

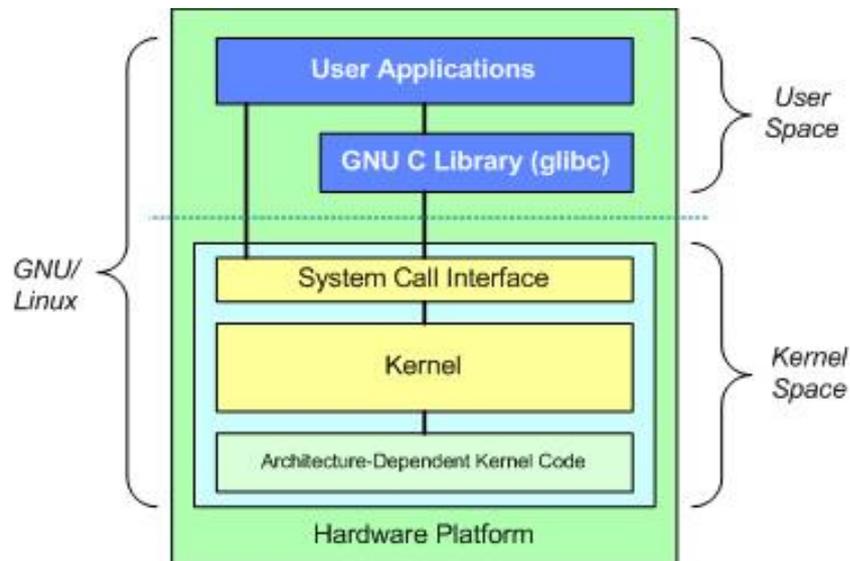


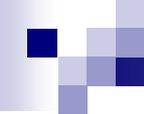
# Computer Architecture

## **LAB2: Building Linux Operating System Environment**

# Linux Kernel Introduction

- The Linux kernel is a monolithic Unix-like computer operating system kernel.
- It is an open source software that suits for researchers to develop an experiment environment with OS without any cost.





# Initrd ramdisk Introduction

- Initrd ramdisk (initrd) is a scheme for loading a temporary root file system(RFS) into memory.
- In booting process, kernel will start to prepare environment before enter init process of Linux (not init in initrd) by programs in initrd.

# Building ARM Linux Kernel

- Get Linux kernel source code and ARM patch
  - <ftp://ftp.kernel.org/pub/linux/kernel/v2.6/>
  - <http://arm.com/community/software-enablement/linux.php>
  - 在Linux OS Downloads選項中，可以找到linux 2.6.38的patch
  - 直接下載下來的Linux必須經過此patch修正後才能正常運作
  - 注：此頁教學可直接使用教材檔案(linux-kernel資料夾)，已經附上patch過的linux-2.6.38-patched.tar.gz檔案，同學不用再下載Linux kernel也不需要Patch
- Patch
  - 在家目錄下新增linux-kernel資料夾並將下載的kernel與patch放入此資料夾中
  - 解壓縮Linux kernel後將patch檔放入解壓縮後的linux kernel資料夾
  - # patch -p1 < {patch-file-name}
  - 此時Linux kernel才可編譯成可用的Image檔

# Building ARM Linux Kernel(Cont.)

## ■ Configure

- 更新完kernel source code之後，我們必須設定linux kernel的執行平台環境，由於每個平台的環境不盡相同，我們直接提供realview eb最簡單的設定檔(config-2.6.38-realview-arm1136)。
- 將config-2.6.38-realview-arm1136檔案放進linux kernel資料夾，並更改檔名
- # cp config-2.6.38-realview-arm1136 .config
- # mv .config linux-2.6.38
- linux-2.6.38]# make ARCH=arm menuconfig(選擇EXIT即可離開設定畫面)
- 檔案結構示意圖：(上方指令僅供參考，只要符合此檔案結構圖即可)

```
linux-kernel
└─┬─ linux-2.6.38
   └─┬─ .config
```

## ■ Compile

- linux-2.6.38]# make ARCH=arm CROSS\_COMPILE=arm-none-linux-gnueabi-
- 編譯前必須檢查是否有export cross compiler的路徑
- 編譯完成之後，我們所要使用的zImage檔將會產生在linux-2.6.38/arch/arm/boot裡面，之後我們會透過此檔案來進行Linux作業系統開機的功能。

# Building Initrd ramdisk

## ■ Get Busybox

- <http://busybox.net/>
- We use busybox 1.20.2 as example

## ■ Configure and compile

- 和kernel一樣，busybox也需要先經過設定後才可以編譯。我們一樣提供最簡單的config檔(config-busybox-1.20.2)，記得將config-busybox-1.20.2 修改檔名成.config。
- 將此config檔放入解壓縮後的busybox資料夾
- 檔案結構示意圖：

```
busybox
├── busybox-1.20.2
│   └── .config
```

- busybox-1.20.2]# make ARCH=arm menuconfig
- 設定結束後即可進行編譯
- busybox-1.20.2]# make ARCH=arm CROSS\_COMPILE=arm-none-linux-gnueabi-
- busybox-1.20.2]# make install CROSS\_COMPILE=arm-none-linux-gnueabi-

# Building Initrd ramdisk (Cont.)

## ■ File system setting

- 安裝成功之後，busybox-1.20.2資料夾下會多一個”\_install”資料夾，裡面只包含最基本的資料夾(bin,sbin,usr)還有執行檔linuxrc。此執行檔為linux kernel開機完成後所要執行的第一個程式。我們先將其名字改為init。
- 接著要**新增**必要的資料夾：**dev, etc, lib, opt, proc, sys, tmp, var**
- 檔案結構示意圖：

```
busybox
├── busybox-1.20.2
│   └── _install
│       ├── bin,sbin,usr,dev,etc,lib,opt,proc,sys,tmp,var
```

- **注**：教材有附上**etc-lib.tar.gz**，解壓縮之後有**etc**與**lib**資料夾可直接使用。
- 其中最重要的就是**etc**，裡面放置著各個必要的環境設定檔。
- (以下為**lib**資料夾內容詳細說明，可自行練習)
- **lib**內則是要放置**cross compiler**的函示庫檔，也就是**LAB1**中所提到的**qemu-arm**執行時後面必須接上**-L {library-path}**。將**library-path**中所有的\*.so檔案複製到**lib**資料夾下即可。

# Building Initrd ramdisk (Cont.)

## ■ Build Initrd ramdisk

- 按照上述步驟編譯完成之後，我們即可將這些檔案與資料夾壓縮成 Initrd ramdisk 以提供開機使用。
- 對於設定檔有興趣的，可以繼續研究 etc 中各個檔案。
- 注意: /etc/init.d 資料夾中的 rcS 檔案為 init 執行後會去讀入的 script 檔，也是整個環境的初始設定檔。因此必須修改檔案格式為可執行檔 (# chmod 755 rcS)
- # cd \_install
- \_install]# find . | cpio -H newc -o > ../initrd
- \_install]# gzip ../initrd
- busybox-1.20.2 資料夾下產出 initrd.gz，此檔即為我們所要使用的 Initrd ramdisk。
- 檔案結構示意圖：

```
busybox
├── busybox-1.20.2
│   └── _install
│       └── initrd.gz
```

# Running Linux System on QEMU

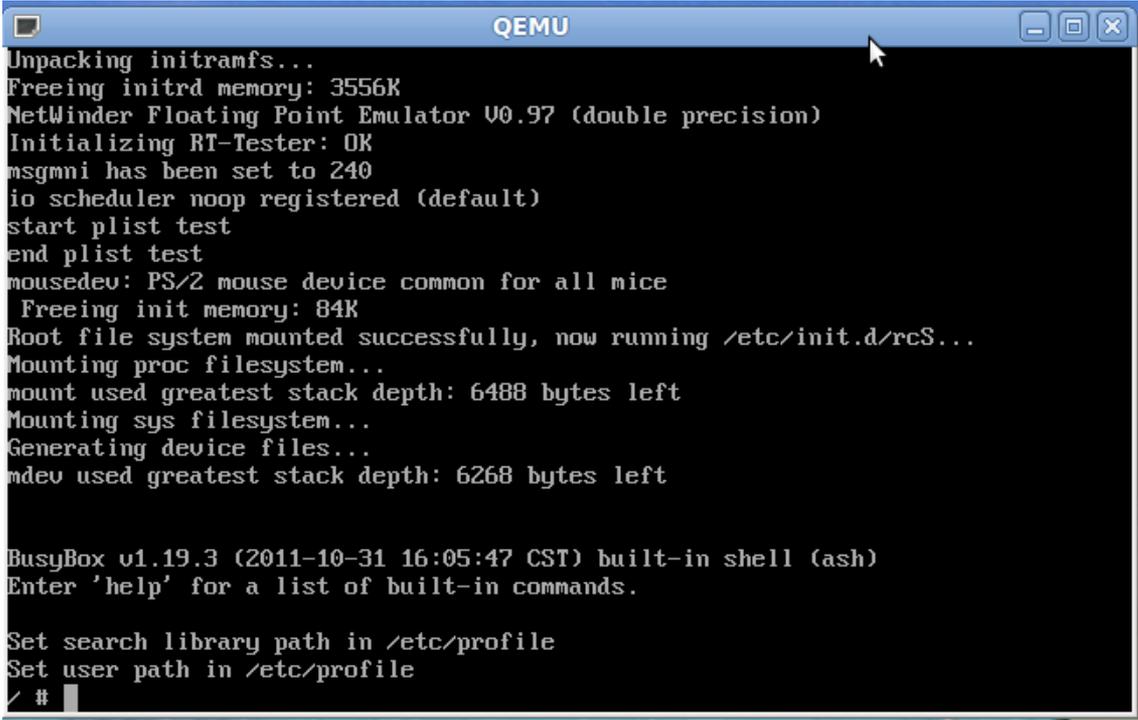
- Necessary Files: `qemu-system-arm`, `zImage`, `initrd.gz`
  - 在經過前面的步驟後，我們可以取得這些必要的檔案。先新增一個資料夾：`qemu-test`，然後將這些必要檔案擺放到此資料夾中(可用複製或是使用 `ln -s` 做鏈結)。
  - 檔案路徑：
    - `qemu-system-arm`: `qemu-bin/bin`
    - `zImage`: `linux-2.6.38/arch/arm/boot/`
    - `Initrd.gz`: `busybox-1.20.2/`

# Running Linux System on QEMU

- `qemu-test]# ./qemu-system-arm -M realview-eb -kernel zImage -initrd initrd.gz -cpu arm1136`
  - 上述指令代表意義為，我們選取了realview emulation board當作模擬平台，linux kernel則為我們所編譯出來的zImage，root file system也是我們編譯的initrd.gz，最後平台上的CPU我們選擇了ARM1136。
  - 此外根據不同的平台與Linux核心設定，模擬所下的參數也會不一樣。

# Running Linux System on QEMU(Cont.)

- 由於我們的linux kernel設定將輸出導向serial port，在QEMU模擬器中，第一輸出畫面預設是LCD panel，所以我們必須將畫面轉移到serial port。在QEMU執行視窗中輸入“ CTRL+ALT+3”。



```
QEMU
Unpacking initramfs...
Freeing initrd memory: 3556K
NetWinder Floating Point Emulator V0.97 (double precision)
Initializing RT-Tester: OK
msgmni has been set to 240
io scheduler noop registered (default)
start plist test
end plist test
mousedev: PS/2 mouse device common for all mice
Freeing init memory: 84K
Root file system mounted successfully, now running /etc/init.d/rcS...
Mounting proc filesystem...
mount used greatest stack depth: 6488 bytes left
Mounting sys filesystem...
Generating device files...
mdev used greatest stack depth: 6268 bytes left

BusyBox v1.19.3 (2011-10-31 16:05:47 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

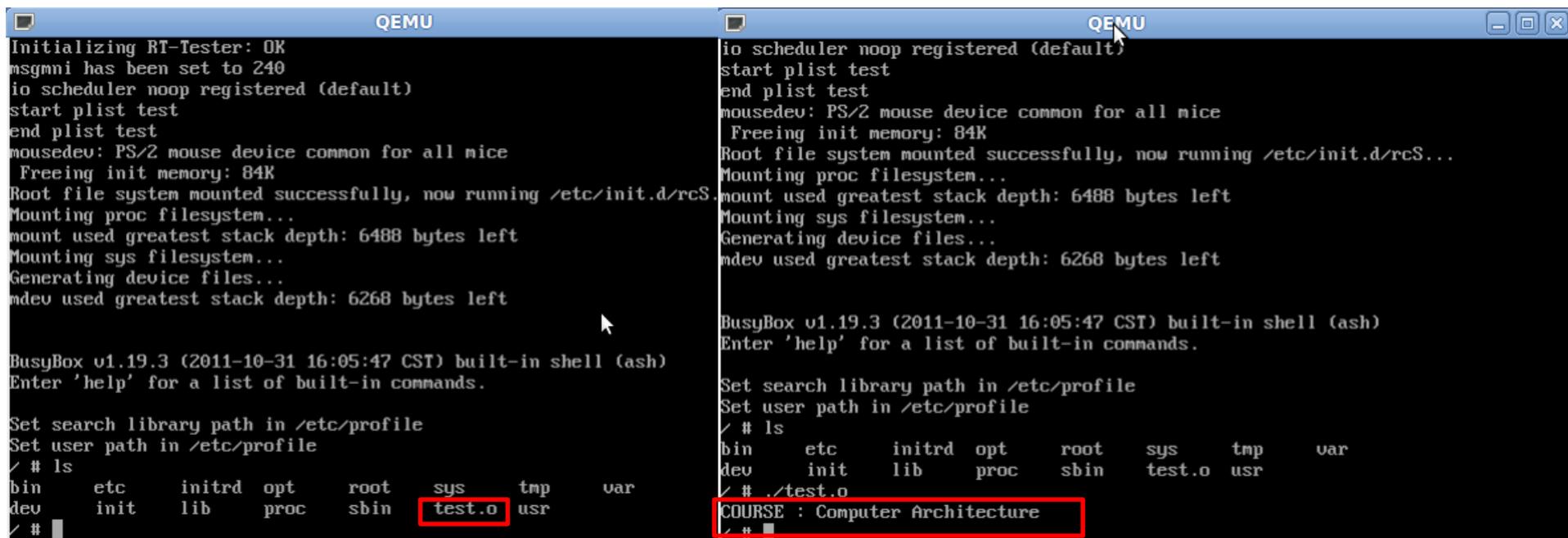
Set search library path in /etc/profile
Set user path in /etc/profile
/ #
```

# Running Application in Linux System

- 前面我們提過採用qemu-arm執行cross compiler編譯出來的程式。現在，我們將這隻程式放入busybox的\_install資料夾中並且再次產出initrd.gz
  - 必須注意\_install資料夾中是否有前次殘留的initrd.gz，必須移除，否則會造成initrd.gz過大的情況。
  - 執行結果如下張投影片所示。
  - 必須注意lib內是否有必要的\*.so檔，因為我們編譯的程式是採用動態函示庫編譯。

# Execution Result

- 執行結果如下



The image shows two side-by-side terminal windows titled "QEMU". The left window displays the initial boot sequence, including messages like "Initializing RT-Tester: OK", "msgmni has been set to 240", and "Root file system mounted successfully, now running /etc/init.d/rcS...". It ends with a BusyBox shell prompt. The right window shows the continuation of the boot process, including "io scheduler noop registered (default)", "start plist test", "end plist test", and "mount used greatest stack depth: 6488 bytes left". It also shows the execution of "ls" and "cat /test.o" commands. The output of "ls" shows a directory listing with "test.o" highlighted in red. The output of "cat /test.o" shows "COURSE : Computer Architecture", also highlighted in red.

```
Initializing RT-Tester: OK
msgmni has been set to 240
io scheduler noop registered (default)
start plist test
end plist test
mousedev: PS/2 mouse device common for all mice
Freeing init memory: 84K
Root file system mounted successfully, now running /etc/init.d/rcS...
Mounting proc filesystem...
mount used greatest stack depth: 6488 bytes left
Mounting sys filesystem...
Generating device files...
mdev used greatest stack depth: 6268 bytes left

BusyBox v1.19.3 (2011-10-31 16:05:47 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

Set search library path in /etc/profile
Set user path in /etc/profile
/ # ls
bin      etc      initrd  opt      root    sys      tmp      var
dev      init    lib     proc     sbin    test.o  usr

/ #
```

```
io scheduler noop registered (default)
start plist test
end plist test
mousedev: PS/2 mouse device common for all mice
Freeing init memory: 84K
Root file system mounted successfully, now running /etc/init.d/rcS...
Mounting proc filesystem...
mount used greatest stack depth: 6488 bytes left
Mounting sys filesystem...
Generating device files...
mdev used greatest stack depth: 6268 bytes left

BusyBox v1.19.3 (2011-10-31 16:05:47 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

Set search library path in /etc/profile
Set user path in /etc/profile
/ # ls
bin      etc      initrd  opt      root    sys      tmp      var
dev      init    lib     proc     sbin    test.o  usr

/ # ./test.o
COURSE : Computer Architecture

/ #
```