Linux PPP HOWTO

Robert Hart, hartr@interweft.com.au v3.0, 31 March 1997

번역:김 창한 1차 수정번역 및 sgml 편집:김 병찬 redhands@kldp.linux-kr.org

이 문서는 어떻게 리눅스 PC 를 PPP 서버에 연결 하는지, 어떻게 두개의 랜을 통하여 PPP 를 사용하는지 그리고 리눅스 박스를 PPP 서버로써 어떻게 활용할것인지를 다룬다. 또한 PPP 연결에 대한 잘못된 것을 고치도록 도와줄 것이다.

목 차

저작권

구할수 있는곳

comp.os.linux.answes 뉴스그룹에서 최신 버전을 구할수 있다. html 포맷을 구할려면 아래를 참고하라.

- *Linux Howto Index* <http://sunsite.unc.edu/mdw/linux.html##howto>
- *PPP-HOWTO* <http://www.interweft.com.au/other/ppp-howto/ppp-howto.html>

다른 포멧(SGML, ASCII, postscript, DVI)의 문서들은 *Ascii, Html 을 제외한 문서들* <ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/other-formats> 에서 구할수 있다.

sunsite.unc.edu 는 아주 많은 부하가 걸리고 있다. 웬만하면 가까운 곳의 미러 사이트를 이용하라.

감사의 말

많은 사람들이 이 문서를 준비하는데 도움을 주었다. Al Longyear 씨께 PPP 자체에 대하여 알려주신데 대해 특별히 감사한다. (그렇지만 이 문서에 잘못이 있다면 내 잘못이지 그의 잘못은 아니다.) Greg Hankins 씨(리눅스 하우투 체계의 운영자)와 Debi Tackett 씨(MaximumAcces s.c om의)에게는 서술 풍과 내용 순서, 논리, 표현의 명확성 등에 대해 도움이 된 많은 제안을 해 주신데 대해 특별히 감사한다.

마지막으로, 전자우편을 통해 주석을 요청해 준 분들께 감사드린다. 모든 하우투 저자들과 마찬가지로, 도와주는데서 얻는 만족이 받는 보수이고, 그걸로 충분하다. 이 하우투를 씀으로써 나-와 다른 모든 사용자들 - 이 우리가 선택한 운영체제를 만들고 운영해나가는 사람들에게 진 빚을 조금이나마 갚는 중이다.

제 1 절 소개

지점 대 지점 프로토콜(PPP:the Point to Point Protocol)은 직렬 연결 - 직접 직렬 연결(널 모뎀 케이블을 쓰는)이 될 것이다-, 텔넷 구축 연결 혹은 모뎀과 전화선을 쓰는 연결(물론 ISDN같은 디지털 회선을 쓰는 경우도) 상에서 인터넷 프로토콜(IP: the Internet Protocol)과 다른 네트워크 프로토콜을 만들고 작동시키는 기제이다.

PPP를 쓰면 리눅스 PC를 PPP 서버에 연결시키고 그 서버가 연결된 네트워크에 직접 연결된 것처럼 그 네트워크의 소스를 쓸 수 있다.

마찬가지로 리눅스 PC를 PPP서버로 설정해서 다른 컴퓨터에서 내 컴퓨터에 전화해 들어와 내 로컬 PC와 네트워크의 소스를 쓸 수 있게 할 수 있다.

PPP가 대등 체계인 만큼, 두 개의 리눅스 PC에서 PPP를 써서 두개의 네트워크를 연결하고 (혹은 로컬 네트워크를 인터넷에 연결) 광역네트워크(WAN)을 만들 수도 있다.

PPP와 이더넷 연결의 커다란 차이 중 하나는 물론 속도이다 - 표준 이더넷 연결은 이론 출력이 최대 10Mbs(초당 만비트-Mega단위)인데비해 아날로그 모뎀은 속도가 56kbs(초당 천비트-kilo단위)정도까지만 올라간다.

또한 PPP 연결 방식을 택할 경우, 몇가지 프로그램과 서비스의 사용에 약간의 제약이 있을 수 있다.

1.1 클라이언트 측과 서버

PPP는 엄밀한 **대등**프로토콜이다; (기술적으로는) 전화하는 기계나 전화받는 기계나 차이가 없다. 하지만 명확성의 면에서 보면 **클라이언트 측**과 **서버**의 용어로 생각하는 것이 유용하다.

PPP연결을 구축하기 위해 어떤 장소로 전화를 걸고 있을 때, 전화거는 쪽이 **클라이언트 측**이고 전화받는 쪽이 **서버**이다.

리눅스 한 통을 PPP연결 상에서 전화받고 다룰 수 있도록 설정하는 경우, PPP **서버**를 설정하는 것이다.

어떤 리눅스 PC도 PPP서버와 클라이언트 측이 될 수 있다. -한개 이상의 직렬 포트(그리고 필요하다면 모뎀도)를 가지고 있다면 심지어 동시에도 가능하다. 이미 위에서 말한 것처럼, 클라이언트 측과 서버 사이에는 PPP에 관한한 일단 연결된 다음에는 아무런 실질적 차이가 없다.

이 문서에서는 전화걸기를 초기화하는 기계를 **클라이언트 측**이라고 간주하고, 전화에 응답하고 이 전화요청을 통해 인증을 검사하는 기계(사용자 이름과 비밀번호와 가능한 기제를 통해)를 **서버**라고 간주한다..

클라이언트 측으로서 PPP를 써서 인터넷 상의 어떤 위치에 있는 하나 혹은 그 이상의 기계에 연결하는 것이 아마도 대부분의 사람들이 관심있는 것 중 하나일 것이다. - 이것이 리눅스 PC를 클라이언트 측으로 사용하는 것이다.

이 문서에 써놓은 순서는 인터넷 연결을 구축하고 자동화하도록 도와줄 것이다.

또한 이 문서는 리눅스 PC를 PPP **서버**로 설정하도록 안내하며 두개의 랜네트워크를 PPP를 써서 (완전히 통하게) 서로 연결하도록 안내한다.(보통 이것을 광역네트워크 -WAN:wide area network-연결이라고들 한다).

1.2 리눅스 배포본의 차이

리눅스 배포본들 사이에는 많은 차이가 있으며 각기 처리 방법에 대한 특징이 있다.

특별히 리눅스(와 유닉스) 컴퓨터의 시작, 인터페이스 설정 등등에는 두가지 다른 방법이 있다.

그것은 **BSD 초기화**와 **System V 체계 초기화**이다. 유닉스 뉴스그룹에 빠져봤다면 이 두 계열 지지자들이 자주 벌이게 되는 종교전쟁을 봤을 것이다. 이런 일을 좋아한다면, 이 뜨거운 논쟁에 참가해 보라.

가장 널리 쓰일 배포본은 다음이다.

- 슬랙 웨어
  BSD형 계열 초기화를 쓴다.
- 레드 햇(그리고 과거의 동맹자 칼데라)
  SysV 계열 초기화를 쓴다.(많이 고쳐진 형태이긴 하지만)
- 데비안
  SysV 계열 초기화를 쓴다.

BSD 형 초기화는 전형적으로 초기화 파일을 `/etc/...`에 넣어두며 이 파일들은 다음과 같다:-

```
        /etc/rc
        /etc/rc.local
        /etc/rc.serial
                (그리고 다른 파일도 있을 것이다)
```

최근에는 몇몇 BSD 체계 초기화가 모든 것을 `/etc`에 넣는 대신 `/etc/rc.d...` 디렉토리에 시작 파일을 넣으려 하고 있다.

System V 초기화는 `/etc/...` 또는 `/etc/rc.d/...` 디렉토리와 그 아래에 있는 많은 부디렉토리 안에 초기화 파일을 둔다:-

```
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 init.d
-rwxr-xr-x   1 root     root         1776 Feb  9 05:01 rc
-rwxr-xr-x   1 root     root          820 Jan  2  1996 rc.local
-rwxr-xr-x   1 root     root         2567 Jul  5 20:30 rc.sysinit
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc0.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc1.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc2.d
drwxr-xr-x   2 root     root         1024 Jul 18 18:07 rc3.d
drwxr-xr-x   2 root     root         1024 May 27  1995 rc4.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc5.d
drwxr-xr-x   2 root     root         1024 Jul  6 15:12 rc6.d
```

만약 이더넷 인터페이스와 연결된 네트워크 루팅이 실제로 설정된 곳이 궁금할 때, 이 파일들을 죽 따라가 보면 이 일을 하는 명령들이 있는 곳을 찾을 수 있게 된다.

1.3 배포본 용의 PPP 설치 도구

몇몇 설치본(예를 들어 레드 햇과 칼데라)에는 X윈도우 설정 PPP 전화걸기 체계가 있다. 이 어쩔거나에서는 이런 배포본 용의 PPP 설정도구에 대해서는 설명하지 않는다. 거기에 문제가 있다면 배포자에게 직접 물어보쇼!

현재 Red Hat 4.x 사용자가 리눅스 소스를 얻는데는 *Red Hat PPP-TIP* `<http://www.interweft.com.au>`이 있고 *Red Hat Software* `<http://www.redhat.com>`에서 지원을 받을 수 있다.

제 2 절 IP 주소

인터넷으로 연결된 모든 장치는 각각 고유한 IP 주소를 갖게된다. 이것은 각 나라마다 정해진 권위기관이 나눠준다.

로컬네트워크(랜)으로 인터넷에 연결되어 있다면 **반드시** 그 로컬네트워크에서 컴퓨터와 장치에 할당되어 있는 네트워크 범위에서 한 IP 주소를 써야한다. 또, **절대**허공에서 IP 주소를 줏어다가 (인터넷에서는 물론이고) 랜에서 쓰면 안된다. 최악의 경우 아예 작동하지 않으며 '훔친' IP 주소를 가지고 허공에서 줏은 IP 주소를 쓰고 있는 컴퓨터와 통신을 나누기 시작하자마자 큰 혼란상태에 빠지게 된다.

이 문서 전체에서 쓰고 있는 IP 주소는 인터넷에 연결될 일이 없는 네트워크에서 사용되도록 빼놓은 '연결않는 네트워크 주소'테이블에서 가져왔다는 것에 주의하라.(일부 예외는 있다.)

특별히 인터넷에 연결되지 않는 랜용의 IP 주소가 있다. 다음과 같다:-

- A 범위에서 하나
  10.0.0.0 (netmask 255.0.0.0)
- B 범위에서 16개
  172.16.0.0 - 172.31.0.0 (netmask 255.255.0.0)
- C 범위에서 256개
  192.168.0.0 - 192.168.255.0 (netmask 255.255.255.0)

자기나라의 책임있는 기관에서 할당받은 IP 주소가 **없다면** 기계의 네트워크 주소는 위에서 골라써야 한다. (krnic 같은...)

이 주소는 인터넷에서 **절대** 사용되지 않는다.

하지만, 인터넷에 연결되어 있는 로컬 이더넷에서는 쓸 수 있다. 왜냐하면 IP 주소가 주어지는 것은 컴퓨터가 아니라 네트워크 인터페이스가기 때문이다. 따라서 예를들어 이더넷 인터페이스가 10.0.0.1을 쓴다고 하고, PPP를 써서 인터넷에 접속할 경우, PPP 인터페이스는 서버에 의해 다른(유효한) IP 주소를 받게 된다.

하지만 리눅스를 쓰면서 리눅스와 ipfwadm 프로그램의 IP 메스커레이드 (NAT:Network address Translation라고도 알려진)기능을 사용하면 이더넷에 있는 기계가 유효한 IP 주소를 갖고 있지 않는 경우에도 (몇가지 서비스 제한은 있지만) 랜을 인터넷에 연결시킬 수 있다.

이것에 대해 더 많은 것을 알고 싶으면 IP Masquerade mini-HOWTO를 보면된다. *Linux IP Masquerade mini HOWTO* `<http://sunsite.unc.edu/mdw/HOWTO/mini/IP-Masquerade>`에 있다.

대부분의 사용자, 그러니까 인터넷 서비스 제공자(ISP:Internet Service Provider 이하 인터넷 서비스 업체)에게 PPP를 통해 연결하는 사람은 IP 주소(보다 정확하게 말한다면 네트워크 주소)를 받지 않아도 된다.

작은 랜을 인터넷에 연결시키겠다고 하면, 많은 인터넷 서비스 업체들이 기존의 IP 주소 공간에서 일정한 하위네트워크(특정한 IP 주소의 범위)을 주게 된다. 대신 IP 메스커레이드를 쓰자.

하나의 PC로 인터넷 서비스 업체를 통해 인터넷에 연결하려는 사용자를 위해서 인터넷 서비스 업체들은 **동적**IP 주소 할당을 사용한다. 이것은 연결과정의 일부로서, 연결 중인 PPP서비스에서 내 기계 쪽에다가 현재 접속주기 동안 사용하게 될 PPP 인터페이스를 알려주는 것이다. 이 주소는 인터넷 서비스 업체에게 접속할 때마다 달라지게 된다.

동적 IP 주소로는 연결할 때마다 같은 IP 주소를 받을 수 **없다.**이것은 sendmail, ftpd, httpd 등등의 리눅스 서버 형 프로그램과 관련된다. 이런 서비스들은 이런 서비스를 제공하는 컴퓨터가 항상 같은 IP 주소로 접근할 수 있다는 전제에 기초하고 있기 때문이다. (아니면 최소한 완전히 자격을 갖춘 도메인 이름 - FQDN : Fully Qualified Domain Name - 과 동등하고, 이름을 IP 주소로 바꿔주는 DNS 번역이 사용가능한 경우라야 한다.)

동적인 IP 주소 할당 때문에 생기는 서비스의 제한에 대해서는 (그리고 가능한 작업 방법에 대해서)이 문서의 뒤에 다루게 된다.

제 3 절 이 문서의 목표

3.1 PPP 클라이언트 측의 설정

이 문서는 리눅스와 PPP를 서서 PPP서버에 연결하고 PPP를 써서 IP 연결을 설정하려고 하는 사람들을 안내한다. PPP가 컴파일되었고, 리눅스 기계에 설치되어 있다고 가정하고 쓰겠다.(하지만 간략하게 PPP 지원을 포함하도록 커널을 재설정/재컴파일하는 데 대해서도 설명한다.)

DIP(SLIP 연결을 만드는 표준 방법)도 PPP 연결을 만드는데 쓸 수 있기는 하지만, DIP 쓰기는 일반적으로 매우 복잡하다. 이 때문에 이 문서는 PPP연결을 설정하려고 DIP를 쓰는 경우는 설명하지 않았다.

대신 이 글에서는 표준 리눅스 PPP 프로그램을 설명한다.(chat/pppd)

3.2 PPP를 써서 랜과 랜, 랜과 인터넷을 잇기

이 문서는 PPP를 써서 두개의 랜을 연결하거나, 한 개의 랜을 인터넷으로 연결하는 (기본적인) 정보를 제공한다.

3.3 PPP서버의 설정

이 문서는 리눅스 PC를 PPP서버로 설정하는 방법을 안내한다.(다른 사람이 내 리눅스 PC로 전화해 들어와 PPP 연결을 할 수 있도록 하는 것을 뜻한다.)

주의할 것은 리눅스를 PPP 서버로 설정하는 데는 부지기수의 방법이 있다는 점이다. 이 문서에는 한가지 방법뿐이다. - 이것은 내가 몇개의 작은 PPP 서버(각각 모뎀 16개가 달린)를 설정하는 데 쓴 방법이다.

이 방법은 잘 되는 것으로 알려져 있지만 가장 좋은 방법은 아니다.

3.4 직접 널 모뎀 연결에서 PPP쓰기

이 문서에서는 두개의 리눅스 PC를 널 모뎀 케이블로 연결하는데 PPP를 쓰는 법에 대해 간략히 소개한다. 다른 운영체제를 리눅스로 연결하는 것도 마찬가지로 가능하다. 그렇게 하려면, 관심있는 운영체제에 대한 문서를 참고해야 한다.

3.5 현재 이 문서에 없는 것들...

- PPP daemon 프로그램의 컴파일
  쓰고 있는 pppd 버전에 따라온 문서를 보라.
- 리눅스에 모뎀을 연결하고 설정하기(상세내용)
  Serial-HOWTO를 보고 모뎀의 특정한 초기화에 대해서는 *Modem Setup Information* `<http://www.in.net/info/modems/index.html>`을 보면 모뎀을 설정하는데 도움이 된다.
- DIP를 써서 PPP연결 만들기
  대신 chat를 쓰라니깐...
- socks나 IP 메스커레이드 쓰기
  이 두가지 일체에 대해 이미 완전한 문서가 있다.
- 자동화 연결을 설정하기 위해 `diald` 쓰기
  `diald` 문서를 보시오
- EQL을 써서 두개의 모뎀을 하나의 PPP 연결로 합치기

- 배포본 특유의 PPP 연결 방법(Red Hat 4.x 네트워크 설정 도구같은 것)
  그 방법을 쓰고 있는 배포본 용 문서를 보시오.
- 점점 늘고 있는 PPP 설정 자동화 도구
  적당한 문서를 보시오

제 4 절 사용가능한 프로그램 버전

이 하우투에서는 리눅스 1.2.x커널과 PPP 2.1.2 또는 리눅스 1.3.X/2.0.x커널과 PPP 2.2를 쓰고 있다고 가정한다.

이 글을 쓰는 시점에서 리눅스 용 PPP의 최신 공식 버전은 ppp-2.2f이다. 새 버전(ppp-2.3)은 아직 테스트(beta)중이다.

PPP 2.2.0을 커널 1.2.13에서 쓸 수 있다. 그럴려면 커널 수정이 필요하다. 몇가지 버그가 고쳐지고 기능도 향상되었으므로 버전 1.2.13 커널 사용자들은 ppp-2.2쪽을 쓰도록 권한다.

마찬가지로 특별히 PPP 2.1.2 프로그램을 리눅스 커널 버전 2.0.X와 쓸 수 없다는 점에 주의하라.

이 문서는 리눅스 커널 2.0.x 용 장전식 모듈을 써서 생기는 문제에 대해서는 다루지 **않았음에** 주의하라. kerneld mini HOWTO와 커널/모듈 2.0.x 문서를 보라. (이것은 리눅스 2.0.x 소스 자리에서 `/usr/src/linux/Documentation/...` 디렉토리에 있다.).

이 문서가 처음 쓰는 사람을 도와주려고 만들어진 만큼, 서로 안정적이라고 알려진 리눅스 커널과 적합한 PPP 버젼을 쓰기 바란다.

제 5 절 다른 유용한/중요한 문서

읽도록 권장함:-

- PPP 일체에 따라오는 문서;
- pppd와 chad의 man 내용;
  (`man chat`와 `man pppd` 명령어를 쓰면 볼 수 있다.)
- 리눅스 네트워크 관리 안내(NAG:the Linux Network Adminstration Guide);
  The Network Administrators' Guide `<http://sunsite.unc.edu/mdw/LDP-books/nag-1.0/nag.html>`을 보라.
- Net-2/3 HOWTO;
  Linux NET-2/3-HOWTO `<http://sunsite.unc.edu/mdw/HOWTO/NET-2-HOWTO.html>`을 보라.
- 리눅스 소스를 설치할 때 `/usr/src/linux/Documentation`에 설치된 리눅스 커널 문서;
- 모뎀 설정 정보 페이지 -*Modem Setup Information* `<http://www.in.net/info/modems/index.html>`을 보라.
- O'Reilly와 연합에서 출판한 훌륭한 유닉스/리눅스 책들(*O'Reilly and Associates On-Line Catalogue* `<http://www.ora.com/>`를 보라). 유닉스/리눅스를 처음 접한다면 가장 가까운 컴퓨터 책방으로 **뛰어가서**(걷지 말고) 즉시 이것들을 사라!
- PPP-FAQ는 알 롱이어씨가 운영하며, *Linux PPP-FAQ* `<ftp://sunsite.unc.edu/pub/Linux/docs/faqs>`에서 볼 수 있다..
  여기에서 왜 PPP가 (제대로) 작동하지 않는지 알아내려고 애쓸 때 쓸모있는 문답식의 유용한 정보가 아주 많다.

- 디양한 출판사와 저자들이 내고 있는 많은 리눅스 책들;
  정력적으로 이런 책들을 찾아보는게 좋다. 리눅스의 개발과 배포본들은 매우 빠르게 발전하는 경향이 있다. 반면에 (일반적으로) 책에는 좀 늦게 반영되곤 한다. 훌륭한 책을 샀어도(거기에는 많은) 날짜지난 정보가 있을 수 있고 처음 리눅스를 쓰는 사람들은 실네트워크와 좌절에 빠질 수 있기 때문이다.

가장 일반적인 리눅스 문서 읽기를 출발점은 *The Linux Documentation Project Home Page* `<http://sunsite.unc.edu/mdw/>`이다. 하우투들은 정기적으로 적당히 내용을 맞게 고치는 편이다.

이렇게 밝힌 다른 문서들을 읽지 않고 이 글만 읽어도 PPP 연결을 만들 수 있기는 하지만, 읽게 된다면 어떻게 돌아가는지 더욱 잘 알 수 있게 될 것이다. 또한 스스로 문제를 해결할 수도 있을 것이다.(아니면 최소한 comp.os.linux 등 뉴스그룹이나 리눅스 메일링 리스트에 훨씬 이지적인 질문을 할 수도 있을 것이다.)

이러한 문서들(적당한 RFCs를 포함한 다른 여러가지와 마찬가지로)은 이 하우투에서 설명할 수 있는 것보다 추가적이고 상세한 설명을 제공한다.

PPP를 써서 인터넷을 랜을 통해 연결한다면 TCP/IP 네트워크 방식에 대해 꽤 많이 알아야 한다. 위의 책에 덧붙여 O'Reilly 책 중 다음과 같은 것이 꽤 쓸만할 것이다. "TCP/IP Network Administration" 그리고 "Building Internet Firewalls" 이다.

5.1 유용한 리눅스 메일링 리스트

다양한 능력의 사용자들 사이의 교류를 위해 많은 리눅스 메일링 리스트(메일링 리스트)가 운영 중이다. 모든 수단을 써서 관심있는 곳에 가입하고 경험과 관점을 알리기 바란다.

조언: 어떤 메일링 리스트는 특정한 "실력 있는" 사용자들을 대상으로 하거나 특정한 주제만을 다룬다. 아무도 훔쳐본다고(가입은 하되 자기는 편지를 보내지 않는 경우) 불평하지는 않지만 만약 적당하지 않은 메일링 리스트에다가 '초보자'질문을 보내면 불총을 맞게 될 것이다.(불꽃은 보이지 않겠지만)

그건 컴퓨터 도사들이 초보자들을 미워해서가 아니라, 그런 메일링 리스트들이 특정한 수준의 난이도가 전제된 특정한 주제를 다루기 위해 만들어졌기 때문이다.

모든 수단을 써서 공개 가입을 허가한 메일링 리스트에 가입하되 그곳의 주제에 적당한 발언만 하도록 해야 한다.

리눅스 메일링 리스트를 접하기에 좋은 장소는 *Linux Mailing List Directory* `<http://summer.snu.ac.kr/~djshin/linux/mail-list/index.shtml>`이다.

제 6 절 PPP를 클라이언트 측에서 작동시키기 위해 해야할 것에 대한

이 문서의 정보는 매우 많다 - 또 각 버젼마다 늘고 있다!

따라서, 이 장에서는 리눅스 체계를 PPP 서버에 클라이언트 측으로서 연결시키기 위해 해야만 하는 동작에 대한 개념적인 개괄을 제공하려고 한다.

6.1 프로그램 받기/깔기

배포본이 PPP 프로그램을 빠뜨렸다면 다음에서 구할 수 있다.*Linux PPP 데몬* `<ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/ppp/ppp-2.2.0f.tar.gz>`.

위의 것이 글을 쓰는 시점에서 최신 공식 버전이다. 어쨌든 이 사이트에서 가장 최신 버젼을 고른다. (ppp-2.3은 글을 쓰는 시점에서는 아직 시험 중이며 아마 곧 발표 될 것이다.)

PPP 일체에 어떻게 컴파일하고 프로그램을 까는지 안내되어 있으므로 **이 하우투에서는 안내하지 않는다!**

6.2 PPP지원을 커널에 컴파일해 넣기

리눅스 PPP 동작은 두 부분으로 되어있다.

- 이미 위에서 말한 PPP 데몬
- PPP용 커널 지원

많은 배포본에서 기본 커널 깔기에 PPP 커널 지원을 제공하지만, 그렇게 하지 않는 배포본도 있다.

부팅할 때 커널에서 다음 메시지를 보여준다면

```
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
```

커널에 PPP 지원이 컴파일되어 있는 것이다.

자기가 갖고 있는 배포본이 어떤 것이든지간에 사람들은 스스로 커널을 컴파일해서 자기 하드웨어 상태에서, 주어진 시스템 소스를 가장 잘 쓸 수 있도록 할 것이다. 이 경우 커널은 기억 장소에서 비울 수 없기 때문에 커널은 기계의 제한된 기억장소를 절약하기 위해 최대한 작게 유지하는 것이 좋다는 점을 염두에 두는 것이 좋다.

7(리눅스 커널 컴파일) 장에서 커널 재컴파일에 대한 최소한의 설명을 제공한다.

더 자세한 사항을 알고 싶으면, Kernal-HOWTO를 보면된다. *The Linux Kernel HOWTO* `<http://sunsite.unc.edu/mdw/HOWTO/Kernel-HOWTO.html>`에 있다.

6.3 인터넷 서비스 업체에게 정보 받기

PPP 서버를 설정할 수 있는 방법에는 무한한 방법이 있다. 인터넷 서비스 업체에 연결하려면 (또는 인트라넷에 접속하기 위한 공용 PPP 서버), PPP 서버가 어떻게 작동하는지 정보를 얻을 필요가 있다.

리눅스를 쓰고 있기 때문에, MS 윈도우 클라이언트 측에 대해서만 알고 있을 몇몇 인터넷 서비스 업체 상담실(과 PPP 인트라넷 서버의 사업장)의 경우는 문의 자체가 어려울 수도 있다.

하지만, 점점더 많은 주소의 인터넷 서비스 업체들이 서비스를 제공할 때 리눅스를 이용하고 있으며 - 리눅스 역시 공용 환경으로 퍼져가고 있기 때문에 문제에 직면했을 때 운이 좋을 수도 있다.

8(PPP 서버에 대해 얻어야만 하는 정보)장은 연결하기 위해 PPP 서버에 대해 알고 있어야 하는 사항에 대해 설명하고 - 어떻게 구하는 지 설명한다.

6.4 모뎀과 직렬 포트 설정

PPP 서버에 연결하고 최고의 자료 이동 속도를 얻으려면, 모뎀이 정확하게 설정되어 있어야 한다.

비슷하게, 모뎀의 직렬 포트와 컴퓨터도 정확하게 설정되어야 한다.

9(모뎀과 직렬 포트의 설정)장에서 이에 대한 내용을 설명한다.

6.5 이름 주소전환 결정(DNS) 설정

PPP를 실행하고 PPP 서버에 로긴을 자동화해주는 파일에 덧붙여 `www.interweft.com.au`같은 이름을 IP 주소로 전환해줄 수 있도록 컴퓨터를 설정하는 몇개의 문서 설정 파일이 있고 이것이 실제 그 컴퓨터에 연결하는데 사용된다. 다음과 같은 것이다.:-

- `/etc/resolv.conf`

- /etc/host.conf

10(DNS 설정)장에서 이것을 설정하는 상세한 내용을 설명한다.

특별히 인터넷에 연결하려고 리눅스 PC에서 네임 서버(Name Server)를 실행시킬 필요는 **없다**(그걸 바라더라도). 필요한 것은 사용할 수 있는(인터넷 서비스 업체 사이트에 있는 것이 좋다) 네임 서버 IP 주소 최소 한개이다.

6.6 PPP와 루트 권한

네트워크 장치(PPP 인터페이스는 네트워크 인터페이스의 하나다.)와 커널 순환 테이블(routing table)의 조작을 요구하는 다른 PPP 서버와 리눅스 컴퓨터 사이에 PPP 연결을 구축할 때 pppd는 루트 특권이 있어야 쓸 수 있다.

상세한 내용은11(PPP쓰기와 루트 특권)장을 보라.

6.7 배포본 PPP 파일 검사와 PPP 선택사항의 설정

PPP를 작동하기 위해 설정해야 할 설정 및 전화걸기 파일이 몇개 있다.PPP 배포본의 일부로 예제가 있으며 이 장에서 어떤 파일을 갖고 있어야 하는지 보여 준다 :-

```
/etc/ppp/options
/etc/ppp/scripts/ppp-on
/etc/ppp/scripts/ppp-on-dialer
/etc/ppp/options.tpl
```

정확히 PPP로 얻고자 하는 것이 무엇인지에 따라 몇몇 추가 파일을 만들 필요가 있다. :-

```
/etc/ppp/options.ttyXX
/etc/ppp/ip-up
/etc/ppp/pap-secrets
/etc/ppp/chap-secrets
```

덧붙여, PPP 데몬은 많은 명령행 선택사항을 쓸 수 있으며 정확한 것을 쓰는 것이 중요하다; 따라서 이 장에서는 표준 PPP 선택사항을 보여줄 것이고, 사용해야 하는 선택사항을 고를 수 있게 도와줄 것이다.

자세한 것은 12(PPP 연결파일 설정)을 보라.

6.8 PPP 서버가 PAP(비밀번호 인증 프로토콜)를 쓸 경우

많은 인터넷 서비스 업체들과 협력 공용 PPP 서버가 PAP를 쓴다. 서버에서 PAP를 쓰라고 요구하지 **않는다면**(수동으로 로긴할 수 있고 표준 사용자 이름/비밀번호 형태의 문자 기반 프롬프트를 받는다면 PAP를 쓰지 않는 것이다.), 이 장은 무시해도 안전하다.

서버에서 사용자 이름과 비밀번호를 입력하라고 프롬프트에 나올 때 입력해 넣어 서버에 로긴하는 방법과 달리, PAP를 쓰는 PPP 서버는 문서 기반의 로긴을 요구하지 않는다.

대신 사용자 인증정보가 PPP 연결 구축의 첫번째 부분인 연결 통제 프로토콜(LCP:Link Control Protocol)의 일부로서 교환된다.

13(PPP서버가 PAP(비밀번호 인증 프로토콜)을 쓸 경우)장에서 PAP를 써서 PPP 연결 구축을 설정하는데 필요한 파일 정보를 제공한다.

6.9 수동으로 PPP연결 설정하기

기본적인 파일을 설정한 다음에, 이것을 리눅스 PC에서 수동으로 연결하고 pppd를 시작해 보는 것이 좋을 것이다.(minicom이나 seyon을 써서)

14(수동으로 PPP연결 설정하기) 장에서 이를 설정하는 데 필요한 상세한 내용을 설명한다.

6.10 PPP 연결의 자동화

일단 수동으로 연결할 수 있었다면 연결 구축을 자동으로 해줄 스크립트 한벌을 설정하는 작업을 할 수 있다.

15(연결 자동화-연결 스크립트의 작성)장에 `chat`에 충분히 주의를 기울인 필수적인 스크립트의 설정과 PPP 서버로 로긴하는 과정의 스크립트쓰기에 대해 설명하고 있다.

이 장에서는 PAP/CHAP 인증 제공측 용 스크립트 뿐만 아니라 사용자 이름/비밀번호 인증 스크립트도 설명한다.

6.11 연결끊기

일단 연결이 만들어지고 작동했다면 이 연결을 다시 끊을 필요가 있다.

17(PPP 연결 끊기)장에 이 내용이 있다.

6.12 문제가 있을 경우

많은 사람이 단번에 PPP를 작동시키는데 문제가 생기기 마련이다. PPP 서버가 다양하고 연결을 설정하는데 요구하는 것도 많다. 비슷하게, PPP에도 많은 선택사항이 있으며 - 선택사항들 사이에는 조합해서 쓰면 아예 작동하지 않는 것도 있다.

로긴할 때와 PPP 서비스를 시작할 때의 문제에 덧붙여서 모뎀과 실제 전화선에서도 문제가 있을 수 있다!

18(문제해결)장에서는 자주 발생하는 에러에 대한 몇가지 기본적인 정보를 주고, 어떤 오류인지 가리고 고치는 법을 알려준다.

다만 이 장에서는 기본적인 것 이상을 바라지는 **않는다**.알 롱이어씨가 운영하는 PPP-FAQ 에 이 주제에 대한 더 많은 정보가 있다!

6.13 연결된 뒤에

일단 PPP 연결이 작동하면(특별히, IP 운영이 작동하면), 리눅스 PPP는 자동적으로 스크립트를 실행시켜 작업을 하기 위해 스크립트에 작성할 수 있는 **모든** 기능을 실행한다(루트 사용자로서).

23(연결된 뒤에)장은 `/etc/ppp/ip-up`에 대한 정보를 알려준다. PPP에서 받게 되는 변수라든가, 전자우편을 인터넷 서비스 업체 계정에서 받거나 내 기계에서 전달을 기다리고 있는 큐의 전자우편을 보내는 등등의 일을 어떻게 하는지 등이다.

6.14 동적 IP 연결 경우 표준 IP 서비스에서 문제

소개에서 주의했던 것처럼, 동적 IP 주소는 리눅스 PC가 인터넷 상에서 서버로 운영되는 능력에 영향을 미친다.

21(동적 IP 연결 경우 표준 IP 서비스에서 문제)장은 (주요한) 서비스 한계에 대한 정보와 (할 수 있다면) 이것을 극복하기 위해 할 수 있는 것에 대해 설명한다.

제 7 절 리눅스 커널의 설정

PPP를 쓰려면 리눅스 커널이 PPP를 포함하도록 컴파일해야 한다. 리눅스 소스코드를 갖고 있지 않다면 - 리눅스 표준 파일 체계에서 /usr/src/linux에 들어 있다. - 리눅스 소스 코드를 구해야 한다.

이 디렉토리를 확인해보라 - 많은 리눅스 배포본들이 설치 과정의 일부로서 이 자리에 소스를 깔아준다(파일과 부디렉토리).

처음 부팅할 때, 리눅스 커널은 많은 양의 정보를 내보낸다. 커널 안에 PPP가 포함되어 있으면 그 정보가 이 때 나타난다. 이 정보를 보려면 syslog 파일을 열어보거나, dmesg |less 명령을 써서 화면에 정보를 나타낼 수 있다. 커널이 PPP 지원을 포함할 경우 아래와 같은 행을 볼 수 있다.

```
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
```

(리눅스 2.0.x 커널 시리즈의 경우).

리눅스 커널 소스은 sunsite.unc.edu나 미러사이트에서 ftp로 구할 수 있다.

7.1 리눅스 커널 소스 깔기

다음은 리눅스 커널 소스를 구하고 까는데 대한 간략한 소개이다. 완전한 정보는 *The Linux Kernel HOWTO* <http://sunsite.unc.edu/mdw/HOWTO/Kernel-HOWTO.html>에서 얻을 수 있다.

리눅스 커널을 구하고 컴파일하려면, 루트로 로긴해야 한다.

1. /usr/src디렉토리로 옮긴다.
   cd /usr/src
2. /usr/src/linux를 열어서 소스가 이미 깔려있는지 확인한다.
3. 소스가 들어있지 않으면 *Linux kernel source directory* <ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0>나 가장 가까운 미러 사이트에서 구한다.
   커널의 이전 버젼을 구하고 싶을 경우(1.2.X같은) *Old Linux kernel source directory* <ftp://sunsite.unc.edu/pub/Linux/kernel/old>에서 찾으면 된다.
4. 적당한 커널을 고른다 - 보통 가장 최신 버젼이 적당할 것이다. 이것을 받아다가 /usr/src에 넣어두면 된다.
   주의:'tar'는 파일묶음이다. - 몇개의 디렉토리에 많은 파일이 압축되어 있을 것이다(리눅스 커널 소스 tar 파일처럼). 이것은 도스의 다중-디렉토리 zip 파일과 꼭같다.
5. 이미 리눅스 소스가 깔려 있는 상태에서 최신 커널로 바꾸려고 하면 옛날 소스를 지워야 한다.다음 명령을 쓴다.
   rm -rf /usr/src/linux
6. 이제 다음 명령을 써서 압축을 푼다.
   tar xzf linux-2.0.XX.tar.gz
7. 이제 cd /usr/src/linux해서 README 파일을 읽는다. 여기에는 설정 및 컴파일을 어떻게 하는지 잘 설명되어 있다. 이 파일을 읽는다.(컴파일 하는 동안 어떻게 하는지 잘 알 수 있는 충분한 시간을 갖고 다 끝낼 때까지 출력을 해서 사본을 갖고 있는 것이 좋다.).

7.2 하드웨어 알기

커널을 다시 컴파일 하려면 **반드시** PC 안에 있는 카드/장치가 어떤 건지 알아야만 한다!!! 몇몇 장치에 대해서는 몇가지 설정에 대해서도 알아둬야 한다. (예를 들어 사운드 카드의 IRQ, I/O 주소 등등)

7.3 커널 컴파일 - 리눅스 1.2.13 커널

설정 과정을 시작하려면, README 파일의 안내에 따라서 적절하게 소스를 설치해야 한다. 커널 설정 과정은 이렇게 시작한다.

```
make config
```

PPP를 쓰려면 커널을 설정해서 PPP 지원을 넣어야 한다.(PPP는 pppd와 PPP 커널 지원을 모두 요구한다)

```
PPP (point-to-point) support (CONFIG_PPP) [n] y
```

다른 make config 선택사항을 PC의 하드웨어와 원하는 리눅스 운영체제의 형태에 맞게 선택한다. 그런 다음 README에 따라 새 커널을 컴파일하고 설치한다.

1.2.13 커널은 PPP장치를 네개만 만든다. 다중 직렬 포트 카드를 쓰려면 커널 PPP 선택사항을 고쳐서 더 많은 포트를 만들어야 한다. (편집해야 할 간략한 내용에 대해 자세하게 알고 싶으면 PPP-2.1.2 배포본에 따라오는 README.linux를 본다.)

주의: 1.2.13 설정 대화창은 뒤로 돌아가는 게 불가능하다. - 그러니까 `make config`에 답하다가 실수하면, CTRL C를 입력해서 설정을 중지한 다음 처음부터 다시 시작해야 한다.

7.4 커널 컴파일 - 리눅스 1.3.x와 2.0.x 커널

리눅스 1.3.x와 2.0.x에서 리눅스 1.2.13과 비슷한 과정을 쓸 수 있다. 역시 README 파일의 안내에 따라 정확히 소스를 설치한다. 커널 설정 과정은 다음과 같이 시작한다.

```
make config
```

하지만 이렇게 할 수도 있다.

```
make menuconfig
```

이렇게 하면 설정 과정 내애서 앞뒤로 움직이는 것이 가능하며 도움말도 있는 메뉴기반의 설정 체계가 나온다.

또한 X윈도우 기반의 설정 인터페이스를 쓰도록 추천한다.

```
make xconfig
```

PPP 지원은 커널에 직접 컴파일할 수도 있고, 장전식 모듈로 컴파일할 수도 있다.

리눅스 기계가 동작하는 시간 중 약간만 PPP를 쓸 경우라면, PPP 지원을 장전식 모듈로 하는 것을 추천한다. 'kerneld'를 쓰면, PPP 연결 과정이 시작할 때 PPP 지원에 요구되는 모듈을 커널이 자동으로 장전한다. 이것은 사용가능한 메모리 공간을 확보한다: 커널은 메모리에서 빠져나올 수 없지만 장전식 모듈은 사용되지 않을 때 자동적으로 제거된다.

이렇게 하려면 장전식 모듈 지원을 사용가능하게 해야 한다:-

```
Enable loadable module support (CONFIG_MODULES) [Y/n/?] y
```

PPP 지원을 추가할 때 다음 물음에 답해야 한다:-

```
PPP (point-to-point) support (CONFIG_PPP) [M/n/y/?]
```

PPP 장전식 모듈을 선택하려면 **M**이라고 답하고, PPP를 커널 일부로 컴파일하려면 **Y**이라고 답하면 된다.

1.2.13 커널과 달리 2.0.x는 PPP 장치를 필요한 만큼 '비행' 중에 만든다. 사용가능한 PPP 장치 주소를 늘리기 위해 소스를 두들겨 고칠 필요가 전혀 없다.

7.5 PPP-2.2와 /proc/net/dev 에 대한 주의

PPP-2.2를 쓸 경우, PPP 장치를 '비행 중에' 만드는 데 따른 부수 효과로서 pppd를 시작해서 장치가 만들어지기 전까지는 /proc/net을 열어봐도 아무 장치도 찾을 수 없다:-

```
[hartr@archenland hartr]$ cat /proc/net/dev
Inter-|   Receive                  |  Transmit
 face |packets errs drop fifo frame|packets errs drop fifo colls carrier
    lo:  92792    0    0    0    0    92792    0    0    0     0    0
  eth0: 621737   13   13    0   23   501621    0    0    0  1309    0
```

ppp 서비스를 시작하자마자 (ppp 서버에서) 다음과 같은 결과를 볼 수 있다.:-

```
[root@kepler /root]# cat /proc/net/dev
Inter-|   Receive                  |  Transmit
 face |packets errs drop fifo frame|packets errs drop fifo colls carrier
    lo: 428021    0    0    0    0   428021    0    0    0     0    0
  eth0:4788257  648  648  319  650  1423836    0    0    0  4623    5
  ppp0:   2103    3    3    0    0     2017    0    0    0     0    0
  ppp1:  10008    0    0    0    0     8782    0    0    0     0    0
  ppp2:    305    0    0    0    0      297    0    0    0     0    0
  ppp3:   6720    7    7    0    0     7498    0    0    0     0    0
  ppp4: 118231  725  725    0    0   117791    0    0    0     0    0
  ppp5:  38915    5    5    0    0    28309    0    0    0     0    0
```

7.6 PPP에 대한 일반적인 커널 설정시 고려해야 할 사항

리눅스 PC를 PPP 서버로 설정할 경우, IP 보내기 (forwarding) 지원을 컴파일해 넣어야 한다. 리눅스를 랜과 연결할 때나 랜을 인터넷에 연결시킬 때도 필요하다.

랜을 인터넷에 연결시킬 때(또는 두개의 랜을 연결시킬 때), 보안에 대해 고려해야 한다. IP 방호벽 역시 커널에 지원해 넣는 것이 필수적이다!

위에서 말한 바 '비연결용' IP 네트워크 주소 중 어느하나를 쓰는 랜을 연결하기 위해 IP 메스커레이드기능을 쓰고자 할 때 이것도 지원해 넣어야 한다.

IP 메스커레이드와 IP 방호벽을 쓰려면 **반드시** `make config` 과정에서 첫번째 질문에 Yes라고 답해야 한다:-

```
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL)?
```

처음 쓰는 사람들은 이게 불필요하다고 생각될 지 모르지만, 많은 사용자들은 실제로 아무 문제 없이 리눅스 2.0.XX커널의 IP 메스커레이드와 IP 방호벽 기능을 쓰고 있다.

일단 커널을 깔고 새 커널을 다시 부팅하게 되면, PPP 링크를 설정하고 시험해볼 수 있다.

제 8 절 PPP 서버에 대한 정보 얻기

서버와 PPP 연결을 만들 수 있기 전에 다음의 정보를 구해야 한다(PPP 서버의 시스템 관리자나 사용자 지원자한테서):-

- 서비스를 받기 위한 전화번호
  외부로 전화를 걸어야 할 경우 외부 전화를 거는 번호도 알아둬야 한다 - 보통 쓰는 번호는 0 또는 9다.
- 서버에서 IP 주소를 동적으로 쓰는지 정적으로 쓰는지?
  서버에서 정적 IP 주소를 쓸 경우에 PPP 연결의 이쪽 끝에서 쓰게 될 IP 주소를 알아야 한다. 만약 인터넷 서비스 업체가 유효한 주소 범위를 알려주었다면, 사용할 수 있는 IP 주소와 네트워크 마스크(netmask)도 알아야 한다.
  대부분의 인터넷 서비스 업체들은 동적 IP 주소를 준다. 위에서 말한 것처럼, 이것은 사용할 수 있는 서비스의 범위에 몇가지 제한이 있다.
  하지만 정적 IP 주소를 쓰고 있다고 해도, 대부분의 PPP 서버는 (보안 상의 이유 때문에) 보안 부담이 있으므로 클라이언트 츠에서 IP 주소를 정하도록 절대 허락하지 않는다. 위에서 말한 정보를 **꼭** 알아야한다!
- 인터넷 서비스 업체 쪽 DNS의 IP 이름은 무엇인가?
  하나만 필요하지만 최소한 두개 이상 있어야 한다.
  여기에 문제가 있을 수 있다. MS 윈도우 95 PPP 설정은 접속 과정의 일부로서 클라이언트 측에 DNS 주소를 넘겨줄 수 있다. 따라서 인터넷 서비스 업체(또는 공통 지원처)는 DNS 서버의 IP 주소가 필요없다고 말하곤 한다.
  리눅스의 경우, 최소한 한개 이상의 DNS 주소가 **있어야 한다**. 리눅스의 PPP 제한은 DNS IP 주소를 연결 시에 동적으로 설정하는 것을 허용하지 않는다 - 그리고 앞으로도 허용할 가능성은 적다.
  주의: 리눅스(PPP 클라이언트 측으로서)가 DNS 주소를 제공측에서 받을 수 없긴 하지만, 서버로 작동할 때는 이 정보를 pppd 선택사항 `dns-addr`를 써서 건네줄 수 있다.
- 서버가 PAP/CHAP를 사용하는가?
  연결할 때 써야되는 "id"와 "secret"이 필요한 경우이다. (아마도 인터넷 서비스 업체 쪽에서 준 사용자 이름과 비밀번호일 것이다.)
- 서버에서 PPP를 자동적으로 시작하는지, 로긴한 다음에 서버에서 PPP를 시작하기 위해 명령을 줘야만 하는가?
- 서버가 MS 윈도우 NT 시스템인지, 그렇다면 MS PAP/CHAP 시스템을 쓰는지?
  많은 공용 랜이 보안성을 높이기 위해 MS 윈도우 NT를 이런 방식으로 사용한다.

이 정보를 잘 적어둔다 - 앞으로 쓰게된다!

제 9 절 모뎀과 직렬 모뎀 설정

모뎀이 제대로 설정됐는지 확인하고 어떤 직렬 포트로 연결되었는지 알아야 한다.

다음과 같다:-

- DOS com1: = Linux /dev/cua0 (and /dev/ttyS0)
- DOS com2: = Linux /dev/cua1 (and /dev/ttyS1)
  이하 동문

포트가 모두 네개인지도 알아두는 게 좋다. 표준 PC 설정에서는 com1과 cpm3이 IRQ4를 같이 쓰고 com2와 com4가 IRQ3을 나눠쓴다.

표준 직렬 포트에서 모뎀과 IRQ를 나누어쓰는 장치가 있을 경우에 문제가 있을 수 있다. 모뎀 직렬 포트가 다른 것과 나눠쓰지 않는 IRQ를 갖고 있는지 확인해야만 한다. 많은 모뎀 직렬 카드(그리고 질좋은 머더보드 직렬포트)는 직렬 포트의 IRQ를 옮길 수 있게 해준다.

리눅스 커널 2를 실행할 경우, `cat /proc/interrupts`라고 입력하면 사용중인 IRQ를 점검할 수 있다. 출력은 다음과 같다.

```
 0:    6766283   timer
 1:      91545   keyboard
 2:          0   cascade
 4:     156944 + serial
 7:     101764   WD8013
10:     134365 + BusLogic BT-958
13:          1   math error
15:    3671702 + serial
```

위에서 보면 IRQ4에 직렬포트가 하나 있고(마우스다) IRQ15에 하나있다(인터넷 PPP 연결에 기반한 영구모뎀이다). (com2, IRQ3에 직렬포트가 하나 있고, com4, IRQ14에도 하나 있지만, 사용하지 않으므로 보이지 않는다.)

경고 - IRQ를 가지고 놀 때는 뭘 하고 있는지 알고 있어야 한다! 컴퓨터를 뜯어내고 카드를 꺼내서 점퍼를 갖고 놀면 끝나는 게 아니며 어떤 IRQ에 무엇이 있는지 알아야 한다. 내 경우 모두 SCSI 기반 PC라서 보통 IRQ14와 IRQ15를 쓰는 마더보드 위의 IDE 인터페이스를 사용불능으로 만들 수 있다.

또한 PC가 다른 운영체제로 부팅할 경우에 IRQ를 옮겨 놓으면 운영체제가 정확히 부팅되지 않거나 -아예 부팅하지 못할 수도 있다는 점을 기억해야 한다.

비표준 IRQ에 직렬 포트를 옮겨 놓을 경우, 각 포트가 쓰는 IRQ가 뭔지 리눅스에 알려줘야 한다. 이것은 `setserial`을 써서 알려줄 수 있고 `rc.local` 또는 시스템V 초기화의 일부로서 부팅과정에서 불려지는 `rc.local` 또는 `rc.serial` 에서 잘 알려줄 수 있다. 위에서 말한 기계의 경우는 아래의 명령을 쓰면 된다.

```
/bin/setserial -b /dev/t
tyS2 IRQ 11
/bin/setserial -b /dev/ttyS3 IRQ 15
```

하지만, `kerneld` 가 요구할 때만 동적으로 장전되는 직렬 모듈을 쓸 경우라면 부팅할 때 한번에 IRQ 등등을 설정할 수 없고 그 값을 잃어버리게 된다. 왜냐하면 직렬 모듈을 내려놓을 때 리눅스는 특별한 설정을 잊기 때문이다.

따라서, 필요할 때 직렬 모듈을 장전하는 경우, 모듈을 장전할 때마다 IRQ를 다시 실정해줘야 한다.

9.1 직렬 포트와 속도 능력에 대한 주의

고속 (외장) 모뎀 (14,400 Baud 또는 이상)을 쓸 경우, 고속 모뎀이 내는 출력 특히 데이터를 압축할 때 내는 출력을 다룰 수 있는 능력이 직렬 포트에 있어야 한다.

이것은 16550(A)같은 최신 범용 비동기 수신 전달자(UART: Universal Asynchronous Receiver Transmitter)이 필요하다. 구형 기계(나 구형 직렬 카드)를 쓸 경우, 직렬 포트가 8250 UART만 갖고 있을 수도 있으며, 이것은 고속 모뎀을 사용할 때 심각한 문제를 일으킬 수 있다.

다음 명령을 쓰면

```
setserial -a /dev/ttySx
```

리눅스는 갖고 있는 UART의 형을 보여준다. 16550A 형 UART를 갖고 있지 않은 경우 새 직렬 카드를 사야 한다(50불 이하에 구할 수 있다.). 새 카드를 살 때는 IRQ를 임의로 옮길 수 있는게 꼭 확인하라!

주의: 16550 UART 칩의 첫 버전에 오류가 있었다. 이 오류는 금방 찾아졌고 고친 칩이 다시 나왔다 - 이것이 16550A UART다. 비교적 적은 주소이긴 하지만 유통되었다. 갖고 있을 가능성은 매우 적지만 특별히 직렬 카드에 무슨 문제가 있을 경우 16550A라고 나오는지 살펴본다.

9.2 직렬 포트 이름

전통적으로 리눅스는 전화해 나가는 쪽을 cuaX장치에 쓰고 전화해 들어오는 쪽을 ttySx장치로 쓴다.

이것이 필요했던 커널 코드가 버전 2.0.x에서 바뀌었으므로 전화받는 쪽이나 나가는 쪽이나 모두 ttySx로 쓸 수 있다. 다음에 나올 커널 버젼에서는 cuaX장치 이름은 없어지는 게 낫다고 생각하고 있다.

9.3 모뎀 설정

PPP를 쓸면 모뎀을 정확히 설정할 필요가 있다 - 그럴려면 **모뎀 설명서를 읽어라**! 대부분의 모뎀은 PPP에 필요한 선택사항이 선택되어 있는 **공장 기본 설정** 상태로 출시된다.정의해야 되는 최소한의 설정은:-

- 하드웨어 흐름제어 (RTS/CTS) (많은 헤이스 모뎀에서 &K3)

표준 헤이스 명령에서 조사해야 하는 설정은 다음과 같다:-

- E1 Command/usr/src/linux-2.0.27/include/linux/serial.h Echo ON (chat 동작에 요구됨)
- Q0 결과 코드 보임(chat 동작에 요구됨)
- S0=0 자동응답 끄기(전화올 때 모뎀이 받기를 원하지 않을 때)
- &C1 연결 후 전달 검출 사용
- &S0 데이타 설정 준비(DSR: Data set Ready) 항상 사용
- (위에 따라 부수적으로) 데이타 단말 준비(DTR: Data Terminal Ready)

다양한 모뎀 제작사와 모델의 서로 다른 모뎀 설정을 제공해주는 곳은 *Modem setup information* <http://www.in.net/info/modems/index.html>이다. 설정에 대해 도와줄 것이다.

조사하는 동안 모뎀과 컴퓨터 사이의 직렬 인터페이스가 어떻게 동작하는지도 도움이 된다. 대부분의 최신 모뎀은 자신과 상대방 모뎀이 모두 다룰 수 있는 최대 속도로 속도를 바꿀 수 있는 전화 회선 인터페이스를 허용하는 한편 일정 속도로 실행하는 것을 허용한다.

이것은 분할 속도 조정으로 알려져 있다. 모뎀이 이를 지원할 경우 모뎀의 직렬 인터페이스를 최고속도로 고정시켜놓을 수 있다(보통 115,200 baud지만 14,400 baud 모뎀의 경우 38,400 baud일 것이다.).

통신 프로그램(minicom이나 seyon같은)을 써서 모뎀 설정을 알아 보고 PPP에 요구되는 것으로 설정한다. 대부분의 모뎀은 AT& V 명령을 주면 현재 설정값을 보여주지만, 모뎀 설명서를 읽어보는 것이 좋다.

설정하다가 정말 돌아버릴 것 같으면 (보통) AT&F 명령을 주면 제정신이 될 수 있다. - 공장 설정값으로 되돌려 준다. (내가 써본 대부분의 모뎀은 공장 기본값에 PPP에 필요한 모든 게 설정되어 있었다. - 하지만 점검해 봐야 한다)

써야할 모뎀 설정 명령을 다 골라냈으면, 결정을 해야 한다: 모뎀의 이 설정값을 적당한 AT 명령으로 불러낼 수 있도록 비휘발설 메모리에 저장할 수 있다. 아니면 PPP 전화걸기 과정의 일부로서 모뎀에 올바른 설정을 건네줄 수 있다.

모뎀을 리눅스에서 인터넷 서비스 업체나 공용 제공측에 전화걸기 위해서만 쓴다면, 비 휘발성 램에 저장하는 것이 가장 간단하다.

그렇지 않고, 모뎀이 여러가지 다른 응용프로그램과 운영체제에서 쓰인다면 모뎀이 전화걸 때마다 정확한 상태에 있을 수 있도록 전화걸 때마다 이 정보를 건네주는 것이 가장 안전하다.(모뎀이 비휘발성 램의 내용을 잃어버릴 경우에 대비해 모뎀 설정을 기억시키는 추가이익을 얻을 수 있으며, 그런 경우는 실제 일어난다)

9.4 직렬 흐름 제어

데이타가 직렬 통신 선 사이를 왔다갔다할 때, 컴퓨터가 다룰 수 있는 것보다 더 빨리 도착하는 경우가 있다.(컴퓨터는 다른 일을 하느라고 바쁠 수 있다. - 리눅스는 다중 사용자, 다중 작업 운영체제라는 점을 생각해보라) 데이타를 잃어버리지 않으려면(데이타가 입력 버퍼를 넘지 않았는데 잃어버리지 않으려면) 데이타 흐름을 제어하는 몇가지 방법이 필요하다.

직렬 회선에서 이것을 하는 데는 두가지 방법이 있다.

- 하드웨어 신호를 사용 (보내기 위해 지우기, 요구하기 위해 지우기 $CTS/RTS : ClearToSend/RequesttoSend]$)$(XON/XOFFCTRLS, CTRLQ)$

터미날 (문자) 연결에서 후자가 괜찮기는 하지만, PPP 링크에서 쓰는 데이타는 모두 8비트다. - 따라서 데이타의 어딘가에 CTRL S, CTRL Q에 해당하는 코드를 가진 데이타가 전달될 가능성이 매우 크다. 따라서, 모뎀이 소프트웨어 흐름제어를 사용하도록 설정되어 있다면 엉네트워크가 된다.

PPP를 쓰는 고속 연결의 경우(8비트 데이타를 쓰는) 하드웨어 흐름제어가 필수적이고, 이 때문에 반드시 하드웨어 흐름제어를 써야 한다.

9.5 모뎀 전화걸기 시험

직렬 포트와 모뎀 설정을 정리해둔 상태에서 인터넷 서비스 업체에게 전화하고 연결해 봄으로써 이 설정이 진짜 동작하는지 확인해 보는 것이 좋다.

단말 통신 일체(minicom같은)를 써서, PPP에 필요한 모뎀 초기화를 설정한 다음, 원하는 PPP서버에 전화해 들어가 PPP 세션을 연다.

(주의: 이 단계에서는 PPP 연결을 만들지는 **않는다**. - 단지 정확한 전화번호를 가지고 있는지, 또 로긴하고 PPP를 시작하기 위해 서버가 보내는 것이 **정확히** 무엇인지 알아내기 위해서 전화하는 것이다.)

이 과정 중에, 전체 로긴 과정을 갈무리(파일로 저장)하거나 조심스럽게 (*굉장히 조심스럽게*) 사용자 이름과 비밀번호를 입력해야 한다고 알려주기 위해 서버가 어떤 명령행을 보내주는지 베껴놓아야 한다(그리고 PPP 연결을 구축하기 위해 필요한 다른 명령도)

서버가 PAP를 쓸 경우, 로긴 명령행을 는 않지만, 대신 연결 제어 프로토콜(의 문자표현)이 화면에 나타나는 것을 보게 될 것이다(쓰레기처럼 보인다).

몇가지 경고:-

- 몇몇 서버는 상당히 지능형이다: 문자 기반 이름/비밀번호나 PAP를 써서 로긴할 수 있다. 따라서 인터넷 서비스 업체나 공용 사이트가 PAP를 사용하는데도 시작할 때 바로 쓰레기가 보이지 않는다고 해서 뭔가 잘못 설정했기 때문이라고 할 수는 없다.
- 어떤 서버는 몇가지 문자형 초기화를 요구한 *다음에* 표준 PAP 과정을 시작한다.
- 어떤 PPP 서버는 수동적이다 - 전화건 클라이언트 측에서 유효한 lcp 패킷을 보내기 전에는 가만히 앉아서 아무 것도 보내지 않는다. 연결한 서버에서 수동 상태에 있을 때는 쓰레기를 전혀 볼수 없다!
- 어떤 서버에서는 실행키를 누르기 전에는 PPP를 실행하지 않는다 - 따라서 정확히 로긴했는데도 쓰레기를 볼 수 없다면 한번 실행키를 눌러보는 것이 좋다!

두번 이상 전화해 보는 것이 좋다 - 어떤 서버는 로긴할 때마다 (예를 들어 로긴 시간) 명령행을 바꾼다. 전화해 들어갈 때 리눅스가 매번 알아내야 하는 필수적인 명령행은 다음과 같다:-

- 사용자 이름을 입력하라는 명령행;

- 비밀번호를 입력하라는 명령행;

서버에서 PPP를 시작할 때 명령을 줘야 한다면, 일단 로긴한 다음에 ppp를 시작한다고 입력할 수 있는 명령행을 알아두어야 한다.

로긴한 다음 자동적으로 서버에서 PPP를 시작하는 경우 화면 상에 쓰레기를 볼 수 있다 - 이것이 PPP 서버에서 내 기계로 PPP 연결을 시작하고 설정할 수 있도록 정보를 보내는 것이다.

다음과 같이 보일 것이다:-

```
~y}#.!}!}!} }8}!}$}%U}"}&} } } } }%}& ...}'}"}(}"} .~~y}
```

(계속 보내도록 냅둔다.!)

몇몇 시스템에서 PPP는 명백히 서버에서 시작할 수도 있다. 이것은 서버가 PPP 로긴가 쉘 로긴에 같은 사용자 이름과 비밀번호 쌍을 쓰도록 설정되었기 때문이다. 이런 경우 로긴한 다음 이 명령을 주어야 한다. 다시 PPP의 서버가 쪽에서 연결이 시작할 때 쓰레기값을 볼 수 있다.

연결한 다음에 (로긴하고 필요하다면 PPP 서버를 시작시킨 다음에) 이것이 즉시 나타나지 않는다면, **실행키**를 눌러 PPP 서버가 시작하는지 보고...

이 시점에서 모뎀을 끊는다. (보통 모뎀이 OK라고 응답할 때까지 +++를 빨리 입력한 다음 ATHO 명령을 주면 된다.)

모뎀을 작동시킬 수 없다면, 모뎀 설명서를 읽고 통신 프로그램의 man 페이지와 Serial HOWTO를 읽는다! 일단 해본 다음에 위의 것을 다시 해보라.

제 10 절 이름 주소 해석(DNS) 설정

사람이 물건에 이름 붙이기를 좋아하는 만큼 컴퓨터는 주소를 좋아한다. TCP/IP 네트워크에서 (인터넷이 바로 이것이다), 우리는 기계를 특정한 이름으로 부른다 - 그리고 모든 기계는 특정한 " 영역(Domain)"에 있다. 예를 들어 내 리눅스 워크스테션은 **archenland**라고 불리며 **interweft.com.au** 영역에 놓여 있다. 사람이 이해할 수 있는 이름은 그래서 archenland.interweft.com.au(보통 완전한 자격을 갖춘 영역 이름-FQDN, Fully Qualified Domain Name이라고 부른다)가 된다.

하지만, 인터넷에서 어떤 컴퓨터가 다른 기계를 이 이름으로 찾았을 때 실제는 인터넷 상에서 통신할 때 사용되는 IP 주소로 이해되는 것이다.

기계(와 영역) 이름이 인터넷에서 실제로 사용되는 주소로 번역(해석)하는 것은 영역 이름 서비스를 제공하는 기계가 하는 일이다.

어떤 일이 일어나느냐 하면:-

- 내 기계는 특정한 컴퓨터의 IP 주소를 알아야 한다. 이 정보를 요구하는 응용프로그램이 내 리눅스 PC의 해석자에게 이 정보를 달라고 요청한다.
- 해석자는 로컬 호스트 파일을 검색하거나 알고 있는 DNS를 검색한다(해석자의 정확한 동작은 `/etc/host.conf`에 정의되어 있다);
- 호스트 파일에서 답이 있으면 답을 돌려주고;
- DNS가 정의되어 있으면, PC는 그 기계를 검색한다;
- DNS 기계가 요구된 이름의 IP 주소를 이미 알고 있으면 이를 돌려주고, 그렇지 않으면 그 정보를 구하기 위해 인터넷의 다른 DNS를 검색한다. 이 네임 서버는 요구한 해석자에게 이 정보를 돌려주고 - 해석자는 요청한 응용프로그램에게 정보를 준다.

내가 기계 이름만 썼는데도 **컴퓨터가** 작업을 하는 데 필요한 IP 주소로 번역할 수 있도록 PPP 연결을 만들 때 리눅스 기계에게 호스트 이름을 IP 주소를 어디서 바꿀 수 있는지 (주소 해석) 정보를 알려주어야 한다.

한가지 방법은 접속할 모든 호스트 이름을 /etc/hosts 파일에 입력하는 것이고(인터넷에 접속하고 있다면 현실적으로 완전히 불가능한 일이다.); 다른 방법은 이름에 해당하는 기계 IP 주소를 쓰는 것이다.(가장 작은 랜에서가 아니라면 번호를 모두 기억하는 것은 불가능하다.)

가장 좋은 방법은 리눅스를 설정해서 어디서 자동으로 이름을 번호를 바꿀 수 있는지 알수 있도록 해주는 것이다. 이 정보는 DNS 시스템에서 제공한다. 필요한 것은 단지 DNS의 IP 번호를 /etc/resolv.conf 파일에 입력하는 것 뿐이다.

10.1 /etc/resolv.conf 파일

PPP 서버 시스템 관리자/ 사용자 지원처에서 두개의 DNS IP 주소를 줄 것이다(필수적인 것은 하나다 - 두개가 있으면 한쪽에 문제가 있을 때 대비할 수 있다.).

앞에서 말한 것처럼, 리눅스는 MS 윈도우 95가 하는 식으로 DNS IP 주소를 설정하지 못한다. 따라서 인터넷 서비스 업체한테 이 정보를 달라고 (친절히) **고집을 부려야 한다**.

/etc/resolv.conf파일은 이런 식이다:-

```
domain your.isp.domain.name
nameserver 10.25.0.1
nameserver 10.25.1.2
```

이 파일을 편집해서(없다면 만들어서) 인터넷 서비스 업체가 제공한 정보를 넣는다. 아래와 같은 소유권과 허가가 되어 있을 것이다:

```
-rw-r--r--   1 root     root           73 Feb 19 01:46 /etc/resolv.conf
```

애초에 랜 상에 있었기 때문에 벌써 /etc/resolv.conf를 갖고 있다면 그냥 원래 있던 파일에다 PPP DNS 서버의 IP 주소를 덧붙이면 된다.

10.2 /etc/host.conf 파일

/etc/host.conf파일이 제대로 설정되었는지도 확인해야 한다. 다음과 같은 모양이다.

```
order hosts,bind
multi on
```

이것은 해석자에게 DNS에 해석 검색을 요청하기 전에 먼저 호스트 파일에 있는 정보를 쓰라고 요청한다.

제 11 절 PPP 쓰기와 루트 특권

PPP가 네트워크 장치를 설정해야 하고, 커널 라우팅 테이블을 바꿔야 하는 등등 때문에 루트 특권이 있어야 PPP를 설정할 수 있다.

루트 이외의 사용자들이 PPP 연결을 설정하려면, pppd 프로그램이 일반사용자도 루트로 사용하도록 설정되어야 한다:-

```
-rwsr-xr-x   1 root     root        95225 Jul 11 00:27 /usr/sbin/pppd
```

/usr/sbin/pppd가 이런 식으로 설정되어 있지 않다면, **루트로써** 다음 명령을 주어야 한다.

```
chmod u+s /usr/sbin/pppd
```

이 명령을 주면 일반 사용자**라도** pppd를 루트 권한으로 사용할 수 있게 해준다. 이것은 일반 사용자가 네트워크 인터페이스와 커널 라우팅 테이블을 설정하는데 필요한 권한을 가지고 pppd를 실행할 수 있게 해준다.

일반 사용자들이 루트 특권으로 사용할 수 있도록 설정해둔 프로그램은 치명적인 보안 구멍이 될 수 있으므로 프로그램의 루트 특권 공개는 대단히 조심스럽게 해야 한다. 많은 프로그램(pppd를 포함해서) 은 일반사용자들이 루트특권으로 사용할 경우의 위험을 최소화하기 위해 조심스럽게 작성되었으므로 이것을 써도 안전하다(하지만 보장은 없다.).

시스템을 어떻게 운영할지에 따라 - 특별히 시스템의 아무라도 PPP 연결을 초기화할 수 있도록 하기를 원한다면, ppp-on/off 스크립트를 읽고, 실행할 수 있도록 하게 될 것이다.(혼자 쓰는 경우라면 괜찮다)

하지만, 아무나 PPP 연결을 시작하는 것이 달갑지 않다면(예를들어, 아이가 내 리눅스 PC에 계정을 갖고 있고 내 감독없이 인터넷에 연결하는 것이 싫다면), PPP그룹을 만들 필요가 있다.(루트로서, /etc/group 파일을 편집한다) 그리고:-

- pppd를 루트와 PPP그룹에게 루트권한으로 공개하고 '다른 사용자에 대한 허가'는 빈 채로 놔둔다. 아래와 같다.

  ```
  -rwsr-x---   1 root     PPP        95225 Jul 11 00:27 /usr/sbin/pppd
  ```

- ppp-on/off 스크립트를 루트와 PPP 그룹이 소유하게 한다.
- ppp-on/off 스크립트를 PPP 그룹이 읽고/실행할 수 있도록 한다.

  ```
  -rwxr-x---   1 root     PPP           587 Mar 14  1995 /usr/sbin/ppp-on
  -rwxr-x---   1 root     PPP           631 Mar 14  1995 /usr/sbin/ppp-off
  ```

- ppp-on/off 에 대한 다른 사용권은 빈 채로 둔다.
- PPP를 사용할 사용자들을 /etc/group 파일의 PPP 그룹에 추가한다.

이렇게 해 두어도 일반 사용자들은 소프트 웨어 제어 상으로 연결을 끊을 수 없다! ppp-off 스크립트를 실행시키려면 여전히 루트 권한이 있어야 한다. 하지만, 어떤 사용자라도 모뎀을 그냥 꺼버릴 수 있다(아니면 내부모뎀의 경우 전화선을 뽑아버릴 수 있다.)

이런 설정의 다른 (그리고 더 나은) 방법은 sudo프로그램을 쓰는 것이다. 이것은 월등한 보안을 주고 대상을 설정해서 어떤 (권한있는) 사용자도 스크립트를 써서 연결을 만들거나/끝내거나 할 수 있도록 한다. sudo를 쓰면 권한있는 사용자를 정해서 PPP 연결을 확실하고 안전하게 만들거나/끝낼 수 있도록 허용할 수 있다.

제 12 절 PPP 연결 파일의 설정

PPP를 모든 사용자들이 쓸 수 있게 하고 싶더라도 일단 **루트**로 로긴해서 PPP 설정에 필요한 디렉토리와 파일을 만들어야 한다.

PPP는 몇 개의 파일을 만들어서 PPP연결을 만들고 설정한다. 파일은 PPP 2.1.2와 2.2가 서로 이름이나 위치가 다르다.

PPP 2.1.2용 파일은:-

```
/usr/sbin/pppd          # PPP 실행파일
/usr/sbin/ppp-on        # 전화 및 연결 스크립트
/usr/sbin/ppp-off       # 연결 끊기 스크립트
/etc/ppp/options        # 모든 연결에 대한 pppd 선택사항
/etc/ppp/options.ttyXX  # 해당 포트 연결에 대한 선택사항
```

PPP 2.2용 파일은:-

```
/usr/sbin/pppd                  # PPP 실행파일
/etc/ppp/scripts/ppp-on         # 전화 및 연결 스크립트
/etc/ppp/scripts/ppp-on-dialer  # 전화걸기 스크립트의 첫부분
/etc/ppp/scripts/ppp-off        # 실제 chat 스크립트 자체
/etc/ppp/options                # 모든 연결에 대한 pppd 선택사항
/etc/ppp/options.ttyXX          # 해당 포트 연결에 대한 선택사항
```

래드햇 리눅스 사용자들은 표준 래드햇 4.X 설치가 이 스크립트들을 /usr/doc/ppp-2.2.0f-2/scripts에 깔았는지 확인해야 한다.

/etc 디렉토리에는 ppp 디렉토리가 있을 것이다:-

```
drwxrwxr-x   2 root     root         1024 Oct  9 11:01 ppp
```

없을 경우에는 위의 소유권과 허가를 갖도록 만든다.

디렉토리가 이미 있었다면 거기에 **options.tpl**이라는 선택사항 예제이 들어있을 것이다.없을 경우에 대비해 아래에 적어 놓았다.

여기에 거의 대부분의 PPP 선택사항에 대한 설명이 있으므로 출력해서 이것을 pppd man 페이지와 비교해가며 읽는 것이 좋다.이것을 /etc/ppp/options 파일을 만들 때 기초로 삼는 것도 좋지만, 아마 이 예제에 있는 모든 명령을 다 넣을 필요없이 자기 파일을 따로 만드는 것이 더 나을 것이다 - 이것이 읽고 관리하는데 훨씬 짧고 쉽다.

여러개의 직렬 회선/모뎀을(PPP 서버가 전형적인 경우다) 갖고 있을 경우에 전화걸기와 받기를 지원할 모든 직렬 포트에 공통적인 선택사항을 갖고 있는 /etc/ppp/options파일을 만들고 각각의 포트에 맞게 개별적인 설정을 갖는 PPP 연결을 할 수 있도록 각 회선마다 개별적인 선택사항 파일을 설정할 수 있다.

포트에 정해놓은 선택사항 파일은 options.ttyx1, options.ttyx2 등등이라고 이름붙이게 된다.(x자리에 직렬 포트에 적합한 문자를 넣는다)

하지만, PPP연결이 하나 뿐이라면 기분좋게 /etc/ppp/options파일을 쓰면된다. 아니면 pppd 명령행 자체에 매개변수로 모든 선택사항을 집어넣을 수도 있다.

/etc/ppp/options.ttySx 파일을 쓰는 설정을 관리하는 쪽이쉽다. 여러개의 다른 장소로 PPP를 연결하는 경우, /etc/ppp/options.site파일에 각각의 장소에 대한 선택사항 파일을 만들 수 있고 연결할 때 PPP 명령의 변수로 줄 수 있다(pppd 명령행에 file [선택파일이름]이라고 pppd 선택사항을 주면 된다).

12.1 제공된 선택사항 예제 파일

PPP 배포본 중 몇개에 이 선택사항 예제파일이 빠져있는 것 같아서 완전한 파일을 옮겨 놓았다. 이 파일을 그대로 /etc/ppp/options파일로 편집하지 않았으면 한다. 대신 새 파일로 복사해다 편집해서 쓰자. 만약 편집하다가 헷갈리면 다시 원래 파일을 보면서 시작할 수 있기 때문이다.

```
# /etc/ppp/options -*- sh -*-
# pppd에 대한 일반적인 선택사항
# created 13-Jul-1995 jmk
# 주석 부분에 대해 김창한이 번역하였음 jimsung@usa.net
# autodate: 01-Aug-1995
# autotime: 19:45
```

```
# 직렬 회선을 설정할 때는 실행 명령 또는 쉘 명령어를 쓴다. 이
# 스크립트는 모뎀에서 전화를 쓰고 ppp로 매번 연결하기 위해 전형적으로
# "chat" 프로그램을 쓸 것이다.
#connect "echo 연결 명령을 써준다."

# pppd가 연결을 끝낸 뒤에 실행명령 또는 쉘 명령을 실행한다.
# 이 스크립트는, 예를 들어, 하드웨어 모뎀 제어 신호가 더이상 유효하지
# 않을 경우에 명령을 내려 모뎀을 끊도록 한다.
#disconnect "chat -- \d+++\d\c OK ath0 OK"
# 32비트 hex 비동기 문자테이블; 각 비트는 pppd가 수신하기 위해
# 탈출하는 문자열이다. 0x00000001은'\x01'을 대신하고 0x80000000은
# '\x1f'를 대신한다.
#asyncmap 0

# 네트워크 패킷을 받거나 보내도록 허용하기 전에 상대방에게 확인을
# 요구한다.
#auth

# 직렬 포트에서 데이타 흐름을 제어하기 위해 하드웨어 흐름제어를 쓴다.
# (RTS/CTS 등)
#crtscts

# 직렬 포트에서 데이타 흐름을 제어하기 위해 소프트웨어 흐름제어를
# 쓴다.(XON/XOFF 등)
#xonxoff

# IPCP 교환이 성공적으로 이루어지면 상대방을 게이트 웨이로 써서 시스템
# 라우팅 테이블에 기본값으로 넣는다.PPP 연결이 끊어지면 이 내용은
# 없어진다.
#defaultroute

# 통신에서 빠져나오기 위한 특정한 문자열을 정한다.(상대방이 자기 쪽
# 비동기 제어 문자테이블을 써서 빠져나오도록 요구하든 아니든 간에)
# 탈출문자 들은 쉼테이블로 나누어진 hex 주소의 목록으로
# 정의된다. 비동기테이블 선택사항이 정의된 제어 문자만을 허용하는데 비해
# 여기서는 탈출 선택사항으로 거의 모든 문자가 허용된다. 탈출할 수 없는
# 문자열은 hex 값 0x20 - 0x3f 또는
# 0x5e이다.
#escape 11,13,ff

# 모뎀 제어 회선을 사용하지 않음
#local

# pppd가 UUCP 형 잠금을 직렬 장치에 사용하여 장치에 대한 포괄적인
# 접근으로부터 안전하게 만든다.
#lock

# 모뎀 제어선을 사용한다. Ultrix 모델에서 이 선택사항은 crtscts
# 선택사항과 같은 하드웨어 흐름제어를 뜻한다.(이 선택사항은 완전히
# 적용되지 않는다)
```

```
#modem

# 통신 중의 최대 수신 값[MRU:Maximum Receive Unit]을 정한다. pppd는
# <n> 바이트 이상의 패킷을 보내지 말도록 상대방에서 요청한다. 최소 MRU는
# 128이다. 기본 MRU는 1500이다. 저속 연결의 경우 128값을
# 추천한다. (40바이트가 TCP/IP 헤더용이고 256 바이트가 데이타용이다)
#mru 542

# 인터페이스 netmask를 <n> 로 설정한다, 255.255.255.0과 같이 <8진수>
# 으로 구성된 32비트 netmask이다.
#netmask 255.255.255.0

# 아무런 로컬 IP 주소도 정해지지 않았을 경우 호스트 이름을 가지고 로컬
# IP 주소를 정하는 (물론 가능할 경우) 기본 동작을 하지 않도록
# 한다. 이것을 선택하면, 상대방은 IPCP 통신 동안 로컬 IP주소를
# 보내주어야 한다.(명령행이나 선택사항 파일에 명확히 정하지 않았을
# 경우).
#noipdefault

# LCP에 "수동" 선택을 사용한다. 이것을 선택하면, pppd는 연결을
# 초기화하려고 하는데; 상대방에서 아무 대답도 없으면 상대방에서 유효한
# LCP 패킷이 올때까지 수동적으로 기다린다(끝내지 않고 기다린다, 이
# 선택을 하지 않으면 대답이 없으면 끝난다).
#passive

# 이 선택을 하면, pppd는 상대방에서 유효한 LCP 패킷을 받을 때까지
# 초기화하기 위한 LCP 패킷을 보내지 않는다(pppd 옛날 버젼의 "passive" 에
# 해당한다.
#silent

# LCP나 IPCP에 대하여 어떤 선택사항에 대해서도 통신을 요청하거나
# 허용하지 않음 (기본값을 사용함)
#-all

# 주소/제어 압축 통신을 사용않음 (기본값으로 주소/제어 영역을 사용하지
# 않는다.)
#-ac

# 비동기 통신을 사용않음 (기본값 비동기 테이블에서는 모든 제어 문자에
# 대해 탈출한다.
#-am

# 분기하여 배경 과정이 되지 않는다.(pppd는 직렬 장치가 정해지지 않았을
# 경우 배경 과정이 된다.)
#-detach

# IP 주소 통신을 사용않음 (이것을 선택하면, 찾아갈 IP 주소가 명령행의
# 선택 사항이나 선택사항 파일안에 정의되어 있어야만 한다.)
#-ip
```

```
# 도사 주소(magic number) 통신을 사용않음 이것을 선택하면, pppd는
# 루프백 회선을 정할 수 없다.
#-mn

# MRU를 통신을 사용하지 않음(1500 등 기본값을 사용함)
#-mru

# 프로토콜 영역 압축 통신을 사용하지 않음(프로토콜 영역 압축 사용않음
# 등 기본값을 사용함).
#-pc

# PAP를 써서 상대방에게 확인을 요구한다. 이것은 양방향 확인을 요구한다
# - 이것은 내 기계에게 인터넷 서비스 업체쪽 기계를 확인하도록 요구하게
# 될 것이기 때문에 (그리고 확인해주지 않으므로) 인터넷 서비스 업체쪽
# 기계에 연결할 때 표준 PAP 확인에 사용하면 안된다.
#+pap

# PAP를 써서 확인하지 않음
#-pap

# 상대방에게 암호교환 확인 프로토콜[CHAP:Cryptographic Handshake
# Authenti-caton Protocol] 확인을 요구한다. 이것은 양방향 확인을
# 요구한다 - 이것은 내 기계에서 인터넷 서비스 업체쪽 기계를 확인하도록
# 요구하게 될 것이기 때문에 (그리고 확인해 주지 않으므로) 인터넷 서비스
# 업체쪽 기계에 연결할 때 표준 CHAP 확인을 사용하면 안된다.
#+chap

# CHAP를 써서 확인하지 않음
#-chap

# 밴 제이콥슨 형의 IP 헤더 압축 통신을 사용않음(압축않음 등 기본값을
# 쓴다)
#-vj

# 오류추적 수준을 올린다(-d와 같다). 이 선택사항이 주어지면, pppd는
# 읽을 수 있는 형태의 모든 송수신 제어 패킷의 목록을 기록한다. 패킷은
# 장비 daemon과 오류추적 수준을 syslog를 통해 기록된다. (pppd가 외부
# 오류추적 가능상태로 컴파일되었으면, daemon 대신 장비 local2를 써서
# 기록한다.)
#debug

# 인증목적으로 로컬 호스트 이름을 영역이름 <d> 에 덧붙인다. 예를들어
# gethostname()이 porsche를 돌려준다고 할 때 완전한 자격을 갖춘
# 영역이름은 porsche.Quotron.com이다. 영역이름(도메인 이름)을
# Quotron.COM으로 정해주려면 이 domain 선택을 쓴다.
#domain <d>

# 커널-수준 PPP 드라이버에 오류추적 코드를 가능하게 한다. 매개변수 n은
# 다음 값을 합친 주소이다: 1은 일반적인 오류추적을 알리도록 하고 2는
# 받은 패킷의 내용을 인쇄하도록 요청하고 4는 전달된 패킷의 내용을
```

```
# 인쇄하도록 요청한다.
#kdebug n

# MTU 값을 <n> 으로 정한다. 상대방이 MRU 통신으로 최소 값을 요구하지
# 않으면 pppd는 커널 네트워크 코드가 PPP 네트워크 인터페이스 n 바이트
# 이상의 데이타 패킷을 보내지 말라고 요청한다.
#mtu <n>

# 로컬 시스템의 이름을 인증목적으로 <n>으로 설정한다. PAP/CHAP를
# 사용할 경우 아마 인터넷 서비스 업체에 접속하는 사용자 이름으로
# 설정해야 할 것이다.
#name <n>

# PAP를 쓰는 상대방에 대해 내 기계를 인증하기 위해 사용자 이름을 <u>로
# 설정한다; 위의 'name'을 쓴 경우 이것을 사용하면 안된다!
#user <u>

# 확인 목적으로 호스트 이름을 로컬 시스템 이름에 쓰게 한다.(name
# 선택사항을 무시하게 한다)
#usehostname

# 인증할 목적으로 가상의 상대방 시스템 이름을 <n>으로 정한다.
#remotename <n>

# 내쪽 시스템의 주소 해석 프로토콜[ARP:Address Resolution Protocol]에
# 상대방의 IP주소와 내쪽 시스템의 이더넷 주소를 추가한다.
#proxyarp

# PAP를 써서 상대방을 인증하기 위해 시스템 비밀번호 데이터베이스를 쓴다.
#login

# 이 선택사항이 주어지면, pppd는 매 n초마다 LCP 응답 요구를
# 보낸다. 리눅스에서는 상대방으로부터 n초동안 아무 패킷도 받지 못할 경우
# 응답요구를 보내게 된다. 보통, 상대방은 응답요구를 보냄으로써
# 응답요구에 답한다. 이 선택사항은 lcp-echo-failure 선택과 함께 상대방과
# 연결이 끊어 졌는지 알아볼 때 쓴다.
#lcp-echo-interval <n>

# 이 선택사항이 주어지면, pppd는 유효한 LCP 응답요구를 받지 못하고
# n개의 LCP 응답요구를 보내면 상대방과의 통신이 끊어졌다고
# 간주한다. 그렇게 되면 pppd는 통신을 끊는다. 이 선택을 쓸 때
# lcp-echo-interval 변수는 0이 아닌 값을 가져야 한다. 이 선택사항은
# 하드웨어 모뎀 제어 회선을 전혀 쓸 수 없는 상황에서 물리적인 연결이
# 끊어졌을 때(예를 들어 모뎀 전화가 끊어졌을 때) pppd를 중지시킬 수
# 있도록 하기 위해 사용된다.
#lcp-echo-failure <n>

# LCP 재시작 간격 (재신호 간격)을 <n>초로 정한다(기본값은 3).
#lcp-restart <n>
```

```
# LCP 종료 요구전달의 최대값을 <n>으로 정한다(기본값은 3).
#lcp-max-terminate <n>

# LCP 설정 요구 전달의 최대값을 <n>으로 정한다;(기본값은 10) 어떤 PPP
# 제공측은 시작하는 게 너무 느리다. 'serial line loopback error'를
# 받았는데 생각하기에 로긴을 제대로 했고, PPP가 서버에서 시작하는 것
# 같으면 이 값을 늘려야 한다.
#lcp-max-configure <n>

# 설정 거절 보내기를 시작하기 전에 LCP 설정-NAK을 보내는 최대값을
# <n>으로 정한다. (기본값 10)
#lcp-max-failure <n>

# IPCP 재시작 간격(재전송 종료)을 <n>로 정한다.(기본값 3)
#ipcp-restart <n>

# IPCP 종료-요구 전달의 최대값을 <n>으로 정한다.(기본값3)
#ipcp-max-terminate <n>

# IPCP 설정-요구 전달의 최대값을 <n>으로 정한다.(기본값10)
#ipcp-max-configure <n>

# 설정-거절 보내기를 시작하기 전에 보내는 IPCP 설정-NAK의 최대값을
# 정한다.(최대 10)
#ipcp-max-failure <n>

# PAP재시작 간격(재전송 종료)을 <n>초로 정한다.(기본값 3)
#pap-restart <n>

# PAP 인증요구 전송의 최대값을 <n>으로 정한다.(기본값 1)
#pap-max-authreq <n>

# CHAP 재시작간격(시도 재전송 종료)을 <n>초로 정한다.(기본값 3)
#chap-restart <n>

# CHAP 시도 전송의 최대 횟수를 <n>으로 정한다. (기본값 10)
#chap-max-challenge

# 이 선택사항이 주어지면, pppd는 상대방에 매 <n>초마다 재시도하게 된다.
#chap-interval <n>

# 이것을 선택하면 로컬 IP 주소가 다른 선택사항에서 정의되어 있더라도
# pppd는 내 로컬 IP 주소에 대한 상대방의 개념을 받아들인다.
#ipcp-accept-local

# 이것을 선택하면 상대방 IP 주소가 다른 선택사항에서 정의되어 있더라도
# 상대방 IP 주소에 대한 상대방의 개념을 받아들인다.
#ipcp-accept-remote
```

12.2 어떤 선택사항을 써야 하나?(PAP/CHAP가 아닌 경우)

글쎄 모든 경우에 따라 다르다고나 할까(휴). 여기에 정해놓은 선택사항은 아마 대부분의 서버에 대해 작동할 것이다.

하지만, 작동하지 않는다면 예제 파일(/etc/ppp/options.tpl) **그리고** pppd man 페이지를 읽어본 **다음**, 연결하려고 하는 서버를 운영하는 시스템 관리자나 사용자 지원부서 사람에게 물어보아야 한다.

여기 있는 연결스크립트 내용 중 일부는 좀더 바꾸기 쉬운 pppd 명령행 선택사항으로도 쓸 수 있다.

```
# /etc/ppp/options (NO PAP/CHAP)
#
# pppd가 배경작업을 분기하는 것을 막는다.
-detach
#
# 모뎀 제어 행을 쓴다.
modem
# uucp 형의 잠금을 써서 직렬 장치에 대한 포괄적인 접근을 막는다.
lock
# 하드웨어 흐름 제어를 쓴다.
crtscts
# 라우팅 테이블에서 현재 연결에 대한 기본 라우트를 선택한다.
defaultroute
# 아무런 "탈출" 제어 문자를 설정하지 않는다.
asyncmap 0
# 최대 전달 패킷의 크기를 552 바이트로 정한다.
mtu 552
# 최대 수신 패킷의 크기를 552바이트로 정한다.
mru 552
#
#-------END OF SAMPLE /etc/ppp/options (no PAP/CHAP)
```

제 13 절 PPP 서버에서 PAP(비밀번호 인증 프로토콜)를 쓸 경우

연결 중인 서버가 PAP나 CHAP 인증을 요구할 경우, 몇가지 작업을 더 해야 한다.

앞서의 선택사항 파일에다 다음 행을 추가한다.

```
#
# 인증 과정에서 인터넷 서비스 업체가 준 사용자 이름을 '호스트
# 이름'으로 쓸 수 있도
# 록 pppd에게 시킨다.
name <your ISP user name>       # 이 행을 편집해야 한다.

#
# PPP *서버*를 운영하고 있고 PAP, CHAP를 쓰려고 한다면 다음 행
# 중적당한 것의 #테이블을 없앤다.PPP 서버에 연결하려고 하는 클라이언트
# 측이라면 (서버가 PAP, CHAP를 쓰는 경우라도) 그러면 안된다. 왜냐하면
# 이 행은 서버에다 내 기계에게 인증받아야 한다고 말하는 셈이기
# 때문이다(그렇게 할 수 없을 뿐만아니라 연결 자체에 실패하게 될
# 것이다).
```

```
#+chap
#+pap
# /etc/ppp/pap-secrets 파일 안에서 암호화된 비밀번호를 사용하고
# 있다면, 다음 행의 #를 없앤다.
# 주의: 이것은 윈도우즈 NT에서 MS RAS에 설정할 수 있는 MS 암호화
# 비밀번호를 쓰는 경우와는 다르다.
#+papcrypt
```

13.1 MSCHAP를 쓸 경우

마이크로소프트 윈도우즈 NT RAS는 CHAP(시도/접속 인증 규약)의 변종을 쓸 수 있도록 설정할 수 있다. PPP 소스 tar 파일 안에서, README.MSCHAP80이라는 파일을 찾아보면 이에 대해 논하고 있다.

pppd에 대해 오류추적을 가능하게 해서 서버가 이 규약을 사용하는 인증을 요구하는지 알 수 있다. 서버에서 MS CHAP 인증을 요구할 경우 아래와 같은 행을 볼 수 있다.

```
rcvd [LCP ConfReq id=0x2 <asyncmap 0x0> <auth chap 80> <magic 0x46a3>]
```

여기서 가장 핵심적인 정보는 **auth chap 80**이다.

MS CHAP를 쓰려면, pppd가 이를 지원하도록 재컴파일해야 한다. 어떻게 컴파일하고 이 변종을 사용할지에 대한 안내를 받을 수 있도록 PPP 소스 파일 안에서 README.MSCHAP80 파일의 내용을 읽어본다.

현재 이 코드는 MS 윈도우즈 NT 서버에 연결할 리눅스 PPP 클라이언트 측만을 지원한다는 점에 주의한다. 이것은 리눅스 PPP 서버가 MSCHAP80 인증을 클라이언트 측으로부터 받을 수 있도록 설정하는 건 지원하지 **않는다**.

13.2 PAP/CHAP 비밀 파일

pap 또는 chap 인증을 사용할 경우, 비밀파일도 만들어야 한다. 다음과 같다:

```
/etc/ppp/pap-secrets
/etc/ppp/chap-secrets
```

이것은 루트 사용자, 루트 그룹에 속해야 하며 허가권은 보안을 위해 740이어야 한다.

PAP와 CHAP에 있어서 제일 주의해야 할 점은 이것이 **컴퓨터 시스템**를 인증하기 위한 것이지 **사용자**를 인증하기 위해서 만들어진 것이 아니라는 점이다.

"예? 뭐가 다르다는 거여?" 하는 소리가 들린다.

음 일단 내 컴퓨터가 PPP연결을 서버와 만들었다고 치자, 내 시스템의 **어떤** 사용자도 이 연결을 쓸 수 있다(당신 뿐만아니라). 이것 때문에 PPP를 써서 두개의 랜(로컬네트워크-LAN:Local Area Network)을 연결해 광역네트워크(WAN:Wide Area Network)을 구축할 수 있는 것이다.

PAP(CHAP **에서처럼**)는 **양방향** 인증을 요구할 수 있다 - 이말은 다른 컴퓨터가 끼어들 때 양쪽 컴퓨터 모두에 유효한 이름과 비밀번호가 요구된다는 뜻이다. 그럼綸Ⓡ워크噬렷딧 PAP-인증 연결을 써서 전화연결 PPP를 설정하는 대부분의 PPP 서버는 이런 방식을 취하지 **않는다**.

인터넷 서비스 업체는 아마 사용자 이름과 비밀번호를 줘서 제 쪽 시스템과 나아가 인터넷에 연결하라고 했을 것이다. 그런데 인터넷 서비스 업체는 내 컴퓨터의 이름에는 전혀 무심하다. 따라서 나는 사용자 이름을 인터넷 서비스 업체 쪽에서 준 내 컴퓨터 이름 대신 써야 한다.

`name user name` 선택사항을 pppd에 줌으로써 그렇게 할 수 있다. 따라서 인터넷 서비스 업체가 준 사용자이름을 쓰려면 다음 행을 `/etc/ppp/options` 파일에다 덧붙인다.

```
name your_user name_at_your_ISP
```

기술적으로 볼때, PAP에는 user [인터넷 서비스 업체가 준 사용자이름]이라고 써야 하지만, pppd는 그 정도는 지능적으로 해결하므로 PAP를 쓰도록 요구할 경우 name을 user로 번역해 준다. name 선택사항을 사용할 경우의 잇점은 이것을 CHAP에서도 쓸 수 있다는 점이다.

PAP가 **컴퓨터**를 인증하는 것이기 때문에, 기술적으로 볼 때 상대방 컴퓨터 이름도 정의해줘야 한다. 하지만, 대부분의 사람들이 오직 하나의 인터넷 서비스 업체에만 연결하므로, 별테이블(*:와일드 카드)를 비밀 파일의 상대방 host 이름으로 쓸 수 있다.

많은 인터넷 서비스 업체들이 서로 다른 단말 서버에 연결하게 되어 있는 다중 모뎀 집합을 운영하고 있음에 주의하면 좋다 - 전화 번호 한개(순환식)로 접속할 수 있지만 각각 다른 이름을 갖고 있는 단말 서버. 따라서 몇가지 환경에서는 상당한 시간을 들이더라도 상대방 컴퓨터 이름이 뭔지를 알아내기 대단히 어렵다. 결국 어느 서버에 연결했는지에 따라 다르니까 말이다!

13.3 PAP 비밀 파일

/etc/ppp/pap-secrets파일은 다음과 같다.

```
# PAP 인증용 비밀파일
# client        server       secret     acceptable_local_IP_addresses
```

위 네 개의 자리는 백공간으로 나뉘어져 있으며 맨 마지막 자리는 빈자리여도 된다(동적인 연결이거나 인터넷 서비스 업체 쪽에서 정적 IP할당을 해 줄 경우 등에)

인터넷 서비스 업체가 사용자 이름으로 fred를 주고 비밀번호로 flintstone를 주었다면 /etc/ppp/option[.ttySx] 파일에 name fred라고 쓰고, /etc/ppp/pap-secrets파일에 다음과 같이 쓴다.

```
# PAP 인증용 비밀파일
# client        server  secret          acceptable local IP addressesfred           *       flintstone
```

이 말은 도메인 기계 이름으로 fred를 주고 (내 도메인 기계 이름이 그게 아니더라도 pppd보고 이걸로 하라고 지정하는 것이다) **모든** 서버에 대해 비밀번호로 flinstone을 쓰라고 하는 것이다.

우리 쪽에서 특정한 도메인, 정적 IP 주소를 강제할 필요가 없다면 도메인IP 주소를 정의할 필요가 없다는 것에 주의한다. 그렇게 할 경우에도 대부분의 PPP 서버가 (보안 때문에) 자기가 준 IP 번호를 내 시스템이 설정하는 것을 허용하지 않으므로 작동하지 않을 성 싶다.

13.4 CHAP 비밀 파일

이건 고도의 인증 수단을 갖고 있을 것을 요구한다 - 즉 내 기계가 상대방 컴퓨터를 인증할 것 **그리고** 상대방 서버가 내 기계를 인증할 것을 허용해야 한다.

따라서 내 기계가 fred이고 상대방이 barney일 경우, 내쪽 기계는 name fred remotename barney라고 설정하고 상대방 기계는 name barney remotename fred라고 각각의 적당한 /etc/ppp/options.ttySx일에 설정해야 한다.

fred 컴퓨터를 위한 /etc/chap-secrets 파일은 다음과 같다.

```
# CHAP 인증용 비밀파일
# client        server  secret             acceptable local IP addresses
fred            barney  flintstone
barney          fred    wilma
```

barney 컴퓨터는 이렇다.

```
# CHAP 인증용 비밀파일
# client        server   secret          acceptable local IP addresses
barney          fred     flintstone
fred            barney   wilma
```

특별히 양쪽 기계가 양방향 인증을 위한 내용이 있어야 한다는 점에 주의하라. 이것은 로컬 기계가 상대방에 대해 인증하도록 허용할 것 **그리고** 상대방 기계가 로컬 기계에 대해 인증하도록 허용할 것을 뜻한다.

13.5 다중 PAP-인증 연결 다루기

몇몇 사용자들은 연결하는, PAP를 사용하는 서버가 하나 이상일 수 있다. 연결하고자 하는 각각의 기계에서 사용자 이름을 받은 것이 서로 다르다고 해도 문제될 것은 없다.

하지만, 어떤 사용자들은 연결하는 시스템 두개(이상일 수도 있고, 모조리 같을 수도 있다)에 같은 사용자 이름을 갖는다. 이런 경우에 /etc/ppp/pap-secrets의 적당한 행을 선택하는데 문제가 생기게 된다.

예상할 수 있듯이, PPP는 이를 극복하는 기제를 제공한다. PPP는 **remotename** 선택사항을 pppd에 씀으로써 서버에 대한 연결이 이루어지면, '알아낸 이름'을 설정하도록 허용한다.

당신이 두개의 PPP 서버에 사용자이름 fred로 접속한다고 치자.

/etc/ppp/pap-secrets는 다음과 같이 설정할 것이다.

```
fred    ppp_서버_1      barney
fred    ppp_서버_2      wilma
```

그러면, ppp_서버_1에 연결하기 위한 설정으로 name fred remotename ppp_서버_1을 ppp-options에 쓰고 ppp_서버_2에 대해서는 name fred remotename ppp_서버_2를 쓴다.

file filename 명령을 써서 pppd에 ppp 선택사항을 고를 수 있으므로, 각각의 PPP 서버에 대해 스크립트를 설정할 수 있으며, 사용할 선택사항 파일을 정확하게 고를 수 있는데다 올바른 remotename 선택사항을 고를 수 있다.

제 14 절 수동으로 PPP 연결 설정하기

이제 /etc/ppp/options와 /etc/resolv.conf파일을 다 만들었고(필요하다면 /etc/ppp/pap|chap-secrets파일도), PPP연결을 수동으로 만들어서 설정을 시험해 볼 수 있다(일단 수동으로 연결이 작동하게 되면, 이 과정을 자동화할 것이다).

수동으로 작업하려면 통신 프로그램이 모뎀을 재설정하지 않고도 중지할 수 있는 기능이 있어야 한다. minicom에 그런 기능이 있다 - ALT Q(혹은 옛날 버젼이면 CTRL A Q)하면 된다.

루트로 로긴했는지 확인한다.

(미니콤같은) 통신 프로그램을 시작하고 PPP 서버로 전화한 다음, 그냥 로긴한다. PPP를 시작할 때 서버에서 무슨 명령을 내려야 한다면, 그렇게 한다. 전에 보았던 쓰레기를 볼 수 있다.

pap나 chap를 쓰는 경우로서, 그냥 상대방 시스템에 연결하기만 해도 ppp가 시작되는 경우라면 로긴하지 않아도 쓰레기를 보게 된다. (어떤 서버에서는 이렇게 되지 않는다 - **실행키**를 눌러서 쓰레기값이 나타나는지 보자.).

이제 통신 프로그램을 *모뎀 재설정 없이* 중지시키고(미니콤에서 ALT Q 또는 CTL A Q), 리눅스 명령행에서(루트로서) 이렇게 입력한다.

```
pppd -d -detach /dev/ttySx 38400 &
```

-d 선택사항은 오류추적 상태로 바꿔준다. - 시스템 로그파일에 ppp 연결 시작 대화내용이 기록된다 - 문제가 있을 때 도움이 된다.

PPP 연결이 만들어지는 동안 모뎀의 불이 반짝반짝할 것이다. PPP 연결이 만들어지려면 약간 걸린다.

이 시점에서 다음 명령을 줘서 PPP 인터페이스를 볼 수 있다.

```
ifconfig
```

가지고 있는 다른 이더넷과 커환 장치 외에, 다음과 같은 것을 볼 수 있을 것이다:-

```
ppp0      Link encap:Point-Point Protocol
          inet addr:10.144.153.104  P-t-P:10.144.153.51 Mask:255.255.255.0
          UP POINTOPOINT RUNNING  MTU:552  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
```

여기서

- inet addr:10.144.153.10 연결된 내 끝의 IP 주소이다.
- P-t-P:10.144.153.5 서버의 IP 주소이다.

(보통, ifconfig는 PPP 서버가 쓰는 것 말고는 이런 IP 주소들을 보여주지 않는다.)

주의: 또한 ifconfig는 연결이 됐으며 실행중이다라 설명해 준다!

위에 보여준 ppp 장치를 못 보거나 아래와 같이 나올 경우

```
ppp0      Link encap:Point-Point Protocol
          inet addr:0.0.0.0  P-t-P:0.0.0.0  Mask:0.0.0.0
          POINTOPOINT  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
```

PPP 연결이 안된 것이다 ... 오류 추적에 관한 다음 장을 본다!

상대방 host(그리고 그 너머)에 대한 라우트도 볼 수 있을 것이다. 그럴려면 다음 명령을 준다.

```
route -n
```

아래와 같은 것을 볼 수 있을 것이다:-

```
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
10.144.153.3    *               255.255.255.255 UH    1500   0        1 ppp0
127.0.0.0       *               255.0.0.0       U     3584   0       11 lo
10.0.0.0        *               255.0.0.0       U     1500   0       35 eth0
default         10.144.153.3    *               UG    1500   0        5 ppp0
```

여기서 특별히 중요한데, ppp 인터페이스를 가리키는 두 개의 항목이 있다는 점이다.

첫번째는 HOST 라우트로서(H테이블이 가리킨다) 우리가 연결하려고 하는 호스트를 보여준다 그 이상은 아님.

두번째는 기본값 라우트로서 pppd 선택사항 `defaultroute`를 줌으로써 구축되었다. 이것은 리눅스 PC에게 로컬 이더넷- 특정한 네트워크 라우트를 갖고 있는 -에 보낼 작정이 아닌 모든 패킷을 PPP 서버에다 보내라고 말해놓았던 그 라우트이다. PPP 서버는 내가 보낸 인터넷을 향한 패킷의 라우트에 응답할 수 있고 돌아오는 패킷을 돌려주게 된다.

두 항목의 라우팅 테이블을 볼 수 있다면, 뭔가 잘못된 것으로, 특별히 syslog 파일에서 기존의 기본라우트를 고칠 수 없다고 알려 줄 경우, 이더넷 인터페이스-이것은 특정한 네트워크 라우트로 **바꿔놓아야** 한다 -을 가리키는 기본값 라우트를 이미 갖고 있는 것이다. **기본값 라우트는 하나만 있을 수 있다!!!**

이 기본값 라우트가 설정되어 있는 곳을 찾아서 시스템 초기화 파일을 찾아 헤매야 한다(`route add default...` 명령을 쓸 것이다.). 이 명령을 `route add net...`로 바꿔 주어야 한다.

ifconfig 출력 등등에 나온 ip 주소에 있는 서버에 핑해보는 것으로 연결을 시험해 본다.

```
ping 10.144.153.51
```

다음과 같은 출력을 받게 된다.

```
PING 10.144.153.51 (10.144.153.51): 56 data bytes
64 bytes from 10.144.153.51: icmp_seq=0 ttl=255 time=328.3 ms
64 bytes from 10.144.153.51: icmp_seq=1 ttl=255 time=190.5 ms
64 bytes from 10.144.153.51: icmp_seq=2 ttl=255 time=187.5 ms
64 bytes from 10.144.153.51: icmp_seq=3 ttl=255 time=170.7 ms
```

이것은 계속 늘어난다 - 끝내려면 CTRL C를 누르면 되고, 그러면 몇가지 정보를 더 얻게 될 것이다:-

```
--- 10.144.153.51 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 170.7/219.2/328.3 ms
```

여기까지 됐으면 좋다.

이번에는 host를 (PPP 서버 것 말고 어딘가 연결해서 실행하고 싶은 다른 자리의 host) 이름으로 핑 해보자. 예를 들어

```
ping sunsite.unc.edu
```

이번에는 리눅스가 핑한 완전히 자격을 갖춘 호스트 이름을 `/etc/resolv.conf`에 정의했던 DNS에서 IP 주소로 바꿔와야 하기 때문에 약간 시간이 걸릴 것이다 - 그러니 걱정할 것은 없다.(하지만 모뎀 불이 반짝반짝 하는지는 보라) 조금 뒤에 다음과 같은 출력을 받게 된다.

```
PING sunsite.unc.edu (152.2.254.81): 56 data bytes
64 bytes from 152.2.254.81: icmp_seq=0 ttl=254 time=190.1 ms
64 bytes from 152.2.254.81: icmp_seq=1 ttl=254 time=180.6 ms
6t bytes from 152.2.254.81: icmp_seq=2 ttl=254 time=169.8 ms
64 bytes from 152.2.254.81: icmp_seq=3 ttl=254 time=170.6 ms
64 bytes from 152.2.254.81: icmp_seq=4 ttl=254 time=170.6 ms
```

다시 CTRL C를 눌러서 통계를 본다.

```
--- sunsite.unc.edu ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 169.8/176.3/190.1 ms
```

아무 응답도 못받았다면, 인터넷 서비스 업체가 운영하는 DNS 서버의 IP 주소를 한번 핑해 본다. 여기서 결과값이 돌아왔다면, /etc/resolv.conf에 문제가 있다고 보아야 한다.

이것조차 안되면, 라우팅 문제가 있거나 인터넷 서비스 업체가 패킷을 돌려보내는 데 문제가 있는 것이다. 위에서 보여줬던 라우팅 테이블을 점검하고 그게 문제가 없다면 인터넷 서비스 업체한테 문의한다. 인터넷 서비스 업체를 시험해 보는 것 중의 좋은 방법은 다른 운영체제로 연결해 보는 것이다. 만약에 인터넷 서비스 업체 너머로 연결이 된다면, 문제는 내 쪽에 있는 것이다.

모든 것이 작동되면, 다음과 같이 입력해서 연결을 끊는다.

```
ppp-off
```

약간 멈춘 다음에 모뎀이 알아서 끊어진다.

동작하지 않으면, 모뎀을 꺼버리거나, 통신 프로그램을 종료시키거나, +++ 를 누른 다음 모뎀에서 OK가 나올 때 ATHO를 넣어서 끝낸다.

pppd가 만든 잠금 파일도 지워야 한다.

```
rm -f /var/lock/LCK..ttySx
```

제 15 절 연결 자동화 - 연결 스크립트의 작성

앞에서처럼 수동으로 계속 로긴할 수 있지만 이를 자동적으로 해주는 스크립트를 만드는 편이 좀더 편할 것이다.

로긴을 자동화하고 PPP를 시작하게 하는 스크립트를 만들어 놓으면 연결 때 (루트 혹은 PPP 그룹의 일원이어야 한다) 명령 하나만 주면 된다.

15.1 사용자 이름/ 비밀번호 인증 경우의 연결 스크립트

PAP/CHAP를 쓰라고 요구하지 않는 경우 이 스크립트를 쓰면 된다!

ppp 일체가 제대로 설치되었다면, 두 개의 예제 파일을 갖게 된다. PPP 2.1.2의 경우 /usr/sbin에 있으며, PPP 2.2의 경우 /etc/ppp/scripts에 있다. 이름은

PPP-2.1.2의 경우

```
ppp-on
ppp-off
```

PPP-2.2의 경우

```
ppp-off
ppp-on
ppp-on-dialer
```

이다.

일단, PPP 2.1.2를 쓸 경우, 예제 파일을 지우도록 강력히 권하는 바다. 거기에 치명적인 문제가 있다 - 잘 된다고 주장하지 말라 - 나도 몇년 동안 썼었다(그리고 이 하우투의 첫 버전에서는 추천하기까지 했다)!

PPP 2.1.2 사용자들을 위해 PPP 2.2 배포본에서 가져온 임시 파일을 실어놓았다. 구식 PPP-2.1.2 스크립트 **대신** 이 스크립트를 복사해서 쓰도록 권한다.

15.2 ppp-on 스크립트

이것이 실제로 연결을 시작하는 스크립트 짝 중 앞의 것이다.

```
#!/bin/sh
#
# PPP 연결을 초기화하는 스크립트다. 이것은 스크립트 쌍의 첫
# 부분이다. 이것은 'ps' 명령으로 볼 수 있으므로 안전한 스크립트 쌍은
# 아니다. 하지만 간단하다.
#
# 이것이 변수이다. 필요하면 고친다.
TELEPHONE=555-1212      # 연결용 전화번호
ACCOUNT=george          # 로긴을 위한 계정 이름('George Burns'같이)
PASSWORD=gracie
# 계정에 대한 비밀 번호(그리고 'Gracie Allen')LOCAL_IP=0.0.0.0
# 알고 있다면 로컬 IP 주소. 동적 = 0.0.0.0REMOTE_IP=0.0.0.0
# 원한다면 상대편 IP 주소. 보통 0.0.0.0NETMASK=255.255.255.0
# 필요한 경우 정확한 netmask
#
# 'ppp-on-dialer'에 쓸 수 있도록 이 정보를 보냄
export TELEPHONE ACCOUNT PASSWORD
#
# 전화를 걸고 로긴하는 스크립트의 위치를 정한다. $PATH 변수가 연결
# 선택사항에서는 먹지 않으므로 파일 이름의 절대 경로를 써야
# 한다. ('루트'계정에서 그렇게 하는 것은 보안구멍이 되므로 요청하지
# 않는다)
#
DIALER_SCRIPT=/etc/ppp/ppp-on-dialer
#
# 연결의 초기화
#
exec /usr/sbin/pppd debug /dev/ttySx 38400 \
        $LOCAL_IP:$REMOTE_IP \
        connect $DIALER_SCRIPT
```

이것이 ppp-on-dialer 스크립트다.:-

```
#!/bin/sh
#
# 이것이 ppp-on 스크립트의 두번째 부분이다. 이것은 원하는 연결을 위해
# 연결 프로토콜을 보여준다.
#
/usr/sbin/chat -v                                                  \
```

```
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDT$TELEPHONE                  \
        CONNECT         ''                              \
        ogin:--ogin:    $ACCOUNT                        \
        assword:        $PASSWORD
```

PPP-2.2에서 ppp-off스크립트는 아래와 같다:-For PPP-2.2, the ppp-off script looks like:-

```
#!/bin/sh
######################################################################
#
# 끊을 장치를 정한다.
#
if [ "$1" = "" ]; then
        DEVICE=ppp0
else
        DEVICE=$1
fi

######################################################################
#
# ppp0 pid 파일이 있고 프로그램이 실행중이면 그것을 중지시킨다.
if [ -r /var/run/$DEVICE.pid ]; then
        kill -INT `cat /var/run/$DEVICE.pid`
#
# kill이 작동하지 않으면 이 pid에 대해 진행 중인 과정이 없는 것이다.
# 아마 잠금 파일이 아직 남아 있음을 뜻할 것이다. 동시에 이 잠금 파일도
# 지워야 할 것이다.
        if [ ! "$?" = "0" ]; then
                rm -f /var/run/$DEVICE.pid
                echo "ERROR: Removed stale pid file"
                exit 1
        fi
#
# 성공. pppd가 필요없는 흔적을 깨끗이 지우도록 한다.
        echo "PPP link to $DEVICE terminated."
        exit 0
fi
#
# ppp 과정이 ppp0에 대해 실행되지 않는다.
echo "ERROR: PPP link is not active on $DEVICE"
exit 1
```

15.3 제공된 PPP 시작 스크립트의 편집

두 부분으로 새 스크립트를 받았으면 순서대로 편집하자.

15.3.1 ppp-on 스크립트

이 스크립트를 인터넷 서비스 업체가 준 사용자 이름과 비밀번호, 전화번호를 감안해 편집한다.

TELEPHONE=같은 각 행은 '=' 오른쪽의 정보(물론 딸린 설명까지 포함해서)를 포함하는 쉘 변수를 실제로 설정한다. 따라서 이 행을 편집해서 인터넷 서비스 업체와 연결에 대해 정확하게 만들어야 한다.

마찬가지로, /etc/ppp/options파일에 있는 IP 주소를 (필요할 경우) 설정할 때는 아래와 같은 행을 삭제한다.

```
$LOCAL_IP:$REMOTE_IP \
```

또한, 쉘 변수 DIALER_SCRIPT가 완전한 위치와 실제로 쓰고자 하는 전화걸기 스크립트 이름을 가리키고 있는지 확인해야 한다. 따라서 이를 옮겼거나 스크립트 이름을 고칠 작정이라면, ppp-on 스크립트에서 이 행을 정확히 편집했는지도 확인해야 한다!

15.3.2 ppp-on-dialer 스크립트

이것이 ppp 연결을 실제로 만드는 스크립트의 뒷 부분이다.

주의: chat 스크립트는 원래 모두 한 줄에 쓰게 되어있으며역슬래쉬는 행이 바뀌었을 때도 한 줄로 인식할 수 있게 해주는 문자이다(사람이 읽을 수 있도록). 따라서 스크립트 자체의 일부는 아니다.

하지만, 어떤 일이 실제로 진행되는지(추측해서) 이해할 수 있도록 세부 사항을 보는 데는 대단히 쓸만하다.

15.4 Chat 스크립트가 뜻하는 것...

chat 스크립트는 "예상_문자열" "전송_문자열" 쌍의 연속이다. 특별히 내 쪽에서 뭔가 보내기 전에 **항상** 받을 것을 **무언가** 기다린다는 점에 주의하자.

아무것도 먼저 **받지 않고서** 뭔가를 보내려고 한다면 예상_문자열에 빈 문자열을 써야만 하고(""로 쓴다). 아무것도 보내지 않고 뭔가를 받으려 할 때도 비슷하다. 또한, 문자열이 몇개의 낱말로 구성되어 있다면(예를 들어 NO CARRIER), 반드시 chat가 하나의 항목으로 받아들이도록 이 문자열을 다 따와야 한다.

예제에서 chat 행은:-

```
exec /usr/sbin/chat -v
```

chat를 부른다. -v는 chat보고 모든 입출력을 시스템 log(보통 /var/log/messages파일)에 복사하도록 시킨다. 일단 기분좋게 chat 스크립트가 믿음직하게 동작했으면 이 행을 편집해서 syslog를 어지럽히는 -v를 지운다.

```
TIMEOUT         3
```

이는 예상하는 문자열의 입력을 삼초동안 기다리는 것이다. 모뎀이 진짜 느린 거라면 5초나 10초로 늘려줄 필요가 있다.

```
ABORT           '\nBUSY\r'
```

BUSY라는 문자열을 받게 되면 동작을 중지한다.

`ABORT '\nNO ANSWER\r'`	
NO ANSWER 라는 문자열을 받으면 동작을 중지한다.	
`ABORT '\nRINGING\r\n\r\nRINGING\r'`	
RINGING이라는 문자열이 (반복적으로) 수신되면, 동작을 중지한다. 이 경우 누군가 당신에게 전화를 거는 중이기 때문이다.	
`'' \rAT`	
모뎀에서 아무것도 기다리지 않고 AT 문자열을 보낸다.	
`OK-+++\c-OK ATH0`	
이건 좀 복잡한데 chat의 오류 복구 능력 중 일부를 쓰기 때문이다. 뭐라는 뜻이냐 하면... OK를 기다렸다가, 받지 못하면 (모뎀이 명령행상태가 아니라서) +++(표준 헤이스 호환 모뎀에서 명령행 상태로 돌아가는 문자열이다)를 보내고 나서 OK를 기다린다. 그런 다음 ATH0(모뎀 끊기 문자열)을 보낸다. 모뎀이 회선 상에서 먹통이 되어 있는 상황에서 스크립트가 처리할 수 있도록 해준다!	
`TIMEOUT 30`	
스크립트의 나머지 부분에 대해 30초의 시간 제한을 설정한다. chat 스크립트가 시간 제한 때문에 취소되는 문제를 겪는다면 45초나 그 이상으로 늘리면 된다.	
`OK ATDT$TELEPHONE`	
OK를 기다렸다가 (모뎀이 ATH0 명령에 대답한 것이다) 원하는 전화번호로 전화한다.	
`CONNECT ''`	
CONNECT를 기다리고 (이것은 상대방 모뎀이 대답할 때 내쪽 모뎀이 보낸다) 아무것도 보내지 않는다.	
`ogin:--ogin: $ACCOUNT`	
다시 몇가지 오류 복구용이 여기 있다. login 프롬프트를 기다리고(...ogin:) 시간이 지나도 받지 못하면, 실행키를 보낸 다음 다시 login 프롬프트를 찾기 시작한다. 프롬프트를 받으면 사용자 이름(쉘 변수 $ACCOUNT에 저장되어 있다)을 보낸다.	
`assword: $PASSWORD`	
비밀번호 프롬프트를 기다리고 내 비밀번호를 보낸다 (역시 쉘 변수에 저장되어 있다). 이 chat 스크립트는 적당한 오류 복구 능력을 갖고 있다. chat는 여기에 보인 것보다 더 많은 기능을 갖고 있다. 더 많은 정보를 원한다면 chat man 페이지를 본다.(`man 8 chat`)	

15.4.1 PPP를 서버에서 시작하기

로긴한 다음 서버에서 자동적으로 pppd를 시작하는 경우에 ppp-on-dialer는 잘 작동하지만 몇몇 서버는 서버에서 PPP를 명확히 시작하도록 요구한다.

서버에서 PPP를 시작하기 위해 어떤 명령을 주어야 한다면, ppp-on-dialer 스크립트를 편집해야 한다.

스크립트의 맨 끝에(비밀번호 행의 뒤에) 추가적으로 **expect send**쌍을 넣어야 한다. 이것은 로긴 프롬프트를 검색할 것이다(본쉘에서 특별한 의미를 갖는 문자-예를 들어 $ 과 [or] (각 인용부호의 여닫이 기호)같은것도 주의한다.

chat가 쉘 프롬프트를 찾으면, chat는 인터넷 서비스 업체 쪽 PPP 서버에서 요구하는 ppp 시작 명령을 내야 한다.

내 경우, PPP 서버는 표준 리눅스 배쉬(bash) 프롬프트를 쓴다.

```
[hartr@kepler hartr]$
```

그리고 서버에서 PPP를 시작하도록 입력할 것을 요구한다.

```
ppp
```

몇가지 오류 복구를 할 수 있도록 하는 것이 좋을 것 같다, 내 경우는 아래와 같이 쓴다.

```
        hartr--hartr    ppp
```

이것은 시간 내에 프롬프트를 받지 못하면, 실행키를 보내고 프롬프트를 다시 찾는 것이다.

프롬프트를 받게 되면 ppp행을 보낸다.

주의: 앞 행의 맨 뒤에 \하나를 집어 넣어 chat가 chat 스크립트 전체를 한 행으로 생각하도록 하는 것을 잊어먹으면 안된다!

운나쁘게, 몇몇 서버는 다양한 변종의 프롬프트를 준다! 미니콤을 써서 수회 로긴해 어떻게 나오는지 알아내고 안정적인 "예상" 문자열을 골라야 한다.

15.5 PAP/CHAP 인증 연결용 chat 스크립트

인터넷 서비스 업체가 PAP/CHAP를 쓸 경우, chat 스크립트는 더 간단하다. chat 스크립트에서 해야할 것은 전화해서, 연결을 기다린 다음 pppd가 로긴을 다루도록 하는 것 뿐이다!

```
#!/bin/sh
#
# 이것은 ppp-on 스크립트의 두번째 부분이다. 원하는 연결을 위해 연결
# 프로토콜을 보여준다.
#
exec /usr/sbin/chat -v                                  \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDT$TELEPHONE                  \
        CONNECT         ''                              \
```

15.6 pppd 오류 추적과 option_file 파일 선택사항

이미 앞에서 본 것처럼, pppd에서 -d를 선택해서 로긴하면 오류 추적 정보를 받을 수 있다. 이 '오류 추적' 선택사항은 아래 것과 동등하다.

새 스크립트를 가지고 새 연결을 구축하는 것이므로, 현재까지는 오류 추적 선택사항을 그냥 두자. (경고: 디스크 공간이 적을 경우, pppd 로긴을 바꾸는 것이 syslog 파일을 급격하게 키우고 문제가 생길 수도 있다 - 이것을 하면 연결에 실패할 것이고 몇분 정도 뒤에 다시 시도할 수 있다)

기분좋게 모든 것이 제대로 작동한다면, 이 선택사항을 없앨 수 있다.

/etc/ppp/options나 /etc/ppp/options.ttySx가 아닌 ppp 선택사항 파일을 불러냈다면, pppd의 file 선택사항에 파일 이름을 적어 주어야 한다 - 다음과 같은 식이다.

```
exec /usr/sbin/pppd debug file options.myserver /dev/ttyS0 38400 \
```

제 16 절 연결 스크립트 시험

새 루트 Xterm을 열거나(X에 있을 경우) 새 가상 콘솔을 열어 루트로 로긴한다.

새 판에, 다음 명령을 준다.

```
tail -f /var/log/messages
```

(또는 해당 시스템 로그 파일 이름)

첫번째 윈도우(또는 가상 콘솔에서)에서 다음 명령을 준다.

```
ppp-on &
```

(아니면 /usr/sbin/ppp-on 파일을 편집해서 새로 붙인 이름으로)

명령 맨 뒤에 &를 써서 스크립트가 배경에서 작업하도록 하지 않으면, ppp가 있는 동안에는 (연결이 끊어질 때까지) 단말 프롬프트를 받지 못하게 된다.

시스템 로그 파일을 보여주는 윈도우로 돌아가 보자.

아래의 내용과 비슷한 것을 보게 될 것이다(chat에 -v를 주고 pppd에 -d를 주었다면) ... 이것은 pppd의 시작 정보를 시스템 로그 파일에 기록하도록 한 것으로 chat 스크립트와 응답이다:-

```
Oct 21 16:09:58 hwin chat[19868]: abort on (NO CARRIER)
Oct 21 16:09:59 hwin chat[19868]: abort on (BUSY)
Oct 21 16:09:59 hwin chat[19868]: send (ATZ^M)
Oct 21 16:09:59 hwin chat[19868]: expect (OK)
Oct 21 16:10:00 hwin chat[19868]: ATZ^M^M
Oct 21 16:10:00 hwin chat[19868]: OK -- got it
Oct 21 16:10:00 hwin chat[19868]: send (ATDT722298^M)
Oct 21 16:10:00 hwin chat[19868]: expect (CONNECT)
Oct 21 16:10:00 hwin chat[19868]: ^M
Oct 21 16:10:22 hwin chat[19868]: ATDT722298^M^M
Oct 21 16:10:22 hwin chat[19868]: CONNECT -- got it
Oct 21 16:10:22 hwin chat[19868]: send (^M)
Oct 21 16:10:22 hwin chat[19868]: expect (ogin:)
Oct 21 16:10:23 hwin chat[19868]: kepler login: -- got it
```

```
Oct 21 16:10:23 hwin chat[19868]: send (hartr^M)
Oct 21 16:10:23 hwin chat[19868]: expect (ssword:)
Oct 21 16:10:23 hwin chat[19868]:  hartr^M
Oct 21 16:10:23 hwin chat[19868]: Password: -- got it
Oct 21 16:10:23 hwin chat[19868]: send (??????^M)
Oct 21 16:10:23 hwin chat[19868]: expect (hartr)
Oct 21 16:10:24 hwin chat[19868]: [hartr -- got it
Oct 21 16:10:24 hwin chat[19868]: send (ppp^M)
Oct 21 16:10:27 hwin pppd[19872]: pppd 2.1.2 started by root, uid 0
Oct 21 16:10:27 hwin pppd[19873]: Using interface ppp0
Oct 21 16:10:27 hwin pppd[19873]: Connect: ppp0 <--> /dev/cua1
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(LCP): Sent code 1, id 1.
Oct 21 16:10:27 hwin pppd[19873]: LCP: sending Configure-Request, id 1
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfreq(LCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd MRU
Oct 21 16:10:27 hwin pppd[19873]: (1500)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd ASYNCMAP
Oct 21 16:10:27 hwin pppd[19873]: (0)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd MAGICNUMBER
Oct 21 16:10:27 hwin pppd[19873]: (a098b898)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd PCOMPRESSION
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: rcvd ACCOMPRESSION
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: lcp_reqci: returning CONFACK.
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(LCP): Sent code 2, id 1.
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfack(LCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 1, id 1.
Oct 21 16:10:27 hwin pppd[19873]: IPCP: sending Configure-Request, id 1
Oct 21 16:10:27 hwin pppd[19873]: fsm_rconfreq(IPCP): Rcvd id 1.
Oct 21 16:10:27 hwin pppd[19873]: ipcp: received ADDR
Oct 21 16:10:27 hwin pppd[19873]: (10.144.153.51)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: ipcp: received COMPRESSTYPE
Oct 21 16:10:27 hwin pppd[19873]: (45)
Oct 21 16:10:27 hwin pppd[19873]:  (ACK)
Oct 21 16:10:27 hwin pppd[19873]: ipcp: returning Configure-ACK
Oct 21 16:10:28 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 2, id 1.
Oct 21 16:10:30 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 1, id 1.
Oct 21 16:10:30 hwin pppd[19873]: IPCP: sending Configure-Request, id 1
Oct 21 16:10:30 hwin pppd[19873]: fsm_rconfreq(IPCP): Rcvd id 255.
Oct 21 16:10:31 hwin pppd[19873]: ipcp: received ADDR
Oct 21 16:10:31 hwin pppd[19873]: (10.144.153.51)
Oct 21 16:10:31 hwin pppd[19873]:  (ACK)
Oct 21 16:10:31 hwin pppd[19873]: ipcp: received COMPRESSTYPE
Oct 21 16:10:31 hwin pppd[19873]: (45)
Oct 21 16:10:31 hwin pppd[19873]:  (ACK)
```

```
Oct 21 16:10:31 hwin pppd[19873]: ipcp: returning Configure-ACK
Oct 21 16:10:31 hwin pppd[19873]: fsm_sdata(IPCP): Sent code 2, id 255.
Oct 21 16:10:31 hwin pppd[19873]: fsm_rconfack(IPCP): Rcvd id 1.
Oct 21 16:10:31 hwin pppd[19873]: ipcp: up
Oct 21 16:10:31 hwin pppd[19873]: local  IP address 10.144.153.104
Oct 21 16:10:31 hwin pppd[19873]: remote IP address 10.144.153.51
```

(주의- 나는 정적인 IP 주소를 쓴다 - 따라서 나의 기계는 이를 PPP 서버로 보낸다 - 만약 동적 IP 주소를 쓴다면 이를 볼 수 없을 것이다) 또한, 이 서버는 접속 된 뒤에 ppp를 시작하는 특정한 명령을 요구한다.

괜찮아 보인다 - 아까처럼 IP 주소와 호스트 이름으로 핑해보자.

웹 검색기나 뭐든 실행시키고 정보의 바다를 헤엄치자 - 당신은 연결되었다!

제 17 절 PPP 연결 끊기

PPP 연결을 끝냈을 때는, 표준 ppp-off 명령을 써서 끝낸다(기억할 것 - 루트이거나 PPP 그룹의 일원이어야 한다!).

시스템 로그파일에서 다음과 같은 것을 볼 수 있다:-

```
Oct 21 16:10:45 hwin pppd[19873]: Interrupt received: terminating link
Oct 21 16:10:45 hwin pppd[19873]: ipcp: down
Oct 21 16:10:45 hwin pppd[19873]: default route ioctl(SIOCDELRT): Bad address
Oct 21 16:10:45 hwin pppd[19873]: fsm_sdata(LCP): Sent code 5, id 2.
Oct 21 16:10:46 hwin pppd[19873]: fsm_rtermack(LCP).
Oct 21 16:10:46 hwin pppd[19873]: Connection terminated.
Oct 21 16:10:46 hwin pppd[19873]: Exit.
```

`SIOCDELRT`갖고 걱정할 필요없다 - 이건 기냥 pppd가 이제 끝냅니다하는 주의사항을 보내는 것 뿐이므로 걱정할 게 아니다.

제 18 절 오류 추적

연결이 작동하지 않는데는 많은 이유가 있을 수 있다 - chat가 올바로 완료되지 않았을 수도 있고, 회선이 엉네트워크일 수도 있고, 등등. 따라서 지적된 것에 따라 syslog를 점검한다.

18.1 PPP 지원을 커널에 컴파일해 넣었는데도...

PPP 지원을 커널에 컴파일해 넣었는데도 pppd를 실행시키려고 하면 커널에서 ppp를 지원하지 않는다고 하는 것이 제일 흔한 문제 중 하나다! 이런 일이 일어나는 데는 많은 이유가 있다.

18.1.1 올바른 커널로 부팅했는지?

앞서 ppp를 지원하도록 커널을 재컴파일했지만 새 커널로 부팅하지 않을 수도 있다. 이것은 `/etc/lilo.conf`를 고치지 않고 lilo를 다시 실행시켰을 때 이런 경우가 생긴다.

커널에 대한 점검은 `uname -a`라고 명령하면 얻어낼 수 있는데, 아래와 같은 줄이 나타날 것이다.

```
Linux archenland 2.0.28 #2 Thu Feb 13 12:31:37 EST 1997 i586
```

여기에는 커널의 버전과 커널이 컴파일된 날짜가 있다. - 이것을 보면 어떻게 된 것인지 잘 알 수 있다.

18.1.2 ppp 커널 지원을 모듈로 컴파일했는가?

ppp지원을 모듈로 컴파일했는데, 모듈을 make, install하지 않았다면, 이런 오류가 나타난다. kernel-HOWTO와 README 파일을 `/usr/src/linux`디렉토리애서 찾아 읽어본다.!

모듈과 관련된 다른 가능성은 자동적으로 모듈이 장전되리라고 생각했는데 `kerneld`(실행 중에 자동장전 및 장전해제를 해준다)를 실행시키지 않은 경우이다. kerneld mini-HOWTO를 읽어서 kerneld의 설정에 대해 알아본다.

18.1.3 커널에 맞는 PPP 버전을 쓰고 있는가?

반드시 ppp-2.2는 커널 버전 2.0.x와 써야 한다. ppp-2.2를 커널 버전 1.2.x와 함께 쓸 수 있기는 하지만(커널 수정을 할 경우), 그렇지 않다면 ppp-2.1.2를 써야 한다.

18.1.4 루트 특권으로 pppd를 실행하고 있는가?

pppd를 루트 사용자로서 쓰고 있는 것이 아닐 경우에도(pppd가 루트 사용자용이 아닐 경우) 이런 출력을 받게 된다.

18.2 모뎀은 연결되었는데 ppp가 되지 않음

여기에는 많은 이유가 있다.(comp.os.linux를 한번 본다)

가장 일반적인 실수는 스크립트 어딘가에 입력을 잘못한 경우다. 이런 경우에는 리눅스 PC와 서버 사이에 이루어진 chat 대화 내용을 syslog(`/var/log/messages`) 안에서 확인해 볼 수밖에 없으며, *각 행별*로 이를 만들어 가야 한다. 이를 점검하기 위하여 수동으로 ppp 서버에 전화해 볼 필요가 있다.

실제 프롬프트의 내용을 매우 조심스럽게 점검해 볼 필요가 있다 - 그리고 인간은 입력해 넣은 것 대신 자기가 생각한 데로 읽는 경향이 있다는 점을 염두에 두자!

18.3 시스템 로그 파일에서 보니 "serial line is not 8bit clean..."라고 나오는데?

여기에도 많은 이유가 있을 수 있다 - 직렬 회선 귀환 등등이며, 여러가지 것 중 하나(혹은 몇개)가 이유가 될 수 있다.

이 경우 무엇이 문제인지 알려면, pppd 자체에 관한 아래 몇가지 장면에 대해 감을 잡고 있어야 한다.

pppd가 시작할 때 원격 기계에 LCP(연결 제어 프로토콜:Link control protocol)패킷을 보내게 된다. 이때 유효한 응답을 받으면 IPCP(IP 제어 프로토콜:IP control protocol) 패킷을 쓰는 다음 단계로 가며 이 통신이 완료되었을 때에만 PPP연결을 시작할 수 있는 실제적인 ip 배치 작업이 시작된다.

lcp 패킷을 보낼 때 원격에서 작동 중인 ppp 서버가 없다면, 로긴 과정에서 원격 쪽에 반영되어 있을 것이다. 이 패킷은 8비트를 사용하므로, 8번째 비트가 빠진 것으로 반영된다(ASCII는 7비트라는 점을 기억할 것). PPP는 이것을 보고 불평한다.

이러한 반영이 일어날 수 있는 데는 몇가지 이유가 있다.

18.3.1 서버에 정확하게 로긴하지 못했을 경우

chat 스크립트가 완료되면, pppd는 내쪽 PC에서 시작한다. 하지만, 서버에서 로긴 과정을 완료하지 않았다면(서버에서 PPP를 시작하도록 요구한 명령을 보내지 않은 경우도 포함해서) , PPP는 시작하지 않는다.

따라서, lcp 패킷은 되돌아오지 않고 이런 오류를 당하게 된다.

조심스럽게 점검해보고 (필요하다면) chat 스크립트를 고쳐야 한다.(앞을 보라)

18.3.2 서버에서 PPP를 시작하지 않았다.

어떤 PPP 서버에서는 자기 쪽에서 ppp를 시작하기 전에 로긴 과정이 끝난 다음 특정 명령을 주거나 실행키를 누르도록 요구한다.

chat 스크립트를 점검한다 (앞을 보라).

수동으로 로긴한 다음 PPP를 시작하기 위해 실행키도 보내야 한다면, chat 스크립트의 맨 밑에다 빈 기다림/보내기 쌍을 추가하기만 하면 된다(보내기 자리에 빈 문자열을 넣어주면 실제로는 실행키 값을 보낸다).

18.3.3 상대방 PPP 과정이 시작하는데 너무 느림

이때는 잔머리를 좀 굴려야 한다!

기본값으로 리눅스 pppd는 최대 10개의 lcp 설정 요구를 보내도록 컴파일 되어 있다. 만약 서버가 시작하는데 좀 느리다면, 상대방 PPP 가 이를 받을 준비가 다 되기 전에 이 10번의 요구를 다 보낼 수 있다.

그러면 내 기계에서는, pppd가 돌아온 10개를 볼 것이고(8비트가 짤린) 끝내게 된다.

이를 해결하는데 두가지 방법이 있다:-

ppp 선택사항에다 `lcp-max-configure 30`를 덧붙인다. 이때 늘린 값만큼 pppd는 lcp 설정 패킷을 보낸 다음에 끝낸다.정말로 서버가 느리다고 치면, 이것보다도 값을 더 늘려야 한다.

좀더 잔머리를 굴리는 방법이 있는데 앞에서 이미 보았듯이 PPP 서버에 수동으로 로긴할 때 PPP가 시작하면서 **처음에** 보내는 쓰레기 값처럼 보이는 것 맨 앞에 나타나는 기호가 (~)이다.

이 지식을 이용해서 chat 스크립트의 맨 뒤에다 ~를 기다렸다가 아무 것도 보내지 않는 `기다림/보내기`쌍을 추가할 수 있다.

```
\~          ''
```

주의: ~이 쉘에서 특별한 의미를 갖고 있으므로, 탈출문자여야 한다(그래서 탈출문자 역슬래쉬가 붙어있다).

18.4 기본 라우트가 설정되지 않았다.

pppd가 기본값 라우트를 설정하지 않는다면, (비교적 정확한데) 기존의 기본값 라우트를 지우거나/대체하지 않기 때문이다.

이런 오류가 생기는 보통의 경우는 몇몇 배포본이 이더넷 카드를 특정한 네트워크라우트로 설정하지 않고 기본값 라우트로 설정하기 때문이다.

Linux NAG와 Net2/3 HOWTO를 읽어보면 이더넷 카드를 올바로 설정하는 것과 이와 관련된 라우트에 대해 정보를 얻을 수 있다.

다른 경우는 랜이 자체적으로 게이트웨이/라우터를 이미 쓰고 있고 내 라우팅 테이블이 이 기본값 라우트를 이미 가리키도록 설정되어 있는 경우다.

이 마지막 상황을 고치려면 IP 네트워크에 대한 약간의 지식이 필요하며 이것은 이 하우투의 범위를 벗어난다. 전문가의 조언을 받기를 바란다(뉴스그룹 안에서 로컬적으로 물어볼 수 있는 있는 누군가에게)

18.5 다른 문제

이것 말고도 ppp를 정확하게 연결하고 실행하는 데 실패할 만한 이유는 많다.

PPP FAQ(실제 문답식으로 되어 있다)를 보자. 이것은 상당히 대화식인 문서이며, 해답이 있다! 내 자신의 (슬픈) 경험에 비추어 본다면, 문제가 거기에 없다면, 문제는 ppp의 오류 때문이 아니다! 내 경우 적당한 커널 모

듈로 판을 올리지 않은 ELF 커널을 쓰고 있었던 것이다. 나는 겨우 이틀(거기에다 하루밤 거진다)을 완벽한 PPP 서버가 뭐가 잘못됐는지 알아내기 위해 썼을 뿐이지만!

제 19 절 도대체 알 수 없을 때 도움을 받을 수 있는 곳

PPP 연결을 만들 수 없을 때는, 이 문서를 다시 읽으면서 모든 것을 점검해 보자 - "chat -v ..."과 "ppp -d"를 써서 만든 syslog 파일 출력과 비교해 가면서.

PPP 문서와 FAQ에다 여기 언급된 다른 문서들도 읽어보자!

그래도 먹통이라면, comp.os.linux.misc와 comp.os.linux.networking 뉴스그룹에 들어가보자. 여기는 comp.protocols.ppp와 같이 PPP에 대해 도움을 줄 수 있을 만한 사람들이 사려깊게 정기적으로 읽는 곳이다.

나에게 개인적으로 email을 보낼 수도 있지만, 낮에 직업이 (그리고 생활도) 있기 때문에 (대답은 모두) 내 일의 부담과 개인적인 삶의 상태에 달려있는 만큼 빨리 답장을 보낸다는 보장이 없다.

특별히 - **오류 추적 출력을 뉴스그룹이나 나에게 보내서는 안된다** - 뉴스그룹에 보내는 것은 거대한 분량의 부하를 네트워크에 부여하는 짓이 될 것이고 후자는 `/dev/null` 로 버려지게 된다 (내가 특별히 요청한 경우를 빼고).

제 20 절 일단 연결된 후의 보편적인 문제

알게 될 수 있는 문제 중 하나로 많은 인터넷 서비스 업체들이 새 계정을 줄때 나눠주는 프로그램 일체만을 지원한다는 것이다. 이것은 (전형적으로) 마이크로소프트 윈도우즈일 것이다 :-(- 그리고 많은 인터넷 서비스 업체 측 지원부서는 유닉스(나 리눅스)에 대해서는 깡통이다. 따라서 제한된 지원을 각오해야 한다!

당신은 물론 개인적으로 친절하게 하고, 리눅스에 대해 교육시켜 줘야 한다.(어떤 인터넷 서비스 업체 쪽 지원부서 사람도 인터넷 용어에 대해 잘 알아야 하고 이것은 집에 리눅스 기계를 갖고 있어야 함을 뜻한다 - 물론이지!)

20.1 연결한 PPP 서버 이상은 볼 수 없다.

좋다 - PPP연결은 됐고 실행되며 PPP 서버를 IP 주소(`ifconfig ppp0`에 나오는 제2, 또는 "원격" IP 주소)로 핑해볼 수 있는데도 그 너머에는 도달할 수 없다는 뜻이다.

무엇보다도 먼저, /etc/resolv.conf에 정의한 IP 주소를 이름으로 핑해본다. 이것이 된다면, PPP 서버 너머를 **볼 수 있다**(이것이 연결한 "상대방" IP 주소와 같은 것일 경우만 빼면). 그러면 인터넷 서비스 업체 등등의 완전한 인터넷 이름을 핑을 해본다.

```
ping my.provider.net.au
```

이것이 실행되지 않으면, 이름 전환에 문제가 있는 것이다. 이것은 아마도 /etc/resolv.conf 파일에 문제가 있기 때문이다. 인터넷 서비스 업체한테 전화해 본 다음에 필요한 정보를 조심스럽게 점검해 보자. 다 제대로 된 것 같으면, 인터넷 서비스 업체에게 전화해서 IP 주소를 정확히 기록했는지 점검해 보자.

그래도 동작하지 않으면(그리고 인터넷 서비스 업체가 자기네 네임 서버가 있고 실행 중이라고 확인해 주었으면), 거기말고 다른데 문제가 있을 것이다 - 따라서 조심스럽게 리눅스 설치 과정을 조심스럽게 점검해보길 권한다(특별히 파일 허가를 볼 것).

그래도 여전히 인터넷 서비스 업체의 IP 네임 서버를 IP 주소로 핑할 수 없다면, 그쪽이 다운됐거나 (전화해서 물어보자) 인터넷 서비스 업체 쪽의 라우트에 문제가 있는 것이다. 다시 전화해서 이것을 점검해 보자.

한가지 가능성은 "원격"이 IP 보내기 선택사항이 커널에 정의되어 있지 않은 리눅스 PPP 서버일 경우이다.

가장 좋은 시험은 마이크로 소프트 윈도우즈를 지원하는 프로그램을 써서 인터넷 서비스 업체에게 연결해 보는 것이다. 다른 운영체제에서 완전히 동일한 계정으로 모든 것이 제대로 작동한다면, 문제는 리눅스 시스템에 있는 것이지 인터넷 서비스 업체에게 있는 것이 아니다.

20.2 전자우편을 보낼 수는 있는데 받을 수는 없다

동적 IP 주소를 쓰고 있다면, 당연한 것이다. 아래의 ”서비스의 설정”을 본다.

20.3 왜 다른 사람들이 내 기계의 finger, WWW, gopher, talk 기타 등등을 쓸 수 없는가?

역시 동적 IP 주소를 쓰고 있다면 완전히 정상적이다. 아래의 ”서비스의 설정”을 본다.

제 21 절 동적 IP 주소를 가지고 인터넷 서비스 쓰기

동적 IP 주소를 사용할 경우(대부분의 인터넷 서비스 업체들이 상당히 돈을 많이 내지 않는 한 동적인 IP 주소만 줄 것이다), 이런 식으로 파는 상술의 한계를 알아두어야 한다.

일단 외부로 향하는 서비스 요구는 다 잘된다. sendmail로 전자우편을 잘 보낼 수 있다(sendemail을 잘 설정했으면), 다른 장소의 ftp 파일을 받을 수 있으며, 다른 기계의 사용자들을 finger할 수 있고, 웹도 검색할 수 있고, 등등.

특별히, 연결이 끊겨있을 때도 기계에 전자우편을 만들어 두었다가 응답할 수 있다. 전자우편은 인터넷 서비스 업체에게 다시 전화할 때까지 메일 큐에서 그냥 기다린다.

하지만, 기계가 인터넷에 하루 24시간 연결되어 있는 것이 아니며, 연결할 때마다 동일한 IP 주소를 받게 되는 것도 아니다. 따라서 기계로 직접 전자우편을 받을 수가 없으며, 웹이나 ftp 서버로 설정하여 친구들이 쓸 수있도록 하는 것이 어렵다! 인터넷이 내 기계에 특정하고 항구적으로 연결될 수 있는 것이 아닌 한 특정한 IP 주소를 가질 수 없는 것이다(기억할 것 - 다른 사람이 인터넷 서비스 업체에게 접속하면서 내가 썼던 IP 주소를 받아서 쓰게 될 것이다.).

만약 WWW을 설정한다면(아니면 다른 서버든), 다른 사용자들이 당신 기계가 연결되어 있고 현재 활동중인 IP 주소를 갖고 있음을 알고 있지 않는 한 있는지조차 알 수 없게 된다. 다른 사람들이 이 정보를 아는 방법은 여러가지가 있다... 전화해서 알려줄 수 있고, 전자 우편으로 알려줄 수도 있고, 인터넷 서비스 업체가 준 계정에다 ”.plan”파일을 끼워넣어서 (이 경우 인터넷 서비스 업체가 쉘과 finger 사용을 허락해 줬어야 한다) 알려줄 수도 있다.

현재로서, 대부분의 사용자에게 이건 문제가 아니다 - 대부분의 사람은 전자우편을 주고 받으면 되고 (인터넷 서비스 업체가 준 계정을 써서), 웹, ftp, 등등 인터넷에 있는 다른 서버로 나가서 연결하는 것을 원하기 때문이다. 밖에서 들어오는 연결을 내 서버에 만들어야 한다면, 정적인 IP 주소를 받아야 한다. 그럼綸®워크淄各만湧ps″ 遐®를 준 방법을 찾아보아야 한다...

21.1 전자우편의 설정

동적 IP 주소를 가지고 있더라도 sendmail을 정확히 설정해서 로컬적으로 만든 어떤 전자우편도 외부로 보낼 수 있다. sendmail의 설정은 난해하고 어렵다 - 따라서 이 문서에서 어떻게 하는지 말하려고 하지는 않겠다. 하지만, 인터넷 서비스 업체를 ”smart relay” 호스트로 지정해 놓도록 sendmail을 설정할 수 있다(`sendmail.cf`의 **DS** 선택사항). (sendmail에 관한 선택사항을 더 알고 싶으면 sendmail 문서를 읽어보면 된다 - sendmail에 따라오는 m4 설정도 본다. 여기에는 필요한 것에 대한 설명이 거의 반드시 있을 것이다.)

sendmail에 관한 좋은 책들도 있지만(특히 O'Reilly와 연합에서 나온 'bible'), 이것은 분명히 대부분의 사용자들을 압사시킬 것이다.

일단 sendmail을 설정했으면, PPP 연결이 이루어지자 마자 외부로 나갈 전자우편 큐에 있던 편지들을 sendmail이 처리하길 바랄 것이다. 그렇게 하려면 다음 명령을 준다.

```
sendmail -q &
```

/etc/ppp/ip-up 스크립트에(아래를 보라).

들어오는 전자우편은 동적 IP 주소에 있어서 문제다. 이것을 다루는 방법은 :-

- 모든 편지에다 인터넷 서비스 업체 전자우편 주소를 "reply to" 헤더로 붙여서 보내도록 전자우편 사용자 처리기를 설정한다.
  할 수 있다면, 보내는 주소도 인터넷 서비스 업체 쪽 주소가 되도록 설정해야 한다.
- popclient, fetchmail 프로그램을 써서 인터넷 서비스 업체에게서 온 전자우편을 관리하든가, 인터넷 서비스 업체가 IMAP을 쓸 경우, pine 같은 IMAP 사용가능한 사용자 처리기를 쓴다.

`/etc/ppp/ip-up` 스크립트에다 필요한 명령을 집어넣어 전화할 때 이 과정을 자동으로 처리하게 할 수 있다.(아래를 보라)

21.2 로컬의 네임 서버 설정

인터넷 서비스 업체 쪽에 설정해 놓은 도메인 네임 서버를 잘 쓰고 있다고 해도, ip-up 스크립트를 만들어서 로컬 캐쉬만 할 수 있는 (제2) 이름 서버도 설정할 수 있다. 로컬 (캐쉬만 하는) 네임 서버를 만들게 되면 긴 연결을 계속할 경우에 같은 장소에 자주 접속하게 되면 시간과 속도를 절약할 수 있게 된다.

캐쉬만 하는 네임 서버를 만드는 DNS 설정(인터넷 서비스 업체의 DNS를 가리키는 named.boot 파일에 "forwarders" 줄을 쓴다)은 비교적 간단하다. O'Reilly 책(DNS and Bind)에서 알아야 할 모든 것을 설명하고 있다.

DNS-HOWTO 역시 이용할 수 있다.

리눅스 PC를 써서 인터넷을 찾아볼 수 있는 작은 랜을 운영하고 있다면(예를 들어 IP 메스커레이드 기능을 쓸 경우), 로컬 네임 서버를 실행하는 것이 좋다(보내기 방향을 갖는). 연결이 이루어질 때 이것이 이름 풀기와 관련된 지연을 최소화해주기 때문이다.

네티켓의 한가지 요점 : 인터넷 서비스 업체의 도메인에다 캐쉬만 하는 2차 네임 서버를 쓸 때는 그 전에 인터넷 서비스 업체에게 허락을 받아야 한다. 정확하게 설정될 경우 DNS는 인터넷 서비스 업체에게 아무런 문제도 일으키지 않겠지만 잘못 설정할 경우 문제가 생길 수도 있다.

제 22 절 PPP를 써서 두 네트워크를 연결하기

기본적으로 리눅스 PC 하나와 PPP 서버를 연결하는 것이나 랜에 있는 기계에서 PPP를 써서 두 네트워크를 연결하는 것에는 아무런 차이가 없다. PPP가 **대등** 프로토콜이라는 점을 기억하자.

하지만 라우트가 어떻게 만들어지는지 **명백히** 알아야 할 필요는 있다. NET-2 howto와 리눅스 네트워크 관리자 지침(NAG:Linux Network Administrator Guide)을 읽어본다. "TCP/IP Network Administrator Administration"(O'Reilly와 연합에서 출판한 - ISBN 0-937175-82-X)는 가치있고 매우 도움이 될 것이다.

연결의 각 끝에서 IP 네트워크 주소의 하위 네트워크를 쓰려고 한다면, (초고인) Linux sub networking mini-howto도 도움이 될 것이다. 이것은 *Linux Sub networking mini-HOWTO* `<http://www.interweft.com.au/other/>`에서 구할 수 있다.

두 랜을 연결할 생각이라면 **반드시** 다른 IP 네트워크 주소(또는 같은 네트워크 주소의 하위 네트워크 번호)를 써야하고 정적인 IP 주소가 필요하다-IP 메스커레이드를 쓰거나-. IP 메스커레이드를 쓰려면, 설정하는 방법을 소개한 IP masquerade mini-howto를 보라.

22.1 IP 주소의 설정

PPP 인터페이스의 각 끝에서 사용하게 될 IP 주소는 다른 쪽 네트워크의 네트워크 관리자와 함께 고친다. 정적 IP를 쓰고 있을 경우, 아마 정해진 전화번호로 접속하라고 할 것이다.

이제 적당한 /etc/ppp/options[.ttyXX] 파일을 편집한다 - 내쪽에서 이 연결을 위해 정해진 모뎀과 포트를 쓰는 것이 좋다.아마 /etc/ppp/options파일을 바꿔야 할 것이다 - 그리고 다른 연결용으로 적당한 options.ttyXX 파일을 만들어야 한다.

앞서 보여준 것처럼 적당한 선택사항 파일에다가 PPP 연결의 내쪽 끝에다 정확히 정적 IP 주소로 IP 주소를 적어주어야 한다.

22.2 라우트의 설정

내 로컬 네트워크에서 패킷이 PPP 연결이 만들어진 인터페이스 사이를 라우트할 수 있도록 맞춰주어야만 한다. 여기에는 두 단계의 과정이 있다.

무엇보다도 먼저, PPP 네트워크를 실행하는 기계로부터 연결의 원격 끝의 네트워크로 라우트를 만들어야 한다. 네트워크가 인터넷으로 서버 된다면, 연결의 내쪽 끝에서 pppd에다 선택사항으로 'default route'를 써서 pppd 자체에 기본 라우트를 고치면 된다.

하지만, 두개의 랜을 연결하는 것 뿐이라면, 연결 사이에 접근할 수 있는 각 네트워크에 대해 특정한 네트워크 라우트를 덧붙여 줘야 한다. /etc/ppp/ip-up 스크립트에서 각 네트워크에 대해 'route' 명령을 써서 해준다. (하는 법에 대한 소개는 연결된 후에.. 장을 본다.)

두번째 해야할 것은 다른 컴퓨터에게 내 컴퓨터가 ppp 연결의 먼쪽 끝에서 실제로 'gateway'라고 알려주는 것이다.

물론, 연결한 반대쪽 끝의 네트워크 관리자가 마찬가지로 이걸 다 해주어야 한다. 하지만 관리자가 내쪽의 특정한 네트워크로 라우트 패킷을 보낼 경우에는 기본 라우트가 아니라 **특정 네트워크 라우트**이 필요하다(먼쪽의 랜과 연결이 접속을 거쳐 내 쪽을 통해 인터넷을 이용하려는 것이 아니라면).

22.3 네트워크 보안

PPP를 써서 인터넷에다 랜을 연결하는 경우라면 - 또는 단순히 "외부" 랜에 연결하는 경우라면, 보안 문제에 대해 생각해 봐야 한다. 방호벽을 설정하는 것을 고려해 보도록 강력히 권한다!!!

외부 랜이나 인터넷에 이런 식으로 연결하기 **전에** 당신 장소의 랜 관리자에게 반드시 말해주어야 한다. 그렇게 못했다면 진짜 심각한 문제에 부딪쳤을 때 아무런 대응도 못하게 될 수도 있다.

제 23 절 연결된 후에 - /etc/ppp/ip-up 스크립트

일단 PPP 연결이 구축되면 pppd는 /etc/ppp/ip-up 파일을 찾는다. 이 스크립트가 있고 실행가능하면 PPP 대몬은 이 스크립트를 실행시킨다. 이것으로 어떤 필수적인 특별 라우트 명령이나 PPP 연결이 활동할 때마다 일어나길 원하는 모든 동작을 자동화할 수 있다.

이것은 단지 쉘 스크립트이며 쉘 스크립트로 할 수 있는 것은 뭐든지 할 수 있다(가상적으로 하고 싶은 것 다)

예를 들어 메일 큐에서 기다리던 밖으로 보낼 편지를 바로 보내게 sendmail을 조정할 수 있다.

비슷하게, 인터넷 서비스 업체 쪽에서 대기 중이던 전자우편(pop를 쓸 때)을 받기 위해 ip-up에다 명령을 넣을 수 있다.

/etc/ppp/ip-up에는 제한이 있다:-

- 이것은 보안을 보장할 수 있도록 심사숙고하여 제한된 환경에서 실행된다. 이것은 실행파일 등에 완전한 경로명을 주어야 한다는 것을 뜻한다.
- 기술적으로 볼 때, /etc/ppp/ip-up은 스크립트가 아니라 프로그램이다. 즉 직접 실행가능하다는 뜻이다 - 그리고 그래서 첫 행에 표준 파일 마법(#!/bin/bash)이 필요하고 루트가 읽기, 실행하기가 가능해야 한다.

23.1 특수 라우트

두 랜을 연결하고 있다면, '외부' 랜에 대해 특정한 라우트를 설정해 주어야 한다. 이것은 /etc/ppp/ip-up를 쓰면 쉽다. 기계가 다중 PPP를 쓰고 있을 경우에만 어렵게 된다.

이는 /etc/ppp/ip-up가 모든 ppp 연결이 될 때 실행되기 때문이다. 따라서 특정한 연결이 이루어지는 데 대해- 어떤 다른 연결이 이루어질 때 아니고- 정확한 라우트 명령이 조심스럽게 실행할 필요가 있다!

23.2 메일 큐 다루기

두개의 랜이 연결될 때, 각 끝에 쌓여있던 전자우편이 *쏟아지기*를 바랄 것이다 -각자 목적지에 보내질 것-. 이것은 적당한 sendmail 명령을 덧붙이면 된다.

pppd 가 이를 완료하는 스크립트 안으로 들어가는 적당한 매개변수에 배쉬 'case' 구문을 사용한다. 내가 우리 광역네트워크 연결과 나의 이더넷에 대한 연결을 다루는 /etc/ppp/ip-up 스크립트가 그런 예이다.(같은 ppp 서버를 다룬다.)

23.3 /etc/ppp/ip-up 예제 스크립트

이 예제는 다양한 경우의 사용법을 보여준다.

```
#!/bin/bash
#
# pppd에 필요한 것과 같은 경우에 보내는 라우트를 다루는 스크립트
# 이렇게 다루는 것이 필요한 초보자를 위한 연결
#
# ppp 연결이 되었을 때, 이 스크립트는 다음의 변수에 따라 호출된다.
#       $1      pppd에서 쓰는 인터페이스 이름(예를 들어 pppd)
#       $2      tty 장치 이름
#       $3      tty 장치 속도
#       $4      인터페이스에 대한 로컬 IP 주소
#       $5      원격 IP 주소
#       $6      pppd에 대해 'ipparam'선택사항으로 정의되는 변수
case "$5" in
# 뉴맨 캠퍼스 서버에 대한 라우트를 다룬다.
        202.12.126.1)
                /sbin/route add -net 202.12.126.0 gw 202.12.126.1
# 메일 큐를 쏟아내서 대기 중인 전자우편을 받아온다.
                /usr/sbin/sendmail -q &
                ;;
        139.130.177.2)
# 인터넷 연결 연결되었을 때, 아직 연결되어 있지 않다면 시간 서버를
# 시작하고 세계 시간과 일치시킨다.
                if [ ! -f /var/lock/subsys/xntpd ]; then
```

```
                        /etc/rc.d/init.d/xntpd.init start &
                fi
                ;;
# (아직 실행중이지 않다면) 뉴스 서버를 시작한다.
                if [ ! -f /var/lock/subsys/news ]; then
                        /etc/rc.d/init.d/news start &
                fi
                ;;
        203.18.8.104)
# 연결되자 마자 집에 기계로 전자우편을 내려 보낸다.
# 나의 이더넷이 IP 메스커레이드와 proxyarp 라우팅으로 운영되므로
# 라우팅은 필요하지 않다.
                /usr/sbin/sendmail -q &
                ;;
        *)
esac
exit 0
```

우리 뉴맨 캠퍼스로 ppp연결과 이 스크립트를 만든 결과로, 다음의 라우팅 테이블 내용을 갖고 끝내게 된다(이 기계는 일반적인 전화걸기 PPP 서버이고, 우리의 인터넷 연결을 다룬다). 각 내용이 무엇인지 설명하는데 돕기 위해서 출력에다 주석을 깔아놓았다.:-

```
[root@kepler /root]# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
# 상대방의 인터넷 게이트웨이를 향한 호스트 라우트이다.
139.130.177.2   *               255.255.255.255 UH    1500   0      134 ppp4
# 우리 뉴맨 캠퍼스 서버를 향한 호스트 라우트이다.
202.12.126.1    *               255.255.255.255 UH    1500   0       82 ppp5
# 내 집 이더넷을 향한 호스트 라우트이다.
203.18.8.104    *               255.255.255.255 UH    1500   0       74 ppp3
# 두개의 일반적인 전화걸기 PPP 회선이다.
203.18.8.64     *               255.255.255.255 UH    552    0        0 ppp2
203.18.8.62     *               255.255.255.255 UH    552    0        1 ppp1
# 뉴맨 캠퍼스 랜에 대한 특정 네트워크 라우트이다.
202.12.126.0    202.12.126.1    255.255.255.0   UG    1500   0        0 ppp5
# 로컬 이더넷에 대한 라우트이다.(두개의 인접한 C 그룹 상위네트워크)
203.18.8.0      *               255.255.254.0   U     1500   0     1683 eth0
# 귀환 장치를 향한 라우트이다.
127.0.0.0       *               255.0.0.0       U     3584   0      483 lo
# 인터넷을 향한 기본값 라우트이다.
default         139.130.177.2   *               UG    1500   0     3633 ppp4
```

23.4 전자우편 다루기

앞 장은 외부로 나가는 전자우편을 다루는 법을 보여준다 - 단순히 연결되었을 때 메일 큐를 쏟아내는 것이다.

광역네트워크 연결을 실행시키는 중이라면,상대방 랜의 네트워크 관리자와 함께 같은 것을 정확하게 맞출 수 있을 것이다. 예를 들어 광역네트워크 연결의 뉴맨 캠퍼스 쪽 끝에서는 , /etc/ppp/ip-up 스크립트는 다음과 같다:-

```
#!/bin/bash
#
# pppd에 필수적일 때 보내는 라우트를 다루는 스크립트 헨드랜드에 대한
# 연결만이 이 연결을 요구한다.
#
# 연결되었을 때 스크립트는 다음 변수에 따라 호출된다.
#       $1      pppd가 쓰는 인터페이스 이름(예를들어 ppp3)
#       $2      tty 장치 이름
#       $3      tty 장치 속도
#       $4      인터페이스에 대한 로컬 IP 주소
#       $5      상대방 IP 주소
#       $6      pppd에 'ipparam' 선택사항으로 정의한 매개변수
case "$5" in
        203.18.8.4)
                /usr/sbin/sendmail -q
                ;;
        *)
esac
exit 0
```

그럼 동적 IP PPP 연결만 인터넷 서비스 업체에서 제공 하고 있을 경우, 인터넷 서비스 업체 서버의 계정에서 전자우편을 받아올 필요가 있다. 보통 이것은 POP(우체국 프로토콜:Post Office Protocol)를 써서 하게 된다. 이 과정은 'popclient' 프로그램을 쓰면 다룰 수 있다 - 또한 ip-up 스크립트는 이 과정 역시 자동화할 수 있다!

간단하게 popclient의 적당한 호출을 포함하는 /etc/ppp/ip-up 스크립트를 만들면 된다. 레드햇 리눅스를 쓰는 내 랩탑(여행할 때 들고 다닌다)의 경우는 이렇다.

```
popclient -3 -c -u hartr -p <password> kepler.hedland.edu.au |formail -s procmail
```

뉴스 등등에 대해서도 slurp이나 같은 기능의 어떤 것도 쓸 수 있다. 기억할 것은 ip-up 스크립트가 배쉬 스크립트일 뿐이며, 따라서 적당한 PPP 연결이 이루어질 때마다 이루어질 필요가 있는 어떤 기능도 자동화하는데 사용될 수 있다는 것이다.

제 24 절 /etc/ppp/ip-down 의 사용

연결이 끝난 다음에 실행시킬 스크립트를 만들 수 있다. 이는 /etc/ppp/ip-down에 저장된다. 이것은 /etc/ppp/ip-up 스크립트에 병행해서 특별한 어떤 것을 그만두는 데 쓸 수 있다.

제 25 절 랜에서 보내는 라우트

랜에 연결했지만 개인적인 리눅스 기계에서 PPP를 쓰고 싶다면, 내 기계에서 랜까지 (이더넷 인터페이스를 통해) 받아야 라우트 패킷과 상대방 서버에서 그 너머로 가는 라우트 패킷에 대해 몇가지 명령을 주어야 한다.

이 장에서는 라우트에 대해 당신을 교육하려고 하는 것이 아니다 - 여기서는 (정적인) 라우트의 간단하고도 특별한 경우를 다룰 뿐이다.

라우트가 낯설다면 리눅스 네트워크 관라자 지침(NAG)를 읽도록 강력히 권하는 바다. 또한, O'Reilly 책 "TCP/IP Network Administration"에는 이 주제가 아주 이해하기 쉽게 다뤄져 있다.

정적 라우트의 기본 법칙은 네트워크 주소의 대부분을 가리키는 것이 기본값 라우트가라는 점이다. 다른 네트워크에 대해서는 라우팅 테이블에 특정한 라우트를 넣어줘야 한다.

여기서 다루려고 하는 이 상황은 리눅스 기계가 인터넷에 연결되어 있지 않은 랜 상에 있고 - 랜에 여전히 연결되어 있는 조건에서 개인적인 사용을 위해 인터넷에 전화해 들어가려는 상황이다.

무엇보다도 먼저, 이더넷 라우트가 랜 사이에서 사용가능한 특별한 네트워크 주소로 설정되어 있는지 - 기본값 라우트로는 설정되어 있지 않고 - 확인해야 한다!

route 명령을 줘서 라우트를 점검해 보면, 다음과 같은 것을 볼 수 있다:-

```
[root@hwin /root]# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
loopback        *               255.255.255.0   U     1936   0        50 lo
10.0.0.0        *               255.255.255.0   U     1436   0       565eth0
```

만약 이더넷 인터페이스(eth0)가 기본값 라우트를 가리키고 있다면, (eth0 행에 있는 첫번째 칸에 "default"라고 찍힌다.) 이더넷 초기화 스크립트를 고쳐서 기본 라우트가 아니라 특정한 네트워크 주소를 가리키도록 해야 한다(NET2 HOWTO와 NAG를 참고할 것).

pppd가 아래처럼 기본 라우트를 설정할 수 있게 한다.:-

```
[root@hwin /root]# route -n
Kernel routing table

Destination     Gateway         Genmask         Flags MSS    Window Use Iface
10.144.153.51   *               255.255.255.255 UH    488    0         0 ppp0
127.0.0.0       *               255.255.255.0   U     1936   0        50 lo
10.1.0.0        *               255.255.255.0   U     1436   0       569 eth0
default         10.144.153.51   *               UG    488    0         3 ppp0
```

볼 수 있는 것처럼, ppp0을 통해 PPP 서버(10.144.153.51)에 대해 호스트 라우트를 갖게 되었고, PPP 서버를 게이트 웨이로 쓰는 기본 네트워크 라우트를 갖게 되었다.

설정해야 하는 것이 이것보다 더 복잡하다면 - 이미 언급했던 라우트 관련 문서들을 읽고 가까운 곳의 전문가에게 문의해야 한다.

랜이 이미 라우터를 갖고 있다면, 사이트에서 사용가능한 보다 넓은 네트워크가 구축된 게이트웨이를 이미 갖고 있는 것이다. 역시 기본 라우트를 PPP 인터페이스를 가리키게 하고 - 제공하고 있는 네트워크를 가리키는 다른 특정한 라우트를 만들어 준다.

25.1 보안에 대한 주의

기존의 랜 상에 있는 리눅스 기계를 인터넷에 연결할 때는, 당신은 함축적으로 전체 랜을 인터넷에 공개하는 셈이다 - 크래커는 인터넷에 있다. 이렇게 하기 전에, 강력하게 추천하는 바, 네트워크 관리자와 사이트 보안 관리자에게 문의해야 한다. 만일 인터넷에 대한 당신의 PPP 연결 때문에 사이트가 공격당했다면, 최소한 동료 사용자들, 네트워크와 시스템 관리자이 무척 화를 낼 것이다. 아니면 더 심각한 문제에 직면하게 될 것이다.

랜을 인터넷에 연결하기 전에, 동적 연결일지라도 보안 장치에 대해 고려해보아야 한다 - 그러니 O'Reilly의 "Building Internet Firewalls"를 빨리 참고해라!

제 26 절 PPP 서버의 설정

이미 언급한 대로, 이것을 하는데는 여러가지 방법이 있다. 여기 있는 방법은 내가 쓰는 방법이다(Cyclades 다중 포트 직렬 카드와 몇개 전화회선의 회전식 전화를 쓴다).

이 방법이 마음에 들지 않는다면, 하고 싶은 데로 하면 된다. 하지만, 이 하우투에다 훗날 더 많은 방법을 넣고 싶다. 그러니까 설명과 방법을 보내줬으면 한다!

주의할 것은, 이 장이 리눅스를 PPP 서버로 설정하는 방법에만 관여한다는 점이다. 나는 특별한 단말기 서버나 그 비슷한 것을 설정하는 정보를 포함할 생각이 전혀 없다.

또한, shadow 비밀번호를 경험하지 못했다(언젠가 하게 될 테지만). 현재 여기에 있는 정보들은, 따라서, shadow 관련해서 요구되는 것은 전혀 포함할 수 없었다.

26.1 커널 컴파일

커널 컴파일과 커널 버전 대 pppd에 대해서 앞에 한 설명은 그대로 적용된다. 이 장에서는 이 문서의 앞 부분을 이미 읽은 것으로 간주한다!

PPP 서버에 대해서는, 커널에다 IP forwarding을 **반드시** 포함해야 한다. 다른 능력을 포함하고 싶으면 그렇게 한다(IP fire walls, accounting 등등).

만약 다중-포트 직렬 카드를 쓰고 있다면, 커널 도구에 필요한 드라이버를 명확히 포함해야만 한다.

26.2 서버 시스템에 대한 개관

우리는 동일한 이름/비밀번호 쌍을 쓰는 전화걸기 PPP(와 SLIP) 계정과 쉘 계정을 제공한다. 이렇게 하면 (우리로서는) 한 사용자가 하나의 계정만 가지면 되고 모든 연결 가능한 형태에 대해 이것을 쓸 수 있는 잇점이 있다.

우리는 교육 기관이기 때문에, 접속하는 간부나 학생들에게 돈을 받을 필요가 없으며, 계정 만들기나 요율 문제 등에 대해 걱정할 필요가 없다.

사이트와 인터넷 사이에 방호벽을 운영하고 있는데, 전화 회선이 우리 (인터넷) 방호벽 안에 있기 때문에 몇몇 사용자 접근을 제한한다(참으로 명백한 이유 때문에, 우리의 다른 내부 방호벽에 대한 세부사항은 여기에 쓸 수 없으며, 어떤 경우에도 적절하지 않다.)

우리 사이트에 PPP 연결을 구축하려는 사용자가 거치는 과정은(물론 유효한 계정을 가지고 있을 경우에):-

- 라우트식 전화에 전화한다(이건 모뎀 은행에 연결되는 전화번호 하나다 - 놀고 있는 첫번째 전화가 쓰이게 된다.)
- 유효한 사용자 이름과 비밀번호 쌍으로 접속한다.
- 쉘 명령행에서. ppp 명령을 주면 서버에서 PPP를 시작하게 한다.
- 자기쪽 PC에서 PPP를 시작한다(윈도우즈든, 도스든, 리눅스든, MAC OS, 어떤 것이든지 - 그건 그쪽 문제다).

서버는 각각의 전화해 들어온 포트에 대해 IP 번호를 동적으로 나눠주기 위해 개별적인 `/etc/ppp/options.ttyXX` 파일을 쓴다. 이것을 만들면 라우트나 게이트를 만들 필요가 없어진다.

사용자 쪽에서 끊게 되면, pppd는 이것을 알아내고 모뎀에게 끊도록 지시하며, 동시에 PPP 연결을 끊어 준다.

26.3 프로그램을 함께 받기

다음의 프로그램이 필요하다:-

- 리눅스, 필요한 선택사항을 포함해서 정확히 컴파일된 것
- 커널에 대해 적당한 버전의 pppd

- 모뎀 연결을 지능적으로 다루는 'getty' 프로그램
  우리는 getty_ps2.0.7h를 쓰지만, mgetty도 크게 고려하고 있다. mgetty가 pap/chap(pap는 윈도우즈95의 표준이다)를 쓰는 전화를 가려내고 pppd를 자동적으로 불러내는 것을 알고 있지만, 여전히 탐구하는 중이다.

- 전화 건 사용자들이 접근할 수 있는 작동 중인 로컬 네임 서버(DNS).
  가능하다면 독자적인 DNS를 진짜 실행하고 있어야 ...

26.4 표준 (쉘 접근) 전화걸기 설정

PPP 서버를 설정하기 전에, 리눅스 기계는 전화걸기 접근을 다룰 수 있는 능력이 있어야 한다.

이 어쩔거나에서는 이것을 설정하는 법을 다루지는 않는다. 이에 대한 정보는 선택한 getty의 문서와 serial HOWTO를 보라.

26.5 PPP 선택사항 파일의 설정

모든 전화걸기 포트에 대해 보편적인 선택사항은 /etc/ppp/options 파일에서 설정해야 한다. 우리가 쓰는 선택사항은:-

```
asyncmap 0
netmask 255.255.254.0
proxyarp
lock
crtscts
modem
```

주의 - 어떤 (명백한) 라우트도 쓰지 않는다 - 특별히 기본 라우트 선택사항은 없다. 그렇게 하는 이유는 (PPP 서버로서)요구되는 것이라고는 ppp 클라이언트 측**으로부터** 패킷을 받아 랜/인터넷으로 라우트시키고 클라이언트측**에게** 랜/인터넷에서 패킷을 받아 라우트시키는 것 뿐이기 때문이다.

여기에 필요한 것이라고는 클라이언트 측에 대한 호스트 라우트와 pppd에 대한 'proxyarp' 선택사항을 쓰는 것이다.

기본적으로 'proxyarp' 선택사항은 'PPP 클라이언트 쪽에 보내는 모든 패킷을 나한테 보내라'라고 하는 PPP 서버의 ARP 테이블에 있는 프락시 ARP 내용을 설정한다(놀랍게도). 이것이 하나의 PPP 클라이언트에 대해 라우트를 설정하는 가장 쉬운 방법이다 - 하지만 두개의 랜을 라우트하는 중이라면 쓸 수 없다. - 프락시 ARP를 쓸 수 없는 적당한 네트워크 라우트를 추가해야만 한다.

아마 당신은 전화 거는 사용자들한테 동적 IP 주소 할당을 제공하기 원하는게 거의 확실할 것이다. 이것은 각각의 전화받기 포트에다 IP 번호를 할당하면 할 수 있다. 이제 각각의 전화받기 포트에다 /etc/ppp/options.ttyXX를 만든다.

여기에다, 간단히 로컬(서버)IP 주소를 넣어주고, 각가의 포트에서 쓸 IP 주소를 넣어주면 된다. 예를 들어

```
kepler:slip01
```

특별히, 이 파일에다 유효한 호스트 이름을 쓸 수 있다는 것에 주의한다(나는 내가 내 네트워크에 있는 기계와 장치의 IP 주소를 기억하는데 한계가 있음을 알고 있다 - 이름이 훨씬 더 의미있다)!

26.6 사용자들이 (잘) 실행할 수 있도록 pppd를 설정하기

ppp연결 시작이 커널 장치(네트워크 인터페이스)을 설정하고 커널 라우팅 테이블을 잘 조작할 것을 요구하기 때문에 특별한 특권이 요구된다 - 사실 완전한 루트 권한.

다행히도, pppd는 일반사용자가 루트 특권으로 쓸 수 있도록 설정해도 '안전'하도록 설계되었다. 그러므로 이렇게 하면 된다.

```
chmod u+s /usr/sbin/pppd
```

파일을 열거해보면, 이런 식으로 보인다.

```
-rwsr-xr-x   1 root     root        74224 Apr 28 07:17 /usr/sbin/pppd
```

이렇게 하지 않으면, 사용자는 ppp 연결을 설정할 수 없다.

26.7 pppd 용의 별명 설정

전화거는 PPP 사용자들이 하는 일을 간편하게 해주기 위해서, 우리는 사용자들이 로긴했을 때 간단한 명령으로 서버 쪽에서 간단히 ppp를 시작하도록 범용 별명(/etc/bashrc안에다)을 만들었다.

다음과 같다.

```
alias ppp="exec /usr/sbin/pppd -detach"
```

이것이 하는 일은

- exec : 실행 중인 프로그램(이 경우 쉘 자체)과 실행할 프로그램을 바꿔라.
- pppd -detach : pppd를 시작하고 배경작업으로 분기하지 마라. pppd가 있는 동안 다른 과정이 없음을 보장해 준다.

사용자가 이렇게 로긴했을 때 'w'의 출력으로 아래와 같이 보인다.

```
  6:24pm  up 3 days,  7:00,  4 users,  load average: 0.05, 0.03, 0.00
User     tty       login@  idle   JCPU   PCPU  what
hartr    ttyC0     3:05am  9:14                -
```

그리고 이게 다다... 이게 간단하고도, 기본적인 PPP 서버 시스템이라고 말할 수 있다!

제 27 절 널 모뎀(직접 직렬) 연결 사이에서 PPP 쓰기

이것은 매우 간단하다 - 사이에 모뎀이 없으니 아주 간단하다.

무엇보다도 기계 중 하나를 '서버'로 고르고, '클라이언트 측'에서 직렬 포트를 실행시키기 위해 미니콤을 쓸 수 있는 연결능력을 갖고 있는지 테스트 할 수 있도록 직렬 포트에 대해 getty를 설정한다.

이 기능을 갖추었으면, 전화걸기 연결에서 처럼 사용자 이름/비밀번호를 서서 유효화 시키는 연결을 만들 작정이 아니라면 getty를 없앤다. 기계 모두를 '내 손안에' 두고 있는 것이라면, 이런 걸 원하지 않으리라고 추측했다.

이제, 서버에서, getty를 없애고 'setserial'을 써서 양쪽 기계에 직렬 포트가 정확하게 설정되었는지 확인해야 한다.

이제 해야 할 것은 양쪽 기계에서 pppd를 시작하는 것이다. 양쪽 기계에서 연결에 대해 /dev/ttyS334를 사용한다고 가정하겠다. 그러면 양쪽 기계에서 다음 명령을 실행한다:-

```
pppd -detach crtscts lock <local IP>:<remote IP> /dev/ttyS3 38400 &
```

이것이 연결을 만들어 준다 - 하지만 아직 아무 라우트도 정의하지 않았다. 연결을 각 기계에다 핑해 봄으로써 시험해 볼 수 있다. 이게 작동하면, 한쪽의 pppd 과정을 죽여서 연결을 끊는다.

필요한 라우트는 물론 무엇을 하려고 하는가에 달려있다. 일반적으로, 기계 하나가 이더넷(과 그너머)에 연결되어 있으며 따라서 라우트가 정확히 PPP 서버와 클라이언트측에 모두 요구된다.

따라서 이더넷이 장치된 기계에서 pppd 명령은 이렇게 될 것이다.

```
pppd -detach crtscts lock proxyarp <local IP>:<remote IP> /dev/ttyS3 38400 &
```

다른 기계에서는 이렇게 된다.

```
pppd -detach crtscts lock defaultroute <local IP>:<remote IP> /dev/ttyS3 38400 &
```

두개의 네트워크를 (직렬 연결로!) 연결할 경우나 더 복잡한 라우트 요구를 갖고 있다면, 이 문서 앞에서 이미 언급한 것과 완전히 같은 방식으로 /etc/ppp/ip-up을 쓸 수 있다.

로버트 하트
포트 헤들랜드, 서부 오스트레일리아
멜버른, 빅토리아, 오스트레일리아 1996 8월/10월 1997 1월/ 3월