```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd" []>

<book id="LinuxKernelAPI">
 <bookinfo>
  <title>The Linux Kernel API</title>

  <legalnotice>
   <para>
     This documentation is free software; you can redistribute
     it and/or modify it under the terms of the GNU General Public
     License as published by the Free Software Foundation; either
     version 2 of the License, or (at your option) any later
     version.
   </para>

   <para>
     This program is distributed in the hope that it will be
     useful, but WITHOUT ANY WARRANTY; without even the implied
     warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
     See the GNU General Public License for more details.
   </para>

   <para>
     You should have received a copy of the GNU General Public
     License along with this program; if not, write to the Free
     Software Foundation, Inc., 59 Temple Place, Suite 330, Boston,
     MA 02111-1307 USA
   </para>

   <para>
     For more details see the file COPYING in the source
     distribution of Linux.
   </para>
  </legalnotice>
 </bookinfo>

<toc></toc>

  <chapter id="adt">
     <title>Data Types</title>
     <sect1><title>Doubly Linked Lists</title>
!Iinclude/linux/list.h
     </sect1>
  </chapter>

  <chapter id="libc">
     <title>Basic C Library Functions</title>

     <para>
       When writing drivers, you cannot in general use routines which are
       from the C Library.  Some of the functions have been found generally
       useful and they are listed below.  The behaviour of these functions
       may vary slightly from those defined by ANSI, and these deviations
       are noted in the text.
```

```
     </para>

     <sect1><title>String Conversions</title>
!Ilib/vsprintf.c
!Elib/vsprintf.c
     </sect1>
     <sect1><title>String Manipulation</title>
<!-- All functions are exported at now
X!Ilib/string.c
 -->
!Elib/string.c
     </sect1>
     <sect1><title>Bit Operations</title>
!Iarch/x86/include/asm/bitops.h
     </sect1>
  </chapter>

  <chapter id="kernel-lib">
     <title>Basic Kernel Library Functions</title>

     <para>
       The Linux kernel provides more basic utility functions.
     </para>

     <sect1><title>Bitmap Operations</title>
!Elib/bitmap.c
!Ilib/bitmap.c
     </sect1>

     <sect1><title>Command-line Parsing</title>
!Elib/cmdline.c
     </sect1>

     <sect1 id="crc"><title>CRC Functions</title>
!Elib/crc7.c
!Elib/crc16.c
!Elib/crc-itu-t.c
!Elib/crc32.c
!Elib/crc-ccitt.c
     </sect1>
  </chapter>

  <chapter id="mm">
     <title>Memory Management in Linux</title>
     <sect1><title>The Slab Cache</title>
!Iinclude/linux/slab.h
!Emm/slab.c
     </sect1>
     <sect1><title>User Space Memory Access</title>
!Iarch/x86/include/asm/uaccess_32.h
!Earch/x86/lib/usercopy_32.c
     </sect1>
     <sect1><title>More Memory Management Functions</title>
!Emm/readahead.c
!Emm/filemap.c
!Emm/memory.c
```

```
!Emm/vmalloc.c
!Imm/page_alloc.c
!Emm/mempool.c
!Emm/dmapool.c
!Emm/page-writeback.c
!Emm/truncate.c
     </sect1>
  </chapter>


  <chapter id="ipc">
     <title>Kernel IPC facilities</title>

     <sect1><title>IPC utilities</title>
!Iipc/util.c
     </sect1>
  </chapter>

  <chapter id="kfifo">
     <title>FIFO Buffer</title>
     <sect1><title>kfifo interface</title>
!Iinclude/linux/kfifo.h
!Ekernel/kfifo.c
     </sect1>
  </chapter>

  <chapter id="relayfs">
     <title>relay interface support</title>

     <para>
        Relay interface support
        is designed to provide an efficient mechanism for tools and
        facilities to relay large amounts of data from kernel space to
        user space.
     </para>

     <sect1><title>relay interface</title>
!Ekernel/relay.c
!Ikernel/relay.c
     </sect1>
  </chapter>

  <chapter id="modload">
     <title>Module Support</title>
     <sect1><title>Module Loading</title>
!Ekernel/kmod.c
     </sect1>
     <sect1><title>Inter Module support</title>
        <para>
           Refer to the file kernel/module.c for more information.
        </para>
<!-- FIXME: Removed for now since no structured comments in source
X!Ekernel/module.c
-->
     </sect1>
  </chapter>
```

```
  <chapter id="hardware">
     <title>Hardware Interfaces</title>
     <sect1><title>Interrupt Handling</title>
!Ekernel/irq/manage.c
     </sect1>

     <sect1><title>DMA Channels</title>
!Ekernel/dma.c
     </sect1>

     <sect1><title>Resources Management</title>
!Ikernel/resource.c
!Ekernel/resource.c
     </sect1>

     <sect1><title>MTRR Handling</title>
!Earch/x86/kernel/cpu/mtrr/main.c
     </sect1>

     <sect1><title>PCI Support Library</title>
!Edrivers/pci/pci.c
!Edrivers/pci/pci-driver.c
!Edrivers/pci/remove.c
!Edrivers/pci/search.c
!Edrivers/pci/msi.c
!Edrivers/pci/bus.c
!Edrivers/pci/access.c
!Edrivers/pci/irq.c
!Edrivers/pci/htirq.c
<!-- FIXME: Removed for now since no structured comments in source
X!Edrivers/pci/hotplug.c
-->
!Edrivers/pci/probe.c
!Edrivers/pci/slot.c
!Edrivers/pci/rom.c
!Edrivers/pci/iov.c
!Idrivers/pci/pci-sysfs.c
     </sect1>
     <sect1><title>PCI Hotplug Support Library</title>
!Edrivers/pci/hotplug/pci_hotplug_core.c
     </sect1>
     <sect1><title>MCA Architecture</title>
        <sect2><title>MCA Device Functions</title>
           <para>
              Refer to the file arch/x86/kernel/mca_32.c for more information.
           </para>
<!-- FIXME: Removed for now since no structured comments in source
X!Earch/x86/kernel/mca_32.c
-->
        </sect2>
        <sect2><title>MCA Bus DMA</title>
!Iarch/x86/include/asm/mca_dma.h
        </sect2>
     </sect1>
  </chapter>
```

```
  <chapter id="firmware">
     <title>Firmware Interfaces</title>
     <sect1><title>DMI Interfaces</title>
!Edrivers/firmware/dmi_scan.c
     </sect1>
     <sect1><title>EDD Interfaces</title>
!Idrivers/firmware/edd.c
     </sect1>
  </chapter>

  <chapter id="security">
     <title>Security Framework</title>
!Isecurity/security.c
!Esecurity/inode.c
  </chapter>

  <chapter id="audit">
     <title>Audit Interfaces</title>
!Ekernel/audit.c
!Ikernel/auditsc.c
!Ikernel/auditfilter.c
  </chapter>

  <chapter id="accounting">
     <title>Accounting Framework</title>
!Ikernel/acct.c
  </chapter>

  <chapter id="blkdev">
     <title>Block Devices</title>
!Eblock/blk-core.c
!Iblock/blk-core.c
!Eblock/blk-map.c
!Iblock/blk-sysfs.c
!Eblock/blk-settings.c
!Eblock/blk-exec.c
!Eblock/blk-barrier.c
!Eblock/blk-tag.c
!Iblock/blk-tag.c
!Eblock/blk-integrity.c
!Ikernel/trace/blktrace.c
!Iblock/genhd.c
!Eblock/genhd.c
  </chapter>

  <chapter id="chrdev">
        <title>Char devices</title>
!Efs/char_dev.c
  </chapter>

  <chapter id="miscdev">
     <title>Miscellaneous Devices</title>
!Edrivers/char/misc.c
  </chapter>
```

```
  <chapter id="clk">
    <title>Clock Framework</title>

    <para>
      The clock framework defines programming interfaces to support
      software management of the system clock tree.
      This framework is widely used with System-On-Chip (SOC) platforms
      to support power management and various devices which may need
      custom clock rates.
      Note that these "clocks" don't relate to timekeeping or real
      time clocks (RTCs), each of which have separate frameworks.
      These <structname>struct clk</structname> instances may be used
      to manage for example a 96 MHz signal that is used to shift bits
      into and out of peripherals or busses, or otherwise trigger
      synchronous state machine transitions in system hardware.
    </para>

    <para>
      Power management is supported by explicit software clock gating:
      unused clocks are disabled, so the system doesn't waste power
      changing the state of transistors that aren't in active use.
      On some systems this may be backed by hardware clock gating,
      where clocks are gated without being disabled in software.
      Sections of chips that are powered but not clocked may be able
      to retain their last state.
      This low power state is often called a <emphasis>retention
      mode</emphasis>.
      This mode still incurs leakage currents, especially with finer
      circuit geometries, but for CMOS circuits power is mostly used
      by clocked state changes.
    </para>

    <para>
      Power-aware drivers only enable their clocks when the device
      they manage is in active use.  Also, system sleep states often
      differ according to which clock domains are active:  while a
      "standby" state may allow wakeup from several active domains, a
      "mem" (suspend-to-RAM) state may require a more wholesale shutdown
      of clocks derived from higher speed PLLs and oscillators, limiting
      the number of possible wakeup event sources.  A driver's suspend
      method may need to be aware of system-specific clock constraints
      on the target sleep state.
    </para>

    <para>
      Some platforms support programmable clock generators.  These
      can be used by external chips of various kinds, such as other
      CPUs, multimedia codecs, and devices with strict requirements
      for interface clocking.
    </para>

!Iinclude/linux/clk.h
  </chapter>

</book>
```