Linux Operating System

History

- Linux is a modern, free operating system based on UNIX standards.
- First developed as a small but self-contained kernel in 1991 by Linus Torvalds, with the major design goal of UNIX compatibility.
- Its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet.
- It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms.
- The core Linux operating-system kernel is entirely original, but it can run much existing free UNIX software, resulting in an entire UNIX-compatible operating system free from proprietary code.

The Linux Kernel

- Version 0.02 (May 1991) had no networking, ran only on 80386-compatible Intel processors and on PC hardware, had extremely limited device-driver support, and supported only the Minix file system.
- Linux 1.0 (March 1994) included these new features:
  - Support for UNIX's standard TCP/IP networking protocols
  - BSD-compatible socket interface for networking programming
  - Device-driver support for running IP over an ethernet
  - Enhanced file system
  - Support for a range of SCSI controllers for high-performance disk access
- Version 1.2 (March 1995) was the final PC-only Linux kernel.

Linux 2.0

- Released in June 1996, 2.0 added two major new capabilities:
  - Support for multiple architectures
  - Support for multiprocessor architectures
- Other new features included:
  - Improved memory-management code
  - Improved TCP/IP performance
  - Support for internal kernel threads, for handling dependencies between loadable modules, and for automatic loading of modules on demand.
  - Standardized configuration interface
- Available for 68000-series, Sun Sparc, and PowerMac systems.

The Linux System

- Linux uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project.
- The main system libraries were started by the GNU project, with improvements provided by the Linux community.
- Linux networking-administration tools were derived from 4.3BSD code; recent BSD derivatives such as FreeBSD have borrowed code from Linux in return.
- The Linux system is maintained by a loose network of developers collaborating over the Internet, with a small number of public ftp sites acting as de facto standard repositories.

Linux Distributions

- Standard, precompiled sets of packages, or *distributions*, include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools.
- The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management.
- The RPM package file format permits compatibility among the various Linux distributions.

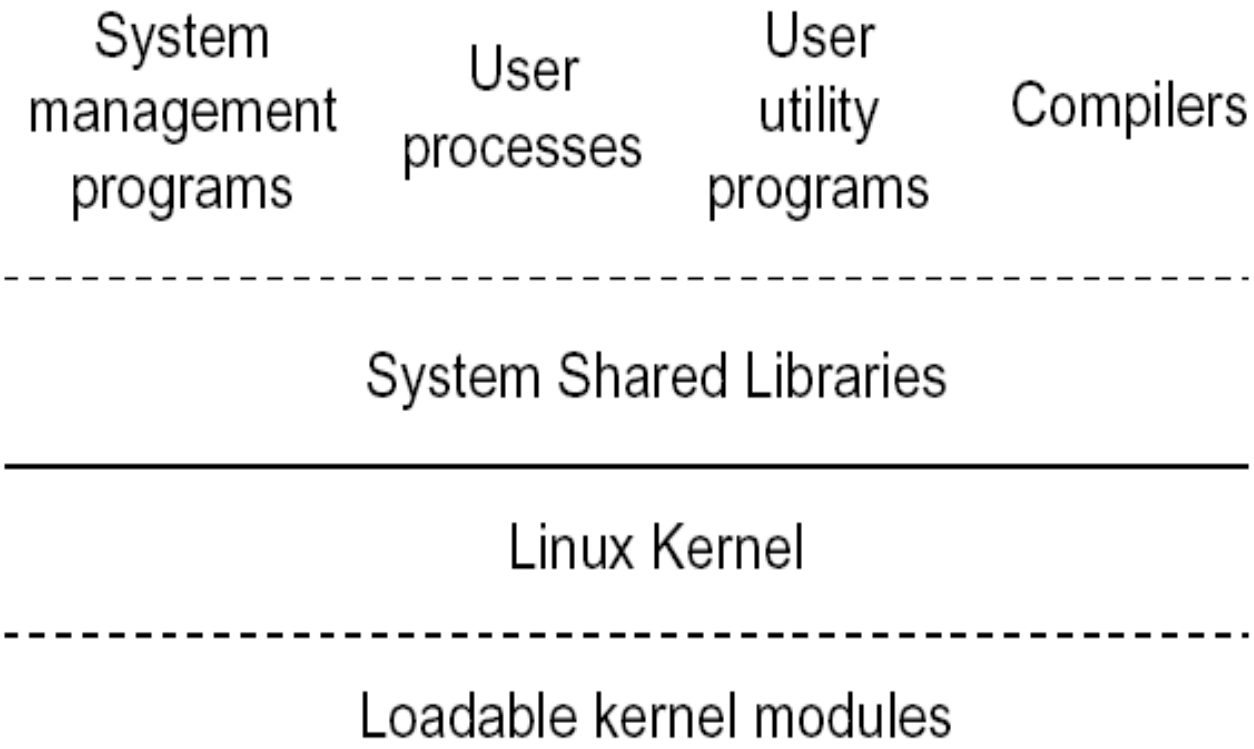
Linux Licensing

- The Linux kernel is distributed under the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation.
- Anyone using Linux, or creating their own derivate of Linux, may not make the derived product proprietary; software released under the GPL may not be redistributed as a binary-only product.

Design Principles

- Linux is a multiuser, multitasking system with a full set of UNIX-compatible tools.
- Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model.
- Main design goals are speed, efficiency, and standardization.
- Linux is designed to be compliant with the relevant POSIX documents
- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior.

Components of a Linux System



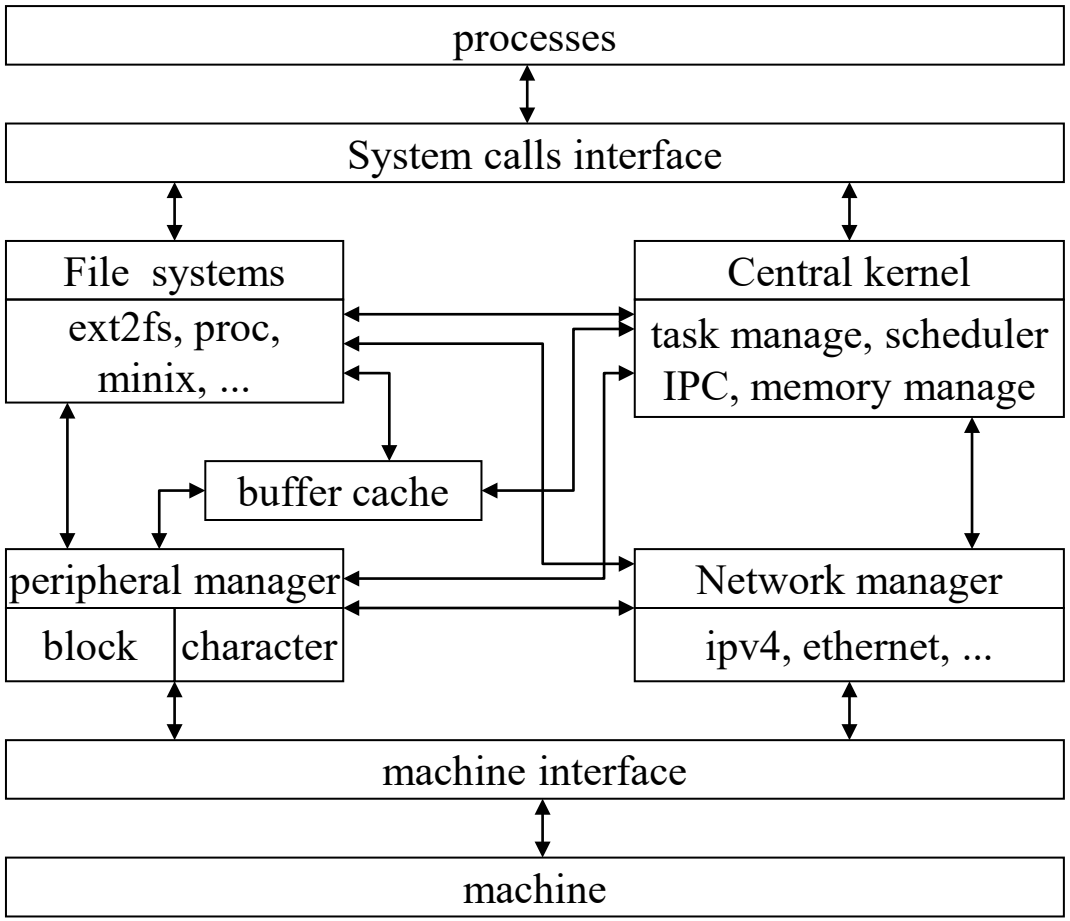
Linux Components (1)

- Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction is between the kernel and all other components.
- The *kernel* is responsible for maintaining the important abstractions of the operating system.
  - Kernel code executes in *kernel mode* with full access to all the physical resources of the computer.
  - All kernel code and data structures are kept in the same single address space.

Linux Components (2)

- The *system libraries* define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code.
- The *system utilities* perform individual specialized management tasks.

System Structure



Linux Kernel Components

- Process Management
- Memory & Virtual memory
- File system
- Inter-process Communication
- Network

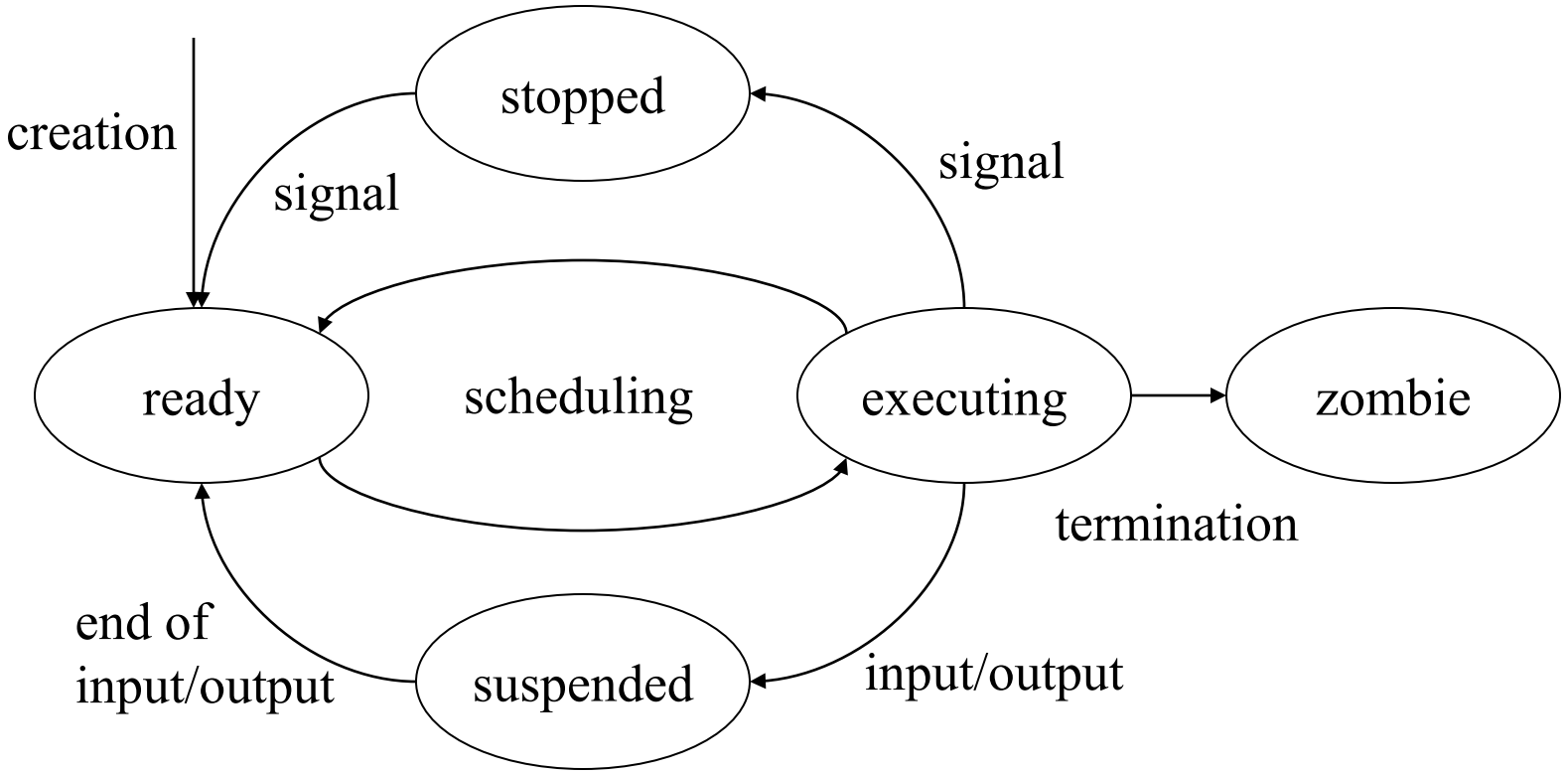
Linux Process

- A process can be considered to be a program being run.
- At any given time a single instruction is carried out within the process.
- Consists of at least three part, called region or segment:
  - Text
  - Data
  - Stack
- Every process has an unique identifier called *pid*.
- Program is a passive entity, but process is an *active* entity.
- The process table entry and U-area contains control and status information about process.

Attributes of a Process

- State
- Identification (unique number)
- Values of the registers, including the program counter
- User identity: under whose name the process is executing
- Information used by the kernel to establish the schedule of the processes:
  - Priority, execution time, ...
- Information concerning the address space of the process:
  - Segments for the code, data, stack
- Information concerning the inputs/outputs carried out by the process:
  - Descriptions of open files, current directory, ...
- Compatibility information summarizing resources used by the process

Process States



Process Management

- Fork/Exec process model
  - UNIX process management separates the creation of processes and the running of a new program into two distinct operations.
    - The fork system call creates a new process.
    - A new program is run after a call to execve.
- Under UNIX, a process encompasses all the information that the operating system must maintain to track the context of a single execution of a single program.
- Under Linux, process properties fall into three groups: the process's identity, environment, and context.

Fork/Exec Process Model

```
#include <stdio.h>
void main(int argc, char *argv[])
{
   int pid;
   pid = fork();
   if (pid < 0) { /* error occurred */
            printf(stderr, "Fork Failed"); exit(-
   1);
   } else if (pid==0) { /* child process */
            execlp("/bin/ls", "ls", NULL);
   } else { /* parent process */
            wait(NULL);
            printf("child Completed");
            exit(0);
   }
}
```