RTLinux 설치 방법 (Ubuntu Breezy Version)

Last update: 2006.03.17 by deathymn
- Mechatrobotics Lab, Mechanical Eng.Dep. Yonsei Univ

* 이 작업은 반드시 root 사용자로 로그인 해서 수행한다. *
* 이 문서는 Ubuntu linux 5.10 버전을 기준으로 설명된다 *
* 이 문서를 작성할 당시, 사용한 커널은, 2.6.9 를 사용했다. 사실, 2.6 버전대로 오면서, 커널 내부적으로 모듈을 다루는 방식에 많은 변화가 발생했다. 그러므로, 왠만하면 2.6.9 버전의 커널을 사용하자.
* Ubuntu 에서 패키지 관리자는, '시냅틱 패키지 관리자'를 사용해도 되며, 만약 Kubuntu 환경일 경우에는 ' Adept 패키지 관리자'를 사용해도 상관없다.

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

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

```
apt-get install build-essential
apt-get install kernel-package
```

시스템-> 관리-> 시냅틱 패키지 관리자를 실행시켜, Search 로 "ncurse"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 확일할 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

리눅스 커널 및, RTLinux 구하기.

1. RTLinux를 설치하기 위해서는, 일반적인 리눅스 커널과 RTLinux 모듈이 들어있는 화일이 필요하다.

일반적인 리눅스 커널 :	linux-2.4.29.tgz or linux-2.6.9.tgz
RTLinux 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 rtlinux에 대한 문서화일이 있다.
ex) rtldoc-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 kernel을 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/rtlinux-3.1 이라는 폴더로 자동 생성되어 저장된다. 즉, 이 두 폴더가 이름이 똑같이 때문에 (물론 경로는 틀리지만) 혼동될 수 있으므로, /usr/src/rtlinux-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 화일을 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 설정화일(UbuntuSetting.config)을 같이 첨부한다. 이를 로드하기위해서는, 일단 이 화일을 /usr/src/linux 에 복사해 놓고, make menuconfig 설정 창에서,

```
Load an Alternative Configuration File
```

으로 불러오면 된다. 이렇게 불러온 후, 위에서 언급한 processor 부분은 반드시 해당 컴퓨터의 processor 에 맞게 변경하자.

주의) 혹, 이 설정이 해당 컴퓨터 환경과 안 맞을 수도 있다. 이 경우, 현재 부팅되어 있는 커널 설정의 환경을 불러와서 필요한 것만 바꿀 수 있다. 현재 부팅되어있는 커널이 2.6.10 이라 하자. 그러면 /boot 폴더에 'config-2.6.10'이라는 화일을 볼 수 있다. 그러면 일단 현재 커널 컴파일 하는 폴더로 이동 후(cd /usr/src/linux), 위 화일을 다음과 같이 카피한다. (cp /boot/config-2.6.10 ./.config) 그런 후에, 다음과 같은 명령을 수행한다. (make oldconfig). 그러면 config-2.6.10 에 설정되어 있는 환경설정 파라미터가, 현재 컴파일하려는 커널의 환경설정 파라미터에 있으면 자동으로 설정하고, 그렇지 않은 것은 사용자에게 물어보게 된다. 이 경우, 왠만한 건, no, 즉, n 을 치고 넘어가고록 하자. 그러면 make oldconfig 이 끝난 후에, 다시 make menuconfig 등을 실행해서 다시 한번 검토해 볼 수 있다.

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 화일로 만들기 위해,

```
make-kpkg -initrd -stem linux --append-to-version=.01 kernel_image kernel_headers
```

: 여기서 --append-to-version 다음의 .01 은 임의값을 설정해도 된다. 즉, 말그대로 자신만의 버전명을 임의로 붙이는 것이다. 위의 명령으로 컴파일하게 되면 화일이름이 linux-headers-2.6.9rtlfree.01_10.00.Custom_i386.deb 와 같은 방식으로 생성되게 된다. 커널을 컴파일 한 후, 20 번 과정으로 설치를 하게 되면, 커널과 관련된 모듈들이 '/lib/modules/커널 버전' 폴더에 설치되게 된다. 즉, 위의 옵션 없이 하나의 커널 소스를 여러번 컴파일하게 되면, 모듈들이 '/lib/modules/커널 버전' 폴더에 계속 overwrite 되게되는 것이다. 그러므로, 위의 옵션을 주어 같은 커널이라도 버젼을 달리해서 컴파일 하게되면, 서로 다른 폴더에 모듈

들이 설치되므로 모듈들이 중복되지 않고, 커널을 관리하는 것도 쉬워진다.

19. 약간의 시간이 지난 후, 커널 컴파일이 정상적으로 수행됐다면, /usr/src 폴더를 보면, 다음과 같이 두 화일이 생성된 것을 볼 수 있다.
: linux-headers-2.6.9rtlfree.01_10.00.Custom_i386.deb
linux-image-2.6.9rtlfree.01_10.00.Custom_i386.deb

20. 이를 시스템에 install 하는 것은 다음과 같은 명령을 사용한다. (현재 폴더: /usr/src)

```
dpkg -i linux-headers-2.6.9rtlfree.01_10.00.Custom_i386.deb
dpkg -i linux-image-2.6.9rtlfree.01_10.00.Custom_i386.deb
```

: 이렇게 설치된 커널 이미지는 시냅틱 패키지 관리자를 통해서 제거 가능하다.

21. 지금 컴파일 한 커널이 부트로더에 등록되어 있는 지 확인해 보자.

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

22. 여기까지 해서, 커널 컴파일 하는 과정은 모두 끝났다.

23. 만약, 커널 컴파일이 잘못됐거나, 옵션 설정을 다시 해서 다시 컴파일 해 줘야 하는 경우, 일단 다른 커널로 부팅한다.

24. 부팅 후에, '시냅틱 패키지 관리자'를 실행해서, 지우려는 커널을 찾는다. 즉, 'search'버튼을 눌러서, 20 과정에서 설치한 이름을 이용해서 찾는다. 찾은 후에, 삭제를 하면 된다. (만약 지우지 않고, 같은 커널을 다른 옵션을 주어서 추가로 설치할 경우는, 이 과정은 생략)

25. 그리고나서, /usr/src/linux 폴더로 이동.

```
cd /usr/src/linux
```

26. 여기서, 기존 컴파일 했던, 중간 생성화일들과 결과물들을 삭제하기 위해 다음 명령을 실행한다.

```
make-kpkg clean
```

27. 이 명령을 실행한 후에, 16 과정부터 다시 실행한다.

RT 모듈 컴파일.

28. 컴퓨터를 재부팅하고, Boot Loader 에 21 과정에서 확인한 타이틀이 뜨는지 확인하고, 그걸 선택해 부팅한다.

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

30. 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
```

31. 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.

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

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

34. 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
```

35. 다음으로

```
make
```

를 입력하여, 모듈들을 컴파일 하여 생성한다.

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

```
make devices
```

37. 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.

38. 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 31 과정을 하지 않아, /dev/ 밑에 디바이스 화일들이 제대로 생성되지 않은 경우이다. 이를 체크해 본다.

39. kernel 2.6.9 인 경우, rtlinux 모듈을 위해 사용하는 "rtlinux stop" 명령에 약간의 오류가 있다. rtlinux stop 은 내부적으로 rmmod 명령을 사용하여 모듈을 내리는데, rtlinux stop 명령에서는 "rmmod -r"을 수행한다. 그러나 rmmod 의 버전이 바뀌면서, -r 이란 옵션이 없어진 것 같다. 그래서 모듈을 내리기 위해 rtlinux stop 명령을 치면, 오류가 출력되며 모듈이 안 내려간다. 이는 다음과 같이 수정할 수 있다. 일단 /etc/rc.d/init.d/rtlinux 화일을 편집기로 불러들인다. (ex: vi /etc/rc.d/init.d/rtlinux) 그러면, 66 번 째 줄과, 68 번 째 줄에서, "RMMOD="/sbin/rmmod -r"과 "RMMOD="'which rmmod' -r"이 있을 것이다. 여기서 -r 옵션을 모두 지우고 저장을 한다.

40. 다음, "rtlinux stop"명령을 내린다.
그러면, 실행 결과는 다음과 같을 것이다.

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

여기서도, 아직 오류가 있다. 즉, 모든 모듈이 내려 가야 하는데, rtl 은 안내려간다. 이는 내부적으로 모듈을 다른 모듈들을 내린 다음에, rtl 을 마지막으로 내려야 하는데, 그 순서가 잘못 되어있는 것 같다.
"rtlinux stop"명령을 한번 더 치면 rtl 모듈마저도 내려가지만, 깔끔하지가 않다.
(참고:) 이를 해결하기 위한 한가지 편법으로, 다시 편집기로 /etc/rc.d/init.d/rtlinux 화일을 연다. 그리고 143 번째 줄 정도에 다음과 같은 코드를 삽입한다.

```
for modules in ${RTL_MODULES_LIST}
do
   MODINS='${LSMOD} | ${GREP} ^$modules'
   if [ " $MODINS" ]; then
      ${RMMOD} $modules
   fi
done

${RMMOD} 'rtl'          <- 삽입한 코드
```

즉, 정말 편법으로 rmmod 명령을 한번 더 실행시켜서, "rmmod rtl"을 실행한거와 똑같다..

이로써 RTLinux 를 사용하기 위한 준비는 다아 끝났다.
제대로 되는 지 확인해 보기 위해서,
/usr/src/rtlinux-3.1_source/examples/frank 폴더에 있는 내용을 실행 해 보자.
폴더 안을 보면, 다음과 같은 두 화일을 볼 수 있다.
"frank_app" & "frank_module.ko"
실행 순서는 먼저 모듈을 올린다. 즉, "rtlinux start frank_module.ko" : .ko 를 반드시 붙인다.

다음으로 frank_app 을 실행시킨다. 즉, "./frank_app"
그러면 화면에 "frank, zappa"가 교대로 뜨는걸 볼 수 있다. 응용 프로그램이 종료하면,
"rtlinux stop frank_module"명령을 내려서 모듈을 내려준다...

41. 만약 리눅스 커널을 재 컴파일 한 경우에는 RTLinux 도 다시 컴파일 해 주어야 한다. 일단 30 번 과정으로 해당 폴더로 이동한다.

42. 다음 명령을 실행하여, 재 컴파일을 위한 준비를 한다.

```
make clean
```

43. 다음으로 34 과정을 수행한다. 하지만 이번에는 rtlinux 와 rtl-config 링크화일이 안 깨져 있을 것이다.

44. 그리고 35 - 38 과정을 수행한다. 이번에도 역시 'rtlinux'관련 스크립트에 오류가 있으므로, 38 과정처럼 수정해 준다. 마지막으로, 40 번 과정에서 언급된 예제를 실행해 보고 제대로 되는 지 확인하자..