Linux

CS 377 Operating Systems Guest Lecture

Prateek Sharma

November-4-2014

Overview

Linux

1. A bit of OS history
2. Linux kernel architecture
3. Linux source code exploration
4. Implementation highlights
5. Some practical considerations in OS design

UNIX

- At the beginning of time was UNIX(circa 1970) [1]
- UNIX was a groundbreaking Operating System developed at AT&T Bell Labs
- Introduced and popularized powerful abstractions (files, shell)

Around 1990:

1. Patents and copyrights had left UNIX in limbo.
2. Couldn't run it on Personal Computers (x86 PCs).
3. UNIX became standardized (POSIX) [2]

[a] UNIX time starts on Jan 1 1970
[b] Which is why BSD, Solaris, MacOS-X, Linux are all "UNIX"

GNU: GNU is Not Unix

GNU

Free *(Free as in Freedom)* programs developed to be POSIX compatible and mimix UNIX functionality.

- Shell (command interpreter); compile, link, run, debug programs
- UNIX core utilities : `cp,mv,cat,ls,awk,sed,grep,less,man,dd,kill,ps`
- Text-editor (Emacs)
- Graphical interface (GNOME).

GNU was great, but an OS needs a kernel to run on actual hardware

Linux

- Kernel developed by Linus Torvalds in 1992 which ran on x86.
- "UNIX compatible" - Same system-call interface, similar design and architecture.
- GNU programs could run without additional effort!
- "Distributions = kernel + useful software" - Ubuntu, RedHat, Debian, Slackware.

Linux, GNU/Linux

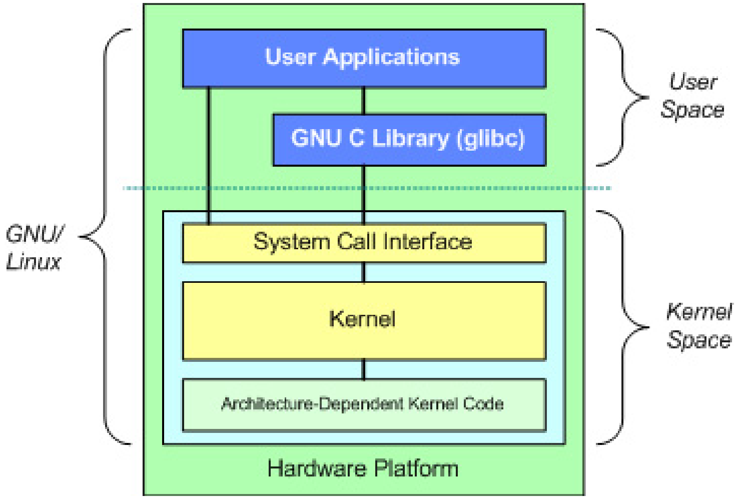
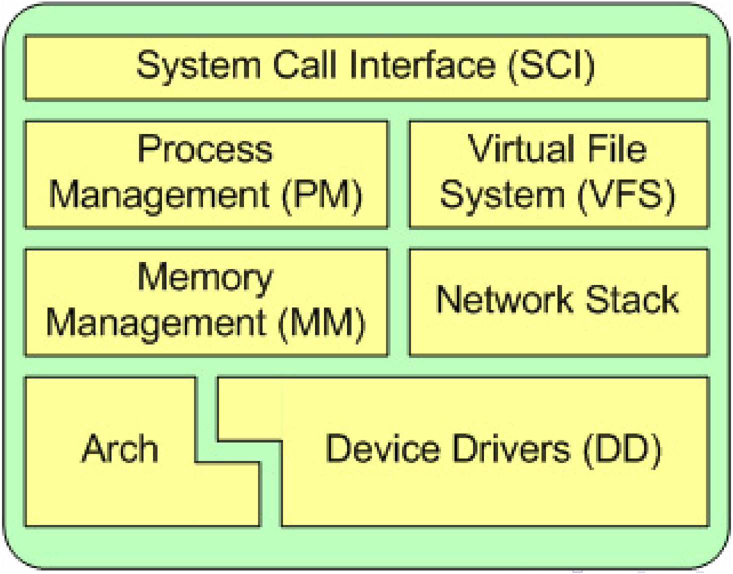


Figure : GNU/Linux

Linux Runs on everything

- Vast majority of all servers (»50%)
- Smartphones (Android, Tizen, etc)
- Embedded devices (Wireless routers, GPS, Raspberry Pi)
- Supercomputers
- Desktops (Ubuntu)

Kernel components



Navigating the kernel source

- Just over 17 million lines of code
- Mostly written in C. Architecture specific code in assembly
- Limited documentation. Things change too fast — features added/removed/changed at great pace
- New kernel release every 3 months or so
- Fundamental design principles fairly stable

Kernel Modules

- Monolithic = one giant program
- One program = one address-space. Different components can call each other, share data, etc.
- Support multiple hardware devices, file-systems, protocols etc
- Can't have one kernel which supports *everything* out of the box.
- Precompiled modules can be dynamically loaded into the kernel: `http://lxr.free-electrons.com/source/arch/x86/kernel/module.c`
- Kernel components can be compiled into the kernel or as modules.
- Modules are specially compiled relocatable ELF object files.
- `lsmod`, `modprobe` to list, add modules.

Linux Syscalls

1. Programs call system calls using a C-library wrapper
2. Syscall gate in kernel `http://lxr.free-electrons.com/source/arch/x86/kernel/syscall_64.c`
3. Actual context-switch and system-call handling done in assembly language: `http://lxr.free-electrons.com/source/arch/x86/kernel/entry_64.S`
4. Syscall-table: `http://lxr.free-electrons.com/source/arch/x86/syscalls/syscall_64.tbl`
5. Strace to record all system calls and arguments. Useful in debugging.

Kernel Components

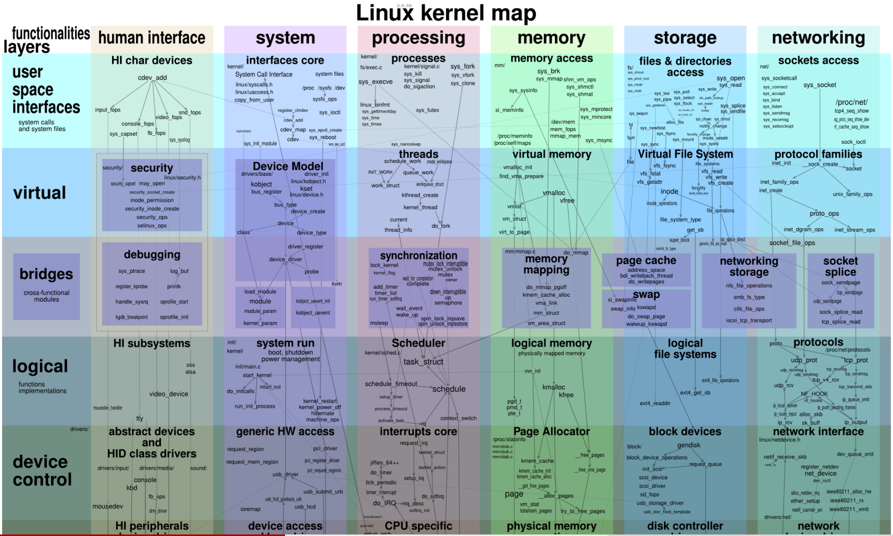
- CPU Scheduling :
  http://lxr.free-electrons.com/source/kernel/sched/fair.c
- Memory management : Page Cache
  http://lxr.free-electrons.com/source/mm/filemap.c
- VFS and I/O :
  http://lxr.free-electrons.com/source/fs/open.c
- Synchronization - spinlocks, RCU (Read-Copy-Update)

Other Kernel Functionality

- Context switching : http://lxr.free-electrons.com/source/arch/x86/kernel/process_64.c#L278
- Fork:
  http://lxr.free-electrons.com/source/kernel/fork.c#L1617
- Process Control Block : http://lxr.free-electrons.com/source/include/linux/sched.h#L1223

Detailed Kernel components

http://www.makelinux.net/kernel_map/



Compiling your kernel

- Get sources from `http://kernel.org`
- make menuconfig ;
- make ; make modules ; make headers;
- make install ; make modules-install;
- Whole process can take 1-2 hours.

What is my kernel doing?!

- Logs: `dmesg`, `/var/log/messages`
- `proc`,`sys` filesystems
- Function-level tracing using `ftrace`, `perf`
- Attaching a debugger — serial port, Virtual Machine.

What makes Linux so popular?

- Good implementation of very good UNIX ideas
- Huge open-source community adding features, fixing bugs, testing, etc.
- Licensing : Gnu Public License allows users to modify source code, but also needs to be redistributed along with binaries.
- High performance, good scalability, supports large number of devices

Other UNIX variants such as FreeBSD, Solaris/Illumos also share most of these features

Resources

- Linux Kernel Mailing List : `http://lkml.org`
- The Design of the UNIX Operating System - Maurice J. Bach *(UNIX kernel design and internals)*
- Operating Systems Design and Implementation - Andrew Tanenbaum *(Introduces MINIX with complete source code)*