# PKU--LL103-内核模块

## 阶段一：Build与Make

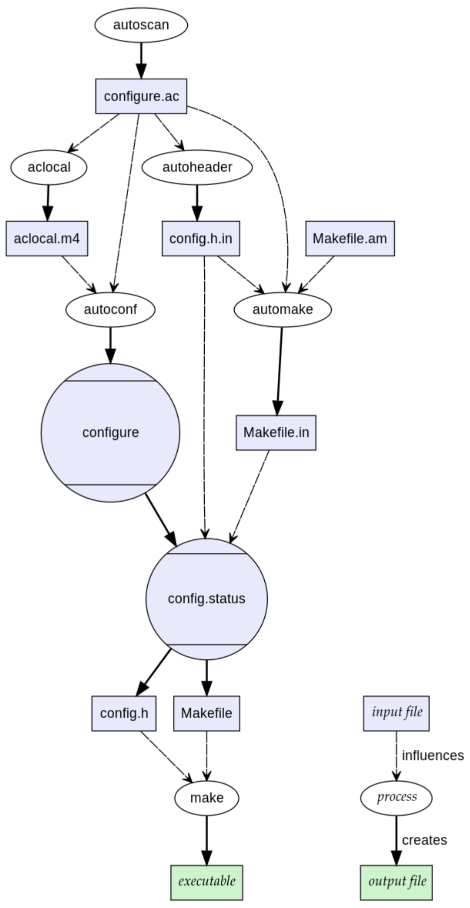
- 问题：为什么需要建立project？为什么需要Make？
  - 多个源代码的重新编译，头文件的依赖关系
  - 根本目的：提高生产力
- GNU build system: http://en.wikipedia.org/wiki/GNU_build_system
  - The GNU build system, also known as the Autotools, is a suite of programming tools designed to assist in making source code packages portable to many Unix-like systems.
  - It can be difficult to make a software program portable:
    - the C compiler differs from system to system;
    - certain library functions are missing on some systems;
    - header files may have different names.
  - One way to handle this is to write conditional code, with code blocks selected by means of preprocessor directives (#ifdef); but because of the wide variety of build environments this approach quickly becomes unmanageable. Autotools is designed to address this problem more manageably.
  - The GNU build system makes it possible to build many programs using a two-step process: configure followed by make.
  - The final executable software is most commonly obtained by executing the following commands:
    - ./configure
    - make
    - make install
- Build automation: http://en.wikipedia.org/wiki/Build_automation
  - Build automation is the act of scripting or automating a wide variety of tasks that software developers do in their day-to-day activities including things like:
    - compiling computer source code into binary code
    - packaging binary code
    - running automated tests
    - deploying to production systems
    - creating documentation and/or release notes
  - The advantages of build automation to software development projects include
    - Improve product quality
    - Accelerate the compile and link processing
    - Eliminate redundant tasks
    - Minimize "bad builds"
    - Eliminate dependencies on key personnel
    - Have history of builds and releases in order to investigate issues
    - Save time and money - because of the reasons listed above
  - Continuous integration (CI) is the practice, in software engineering, of merging all developer working copies with a shared mainline several times a day.
    - It was adopted as part of extreme programming (XP), which did advocate integrating more than once per day, perhaps as many as tens of times per day.
    - The main aim of CI is to prevent integration problems, referred to as "integration hell" in early descriptions of XP.
- Make and makefile
  - In software development, Make is a utility that automatically builds executable programs and libraries from source code by reading files called makefiles which specify how to derive the target program.
  - Though integrated development environments and language-specific compiler features can also be used to manage a build process, Make remains widely used, especially in Unix.
  - in kernel source dir:
    - make help
      - Cleaning targets: clean, mrproper, distclean
      - Configuration targets: config, menuconfig, defconfig, ...
      - Other generic targets: all, vmlinux, modules, ...
      - Static analysers: checkstack, namespacecheck, ...
      - Kernel packaging: rpm-pkg, targz-pkg, ...
      - Documentation targets: htmldocs, pdfdocs, ...
      - Architecture specific targets (x86): bzImage, install, ...
  - 真实的Makefile
    - （提问）内核Makefile的缺省规则（all，即vmlinux）
    - （提问）寻找内核Makefile中的.c生成.o的内置规则（scripts/Makefile.build L255~258）
    - （提问）寻找内核Makefile中生成依赖关系.d文件的命令（在tools/lib/traceevent/Makefile中采用gcc -MM实现命令check_deps）

自学：

- GNU build system: http://en.wikipedia.org/wiki/GNU_build_system
- Autoconf: https://en.wikipedia.org/wiki/Autoconf
- Automake: https://en.wikipedia.org/wiki/Automake
- GNU Libtool: https://en.wikipedia.org/wiki/GNU_Libtool
- Configure script: https://en.wikipedia.org/wiki/Configure_script
- Build automation: http://en.wikipedia.org/wiki/Build_automation
- CI（Continuous integration）：https://en.wikipedia.org/wiki/Continuous_integration
- Make: https://en.wikipedia.org/wiki/Make_(software)
- Makefile: https://en.wikipedia.org/wiki/Makefile
- Documentation/kbuild/makefiles.txt

See Also:

- Autotools Tutorial: https://www.lrde.epita.fr/~adl/autotools.html
- Autotools: a practitioner's guide to Autoconf, Automake and Libtool
  - http://www.freesoftwaremagazine.com/books/autotools_a_guide_to_autoconf_automake_libtool
- GNU Make: https://www.gnu.org/software/make/
- What is wrong with make: http://freecode.com/articles/what-is-wrong-with-make



---

## 阶段二：内核编译及更新

- 进入内核所在目录
  - git branch gxt
  - git checkout gxt
  - git show-branch master gxt
  - vim Makefile
    - 在EXTRAVERSION后增加 -gxt1
  - git commit -asm"Add new branch and change EXTRAVERSION"
    - 目的：不增加-dirty
  - git show
- 配置内核
  - cp /boot/config-3.16.0-23-generic .config
  - 浏览内核的配置选项
    - sudo apt-get install libncurses-dev
    - make menuconfig
      - 注意：Exit时选择Save

        - 在Xwindow上，可以用make xconfig来查看（需要安装qt库）
    - ./scripts/diffconfig .config.old .config
        - 会看到几个新的选项（这是因为下载的内核比原来的内核更新）
- 编译内核
    - make
        - 在虚拟机下运行了145分钟
        - 在不同的体系结构下，该规则并不需要完全相同，通常会完成：
            - make bzImage
            - make modules
    - 小技巧：
        - 忽略编译信息：make > /dev/null
        - 加速内核编译：make -j（如为服务器，建议增加数字，如-j4，避免过度占据资源）
    - 可以采用ccache来提高编译速度
        - sudo apt-get install ccache
        - 如果经常执行make clean，ccache非常有用，它可以作为编译时的缓冲，从而加快重新编译的速度
        - 方案一：修改内核根目录的Makefile
            - CC = ccache $(CROSS_COMPILE)gcc
            - HOSTCC = ccache gcc
        - 方案二：symlinks method
            - ln -s /usr/bin/ccache /usr/local/bin/gcc
            - ln -s /usr/bin/ccache /usr/local/bin/g++
            - ln -s /usr/bin/ccache /usr/local/bin/cc
            - ln -s /usr/bin/ccache /usr/local/bin/c++
            - 查询PATH变量：/usr/local/bin在/usr/bin之前
- 内核打包（对本地更新内核，该操作并非必需）
    - make targz-pkg
        - 源代码目录下：linux-3.16.7-ckt10-gxt1+-x86.tar.gz
        - 查看tar-install目录：
            - boot目录下的四个文件
            - lib/firmware
            - lib/modules/3.16.7-ckt10-gxt1+
- 安装内核
    - sudo make modules_install
        - /lib/目录下
    - sudo make install
        - sh ./arch/x86/boot/install.sh 3.16.7-ckt10-gxt1+ arch/x86/boot/bzImage System.map "/boot"
            - /sbin/installkernel <version> <image> <System.map> <directory>
            - run-parts /etc/kernel/postinst.d
                - run-parts runs all the executable files named within constraints described below, found in directory <directory>.
                - /etc/kernel/postinst.d/apt-auto-removal ...
                    - Mark as not-for-autoremoval those kernel packages
                    - See /etc/apt/apt.conf.d/01autoremove-kernels
                - /etc/kernel/postinst.d/dkms ...
                    - dkms - Dynamic Kernel Module Support
                - /etc/kernel/postinst.d/initramfs-tools ...
                    - update-initramfs: generate an initramfs image
                    - man update-initramfs
                        - 配置文件：/etc/initramfs-tools
                - /etc/kernel/postinst.d/zz-update-grub ...
                    - update-grub: Generating grub configuration file
                    - See /boot/grub/grub.cfg
                - /etc/kernel/postinst.d/update-notifier
                    - See /var/run/reboot-required and /var/run/reboot-required.pkgs
- 重启查看版本号
    - sudo reboot
    - 在GRUB菜单中选择新的内核
    - uname -a
        - Linux ubuntu 3.16.7-ckt10-gxt1+ ......

---

## 阶段三：内核模块

- 问题：如何向kernel中加入新的功能
- LKM：http://en.wikipedia.org/wiki/Loadable_kernel_module
    - In computing, a loadable kernel module (or LKM) is an object file that contains code to extend the running kernel, or so-called base kernel, of an operating system.
    - LKMs are typically used to add support for new hardware and/or filesystems, or for adding system calls. When the

functionality provided by a LKM is no longer required, it can be unloaded in order to free memory and other resources.

  - Fragmentation Penalty:
    - In computer storage, fragmentation is a phenomenon in which storage space is used inefficiently, reducing capacity or performance and often both.
  - 单内核结构：运行时是一个大的二进制映像，模块间的交互通过直接调用其他模块中的函数来实现
    - 优点：系统运行开销小、效率高
    - 缺点：内核功能的适应性、灵活性和可伸缩性差
    - 因此，Linux内核引入“模块”机制，用来动态装入和删除一个文件系统或设备驱动程序，有效地弥补单内核结构的不足
- The Linux Kernel Module Programming Guide:（只需要学习第1章和第2章）
  - http://tldp.org/LDP/lkmpg/2.6/html/index.html
    - 虽然上文讲的是2.6内核中的module调用方式，但至今基本未变
  - 第2章：从hello-1.c到hello-5.c
    - Here's another exercise for the reader. See that comment above the return statement in init_module()? Change the return value to something negative, recompile and load the module again. What happens?
    - hello-5.c加入了parameters的支持，可以查看/sys/module/hello/parameters/
      - 注意最后一个参数（权限）的实际影响与其用途
  - Makefile
    - `obj-m += hello-1.o`
    - 
    - `all:`
      - `make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules`
    - 
    - `clean:`
      - `make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean`
  - modinfo hello-1.ko
  - sudo insmod hello-1.ko
  - sudo rmmod hello-1.ko
  - dmesg 或者 查看/var/log/syslog可以看到信息
- Two packages: dkms and kmod
  - Dynamic Kernel Module Support (DKMS) is a program/framework that enables generating Linux kernel modules whose sources generally reside outside the kernel source tree. The concept is to have DKMS modules automatically rebuilt when a new kernel is installed.
  - kmod: tools for managing Linux kernel modules
    - aptitude show kmod
    - dpkg -L kmod
    - This package contains a set of programs for loading, inserting, and removing kernel modules for Linux. It replaces module-init-tools.
    - insmod, depmod, rmmod, lsmod, modprobe, modinfo
      - modprobe is a Linux program originally written by Rusty Russell and used to add a loadable kernel module (LKM) to the Linux kernel or to remove a LKM from the kernel. It is commonly used indirectly.
      - lsmod:
        - lsmod is a command on Linux systems which prints the contents of the /proc/modules file. It shows which loadable kernel modules are currently loaded.
        - 例如：查看virtualbox安装的modules
          - lsmod | grep vbox
          - 共三个：vboxsf，vboxvideo，vboxguest
            - vboxguest：the main Guest Additions module
            - vboxsf: the shared folder support module
            - vboxvideo: the OpenGL support module
      - See /proc/modules
- 与module相关的两个重要内容
  - CONFIG_MODULE_FORCE_LOAD and CONFIG_MODULE_FORCE_UNLOAD
  - /proc/kallsyms and EXPORT_SYMBOL()
- Rootkit:
  - A rootkit is a stealthy type of software, typically malicious, designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a computer.
  - The term rootkit is a concatenation of "root" (the traditional name of the privileged account on Unix operating systems) and the word "kit" (which refers to the software components that implement the tool).
  - While loadable kernel modules are a convenient method of modifying the running kernel, this can be abused by attackers on a compromised system to prevent detection of their processes or files, allowing them to maintain control over the system. Many rootkits make use of LKMs in this way.

自学：

- LKM: http://en.wikipedia.org/wiki/Loadable_kernel_module
- Fragmentation: https://en.wikipedia.org/wiki/Fragmentation_(computing)

- DKMS: https://en.wikipedia.org/wiki/Dynamic_Kernel_Module_Support
- modprobe: https://en.wikipedia.org/wiki/Modprobe
- lsmod: https://en.wikipedia.org/wiki/Lsmod
- Rootkit: https://en.wikipedia.org/wiki/Rootkit
- Documentation/kbuild/modules.txt
- Documentation/module-signing.txt

See also:

- Rebuilding A Kernel Module On The Fly: https://wiki.ubuntu.com/Kernel/Dev/KernelModuleRebuild
- 内核代码：samples目录下的源代码

---

## 阶段四：编码风格

- Why coding style matters
  - http://www.smashingmagazine.com/2012/10/25/why-coding-style-matters/
  - http://blog.8thcolor.com/en/2013/05/style-matters/
  - Style is what separates the good from the great.
  - Coding style is made up of numerous small decisions based on the language: (This is by no means an exhaustive list, as coding style can be extremely fine-grained)
    - How and when to use comments,
    - Tabs or spaces for indentation (and how many spaces),
    - Appropriate use of white space,
    - Proper naming of variables and functions,
    - Code grouping an organization,
    - Patterns to be used,
    - Patterns to be avoided.
- Programming style: https://en.wikipedia.org/wiki/Programming_style
  - Programming style is a set of rules or guidelines used when writing the source code for a computer program. It is often claimed that following a particular programming style will help programmers to read and understand source code conforming to the style, and help to avoid introducing errors.
  - Elements of good style
    - Good style is a subjective matter, and is difficult to define.
    - However, there are several elements common to a large number of programming styles
      - the layout of the source code, including indentation;
      - the use of white space around operators and keywords;
      - the capitalization or otherwise of keywords and variable names;
      - the style and spelling of user-defined identifiers, such as function, procedure and variable names;
      - the use and style of comments.
  - Coding conventions
    - Coding conventions are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in this language. These conventions usually cover file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc.
  - Naming convention
    - In computer programming, a naming convention is a set of rules for choosing the character sequence to be used for identifiers which denote variables, types, functions, and other entities in source code and documentation.
  - Indent style:
    - In computer programming, an indent style is a convention governing the indentation of blocks of code to convey the program's structure. This article largely addresses the free-form languages, such as C programming language and its descendants, but can be (and frequently is) applied to most other programming languages (especially those in the curly bracket family), where whitespace is otherwise insignificant. Indent style is just one aspect of programming style.
    - K&R Style -> variants: linux kernel
    - Tool: GNU indent
      - sudo apt-get install indent
      - man indent
  - Comments:
    - In computer programming, a comment is a programmer-readable annotation in the source code of a computer program. They are added with the purpose of making the source code easier to understand, and are generally ignored by compilers and interpreters.
- (BOOK) The Elements of Programming Style
  - The Elements of Programming Style, by Brian W. Kernighan and P. J. Plauger, is a study of programming style, advocating the notion that computer programs should be written not only to satisfy the compiler or personal programming "style", but also for "readability" by humans, specifically software maintenance engineers,

programmers and technical writers.
    - Its lessons are summarized at the end of each section in pithy maxims, such as "Let the machine do the dirty work":
        - 1.Write clearly -- don't be too clever.
        - 2.Say what you mean, simply and directly.
        - 3.Use library functions whenever feasible.
        - 4.Avoid too many temporary variables.
        - 5.Write clearly -- don't sacrifice clarity for efficiency.
        - 6.Let the machine do the dirty work.
        - 7.Replace repetitive expressions by calls to common functions.
        - 8.Parenthesize to avoid ambiguity.
        - 9.Choose variable names that won't be confused.
        - 10.Avoid unnecessary branches.
        - 11.If a logical expression is hard to understand, try transforming it.
        - 12.Choose a data representation that makes the program simple.
        - 13.Write first in easy-to-understand pseudo language; then translate into whatever language you have to use.
        - 14.Modularize. Use procedures and functions.
        - 15.Avoid gotos completely if you can keep the program readable.
        - 16.Don't patch bad code -- rewrite it.
        - 17.Write and test a big program in small pieces.
        - 18.Use recursive procedures for recursively-defined data structures.
        - 19.Test input for plausibility and validity.
        - 20.Make sure input doesn't violate the limits of the program.
        - 21.Terminate input by end-of-file marker, not by count.
        - 22.Identify bad input; recover if possible.
        - 23.Make input easy to prepare and output self-explanatory.
        - 24.Use uniform input formats.
        - 25.Make input easy to proofread.
        - 26.Use self-identifying input. Allow defaults. Echo both on output.
        - 27.Make sure all variables are initialized before use.
        - 28.Don't stop at one bug.
        - 29.Use debugging compilers.
        - 30.Watch out for off-by-one errors.
        - 31.Take care to branch the right way on equality.
        - 32.Be careful if a loop exits to the same place from the middle and the bottom.
        - 33.Make sure your code does "nothing" gracefully.
        - 34.Test programs at their boundary values.
        - 35.Check some answers by hand.
        - 36.10.0 times 0.1 is hardly ever 1.0.
        - 37.7/8 is zero while 7.0/8.0 is not zero.
        - 38.Don't compare floating point numbers solely for equality.
        - 39.Make it right before you make it faster.
        - 40.Make it fail-safe before you make it faster.
        - 41.Make it clear before you make it faster.
        - 42.Don't sacrifice clarity for small gains in efficiency.
        - 43.Let your compiler do the simple optimizations.
        - 44.Don't strain to re-use code; reorganize instead.
        - 45.Make sure special cases are truly special.
        - 46.Keep it simple to make it faster.
        - 47.Don't diddle code to make it faster -- find a better algorithm.
        - 48.Instrument your programs. Measure before making efficiency changes.
        - 49.Make sure comments and code agree.
        - 50.Don't just echo the code with comments -- make every comment count.
        - 51.Don't comment bad code -- rewrite it.
        - 52.Use variable names that mean something.
        - 53.Use statement labels that mean something.
        - 54.Format a program to help the reader understand it.
        - 55.Document your data layouts.
        - 56.Don't over-comment.
- Free-form language: https://en.wikipedia.org/wiki/Free-form_language
    - curly-bracket or curly-brace programming languages
    - In computer programming, a free-form language is a programming language in which the positioning of characters on the page in program text is insignificant. Program text does not need to be placed in specific columns as on old punched card systems, and frequently ends of lines are insignificant. Whitespace characters are used only to delimit tokens, and have no other significance.
- Obfuscation

    - Code Obfuscation 模糊代码；代码混淆；代码混乱；代码迷惑
    - 问题：混乱代码有什么用？
    - In software development, obfuscation is the deliberate act of creating obfuscated code, i.e. source or machine code that is difficult for humans to understand. Like obfuscation in natural language, it may use needlessly roundabout expressions to compose statements.
    - IOCCC（国际C语言混乱代码大赛）
        - https://en.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest
    - Underhanded C Contest
        - underhanded: 卑劣的，秘密的，不光明的
        - https://en.wikipedia.org/wiki/Underhanded_C_Contest

自学：

- Programming style: https://en.wikipedia.org/wiki/Programming_style
- Coding conventions: https://en.wikipedia.org/wiki/Coding_conventions
- Free-form language: https://en.wikipedia.org/wiki/Free-form_language
- The Elements of Programming Style: https://en.wikipedia.org/wiki/The_Elements_of_Programming_Style
- Naming convention: https://en.wikipedia.org/wiki/Naming_convention_(programming)
- Indent sytle: https://en.wikipedia.org/wiki/Indent_style
- Comment: https://en.wikipedia.org/wiki/Comment_(computer_programming)
- Whitespace character: https://en.wikipedia.org/wiki/Whitespace_character
- Tab stop: https://en.wikipedia.org/wiki/Tab_stop
- Obfuscation：https://en.wikipedia.org/wiki/Obfuscation_(software)

See also:

- http://www.smashingmagazine.com/2012/10/25/why-coding-style-matters/
- http://blog.8thcolor.com/en/2013/05/style-matters/
- Online book: "C Style: Standards and Guidelines"
    - http://syque.com/cstyle/index.htm
- IOCCC（国际C语言混乱代码大赛）: https://en.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest
- Underhanded C Contest: https://en.wikipedia.org/wiki/Underhanded_C_Contest

---