CSC0016 - Advanced Operating Systems

HW1\_Build Linux Kernel 2.6.32

NTUST\_M11207521陳俊博

1. **環境設置**

在硬體方面，筆電的作業系統為 Windows 11，搭載 AMD Ryzen 5 Pro 7540U 處理器，擁有 32GB 記憶體與 1TB 的硬碟儲存空間。在軟體環境中使用 VirtualBox 7.0.18 作為虛擬化平台，並在其中運行 Ubuntu 16.04.7 LTS (Xenial Xerus) 作業系統。虛擬機器配置為使用 4 顆虛擬 CPU 和 16GB 記憶體，並設定了 100GB 的虛擬硬碟。另外，在這個虛擬化環境中原先所使用的 Linux 核心版本為 4.15.0-112-generic。

一張含有 文字, 軟體, 多媒體軟體, 螢幕擷取畫面 的圖片

自動產生的描述

圖1 初始環境

1. **核心編譯過程**
2. **先決條件**

首先，需要移除現有的 GCC 編譯器並安裝特定版本的 GCC 4.9。這是因為編譯較舊版本的 Linux 核心需要降低編譯器版本，確保與核心原始碼的相容性。

1. 移除現有的 GCC：使用 sudo apt-get remove gcc-5 移除 GCC 5，並使用 sudo apt-get autoremove 移除相關的相依套件。
2. 安裝 GCC 4.9：透過 sudo apt-get install gcc-4.9 g++-4.9 安裝 GCC 4.9 和 G++ 4.9。
3. 更新 Alternatives 設定：刪除現有的 update-alternatives 設定，並將 GCC 和 G++ 指向新的版本 4.9。
4. 驗證 GCC 和 G++ 版本：使用gcc -version與g++ -version確保 GCC 和 G++ 的版本已正確更新為 4.9。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

圖2 GCC版本

1. **預建需求調整(請先從「IV. 開始編譯」開始，進到**linux-2.6.32**目錄後再回來這邊)**

在開始編譯 Linux 2.6.32 核心之前，需要修改一些核心原始碼文件，主要原因為GDB版本不同，建議先修改再進行編譯可以省很多時間。

1. 修改 Makefile：在 linux-2.6.32/arch/x86/vdso/Makefile 中，將 -m elf\_x86\_64 改為 -m64，將 -m elf\_i386 改為 -m32。
2. 調整網路驅動程式：註解掉 linux-2.6.32/drivers/net/igbvf/igbvf.h 中的 struct page \*page;。
3. 修正 Perl 腳本：在 linux-2.6.32/kernel/timeconst.pl 中，將defined(@val) 改為 @val。
4. **核心選項配置(請先從「IV. 開始編譯」開始，準備執行**make menuconfig**時再回來這邊)**

使用 make menuconfig 進入核心配置選單時，需調整核心的編譯選項。

1. Kernel Hacking

* 開啟選項：
  + [\*] Compile the kernel with debug info：開啟除錯資訊，這可以在除錯時提供更詳細的訊息。
  + -\*- Compile the kernel with frame pointers：設定編譯時使用框架指標，有助於除錯工具在呼叫堆疊時提供更完整的資訊。
* 關閉選項：
  + [ ] Write protect kernel read-only data structures：不啟用寫保護核心唯讀資料結構，方便後續更改核心程式碼。

1. Processor Type and Features

* 關閉選項：
  + [ ] Paravirtualized guest support (NEW)：關閉虛擬化客戶支援。

1. **開始編譯**

下載核心原始碼、解壓縮、安裝所需工具以及最終編譯核心。

1. 下載和解壓縮核心原始碼：使用 wget 下載 Linux 2.6.32 核心原始碼，執行wget https://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.32.tar.gz，並用 tar 解壓縮。
2. 進入核心目錄：切換到核心原始碼所在的目錄。
3. 安裝編譯工具：使用 sudo apt-get install 安裝編譯核心所需的開發工具和套件，包刮libncurses5-dev、build-essential 和 kernel-package。
4. 清理環境：執行 make mrproper 以確保編譯環境的整潔，刪除配置文件（.config）及產生的核心映像檔案。
5. 開啟核心設定選單：通過 make menuconfig 開啟核心配置選單，以調整核心的選項。
6. 編譯核心：執行 make 根據 Makefile 的設定進行核心編譯，需要一些時間。

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述

圖3 Linux核心編譯完成

1. **使用QEMU和BusyBox 開啟linux kernel 2.6.32**
2. **先決條件**

首先需要安裝 QEMU，這是一個開放原始碼的虛擬機器模擬器和虛擬化工具，允許在虛擬環境中運行不同的作業系統核心。

* 安裝 QEMU：sudo apt-get install qemu qemu-system，會安裝 QEMU 和相關的系統模組，為後續模擬環境做準備。

1. **BusyBox 配置(請先跳到「III. 開始編譯」，執行到make menuconfig時再回來這邊)**

BusyBox 是一個整合了多個 Unix 工具的小型軟體，用於在嵌入式系統中提供精簡的 Linux 環境。需要配置 BusyBox，使得與 Linux 2.6.32 核心相容。

1. BusyBox Configuration：

* 進入 BusyBox 的 menuconfig 配置選單，選擇 "Settings -> Build Options"。
* 開啟 [\*] Build static binary (no shared libs)。

1. Linux System Utilities：

* 關閉 Support mounting NFS file systems on linux < 2.6.23 (NEW)。

1. Networking Utilities：

* 關閉 inetd。

1. **開始編譯**

下載、解壓縮並編譯 BusyBox，用作核心的初始化檔案系統。

1. 下載和解壓縮 BusyBox：使用wget 下載 BusyBox 1.31.0，執行wget https://busybox.net/downloads/busybox-1.31.0.tar.bz2，並使用tar 解壓縮。
2. 進入 BusyBox 目錄：切換到 BusyBox 原始碼所在的目錄。
3. 配置 BusyBox：執行 make menuconfig 進入 BusyBox 配置選單，調整配置以符合需求。
4. 編譯並安裝 BusyBox：執行 make install，這會將 BusyBox 安裝到 \_install 目錄中，作為核心初始化的根檔案系統。

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述

圖4 BusyBox編譯完成

1. **後置建置**

配置根檔案系統的目錄結構和啟動腳本，以便核心能夠正確啟動和掛載必要的虛擬檔案系統。

1. 建立目錄結構：在 \_install 目錄中建立 proc、sys、dev、etc 和 etc/init.d 目錄，執行mkdir proc sys dev etc etc/init.d，為根檔案系統必要目錄。
2. 編寫啟動腳本：創建 etc/init.d/rcS 檔案，添加啟動腳本內容以掛載 proc 和 sysfs 檔案系統，並啟動 mdev。添加以下程式碼：
3. #!/bin/sh
4. mount -t proc none /proc
5. mount -t sysfs none /sys
6. /sbin/mdev -s
7. 設置腳本為可執行：使用 chmod +x 命令將腳本設為可執行，確保它能在啟動時運行。執行chmod +x etc/init.d/rcS。
8. 在\_install下執行：find . | cpio -o --format=newc > ../rootfs.img
9. **測試環境**

使用QEMU啟動編譯好的Linux Kernel和BusyBox根檔案系統，以驗證整個環境是否正常工作。

1. 啟動核心：進入 linux-2.6.32/ 目錄，使用 qemu-system-x86\_64 命令來啟動編譯好的核心，並使用 -initrd 參數指定 BusyBox 產生的根檔案系統映像。執行qemu-system-x86\_64 -m 512 -kernel arch/x86\_64/boot/bzImage -initrd ../busybox-1.31.0/rootfs.img -append "root=/dev/ram rdinit=/sbin/init console=ttyS0" -nographic。使用 -nographic 參數，不開啟圖形介面。
2. 驗證核心版本：啟動後執行 uname -r會顯示 2.6.32，表示成功啟動指定版本的 Linux 核心。

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

圖5 確認核心版本

1. **遇到的困難點**

雖然已經成功地編譯了Linux核心，但不清楚如何在一個虛擬環境中使用不同於主機系統核心版本的自訂核心，讓我進入了沉思者的狀態：如何才能在不干擾現有系統的前提下，運行並測試這個新編譯的核心？所以就開始上網找資料，終於在凌晨五點的時候找到一份關於使用QEMU和BusyBox來模擬Linux核心環境的文章。



圖6 沉思者雕像

現在就來介紹一下吧。QEMU 是一個虛擬機模擬器，它可以模擬不同的硬體環境，並允許運行自訂的 Linux 核心。而 BusyBox 是一個精簡的 Unix 工具集，能夠在嵌入式系統或簡化的 Linux 環境中提供基本的命令和功能。

使用 QEMU的話我可以指定我們自己編譯的 Linux 核心，並搭配 BusyBox 作為根檔案系統，從而創建一個獨立的虛擬機環境。這個環境不依賴於主機的核心版本，以便讓我能夠測試和驗證新編譯的 Linux 2.6.32 核心是否運行正常。所以，使用這兩套軟體可以輕鬆地完成這次的作業。

一張含有 標誌, 圖形, 字型, 設計 的圖片

自動產生的描述一張含有 容器, 箱子, 硬紙盒, 紙板 的圖片

自動產生的描述

圖7 QMEU與BusyBox 圖示

1. **HackMD筆記**

Build Linux Kernel 2.6.32: <https://hackmd.io/@CHUN-PO-CHEN/SkWNhO460>

1. **參考資料**
2. Love, R. (2010). Linux Kernel Development (Third Edition). Addison Wesley. <https://github.com/jyfc/ebook/blob/master/03_operating_system/Linux.Kernel.Development.3rd.Edition.pdf>.
3. Linux Kernel 2.6.32 Build, <https://hackmd.io/@qqgnoe466263/SkN1lw0QB?utm_source=preview-mode&utm_medium=rec>.
4. QEMU + Busybox 模拟 Linux 内核环境, <https://www.v4ler1an.com/2020/12/qemu/>.
5. GCC version 4.9 has no installation candidate, <https://stackoverflow.com/questions/62177887/gcc-version-4-9-has-no-installation-candidate>.
6. Linux kernel compile error elf\_x86\_64 missing, <https://stackoverflow.com/questions/22662906/linux-kernel-compile-error-elf-x86-64-missing>.
7. Can't use 'defined(@array)' (Maybe you should just omit the defined()?)at kernel/timeconst.pl, <https://blog.csdn.net/qq_37363920/article/details/105391961>.
8. Kernel doesn't support PIC mode for compiling ?, <https://askubuntu.com/questions/851433/kernel-doesnt-support-pic-mode-for-compiling?noredirect=1&lq=1>.
9. cc1: error: code model kernel does not support PIC mode, <https://blog.csdn.net/jasonLee_lijiaqi/article/details/84651138>.
10. Linux内核调试, <https://xz.aliyun.com/t/2024?time__1311=n4%2Bxni0%3DoQwxBDmxmqGNDQaae7T3zce6mAoD>.