
- **uCLinux**
  - for microprocessors that don’t have MM
- **uLinux (muLinux)**
  - distro fits on a single floppy
Commercial Linux RTOS

- FSMLabs - Open RT Linux
- Lineo - Embedix Realtime
- LynuxWorks - BlueCat RT
- MontaVista Software - Real Time Extensions
- REDSonic - REDICE Linux
- TimeSys - Linux/Real-Time
Open Source Linux RTOS

• ART Linux - real time extension based on RTLinux
• KURT - event schedules with 10us resolution
• Linux-SRT - for soft real time apps like multimedia
• Qlinux - provides Quality of Service guarantees
• RTAI - “hard” Real Time Application Interface
RTLinux

- A “hard real-time” mini operating system
- runs Linux as it’s lowest priority execution thread
- Linux thread completely preemptible
- Real time threads and interrupt handlers never delayed by non-realtime operations
- Supports user level programming
- MiniRTL implementation fits on a floppy
What’s so special about Linux?

• Multiple choices vs. sole source
• Source code freely available
• Robust and reliable
• Modular, configurable, scalable
• Superb support for networking and Internet
• No runtime licenses
• Large pool of skilled developers
What’s so special about Linux?

What are your main reasons for wanting to use Linux in embedded applications?

- Source code is available (and free)
- No runtime royalties
- Linux is more robust/reliable
- Linux has excellent networking support
- There are more drivers and tools available
- Lots of programmers are familiar with Linux
- It’s not from Microsoft
- Other

What’s special about Open Source?

What do you value most about using open source software in embedded applications?

- Collaborative open source development produces superior s/w
- It allows fully understanding what’s going on inside the OS
- It eliminates dependence on a single OS vendor
- It represents “insurance”, even if it’s never needed
- It facilitates debugging and troubleshooting the application
- It lets me add functionality directly within the OS
- It lets me immediately fix OS bugs, in case they arise
- None -- I don't need or want OS source!
- Other

Flies in the Ointment

- Lack of hardware device drivers
- Competing and/or lacking standards
- No formalized qualification testing
- No single source for marketing
- GPL license issues
- Startup vendors with shaky futures
What CPU’s will it run on?

- Intel X86
- MIPS
- ARM
- StrongARM
- PowerPC
- Hitachi SuperH
Projected Target CPU’s
Single Board Computers

- **Little Board** (5.75 x 8.0 in.) -- complete systems on a single compact board, expandable with plug-on function modules

- **ISA "slot boards"** (full-length, 13.8 x 4.8 in.; half-length, 7.1 x 4.8 in.) -- IBM PC plug-in cards which could function as standalone SBCs backplanes

- **PC/104 modules** (3.6 x 3.8 in.) -- compact, rugged, self-stacking modules featuring a reliable pin-and-socket board-to-board expansion bus
Single Board Computers with PCI

- **PC/104-Plus** -- PCI added to PC/104

- **EBX** -- PC/104-Plus added to Little Board
Target Devices

- Custom, non-PC architecture: 36%
- Custom, PC architecture: 15%
- Other off-the-shelf SBC: 4%
- VME: 6%
- Passive backplane: 6%
- PC/104, PC/104-Plus, EBX: 24%
- CompactPCI: 2%
- Desktop-PC hardware: 11%

Copyright © 2001, CNET Networks, Inc.
Any Development Tools Available?

- QT/Embedded
- Other GUI/Windowing toolkits
- Arcom Control Systems
- GNUPro Tools
- Vendor specific
- Standard Linux toolset
C or C++ for Development?

• **In general C is a better choice**
  – Advanced OOP features can cause code bloat
  – C++ compilers can generate many routines for a single function
  – Virtual methods and polymorphism slow program launch times significantly

• **Size really *does* matter**
What’s It Being Used For?

- Control and Monitoring Applications
- Industrial Controllers
- TV Set Top Boxes (TiVO)
- Handheld PDA’s
- Automobile Computers
- Telecomm and Networking Hardware
- Myriad and sundry other uses...
Target Applications

- Internet appliance or thin server: 26%
- Telecom or telephony: 13%
- Set-top box: 6%
- Aerospace or Military: 11%
- Vending machine / POS: 2%
- Other: 4%
- Industrial data acq / control: 31%

Gaming or entertainment: 7%
Cyclades TS-100

- Netlinos OS
- Dual 50MHz PowerPC
- 16MB SDRAM 4MB Flash
- Ethernet/Serial/RS485
- Size of a deck of cards
Sixnet VersaTRAK IPm

- Embedded Linux
- PowerPC CPU
- 16MB DRAM 16MB Flash
- Serial & Ethernet Ports
- Linux and IEC1131 Programming/Modbus
Sharp Zaurus

- Lineo Embedix
- 206 MHz StrongARM
- 64 MB DRAM 16MB Flash
- 3.5” display (320x240) 64K colors
- Opera browser & Qtopia
- QT/Embedded GUI
Cell and Web Phones

Telepong Mobile Phone

GITWiT Mobile Phone

Aplio/PRO IP Phone
TiVO Set Top Box

- Home grown port of Embedded Linux
- 54MHz PowerPC
- Multi GB hard disk
Axis 2120 Network Camera

- uCLinux
- Built-in Ethernet port
- 100 MHz ETRAX CPU
- 16 MB RAM
Humanoid Robots

• Univ. of Tokyo/Kawanda Ind.
• Dual Pentium CPU
• RT-Linux
• Height: 53 inches
• Weight: 121 lbs.

• Isamu
Humanoid Robots

- Fujitsu
- RT-Linux
- Height: 48 cm
- Weight: 6 kg
- 100 units/yr

• HOAP
The Embedded OS Market 2001

Embedded OS trends 2001–2002, sorted by 2001 usage

(multiple selections permitted; top 10 for 2001 shown)

Source: Evans Data Corporation 2001 Embedded Systems Developer Survey
The Embedded OS Market 2002

(multiple selections permitted; top 10 for 2002 shown)

Source: Evans Data Corporation 2001 Embedded Systems Developer Survey
Growth of Embedded Linux

Worldwide Shipments of Embedded Linux OSes, Software Development Tools, and Related Services
(in millions of dollars)

Source: Venture Development Corporation (VDC) 2000 Embedded Linux Market Study

(C) 2001, CNET Networks, Inc.)
For Further Info
Web Sites

- LinuxDevices.com
  http://www.linuxdevices.com/
- Embedded Linux Consortium
  http://www.embedded-linux.org/index.php3
- All Linux Devices
  http://alllinuxdevices.com/
- Embedded Linux StarGate
  http://linux-embedded.com/
For Further Info
Web Sites/Magazines

• Dr. Dobbs Embedded Systems
  http://www.ddjembedded.com/about/

• Embedded Linux Journal
  http://embedded.linuxjournal.com/

• Embedded Systems Programming
  http://www.embedded.com/mag.htm
References

Embedded Operating Systems and Linux
Bill Latura, April 23, 2002