PKU--LL102-启动流程

阶段一：Booting

- 问题：先有鸡还是先有蛋（"chicken-and-egg problem"）
  - Bootstrapping, a technique in computer programming used to avoid chicken-and-egg scenarios where two programs are mutually needed for compiling or loading each other
  - 程序的加载就是蛋，程序的运行就是鸡；要让程序运行需要先加载，而加载的过程需要程序运行
    - 编译是另外一种chicken-and-egg问题
- Booting: http://en.wikipedia.org/wiki/Booting
  - bootstrap的简写
    - bootstrapping：马丁靴鞋后跟上的小带子；所谓自举(Bootstrap)就是“长筒靴后面连接的皮纽”的意思。在穿长筒靴时，由自身将这个Bootstrap提拉起来穿的。
      - In general parlance（说法、用语）, bootstrapping usually refers to the starting of a self-sustaining process that is supposed to proceed without external input.
    - The term appears to have originated in the early 19th century United States (particularly in the phrase "pull oneself over a fence by one's bootstraps"), to mean an absurdly impossible action, an adynaton（怪诞诗）.
  - 重点学习Modern boot loaders一节
    - multiple-stage boot loader
    - Some computer systems, upon receiving a boot signal from a human operator or a peripheral device, may load a very small number of fixed instructions into memory at a specific location, initialize at least one CPU, and then point the CPU to the instructions and start their execution.
    - The small program that starts this sequence is known as a bootstrap loader, bootstrap or boot loader. This small program's only job is to load other data and programs which are then executed from RAM. Often, multiple-stage boot loaders are used, during which several programs of increasing complexity load one after the other in a process of chain loading.
    - Master Boot Record: The purpose of a boot sector is to allow the boot process of a computer to load a program (usually, but not necessarily, an operating system) stored on the same storage device.
    - Second-stage boot loaders, such as GNU GRUB, BOOTMGR, Syslinux, NTLDR or BootX, are not themselves operating systems, but are able to load an operating system properly and transfer execution to it; the operating system subsequently initializes itself and may load extra device drivers.
  - Network Booting:
    - Network booting, shortened netboot, is the process of booting a computer from a network rather than a local drive. This method of booting can be used by routers, diskless workstations and centrally managed computers (thin clients) such as public computers at libraries and schools. Network booting can be used to centralize management of disk storage, which supporters claim can result in reduced capital and maintenance costs. It can also be used in cluster computing, in which nodes may not have local disks.
    - As with the second-stage boot loader, network booting begins by using generic network access methods provided by the network interface's boot ROM, which typically contains a Preboot Execution Environment (PXE) image.
  - GRUB version 2:
    - Stage 1: boot.img is stored in the master boot record (MBR), or optionally in any of the volume boot records (VBRs), and addresses the next stage by an LBA48 address (the 1024 cylinder boundary of GRUB legacy is omitted); at installation time it is configured to load the first sector of core.img.
    - Stage 1.5: core.img is by default written to the sectors between the MBR and the first partition, when these sectors are free and available. For legacy reasons, the first partition of a hard drive does not begin at sector 1 (counting begins with 0) but at sector 63, leaving a gap of 62 sectors of empty space. That space is not part of any partition or file system, and therefore not prone to any problems related with it. Once executed, core.img will load its configuration file and any other modules needed, particularly file system drivers; at installation time, it is generated from diskboot.img and configured to load the stage 2 by its file path.
    - Stage 2: files belonging to the stage 2 are all being held in the /boot/grub directory, which is a subdirectory of the /boot directory specified by the Filesystem Hierarchy Standard (FHS).
- Chain loading（链式加载）
  - https://en.wikipedia.org/wiki/Chain_loading
  - Chain loading is a method used by computer programs to replace the currently executing program with a new program, using a common data area, to pass information from the current program to the new program. It occurs in several areas of computing.
  - Chain loading is similar to the use of overlays. Unlike overlays, however, chain loading replaces the currently executing program in its entirety. Overlays usually replace only a portion of the running program. Like the use of overlays, the use of chain loading increases the I/O load of an application.
- UEFI
  - https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface
  - http://www.uefi.org/
  - The Unified Extensible Firmware Interface (UEFI, pronounced as an initialism U-E-F-I or like "unify" without the n[a]) is a specification that defines a software interface between an operating system and platform firmware.

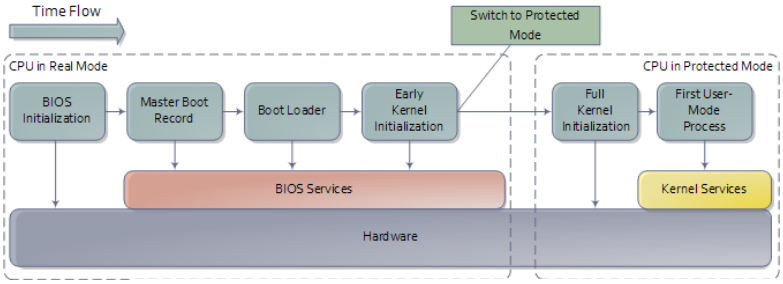
    - UEFI is meant to replace the Basic Input/Output System (BIOS) firmware interface, originally present in all IBM PC-compatible personal computers.
    - UEFI（Unified Extensible Firmware Interface），中文为“统一的可扩展固件接口”，该规范为操作系统和平台固件之间的接口定义了一种新的模型。这些接口包括平台相关信息的数据表，以及启动服务（boot services）和运行时服务（runtime services），从而为操作系统的启动和启动前应用程序（pre-boot applications）的运行提供了一个标准环境。

自学：

- Booting: http://en.wikipedia.org/wiki/Booting
- Firmware: https://en.wikipedia.org/wiki/Firmware
- Non-volatile memory: https://en.wikipedia.org/wiki/Non-volatile_memory
- BIOS：http://en.wikipedia.org/wiki/BIOS
- MBR: Master boot record: http://en.wikipedia.org/wiki/Master_boot_record
    - 硬盘的区块表示方法是（磁道，柱面，扇区），1个扇区的大小是512B
    - （0磁道，0柱面，1扇区）就是MBR
    - （0磁道，1柱面，1扇区）是第一分区的启动扇区PBR，其他分区的第一个扇区也被称为PBR
- GNU GRUB: https://en.wikipedia.org/wiki/GNU_GRUB
- POST (power-on self-tests): https://en.wikipedia.org/wiki/Power-on_self-test
- UEFI: https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

See Also：

- http://duartes.org/gustavo/blog/post/how-computers-boot-up/
- http://processors.wiki.ti.com/index.php/The_Boot_Process
    - The Boot Process: The 4 Bootloader Stages
        - 1) ROM
        - 2) SPL (or Secondary Program Loader)
        - 3) U-BOOT
        - 4) Linux Kernel
- Comparison of boot loaders: https://en.wikipedia.org/wiki/Comparison_of_boot_loaders
- u-boot: http://www.denx.de/wiki/U-Boot
- UEFI：http://www.uefi.org/



（上图来源：http://duartes.org/gustavo/blog/post/how-computers-boot-up/）

阶段二：GIT与内核代码

- Revision Control：https://en.wikipedia.org/wiki/Revision_control
    - revision: 修订，修订版
    - A component of software configuration management, Revision control, also known as version control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information.
    - Changes are usually identified by a number or letter code, termed the "revision number," "revision level," or simply "revision." For example, an initial set of files is "revision 1." When the first change is made, the resulting set is "revision 2," and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.
    - Version control systems (VCS) most commonly run as stand-alone applications, but revision control is also embedded in various types of software such as word processors and spreadsheets, e.g., Google Docs and Sheets and in various content management systems, e.g., Wikipedia's Page history.
- DRCS/DVCS：distributed revision control system (DRCS), also known as a distributed version control system (DVCS)
    - https://en.wikipedia.org/wiki/Distributed_revision_control
    - In computer programming, distributed revision control, also known as distributed version control or decentralized version control, allows many software developers to work on a given project without requiring them to share a common network.
    - Distributed vs. centralized
        - Distributed revision control takes a peer-to-peer approach to version control, as opposed to the client-server approach of centralized systems. Rather than a single, central repository on which clients synchronize, each peer's working copy of the codebase is a complete repository. Distributed revision control synchronizes repositories by exchanging patches (sets of changes) from peer to peer.

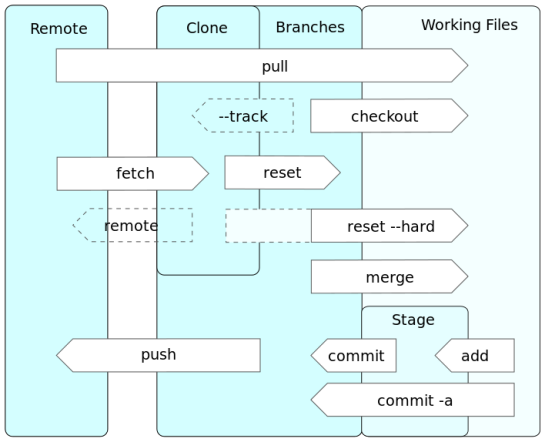
        - This results in some important differences from a centralized system:
            - No canonical, reference copy of the codebase exists by default; only working copies.
            - Common operations (such as commits, viewing history, and reverting changes) are fast, because there is no need to communicate with a central server.
            - Communication is only necessary when sharing changes among other peers.
            - Each working copy effectively functions as a remote backup of the codebase and of its change-history, protecting against data loss.
- git：https://en.wikipedia.org/wiki/Git_(software)
    - Git (/g t/) is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.
    - Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.
    - 安装git
        - sudo aptitude install git
    - git工作原理（见下方的图）
    - 十五分钟学会git：http://try.github.io/
    - 基本配置：
        - git config --global user.email "gxt@mprc.pku.edu.cn"
        - git config --global user.name "GUAN Xuetao"
        - cat ~/.gitconfig
- 获取内核源代码包
    - 内核版本号：
        - uname -a
        - head Makefile
    - 方案一：sudo aptitude install kernel-package
        - 不建议，因为要安装texlive、tk、x11等很多server版不需要的内容
    - 方案二：sudo aptitude install linux-source
        - 注意：这是ubuntu提供的最新版本，要比server中实际安装的内核版本高
        - 如果未安装build-essential，这个安装过程也会自动安装binutils，gcc，make，以及需要的各种库
        - 安装的位置：/usr/src/linux-source-3.16.0.tar.bz2
        - 注意：内核解压缩的位置不要用/usr/src/linux目录
            - 内核中的README中：
                - Do NOT use the /usr/src/linux area! This area has a (usually incomplete) set of kernel headers that are used by the library header files. They should match the library, and not get messed up by whatever the kernel-du-jour happens to be.
                - du jour：当天供应的，当日特色的
    - 方案三：获取官方源代码
        - wget http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.16.7.tar.xz
    - 方案四：获取Ubuntu修改后的源代码
        - git clone git://kernel.ubuntu.com/ubuntu/ubuntu-utopic.git

自学：

- Git: https://en.wikipedia.org/wiki/Git_(software)
- Revision control: https://en.wikipedia.org/wiki/Revision_control
- Distributed revision control: https://en.wikipedia.org/wiki/Distributed_revision_control
- Tux: http://en.wikipedia.org/wiki/Tux (Linux mascot)
- Linux kernel: http://en.wikipedia.org/wiki/Linux_kernel
- LKML：https://en.wikipedia.org/wiki/Linux_kernel_mailing_list
- LXR Cross Referencer: http://en.wikipedia.org/wiki/LXR_Cross_Referencer

See Also:

- Comparison of revision control software: https://en.wikipedia.org/wiki/Comparison_of_revision_control_software
- Ubuntu Kernel Development FAQ: https://wiki.ubuntu.com/Kernel/FAQDeveloper
- LXR linux: https://lxr.missinglinkelectronics.com/linux
- The Linux Documentation Project: http://www.tldp.org/
- Kernel Newbies（初学者）: http://kernelnewbies.org/



（上图来源：https://commons.wikimedia.org/wiki/File:Git_operations.svg）

阶段三：Linux启动过程

- Linux startup process is the multi-stage initialization process performed during booting a Linux installation.
  - including firmware initialization, execution of a boot loader, loading and startup of a Linux kernel image, and execution of various startup scripts and daemons
- Boot loader phase
  - The boot loader phase varies by computer architecture.
  - GRUB, LILO, SYSLINUX, Loadlin
- Kernel phase: kernel is loaded in two stages
  - Kernel loading stage:
    - The kernel as loaded is typically an image file, compressed into either zImage or bzImage formats with zlib. A routine at the head of it does a minimal amount of hardware setup, decompresses the image fully into high memory, and takes note of any RAM disk if configured. It then executes kernel startup via ./arch/i386/boot/head and the startup_32 () (for x86 based processors) process.
  - Kernel startup stage:
    - According to Red Hat, the detailed kernel process at this stage is therefore summarized as follows:
      - "When the kernel is loaded, it immediately initializes and configures the computer's memory and configures the various hardware attached to the system, including all processors, I/O subsystems, and storage devices. It then looks for the compressed initrd image in a predetermined location in memory, decompresses it, mounts it, and loads all necessary drivers. Next, it initializes virtual devices related to the file system, such as LVM or software RAID before unmounting the initrd disk image and freeing up all the memory the disk image once occupied. The kernel then creates a root device,[clarification needed] mounts the root partition read-only, and frees any unused memory. At this point, the kernel is loaded into memory and operational. However, since there are no user applications that allow meaningful input to the system, not much can be done with it."
    - An initramfs-style boot is similar, but not identical to the described initrd boot.
    - At this point, with interrupts enabled, the scheduler can take control of the overall management of the system, to provide pre-emptive multi-tasking, and the init process is left to continue booting the user environment in user space.
- Early user space
  - initramfs, also known as early user space, has been available since version 2.5.46 of the Linux kernel, with the intent to replace as many functions as possible that previously the kernel would have performed during the start-up process. Typical uses of early user space are to detect what device drivers are needed to load the main user space file system and load them from a temporary filesystem.
  - ramfs, rootfs and initramfs:
    - Ramfs is a very simple filesystem that exports Linux's disk caching mechanisms (the page cache and dentry cache) as a dynamically resizable RAM-based filesystem.
    - Rootfs is a special instance of ramfs (or tmpfs, if that's enabled), which is always present in 2.6 systems. Most systems just mount another filesystem over rootfs and ignore it.
    - Initramfs: All 2.6 Linux kernels contain a gzipped "cpio" format archive, which is extracted into rootfs when the kernel boots up.
  - RAM drive:
    - A RAM drive (also called a RAM disk) is a block of random-access memory (primary storage or volatile memory) that a computer's software is treating as if the memory were a disk drive (secondary storage).
    - The difference between a tmpfs and ramfs ram disk: http://www.jamescoyle.net/knowledge/951-the-difference-between-a-tmpfs-and-ramfs-ram-disk
  - "initrd" and "initramfs"--What's Up With That? http://www.linux.com/learn/linux-training/92607-the-kernel-newbie-corner-qinitrdq-and-qinitramfsq-whats-up-with-that

- init process
  - init is the parent of all processes on the system, it is executed by the kernel and is responsible for starting all other

processes; it is the parent of all processes whose natural parents have died and it is responsible for reaping those when they die. Processes managed by init are known as jobs and are defined by files in the /etc/init directory.

    - As of February 2014 Linux distributions such as Debian, Arch, Fedora, and openSUSE have moved to systemd, which replaces init process as the first process executed in user space during the Linux startup.
    - Upstart is an event-based replacement for the /sbin/init daemon which handles starting of tasks and services during boot, stopping them during shutdown and supervising them while the system is running. It was originally developed for the Ubuntu distribution, but is intended to be suitable for deployment in all Linux distributions as a replacement for the venerable System-V init.

自学：

- Linux startup process: https://en.wikipedia.org/wiki/Linux_startup_process
- gzip: https://en.wikipedia.org/wiki/Gzip
- zlib: https://en.wikipedia.org/wiki/Zlib
- vmlinux: https://en.wikipedia.org/wiki/Vmlinux
    - try ./scripts/extract-vmlinux
- RAM drive: https://en.wikipedia.org/wiki/RAM_drive
- ramfs, rootfs and initramfs: https://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt
- The difference between a tmpfs and ramfs ram disk: http://www.jamescoyle.net/knowledge/951-the-difference-between-a-tmpfs-and-ramfs-ram-disk
- "initrd" and "initramfs"--What's Up With That? http://www.linux.com/learn/linux-training/92607-the-kernel-newbie-corner-qinitrdq-and-qinitramfsq-whats-up-with-that
- init: https://en.wikipedia.org/wiki/Init

See Also:

- http://www.golinuxhub.com/2014/03/step-by-step-linux-boot-process.html
- http://duartes.org/gustavo/blog/post/kernel-boot-process/
- http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html （v0.9, November 2000）（obsolete, but very useful）
- Upstart Intro, Cookbook and Best Practises : http://upstart.ubuntu.com/cookbook/

阶段四：内核中的initcalls

- 学习initcall：
    - include/linux/init.h
    - init/main.c
        - do_initcalls -> do_initcall_level -> do_one_initcall
    - initcall level
        - early
            - early initcalls run before initializing SMP
        - pure
            - id: 0
            - no indenpendence on anything else
        - core -- Level 1
            - core_initcall, core_initcall_sync
            - Used for core kernel services and for some early arch/driver specific initialization that do not depend on any kernel core initialization.
        - postcore -- Level 2
            - postcore_initcall, postcore_initcall_sync
            - Initialization of functions that depend on core initialization and that do not fall into the other categories.
        - arch -- Level 3
            - arch_initcall, arch_initcall_sync
            - Used for architecture specific initialization (i386, ARM, Mips, etc.).
        - subsys -- Level 4
            - subsys_initcall, subsys_initcall_sync
            - Used to initialize kernel subsystems.
        - fs -- Level 5
            - fs_initcall, fs_initcall_sync, rootfs_initcall
        - device -- Level 6
            - device_initcall, device_initcall_sync
            - Device initialization.
        - late -- Level 7
            - late_initcall, late_initcall_sync
        - console
            - con_initcall

        - Console specific initialization.
    - security
        - security_initcall
        - Security specific initialization.
- Documentation/kernel-parameters.txt
    - initcall_debug
    - initcall_blacklist

See Also：

- Understanding The Linux Kernel Initcall Mechanism：
    - http://www.compsoc.man.ac.uk/~moz/kernelnewbies/documents/initcall/
    - 虽然上文是2003年的，但是基本原理没变

建议内容：

- Upstart