

RTLlinux 설치 방법

Last Modified at 2005.12.14 by deathymn
- Mechatronics Lab, at Mechanical Eng.Dep. in Yonsei Univ

- * 이 작업은 반드시 root 사용자로 로그인 해서 수행한다. *
- * 이 문서는 Ubuntu linux 5.10 버전을 기준으로 설명된다 *

* 준비작업: 커널을 컴파일하기 위해서는, gcc 와 ncurses library 등이 미리 깔려있어야 한다.

처음 ubuntu 를 설치하면 개발환경이 설치되어있지 않기 때문에, 다음과 같은 명령으로 기본적인 개발환경을 설치한다.

```
apt-get install build-essential
```

시스템-> 관리-> 시냅틱 패키지 관리자를 실행시켜, Search 로 “ncurses”를 검색해 본다. 결과 창에서 봤을 때, “libncurses5”, “libncurses5-dev”, “libncursesw5”, “ncurses-base”, “ncurses-bin”이 설치되어 있어야 한다.

또한 시냅틱 패키지 관리자에서, 이번에는 “gcc”로 검색해 본다. 이 경우, “gcc”, “gcc-4.0”, “gcc-4.0-base” 정도가 설치되어있음을 확인할 수 있을것이다. 여기에 추가적으로 검색창에서 “gcc-2.95”란 패키지를 설치해 준다. 테스트 해 본 결과, gcc-4.0 을 이용해서 컴파일 할 경우, 에러가 나는것을 확인.... 커널 컴파일을 위해서는 gcc-2.95 패키지가 절대적으로 필요하다..

터미널을 하나 띄워서, “ls -al /usr/bin/gcc” 를 확인해 보자. 결과 창에 보면, gcc 라는 화일이 gcc-4.0 을 가르키는 링크 화일임을 알 수 있다. 앞의 과정에서 gcc-2.95 패키지가 정상적으로 설치되었으면, /usr/bin/ 폴더 안에, “gcc-2.95”화일이 존재할 것이다. (ls -al /usr/bin/gcc-2.95)
결국, 커널 컴파일을 위해서 gcc-2.95 를 이용하기 위해서는, gcc 의 링크만 바꾸어주면 되는 것이다. 다음과 같은 명령으로 일단 gcc 를 지운다.

```
rm -rf /usr/bin/gcc
```

그리고, gcc-2.95 화일을 가르키는 링크를 다음과 같이 건다.

```
ln -s /usr/bin/gcc-2.95 /usr/bin/gcc
```

링크가 제대로 걸렸는 지 확인해 본다.

```
ls -al /usr/bin/gcc
```

result)

```
/usr/bin/gcc->/usr/bin/gcc-2.95
```

또한, 나중에(26 과정) initrd.img 를 만들기 위해서, 시냅틱 패키지 관리자에서, “initrd-tools”를 검색해서 설치한다.

리눅스 커널 및, RTLlinux 구하기.

1. RTLlinux 를 설치하기 위해서는, 일반적인 리눅스 커널과 RTLlinux 모듈이 들어있는 화일이 필요하다.

일반적인 리눅스 커널 :	linux-2.4.29.tgz or linux-2.6.9.tgz
RTLlinux 모듈 화일 :	rtlinux-3.1.tgz or rtlinux-3.1-2.6_kernel.tgz

2. 두 화일을 다운 받는 경로는 다음과 같다.

일반적인 리눅스 커널 :	ftp.kernel.org
RTLinux 모듈 화일 :	www.rtlinuxfree.com (회원가입 필요)

참고) www.rtlinuxfree.com 사이트에 들어가면, 다운 받을 수 있는 곳에 rtlinux 에 대한 문서화일이 있다.
ex) rtdoc-3.2-pre1.tar.bz2 이 문서화일에는 설치방법과 튜토리얼이 들어있다.

3. 설치의 모든 작업은 /usr/src 폴더에서 진행하기 때문에 이쪽으로 다운받은 화일들을 카피한다.

```
ex) cp linux-2.4.29.tgz /usr/src/linux-2.4.29.tgz
    cp rtlinux-3.1.tgz /usr/src/rtlinux-3.1.tgz
```

카피가 끝난 후, 이 폴더로 이동한다.

```
ex) cd /usr/src
```

리눅스 커널 컴파일.

(참고) www.rtlinuxfree.com 사이트에 보면, prepatched 커널이라는게 있다. (ex) prepatched_linux_kernel-2.4.29-rtl.tgz 즉, 이 화일을 다운 받으면, RT 커널로 변환된 리눅스 커널을 얻을 수 있다. 즉, 10 과정이 필요없다는 말이다. 하지만 이 화일안에 모든것이 포함되어있는게 아니라, 일반적인 리눅스 커널+ RT 패치만 들어있기 때문에, RT 모듈이 들어있는 화일도 반드시 받아주어야 한다.

4. 먼저 리눅스 커널을 압축을 푼다.

```
ex) tar xvzf linux-2.4.29.tgz 만약 prepatch 커널일 경우, tar xvzf prepatched_linux_kernel-2.4.29-rtl
```

압축을 푼 후에, /usr/src/ 폴더밑에 새로운 폴더가 하나 생겼음을 볼 수 있다.

```
ex) /usr/src/linux-2.4.29/ , or /usr/src/linux-2.6.9/
```

5. 작업을 편하게 하기위해 이 폴더를 'linux'라고 링크를 걸어둔다.

```
ex) ln -s /usr/src/linux-2.4.29 linux or, ln -s /usr/src/linux-2.6.9 linux
```

링크가 걸렸는 지 확인

ls -al	result) lwxrwxrwx linux -> linux-2.4.29 or lwxrwxrwx linux -> linux-2.6.9
--------	--

* 커널 컴파일을 진행하기 전에, 이 커널을 RT 커널로 만들어 주는 과정이 필요하다.

이는 앞에서 언급했듯이, RTLinux 모듈화일에 들어있는 패치화일로 패치해주는 과정이 필요하다는 의미이다. (Prepatch kernel 을 받은 경우에는 이과정이 필요없다.. 패치가 되어있는걸 받았기 때문...

즉, 7 -> 11 까지의 과정이 필요없다.)

6. 커널을 패치하기 위해, RTLinux 모듈화일을 압축을 푼다.

```
ex) tar xvzf rtlinux-3.1.tgz
```

압축을 푼 후에, /usr/src/ 폴더밑에 새로운 폴더가 하나 생겼음을 볼 수 있다.

```
ex) /usr/src/rtlinux-3.1/
```

나중에 작업이 끝나고 나면, RTLinux 에 관련된 화일들이, /usr/rtllinux-3.1 이라는 폴더로 자동 생성되어 저장된다. 즉, 이 두 폴더가 이름이 똑같이 때문에 (물론 경로는 틀리지만) 혼동될 수 있으므로, /usr/src/rtllinux-3.1 폴더를 rtlinux-3.1_source 이런식으로 이름을 바꾸어주자.

결국 작업을 할 폴더들의 구조는 다음과 같다.

```

(kernel 2.4.29 일 경우)
/-----/usr-----/src-----/linux-2.4.29
      |               |-----/rtlinux-3.1_source
      |
      |-----/rtlinux-3.1 (작업이 끝난 후, 자동으로 생성되는 폴더)

(kernel 2.6.9 일 경우)
/-----/usr-----/src-----/linux-2.6.9
      |               |-----/rtlinux-3.1_source
      |
      |-----/rtlinux-3.1 (작업이 끝난 후, 자동으로 생성되는 폴더)

```

7. 이 폴더로 이동 후 (cd /rtlinux-3.1_source), 폴더 안을 보면 (ls -al) 'patches' 라는 폴더가 보일 것이다. 이 폴더로 이동한다. (cd patches)

8. 그럼 현재 폴더는 /usr/src/rtlinux-3.1_source/patches 이어야 한다. 이 폴더 안에 내용을 보면(ls -al),

```

kernel_patch-2.4.20-rtl
kernel_patch-2.4.29-rtl
kernel_patch-2.6.9rtlfree

```

등의 화일이 보일 것이다. 이 중 다운받은 리눅스 커널, 예를들어 linux-2.4.29 에 맞는 패치화일은 kernel_patch_2.4.29-rtl 이다.

9. 이제 패치를 진행하기 위해 linux 폴더로 이동한다.

```
ex) cd /usr/src/linux
```

10. 다음의 명령으로 커널 패치를 진행한다.

```

patch -p1 < /usr/src/rtlinux-3.1_source/patches/kernel_patch-2.4.29-rtl 또는
patch -p1 < /usr/src/rtlinux-3.1_source/patches/kernel_patch-2.6.9rtlfree

```

11. 여기까지 해서, 일반적인 리눅스 커널의 소스를 RT 커널의 소스로 변환하였다.

12. 이제부터 커널 컴파일을 시작한다. 커널 컴파일을 하기위해 커널소스가있는 폴더로 이동한다.

```
ex) cd /usr/src/linux
```

13. 커널 컴파일을 크게 세가지 과정으로 진행된다.

- 1) 자신의 컴퓨터에 달려있는 장치들에 적합한 커널 옵션 설정
- 2) 커널 컴파일
- 3) 여러 장치 디바이스 드라이버들의 컴파일

14. 커널 컴파일을 하기전에 먼저, 기존 환경설정값을 없애기 위해,

```
make mrproper
```

15. 또한 한번이상 커널 컴파일을 수행한 경우, 여러 쓰레기 화일들이 남아있을 수 있으므로,

```
make clean
```

16. 다음으로, 커널의 옵션을 설정하기 위해,

```
make menuconfig 또는, make xconfig
```

주의) 가장 어려운 부분이 이 부분이 아닐까 싶다. 가장 쉬운 방법은 커널 옵션 세팅이 되어있는 기존 배포판의 config 파일을 load 해서, 필요없는 부분을 지워가는 것이라 할 수 있다. 커널 옵션 중에 다음 사항은 반드시 체크하자. (단, 2.6.9 버전의 커널 옵션에 대해 설명)

Power management setup -> APM(Advanced Power Management) BIOS Support -> APM(Advanced Power Management) BIOS Support 는 n 으로 설정

Power management setup -> CPU Frequency Scaling -> CPU Frequency Scaling 는 n 으로 설정

Processor Type and Features-> Processor Family-> 에서 자신의 CPU 에 맞는 Processor 설정

Processor Type and Features-> Local APIC Support on uniprocessors 는 n 으로 설정 (선택안함)

현재, Ubuntu 5.10 (Breezy)에 대한, 커널 옵션 설정파일(UbuntuSetting.config)을 같이 첨부한다. 이를 로드하기 위해서는, 일단 이 파일을 /usr/src/linux 에 복사해 놓고, make menuconfig 설정 창에서,

```
Load an Alternative Configuration File
```

으로 불러오면 된다.

17. 커널 옵션을 다 설정한 후에는,

```
make dep
```

명령을 수행해서, 옵션들 사이의 의존관계가 제대로 되어있는지 체크한다.

(2.6.9 커널인 경우, 이 과정이 필요없음)

18. 커널을 컴파일 하여, 바이너리 파일로 만들기 위해,

```
make bzImage,
```

2.6.9 커널인 경우에는, make 만 입력

19. 약간의 시간이 지난 후, 커널 컴파일이 정상적으로 수행됐다면, bzImage 란 바이너리 파일이 생성된다.

이는, /usr/src/linux/arch/i386/boot 폴더 밑에 저장된다.

20. 커널 이미지를 만들었기 때문에, 다음으로 커널에 연결되는 장치 디바이스 드라이버들을 만든다.

```
make modules
```

21. 모듈생성 과정이 끝났으면, 만들어진 모듈들을 적당한 폴더에 등록하기 위해,

```
make modules_install
```

이 명령을 수행하면, /lib/modules/ 폴더 밑에 지금 컴파일하는 커널 버전에 해당하는 폴더가 생성되고, 그 안에 모듈 파일들이 적재. (ex) /lib/modules/2.6.9rtlfree

주의) 만약 기존에 같은 커널 버전에 대해서 커널 컴파일을 했다면, /lib/modules/ 폴더 밑에 해당 커널버전의 폴더가 존재한다. 이 경우, 먼저 이 폴더를 지운 후에 (rm -rf /lib/modules/2.6.9rtlfree) 21 번 과정을 수행한다.

22. 여기까지 해서, 커널 컴파일 하는 과정은 모두 끝났다.

23. 이제 해야할 작업은 컴파일 된 커널로 부팅하기 위해 Boot Loader (ex, Lilo or Grub) 에 등록시켜주는 과정이다.

24. 먼저 컴파일된 커널을 부팅 이미지가 모여있는 폴더로 카피한다.

```
ex) cp /usr/src/linux/arch/i386/boot/bzImage /boot/RTKernel-2.4.29
```

여기서 복사되는 파일 이름 (ex, RTKernel-2.4.29)는 임의로 정해도 상관없다.

25. 또한 System.map 파일도 카피하자.

```
ex) cp /usr/src/linux/System.map /boot/System.map-2.4.29
```

(참고) 2.6 버전 대의 커널은 위 두 과정이 make install 을 실행함으로써 끝난다. (/usr/src/linux 폴더 안에서)

26. 다음으로, 부팅 과정에서 사용될 initrd.img 화일을 다음과 같은 명령을 수행해서 생성하자. (이 명령은 임의의 폴더에서 수행해도 무관하다.)

```
ex) mkinitrd -o /boot/initrd-2.6.9rtlfree.img 2.6.9rtlfree
```

위 명령에서, 두 번째 파라미터는 현재 컴파일 하는 커널의 버전 명이다. (이에 대한 정확한 이름은, /lib/modules/아래의 폴더 명으로 확인 가능하다.)

27. Grub 이라는 Boot Loader 에 새로 컴파일한 커널로 부팅되도록, 등록시키기 위해 /boot/grub/menu.lst 화일을 수정한다.

```
ex) vi /boot/grub/menu.lst
```

28. 에디터 (ex, vi)를 써서, menu.lst 화일에 다음과 같은 라인들을 추가한다.

```
title      RTLinux-2.6.9
root       (hd0,1)
kernel     /boot/vmlinuz-2.6.9rtlfree
initrd     /boot/boot/initrd-2.6.9rtlfree.img
```

여기서, title 은 부팅 시에 Boot Loader 화면에서 보이는 글자를 쓰는 것이므로, 임의의 글자를 써도 된다. kernel 은 24 과정에서 카피한 커널의 이름을 위와같이 써준다. 또한, initrd 부분은, 26 과정에서 생성된 이미지 화일이름을 써준다. 여기서, (hd0,6)은 리눅스가 깔려있는 파티션을 의미하는데, 이부분은 menu.lst 화일 내에 다른 부분을 참고해서 똑같이 써주면 된다.

29. 저장하고 빠져나오면 커널 컴파일 하는 모든 과정은 끝난다.

RT 모듈 컴파일.

30. 컴퓨터를 재부팅하고, Boot Loader 에 28 과정에서 입력한 타이틀이 뜨는지 확인하고, 그걸 선택해 부팅한다.

31. 부팅과정에서 Kernel panic 하고 멈추는 경우가 발생할 수 있다. 이 경우, 기존 커널로 다시 부팅해서 12 과정 부터 다시 수행한다.

32. 정상적으로 부팅에 성공하면, 커널 컴파일은 성공적으로 이루어진것이다.

이제 RT 모듈을 컴파일 하기위해, 터미널을 하나 띄우고, RTLinux 폴더로 이동한다.

```
ex) cd /usr/src/rtlinux-3.1_source
```

33. 이 폴더 아래에, 리눅스 커널을 가르키는 링크를 하나 만든다.

```
ex) ln -s /usr/src/linux-2.6.9 linux
```

그러면, /usr/src/rtlinux-3.1_source 폴더 안에, "linux"라는 링크 화일을 볼 수 있다.

34. RT 모듈들에 대한 옵션을 설정하기 위해 (ex, RTFifo 의 개수 등..) make menuconfig 또는 make xconfig 을 수행한다.

35. 이를 수행하면, 라이선스 관련 텍스트 화일이 나오는데, q 키를 치고 빠져나오면 라이선스에 동의하는지 물어본다. 여기서 y 키를 누른다. 옵션은 그냥 기본적인 설정을 사용하고 저장을 하고 나온다.

36. 모듈을 컴파일 하기 전에,

```
make dep
```

명령으로 의존 관계를 설정한다.

(주의) 이 명령을 수행하고 나면, /usr/src/rtlinux-3.1_source/scripts 라는 폴더 안에, “rtlinux” 와 “rtl-config” 이라는 링크화일이 만들어 진다. 그러나, Makefile 의 버그인지는 몰라도, 이 링크가 깨져있다. 이를 수정하기 위해 scripts 란 폴더로 이동 후 (cd ./scripts)

```
rm -rf rtlinux
rm -rf rtl-config
```

으로 두 링크 화일을 지워준다.

다음으로 현재 RTLinux 를 사용하는 커널 버전에 따라서 다음과 같이 링크를 걸어둔다.

```
ln -s rtlinux-2.6 rtlinux
ln -s rtl-config-2.6 rtl-config
```

ls -al 명령으로 두 화일의 링크가 제대로 걸렸는 지 확인 한 후, 상위 폴더로 이동한다.

```
cd .. 또는
cd /usr/src/rtlinux-3.1_source
```

37. 다음으로

```
make
```

를 입력하여, 모듈들을 컴파일 하여 생성한다.

38. 컴파일 하는 도중, warning 이 많이 뜨는데, 컴파일만 정상적으로 끝나면 이는 상관없다. 다음으로 /dev 폴더 안에, Rtllinux 와 관련된 장치 화일들을 만들기 위해 다음 명령을 수행한다.

```
make devices
```

39. 적절한 위치에 화일들을 옮기기 위해,

```
make install
```

명령을 수행한다. 이 명령의 결과로, 6 번 과정에서 언급했던, /usr/rtlinux-3.1 이라는 폴더가 자동 생성되며, 또한 /usr 폴더 아래에 “rtlinux”라는 링크 화일이 생성된다.

40. 모든 과정은 끝났다.

마지막으로 제대로 모든 것들이 설치되었는 지를 확인하기 위해서, 임의의 폴더에서

```
rtlinux start
```

를 실행해 보자. 그러면 결과는 다음과 같다.

```
Scheme: (-) not loaded, (+) loaded
(+) rtl
(+) rtl_fifo
(+) rtl_posixio
(+) rtl_sched
(+) rtl_time
```

만약, 이 중 하나라도 (-) 가 뜨면, 이는 어딘가 잘못되었다는 것이다. 보통 두가지 문제를 지적한다. 첫번째는, 16 과정에서 제시된 옵션 설정이 제대로 안 되어있을 경우이고, 두번째는 38 과정을 하지 않아, /dev/ 밑에 디바이스 화일들이 제대로 생성되지 않은 경우이다. 이를 체크해 본다.