Mesleki İngilizce - Technical English

II

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr

http://www.yildiz.edu.tr/~naydin

- Notes:
  - In the slides,
    - texts enclosed by curly parenthesis, {…}, are examples.
    - texts enclosed by square parenthesis, […], are explanations related to examples.

Anatomy of the Linux Kernel

- Learning Objectives
  - to acquire basic vocabulary related to operating systems
  - to understand the basics of Linux kernel architecture
- Sub-areas covered
  - Linux kernel and its subsystems

Anatomy of the Linux Kernel

- Keywords
  - kernel
    - the central component of most computer operating systems (OS).
    - Its functions include managing the system's resources
      - the communication between hardware and software components
  - Linux kernel
    - Unix-like operating system kernel
  - VFS (Virtual file system)
    - an abstraction layer on top of a more concrete file system

Anatomy of the Linux Kernel

- Keywords
  - GNU
    - a computer operating system composed entirely of free software,
      - initiated in 1984 by Richard Stallman
      - [GNU is a recursive acronym for "**GNU's Not Unix**!", chosen because GNU's design is Unix-like, but differs from Unix by being free software and containing no Unix code. The GNU project includes an operating system kernel, GNU HURD, which was the original focus of the Free Software Foundation (FSF).]
  - GPL
    - a widely used free software license
      - originally written by Richard Stallman for the GNU project

Anatomy of the Linux Kernel

- Keywords
  - Minix
    - free/open source, Unix-like operating system (OS) based on a microkernel architecture
  - Unix
    - a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs including Ken Thompson, Dennis Ritchie and Douglas Ilroy
  - operating system
    - the software that
      - manages the sharing of the resources of a computer
      - provides programmers with an interface used to access those resources

Anatomy of the Linux Kernel

- Keywords
  - buffer
    - a region of memory used to temporarily hold data while it is being moved from one place to another
  - buffer cache
    - a collection of data duplicating original values stored elsewhere or computed earlier,
      - where the original data is expensive to fetch (owing to longer access time) or to compute, compared to the cost of reading the cache

7

Anatomy of the Linux Kernel

- Reading text
- Pre-reading questions
  - What are the most popular operating systems?
  - What are the advantages of Linux?
  - What are the disadvantages of Linux?

8

Anatomy of the Linux Kernel

- The Linux® kernel
  - the core of a large and complex operating system
    - over six million lines of code
  - well organized in terms of subsystems and layers.
  - a monolithic Unix-like computer OS kernel.
  - The Linux family of OSs is based on this kernel
- Developer
  - Linus Torvalds and thousands of collaborators
- Written in
  - C and assembly
- Latest release
  - 4.13.4 (28 September 2017)
- Initial release
  - 0.01 (17 September 1991)

9

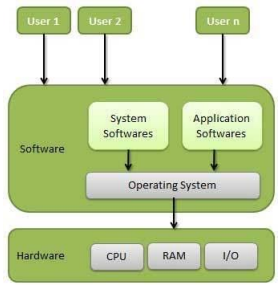
What is an Operating System

- You need two types of software in order to use your computer
  - applications
    - the programs you use to do tasks, such as write a document, surf the web, or play games
  - system software
    - runs the computer system for you,
      - an operating system
- There are many different operating systems,
  - but they all have a similar architecture (or structure).
  - That is because they must all overcome the same problems and perform the same basic functions.

10

What is an Operating System

- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



11

What is an Operating System

- An operating system must be able to:
  - Manage system resources
    - CPU scheduling
    - Process management
    - Memory management
    - Input/Output device management
    - Storage device management (hard disks, CD/DVD drives, etc)
    - File System Management
  - Simplify the development and use of applications

12

What is an Operating System

- Other Important Activities that an OS performs
  - Security
    - By means of password and similar other techniques, it prevents unauthorized access to programs and data.
  - Control over system performance
    - Recording delays between request for a service and response from the system.
  - Job accounting
    - Keeping track of time and resources used by various jobs and users.
  - Error detecting aids
    - Production of dumps, traces, error messages, and other debugging and error detecting aids.
  - Coordination between other softwares and users
    - Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

13

What is an Operating System

- Types of OSs
  - Batch operating system
  - Time-sharing operating systems
  - Distributed operating System
  - Network operating System
  - Real Time operating System
    - Hard real-time systems
    - Soft real-time systems
- Some popular Operating Systems
  - Unix, Linux, Windows, DOS, OS X, VMS, OS/400, AIX, z/OS, etc.

14

Linux history

- Linux is arguably the most popular open source operating system
  - Its history is actually quite short considering the timeline of OSs
- In the early days of computing, programmers developed on the bare hardware in the hardware's language.
  - The lack of an operating system meant that only one application (and one user) could use the large and expensive device at a time.

15

Linux history

- Early operating systems
  - developed in the 1950s to provide a simpler development experience.
- Examples include
  - the General Motors Operating System (GMOS)
    - developed for the IBM 701
  - the FORTRAN Monitor System (FMS)
    - developed by North American Aviation for the IBM 709.

16

Linux history

- In the 1960s, the MIT and a host of companies developed an experimental operating system called Multics for the GE-645
  - [Multics: Multiplexed Information and Computing Service]
- In 1970, one of the developers of this operating system, AT&T, left Multics and developed their own operating system called Unics
  - [Unics: Uniplexed Information and Computing Service]
  - Along with this operating system was the C language,
    - for which C was developed and then rewritten to make operating system development portable.

17

Linux history

- Twenty years later, Andrew Tanenbaum created a microkernel version of UNIX®,
  - called MINIX (for minimal UNIX), that ran on small personal computers.
- This open source operating system inspired Linus Torvalds' initial development of Linux in the early 1990s.
- Linux quickly evolved from a single-person project to a world-wide development project involving thousands of developers.

18

Linux history

- One of the most important decisions for Linux was its adoption of the GNU General Public License (GPL).
  - Under the GPL, the Linux kernel was protected from commercial exploitation,
  - It also benefited from the user-space development of the GNU project.
    - This allowed useful applications such as the GNU Compiler Collection (GCC) and various shell support.

Linux history

- Linux has its roots in a student project. In 1992, an undergraduate called Linus Torvalds was studying computer science in Helsinki, Finland. Like most computer science courses, a big component of it was taught on (and about) Unix. Unix was the wonder operating system of the 1970s and 1980s: both a textbook example of the principles of operating system design, and sufficiently robust to be the standard OS in engineering and scientific computing. But Unix was a commercial product (licensed by AT&T to a number of resellers), and cost more than a student could pay.
- Annoyed by the shortcomings of Minix (a compact Unix clone written as a teaching aid by Professor Andy Tannenbaum) Linus set out to write his own 'kernel' — the core of an operating system that handles memory allocation, talks to hardware devices, and makes sure everything keeps running.
- He used the GNU programming tools developed by Richard Stallman's Free Software Foundation, an organisation of volunteers dedicated to fulfilling Stallman's ideal of making good software that anyone could use without paying. When he'd written a basic kernel, he released the source code to the Linux kernel on the Internet.
- Source code is important. It's the original from which compiled programs are generated. If you don't have the source code to a program, you can't modify it to fix bugs or add new features. Most software companies won't sell you their source code, or will only do so for an eyewatering price, because they believe that if they make it available it will destroy their revenue stream.

Linux history

- What happened next was astounding, from the conventional, commercial software industry point of view — and utterly predictable to anyone who knew about the Free Software Foundation. Programmers (mostly academics and students) began using Linux. They found that it didn't do things they wanted it to do -so they fixed it. And where they improved it, they sent the improvements to Linus, who rolled them into the kernel. And Linux began to grow.
- There's a term for this model of software development; it's called Open Source (see www. opensource. org/ for more information). Anyone can have the source code - it's free (in the sense of free speech, not free beer). Anyone can contribute to it. If you use it heavily you may want to extend or develop or fix bugs in it — and it is so easy to give your fixes back to the community that most people do so.
- An operating system kernel on its own isn't a lot of use; but Linux was purposefully designed as a near-clone of Unix, and there is a lot of software out there that is free and was designed to compile on Linux. By about 1992, the first 'distributions' appeared.
- A distribution is the Linux-user term for a complete operating system kit, complete with the utilities and applications you need to make it do useful things — command interpreters, programming tools, text editors, typesetting tools, and graphical user interfaces based on the X windowing system. X is a standard in academic and scientific computing, but not hitherto common on PCs; it's a complex distributed windowing system on which people implement graphical interfaces like KDE and Gnome.
- As more and more people got to know about Linux, some of them began to port the Linux kernel to run on non-standard computers. Because it's free, Linux is now the most widelyported operating system there is.

Introduction to the Linux kernel

- GNU/Linux OS architecture can be thought as an operating system from two levels:
- At the top is the user (or application) space.
  - This is where the user applications are executed.
- Below the user space is the kernel space
  - Here, the Linux kernel exists.
- There is also the GNU C Library (glibc), providing
  - the system call interface that connects to the kernel
  - the mechanism to transition between the user-space application and the kernel.
    - This is important because the kernel and user application occupy different protected address spaces.
      - While each user-space process occupies its own virtual address space, the kernel occupies a single address space.

Properties of the Linux kernel

- The Linux kernel can be further divided into three gross levels:
  - At the top is the system call interface (SCI),
    - which implements the basic functions such as read and write.
  - Below the system call interface is the kernel code,
    - which can be more accurately defined as the architecture-independent kernel code.
    - This code is common to all of the processor architectures supported by Linux.
  - Below this is the architecture-dependent code,
    - which forms what is more commonly called a BSP (Board Support Package).
    - This code serves as the processor and platform-specific code for the given architecture.

Properties of the Linux kernel

- The Linux kernel implements a number of important architectural attributes.
- At a high level, and at lower levels, the kernel is layered into a number of distinct subsystems.
- Linux can also be considered monolithic because it lumps all of the basic services into the kernel.
  - This differs from a microkernel architecture, where the kernel provides basic services such as communication, I/O, memory and process management, and more specific services are plugged in to the microkernel layer.
- Each has its own advantages.

Properties of the Linux kernel

- Over time, the Linux kernel has become efficient in terms of both memory and CPU usage, as well as extremely stable.
- But the most interesting aspect of Linux, given its size and complexity, is its portability.
  - Linux can be compiled to run on different processors and platforms with different architectural constraints and needs.
    - {One example is the ability of Linux to run on a process with a memory management unit (MMU), as well as those that provide no MMU.
    - The uClinux port of the Linux kernel provides for non-MMU support.}

25

Major subsystems of the Linux kernel

- **System call interface (SCI)**
  - a thin layer that provides the means to perform function calls from user space into the kernel.
  - can be architecture dependent,
    - even within the same processor family.
  - The SCI is actually an interesting function-call multiplexing and demultiplexing service.
  - You can find the SCI implementation in ./linux/kernel, as well as architecture-dependent portions in ./linux/arch.

26

Major subsystems of the Linux kernel

- **Process management**
  - focused on the execution of processes.
  - In the kernel, these are called threads and represent an individual virtualization of the processor (thread code, data, stack, and CPU registers).
  - In user space, the term process is typically used, though the Linux implementation does not separate the two concepts.
  - The kernel provides an application program interface (API) through the SCI
    - to create a new process (fork, exec, or Portable Operating System Interface [POSIX] functions),
    - to stop a process (kill, exit),
    - to communicate and synchronize between them (signal, or POSIX mechanisms).

27

Major subsystems of the Linux kernel

  - Also in process management there is a need to share the CPU between the active threads.
  - The kernel implements a novel scheduling algorithm that operates in constant time,
    - regardless of the number of threads vying for the CPU.
  - This is called the O(1) scheduler, denoting that the same amount of time is taken to schedule one thread as it is to schedule many.
  - The O(1) scheduler also supports multiple processors (called Symmetric MultiProcessing, or SMP).
  - You can find the process management sources in ./linux/kernel and architecture-dependent sources in ./linux/arch).

28

Major subsystems of the Linux kernel

- **Memory management**
  - Another important resource that is managed by the kernel is memory.
  - For efficiency, given the way that the hardware manages virtual memory, memory is managed in what are called pages,
    - 4KB in size for most architectures.
  - Linux includes the means to manage the available memory, as well as the hardware mechanisms for physical and virtual mappings.

29

Major subsystems of the Linux kernel

  - But memory management is much more than managing 4KB buffers.
  - Linux provides abstractions over 4KB buffers, such as the slab allocator.
    - [Slab allocation: a memory management mechanism intended for the efficient memory]
  - This memory management scheme uses 4KB buffers as its base, but then allocates structures from within, keeping track of which pages are full, partially used, and empty.
  - This allows the scheme to dynamically grow and shrink based on the needs of the greater system.

30

Major subsystems of the Linux kernel

- In supporting multiple users of memory, there are times when the available memory can be exhausted.
- For this reason, pages can be moved out of memory and onto the disk.
- This process is called swapping because the pages are swapped from memory onto the hard disk.
- You can find the memory management sources in ./linux/mm..

31

Major subsystems of the Linux kernel

- **Virtual file system (VFS)**
  - provides a common interface abstraction for file systems.
  - provides a switching layer between the SCI and the file systems supported by the kernel.
  - At the top of the VFS is a common API abstraction of functions such as open, close, read, and write.
  - At the bottom of the VFS are the file system abstractions that define how the upper-layer functions are implemented.
  - These are plug-ins for the given file system (of which over 50 exist).
  - You can find the file system sources in ./linux/fs.

32

Major subsystems of the Linux kernel

- Below the file system layer is the buffer cache,
  - which provides a common set of functions to the file system layer (independent of any particular file system).
- This caching layer optimizes access to the physical devices by keeping data around for a short time
  - or speculatively read ahead so that the data is available when needed.
- Below the buffer cache are the device drivers,
  - which implement the interface for the particular physical device.

33

Major subsystems of the Linux kernel

- **Network stack**
  - The network stack, by design, follows a layered architecture modeled after the protocols themselves.
    - Recall that the Internet Protocol (IP) is the core network layer protocol that sits below the transport protocol (most commonly the Transmission Control Protocol, or TCP).
  - Above TCP is the sockets layer, which is invoked through the SCI.
    - The sockets layer is the standard API to the networking subsystem and provides a user interface to a variety of networking protocols.
    - From raw frame access to IP protocol data units (PDUs) and up to TCP and the User Datagram Protocol (UDP), the sockets layer provides a standardized way to manage connections and move data between endpoints.
  - You can find the networking sources in the kernel at ./linux/net.

34

Major subsystems of the Linux kernel

- **Device drivers**
  - The vast majority of the source code in the Linux kernel exists in device drivers that make a particular hardware device usable.
  - The Linux source tree provides a drivers subdirectory that is further divided by the various devices that are supported, such as
    - Bluetooth, I2C, serial, and so on.
  - You can find the device driver sources in ./linux/drivers.

35

Major subsystems of the Linux kernel

- **Architecture-dependent code**
  - While much of Linux is independent of the architecture on which it runs, there are elements that must consider the architecture for normal operation and for efficiency.
  - The /linux/arch subdirectory defines the architecture-dependent portion of the kernel source contained in a number of subdirectories that are specific to the architecture (collectively forming the BSP).
  - For a typical desktop, the i386 directory is used.
  - Each architecture subdirectory contains a number of other subdirectories that focus on a particular aspect of the kernel, such as boot, kernel, memory management, and others.
  - You can find the architecture-dependent code in ./linux/arch.

36

Interesting features of the Linux kernel

- Linux, being a production operating system and open source, is a great test bed for new protocols and advancements of those protocols.
- Linux supports a large number of networking protocols,
  - including the typical TCP/IP, and also extension for high-speed networking (greater than 1 Gigabit Ethernet [GbE] and 10 GbE).
- Linux also supports protocols such as the Stream Control Transmission Protocol (SCTP),
  - which provides many advanced features above TCP (as a replacement transport level protocol).

Interesting features of the Linux kernel

- Linux is also a dynamic kernel, supporting the addition and removal of software components on the fly.
  - These are called dynamically loadable kernel modules
  - They can be inserted at boot when they're needed (when a particular device is found requiring the module) or at any time by the user.
- A recent advancement of Linux is its use as an operating system for other operating systems (called a hypervisor).
- Recently, a modification to the kernel was made called the Kernel-based Virtual Machine (KVM).
  - This modification enabled a new interface to user space that allows other operating systems to run above the KVM-enabled kernel.
- In addition to running another instance of Linux, Microsoft® Windows® can also be virtualized.
  - The only constraint is that the underlying processor must support the new virtualization instructions.

Some resouces

- The GNU site (http://www.gnu.org/licenses) describes the GNU GPL that covers the Linux kernel and most useful applications provided with it. Also described is a less restrictive form of the GPL called the Lesser GPL (LGPL).
- UNIX (http://en.wikipedia.org/wiki/Unics), MINIX (http://en.wikipedia.org/wiki/Minix) and Linux (http://en.wikipedia.org/wiki/Linux) are covered in Wikipedia, along with a detailed family tree of the operating systems.
- The GNU C Library (http://www.gnu.org/software/libc/), or glibc, is the implementation of the standard C library. It's used in the GNU/Linux operating system, as well as the GNU/Hurd (http://directory.fsf.org/hurd.html) microkernel operating system.

Some resouces

- uClinux (http://www.uclinux.org/) is a port of the Linux kernel that can execute on systems that lack an MMU. This allows the Linux kernel to run on very small embedded platforms, such as the Motorola DragonBall processor used in the PalmPilot Personal Digital Assistants (PDAs).
- "Kernel command using Linux system calls" (http://www.ibm.com/developerworks/linux/library/l-system-calls/) (developerWorks, March 2007) covers the SCI, which is an important layer in the Linux kernel, with user-space support from glibc that enables function calls between user space and the kernel.
- "Inside the Linux scheduler" (http://www.ibm.com/developerworks/linux/library/lscheduler/) (developerWorks, June 2006) explores the new O(1) scheduler introduced in Linux 2.6 that is efficient, scales with a large number of processes (threads), and takes advantage of SMP systems.

Some resouces

- "Access the Linux kernel using the /proc filesystem" (http://www.ibm.com/developerworks/linux/library/l-proc.html) (developerWorks, March 2006) looks at the /proc file system, which is a virtual file system that provides a novel way for userspace applications to communicate with the kernel. This article demonstrates /proc, as well as loadable kernel modules.
- "Server clinic: Put virtual filesystems to work" (http://www.ibm.com/developerworks/linux/library/l-sc12.html) (developerWorks, April 2003) delves into the VFS layer that allows Linux to support a variety of different file systems through a common interface. This same interface is also used for other types of devices, such as sockets.

Some resouces

- "Inside the Linux boot process" (http://www.ibm.com/developerworks/linux/library/l-linuxboot/index.html) (developerWorks, May 2006) examines the Linux boot process, which takes care of bringing up a Linux system and is the same basic process whether you're booting from a hard disk, floppy, USB memory stick, or over the network.
- "Linux initial RAM disk (initrd) overview" (http://www.ibm.com/developerworks/linux/library/l-initrd.html) (developerWorks, July 2006) inspects the initial RAM disk, which isolates the boot process from the physical medium from which it's booting.
- "Better networking with SCTP" (http://www.ibm.com/developerworks/linux/library/l-sctp/) (developerWorks, February 2006) covers one of the most interesting networking protocols, Stream Control Transmission Protocol, which operates like TCP but adds a number of useful features such as messaging, multi-homing, and multi-streaming. Linux, like BSD, is a great operating system if you're interested in networking protocols.

Some resouces

- "Anatomy of the Linux slab allocator" (http://www.ibm.com/developerworks/linux/library/l-linux-slab-allocator/) (developerWorks, May 2007) covers one of the most interesting aspects of memory management in Linux, the slab allocator. This mechanism originated in SunOS, but it's found a friendly home inside the Linux kernel.
- "Virtual Linux" (http://www.ibm.com/developerworks/linux/library/l-linuxvirt/) (developerWorks, December 2006) shows how Linux can take advantage of processors with virtualization capabilities.
- "Linux and symmetric multiprocessing" (http://www.ibm.com/developerworks/library/l-linux-smp/) (developerWorks, March 2007) discusses how Linux can also take advantage of processors that offer chip-level multiprocessing.

Some resouces

- "Discover the Linux Kernel Virtual Machine" (http://www.ibm.com/developerworks/linux/library/l-linux-kvm/) (developerWorks, April 2007) covers the recent introduction of virtualization into the kernel, which turns the Linux kernel into a hypervisor for other virtualized operating systems.
- Check out Tim's book GNU/Linux Application Programming (http://www.charlesriver.com/Books/BookDetail.aspx?productID=91525) for more information on programming Linux in user space.

Some resouces

- In the developerWorks Linux zone (http://www.ibm.com/developerworks/linux/), find more resources for Linux developers, including Linux tutorials (http://www.ibm.com/developerworks/views/linux/libraryview.jsp?type_by=Tutorials), as well as our readers' favorite Linux articles and tutorials (http://www.ibm.com/developerworks/linux/library/l-top-10.html) over the last month.
- Stay current with developerWorks technical events and Webcasts (http://www.ibm.com/developerworks/offers/techbriefings/?S_TACT=105AGX03&S_CMP=art).

Some resouces

- In the developerWorks Linux zone (http://www.ibm.com/developerworks/linux/), find more resources for Linux developers, including Linux tutorials (http://www.ibm.com/developerworks/views/linux/libraryview.jsp?type_by=Tutorials), as well as our readers' favorite Linux articles and tutorials (http://www.ibm.com/developerworks/linux/library/l-top-10.html) over the last month.
- Stay current with developerWorks technical events and Webcasts (http://www.ibm.com/developerworks/offers/techbriefings/?S_TACT=105AGX03&S_CMP=art).

Grammar revision

- *-ing* form: as noun and after prepositions
- We can use the -ing form of the verb as a noun.
- It can be the subject, object, or complement of a sentence.
- For example:
  - {*Managing* the computer's resources is an important function of the operating system.}
  - {The operating system starts *running* the user interface as soon as the PC is switched on.}
  - {Another function of the operating system is *executing* and *providing* services for applications software.}

Grammar revision

- The *-ing* form is also used after prepositions.
- This includes *to* when it is a preposition and not part of the infinitive.
- For example:
  - {*Without* the user *being* aware of the details, the operating system manages the computer's resources.}
  - {We begin *by focusing* on the interaction between a user and a PC operating system.}
  - {We look forward to *having* cheaper and faster computers.}