

Môn học Lập trình hệ nhúng

GV: Phạm Văn Thuận

Bộ môn Kỹ thuật Máy tính

Viện CNTT&TT- ĐH BKHN

email: thuanpv@soict.hut.edu.vn

Website: <http://soict.hut.edu.vn/~thuanpv>



Mục tiêu môn học

- Sau khi kết thúc môn học này, sinh viên có thể
 - Lập trình vào ra căn bản và nâng cao trên Linux
 - Trình bày được cơ chế lập trình driver cho thiết bị trên Linux
 - Lập trình giao diện đồ họa sử dụng nền tảng QT trên Linux
 - Lập trình truyền thông qua Ethernet, USB 3G
 - Nắm bắt các công nghệ mới: công nghệ định vị GPS, định vị quán tính, công nghệ mạng cảm biến không dây, lập trình iPhone, Android



Nội dung khóa học

Chương 1. Cài đặt, tùy biến hệ điều hành nhúng Linux

Chương 2. Lập trình vào ra căn bản trên Linux

Chương 3. Lập trình vào ra nâng cao trên Linux

Chương 4. Các kỹ thuật lập trình nâng cao

Chương 5. Lập trình device driver trên Linux

Chương 6. Lập trình giao diện đồ họa trên Linux sử dụng nền tảng QT

Chương 7. Lập trình mạng trên nền nhúng

Semimar: công nghệ định vị GPS, định vị quán tính, mạng cảm biến không dây, lập trình iPhone, android



Chương 1: Cài đặt, tùy biến HĐH Linux

- Giới thiệu các thành phần cơ bản của hệ điều hành nhúng Linux
- Cài đặt hệ điều hành nhúng Linux trên KIT micro2440
- Cấu hình, tùy chỉnh, biên dịch nhân hệ điều hành nhúng Linux

cuu duong than cong . com



Chương 2: Lập trình vào ra căn bản

- Xây dựng môi trường lập trình
- Lập trình giao tiếp led đơn, nút nhấn

cuu duong than cong . com

cuu duong than cong . com



Chương 3: Lập trình vào ra nâng cao

- Lập trình giao tiếp cổng COM theo chuẩn RS232
- Lập trình giao tiếp cổng USB
- Lập trình ghép nối ADC

cuu duong than cong . com

cuu duong than cong . com



Chương 4: Các kỹ thuật lập trình nâng cao

- Giới thiệu cơ chế đa tiến trình, đa luồng và giao tiếp giữa các tiến trình, các luồng
- Lập trình xử lý đa tiến trình
- Lập trình xử lý đa luồng
- Xử lý xung đột dữ liệu





Chương 5: Lập trình Driver trên Linux

- Giới thiệu cơ chế quản lý driver trên Linux, các thành phần cơ bản của Driver
- Lập trình tạo Driver điều khiển Led 7 thanh

cuu duong than cong . com

cuu duong than cong . com



Chương 6: Lập trình giao diện QT

- Giới thiệu về nền tảng QT
- Lập trình giao diện với các điều khiển cơ bản trên QT
- Lập trình xử lý đồ họa, âm thanh trên QT





Chương 7: Lập trình mạng trên nền nhúng

- Thư viện lập trình mạng trên QT
- Lập trình gửi nhận dữ liệu qua mạng Ethernet
- Cài đặt, kết nối KIT micro2440 với USB 3G
- Lập trình gửi, nhận tin nhắn qua mạng 3G
- Lập trình truyền hình ảnh qua mạng 3G

cuu duong than cong . com



- Giới thiệu công nghệ định vị dựa trên GPS
- Giới thiệu công nghệ định vị quán tính
- Giới thiệu công nghệ mạng cảm biến không dây
- Giới thiệu lập trình iPhone
- Giới thiệu lập trình Android

cuu duong than cong . com



Tài liệu tham khảo

▪ Tài liệu tham khảo chính:

- Micro2440 User Manual
- S3C2440 Datasheet
- Beginning Linux Programming
- Advanced Linux Programming
- Linux Device Driver
- C++ GUI programming with QT



Chương 1

1.1. Tổng quan hệ điều hành nhúng Linux

1.2. Cài đặt hệ điều hành nhúng Linux

1.3. Tùy biến và biên dịch nhân hệ điều hành
nhúng Linux

cuu duong than cong . com

cuu duong than cong . com



1.1. Tổng quan hệ điều hành nhúng Linux

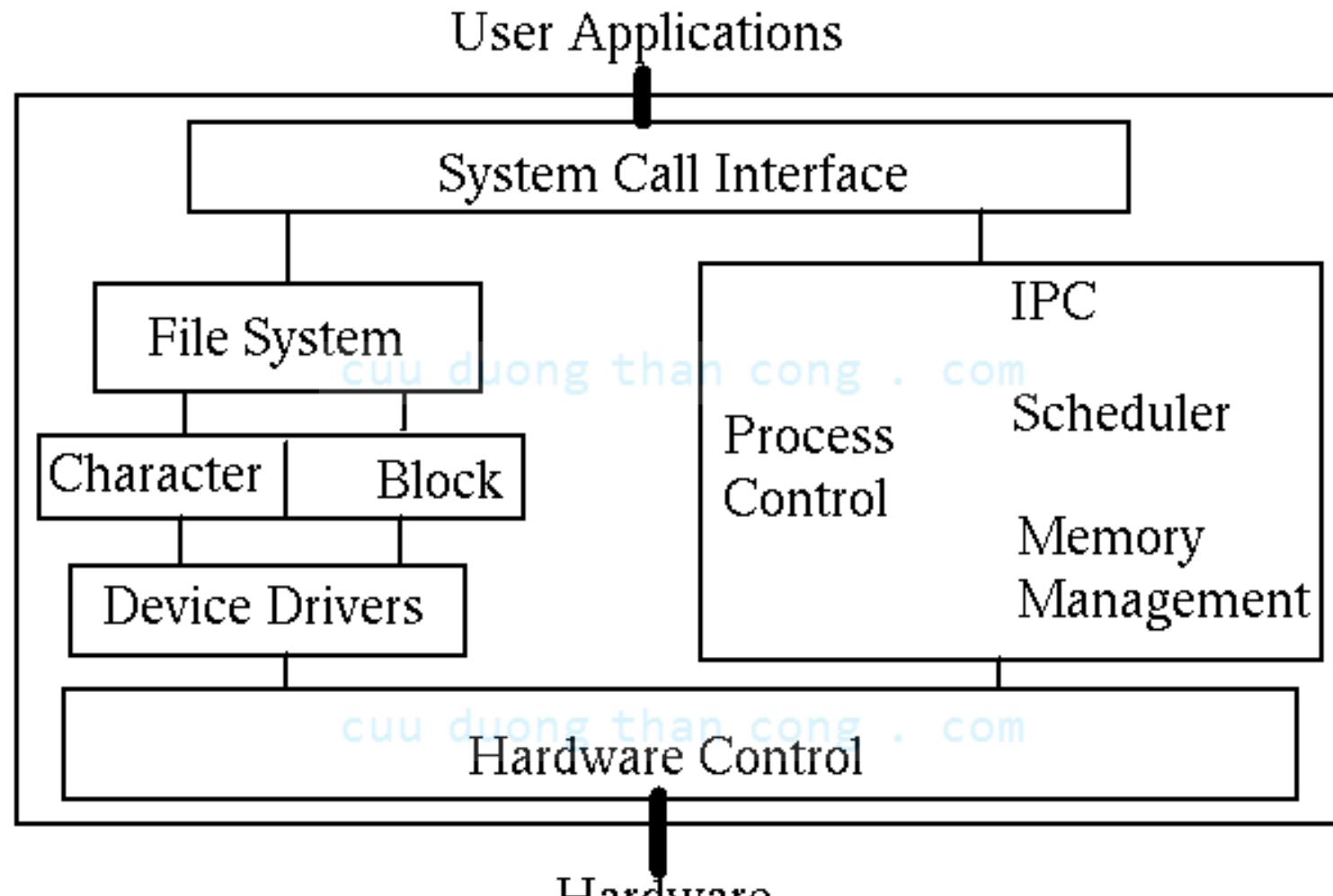
- Kiến trúc hệ điều hành nhúng Linux
- Kiến trúc nhân hệ điều hành
- Quá trình khởi động hệ điều hành nhúng Linux

cuu duong than cong . com

cuu duong than cong . com

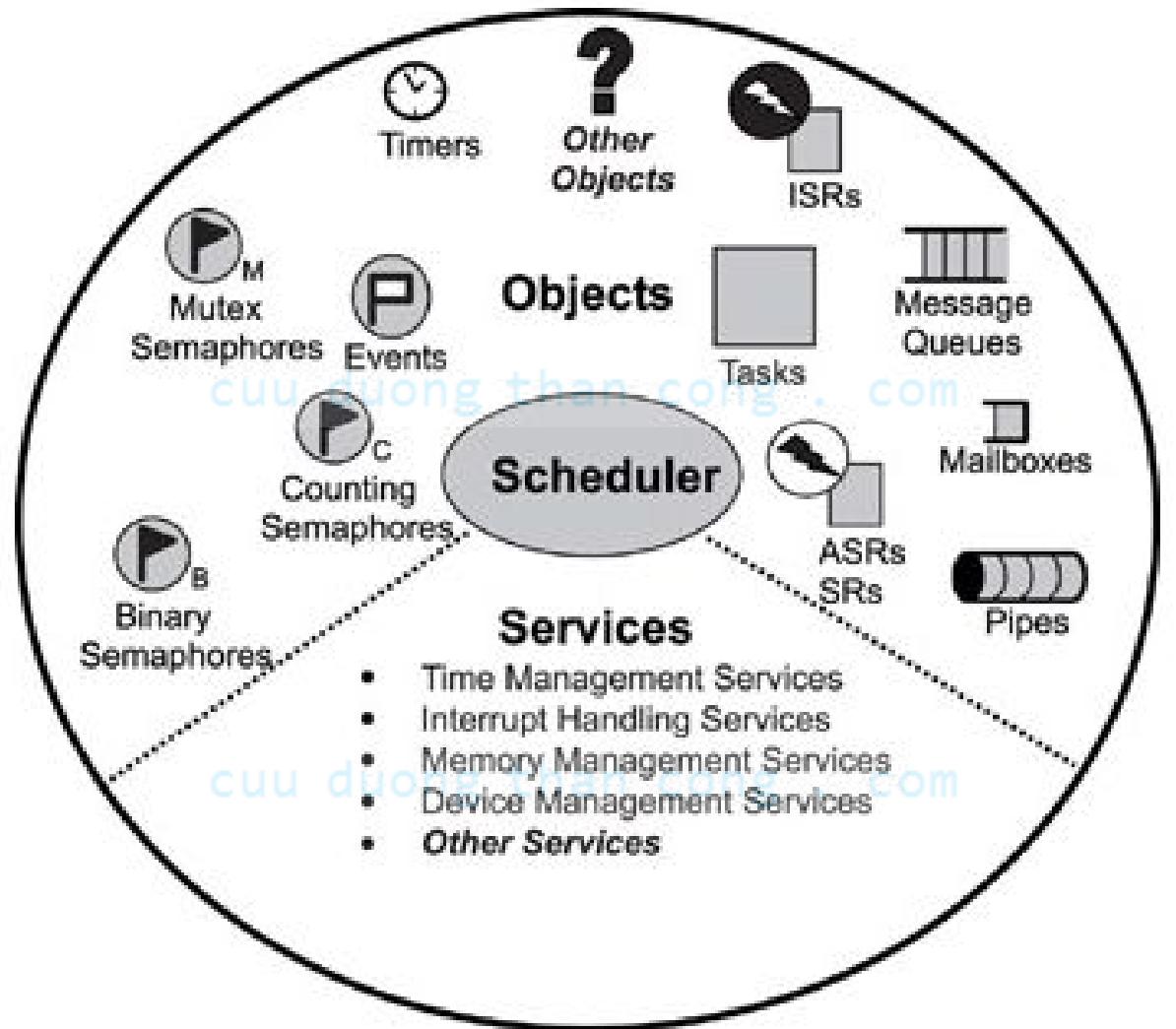


Kiến trúc hệ điều hành Linux





Cấu trúc nhân hệ điều hành

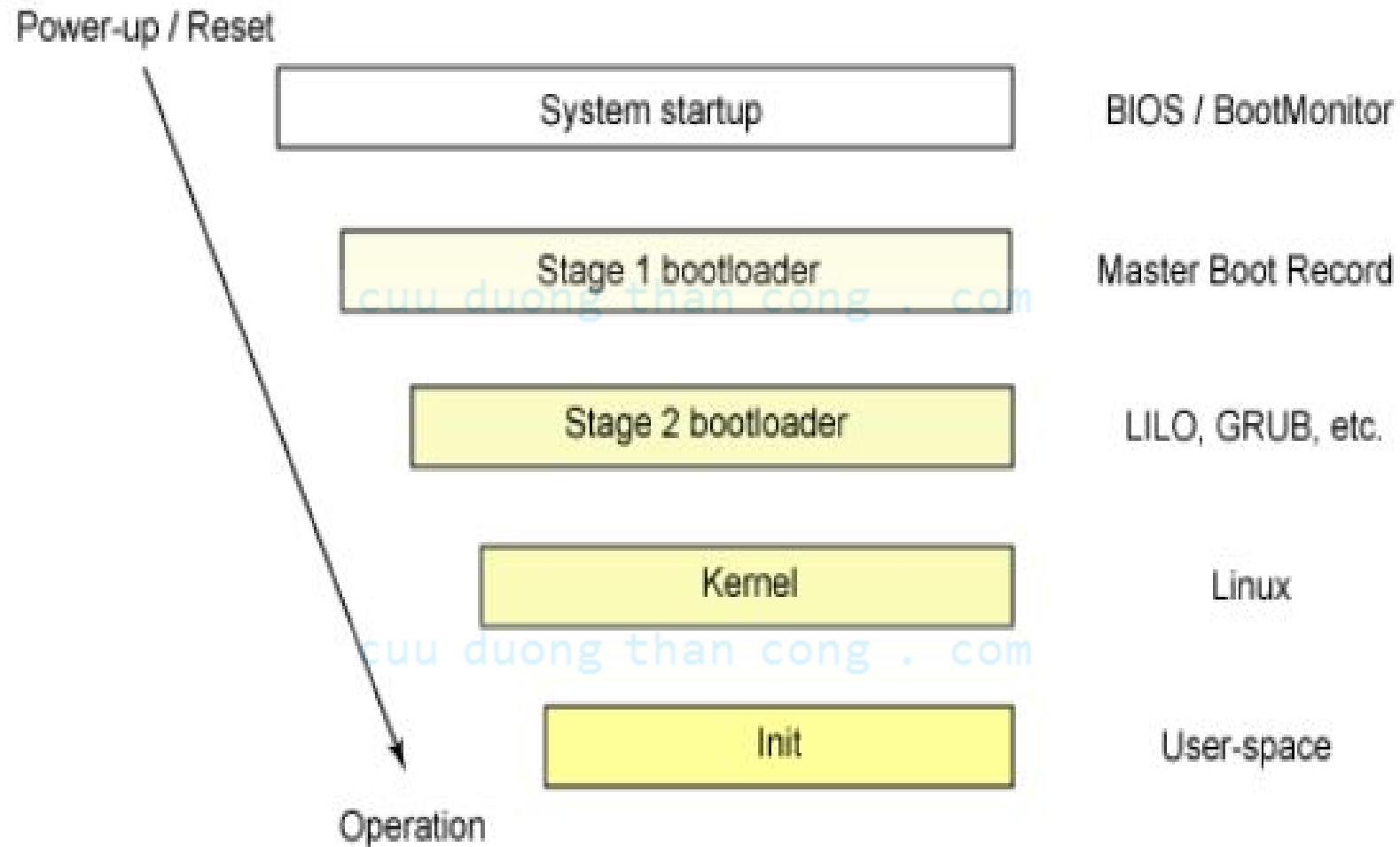




- Hỗ trợ rất nhiều kiến trúc (cả 32 bit và 64 bit)
 - X86, ARM, PowerPC, MIPS, SuperH, AVR32, ...
- Không hỗ trợ các vi điều khiển hiệu năng thấp
- Hỗ trợ cả kiến trúc có và không có khối quản lý bộ nhớ (MMU)
- Các hệ thống có thể dùng chung toolchains, bootloader và kernel, các thành phần khác phải riêng biệt và tương thích với từng hệ thống

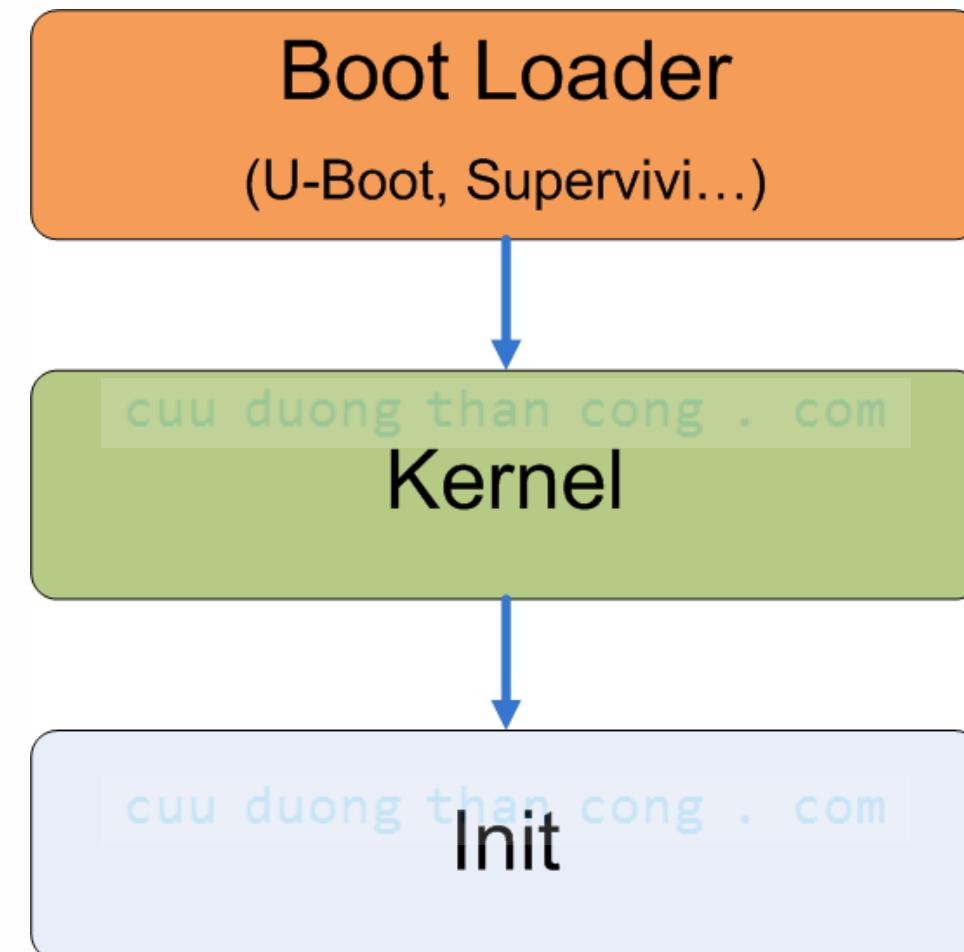


Quá trình boot hệ thống Linux trên PC





Quá trình boot hệ thống Linux nhúng





Quá trình boot hệ thống Linux nhúng

- Boot loader: chương trình mồi, thực hiện kiểm tra phần cứng hệ thống và nạp nhân (kernel) của hệ điều hành
- Kernel: nhân hệ điều hành, chứa các thành phần cơ bản nhất
- Root file system: hệ thống file, chứa các modules bổ sung và các phần mềm ứng dụng

cuu duong than cong . com



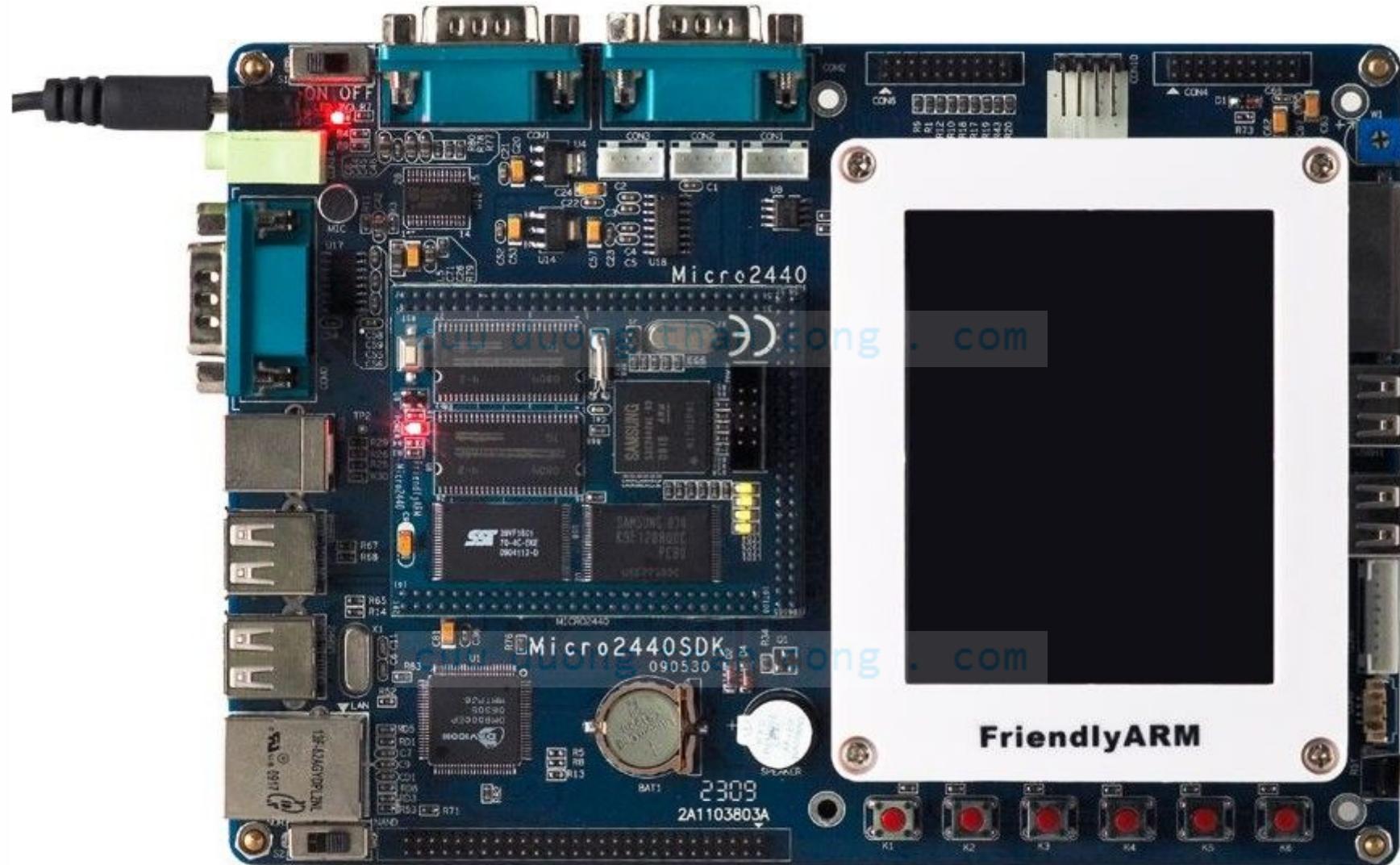
1.2. Cài đặt hệ điều hành nhúng Linux

- **Bước 1:** Cài đặt bootloader (VD: U-Boot, Supervivi)
- **Bước 2:** Cài đặt kernel
- **Bước 3:** Cài đặt hệ thống file (root file system)

cuu duong than cong . com



Giới thiệu KIT Micro2440

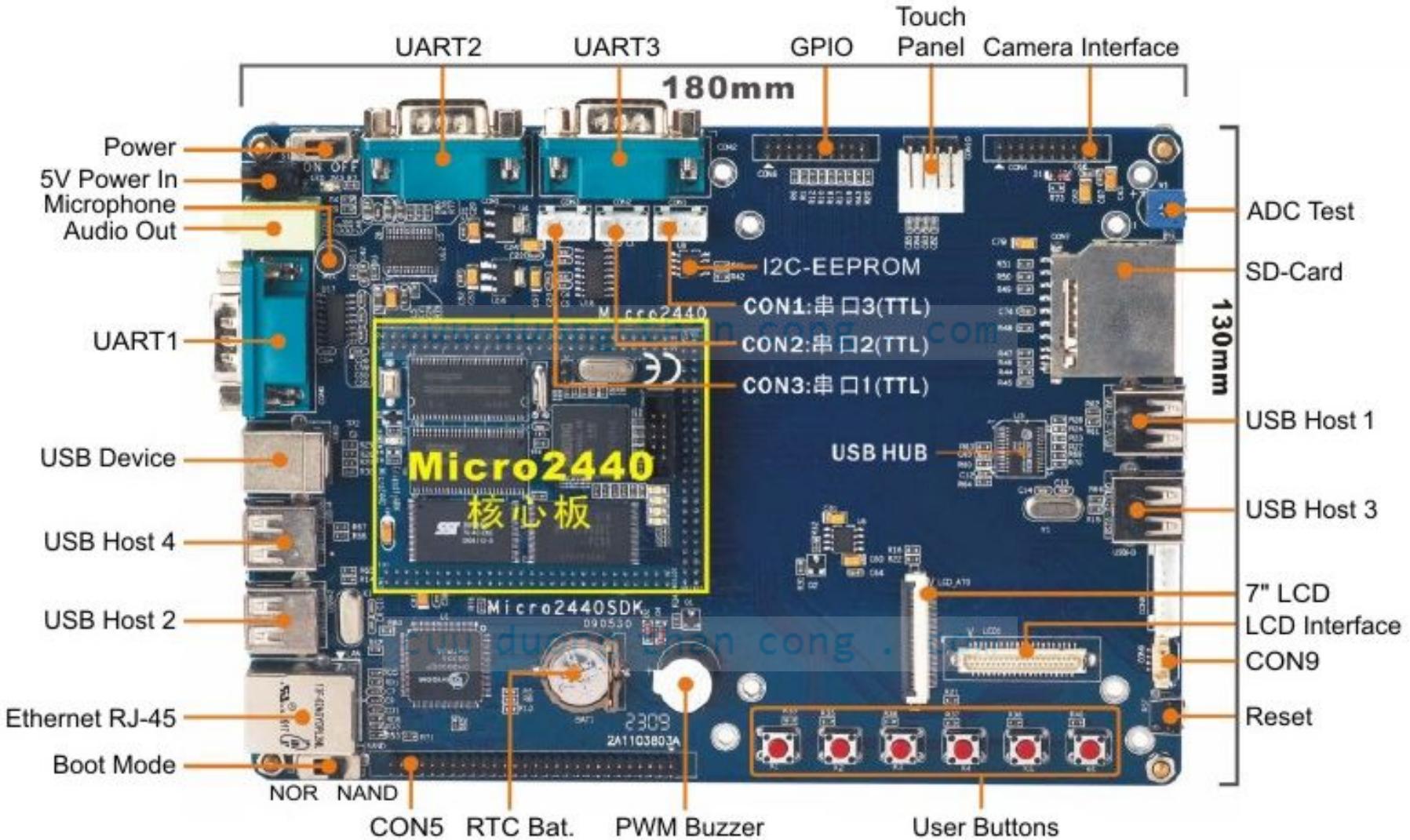


Lập trình hệ nhúng

22



Giới thiệu KIT Micro2440





Giới thiệu KIT Micro2440





Giới thiệu KIT Micro2440

■ Thông số kỹ thuật

Khối chức năng	Thông số kỹ thuật
CPU	Samsung S3C2440A, 400MHz, Max. 533Mhz
SDRAM	<ul style="list-style-type: none">- 64M SDRAM- 32bit DataBus- SDRAM Clock 100MHz
Flash	<ul style="list-style-type: none">- 64M or 128M Nand Flash,- 2M Nor Flash (đã được cài đặt sẵn BIOS)
Màn hình LCD	<ul style="list-style-type: none">- Màn hình cảm ứng- Tối đa 4096 màu STN, kích thước từ 3.5 inches tới



Giới thiệu KIT Micro2440

Các thiết bị ngoại vi	<ul style="list-style-type: none">- 1 khối 10/100M Ethernet RJ-45(DM9000)- 3 Serial Port- 1 USB Host- 1 USB Slave Type B- 1 giao tiếp SD Card- 1 Stereo Audio out, 1 Micro In- 1 20-Pin JTAG (kết nối với mạch nạp, debug)- 4 đèn LED đơn- 6 nút bấm- 1 còi điều khiển sử dụng PWM- 1 biến trở sử dụng kiểm tra bộ chuyển đổi số/tương tự (A/D converter)- 1 EEPROM giao tiếp theo chuẩn I²C- 1 giao tiếp với cảm biến ánh (20-chân)- 1 pin cho đồng hồ thời gian thực
Các hệ điều hành được hỗ trợ	<ul style="list-style-type: none">- Linux 2.6- Android- WinCE 5 and 6



Giới thiệu KIT nhúng micro2440

Các thiết bị ngoại vi	<ul style="list-style-type: none">- 1 khối 10/100M Ethernet RJ-45(DM9000)- 3 Serial Port- 1 USB Host- 1 USB Slave Type B- 1 giao tiếp SD Card- 1 Stereo Audio out, 1 Micro In- 1 20-Pin JTAG (kết nối với mạch nạp, debug)- 4 đèn LED đơn- 6 nút bấm- 1 còi điều khiển sử dụng PWM- 1 biến trở sử dụng kiểm tra bộ chuyển đổi số/tương tự (A/D converter)- 1 EEPROM giao tiếp theo chuẩn I²C- 1 giao tiếp với cảm biến ánh (20-chân)- 1 pin cho đồng hồ thời gian thực
Các hệ điều hành được hỗ trợ	<ul style="list-style-type: none">- Linux 2.6- Android- WinCE 5 and 6



Cài đặt trên môi trường Windows

■ Công cụ

- Phần mềm **HyperTerminal**: kết nối với KIT micro2440 qua cổng COM
- Phần mềm **DNW**: kết nối với KIT micro2440 qua cổng USB

■ Cách thức

- Phần mềm HyperTerminal truyền các lệnh điều khiển
- Phần mềm DNW trao đổi file



Cài đặt trên môi trường Linux

■ Công cụ:

- Phần mềm **minicom**: kết nối với KIT micro2440 qua cổng COM
- Phần mềm **usbpush**: kết nối với KIT micro2440 qua cổng USB

■ Cách thức

- Phần mềm minicom truyền các lệnh điều khiển
- Phần mềm usbpush trao đổi file



1.3. Tùy biến, biên dịch nhân Linux

- Khi nào cần biên dịch lại nhân?
 - Khi nâng cấp hệ thống lên các phiên bản mới hơn
 - Khi vá lỗi hệ thống
- Trình tự quá trình biên dịch nhân
 - Download nhân tại địa chỉ: kernel.org
 - Biên dịch nhân theo các bước:
 - ✓ Make menuconfig: chọn các thiết lập phù hợp cho thiết bị (**Có thể chọn các file config sẵn có của các nhà sản xuất và ghi đè vào file .config**)
 - ✓ Make zImage: tạo ảnh cho nhân. Ảnh này có thể nạp xuống KIT.

30

Lập trình hệ nhúng



Thảo luận





Chương 2. Lập trình vào ra căn bản

- 2.1. Cài đặt môi trường phát triển
- 2.2. Cơ bản về lập trình Linux
- 2.3. Cơ chế lập trình giao tiếp thiết bị
- 2.4. Lập trình điều khiển led
- 2.5. Lập trình ghép nối nút bấm





2.1. Cài đặt môi trường phát triển

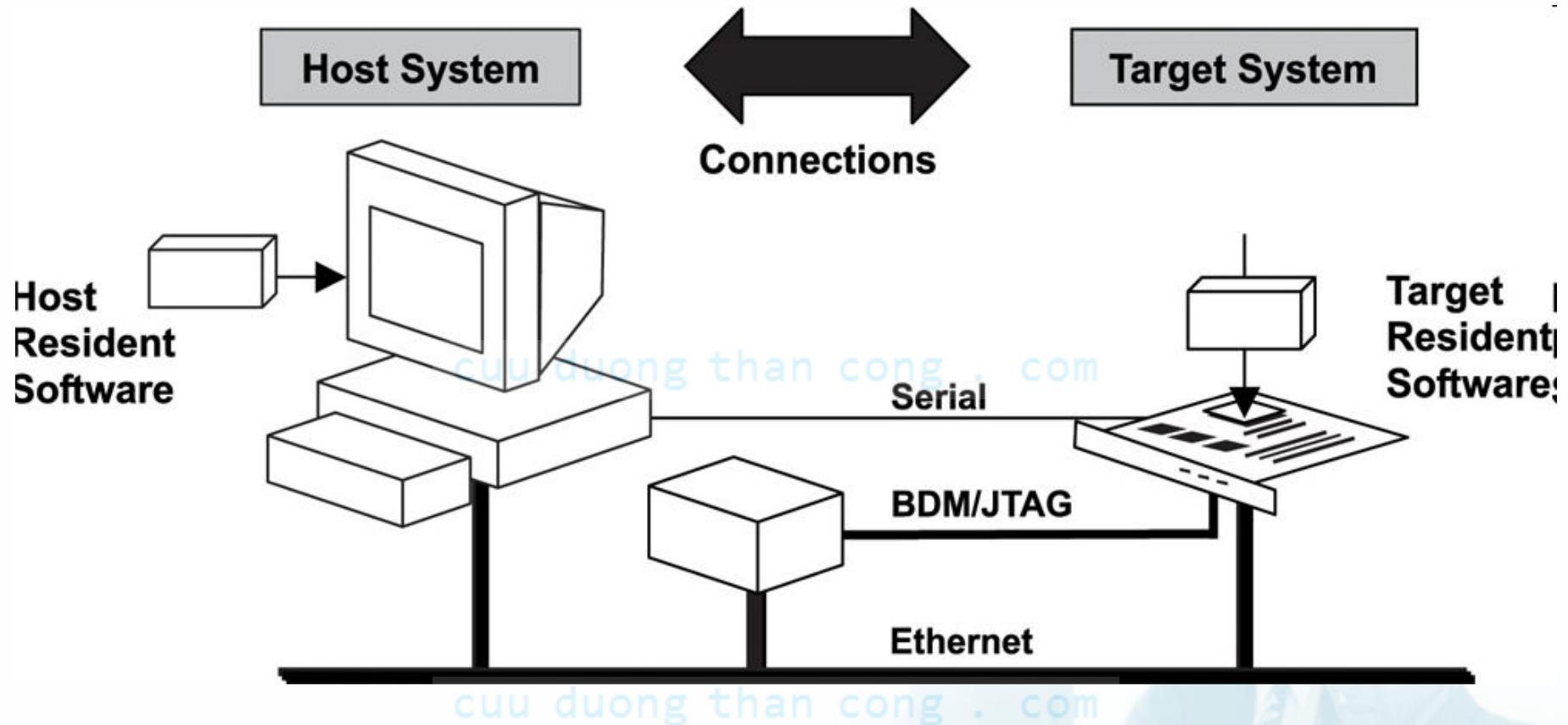
- Mô hình lập trình
- Môi trường phát triển ứng dụng
- Cài đặt môi trường

cuu duong than cong . com

cuu duong than cong . com



Mô hình lập trình hệ thống nhúng

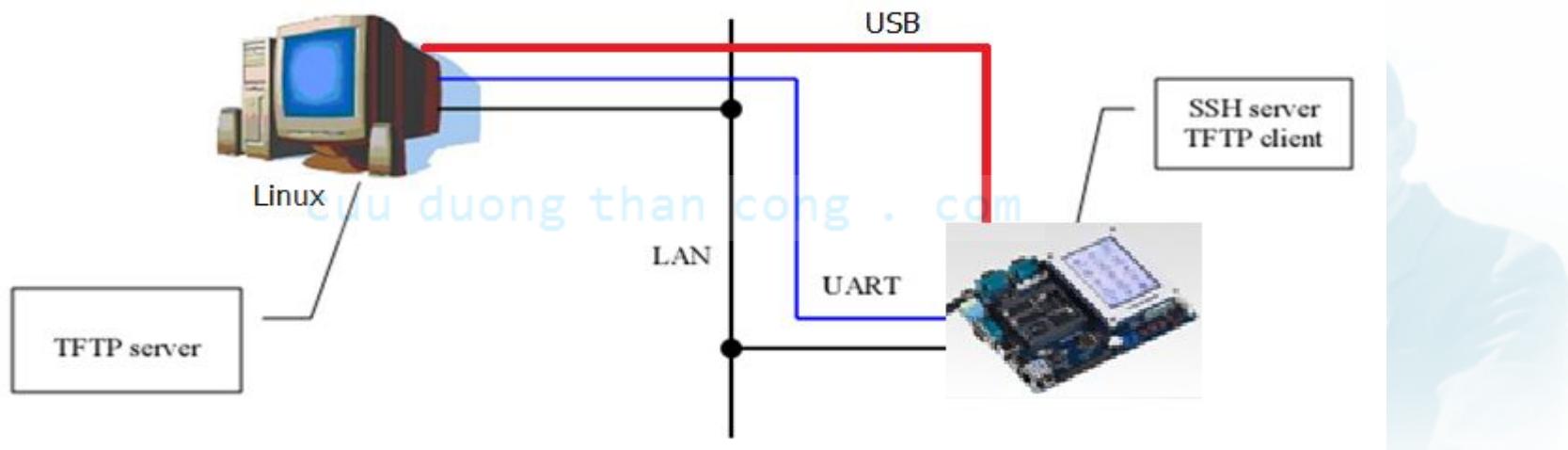


- **Host: hệ thống chứa môi trường phát triển**
- **Target: hệ nhúng cần phát triển ứng dụng**



▪ Phần mềm

- Hệ điều hành Linux
- Cross toolchains (gcc 4.4.3): biên dịch, GDB: công cụ debug
- gFTP: truyền nhận file Host<->KIT qua giao thức TFTP
- Telnet: kết nối KIT qua Ethernet (sử dụng cross cable)





3.2. Cài đặt môi trường phát triển

- Môi trường phát triển

- Hệ điều hành Linux (Ubuntu 9.04 hoặc mới hơn)
- Trình biên dịch chéo: ARM Linux GCC 4.4.3

- Phần mềm hỗ trợ

- gFTP

- Cấu hình mạng sử dụng

- Linux host: 192.168.1.30
- Linux target: 192.168.1.230



Cài đặt trình biên dịch chéo

- **Bước 1:** Giải nén arm-linux-gcc-4.4.3.tar.gz

tar -zxvf arm-linux-gcc-4.4.3.tar.gz

- **Bước 2:** Cập nhật biến môi trường PATH

- Thêm đường dẫn tới thư mục **bin** của arm-linux-gcc-4.4.3 (**Cập nhật biến môi trường PATH trong file .bashrc**)

- **Bước 3:** Kiểm tra trình biên dịch

- Mở cửa sổ console, gõ lệnh: ***arm-linux-gcc --version***

- Thông báo về phiên bản của arm-linux-gcc hiện ra
=> quá trình cài đặt thành công

Lập trình hệ nhúng

37



Kiểm tra trình biên dịch chéo

```
root@thuan-laptop: /home/thuan
File Edit View Terminal Help
thuan@thuan-laptop:~$ su
Password:
root@thuan-laptop:/home/thuan# arm-linux-gcc --version
.arm-none-linux-gnueabi-gcc (ctng-1.6.1) 4.4.3
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@thuan-laptop:/home/thuan# cuu duong than cong . com
```

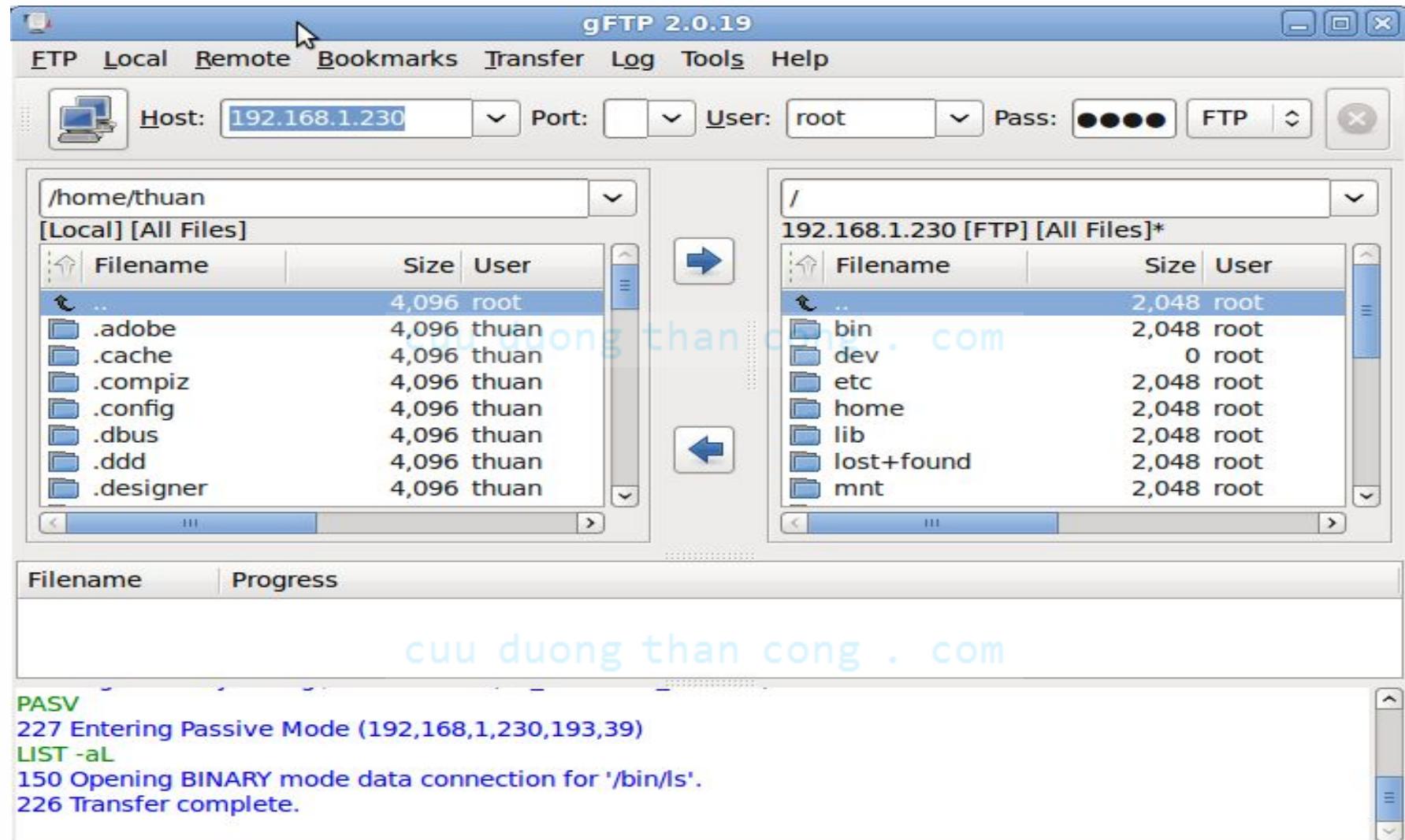


Cài đặt phần mềm gFTP

- **Bước 1:** Cài đặt phần mềm gFTP
 - Gõ lệnh: **apt-get install gftp**
- **Bước 2:** Kiểm tra kết nối giữa Host và Target
 - Mở phần mềm gFTP: **Applications->Internet->gFTP**
 - Thiết lập các tham số
 - ✓ Địa chỉ IP của KIT: 192.168.1.230
 - ✓ Username: root
 - ✓ Password: ktmt (**Có thể đổi bằng lệnh passwd**)
 - Mở kết nối



Kết nối sử dụng gFTP



40

Lập trình hệ nhúng



2.2. Cơ bản về lập trình Linux

- Cấu trúc chương trình đơn giản
- Cách thức biên dịch chương trình
- Nạp file thực thi xuống KIT và chạy ứng dụng

cuu duong than cong . com

cuu duong than cong . com



Cấu trúc chương trình

■ Tuân thủ cấu trúc chương trình ANSI C

Khai báo tệp tiêu đề

```
#include
```

Định nghĩa kiểu dữ liệu

```
typedef ...
```

Khai báo các hàm nguyên mẫu

Khai báo các biến toàn cục

Định nghĩa hàm main()

```
main()
```

```
{
```

```
...
```

```
}
```

Định nghĩa các hàm đã khai báo nguyên mẫu



Chương trình HelloWorld

```
int main(int argc, char** argv)
{
    int min,max,i;
    if(argc<3)
    {
        printf("\nUsage: Hello min max\n");
    }
    else    cuu duong than cong . com
    {
        min=atoi(argv[1]);
        max=atoi(argv[2]);
        for(i=min;i<max;i++)
        {
            printf("\nGia tri hiên tai=%d",i);
        } cuu duong than cong . com
    }
    return 0;
}
```

43



Cách thức biên dịch chương trình

- **Cách 1:** Sử dụng lệnh của cross compiler
 - VD: `arm-linux-gcc -g -o Hello Hello.c`
 - Kết quả: biên dịch ra một file thực thi có tên là Hello từ một file mã nguồn là Hello.c, file này có hỗ trợ khả năng debug
- **Cách 2:** Tạo và sử dụng Makefile
 - make là một tool cho phép quản lý quá trình biên dịch, liên kết ... của một dự án với nhiều file mã nguồn.
 - Tạo Makefile lưu các lệnh biên dịch theo định dạng của Makefile
 - Sử dụng lệnh `make` để chạy Makefile và biên dịch chương trình
- **Cách 3:** Sử dụng **automake** và **autoconf**
 - Tạo makefile tự động



Cấu trúc Makefile

- Makefile cấu thành từ các target, variables và comments
- Target có cấu trúc như sau:

target: dependencies
[tab] system command

- target: make target
- Dependencies: các thành phần phụ thuộc (file mã nguồn, các file object...)
- System command: các câu lệnh (lệnh biên dịch, lệnh linux)



VD 1: Makefile đơn giản

```
CC=arm-linux-gcc
```

```
all: Hello.c
```

```
        $(CC) -g -o Hello Hello.c
```

```
clear:
```

cuu duong than cong . com

```
        rm Hello
```

- **Biên dịch chương trình: make all**
- **Xóa file sinh ra trước đó: make clear**



VD 2: Makefile liên kết

```
CC=arm-linux-gcc
```

```
OUTPUT=Hello
```

```
all:Hello.o display.o
```

```
$(CC) -o $(OUTPUT) Hello.o display.o
```

```
Hello.o:Hello.c
```

```
$(CC) -c Hello.c
```

```
display.o:display.c
```

```
$(CC) -c display.c
```



Nạp file thực thi xuống KIT

- **Bước 1:** sử dụng phần mềm gFTP chuyển file Hello (đã được biên dịch trước đó) xuống KIT, ví dụ xuống thư mục: /ktmt/bin
- **Bước 2:** telnet xuống KIT, chuyển tới thư mục /dks/bin, thực thi chương trình
 - Gõ lệnh: ./Hello
 - Nếu chương trình chưa có quyền thực thi, thực hiện cấp quyền: chmod +x Hello
- **Bước 3:** quan sát kết quả



2.3. Cơ chế lập trình giao tiếp thiết bị

- Device files, Device number
- Kiểm tra danh sách device driver, thiết bị
- Cơ chế giao tiếp

cuu duong than cong . com

cuu duong than cong . com



Device files, Device number

■ Device files: **ls -l /dev**

- Device file không phải là file thông thường, không phải là một vùng dữ liệu trên hệ thống file
- Quá trình đọc ghi device file
 - ✓ Giao tiếp với device driver
 - ✓ Đọc, ghi phần cứng của thiết bị

■ Phân loại device files

- Character device: thiết bị phần cứng đọc, ghi một chuỗi các byte dữ liệu
- Block device: thiết bị phần cứng đọc, ghi một khối dữ liệu

50



Device files, Device number

- Device number: mỗi thiết bị được xác định bởi hai giá trị
 - Major device number: xác định thiết bị này sử dụng driver nào
 - Minor device number: phân biệt giữa các thiết bị khác nhau cùng sử dụng chung một device driver

cuu duong than cong . com



Kiểm tra danh sách thiết bị

- Kiểm tra danh sách các nhóm thiết bị
 - Gõ lệnh **cat /proc/devices**

```
Character devices:  
 1 mem  
 4 /dev/vc/0  
 4 tty  
 4 ttys  
 5 /dev/tty cuu duong than cong . com  
 5 /dev/console  
 5 /dev/ptmx  
 7 vcs  
 10 misc  
 13 input  
 29 fb  
 36 netlink  
 128 ptm  
 136 pts  
 180 usb cuu duong than cong . com  
  
Block devices:  
 1 ramdisk  
 3 ide0  
 9 md  
 22 ide1  
 253 device-mapper  
 254 mdp
```

52



Kiểm tra danh sách thiết bị

- Kiểm tra danh sách các thiết bị mount vào hệ thống

- ❖ Gõ lệnh `cat /proc/mounts`

- ❖ Gõ lệnh `mount`

```
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0 none
/dev ramfs rw 0 0
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
none /dev ramfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
```

53



Cơ chế lập trình giao tiếp thiết bị

- Cơ chế lập trình: giao tiếp qua các device files
 - Sử dụng các hàm vào ra file
 - ✓ open
 - ✓ close
 - ✓ read
 - ✓ write
 - Sử dụng hàm điều khiển vào ra: **ioctl**

cuu duong than cong . com

cuu duong than cong . com



2.4. Lập trình điều khiển led

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
int main(int argc, char **argv)
{
    int on;
    int led_no;
    int fd;
    sscanf(argv[1], "%d", &led_no);
    sscanf(argv[2], "%d", &on);
    fd = open("/dev/leds", 0);
    if (fd < 0) {
        perror("open device leds");
        exit(1);
    }
    ioctl(fd, on, led_no);
    close(fd);
    return 0;
}
```



Lập trình điều khiển led đơn

- **fd=open("/dev/leds",0)**

- fd: file id
- /dev/leds: device file
- 0: WRITE_ONLY

cuu duong than cong . com

- **ioctl(fd, on, led_no)**

- ioctl: IO control
- Điều khiển bật/tắt led đơn có số hiệu led_no

- Driver cho led đơn:

linux-2.6.32.2/drivers/char/mini2440_leds.c



2.5. Lập trình ghép nối nút bấm

```
int main(void)
{
    int buttons_fd;
    char buttons[6] = {'0', '0', '0', '0', '0', '0'};

    buttons_fd = open("/dev/buttons", 0);
    if (buttons_fd < 0) {
        perror("open device buttons");
        exit(1);
    }
    for (;;) {
        char current_buttons[6];
        int count_of_changed_key;
        int i;
        read(buttons_fd, current_buttons, sizeof current_buttons);
        for (i = 0, count_of_changed_key = 0;
             i < sizeof buttons / sizeof buttons[0]; i++) {
            if (buttons[i] != current_buttons[i]) {
                buttons[i] = current_buttons[i];
                count_of_changed_key++;
            }
        }
        close(buttons_fd);
        return 0;
    }
}
```

57



Lập trình ghép nối nút bấm

- **buttons_fd=open("/dev/buttons",0)**
 - buttons_fd: file id
 - /dev/buttons: device file
- **read(buttons_fd,current_buttons,sizeof(current_buttons))**
 - Đọc trạng thái các nút bấm
- **close(buttons_fd):** đóng file
- Driver cho nút nhấn
**linux-
2.6.32.2/drivers/char/mini2440_buttons.c**



Chương 3. Lập trình vào ra nâng cao

3.1. Lập trình giao tiếp cổng COM theo chuẩn RS232

3.2. Lập trình giao tiếp cổng USB

3.3. Lập trình ghép nối ADC

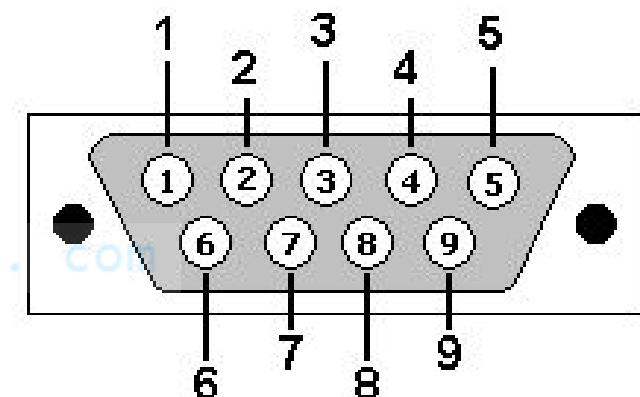
cuu duong than cong . com



3.1. Lập trình giao tiếp cổng RS232

▪ Chuẩn đầu nối trên PC

- Chân 1 (DCD-Data Carrier Detect): phát hiện tín hiệu mang dữ liệu
- Chân 2 (RxD-Receive Data): nhận dữ liệu
- Chân 3 (TxD-Transmit Data): truyền dữ liệu
- Chân 4 (DTR-Data Terminal Ready): đầu cuối dữ liệu sẵn sàng
- Chân 5 (Signal Ground): đất của tín hiệu
- Chân 6 (DSR-Data Set Ready): dữ liệu sẵn sàng
- Chân 7 (RTS-Request To Send): yêu cầu gửi
- Chân 8 (CTS-Clear To Send): Xóa để gửi
- Chân 9 (RI-Ring Indicate): báo chuông





Chuẩn RS232

▪ Khuôn dạng khung truyền

- PC truyền nhận dữ liệu qua cổng nối tiếp RS-232 thực hiện theo kiểu không đồng bộ (Asynchronous)
- Khung truyền gồm 4 thành phần
 - ✓ 1 Start bit (Mức logic 0): bắt đầu một gói tin, đồng bộ xung nhịp clock giữa DTE và DCE
 - ✓ Data (5,6,7,8 bit): dữ liệu cần truyền
 - ✓ 1 parity bit (chẵn (even), lẻ (odd), mark, space): bit cho phép kiểm tra lỗi
 - ✓ Stop bit (1 hoặc 2 bit): kết thúc một gói tin





Lập trình giao tiếp RS232

- **Khởi tạo:** Khai báo thư viện
- **Bước 1:** Mở cổng
- **Bước 2:** Thiết lập tham số
- **Bước 3:** Đọc, ghi cổng
- **Bước 4:** Đóng cổng





Khai báo thư viện

- #include <stdio.h>
- #include <stdlib.h>
- #include <string.h>
- #include <unistd.h> // UNIX standard function
- #include <fcntl.h> // File control definitions
- #include <errno.h> // Error number definitions
- #include <termios.h> // POSIX terminal control
- #include <time.h> // time calls



Bước 1: Mở cổng

- Sử dụng lệnh mở file

int fd = open ("/dev/ttySAC0", O_RDWR);

- Fd >0 nếu mở file thành công
- Fd<0 nếu mở file thất bại

cuu duong thanh cong . com



Bước 2: Thiết lập tham số

- Sử dụng cấu trúc termios

```
struct termios port_settings;
```

- Thiết lập tham số (9600, 8, n, 1)

```
cfsetispeed(&port_settings, B9600);
```

```
cfsetospeed(&port_settings, B9600);
```

```
port_settings.c_cflag &= ~PARENB;
```

```
port_settings.c_cflag &= ~CSTOPB;
```

```
port_settings.c_cflag &= ~CSIZE;
```

```
port_settings.c_cflag |= CS8;
```

```
tcsetattr(fd, TCSANOW, &port_settings);
```



Bước 3: Đọc, ghi cổng

- Đọc cổng: sử dụng lệnh đọc file

`n=read(fd,&result,sizeof(result));`

- N: số ký tự đọc được
- Result: chứa kết quả
- Ghi cổng: sử dụng lệnh ghi file

`n=write(fd,"Hello World\r",12);`

- N:số ký tự đã ghi
- Fd: file id (có được từ thao tác mở file thành công)



Bước 4: Đóng cổng

- Đóng cổng: sử dụng lệnh đóng file
close (fd);

Fd: file ID (có được từ thao tác mở file thành công)

cuu duong than cong . com

cuu duong than cong . com



3.2. Lập trình giao tiếp USB

cuu duong than cong . com

cuu duong than cong . com



3.3. Lập trình giao tiếp ADC

cuu duong than cong . com

cuu duong than cong . com



Chương 4. Các kỹ thuật lập trình nâng cao

- 4.1. Tiến trình (process) và cơ chế sử dụng signal
- 4.2. Lập trình xử lý đa tiến trình
- 4.3. Giới thiệu về luồng
- 4.4. Lập trình đa luồng





4.1. Tiến trình và cơ chế sử dụng signal

- Khái niệm tiến trình
- Cơ chế sử dụng signal

cuu duong than cong . com

cuu duong than cong . com



Khái niệm tiến trình

- Tiến trình được tạo ra khi ta thực thi một chương trình
- Đa tiến trình cho phép nhiều chương trình cùng thực thi và chia sẻ dữ liệu với nhau
- Các tham số của một tiến trình
 - PID (Process ID): số hiệu tiến trình
 - PPID (Parent Process ID): số hiệu tiến trình cha
 - Command: câu lệnh được gọi để thực thi tiến trình

ls -e -o pid,ppid,command



- **Lấy về PID: sử dụng hàm getpid()**
- **Lấy về PPID: sử dụng hàm getppid()**
- **Hàm getpid() và getppid() trả giá trị kiểu pid_t (bản chất là kiểu int)**

```
#include <stdio.h>  
#include <unistd.h>  
  
int main ()  
{  
    printf ("The process ID is %d\n", (int) getpid ());  
    printf ("The parent process ID is %d\n", (int) getppid ());  
    return 0;  
}
```



Dừng tiến trình

- Cách 1: Sử dụng tổ hợp phím Ctrl + C
- Cách 2: Sử dụng shell command

kill PID

cuu duong than cong . com

cuu duong than cong . com



Tạo tiến trình mới

▪ Cách 1: sử dụng hàm system

```
#include <stdlib.h>

int main ()
{
    int return_value;
    return_value = system ("ls -l /");
    return return_value;
}
```

cuu duong than cong . com



Tạo tiến trình mới

▪ Cách 2: sử dụng hàm fork và exec

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
    pid_t child_pid; cuu duong than cong . com

    printf ("the main program process ID is %d\n", (int) getpid ());

    child_pid = fork ();
    if (child_pid != 0) {
        printf ("this is the parent process, with id %d\n", (int) getpid ());
        printf ("the child's process ID is %d\n", (int) child_pid);
    }
    return 0;
}
```



Cơ chế sử dụng signal

- Signal là cơ chế cho phép giao tiếp giữa các tiến trình
- Signal là cơ chế không đồng bộ
- Khi tiến trình nhận được signal, tiến trình phải xử lý signal ngay lập tức
- Linux hỗ trợ 32 SIGNAL

cuu duong than cong . com



Danh sách signal thường dùng

Kiểu SIGNAL	Lý do gửi SIGNAL
SIGHUP	Báo cho chương trình khi thoát khỏi terminal
SIGINT	Khi người dùng nhấn Ctrl + C để tắt chương trình
SIGILL	Khi chương trình chạy lệnh không hợp lệ
SIGABRT	Khi chương trình nhận được lệnh abort
SIGKILL	Khi chương trình nhận được lệnh kill (đóng chương trình)
SIGUSR1	Tùy biến theo ứng dụng.
SIGUSR2	Tùy biến theo chương trình



Gửi SIGNAL tới process

- **Cách 1:** sử dụng shell command

kill [-SIGNAL_TYPE] PID

- **Cách 2:** sử dụng hàm kill trong chương trình, cho phép process này gửi signal tới process khác

kill(PID, SIGNAL_TYPE)

cuu duong than cong . com



4.2. Lập trình giao tiếp đa tiến trình

▪ Cơ chế:

- Tiến trình chính tạo ra các tiến trình con sử dụng lệnh fork và exec
- Sử dụng cơ chế signal để trao đổi tín hiệu giữa các tiến trình

cuu duong than cong . com



4.3. Giới thiệu về luồng (thread)

- Một chương trình mặc định chạy một luồng -> luồng chính
- Luồng chính có thể tạo ra các luồng khác, các luồng sẽ chạy đồng thời -> tăng tốc chương trình
- Các luồng chia sẻ không gian nhớ, truy xuất file và các tài nguyên khác
- Tham số của một luồng:
 - **thread ID:** số hiệu luồng (kiểu dữ liệu pthread_t)



4.4. Lập trình xử lý đa luồng

- Tạo luồng
- Truyền tham số cho luồng
- Nhận giá trị trả về từ luồng
- Tắt luồng

cuu duong than cong . com

cuu duong than cong . com



- Khai báo thư viện: pthread.h
- Hàm tạo luồng: pthread_create

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,  
                  void *(*start_routine) (void *), void *arg);
```

- ❖ **thread:** thread id
- ❖ **attr:** các thuộc tính của luồng, mặc định để NULL
- ❖ **start_routine:** hàm thực thi trong luồng
- ❖ **arg:** các tham số truyền cho luồng

- **Biên dịch chương trình:**
gcc -o multithread multithread.c -pthread



Mã nguồn tạo luồng

```
#include <pthread.h>
#include <stdio.h>
//thread function
void* print_xs (void* unused)
{
    while (1)
        fputc ('x', stdout);
    return NULL;
}
int main ()
{
    pthread_t thread_id;
    //Tao thread moi
    pthread_create (&thread_id, NULL, &print_xs, NULL);
    //Thuc hien main thread
    while (1)
        fputc ('o', stdout);
    return 0;
}
```



Truyền tham số cho luồng

- Khai báo cấu trúc dữ liệu chứa dữ liệu cần truyền cho luồng.
Ví dụ:

```
struct arg
{
    //Ký tự cần in
    char character;
    //Số lần cần in
    int count;
};
```

- Truyền dữ liệu cho luồng khi tạo luồng qua tham số **arg**
- Chương trình con thực thi luồng nhận tham số về và xử lý



Mã nguồn truyền tham số cho luồng

```
#include <pthread.h>
#include <stdio.h>
//thread function
void* print_xs (void* param)
{
    int i;
    struct arg* p=(struct arg*)param;
    for(i=0;i<p->count;i++)
        fputc (p->character, stdout);
    return NULL;
}
int main ()
{
    pthread_t thread_id1, thread_id2;
    struct arg arg1,arg2;
    //Tao thread moi
    arg1.character='a';
    arg1.count=10000;
    arg2.character='b';
    arg2.count=10000;
    pthread_create (&thread_id1, NULL, &print_xs, &arg1);
    pthread_create (&thread_id2, NULL, &print_xs, &arg2);
    //Thuc hien main thread
    while (1);
    return 0;
}
```



Tắt luồng

- Sử dụng hàm pthread_cancel:

```
int pthread_cancel(pthread_t thread);
```

- **thread: nhận tham số thread id của luồng muốn tắt**

cuu duong than cong . com



Mã nguồn tắt luồng

```
int count=0;
//thread function
void* print_xs (void* param)
{
    int i;
    struct arg* p=(struct arg*)param;
    while(1)
        fputc (p->character, stdout);
    return NULL;
}
int main ()
{
    pthread_t thread_id1, thread_id2;
    struct arg arg1,arg2;
    int thread_number,i;
    //Tao thread moi
    arg1.character='a';
    arg1.count=1000;
    arg2.character='b';
    arg2.count=200000;
    pthread_create (&thread_id1, NULL, &print_xs, &arg1);
    pthread_create (&thread_id2, NULL, &print_xs, &arg2);
    for(i=0;i<1000000;i++);
    pthread_cancel(thread_id1);
    printf("\nThread thu 1 da bi huy");
    pthread_cancel(thread_id2);
    printf("\nThread thu 2 da bi huy");
    return 0;
}
```



Chương 5. Lập trình Device Driver trên Linux

5.1. Kernel module

5.2. Device driver

cuu duong than cong . com

cuu duong than cong . com



5.1. Kernel Module

- Hoạt động trên Kernel Space, có thể truy xuất tới các tài nguyên của hệ thống
- Kernel Module cho phép thêm mới các module một cách linh hoạt, tránh việc phải biên dịch lại nhân hệ điều hành
- Kernel Module là cơ chế hữu hiệu để phát triển các device driver
- Xem danh sách các module đang chạy: **psmod**



- Các bước để thêm một kernel module vào hệ thống
 - Viết mã nguồn: chỉ sử dụng các thư viện được cung cấp bởi kernel, không sử dụng được các thư viện bên ngoài
 - Biên dịch mã nguồn module
 - Cài đặt module: dùng lệnh **insmod Tên_Module.ko**
 - Gỡ module: dùng lệnh **rmmmod Tên_Module**
 - Xem các thông tin log: sử dụng System Log Viewer



Mã nguồn kernel Module

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
static int hello_init(void)
{
    printk(KERN_ALERT "Khoi tao thanh cong\n");
    return 0;
}
static void hello_exit(void)
{
    printk(KERN_ALERT "Ket thuc thanh cong\n");
}
module_init(hello_init);
module_exit(hello_exit);
```



Kernel Module Makefile

```
obj-m += hello.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
```

cuu duong than cong . com



5.2. Device Driver

- Thêm các device driver theo cơ chế sử dụng Kernel Module
- Các thao tác thêm driver vào hệ thống
 - Viết mã nguồn (cấu trúc tương tự kernel Module).
Đăng ký Major ID
 - Biên dịch mã nguồn
 - Cài đặt sử dụng lệnh **insmod**
 - Sử dụng lệnh **mknod** để tạo device file trong /dev
mknod [options] NAME Type [Major Minor]



Chương 6. Lập trình giao diện QT

6.1. Giới thiệu nền tảng QT

6.2. Cài đặt và cấu hình nền tảng QT

6.3. Lập trình QT

cuu duong than cong . com

cuu duong than cong . com



6.1. Giới thiệu nền tảng QT

- Ứng dụng đa nền: Desktop, mobile, embedded computer
- Viết code 1 lần duy nhất, chạy trên nhiều nền tảng khác nhau
- Sử dụng ngôn ngữ C/C++
- Hỗ trợ các nền tảng: Windows, Linux, Embedded Linux, Win CE, Symbian, Maemo...
- Có thể tích hợp với các IDE thông dụng: Visual Studio, Eclipse
- Tham khảo: qt.nokia.com; qtcentre.org



Ứng dụng QT

From embedded devices to desktop applications



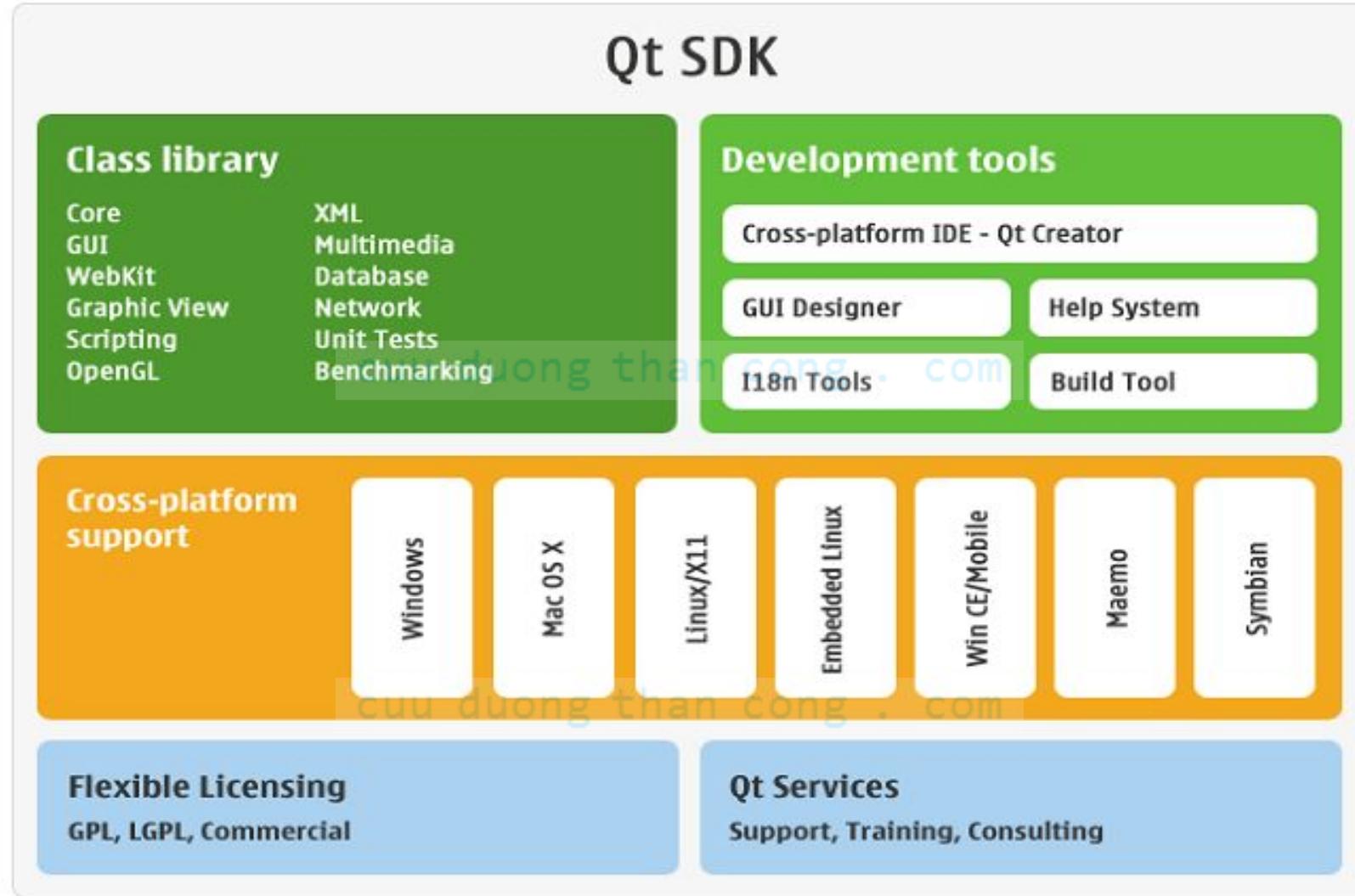
By companies from many industries





Kiến trúc nền tảng QT

Qt SDK





6.2. Quy trình cài đặt QT-Embedded

▪ **Bước 1:** Cài đặt QT Embedded (QT Everywhere)

- Cài đặt thư viện tslib để hỗ trợ màn hình touchscreen
 - ✓ Chuẩn bị các thư viện cần thiết:
 - ✓ Download mã nguồn của tslib
 - ✓ Cấu hình biên dịch
 - ✓ Dịch và cài đặt tslib
- Cài đặt QT-Embedded FrameWork
 - ✓ Chỉnh sửa file cấu hình
 - ✓ Cấu hình biên dịch
 - ✓ Dịch và cài đặt QT-Embedded



Cài đặt thư viện tslib

Cài một số công cụ để hỗ trợ việc dịch tslib:

```
$ sudo apt-get install autoconf
```

```
$ sudo apt-get install libtool
```

cuu duong than cong . com

Download tslib:

```
$ mkdir tslib-arm
```

```
$ cd tslib-arm
```

```
$ apt-get source tslib
```



Cài đặt thư viện tslib

Cấu hình tslib:

```
$ cd tslib-1.0  
$ ./autogen.sh  
$ ./configure --prefix=/opt/tslib/ --host=arm-none-linux-gnueabi
```

cuu duong than cong . com

Sửa file config.h

```
#define malloc rpl_malloc --> /* #define malloc rpl_malloc */
```

Dịch thư viện tslib:

```
$ make
```

```
$ sudo make install
```



Cài đặt QT-Embedded

Sửa file `../qt-everywhere- opensource-src-4.7.2/mkspecs/qws/linux-arm-g++/qmake.conf` như sau :

```
include(../../common/g++.conf)
include(../../common/linux.conf)
include(../../common/qws.conf)
# modifications to g++.conf
QMAKE_CC = arm-none-linux-gnueabi-gcc
QMAKE_CXX = arm-none-linux-gnueabi-g++
QMAKE_LINK = arm-none-linux-gnueabi-g++
QMAKE_LINK_SHLIB = arm-none-linux-gnueabi-g++
# modifications to linux.conf
QMAKE_AR = arm-none-linux-gnueabi-ar cqs
QMAKE_OBJCOPY = arm-none-linux-gnueabi-objcopy
QMAKE_STRIP = arm-none-linux-gnueabi-strip
QMAKE_INCDIR += /opt/tslib/include
QMAKE_LIBDIR += /opt/tslib/lib
load(qt_config)
```



Cài đặt QT-Embedded

Sau đó dịch qmake

```
$ cd  
$ cd qt-everywhere-opensource-src-4.7.2  
$ ./configure --prefix=/opt/qtembedded arm -xplatform qws/linux-arm-  
g++ -qt-mouse-tslib -no-qt3support -fast -no-largefile -qt-sql-sqlite -nomake tools -  
nomake demos -nomake examples  
$ make  
$ sudo make install
```

cuuduongthancong.com



Quy trình cài đặt QT-Embedded

- **Bước 2:** Copy các file thư viện cần thiết của QT-Embedded (vừa dịch thành công) xuống KIT
 - 3 thư viện quan trọng (VD: copy xuống thư mục /opt/qte/lib)
 - ✓ libQtCore.so.4
 - ✓ libQtGui.so.4
 - ✓ libQtNetwork.so.4
 - Copy các fonts vào thư mục **/opt/qte/lib/fonts**
 - Copy toàn bộ thư mục **/opt/tslib** trên HOST xuống thư mục /opt/tslib trên KIT
- **Bước 3:** Chỉnh file cấu hình **/etc/init.d/rcS**, tắt Qtopia để tránh tranh chấp
- **Bước 4:** Chỉnh sửa file cấu hình **/opt/tslib/etc/ts.conf** (trên KIT): bỏ chú thích dòng lệnh: **module_raw input**



Quy trình cài đặt QT-Embedded

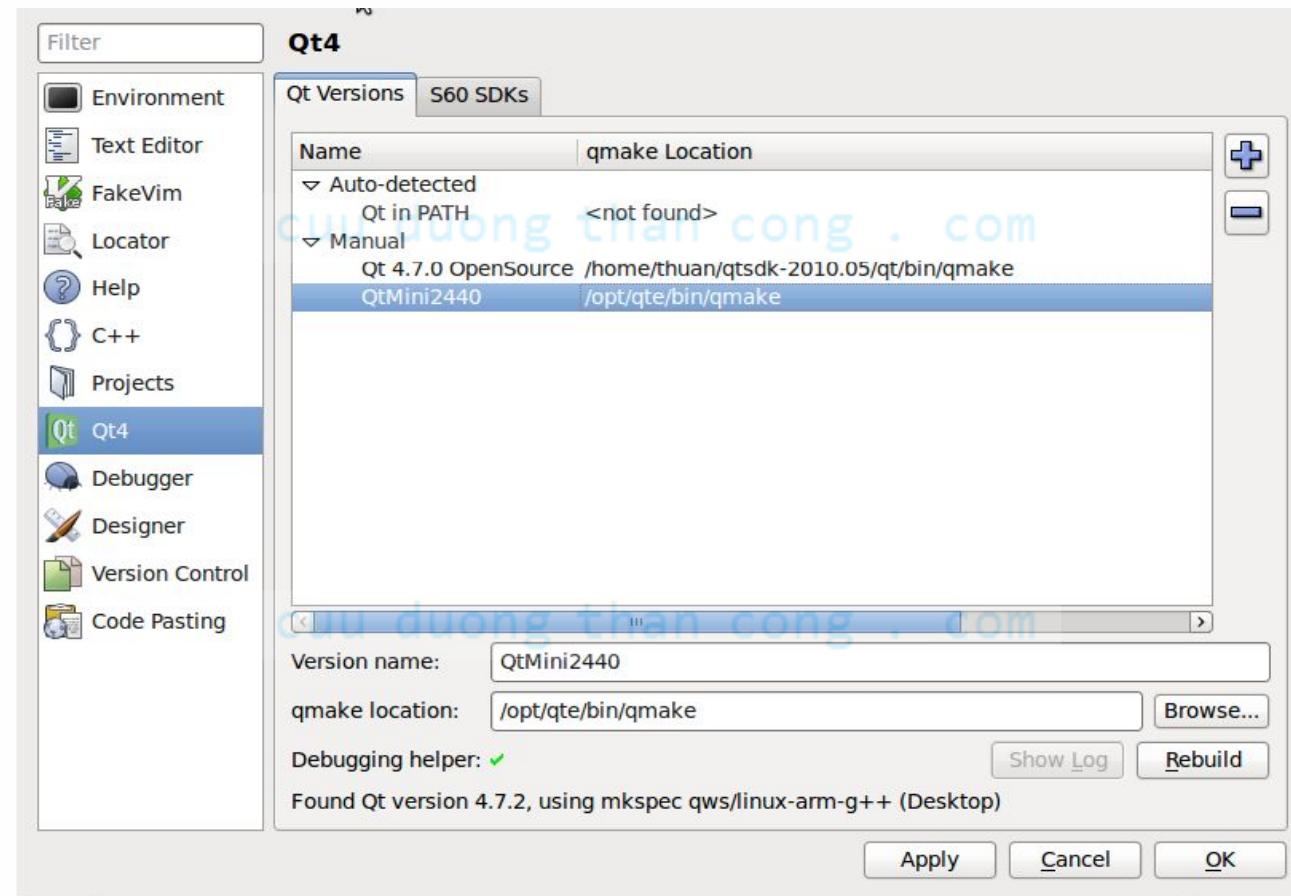
- **Bước 5:** thêm biến môi trường để sử dụng thư viện tslib: sửa file /etc/profile trên KIT

```
export TSLIB_TSEVENTTYPE=INPUT
export TSLIB_ROOT=/opt/tslib
export TSLIB_TSDEVICE=/dev/input/event0
export LD_LIBRARY_PATH=$TSLIB_ROOT/lib:$LD_LIBRARY_PATH
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CONSOLEDEVICE=none
export TSLIB_CONFFILE=$TSLIB_ROOT/etc/ts.conf
export POINTERCAL_FILE=/etc/pointercal
export TSLIB_CALIBFILE=/etc/pointercal
export QWS_MOUSE_PROTO=TSLIB:/dev/input/event0
```



Cấu hình trình dịch Qmake cho kit Mini2440

- **Bước 6:** Tạo cấu hình biên dịch cho Mini2440, trả tới Qmake đã biên dịch được ở trên

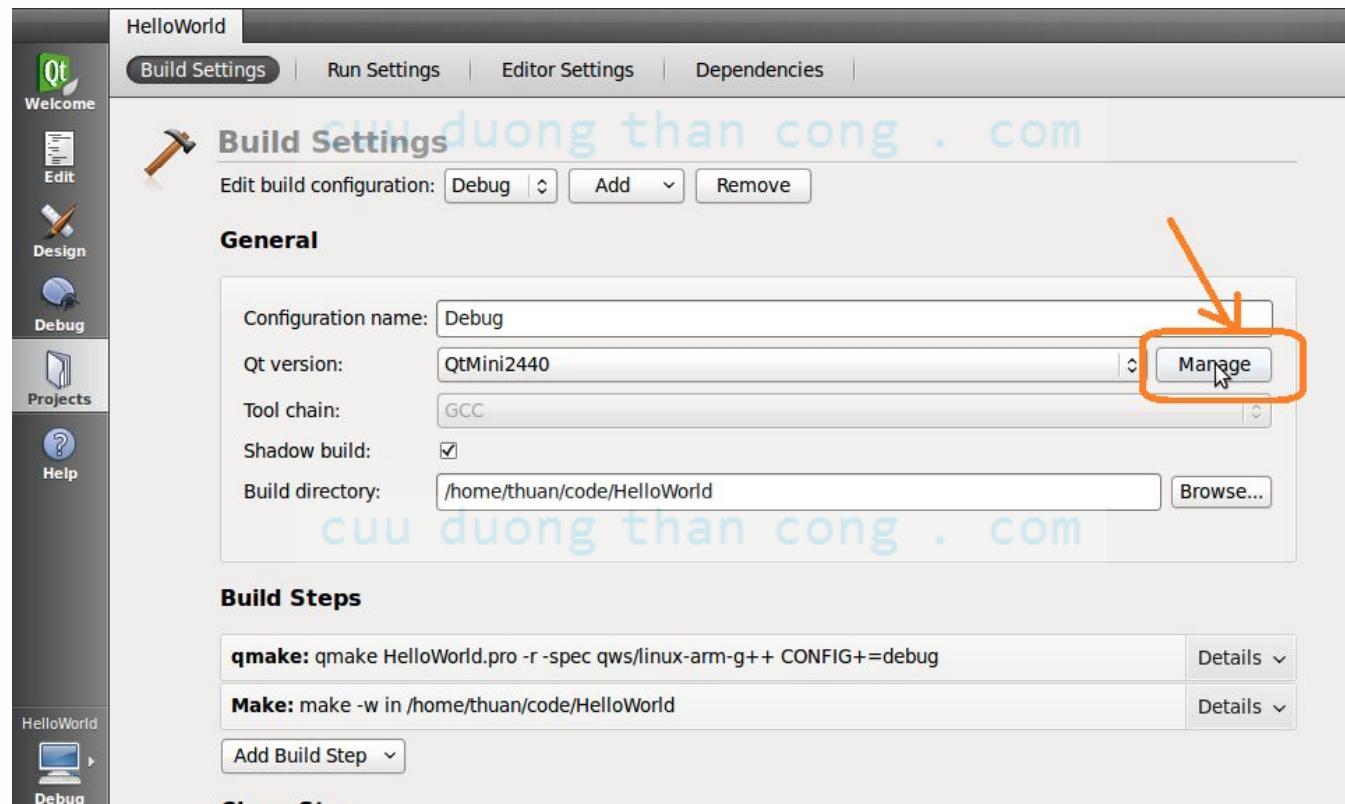




Cấu hình trình dịch Qmake cho kit Mini2440

■ Bước 7: Dịch chương trình QT cho KIT

- Chọn đúng bộ biên dịch Qmake cho QT Embedded





Ví dụ



108

Lập trình hệ nhúng



6.3. Lập trình QT

- Môi trường phát triển
- Cơ chế signal và slot
- Chương trình HelloWorld
- Cấu hình trình dịch Qmake cho KIT micro2440

cuu duong than cong . com



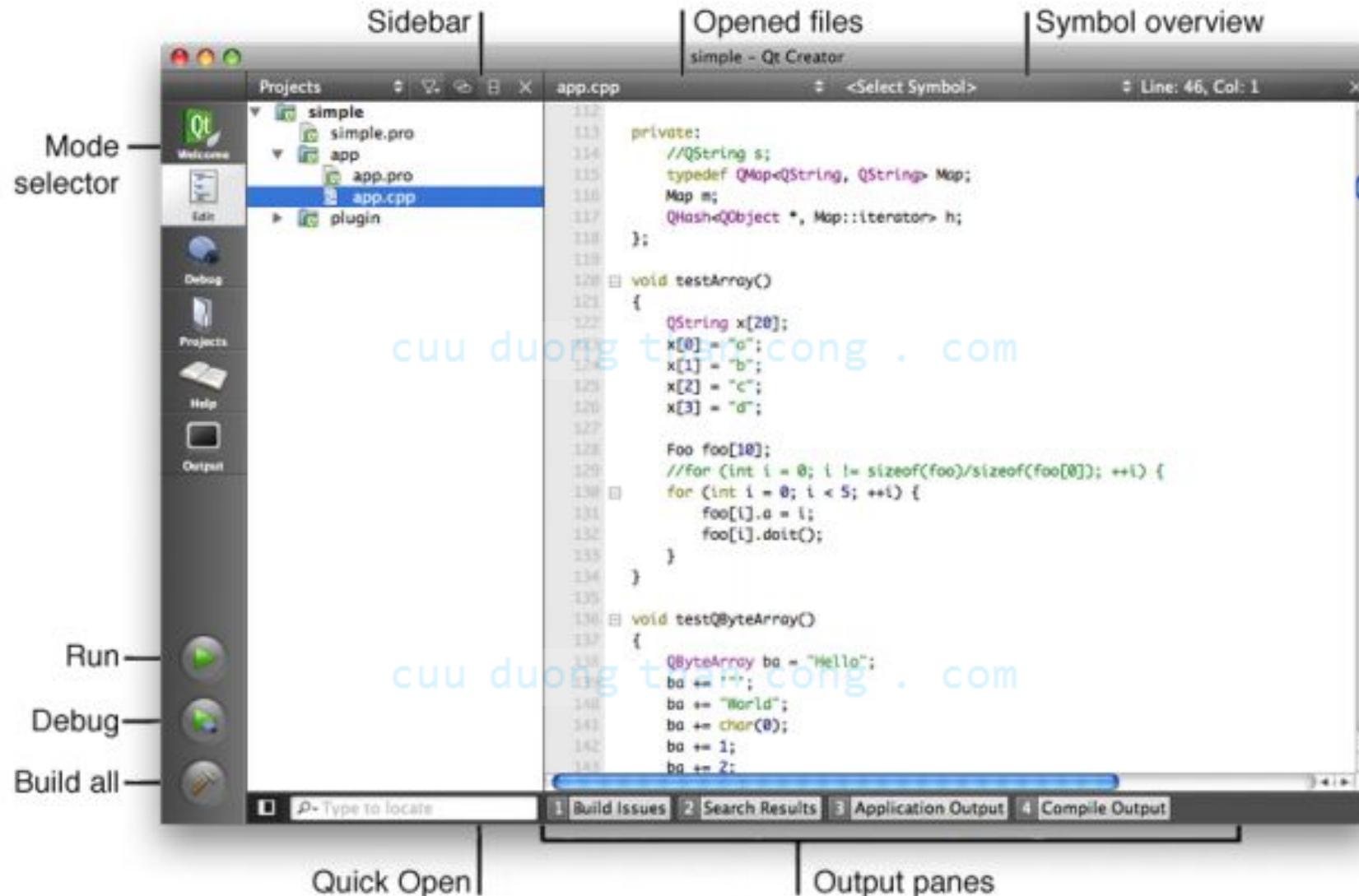
Môi trường phát triển

- IDE
 - 1) QT Creator
 - 2) Tích hợp vào Visual Studio, Eclipse
- Chương trình dịch: qmake
 - 1) Qmake for Windows
 - 2) Qmake for Linux
 - 3) Qmake for Embedded Linux

...



QT Creator





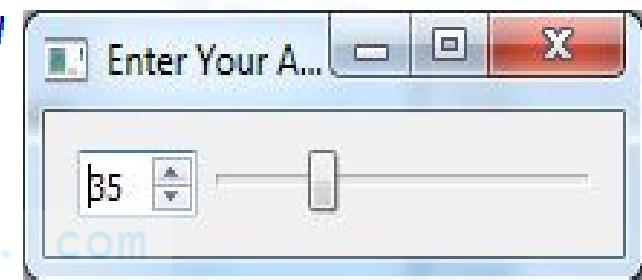
Cơ chế Signals and Slot của QT

- Signals: tương tự Event
- Slot: tương tự Event Handler

connect(sender, SIGNAL(signal), receiver, SLOT(slot));

VD: đồng bộ hai điều khiển trên QT

```
QObject::connect(spinBox, SIGNAL(valueChanged(int)),  
                  slider, SLOT(setValue(int)));  
  
QObject::connect(slider, SIGNAL(valueChanged(int)), . . .  
                  spinBox, SLOT(setValue(int)));\n
```





Chương trình HelloWorld

