W90N745 uClinux BSP User's Manual

Winbond Electronics Corp.

May 9, 2008

Histroy List:

Version	Date	Author	Comment
1.0	July 7, 2006		Initial Version
1.1	July 25, 2007		Update kernel config section
1.2	May 9, 2008		Change Header

UCLINUX BSP USER'S MANUAL

1 Introduction

Winbond uClinux is an embedded Linux kernel developed based on the Linux operating system, it supports the ARM hardware platform without the MMU installed. It supports almost all of the functions of Linux operation system, including the memory management, task scheduling, interrupt handling, and almost all of system calls that Linux supported.

For the file systems, the Winbond uClinux supports the ROMFS、RAMFS、 PROC、FAT、VFAT, and it can support other file system if desired.

ROMFS is a Read-Only file system, and it is the root file system of kernel. It uses for store the utilities, device files, and user configuration files. These files need to be saved in the directory of **romdisk**, and use the **genromfs** utility to generate the ROM file system image document.

RAMFS is a RAM based read/write file system, mounted to the directory **/usr**, it uses for the storing of some temporary files, its contents will be lost after the power off.

Besides, Winbond uClinux supports the TCP/IP、PPP、UDP, … and others network protocols. It supports the multi-threads operating environment. It also supports the dynamic kernel module installation and removal (insmod, rmmod).

The Winbond uClinux implements the **execve()** function to execute the **"FLAT"** file format user applications, the **"FLAT"** file format is the only supported file format supported on the wbLinux kernel.

Winbond uCLinux software package provides the following items:

- Based on uClinux-2.4.20
- arm-elf-gcc-3.0
- Sample application execute under uCLinux environment
- arm-elf-gcc-3.3 for C++ Applications

2 Target Processor

Winbond W90N745 - ARM7TDMI based MCU.

3 Supported Functions

Drivers： Ethernet MAC, four UARTs、 console, PS2 keyboard, keypad, USB host controller, USB mass storage device, I2C, USI, MTD, AC97, I2S.

File System： romfs、 ramfs、 proc、 fat、 vfat、ext2, jffs2, yaffs2

Interrupt handler： timer、 UART 、net、 host controller

Signal manage：default signals in linux kernel (SIGCHLD,SIGKILL,SIGALRM…)、user defined

Network Protocols：TCP, UDP, IP

4 *Development Platform*

Linux environement with a RedHat 6.x or higher version of Linux installed host computer with at least 800 MB free disk space.

5 *Installation Procedure*

Login a Linux PC.

5.1 Install the tools and development

1. Copy the **W90N745.tar.gz** to development directory, and then decompress it

```
$ tar xzvf w90n745.tar.gz
```

2. Use the root account:

```
$ su
$ Password:
# sh install.sh
```

3. Specify the absolute PATH that the SDK want to install, for example, '`/home/W90N745/`'

   After decompress, the ARM GNU development tools will be installed on the `/usr/local/arm_tools`, and the uClinux-dist will be decompress the `<installed directory>/ W90N745-uClinux` directory.

4. Exit the super user mode

```
# exit
```

5. After the installation, please logout and relogin to make sure "`/usr/local/arm_tools/bin`" is in your path, if not, you may set the compiler path manually

"`export PATH=/usr/local/arm_tools/bin:$PATH`"

NOTE: Winbond provides a defaul kernel configuration files in uClinux-dist/linux2.4.x/, .wb_ev_board. ***It is strongly urged that users load the default setting according to the target board before first time build the kernel.*** The configuration should be loaded in kernel configuration menu, Please refer to chapter 6 for detail about the kernel configuration

5.2 *Files installed*

Four directories will be created under <installed directory>/ W90N745-uClinux.

Directories	Comment
`uClinux-dist`	W90N745 uClinux Kernel Source Code
`romdisk`	ROM File System Tree for W90N745 Board SYSTEM
`image`	Built image, **romfs.img** is the root file system image. **linux.bin** is the kernel binary execution code
`TestApps`	Test program on uClinux

The arm_tools.tar.gz will be installed in `/usr/local/arm_tools`, arm_tools_3.3.tar.gz can be installed on `/usr/local/arm_tools_3.3` by manually later if support for C++ is required.

Tools installed under /usr/local/arm_tools/bin listed below:

`/usr/local/arm-tools/bin`	Bin Utilities include the compiler and linker etc, "install.sh" will add it to every user's PATH (/etc/Profile updated)
`/usr/local/arm-tools/arm-elf/inc`	Header files
`/usr/local/arm-tools/arm-elf/lib`	Linkable libraries(C and pthread library)

5.3 *Drivers and their location*

The driver, their location, support hardware and device file are listed below.

Device	Device File	Driver Name	Support Hardware
MAC		`w90n745_mac.c`	W90N745 EMC, with Davicom DM9161A, ICPLUS IP101A
UART	`ttyS[0~3]`	`w90n745_uart.c, w90n745_uart_?.c`	W90N745 UART[0~3]

PS/2		`w90n745_ps2.c`	PS/2 keyboard
Keypad	keypad	`w90n745_keypad.c`	W90N745 keypad interface
I2C	i2c[0~1]	`w90n745_i2c.c`	W90N745 I2C interface
USBH		`usb/*`	Mass storage device
USBD	usbclient	`w90n745_mass.c` `w90n745_vcom.c`	W90N745 USB device controller
Flash		`block/flash/*`	NOR flash
MTD	mtd?, mtdblock?	`mtd/*`	NOR flash and SLC NAND flash

Audio	dsp[0~1],
	mixer[0~1].
	0: I2S
	1:AC97
		`w90n745_audio.c`
		`w90n745_ac97.c`
		`w90n745_i2s.c`
			Burr-Brown PCM3003
			Realtek ALC203

USI	usi	`w90n745_usi.c`	W90N75 USI

5.4 *Build the kernel and test program*

- Build kernel

```
$ cd <installed directory>/uClinux-dist/

$ make clean;make dep; make
```

The built image linux.bin will be copied to ../image

- Build W90N745 test program

Enter any folder under "TestApps", type command "make", then a new image file "romfs.img" will be found at folder "image"

Or you can use the following commands to generate a new image file:

```
$ genromfs -d romdisk -f romfs.img
```

There are 2 methods to execute the application programs. One is enter the shell command prompt after system start up, key in the application program name and execute it. Another way is to put the application program name in a file named **"init"** in the **bin** directory, the **"init"** contains all of the programs that want to be executed immediately when the system startup.

6 Kernel configuration

User can type ether "make linux_menuconfig" or "make menuconfig"under uClinux-dist\, and select the option "Customize Kernel Settings", to enter the main menu of kernel configuration page. User can use arrow key to move high light item and use space key to select or un-select them.

It is strongly suggested run "make dep" after any configuration made and before build kernel.

6.1 Simplest configuration

The following items show the configuration of the most compact kernel.

```
System Type--->
      (Winbond) ARM system type
  (RAM) Kernel executes from
  (WINBOND-W90N745) Board Implementation
General setup  --->
          (ELF) Kernel core (/proc/kcore) format
Block devices  --->
           [*] ROM disk memory block device (blkmem)
 (W90N745)   FLASH type
File systems  --->
    [*] ROM file system support
Character devices  --->
          [*] Winbond W90N745 serial port support
     [*]   Support for console on Winbond W90N745 serial port
```

This configuration support the ramfs and romfs two basic file system, and only the ELF-flat format binary file can be executed.(no compressed flat format supported)

6.2 Detailed configuration

If more functions need to be supported on kernel, then before the kernel rebuild, it needs to do the selected kernel configuration, the following sessions describe the procedures to do configurations. These configurations are verified on the system. However, some of configurations of the network drivers, file systems, and network protocols are still on the alpha-test stage (such as, PPPoE), it needs to turn on the configuration of "**Code maturity level options ---> Prompt for development and/or incomplete code/drivers"**.

Due to the pin number limitation, some interfaces in W90N745 share the same group of pins, for examlpe PS/2 and UART2 and UART1 CTS/RTS, keypad and MAC, KPI and EMC, UART3 and audio. While one of those functions is enabled, the other functions will disappeared from kernel configuration menu, and eliminate the chance of configuration error.

6.2.1 Enable the code maturity selection

```
        Code maturity level options  --->
               [*] Prompt for development and/or incomplete code/drivers
```

6.2.2 Loadable module support

```
        Loadable module support  --->
                         [*] Enable loadable module support
```

6.2.3 Compressed FLAT file format support (ZFLAT)

```
        General setup  --->
                  [*]    Enable ZFLAT support
```

6.2.4 Character devices support

6.2.4.1 UART[1-3] support

```
      Character devices  --->
                        [*] Winbond W90N745 serial port [1-3]
                        [*] Enable serial port 1 CTS/RTS pins (Please touch
                        w90n745_uart_1.c if this setting changed)
```

6.2.4.2 I2C support

```
      Character devices  --->
                        [*] Winbond W90N745 I2C Module
```

6.2.4.3 USI support

```
      Character devices  --->
                        [*] Winbond W90N745 USI
```

6.2.4.4 Keypad support

```
      Character devices  --->
                        [*] Winbond W90N745 Keypad
```

6.2.4.5 PS/2 support

```
      Character devices  --->
                        [*] Winbond W90N745 ps/2 port support
                        [*] Virtual terminal
```

6.2.5 Network protocols and devices support

Select the "Networking support" in "General setup", then the menus of the "Networking options" and "Network drvice support" of main menu on the linux kernel will appear.

```
        General setup  --->
            [*] Networking support
```

6.2.5.1 TCP/IP protocol support

```
        Networking options  --->
                        [*] TCP/IP networking
```

6.2.5.2 PPP support (point-to-point protocol)

```
        Network device support  --->
                         [*] Network device support?
                         <*> PPP (point-to-point protocol) support
                         [*]   PPP Deflate compression
                         [*]   PPP over Ethernet (EXPERIMENTAL)
```

6.2.5.3 W90N745 Ethernet Controller device driver support

```
        Network device support  --->
               Ethernet (10 or 100Mbit)  --->
                         [*] Ethernet (10 or 100Mbit)
                     [*]   Winbond W90N745 Embedded Ethernet support
```

6.2.6 File systems support

6.2.6.1 ext3 support

```
        File systems  --->
             [*] Ext3 journalling file system support
             Partition Types  --->
              [*] Advanced partition selection
              [*]   PC BIOS (MSDOS partition tables) support
```

6.2.6.2 FAT-based file systems(MS-DOS, VFAT) support

```
        File systems  --->
      [*] DOS FAT fs support
        [*]   MSDOS fs support
       [*]   VFAT (Windows-95) fs support
     Partition Types  --->
               [*] Advanced partition selection
               [*]   PC BIOS (MSDOS partition tables) support
               Native Language Support  --->
```

```
                 [*] Codepage 437 (United States, Canada) (NEW)
                 [*] NLS ISO 8859-1 (Latin 1; Western European Languages) (NEW)
```

6.2.6.3 ext2 support

```
        File systems  --->
                   [*] Second extended fs support
             Partition Types  --->
              [*] Advanced partition selection
               [*]   PC BIOS (MSDOS partition tables) support
```

6.2.6.4 /proc file system support

```
        File systems  --->
                     [*] /proc file system support
```

6.2.6.5 nfs support

```
        File systems  --->
           Network File Systems  --->
                  [*] NFS file system support
               [*]   Provide NFSv3 client suppor
```

6.2.6.6 devfs support

```
        File systems  --->
                  [*] /dev file system support (EXPERIMENTAL)
                  [*]   Automatically mount at boot
```

6.2.6.7 YAFFS support

This option shows up if MTD NAND support is enabled.

```
        File systems  --->
               [*] YAFFS2 file system support
```

6.2.6.8 JFFS2 support

This option shows up if MTD support is enabled.

```
        File systems  --->
               [*] Journalling Flash File System v2 (JFFS2) support
```

6.2.6.9 ISO9660 support

```
      File systems  --->
               [*]   ISO 9660 CDROM file system support
               [*]     Microsoft Joliet CDROM extentions
```

6.2.7 USB device support

6.2.7.1 USB host

First, configure the the support of USB host controller - OHCI host interface.

```
        USB support  --->
                       [*] Support for USB
                       [*]  OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support
```

USB Mass Storage device support

```
        SCSI support  --->
                          [*] SCSI support
                          [*]   SCSI disk support
        USB support  --->
                         [*]   USB Mass Storage support
```

If USB CD-ROM support is required, enable following option as well

```
        SCSI support  --->
                          [*]   SCSI CD-ROM support
```

6.2.7.2 USB device

```
          --- Winbond USB Device 1.1 drivers
          Support for W90N745 USB Device 1.1  --->
               [*] Support W90N745 USB Device
               (MASS)  usbd function support
```

6.2.8 MTD

6.2.8.1 NOR flash

```
        Memory Technology Devices (MTD)  --->
          [*] Memory Technology Device (MTD) support
          [*]   MTD partitioning support
          [*]   Direct char device access to MTD devices
          [*]   Caching block device access to MTD devices
          RAM/ROM/Flash chip drivers  --->
               [*]   Detect flash chips by Common Flash Interface (CFI) probe
               [*]   Flash chip driver advanced configuration options
               (NO) Flash cmd/query data swapping
               [*]   Specific CFI Flash geometry selection
               [*]     Support  16-bit buswidth
               [*]     Support 1-chip flash interleave
               [*]   Older (theoretically obsoleted now) drivers for non-CFI
           chips
               [*]   AMD compatible flash chip support (non-CFI)
          Mapping drivers for chip access  --->
               [*]   Support for non-linear mappings of flash chips
               [*]   W90N745 board mappings
```

6.2.8.2 NAND flash

```
        Memory Technology Devices (MTD)  --->
          [*] Memory Technology Device (MTD) support
          [*]   MTD partitioning support
          [*]   Direct char device access to MTD devices
          [*]   Caching block device access to MTD devices
          NAND Flash Device Drivers  --->
               [*]   NAND Device Support
               [*]   NAND Flash device on WINBOND board
               (128MB_2k_page_size) NAND is to be used in the system
```

7 Kernel module programming

If the developer wants to the do the programming of kernel module (such as, device drivers), the kernel module can only calls the kernel functions, and its associated header files. Use the gcc to compile to .o object files. There are 2 ways to install the .o object files into the kernel.

Method 1:

Modify the makefile for the kernel, add the .o object file name into the link section, rebuild the kernel to get a updated kernel, tftp the Linux binary code to development boards.

Method 2:

Put the **.o** file to the **romdisk**, build the **romfs.img**. Download it to the development board, run it. On the shell command prompt, key in **insmod XXX.o**, the **insmod** will install the module into the kernel, and it use the **rmmod XXX** to remove the module from kernel. User can use **lsmod** to display all of the installed modules.

8 Virtual debug device usage

If the default console device (dev/console) or the serial port wants to be used as other purpose (such as modem), this platform provides another "Virtual debug device" - /dev/vdd0 as the console device to save the message print to *stdout* and *strerr*. To enable this device, it needs to configur it into the kernel.

```
        Character devices  --->

                                 [*] Virtual debug device support
```

If this configuration is selected, then the CONFIG_VDD was defined on the include/linux/autoconf.h, then the init functions in init/main.c will be compiled to use the /dev/vdd0 as the default console.

```
          #ifndef CONFIG_VDD
               if (open("/dev/console", O_RDWR, 0) < 0)
                    printk("Warning: unable to open an initial console.\n");
          #else
               if (open("/dev/vdd0", O_RDWR, 0) < 0)
                    printk("Warning: unable to open an initial console.\n");
          #endif
```

```
dup(0);
dup(0);
……
execve("/bin/sh",argv_sh,envp_init);
```

Open the /dev/vdd0 as the fd 0, the call the dup(0) twice to get the fd 1(stdout), and 2(stderr), then call execve() to execute the user program. All of the printed message will be redirected to /dev/vdd0.

Following example illustrate how to check the message recorded on `/dev/vdd0`

The user program test.c:

```
#include <stdio.h>

int main(void)
{
     int i=0;
     while(1)
     {
          printf("hello ");
          fflush(stdout);
          fprintf(stderr,"world %d\n",i++);
          if(i == 1000)
          i=0;
     }
     return 0;
}
```

Copy the compiled and executable program test to romdisk/bin/, change it to executable permission, and make a device file vdd0. The /dev/vdd0 use major device number 99, minor device number 0.

```
$ mknod vdd0 c 99 0
```

Modify the `romdisk/bin/init` as followed,

```
mount -t proc none /proc
mount -t ramfs none /usr
mount -t ramfs none /swap
ifconfig eth1 10.130.2.103 netmask 255.255.0.0
```

```
inetd&
test
```

Use the genromfs to generate the romfs.img, and the updated linux.bin with /dev/vdd0 supported. When system startup, telnet to the target system, then 'cat /dev/vdd0' to get the message printed by test.

Currently, the buffer size provides by the /dev/vdd0 is 2KB, if the buffer size needs to be increase, the Line 27 in uClinux-dist/linux-2.4.x/drivers/char/vdd.c can be changed to set the desired buffer size, then rebuild the kernel.

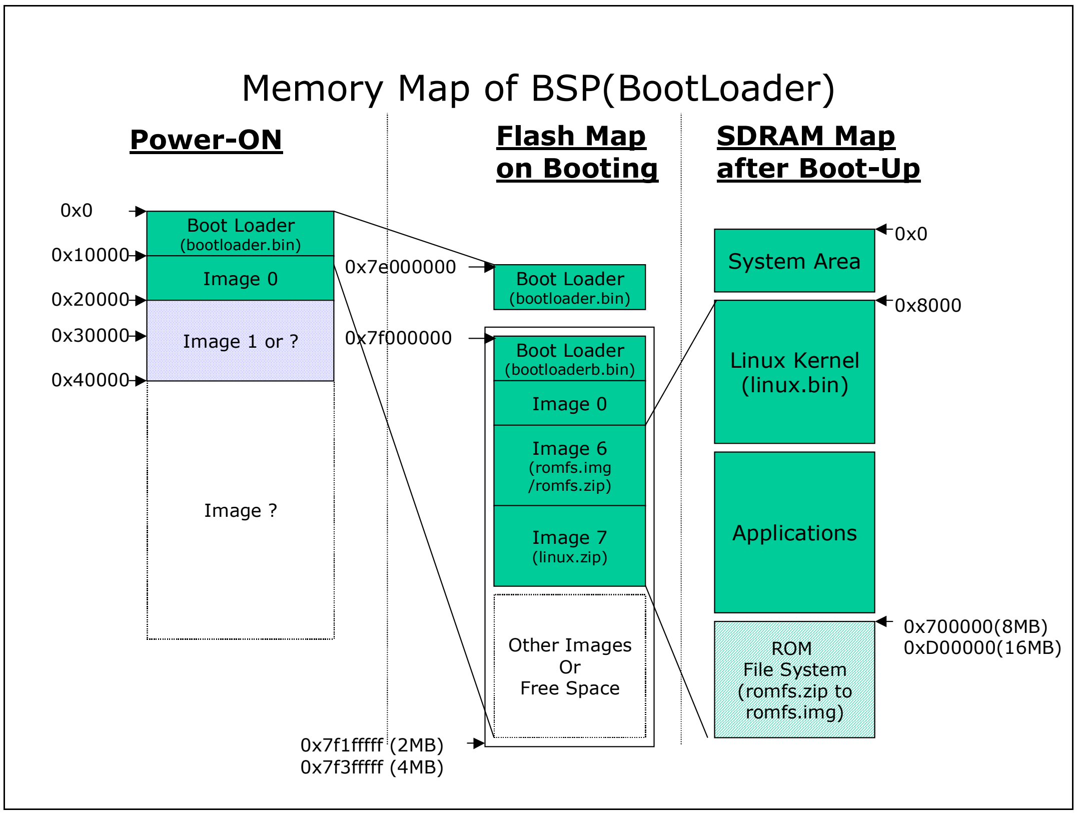
```
#define DEBUGBUF_SIZE   (1024 * 2)
```

9 Update Kernel and ROM File System

This chapter describes how to update kernel and ROM file system onto FLASH, as well as some kernel configuration regarding the ROM file system location. Users could also refer to "**W90N745 Bootloader users manual.pdf**" for more detail of bootloader operation

The bootloader will scan the image sequentially, and process the active images according to their attribute. So you need to configure ROM file system image as the image 6(or other number smaller than 7), and Linux kernel as image 7(or other unused ID number grater than ROM file system's image ID). The address of the images in FLASH is not relevant as long as they are not overlapped. The following table lists the basic flash map of uClinux system and the bootloader.

We suggest user compress the Linux kernel image to save flash space. And the ROM file system image could also be a compressed if you want to copy it to SDRAM during run time. Figure below shows the Memory Map of the BootLoader after loads uCLinux kernel and Romfs to SDRAM.



9.1 Flash Configurations

The Linux kernel image should always be compressed to save FLASH space, and uncompress to SDRAM address 0x8000 for executing. The command for updating kernel listed below:

```
bootloader> ft 7 linux.zip 0x7f020000 0x8000 –acxz
```
or
```
bootloader> fx 7 linux.zip 0x7f020000 0x8000 –acxz
```

Three types of ROM file system configuration are supported. it can either be compressed or not. If ROM file system is not compressed, it could reside in ether SDRAM or FLASH during run time.

9.2 Configuration method 1

ROM file system image (romfs.img) located at 0x7F0C0000 as image 6, runtime address 0x7F0C0000. Command for updating ROM file system listed below:

```
bootloader> ft 6 romfs.img 0x7F0C0000 0x7F0C0000 –a
```
or
```
bootloader> fx 6 romfs.img 0x7F0C0000 0x7F0C0000 –a
```

9.3 Configuration method 2

Compressed ROM file system image (`romfs.zip`) located at 0x7F0C0000 as image 6, runtime address 0x700000. Command for updating ROM file system listed below:

```
bootloader> ft 6 romfs.zip 0x7F0C0000 0x700000 –acz
or
bootloader> fx 6 romfs.zip 0x7F0C0000 0x700000 –acz
```

9.4 Configuration method 3

ROM file system image (`romfs.img`) located at 0x7F0C0000 as image 6, runtime address 0xD00000. Command for updating ROM file system listed below:

```
bootloader> ft 7 romfs.img 0x7F0C0000 0x700000 –ac
or
bootloader> fx 7 romfs.img 0x7F0C0000 0x700000 –ac
```

9.5 Adjust ROMFS starting address

The default ROMFS memory location is 0x700000 in SDRAM. Here use the chang of ROMFS location to 0x60000 as an example to point out the modification needs to be take:

```
  linux-2.4.x\drivers\block\blkmem.c in blkmem_init()
  arena[i].address=0x600000;
linux-2.4.x/arch/armnommu/kernel/setup.c
 #define MEM_SIZE (6*1024*1024)
make menuconfig
 Change the DRAM Size from 0xD00000 to 0x600000
```

9.6 Load the images to SDRAM

During the development stage, user may choose to load images to SDRAM instead writing them to FLASH, below listed the steps to do so:

Load ROMFS to SDRAM:
```
mt 0x700000 or mx 0x700000
```

Load kernel to SDRAM:
```
mt 0x8000 or mx 0x8000
```

Execute image:
```
g 0x8000
```

9.7 Build the production F/W image of flash

The final flash image on production can be built by the mkrom tool, the mkrom tool will combine the bootloader image (bootloader.bin), image 0 for system configuration information, image 6 ROM file system image (romfs.img), and image 7 compressed linux kernel (linux.zip) to a final image of flash.

User can consult "Make a Production ROM" for the usage to mkrom tool.

10 Shell and other applications

Shell is the basic application on the Linux system, default shell provided in BSP is "sh". "sh" uses the current directory as the prompting string. Commands can be executed under shell. (It works the same way as PC Linux). Key in help under shell will display the internal commands provided by shell.

command	**description**	**usage**
cat	Show file on screen	Cat filename
cd	change current directory	cd [directory]
chgrp	change the group membership of each FILE to GROUP	chgrp GROUP FILE...
chmod	change file/directory mode	chmod mode file/dir
chown	change file/directory own	chown group:user file/dir
cmp	cmpare two files	cmp file1 file2
date	Get/set date	date [MMDDhhmm[YYYY]]
cp	copy source to destination	cp file1 file2
df	Show information about	df [device]

	the filesystem on which each FILE resides, or all filesystems by default.	
echo	Output the ARGs or redirectory to file	echo arguments [> filename]
exec	Exec FILE, replacing this shell with the specified program	exec file
exit	Exit the shell with a status of N. If N is omitted, the exit status is that of the last command executed	exit [N]
free	show memory status	free
help	show help message	help
hexdump	hex dump file	hexdump file
hostname	show host name	hostname
kill	send signal to process	kill [-s sigspec | -n signum | -sigspec] [pid | job]...\n or kill -l [sigspec]
ln	Create a link to the specified TARGET	ln –s file1 file2
ls	List information about the FILEs	ls [options]
mkdir	Create the DIRECTORY	mkdir dirname
mknod	Create device file	mknod type major minor

more	File perusal filter	more filename
mount	Mount file system	mount –t type device dir
mv	Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY	mv source dest
printenv	Print environment varables	printenv
pid	Show current process	pid
ps	Show process information	ps
pwd	Show current dirctory	pwd
quit	Quit current process	quit
rm	Remove file	rm file
rmdir	Remove dir	rmdir dir
sleep	sleep several seconds	sleep number
setenv	Set environment varable	setenv var value
source	Run command in file	source file
sync	System sync	sync
touch	Update the access and modification times of each FILE to the current time	touch [option] file
umask	The user file-creation mask is set to MODE	umask octal number

umount	Umount file system	umount dir

a. ifconfig

It used to configure the network interfaces, such as,

```
# ifconfig eth0 192.168.0.10 netmask 255.255.255.0
```

b. route

It used to manipulates the IP routing tables

```
# route add default netmask 255.255.255.0 gw 192.168.0.1 eth0
```

c. dhcpc

dhcp client application

```
# dhcpc eth0
```

d. init

Script "init", which located under "romdisk\bin" directory, contains all the applications want to be executed while starting up. There are some commands in "init" which have been marked. They are examples of using "ifconfig", mount command etc. You may un-comment and modify them to fit your need.

Please note, the last user program in init should be an endless loop program, such as "sh", or it will generate the Kernel panic.

11 Example codes

There are some example codes in the TestApps/ directory. Following table list the description of each test applications.

Application	Description
audio_test	AC97/IIS sample program The test sequence listed below: 1. Select Device

	W90N745 Audio can support two kind of codec standard, IIS and AC97. /dev/dsp0 and /dev/mixer0 (IIS) /dev/dsp1 and /dev/mixer1 (AC97) 2. IOCTL Test First make sure that the device is audio device, which based on OSS standard. Then get internal fragment size of audio driver. It is most efficient if user read/write audio device with the size. And set volume to maximum. 3. Play and Sample Rate Test Test all sample rate supported by codec (8K, 11.025K, 16K, 22.05K, 24K, 32K, 44.1K, 48K). *Note: at least 8k.pcm must be provided at current directory. 4. Mixer Test Change volume to 33%, 66% and 100% for mixer test 5. Record Test In this part, the program will record 6 second from input path (LINE IN, or MIC) and then playback. 6. Poll Test First test whether driver can work in Non-black mode. And then test poll system call of audio driver.
boa-dev-ssl	Boa web server. Following list the steps to build this application. There's a document `boa-dev-ssl/boa-dev/Help/Cgi.doc` describes how to implement CGI function in BOA under uClinux. 1. Execute openssl/createlink.sh before build the application 2. Execute boa-dev/mkcert.sh and copy the *.pem to romdisk/etc 3. Remove -DSERVER_SSL if you don't need SSL support 4. Copy web pages into romdisk/etc/Html

cpp_sample	C++ sample program, You should have install the arm_tools_3.3.4 (to "/usr/local" directory) when you compile it.
i2c	I2C sample program. This demo application will access the EEPROM on EV board
keypad	Keypad sample program. This demo application can display the key user pressed on console.
vcom	USB VCOM device demo program
mass	USB mass storage device sample program Usage: mass [device] Example: mass /dev/sda1 Note: Don't mount [device], or the result is unpredictable.
ppp-oe-modem	PPPoE & PPP dial up utility PPPoE: 1. Edit and copy chap-secret, pap-secret, and pppoe-options into romdisk/etc/ppp 2. Usage: pppd <dial_type> <interface> <username> <password> PPP: 1. Edit and copy chap-secret, pap-secret to romdisk/etc/ppp. mppp.cfg and fccc.scr to romdisk/etc 2. Build applications in mppp, and pppd 3. Usage: mppp
rtc	RTC sample program This program support "set" command to set rtc, “read” command to read current time, and “mode” command to get display mode. Examples: `#>rtc --set "2008/2/15 15:56:15`

	`#>rtc --read`
	
	`#>rtc --mode`

thread_demo	Thread sample program. There are 7 thread demo application in this directory demonstrate different APIs of pthread library.
uart	UART sample program. This test program only test the send data on the uart port. Usage: uarttest <port num> Example: uarttest 1
usi	USI sample program. This application will try to access the serial flash on EV board.

12 System call and library

Almost all of the functions are compatible with the POSIX defined functions, a little part of them were designed for the convenience of program developing. The verification programs are included on the SDK. Note that the functions end with _**r** have the same functionality as no _**r** appended, but functions end with _**r** indicate that these functions are reentrant, it indicates that they can be used on the multi-thread operation environment without the risk of inconsistence.

12.1 time/

- char *asctime(const struct tm * timeptr);
- char *asctime_r(const struct tm * timeptr, char * buf);
- clock_t clock();
- char * ctime(const time_t * timep);
- char *ctime_r(const time_t * timep, char * buf);
- struct tm * gmtime(const time_t * timep);
- struct tm * gmtime_r(const time_t * timep, struct tm * tp);
- struct tm * localtime(const time_t * timep);
- struct tm * localtime_r(const time_t * timep, struct tm * tp);
- time_t mktime (struct tm *tp);
- size_t strftime(char *s , size_t maxsize , const char *format , register const struct tm *tp);
- void tzset (void);
- time_t time(time_t *t);
- int stime(time_t *t);
- int ftime(struct timeb *tp);

12.2 regex/

- int *regcomp(regex_t *preg, const char *regex, int cflags);
- int regexec(const regex_t *preg, const char *string, size_t nmatch, regmatch_t pmatch[], int eflags);

- size_t regerror(int errcode, const regex_t *preg, char *errbuf, size_t   errbuf_size);
- void regfree(regex_t *preg);

12.3 termios/

- int tcgetattr(int fd, struct termios *term);
- int tcsetattr(int fildes, int optional_actions, struct termios *termios_p);
- int tcflush(int fd,int queue_selector);
- speed_t cfgetispeed(struct termios *tp);
- speed_t cfgetospeed(struct termios *tp);
- int cfsetospeed(struct termios *tp, speed_t speed);
- int cfsetispeed(struct termios *tp, speed_t speed);
- void cfmakeraw(struct termios *t);
- int tcsendbreak (int fd, int duration);
- int tcflow (int fd, int action);
- pid_t tcgetpgrp (int fd);
- int tcsetpgrp (int fd, pid_t pgrpid);

12.4 stdio/

- int fputc(int ch, FILE *fp);
- int fgetc(FILE *fp);
- int fflush(FILE *fp);
- char *fgets(char *s, size_t count, FILE *f);
- char *gets(char *str);
- int fputs(const char *str, FILE *fp);
- int puts(const char *str);
- int puts(const char *str);
- size_t fread(void *buf, size_t size, size_t nelm, FILE *fp);
- size_t fwrite(const void *buf, size_t size, size_t nelm, FILE *fp);
- void rewind(FILE * fp);
- int fseek(FILE *fp, long offset, int   ref);
- long ftell(FILE * fp);
- int fclose(FILE *fp);
- int getc(FILE *stream);
- int putc(int c, FILE *stream);
- int ungetc(int c, FILE *fp);
- int printf(const char * fmt, ...);
- int sprintf(char * sp, const char * fmt, ...);
- int fprintf(FILE * fp, const char * fmt, ...);
- int vprintf(const char *fmt, va_list ap);
- int vsprintf(char * sp, const char *fmt, va_list ap);
- int vfprintf(FILE *op, const char *fmt, va_list ap);
- int putchar(char c);

- int getchar();
- void setbuf(FILE *stream, char *buf);
- void setbuffer(FILE *stream, char *buf, size_tsize);
- void setlinebuf(FILE *stream);
- int setvbuf(FILE * fp, char * buf, int mode, size_t size);
- int scanf(const char *format, ...);
- int fscanf(FILE *stream, const char *format, ...);
- int sscanf(const char *str, const char *format, ...);
- int vscanf(const char *format, va_list ap);
- int vsscanf(const char *str, const char *format, va_list ap);
- int vfscanf(FILE *stream, const char *format, va_list ap);

12.5 string/

- size_t strcspn(register const char *string, const char *set);
- size_t strlen(const char * str);
- char * strcat(char *d, const char * s);
- char * strcpy(char *d, const char * s);
- int strcmp(const char *d, const char * s);
- char * strncat(char *d, const char *s, size_t l);
- char * strncpy(char *d, const char *s, size_t l);
- int strncmp(const char *d,const char *s, size_t l);
- char *strchr(const char * s, int c);
- char * strrchr(const char * s, int c);
- char * strdup(const char * s);
- void *memcpy(void *d, const void *s, size_t l);
- void * memccpy(void *d, const void *s, int c, size_t l);
- void * memchr(const void * str, int c, size_t l);
- void * memset(void * str, int c, size_t l);
- int memcmp(const void *s,const void *d, size_t l);
- void *memmove(void *d, const void *s, size_t l);
- char *strpbrk(register const char *str, const char *set);
- size_t strspn(const char *s, const char *accept);
- char *strstr(const char *s1, const char *s2);
- char *strtok(register char *s, register const char *delim);
- char *strtok_r (char *s, const char *delim, char **save_ptr);
- char *stpcpy (char *dest, const char *src);
- int strcasecmp(const char *s, const char * d);
- char *strcasestr(const char *str1, const char * str2);
- int strncasecmp(const char *s, const char *d, size_t l);
- char *strsep(char **pp, const char *delim);

12.6 stdlib

- void *calloc(size_t num, size_t size);
- void *malloc(size_t len);
- void free(void * ptr);
- void *realloc(void * ptr, size_t size);
- void abort();
- int atexit(void (*function)(void));
- int atoi(const char *nptr);
- long atol(const char *nptr);
- long long atoll(const char *nptr);
- void *bsearch(const void *key, const void *base, size_t nmemb,   size_t size, int (*compar)(const void *, const void *));
- div_t div(int numer, int denom);
- void exit(int rv);
- char *getenv(const char *var);
- int getpt (void);
- int grantpt (int fd);
- int abs(int j);
- long int labs(long int j);
- ldiv_t ldiv(long int numer, long int denom);
- intmax_t imaxabs(intmax_t j);
- long long int llabs(long long int j);
- char * mkdtemp (char *template);
- int mkstemp(char *template);
- char *mktemp(char *template);
- int on_exit(void (*function)(int , void *), void *arg);
- char *ptsname (int fd);
- int ptsname_r (int fd, char *buf, size_t buflen);
- void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *,
  const void *));
- int rand(void);
- int rand_r (unsigned int *seed);
- char * initstate (unsigned int seed, char *arg_state, size_t n);
- int initstate_r (unsigned int seed, char *arg_state, size_t n,
  struct random_data *buf);
- long int random(void);
- int random_r (struct random_data *buf, int32_t * result);
- char *setstate(char *state);
- int setstate_r (char *arg_state, struct random_data *buf);
- void srandom (unsigned int x);
- int srandom_r (unsigned int seed, struct random_data *buf);
- char *realpath(const char *path, char *resolved_path);

- int clearenv (void);
- int putenv (char *string);
- int setenv(const char *name, const char *value, int overwrite);
- void unsetenv(const char *name);
- long int strtol(const char *nptr, char **endptr, int base);
- long int strtoimax (const char *nptr, char **endptr, int base);
- unsigned long int strtoul(const char *nptr, char **endptr, int base);
- int system(char *command);
- int unlockpt (int fd);
- __ptr_t valloc (size_t size);

12.7 inet/

- int accept(int s, struct sockaddr *addr, socklen_t * addrlen);
- int bind(int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);
- int connect(int sockfd, const struct sockaddr *saddr, socklen_t addrlen);
- int __dns_lookup(const char *name, int type, int nscount, char **nsip,
  unsigned char **outpacket, struct resolv_answer *a);
- struct ether_addr *ether_aton(const char *asc);
- struct ether_addr *ether_aton_r(const char *asc, struct ether_addr *addr);
- char *ether_ntoa(const struct ether_addr *addr);
- char *ether_ntoa_r(const struct ether_addr *addr, char *buf);
- const char * gai_strerror (int code);
- static int addrconfig (sa_family_t af);
- void freeaddrinfo (struct addrinfo *ai);
- static int gaih_inet (const char *name, const struct gaih_service *service,
  const struct addrinfo *req, struct addrinfo **pai);
- static int gaih_inet_serv (const char *servicename, const struct gaih_typeproto
  *tp,const struct addrinfo *req, struct gaih_servtuple *st);
- int getaddrinfo (const char *name, const char *service,
  const struct addrinfo *hints, struct addrinfo **pai);
- struct hostent *gethostbyaddr (const void *addr, socklen_t len, int type);
- int gethostbyaddr_r (const void *addr, socklen_t len, int type, struct hostent *
  result_buf, char * buf, size_t buflen, struct hostent ** result,int * h_errnop);
- struct hostent *gethostbyname(const char *name);
- struct hostent *gethostbyname2(const char *name, int family);
- int gethostbyname2_r(const char *name, int family, struct hostent * result_buf,
  char * buf, size_t buflen,struct hostent ** result,int * h_errnop);
- int gethostbyname_r(const char * name,struct hostent * result_buf,char * buf,
  size_t buflen,struct hostent ** result,int * h_errnop);
- void endhostent (void);
- struct hostent *gethostent (void);
- void sethostent (int stay_open);

- int getnameinfo (const struct sockaddr *sa, socklen_t addrlen, char *host,
  socklen_t hostlen, char *serv, socklen_t servlen, unsigned int flags);
- struct netent *getnetbyaddr (uint32_t net, int type);
- struct netent *getnetbyname(const char *name);
- void endnetent(void);
- struct netent * getnetent(void);
- void setnetent(int f);
- int getpeername(int sockfd, struct sockaddr *addr, socklen_t * paddrlen);
- void endprotoent(void);
- struct protoent * getprotobyname(const char *name);
- struct protoent * getprotobynumber(int proto);
- struct protoent * getprotoent(void);
- void setprotoent(int f);
- void endservent(void);
- struct servent *getservbyname(const char *name, const char *proto);
- int getservbyname_r(const char *name, const char *proto, struct servent *
  result_buf, char * buf, size_t buflen, struct servent ** result);
- struct servent * getservbyport(int port, const char *proto);
- int getservbyport_r(int port, const char *proto, struct servent * result_buf,
  char * buf, size_t buflen, struct servent ** result);
- struct servent * getservent(void);
- int getservent_r(struct servent * result_buf, char * buf, size_t buflen,
  struct servent ** result);
- void setservent(int f);
- int getsockname(int sockfd, struct sockaddr *addr, socklen_t * paddrlen);
- int getsockopt(int fd, int level, int optname, __ptr_t optval,;
- long int gethostid(void);
- int sethostid(long int new_id);
- unsigned long inet_addr(const char *cp);
- int inet_aton(const char *cp, struct in_addr *inp);;
- unsigned long inet_lnaof(struct in_addr in);
- struct in_addr inet_makeaddr(unsigned long net, unsigned long host);
- u_int32_t inet_network(const char *cp);
- u_int32_t inet_netof(struct in_addr in);
- char *inet_ntoa(struct in_addr in);
- char *inet_ntoa_r(struct in_addr in, char *buf);
- int listen(int sockfd, int backlog);
- const char *inet_ntop(int af, const void *src,char *dst, size_t cnt);
- int inet_pton(int af, const char *src, void *dst);
- ssize_t recv(int sockfd, __ptr_t buffer, size_t len, int flags);
- ssize_t recvfrom(int sockfd, __ptr_t buffer, size_t len, int flags,struct sockaddr *to,
  socklen_t * tolen);

- ssize_t recvmsg(int sockfd, struct msghdr *msg, int flags);
- int res_init(void);
- void res_close(void);
- int res_query(const char *dname, int class, int type,unsigned char *answer,
  int anslen);
- ssize_t send(int sockfd, const void *buffer, size_t len, int flags);
- ssize_t sendmsg(int sockfd, const struct msghdr *msg, int flags);
- ssize_t sendto(int sockfd, const void *buffer, size_t len, int flags,
  const struct sockaddr *to, socklen_t tolen);;
- int setsockopt(int fd, int level, int optname, const void *optval, socklen_t optlen);
- int shutdown(int sockfd, int how);
- int socket(int family, int type, int protocol);
- int socketpair(int family, int type, int protocol, int sockvec[2]);

12.8 types.h

- isalnum(c);
- isalpha(c);
- isascii(c);
- iscntrl(c);
- isdigit(c);
- isgraph(c);
- islower(c);
- isprint(c);
- ispunct(c);
- isspace(c);
- isupper(c);
- isxdigit(c);
- toupper(c);
- tolower(c);
- _toupper(c)
- _tolower(c);
- toascii(c);

12.9 error/

- char *strerror_r(int err,char *retbuf,unsigned int n);
- char *strerror(int err);
- void perror(const char * str);

12.10 misc/

- void __assert(const char *assertion, const char * filename, int linenumber,
  register const char * function);
- int alphasort(const void * a, const void * b);

- int closedir(DIR * dir);
- int dirfd(DIR * dir);
- DIR *opendir(const char *name);
- struct dirent *readdir(DIR * dir);
- int readdir_r(DIR *dir, struct dirent *entry, struct dirent **result);
- void rewinddir(DIR * dir);
- int scandir(const char *dir, struct dirent ***namelist, int (*selector) (const struct dirent *), int (*compar) (const void *, const void *));
- void seekdir(DIR * dir, long int offset);
- long int telldir(DIR * dir);
- int lockf (int fd, int cmd, off_t len);
- int fnmatch(const char *pattern, const char *string, int flags);
- int glob(const char *pattern, int flags, int errfunc(const char * epath, int eerrno), glob_t *pglob);
- void globfree(glob_t *pglob);
- int addmntent(FILE * filep, const struct mntent *mnt);
- int endmntent(FILE * filep);
- struct mntent *getmntent(FILE * filep);
- struct mntent *getmntent_r (FILE *filep, struct mntent *mnt, char *buff, int bufsize);
- char *hasmntopt(const struct mntent *mnt, const char *opt);
- FILE *setmntent(const char *name, const char *mode);
- char *setlocale(int category, register const char *locale);
- struct lconv *localeconv(void);
- char *nl_langinfo(nl_item item);
- int regcomp(regex_t *preg, const char *regex, int cflags);
- int regexec(const regex_t *preg, const char *string, size_t nmatch, regmatch_t pmatch[], int eflags);
- void regfree(regex_t *preg);
- size_t regerror(int errcode, const regex_t *preg, char *errbuf, size_t errbuf_size);
- int hcreate (size_t nel);
- int hcreate_r (size_t nel, struct hsearch_data *htab);
- void hdestroy (void);
- void hdestroy_r (struct hsearch_data *htab);
- ENTRY *hsearch (ENTRY item, ACTION action);
- int hsearch_r (ENTRY item, ACTION action, ENTRY **retval, struct hsearch_data *htab);
- void *tsearch(const void *key, void **vrootp, __compar_fn_t compar);
- void *tfind(const void *key, void * const *vrootp, __compar_fn_t compar);
- void *tdelete (const void *key, void **rootp, int (*compar)(const void *, const void *));
- void twalk (const void *root, void (*action) (const void *nodep, const VISIT which,

const int depth));

- void *lfind(const void *key, const void *base, size_t *nmemb, size_t size,
  int (*compar)(const void *, const void *));
- void *lsearch(const void *key, void *base, size_t *nmemb, size_t size,
  int (*compar)(const void *, const void *));
- void insque(struct qelem *elem, struct qelem *prev);
- void remque (void *elem);
- int statvfs (const char *file, struct statvfs *buf);
- int fstatvfs (int fd, struct statvfs *buf);
- void closelog(void);
- void openlog(const char *ident, int logstat, int logfac);
- int setlogmask(int pmask);
- void syslog(int pri, const char *fmt, ...);
- void vsyslog(int pri, const char *fmt, va_list ap);
- int semget (key_t key, int nsems, int semflg);
- int semctl(int semid, int semnum, int cmd, ...);
- int semop (int semid, struct sembuf *sops, size_t nsops);
- void * shmat (int shmid, const void *shmaddr, int shmflg);
- int shmctl (int shmid, int cmd, struct shmid_ds *buf);
- int shmdt (const void *shmaddr);
- int shmget (key_t key, size_t size, int shmflg);
- int msgctl (int msqid, int cmd, struct msqid_ds *buf);
- int msgget (key_t key, int msgflg);
- int msgrcv (int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);
- int msgsnd (int msqid, const void *msgp, size_t msgsz, int msgflg);
- key_t ftok (char *pathname, char proj);
- int endttyent(void);
- struct ttyent * getttyent(void);
- struct ttyent * getttynam(const char *tty);
- int setttyent(void);
- void endutent(void);
- struct utmp *getutent(void);
- struct utmp *getutid(struct utmp *ut);
- struct utmp *getutline(struct utmp *ut);
- void pututline(struct utmp *ut);
- struct utmp *getutline(struct utmp *ut);
- void utmpname(const char *file);
- void updwtmp(const char *wtmp_file, const struct utmp *ut);

12.11 sysdeps/

- void _exit(int status);
- ssize_t read(unsigned int fd, char * buf, size_t count);

- ssize_t write (int __fd, __const void *__buf, size_t __n);
- int open (const char * fn, int flags, mode_t mode);
- int close(int fd);
- pid_t waitpid(pid_t pid, int *status, int options);
- int creat (const char *file, mode_t mode);
- int link(const char * oldpath, const char * newpath);
- int unlink(const char *pathname);
- int execve (const char *filename, char *const argv [], char *const envp[]);
- int chdir(const char *path);
- time_t time (time_t *t);
- int mknod(const char *path, mode_t mode, dev_t dev);
- int chmod(const char *path, mode_t mode);
- int lchown(const char *path, uid_t owner, gid_t group);
- off_t lseek(int fildes, off_t offset, int whence);
- pid_t getpid(void);
- int mount(const char *specialfile, const char * dir , const char * filesystemtype,
  unsigned long mountflags , const void * data);
- int umount(const char *dir);
- int setuid(uid_t uid);
- uid_t getuid(void);
- int stime(time_t *t);
- long int ptrace(enum __ptrace_request request, pid_t pid, void * addr,
  void * data);
- unsigned int alarm(unsigned int seconds);
- int pause(void);
- int utime(const char *file, const struct utimbuf *times);
- int access(const char *pathname, int mode);
- int nice(int inc);
- int kill(pid_t pid, int sig);
- int rename(const char *oldpath, const char *newpath);
- int mkdir(const char * pathname, mode_t mode);
- int rmdir(const char *pathname);
- int dup(int oldfd);
- int pipe(int filedes[2]);
- clock_t times(struct tms *buf);
- int setgid(gid_t gid);
- gid_t getgid(void);
- uid_t geteuid(void);
- gid_t getegid(void);
- int acct(const char *filename);
- int umount2(const char * special_file, int flags);
- int ioctl(int d, int request, ...);

- int __libc_fcntl(int fd, int command, ...);
- int setpgid(pid_t pid, pid_t pgid);
- mode_t umask(mode_t mask);
- int chroot(const char *path);
- int dup2(int oldfd, int newfd);
- pid_t getppid(void);
- pid_t getpgrp(void);
- pid_t setsid(void);
- int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
- int setreuid(uid_t ruid, uid_t euid);
- int setregid(gid_t rgid, gid_t egid);
- int sigsuspend(const sigset_t *mask);
- int sigpending(sigset_t *set);
- int sethostname(const char *name, size_t len);
- int setrlimit (__rlimit_resource_t resource, const struct rlimit *rlimits);
- int getrlimit (int resource, struct rlimit *rlim);
- int getrusage (int who, struct rusage *usage);
- int gettimeofday(struct timeval *tv, struct timezone *tz);
- int settimeofday(const struct timeval *tv , const struct timezone *tz);
- int getgroups(int size, gid_t list[]);
- int setgroups(size_t size, const gid_t *list);
- int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
  struct timeval *timeout);
- int symlink(const char *oldpath, const char *newpath);
- int readlink(const char *path, char *buf, size_t bufsiz);
- int uselib(const char *library);
- int swapon(const char *path, int swapflags);
- int reboot (int flag);
- __ptr_t * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset);
- int munmap(void *start, size_t length);
- int truncate(const char *path, off_t length);
- int ftruncate(int fd, off_t length);
- int fchmod(int fildes, mode_t mode);
- int fchown(int fd, uid_t owner, gid_t group);
- int getpriority(int which, int who);
- int setpriority(int which, int who, int prio);
- int statfs(const char *path, struct statfs *buf);
- int fstatfs(int fd, struct statfs *buf);
- int ioperm(unsigned long from, unsigned long num, int turn_on);
- int __socketcall(int call, unsigned long *args);
- int _syslog(int type, char *bufp, int len);
- int setitimer(int which, const struct itimerval *value, struct itimerval *ovalue);

- int getitimer(int which, struct itimerval *value);
- int stat(const char *file_name, struct stat *buf);
- int lstat(const char *file_name, struct stat *buf);
- int fstat(int filedes, struct stat *buf);
- int iopl(int level);
- int vhangup(void);
- pid_t wait4(pid_t pid, int *status, int options, struct rusage *rusage);
- int swapoff(const char *path);
- int sysinfo(struct sysinfo *info);
- int __ipc(unsigned int call, int first, int second, int third, void *ptr);
- int fsync(int fd);
- int clone(int (*fn)(void *arg), void *child_stack, int flags, void *arg);
- int setdomainname(const char *name, size_t len);
- int uname(struct utsname *buf);
- int modify_ldt(int func, void *ptr, unsigned long bytecount);
- int adjtimex(struct timex *buf);
- int mprotect(const void *addr, size_t len, int prot);
- int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
- caddr_t create_module(const char *name, size_t size);
- int init_module(void * first, void * second, void * third, void * fourth,
  void * fifth);
- int delete_module(const char *name);
- int get_kernel_syms(struct kernel_sym *table);
- long quotactl(int cmd, char *special, qid_t id, caddr_t addr);
- pid_t getpgid(pid_t pid);
- int fchdir(int fd);
- int bdflush(int func, long data);
- int setfsuid(uid_t fsuid);
- int setfsgid(gid_t gid);
- loff_t llseek(int fd, loff_t offset, int whence);
- ssize_t __getdents (int fd, char *buf, size_t nbytes);
- _newselect(int n, fd_set *readfds, fd_set *writefds,fd_set *exceptfds,
  struct timeval *timeout);
- int flock(int fd, int operation);
- int msync(const void *start, size_t length, int flags);
- int readv(int fd, const struct iovec * vector, int count);
- int writev(int fd, const struct iovec * vector, int count);
- pid_t getsid(pid_t pid);
- int fdatasync(int fd);
- int sysctl(int *name, int nlen, void *oldval, size_t *oldlenp,
  void *newval, size_t newlen);
- int sched_setparam(pid_t pid, const struct sched_param * p);

- int sched_getparam(pid_t pid, struct sched_param * p);
- int sched_setscheduler(pid_t pid, int policy,
  const struct sched_param * p);
- int sched_getscheduler(pid_t pid);
- int sched_yield(void);
- int sched_get_priority_max(int policy);
- int sched_get_priority_min(int policy);
- int sched_rr_get_interval(pid_t pid, struct timespec *tp);
- int nanosleep(const struct timespec *req, struct timespec *rem);
- void * mremap(void * old_address, size_t old_size , size_t new_size,
  unsigned long flags);
- int setresuid(uid_t ruid, uid_t euid, uid_t suid);
- int getresuid (uid_t *ruid, uid_t *euid, uid_t *suid);
- int query_module(const char *name, int which,void *buf, size_t bufsize,
  size_t *ret);
- int poll(struct pollfd *ufds, unsigned int nfds, int timeout);
- int setresgid(gid_t rgid, gid_t egid, gid_t sgid);
- int getresgid(gid_t *rgid, gid_t *egid, gid_t *sgid);
- int __syscall_rt_sigaction(int signum, const struct sigaction * act,
  struct sigaction * oldact, size_t size);
- int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
- int sigpending(sigset_t *set);
- int sigtimedwait (const sigset_t *set, siginfo_t *info,
  const struct timespec *timeout);
- int sigsuspend (const sigset_t *mask);
- ssize_t pread(int fd, void *buf, size_t count, off_t offset);
- ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset);
- int chown(const char * path, uid_t owner, gid_t group);
- char *getcwd(char *buf, int size);
- int capget(void* header, void* data);
- int capset(void* header, const void* data);
- int sigaltstack(const stack_t *ss, stack_t *oss);
- ssize_t sendfile(int out_fd, int in_fd, off_t *offset, size_t count);
- pid_t vfork(void);
- int getrlimit (__rlimit_resource_t resource, struct rlimit *rlimits);
- int truncate64 (const char * path, __off64_t length);
- int ftruncate64 (int fd, __off64_t length);
- int stat64(const char * file_name, struct stat64 * buf);
- int lstat64(const char * file_name, struct stat64 * buf);
- int fstat64(int fd, struct stat64 * buf);
- int pivot_root(const char *new_root, const char *put_old);
- ssize_t __getdents64 (int fd, char *buf, size_t nbytes);

- int fcntl64(int fd, int command, ...);

12.12 pthread library

- int pthread_create(pthread_t* thread,pthread_attr_t* attr,
  void* (*start_routine) (void *), void * arg);
- pthread_t pthread_self(void);
- int pthread_equal(pthread_t thread1, pthread_t thread2);
- void pthread_exit(void *retval);
- int pthread_join(pthread_t th, void **thread_return);
- int pthread_detach(pthread_t th);
- int pthread_attr_init(pthread_attr_t *attr);
- int pthread_attr_destroy(pthread_attr_t *attr);
- int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
- int pthread_attr_getdetachstate(const pthread_attr_t *attr, int *detachstate);
- int pthread_attr_setschedparam(pthread_attr_t *attr,
  const struct sched_param *param);
- int pthread_attr_getschedparam(const pthread_attr_t *attr,
  struct sched_param *param);
- int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
- int pthread_attr_getschedpolicy(const pthread_attr_t *attr, int *policy);
- int pthread_attr_setinheritsched(pthread_attr_t *attr, int inherit);
- int pthread_attr_getinheritsched(const pthread_attr_t *attr, int *inherit);
- int pthread_attr_setscope(pthread_attr_t *attr, int scope);
- int pthread_attr_getscope(const pthread_attr_t *attr, int *scope);
- int pthread_attr_setguardsize (pthread_attr_t *attr, size_t guardsize);
- int pthread_attr_getguardsize (const pthread_attr_t *attr, size_t *guardsize);
- int pthread_attr_setstackaddr (pthread_attr_t *attr, void *stackaddr);
- int pthread_attr_getstackaddr (const pthread_attr_t *attr, void **stackaddr);
- int pthread_attr_setstacksize (pthread_attr_t *attr, size_t stacksize);
- int pthread_attr_getstacksize (const pthread_attr_t *attr, size_t *stacksize);
- int pthread_setschedparam (pthread_t target_thread, int policy,
  const struct sched_param *param);
- int pthread_getschedparam (pthread_t target_thread, int *policy,
  struct sched_param *param);
- int pthread_mutex_init(pthread_mutex_t *mutex,
  const pthread_mutexattr_t *mutexattr);
- int pthread_mutex_lock(pthread_mutex_t *mutex);
- int pthread_mutex_trylock(pthread_mutex_t *mutex);
- int pthread_mutex_unlock(pthread_mutex_t *mutex);
- int pthread_mutex_destroy(pthread_mutex_t *mutex);
- int pthread_mutexattr_init(pthread_mutexattr_t *attr);
- int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);

- int pthread_mutexattr_settype (pthread_mutexattr_t *attr, int kind);
- int pthread_mutexattr_gettype (const pthread_mutexattr_t *attr, int *kind);
- int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *cond_attr);
- int pthread_cond_signal(pthread_cond_t *cond);
- int pthread_cond_broadcast(pthread_cond_t *cond);
- int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
- int pthread_cond_timedwait(pthread_cond_t *cond, pthread_mutex_t *mutex,
  const struct timespec *abstime);
- int pthread_cond_destroy(pthread_cond_t *cond);
- int pthread_condattr_init(pthread_condattr_t *attr);
- int pthread_condattr_destroy(pthread_condattr_t *attr);
- int pthread_rwlock_init (pthread_rwlock_t *rwlock,
  const pthread_rwlockattr_t *attr);
- int pthread_rwlock_destroy (pthread_rwlock_t *rwlock);
- int pthread_rwlock_rdlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_tryrdlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_wrlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_trywrlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_unlock (pthread_rwlock_t *rwlock);
- int pthread_rwlockattr_init (pthread_rwlockattr_t *attr);
- int pthread_rwlockattr_destroy (pthread_rwlockattr_t *attr);
- int pthread_key_create(pthread_key_t *key, void (*destr_function)(void *));
- int pthread_key_delete(pthread_key_t key);
- int pthread_setspecific(pthread_key_t key, const void *pointer);
- void * pthread_getspecific(pthread_key_t key);
- int pthread_once(pthread_once_t *once_control, void (*init_routine)(void));
- int pthread_cancel(pthread_t thread);
- int pthread_setcancelstate(int state, int *oldstate);
- int pthread_setcanceltype(int type, int *oldtype);
- void pthread_testcancel(void);
- void pthread_cleanup_push(void (*routine) (void *), void *arg);
- void pthread_cleanup_pop(int execute);
- void pthread_cleanup_push_defer_np(void (*routine) (void *), void *arg);
- void pthread_cleanup_pop_restore_np(int execute);
- void pthread_kill_other_threads_np(void);
- sem_init() /
- sem_wait()
- sem_post()