Linux Internals & Networking

System programming using Kernel interfaces

Team Emertxe

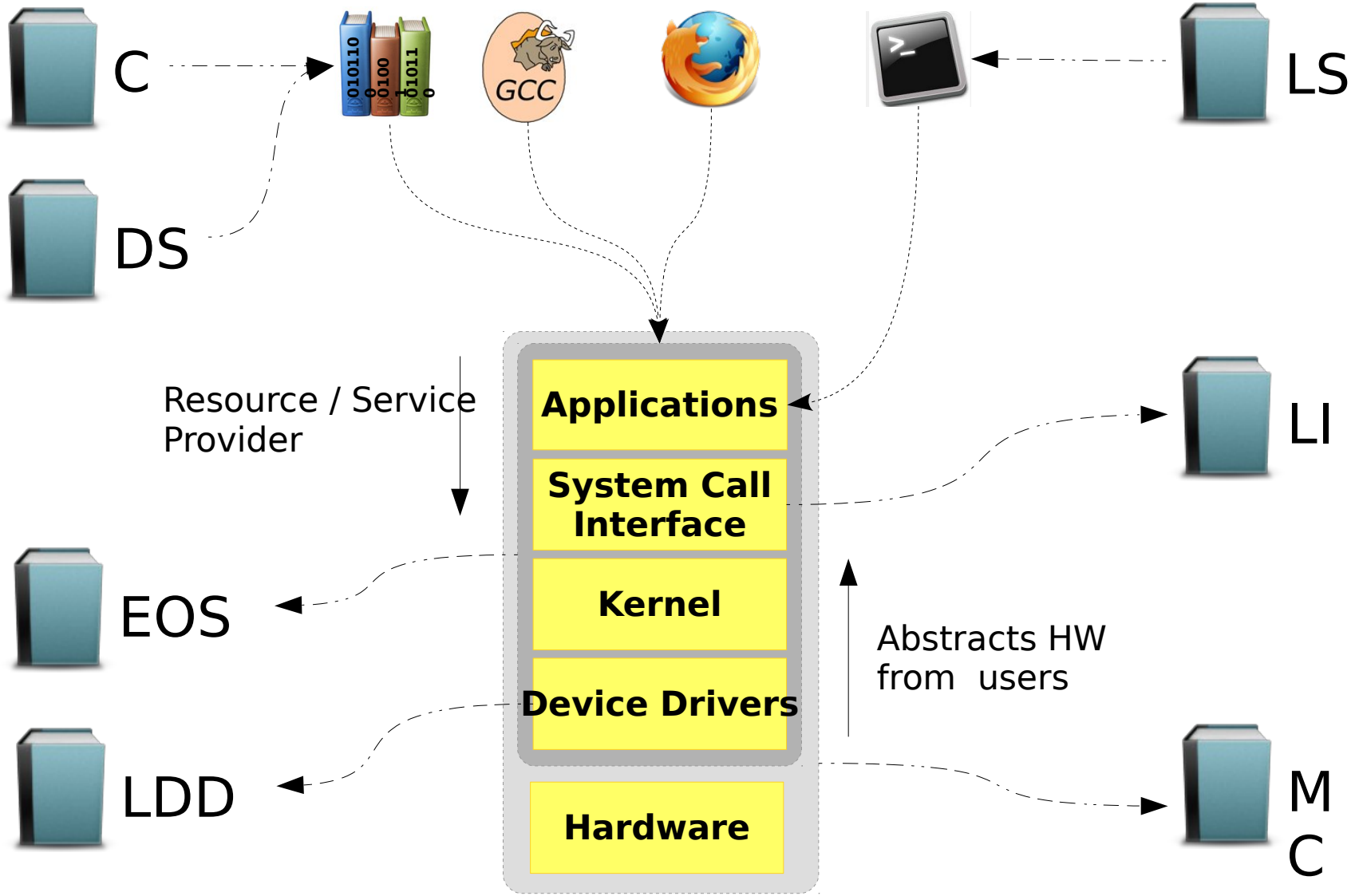
Contents

Linux Internals & Networking
Contents

- Introduction
- Transition to OS programmer
- System Calls
- Process
- IPC
- Signals
- Networking
- Threads
- Synchronization
- Process Management
- Memory Management

Transition to OS programming

Course & modules view



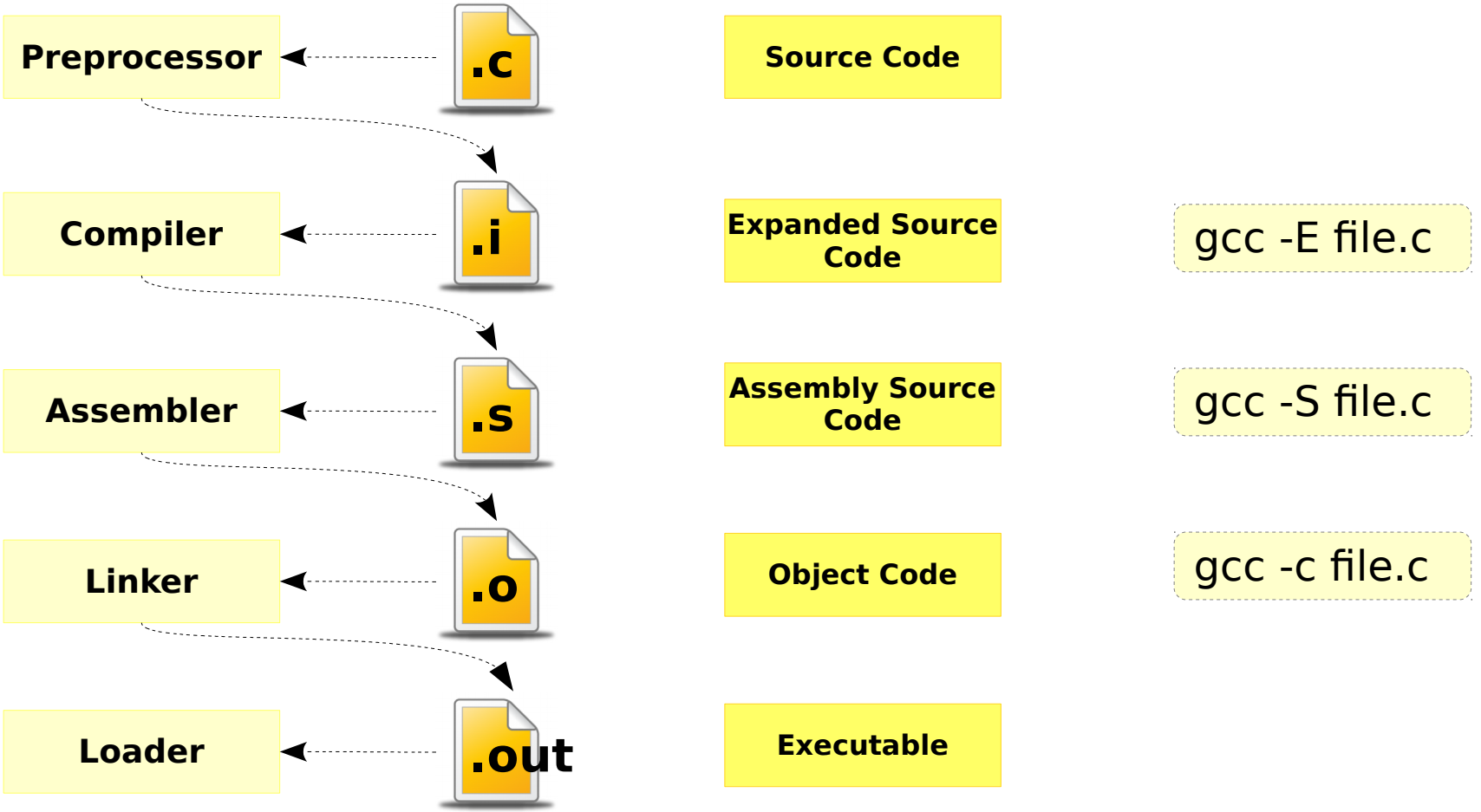
Application **vs OS**

C	Algorithms, Syntax, Logic	• Preprocessor • Compiler • Assembler • Linker • Executable file (a.out)
OS	Memory segments, process, Threads Signals, IPC, Networking	• Executing a program • Loader

Application Programming
Compilation Stages

- **Preprocessor**
  - Expands header files
  - Substitute all macros
  - Remove all comments
  - Expands and all # directives
- **Compilation**
  - Convert to assembly level instructions
- **Assembly**
  - Convert to machine level instructions
  - Commonly called as object files
  - Create logical address
- **Linking**
  - Linking with libraries and other object files

Application Programming
Compilation Stages



gcc -save-temps file.c would generate all intermediate files

EMERTXE

Application Programming
Linking - Static

- Static linking is the process of copying all library modules used in the program into the final executable image.
- This is performed by the linker and it is done as the last step of the compilation process.
- Compiling two .o files also a type of static linking.
- To create a static library first create intermediate object files.
  - Eg: gcc -c fun1.c fun2.c
- Creates two object files fun1.o and fun2.o
- Then create a library by archive command
  - Eg: ar rcs libfun.a fun1.o fun2.o

Application Programming
Linking - Dynamic

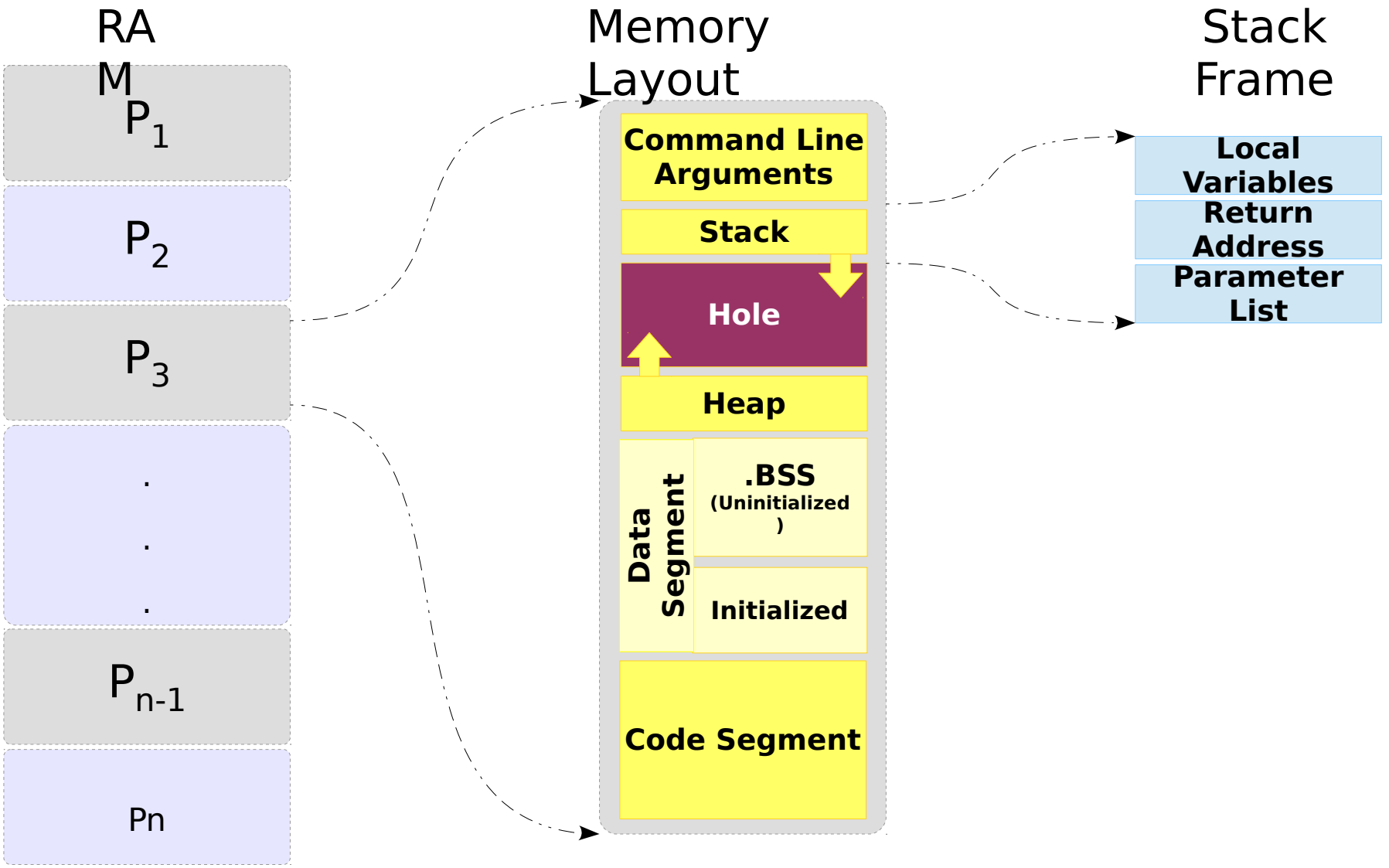
- It performs the linking process when programs are executed in the system.
- During dynamic linking the name of the shared library is placed in the final executable file.
- Actual linking takes place at run time when both executable file and library are placed in the memory.
- The main advantage to using dynamically linked libraries is that the size of executable programs is reduced
- To create a dynamic library (shared object file)
  - Eg: gcc -fPIC -shared fun1.c fun2.c -o libfun.so

Application Programming

Linking - Static vs Dynamic

Parameter \ Linking	Static	Dynamic
Executable Size		
Loading Time		
Memory Usage		
No of Sys. Calls		

Executing a process



Quiz

- How a user defined function works?
- How a library function works?

Storage Classes

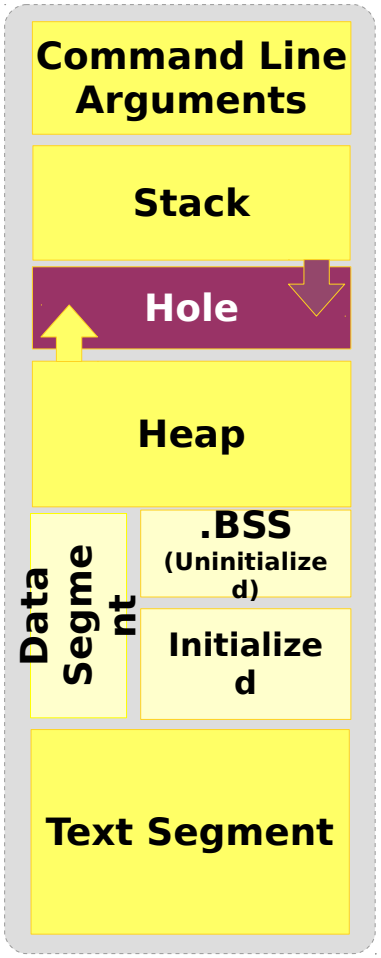
Storage Class	Scope	Lifetime	Memory Allocation
auto	Within the block / Function	Till the end of the block / function	Stack
register	Within the block / Function	Till the end of the block / function	Register
static local	Within the block / Function	Till the end of the program	Data Segment
static global	File	Till the end of the program	Data segment
extern	Program	Till the end of the program	Data segment

Hands-on

- Access a static variable from outside file.
- Access a global variable from outside file.
- Combination of both static and local.

Common Errors

With various memory segments



Stack Overflow / Stack Smashing

- When ever process stack limit is over
  Eg: Call a recursive function infinite times.
- When you trying to access array beyond limits.
  Eg int arr[5]; arr[100];

Memory Leak

- When you never free memory after allocating.
  Eventually process heap memory will run-out

Segmentation Fault

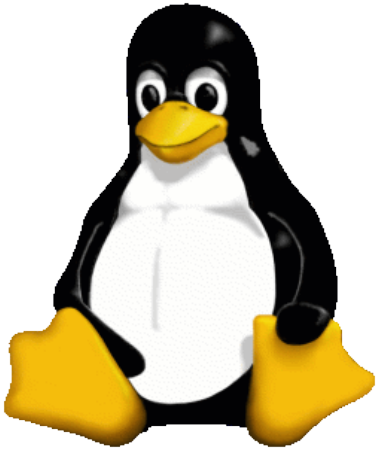
- When you try to change text segment, which is a read-only memory or try trying to access a memory beyond process memory limit (like NULL pointer)

Introduction
What is Linux?

- Linux is a free and open source operating system that is causing a revolution in the computer world

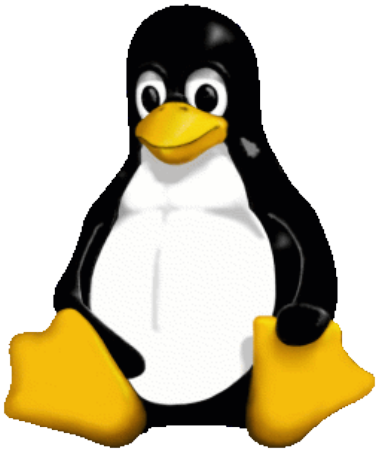
- Originally created by Linus Torvalds with the assistance of developers called community

- This operating system in only a few short years is beginning to dominate markets worldwide

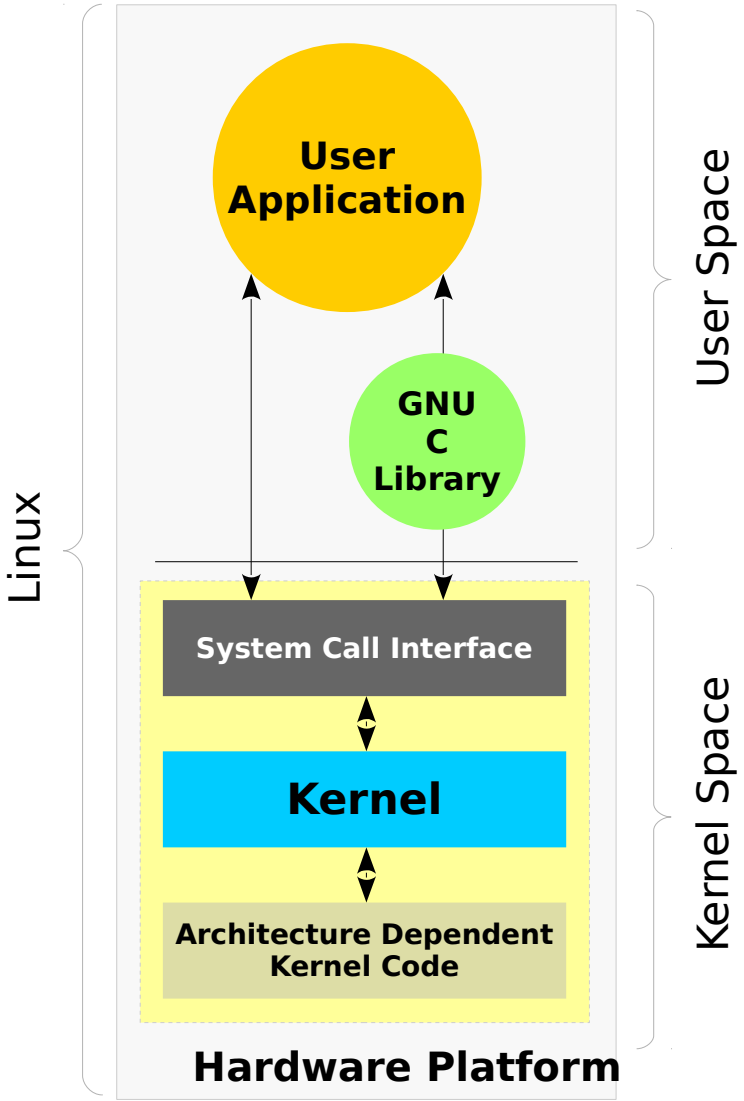


Introduction
What is Linux?

- Free & Open Source
  - GPL license, no cost
- Reliability
  - Build systems with 99.999% upstream
- Secure
  - Monolithic kernel offering high security
- Scalability
  - From mobile phone to stock market servers

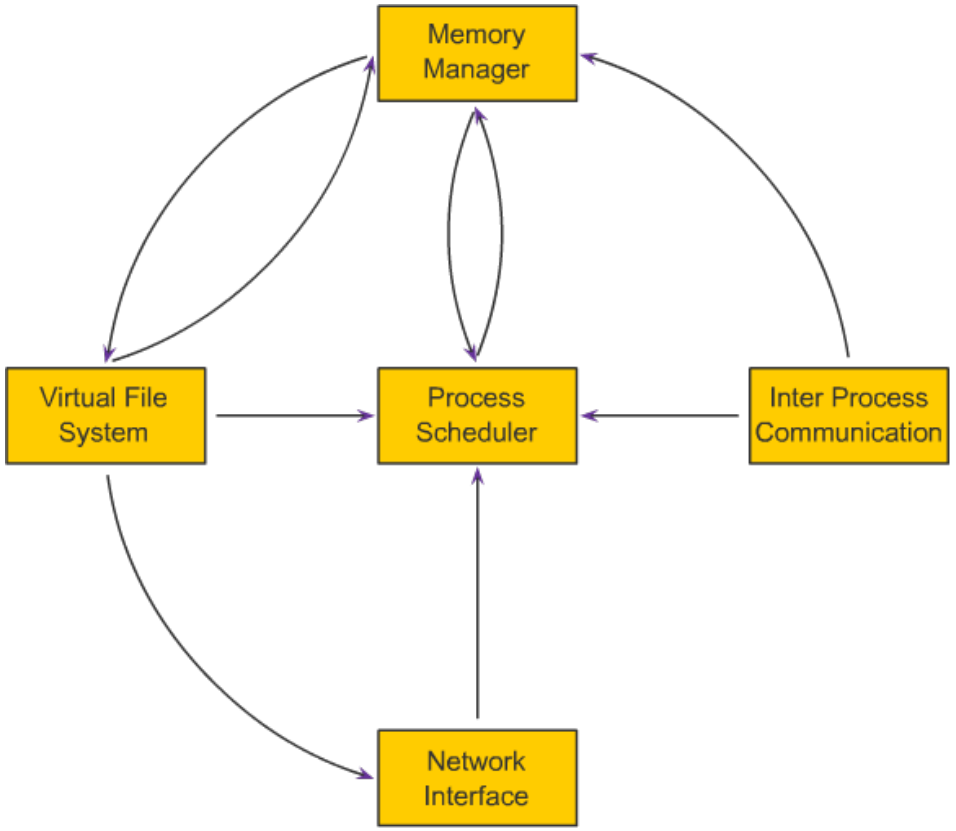


Introduction
Linux Components



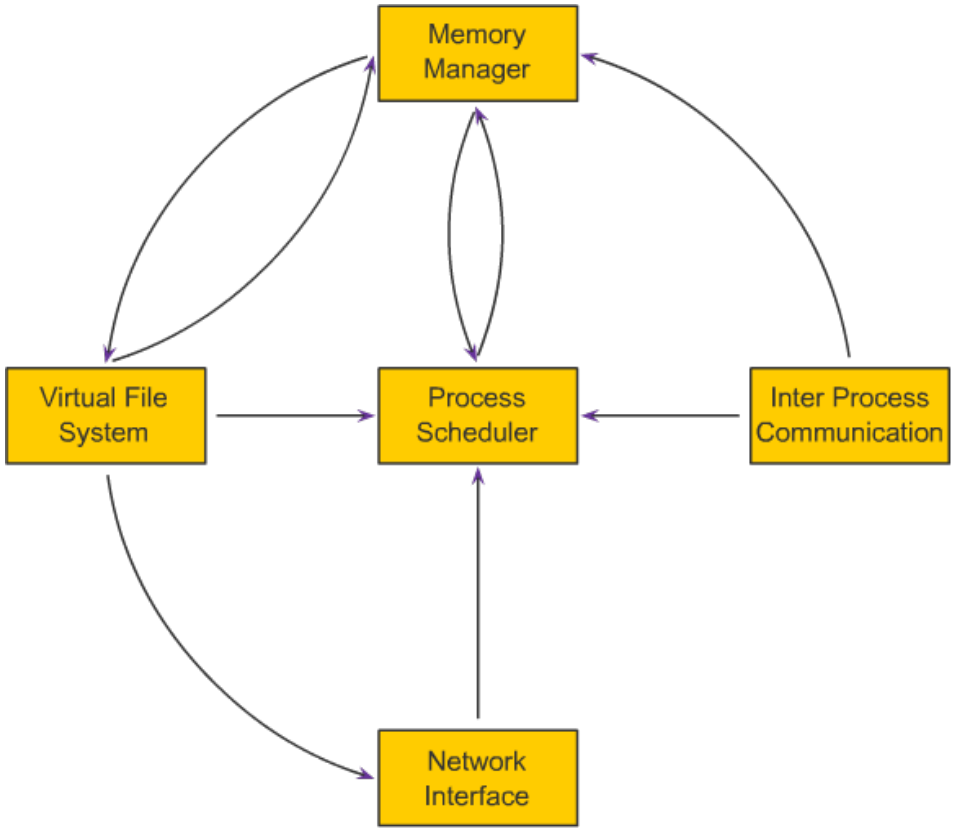
- **Hardware Controllers:** This subsystem is comprised of all the possible physical devices in a Linux installation - CPU, memory hardware, hard disks
- **Linux Kernel:** The kernel abstracts and mediates access to the hardware resources, including the CPU. A kernel is the core of the operating system
- **O/S Services:** These are services that are typically considered part of the operating system (e.g. windowing system, command shell)
- **User Applications:** The set of applications in use on a particular Linux system (e.g. web browser)

Introduction
Linux Kernel Subsystem



- Process Scheduler (SCHED):
  - To provide control, fair access of CPU to process, while interacting with HW on time
- Memory Manager (MM):
  - To access system memory securely and efficiently by multiple processes.
  - Supports Virtual Memory in case of huge memory requirement
- Virtual File System (VFS):
  - Abstracts the details of the variety of hardware devices by presenting a common file interface to all devices

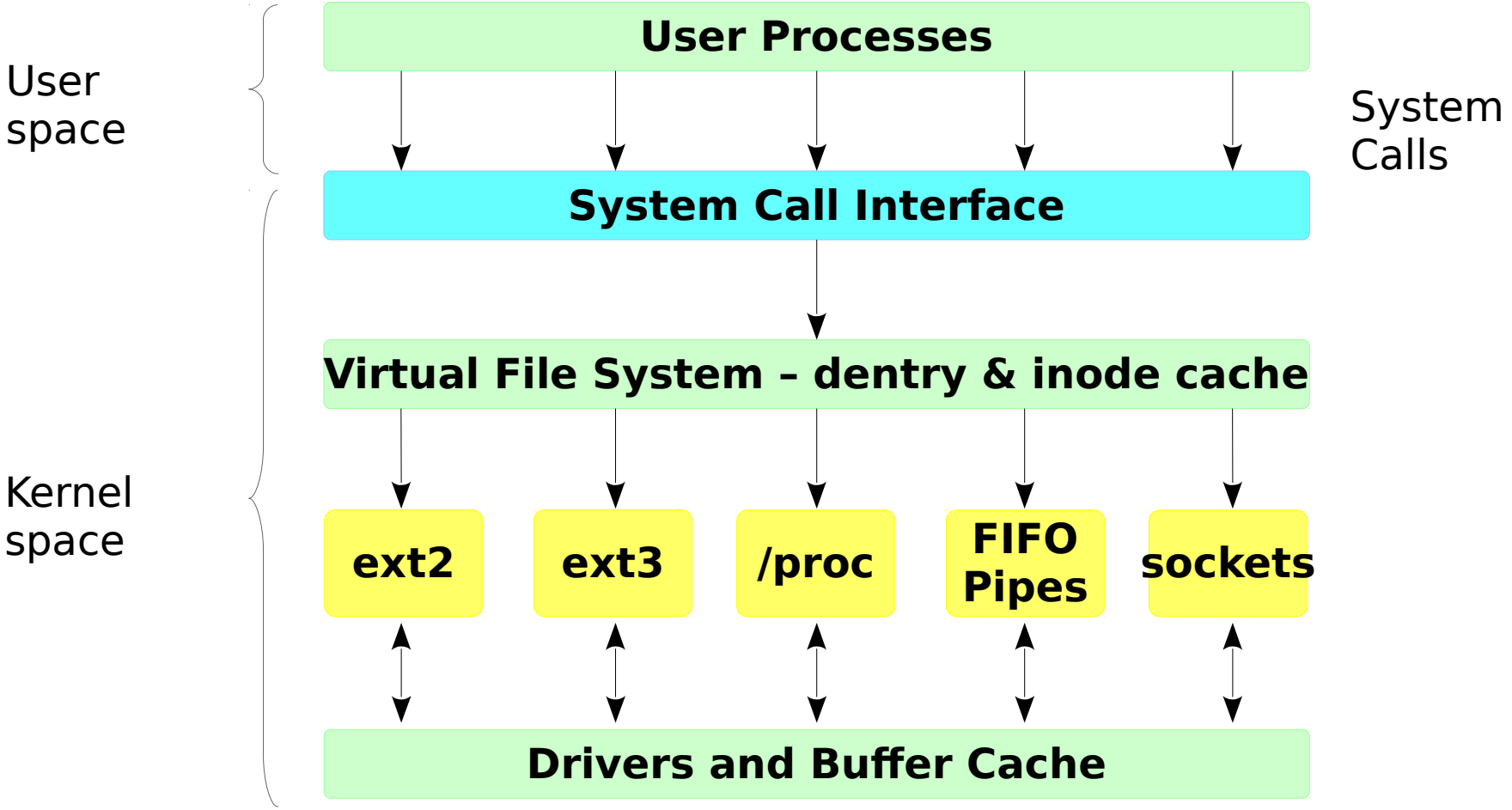
Introduction
Linux Kernel Subsystem



- Network Interface (NET):
  - provides access to several networking standards and a variety of network hardware
- Inter Process Communications (IPC):
  - supports several mechanisms for process-to-process communication on a single Linux system

Introduction

Virtual File System



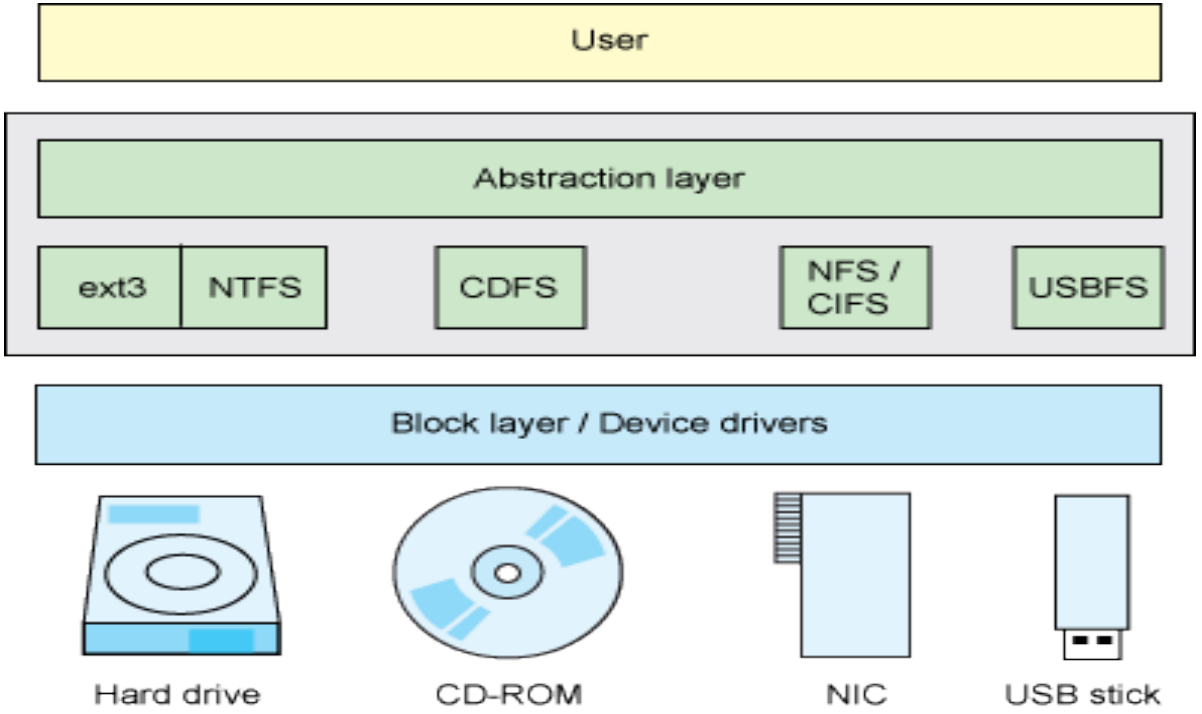
Introduction

Virtual File System

- **What?**
  - A kernel software layer that handles all system calls related to file systems.
  - Aim is to provide a common interface to several kinds of file systems.
- **Key Idea**
  - For each read, write or other function called, the kernel substitutes the actual function that supports a native Linux file system, for example the NTFS.

Introduction

Virtual File System



EMERTXE

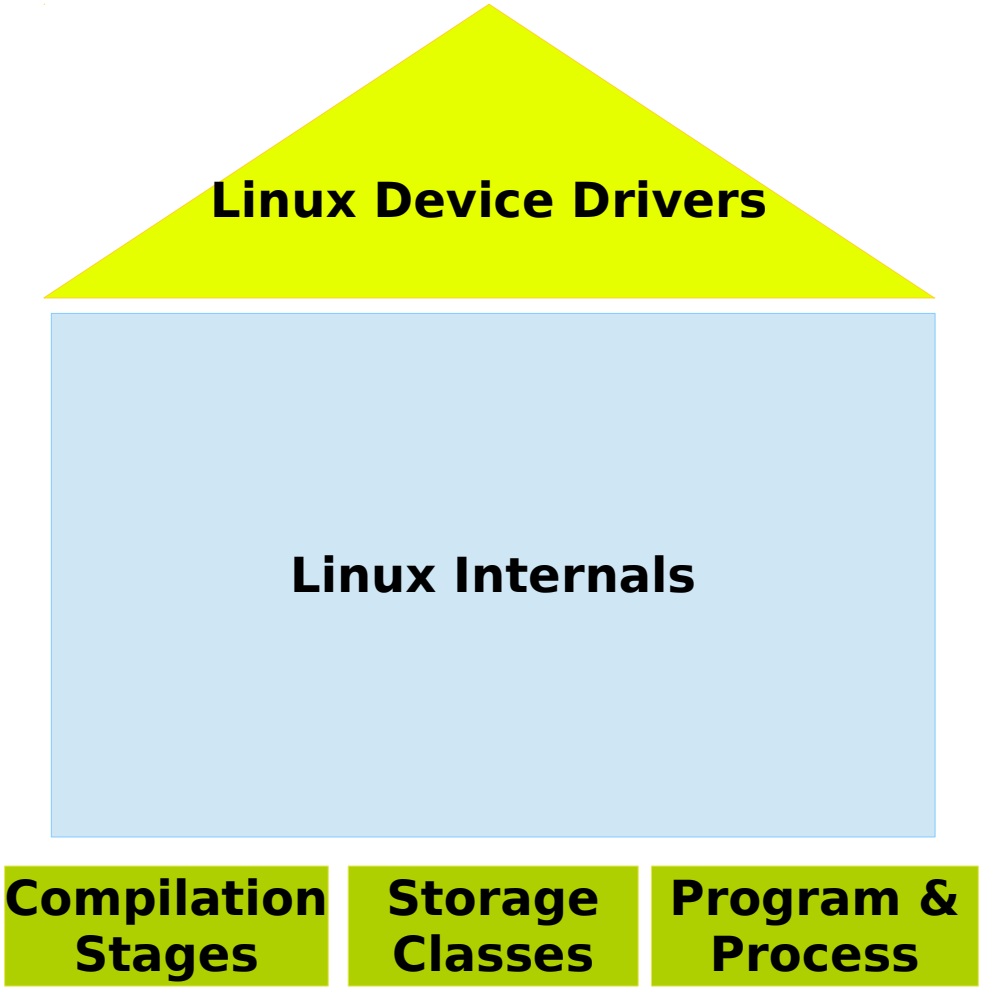
Introduction

Virtual File System

- Presents the user with a unified interface, via the file-related system calls.
- The VFS interacts with file-systems which interact with the buffer cache, page-cache and block devices.
- Finally, the VFS supplies data structures such as the dcache, inodes cache and open files tables.
  - Allocate a free file descriptor.
  - Try to open the file.
  - On success, put the new 'struct file' in the fd table of the process.On error, free the allocated file descriptor.

NOTE: VFS makes “Everythig is file” in Linux

Summary



Stay Connected

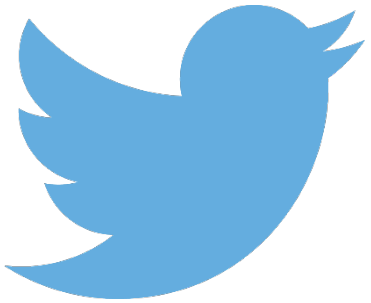
About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,

No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046

T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You