同济大学计算机系

计算机系统实验报告

实验三题目：嵌入式 Linux 系统内核编译和改造

学号：2152118

姓名：　史君宝

指导教师：秦国峰

日期：2024 年 5 月 31 日

一、实验环境

操作系统：Windows11

软件系统：VMware Workstation 10.0.3

虚拟机：Ubuntu 12.04

开发板：Nexys 4 DDR Artix-7

开发工具：Vivado 2018.2

二、实验目的

本实验我们需要完成嵌入式Linux系统内核编译和改造,这个任务比较综合。具体的实验内容可以分为以下几个步骤,通过这些步骤逐层深入能够进一步了解系统结构的相关知识。

实验的主要内容如下：

（1）内核编译流程。

（2）内核定制和扩展。

（3）内核配置优化。

（4）虚拟机环境配置。

（5）FPGA 开发板配置。

（6）Bootloader 配置。

（7）存储体系理解。

下面我们具体解释一下：

（1）内核编译流程:深入了解 Linux 内核编译的各个步骤,包括获取内核源码、配置内核参数（通过 make menuconfig）、编译内核（make）以及安装内核模块（make modules_install）等。熟练掌握这些基本操作。

（2）内核定制和扩展:学习如何修改和定制内核源码,添加自定义的驱动程序或功能模块。理解内核功能的扩展机制,增强对内核工作原理的理解。

（3）内核配置优化:根据嵌入式系统的具体硬件和软件需求,对内核配置

进行针对性的优化和裁剪,去除不需要的功能模块,提高系统性能和资源利用率。

（4）虚拟机环境配置:熟练使用虚拟机软件,如 VMware 或 VirtualBox,在虚拟机上部署嵌入式 Linux 开发环境,包括交叉编译工具链、调试嵌入式 Linux 系统内核编译和改造工具等。

（5）FPGA 开发板配置:学习使用 Vivado 等 FPGA 开发工具,了解 FPGA 平台的基本操作,并能够对 FPGA 开发板进行配置和调试。

（6）开发环境调试:在虚拟机或 FPGA 开发板上,调试和验证嵌入式 Linux 系统的正常运行,解决开发过程中遇到的各种问题。

（7）Bootloader 配置:学习使用 Bootloader(如 U-Boot)实现嵌入式 Linux 系统的引导启动,确保系统能够从 SD 卡加载内核并正常启动。

（8）系统功能测试:在启动的 Linux 系统环境中,进行文件管理、内存管理等基本操作的功能测试,验证系统的稳定性和可靠性。

（9）性能评估:根据实际需求,评估系统的性能指标,如启动速度、响应时间等,并进行适当的优化。

（10）存储体系理解:深入理解嵌入式系统中三级存储体系(SD 卡、DDR、Cache)的工作原理和数据传输机制,将理论知识与实践应用相结合。

三、实验过程及内容

（1）配置环境变量并下载相应依赖。

我们按照实验教程，在 bash shell 配置中导入 export PATH=/opt/gcc-4.9.3-64-gnu/bin:$PATH 后编译重启，确保 gcc 已经成功导入。

（2）下载相关的安装包。

```
yuki@DESKTOP-T6P08C2:~$ sudo apt-get update;sudo apt-get install libncurses5-dev,bc,make,gcc;
[sudo] password for yuki:
Get:1 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu jammy InRelease [48.8 kB]
Get:2 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu jammy/stable amd64 Packages [32.8 kB
Get:3 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [32.8 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1475 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1686 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [313 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1933 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [254 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1876 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [328 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1076 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [247 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [42.7 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.4 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.2 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.2 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [318 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [854 kB]
```

（3）将内核源码拷贝后解压

我们使用下面的语句进行操作：

```
make ARCH=mips CROSS_COMPILE=mips64el-linux- menuconfig
make ARCH=mips CROSS_COMPILE=mips64el-linux-
make ARCH=mips CROSS_COMPILE=mips64el-linux  modules_install
```

最后一步的目的是在虚拟机/lib/modules/文件夹下生成 2k 内核模块的文件夹，3 开头那个文件夹就是，整个打包，和生成的 vmlinuz 一起拷出来，就成为了内核镜像和模块。

```
    Machine selection  --->
    Endianness selection (Little endian)  --->
    CPU selection  --->
    Kernel type  --->
    General setup  --->
[ ] Provide system-wide ring of trusted keys
[ ] Provide system-wide ring of blacklisted keys
[*] Enable loadable module support  --->
[*] Enable the block layer  --->
    Bus options (PCI, PCMCIA, EISA, ISA, TC)  --->
    Executable file formats  --->
    Power management options  --->
-*- Networking support  --->
    Device Drivers  --->
    Firmware Drivers  --->
    File systems  --->
    Kernel hacking  --->
    Security options  --->
-*- Cryptographic API  --->
    Library routines  --->
[ ] Virtualization  --->
[ ] Enables more stringent kabi checks in the macros
```

修改配置：

```
Device Drivers  --->
     Graphics support  --->
        Support for frame buffer devices  --->
            Loongson Frame Buffer Support

Device Drivers  --->
     SPI support  --->
        <*>   Loongson SPI Controller Support


Device Drivers  --->
     Graphics support  --->
      <*> LOONGSON VGA DRM
      [ ]   use platfrom device
```

（4）替换内核为龙芯派出厂的内核，自己编译内核后替换原来的内核。

需要编译两个内核文件，一个带 ramdisk 的内核镜像，一个不带 ramdisk 的内核镜像。在 General setup-->Initial RAM 。 如下图所示选择编译或者不编译带 ramdisk。其中不带 ramdisk 的内核镜像是用于替换使用的。

```
  ()  Cross-compiler tool prefix
  ()  Local version - append to kernel release
  [ ] Automatically append version information to the version string
      Kernel compression mode (Gzip)  --->
  ((none)) Default hostname
  [*] Support for paging of anonymous memory (swap)
  [*] System V IPC
  [*] POSIX Message Queues
  [*] open by fhandle syscalls
  [*] Auditing support
  [*]   Enable system-call auditing support
      IRQ subsystem  --->
      Timers subsystem  --->
      CPU/Task time and stats accounting  --->
      RCU Subsystem  --->
  <*> Kernel .config support
  [*]   Enable access to .config through /proc/config.gz
  (18) Kernel log buffer size (16 => 64KB, 17 => 128KB)
  -*- Control Group support  --->
  [ ] Checkpoint/restore support
  [*] Namespaces support  --->
  [ ] Require conversions between uid/gids and their internal representation
  [*] Automatic process group scheduling
  [ ] Enable deprecated sysfs features to support old userspace tools
  [*] Kernel->user space relay support (formerly relayfs)
  [*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
  (/home/zhangyan/ramdisk_0.cpio) Initramfs source file(s)
  (0)     User ID to map to 0 (user root)
  (0)     Group ID to map to 0 (group root)
  [*]   Support initial ramdisks compressed using gzip
  [*]   Support initial ramdisks compressed using bzip2
  [*]   Support initial ramdisks compressed using LZMA
  [*]   Support initial ramdisks compressed using XZ
  [ ]   Support initial ramdisks compressed using LZO
        Built-in initramfs compression mode (None)  --->
  [ ] Optimize for size
  -*- Configure standard kernel features (expert users)  --->
```

（5）然后保存退出，执行./mymake vmlinuz（开始编译带文件系统的内核）。

同样把不带 ramdisk 也编译一遍，将 vmlinuz 拷贝到 U 盘中，用于替换的内核。

```
scripts/kconfig/conf --silentoldconfig Kconfig
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/qrwlock.h
  CHK     include/generated/qrwlock_api_smp.h
  CHK     include/generated/qrwlock_types.h
  CHK     kernel/qrwlock_gen.c
  CHK     lib/qrwlock_debug.c
  CC      scripts/mod/devicetable-offsets.s
  GEN     scripts/mod/devicetable-offsets.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTLD  scripts/mod/modpost
  CHK     include/generated/utsrelease.h
  Checking missing-syscalls for N32
  CALL    scripts/checksyscalls.sh
  Checking missing-syscalls for O32
  CALL    scripts/checksyscalls.sh
  CALL    scripts/checksyscalls.sh
  CC      init/main.o
  CHK     include/generated/compile.h
  CC      init/do_mounts.o
  LD      usr/built-in.o
  CC      init/do_mounts_rd.o
  CC      arch/mips/kernel/prom.o
  CC      init/noinitramfs.o
  CC      arch/mips/kernel/setup.o
  CC      arch/mips/loongson2/init.o
  CC      arch/mips/loongson2/setup.o
  LD      init/mounts.o
  LD      init/built-in.o
  CC      kernel/sysctl.o
  LDS     arch/mips/kernel/vmlinux.lds
  LD      arch/mips/kernel/built-in.o
  DTC     arch/mips/loongson2/dts/2k1000_board.dtb
```

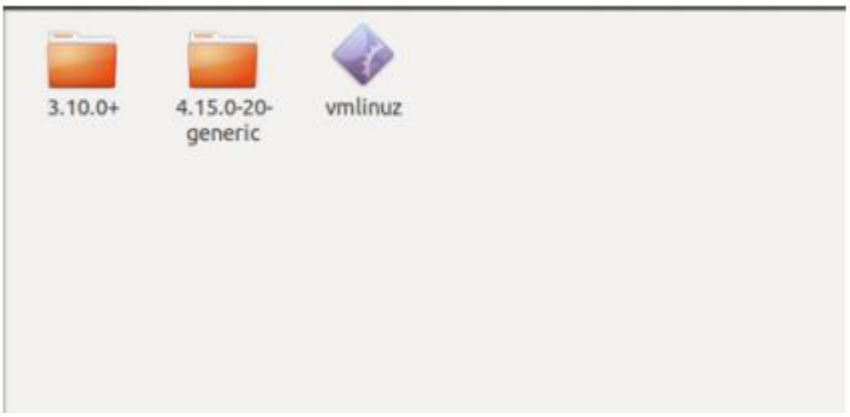
（6）最后工作。

然后按 c 进入到 pmon 命令行下，load 刚编译的内核：

1. load /dev/fs/ext2@usb0/vmlinuz-fs (从 U 盘 load)
2. load tftp://192.168.124.140/vmlinuz-fs (从 tftp 服务器 load)。
3. 在内核命令行下插入 U 盘，将要替换的内核拷贝出来：
4. 执行命令：mount /dev/sdb /mnt (假如 U 盘识别的是 sdb)
5. cp /mnt/vmlinuz / umount /mnt 。

然后开始挂载硬盘 使用 mount /dev/sda1 /mnt 查看/mnt/boot/boot.cfg 将 /mnt 或者/mnt/boot/下的 vmlinuz 替换为我们的 vmlnuz(这里最好将之前的 vmlinuz 保存一下)。

最后使用 sync umount /mnt 重启。



四、实验小结

通过参与内核编译流程的学习实验,我获得了对Linux内核深入了解的机会。首先，我学习了如何获取Linux内核的源代码，通常是从kernel.org下载。然后，我需要通过配置内核参数来选择所需的功能模块，这要求我熟练地使用命令如make menuconfig。最后，我了解了编译和生成内核镜像的过程，并熟悉了其中涉及的各种编译选项。通过这些步骤，我对Linux内核的编译流程和内部结构有了更深入的理解。

在编译过程中，我学会了根据嵌入式系统的需求进行内核配置的定制。举例来说，对于网络设备，我可以选择启用适当的网卡驱动模块，并配置相关的网络协议栈功能。对于多媒体设备，我可以启用音视频编解码模块。这种定制配置可以显著提高系统性能和资源利用率。

此外，我也尝试对内核源代码进行了一些修改。我添加了自定义的驱动程序或调整了内核参数，这进一步提升了我的内核开发能力，并加深了对Linux内核工作原理的理解。我还尝试将内核移植到不同的硬件平台上，这需要进行硬件抽象层（HAL）的适配。通过这些实践，我对嵌入式系统移植和适配有了更深入的了解。

在编译和运行过程中，我遇到了一些问题，例如内核模块依赖错误或启动失败等。为了解决这些问题，我需要查阅内核文档，使用调试工具，并进行问题分析和解决。这个过程提升了我的技术调试能力,并增强了我独立解决问题的信心。

通过这个实验，我对嵌入式 Linux 系统有了更全面的认识。我意识到相较于通用的 Linux 桌面系统，嵌入式系统需要根据硬件资源受限、功耗和实时性等特点进行定制和优化。内核的裁剪、模块化管理以及对硬件的适配都是关键因素。这些经验将有助于我在未来的嵌入式系统开发领域进行更好的设计和实现。

总的来说，这个实验提供了一次宝贵的实践机会，不仅加深了我对 Linux 内核的理解，还培养了我的系统思维和问题解决能力。这些都是成为优秀嵌入式系统工程师所必备的关键技能。