Building, Running and Monitoring the Linux kernel

Praktikum Kernel Programming

University of Hamburg

Scientific Computing

Winter semester 2014/2015

Outline

- The Linux kernel source tree
- Configure, compile, install the Linux kernel
- Linux kernel boot and initialization
- Monitoring
- Summary

Where can I find it

- Download from http://www.kernel.org
  - Mainline, Stable, long term
  - git clone url dir_name
- Tree navigation
  - Web browser view lxr
    - http://lxr.sourceforge.com
    - http://lxr.free-electrons.com
  - Symbolic database from all files, cscope
    - http://cscope.sourceforge.net

Kernel source tree

- **arch**
    - contains all of the architecture specific kernel code. It has further subdirectories, one per supported architecture, for example i386 and alpha.
- **block**
    - contains the implementation of the block I/O layer
- **crypto**
    - implements cipher operations and the cryptography API

Kernel source tree

- **Documentation**
  - kernel source documentation
- **drivers**
  - all of the system's device drivers live in this directory. They are further sub-divided into classes of device driver, for example block
- **firmware**
  - device firmware that is needed to use certain drivers

Kernel source tree

- **fs**
  - file system code. This is further sub-divided into directories, one per supported file system
- **include**
  - contains most of the include files needed to build the kernel code. It too has further subdirectories including one for every architecture supported
- **init**
  - contains the initialization code for the kernel and it is a very good place to start looking at how the kernel works

Kernel source tree

- **ipc**
    - contains the kernels Inter-Process Communication (IPC) mechanism such as message queues, semaphores, shared memory.
- **kernel**
    - core subsystems, for example the scheduler. The architecture specific kernel code is in arch/*/kernel.
- **lib**
    - this directory contains the kernel's library code. The architecture specific library code can be found in arch/*

Kernel source tree

- **mm**
  - contains all of the memory management code. The architecture specific memory management code lives down in arch/*/mm/
- **modules**
  - directory used to hold built modules
- **net**
  - the kernel's networking code, (ethernet, ipv4)
- **samples**
  - demonstrative code

Kernel source tree

- **scripts**
  - contains the scripts (for example awk and tk scripts) that are used when the kernel is configured
- **security**
  - Linux security module, including SELinux
- **sound**
  - Advance Linux Sound Architecture (ALSA), sound card drivers.

Kernel source tree

- **usr**
  - user-space interaction (initramfs)
- **tools**
  - kernel and user development tools, mostly used for performance counting
- **virt**
  - the virtualization infrastructure

Outline

- The Linux kernel source tree
- ➢ Configure, compile, install the Linux kernel
- Linux kernel boot and initialization
- Monitoring
- Summary

Configuration

- Typical kernel has >> 1000 configuration options
- Default configuration part of the board support package (BSP)
- Configuration file .config
- Configuration options are typically Booleans or Tristate
  - Yes
  - No
  - Module
- Examples
  - CONFIG_SMP, enables or disables SMP supports
  - CONFIG_LOCK_STAT, enables or disables lock statistics

Tweak configuration using make, targets

- **config**
  - interactive for each option
- **menuconfig**
  - ncurses text menu
- **xconfig**
  - graphical menus using Qt
- **gconfig**
  - graphical menus using Gtk+
- **defconfig**
  - default configuration based on the architecture

Tweak configuration using make, targets

- **oldconfig**
  - validate and update the configuration
- **randconfig**
  - random answer to all options
- **allmodconfig**
  - selecting modules when possible
- **allyesconfig**
  - all options are accepted with yes
- **allnoconfig**
  - all options are answered with no

Naming the new Kernel

- Edit the top level Makefile

```
VERSION = 3
PATCHLEVEL = 6
SUBLEVEL = 35
EXTRAVERSION = -rc2
NAME = my kernel
```

Compile

- Required packages
  - development tools, make, gcc, gzip, etc.
- Several distributions offer packages
- Cross compile
  - export ARCH=...
  - export CROSS_COMPILE=...

Compile make targets

- **default**
  - builds kernel + modules
- **bzImage**
  - builds kernel
  - generates: arch/arm/boot/bzImage
- **modules**
  - builds loadable modules
  - generates: lib/modules/<kernel.versio-name>

Compile make targets

- **-j<n> e.g. -j2**
  - spawn multiple build jobs
- **clean**
  - generated files
- **mrproper**
  - generated files+config+backup files
- **distclean**
  - all the above+patch files

Install

- **make install**
  - copy kernel image to the proper directory /boot
- **make module_install**
  - install build modules in the correct home under /lib/modules
- Update the boot loader
  - LILO or grub configuration file
  - add the new entry for the newly build kernel

Kernel command line

- Kernel behaviour set by boot “command line”
- see Documentation/kernel-parameters.txt
- Examples
  - console: device to send kernel messages to
    - e.g. console=ttyS0,115200
  - root: set device to load root file system from,
    - e.g. root=/dev/sda1
  - quiet: output fewer console messages
  - debug: output all console messages
  - maxcpus: control the active CPU
- Can be set in Bootloader, e.g. GRUB

The root file system

- Mounted by the kernel during boot
    - Provides additional kernel modules that are needed
- specified the **root** kernel command line parameter
- Loaded from:
    - memory (ram disk / initramfs)
    - storage device
    - network
- The “module” that provides access must be embedded in the kernel or it cannot mount..

Initramfs

- initial ram file system
  - successor of initrd
- cpio archive of the initial file system
  - cpio
    - file archive and file format
    - copy in and out
- gets loaded into memory during startup
- contains device drivers and tools needed to mount the real file system

Outline

- The Linux kernel source tree
- Configure, compile, install the Linux kernel
- ➢ Linux kernel boot and initialization
- Monitoring
- Summary

Boot process of the kernel

- BIOS loads Master Boot Record (MBR) from the boot device
- Code that exist in the MBR reads the partition table of the boot device and reads the bootloader (GRUB, LILO) from the boot partition
- The bootloader reads the compressed kernel image
  - Passes the control to it using the command line options
- The kernel un-compresses itself

Boot process of the kernel

- Proceeds to "real" mode where the first level initializations are done
  - In read mode, it can access only the first 1MB of memory
- Startup is performed in the "protected" mode and begins initializing the CPU subsystem
  - In "protected" mode you can use many advance feature of the processor such as paging
- It follows the memory and the process managements subsystems
- Peripheral buses, I/O buses are stated next

Init process

- At last the kernel invokes the init program that is the parent of all Linux processes
- First program to be run /sbin/init
  - Begins by reading /etc/inittab
- Run levels (system states) for System V init
  - 0 is halt
  - 1 is single user
  - 2-5 are multi-user
  - 6 reboot
- Starts initialisation scripts
  - Fount at /ect/init.d

Outline

- The Linux kernel source tree
- Configure, compile, install the Linux kernel
- Linux kernel boot and initialization

➢ Monitoring
  - Metrics
  - Tools
- Summary

CPU metrics /proc/stat

- **Utilization**
  - overall utilization per processor
- **User time**
  - percentage spent on user processes
- **System time**
  - percentage spent on kernel operations
- **Waiting**
  - time spent waiting for I/O operations
- **Idle time**
  - system was idle waiting tasks

CPU metrics /proc/stat

- **Nice time**
  - time spent on re-nicing processes
- **Runnable processes**
  - processes ready to run
- **Blocked**
  - processes blocked by I/O operations
- **Context switch**
  - number of context switches
- **Interrupts**
  - number of hard and soft interrupts

Memory metrics

- **Free memory**
  - amount of free memory in the system
- **Swap usage**
  - amount of swap used
- **Buffer and cache**
  - memory allocated for I/O
- **Slabs**
  - kernel usage of memory
- **Active VS. inactive memory**
  - Inactive memory is a candidate to be swapped

Network interface metrics

- **Packets sent/received**
- **Bytes sent/received**
- **Collisions per second**
  - Sustained values indicate network infrastructure bottlenecks
- **Packets dropped**
  - Can be caused by the firewall or limited buffers
- **Overruns**
  - how many times runned out of buffers
- **Errors**
  - count the packets that are marked faulty

Block device metrics

- **Iowait**
  - the time CPU spends waiting for I/O to complete
- **Average queue length**
  - amount of outstanding I/O requests
  - high value indicate I/O bottleneck
- **Average wait**
  - average time in ms that takes for an I/O operation to complete
- **Transfer per second**
  - how many I/O operations are performed

Block device metrics

- **Blocks read/write per second**
  - number of blocks that were read or written (usually each block is 1024 Bytes)
- **Kilobytes read/write per second**
  - number of blocks that were read or written in KBytes

Block device metrics

- **Blocks read/write per second**
  - number of blocks that were read or written (usually each block is 1024 Bytes)
- **Kilobytes read/write per second**
  - number of blocks that were read or written in KBytes

Outline

- The Linux kernel source tree
- Configure, compile, install the Linux kernel
- Linux kernel boot and initialization
- Monitoring
    - Metrics
    - ➢ Tools
- Summary

Generic admin tools

- dmesg
  - Prints the message buffer of the kernel
- strace
  - Monitor interaction between user and kernel
- oprofile
  - System-wide statistical profiling tool

CPU Tools

- top
  - Process activity
- ps, pstree
  - Display the running processes
- kill
  - Sends the SIGTERM signal to the process
- mpstat
  - Displays activities for each available processor
- numastat
  - NUMA-related statistics
- pmap
  - Process memory usage

I/O Tools

- vmstat
  - Report virtual memory statistics
- free
  - Display the amount of free and used memory
- iostat
  - Report block device statistics
- lsblk
  - List block devices
- lsof
  - List open files

Network Tools

- ping
  - check if a server responds
- traceroute
  - display the route path
- nslookup
  - get domain name or IP address
- netstat
  - displays network stats
- tcpdump
  - dump traffic on a network

Proc files

- /proc/stat
  - cpu
- /proc/diskstats
  - disk
- /proc/meminfo
  - memo

Outline

- The Linux kernel source tree
- Configure, compile, install the Linux kernel
- Linux kernel boot and initialization
- Monitoring
- Summary

Summary

- Which are the main directories in the Linux kernel source tree
- How to configure and built the Linux kernel
- Which is the start up process of the Linux system
- Which tools are used to monitor the Linux kernel