SMB HOWTO

저자: David Wood, dwood@plugged.net.au v1.0, 10 August 1996

이 창한, chlee@physics.soongsil.ac.kr

이 문서는 SMB HOWTO 이다. NetBIOS 또는 LanManager라고도 불리는 Session Message Block (SMB) 프로토콜을 리눅스에서 사용하는 방법을 설명한다.

목 차

제 1 절 소개 1

제 2 절 좀더 자세한 정보를 원하면 2

제 3 절 설치 2

제 4 절 데몬 실행시키기 3

제 5 절 일반적인 설정(/etc/smb.conf) 4

제 6 절 리눅스 드라이브를 M$와 공유시키기 5

제 7 절 M$의 드라이브를 리눅스와 공유시키기 5

제 8 절 리눅스 프린터와 M$ 공유시키기 7

제 9 절 M$의 프린터를 리눅스와 공유하기 8

제 10 절 저작권 14

제 11 절 감사의 글 15

제 1 절 소개

이 문서는 SMB HOWTO 이다. NetBIOS 또는 LanManager라고도 불리는 Session Message Block (SMB) 프로토콜을 리눅스에서 사용하는 방법을 설명한다.

이 문서는 David Wood (dwood@plugged.net.au) 가 관리한다. 다음 판에 포함되기를 원하는 추가, 수정 또는 정정에 대한 메일을 언제나 환영한다.

SMB 프로토콜은 M$ 3.11, 95, NT(앞으로 M$로 총칭하겠다)에서 디스크와 프린터를 공유하는데 사용된다. Andrew Tridgell이 만든 삼바 도구들을 이용해서 리눅스(유닉스)와 윈도우 간에 디스크, 프린터 공유가 가능하다.

삼바로 할 수 있는 일은 아래의 네 가지가 있다.

1. 리눅스 파티션과 M$와의 공유.
2. M$ 파티션과 리눅스와의 공유.
3. 리눅스의 프린터와 M$와의 공유.
4. M$의 프린터와 리눅스와의 공유.

이 네 가지를 앞으로 설명하겠다.

제 2 절 좀더 자세한 정보를 원하면

삼바는 매우 복잡하고도 완전한 패키지이다. 삼바의 모든 것을 설명한다는 것은 참으로 어려운 일이다. 때문에 이 문서에서는 삼바의 화일, 프린트 서비스를 리눅스에서 어떻게 구성하는지를 설명할 것이다.

더 많은 정보를 원하면 다음 문서들을 참고하기 바란다.

- 삼바에 관련된 문서들은 삼바 패키지에 포함되어 있다. 패키지를 얻을 수 있는 곳 : *ftp://nimbus.anu.edu.au/pub/tridge/samba/* `<ftp://nimbus.anu.edu.au/pub/tridge/samba/>`
- The linux Printing HOWTO.
- The Print2Win Mini-HOWTO.

제 3 절 설치

삼바의 가장 최근의 소스는 다음 사이트에서 구할 수 있다. *ftp://nimbus.anu.edu.au/pub/tridge/samba/* `<ftp://nimbus.anu.edu.au/pub/tridge/samba/>`

레드햇을 설치한다면 삼바패키지는 옵션이므로 선택을 해야만 한다. 다른 배포판들은 삼바 실행화일을 포함하고 있다.

아래의 두 데몬은 삼바를 사용하기 위해 꼭 필요한 것이다. 일반적으로 /usr/sbin 디렉토리 아래에 설치되어있고 시스템의 초기설정 스크립트나 inetd에 의해서 부팅시에 시작되어진다. 4(데몬 실행시키기)에서 스크립트의 예를 볼 수 있다.

```
smbd (SMB 데몬)
nmbd (클라이언트를 위해 NetBIOS nameserver를 지원하다 )
```

삼바 실행화일들은 일반적으로 /usr/bin 아래에 설치되지만 바꿀 수 있다.

```
smbclient       (유닉스를 위한 삼바 클라이언트)
smbprint        (삼바 호스트의 프린터에 프린트를 하기 위한 스크립트)
smbprint.sysv   (위 smbprint와 같지만 SVR4 유닉스를 위한 것이다)
smbstatus       (현재의 삼바 연결 상태를 보여준다)
smbrun          (삼바 호스트의 응용프로그램 실행을 용이하게 하는 스크립트)
```

추가적으로 이 문서에 포함되어있는 'print' 스크립트는 smbprint 스크립트 작업을 준비하는데 유용하다.

삼바 패키지는 설치하기가 간단하다. 위에서 언급한 사이트에서 삼바 소스를 얻은 다음에 소스에 포함된 README 화일을 읽어보아라. 소스에 doc/INSTALL.txt화일이 있을 것이다. 이 화일에서 설치에 대한 자세한 정보를 얻을 수 있을 것이다.

인스톨이 되면 /usr/sbin 아래에 데몬, /usr/bin 아래에 실행화일이 설치된다. 매뉴얼 페이지는 /usr/local/man에 설치된다.

삼바 패키지를 설치할 때 Makefile에서 삼바 설정화일인 smb.conf 화일의 위치를 지정해야 한다. /etc 아래로 하는 것이 일반적이지만 원한다면 어떤 곳이든 가능하다. 여기에서는 삼바설정화일을 /etc/smb.conf로, 로그 화일의 위치는 log file = /var/log/samba-log.%m, 잠금 디렉토리는 lock directory = /var/lock/samba로 지정한 것으로 가정하고 설명할 것이다.

smb.conf 설정화일을 인스톨하자. 이미 만들어진 삼바 디렉토리로 이동하라. example/simple 디렉토리를 살펴보고 README 화일을 읽어 보아라. smb.conf 화일을 /etc 디렉토리에 복사한다. 이때 주의를 해야 한다. 만약 이미 시스템에 삼바가 설치되어 있다면 /etc 아래에 삼바설정 화일이 있을 텐데 그 화일로 시작하기를 권한다.

/etc 아래에 설정화일을 만드는 것을 원하지 않으면 다른 곳에 저장을 하고 /etc 아래에 다음과 같이 심볼릭 링크를 만들어라.

```
ln -s /path/to/smb.conf /etc/smb.conf
```

제 4 절 데몬 실행시키기

삼바의 두 데몬은 /usr/sbin/smbd 와 /usr/sbin/nmbd 이다. 삼바 데몬은 inetd 에 의해서나 독립된 과정으로 실행되어질 수 있다. 영구적인 화일 서버로 설치를 한다면, 삼바 데몬이 중단되면 다시 실행될 수 있도록 inetd 에 의해서 실행될 것이다. 삼바 서비스를 가끔씩 사용하거나 단지 시스템 관리에 이용하기 위해서라면 /etc/rc.d/init.d 스크립트에서 실행시키거나 직접 필요에 따라 수작업으로 실행시킬 수 있다.

inetd 에의해 데몬을 실행시키기 위해서는 아래 줄들을 /etc/inetd.conf에 추가시켜야 한다.

```
# SAMBA NetBIOS services (for PC file and print sharing)
netbios-ssn stream tcp nowait root /usr/sbin/smbd smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd
```

그 다음에 inetd 데몬을 다음과 같이 재실행 시킨다.

```
kill -HUP 1
```

시스템 초기 실행 스크립트에서 데몬을 실행시키기 위해서는 아래의 스크립트를 /etc/rc/d/init.d/smb 로 저장을 하고 주석에 나와있는데로 심볼릭 링크를 해야한다.

```
#!/bin/sh

#
# /etc/rc.d/init.d/smb - 삼바 서비스를 시작, 중지한다.
#
# 아래와 같이 이 화일을 심볼릭 링크를 시킨다.
# symlinks: /etc/rc.d/rc1.d/K35smb  (Kills SMB services on shutdown)
#           /etc/rc.d/rc3.d/S91smb  (Starts SMB services in multiuser mode)
#           /etc/rc.d/rc6.d/K35smb  (Kills SMB services on reboot)
#

# 함수 라이브러리 소스
. /etc/rc.d/init.d/functions

# 네트워크 설정 소스
. /etc/sysconfig/network
```

```
# 네트워크 가능상태를 점검한다
[ ${NETWORKING} = "no" ] && exit 0

# 서비스를 시작 또는 중지시킨다.
case "$1" in
  start)
    echo -n "Starting SMB services: "
    daemon smbd -D
    daemon nmbd -D
    echo
    touch /var/lock/subsys/smb
    ;;
  stop)
    echo -n "Shutting down SMB services: "
    killproc smbd
    killproc nmbd
    rm -f /var/lock/subsys/smb
    echo ""
    ;;
  *)
    echo "Usage: smb {start|stop}"
    exit 1
esac
```

제 5 절 일반적인 설정(/etc/smb.conf)

리눅스(또는 다른 유닉스 시스템)에서 삼바설정은 /etc/smb.conf 화일에 의해서 제어된다. 이 설정화일은 외부와 공유시킬 시스템 자원과 어떤 제약을 둘 것인지를 결정한다.

이 절에서 리눅스 드라이브와 프린터를 M$와 공유시키는 것을 설명할 것이다. 이번 절의 smb.conf 화일은 단지 소개를 위한 것으로서 아주 간단하다. 상세한 것들에 대해 벌써부터 걱정할 필요는 없다. 추후 절에서 중요한 개념들을 소개할 것이다.

각 영역은 [global], [homes], [printer], 등으로 말머리가 시작된다.

[global] 영역에서는 삼바가 모든 자원들의 공유를 정의하는데 사용하는 몇몇 변수들을 정의한다.

[homes] 영역에서는 사용자가 원격으로 리눅스의 사용자 홈 디렉토리로 접속하는 것을 허용한다. 단, 사용자는 자신의 홈 디렉토리 접속만을 허용한다. 즉, M$의 사용자가 M$로부터 리눅스의 공유 드라이브로 접속을 시도한다면, 오직 사용자 개인의 홈 디렉토리만이 접속이 가능하다. 이때 중요한 것은 사용자는 리눅스에 계정을 가지고 있어야만 한다.

아래의 smb.conf 예제 화일은 사용자가 로컬 컴퓨터의 개인 홈 디렉토리 접속과 임시 디렉토리 생성을 허락한다. M$ 사용자가 이들 공유를 볼 수 있기위해서는 리눅스가 로컬 네트워크 상에 있어야 한다. 그러면 사용자는 윈도우즈 화일 매니저나 탐색기의 네트워크 드라이브로 간단히 연결할 수 있다.

더 많은 자원들을 공유시키기 위한 추가적인 항목들을 다음 절에서 설명한다.

```
; /etc/smb.conf
;
; 이 화일을 수정한 뒤에 서비스를 꼭 재실행한다. 예:
; /etc/rc.d/init.d/smb stop
; /etc/rc.d/init.d/smb start

[global]
; guest 계정을 허용하려면 아래 주석 기호를 삭제한다.
; guest account = nobody
```

```
   log file = /var/log/samba-log.%m
   lock directory = /var/lock/samba
   share modes = yes

[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mode = 0750

[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes
```

제 6 절 리눅스 드라이브를 M$와 공유시키기

앞의 간단한 smb.conf에서와 같이 M$ 사용자에게 리눅스 드라이브를 공유시키기는 쉽다. 어쨌든, 삼바의 여러가지 자원을 관리하는 방법들을 몇가지 예를 들어 설명하겠다.

디렉토리를 공용으로 공유시키기 위해서는 앞의 smb.conf화일 [tmp] 영역을 복사해서 아래와 같이 바꾼 다음 화일에 추가한다.

```
[public]
   comment = Public Stuff
   path = /home/public
   public = yes
   writable = yes
   printable = yes
```

위의 디렉토리를 staff 그룹에 속한 사용자에게만 쓰기 권한을 주고 그외 일반 사용자에게는 읽기 권한 만을 주기위해서는 아래와 같이 수정한다.

```
[public]
   comment = Public Stuff
   path = /home/public
   public = yes
   writable = yes
   printable = no
   write list = @staff
```

드라이브 공유에 대한 또다른 방법들은 삼바 문서나 매뉴얼 페이지를 보기 바란다.

제 7 절 M$의 드라이브를 리눅스와 공유시키기

유닉스를 위한 삼바 클라이언트 프로그램은 삼바 소스에 포함되어 있다. 이 것은 코맨드 라인에서 ftp와 같은 인터페이스를 제공한다. 이것을 M$ 서버와 리눅스 클라이언트 간에 화일 전송 유틸리티로 사용할 수 있다.

원하는 호스트의 사용가능한 공유를 확인하기 위해서는 다음 명령을 실행해 보아라.

```
/usr/sbin/smbclient -L host
```

'host'는 살펴보기를 원하는 컴퓨터의 이름이다. 이 명령은 사용 가능한 '서비스'의 이름을 보여줄 것이다. 즉, 공유할 수 있는 드라이브나 프린터의 이름들을 보여준다. 삼바 서버가 암호 설정이 되어있다면 암호를 물어볼 것이다. 서버의 'guest' 계정이나 사용자 계정의 암호를 알고 있어야 한다.

예:

```
smbclient -L zimmerman
```

이 명령의 결과는 다음 처럼 나타날 것이다.

```
Server time is Sat Aug 10 15:58:27 1996
Timezone is UTC+10.0
Password:
Domain=[WORKGROUP] OS=[Windows NT 3.51] Server=[NT LAN Manager 3.51]

Server=[ZIMMERMAN] User=[] Workgroup=[WORKGROUP] Domain=[]

        Sharename      Type      Comment
        ---------      ----      -------
        ADMIN$         Disk      Remote Admin
        public         Disk      Public
        C$             Disk      Default share
        IPC$           IPC       Remote IPC
        OReilly        Printer   OReilly
        print$         Disk      Printer Drivers


이 컴퓨터는 다음과 같은 브라우즈(browse) 목록을 가지고 있다.

        Server               Comment
        ---------            -------
        HOPPER               Samba 1.9.15p8
        KERNIGAN             Samba 1.9.15p8
        LOVELACE             Samba 1.9.15p8
        RITCHIE              Samba 1.9.15p8
        ZIMMERMAN
```

이 브라우즈 목록은 연결된 네트웍 상에 공유된 자원을 가진 삼바 서버들을 보여준다.

클라이언트를 사용하기 위해서는 다음과 같이 한다.

```
/usr/sbin/smbclient service <password>
```

'service'는 컴퓨터와 공유명이다. 예를 들어 'zimmerman' 컴퓨터의 'public'으로 공유된 디렉토리를 사용하기 위해서는, service가 \\zimmerman\public 으로 된다. 그런데 쉘의 제약때문에 역슬래쉬 사용에 약간의 문제가 있다. 그래서 다음과 같이 해야 한다.

```
/usr/sbin/smbclient \\\\zimmerman\\public mypasswd
```

'mypasswd'는 자신의 암호이다.

성공하면 다음과 같은 smbclient 프롬프트가 나타난다.

```
Server time is Sat Aug 10 15:58:44 1996
Timezone is UTC+10.0
Domain=[WORKGROUP] OS=[Windows NT 3.51] Server=[NT LAN Manager 3.51]
smb: \>
```

'h'를 치면 smbclient 도움말을 볼 수 있다.

```
smb: \> h
ls          dir         lcd         cd          pwd
get         mget        put         mput        rename
more        mask        del         rm          mkdir
md          rmdir       rd          prompt      recurse
translate   lowercase   print       printmode   queue
cancel      stat        quit        q           exit
newer       archive     tar         blocksize   tarmode
setmode     help        ?           !
smb: \>
```

만약 ftp를 사용할 수 있는 사용자라면 어려움없이 사용할 수 있다.

제 8 절 리눅스 프린터와 M$ 공유시키기

리눅스의 프린터를 M$와 공유시키기 위해서는 프린터가 리눅스에서 작동되도록 설정이 되어있어야 한다는 것은 당연한 것이다. 리눅스에서 프린트가 가능하다면 삼바 공유를 설정하는 것은 간단한 일이다.

로컬 프린트를 설정하는 부분에 대해서는 Printing HOWTO 문서를 참고하기를 바란다.

이 문서의 저자는 M$ NT에 연결된 프린터를 사용하기 때문에 이 부분을 완전한 것으로 생각하지 말고, 단지 제안하는 정도로 생각하기 바란다. 만약 리눅스 프린터와 M$를 공유시키는 자세한 방법을 알고 있다면 메일 dwood@plugged.net.au 을 보내주기 바란다. 그것을 참고로해서 이 부분이 완벽하게 될 수 있다.

smb.conf에 프린트 설정을 추가한다.

```
[global]
   printing = bsd
   printcap name = /etc/printcap
   load printers = yes
   log file = /var/log/samba-log.%m
   lock directory = /var/lock/samba

[printers]
   comment = All Printers
   security = server
   path = /var/spool/lpd/lp
   browseable = no
   printable = yes
   public = yes
   writable = no
   create mode = 0700

[ljet]
   security = server
   path = /var/spool/lpd/lp
   printer name = lp
   writable = yes
```

```
public = yes
printable = yes
print command = lpr -r -h -P %p %s
```

프린터의 경로(이 경우엔 [ljet]의 path)를 /etc/printcap에 정의된 스풀디렉토리와 일치시키는 것을 잊지말아라.

주의: 삼바를 이용해 유닉스에 연결된 프린터와 M$ NT를 공유시키는 것에는 약간의 문제가 있다. 문제의 하나는 공유된 프린터를 NT가 적절하게 인식할 수 있느냐에 대한 것이다. 이 문제를 고치기 위해서는 삼바 소스에 포함되어 있는 docs/WinNT.txt 문서를 보기 바란다. 또다른 문제는 암호를 처리하는 문제이다. 위의 문서를 살펴보면 이 문제의 수정에 실패한 것을 이해하기 어렵게 써놓았다.

제 9 절 M$의 프린터를 리눅스와 공유하기

M$의 프린터를 공유하기 위해서는 아래의 작업을 꼭 해야만 한다.

a) /etc/printcap 화일에 로컬 디렉토리 구조(스풀디렉토리 등)와 정확히 일치하는 적당한 사항들이 있어야 한다.

b) /usr/bin/smbprint 화일이 있어야 한다. 이 화일은 삼바 소스에 같이 포함되어 있는데, 삼바 실행 화일만이 들어있는 배포본에는 빠져있을 수 있다. 이 화일을 약간 수정한 것이 아래에 설명되어 있다.

c) 만약 아스키 화일을 포스트스크립트 화일로 변환하기 위해서는 nenscript나 그와 같은 역할을 하는 것을 가지고 있어야만 한다. nenscript는 포스트스크립트 변환기이고 보통은 /usr/bin 아래에 설치된다.

d) 삼바로 프린팅할때 사용하기 편한 도구를 이용하기를 원할 것이다. 아스키 화일, 포스트스크립트 화일을 처리해주는, 또는 포스트스크립트 화일을 생성해주는 펄 스크립트가 아래에 주어져 있다.

M$ NT 호스트의 HP 5MP 프린터를 위한 /etc/printcap 화일의 설정이 아래에 주어져 있다. 각 항목들은 다음의 의미가 있다

```
cm - 주석(프린터 설명)
lp - 출력을 위한 장치명
sd - 로컬 컴퓨터의 프린터 스풀 디렉토리
af - the accounting file
mx - 최대 화일 크기(0은 제한이 없음을 뜻한다)
if - 입력 필터(스크립트)의 이름
```

더 많은 정보를 원하면 Printing HOWTO나 printcap의 매뉴얼을 보기 바란다

```
# /etc/printcap
#
# //zimmerman/oreilly via smbprint
#
lp:\
        :cm=HP 5MP Postscript OReilly on zimmerman:\
        :lp=/dev/lp1:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/spool/lpd/lp/acct:\
        :mx#0:\
        :if=/usr/bin/smbprint:
```

스풀 디렉토리와 accounting 디렉토리가 존재하고 쓰기가능 모드로 설정을 해주어야 한다. 또한 'if' 줄에 smbprint 스크립트(아래에 주어진)의 정확한 경로가 적혀있는지, 적절한 장치를 가리키고 있는지(/dev 아래의 특별 화일)를 확인해야 한다.

다음으로 smbprint 스크립트에대한 설명이다. 스크립트는 대개 /usr/bin 아래에 위치하고, 삼바를 만든사람으로 여겨지는 Andrew Tridgell에 의한 것이다. 이 스크립트는 삼바소스 배포본에는 포함되어있지만 실행화일 배포본에는 들어있지 않다. 그래서 여기에서 다시 만들어 볼 것이다.

이 스크립트를 주의깊게 살펴볼 것으로 여겨지는데, 이 스크립트를 그냥 그대로 사용하는 것보다는 자신의 환경에 알맞게 수정하는 것이 더 바람직하고 유용하다.

```
#!/bin/sh -x

# 이 스크립트는 유닉스 상에서 printcap 프린팅을 위한 입력 필터이다.
# 이것은 지정된 삼바 서버와 서비스에 화일을 프린트하기 위한 smbclient
# 프로그램을 이용한다.
# 예를 들어 printcap 설정을 아래처럼 할 수 있다
#
# smb:lp=/dev/null:sd=/usr/spool/smb:sh:if=/usr/local/samba/smbprint
#
# 위의 설정은 이 스크립트를 통해 프린트하는 "smb" 라는 유닉스 프린터를
# 만든다. 또한 적절한 권한과 소유권을 가진 /usr/spool/smb 스풀디렉토리를
# 만들어야 한다.

#
# 스크립트는 hamiltom@ecnz.co.nz (Michael Hamilton)에 의해서 서버, 서비스
# 그리고 암호를 /usr/var/spool/lpd/PRINTNAME/.config 화일에서 읽을 수
# 있도록 바꿔어졌다.
#
# 이것을 가능하게 하기 위해서는 /etc/printcap에 accounting 화일(af=...)이 꼭
# 포함되어야만 한다.
#
#   cdcolour:\
#       :cm=CD IBM Colorjet on 6th:\
#       :sd=/var/spool/lpd/cdcolour:\
#       :af=/var/spool/lpd/cdcolour/acct:\
#       :if=/usr/local/etc/smbprint:\
#       :mx=0:\
#       :lp=/dev/null:
#
# /usr/var/spool/lpd/PRINTNAME/.config 화일에 다음 사항들을 포함한다:
#   server=PC_SERVER
#   service=PR_SHARENAME
#   password="password"
#
# E.g.
#   server=PAULS_PC
#   service=CJET_371
#   password=""

#
# Debugging log file, 원한다면 /dev/null로 바꿔도 된다.
#
logfile=/tmp/smb-print.log
# logfile=/dev/null


#
# 필터의 마지막 파라미터는 accounting 화일 이름이다.
#
```

```
spool_dir=/var/spool/lpd/lp
config_file=$spool_dir/.config

# config 화일에서 다음 변수 설정을 읽어 들인다.
#   server
#   service
#   password
#   user
eval `cat $config_file`

#
# Some debugging help, change the >> to > if you want to same space.
#
echo "server $server, service $service" >> $logfile

(
# 주의: 프린트시에 자동으로 CR/LF 변환을 가능하게 하려면 `echo translate'를
# 추가한다.
  echo translate
        echo "print -"
        cat
) | /usr/bin/smbclient "\\\\$server\\$service" $password -U $user -N -P >> $logfile
```

대부분의 리눅스 배포판은 아스키 문서를 포스트스크립트로 변환해주는 nenscript를 가지고 있다. 다음 perl 스크립트는 smbprint를 통해 리눅스에서 프린트하는 것을 아주 쉽게 해준다.

```
Usage: print [-a|c|p] <filename>
       -a prints <filename> as ASCII
       -c prints <filename> formatted as source code
       -p prints <filename> as Postscript
        If no switch is given, print attempts to
        guess the file type and print appropriately.
```

아스키 화일을 프린트하기 위해서 smbprint를 이용할때 긴 줄은 잘리는 경우가 있다. 이 스크립트는 가능하다면 긴 줄이 단어 중간이 잘리는 대신에 공백에서 잘리도록 해준다.

nenscript는 소스코드에 형식을 준다. 장식적인 머릿말(날짜, 화일이름 등)을 추가하여 아스키 화일과 형식을 생성한다. 또한, 줄번호를 매긴다. 아래 예를 이용하여 다른 모양의 형식도 가능하다.

포스트스크립트 문서는 이미 적절한 형식을 가지고 있으므로 직접 출력이 된다.

```
#!/usr/bin/perl

# Script:   print
# Authors:  Brad Marshall, David Wood
#           Plugged In Communications
# Date:     960808
#
# Script to print to oreilly which is currently on zimmerman
# 목적: 여러가지 형식의 화일들을 인자로서 받아들여 삼바 프린트 스크립트에
# 파이프로 보내는데 적당하게 처리를 한다.
#
# 가능한 화일 형식:
#
# ASCII      - $line_length 보다 더 긴 글자들을 공백으로 줄 길이를 조정한다.
```

```
# Postscript - 아무런 처리도 하지 않는다.
# Code       - 적당하게(출력방향, 글자모양 등) 출력하기 위해서 nenscript를
#                                             이용하여 포스트스크립트 문서로 만든다.
#

# 아스키 텍스트 문서에서 한 줄당 최대 글자수.
$line_length = 76;

# 삼바 프린트 스크립트의 경로와 이름.
$print_prog = "/usr/bin/smbprint";

# nenscript의 경로와 이름(아스키-->포스트스크립트 변환기)
$nenscript = "/usr/bin/nenscript";

unless ( -f $print_prog ) {
        die "Can't find $print_prog!";
}
unless ( -f $nenscript ) {
        die "Can't find $nenscript!";
}

&ParseCmdLine(@ARGV);

# DBG
print "filetype is $filetype\n";

if ($filetype eq "ASCII") {
        &wrap($line_length);
} elsif ($filetype eq "code") {
        &codeformat;
} elsif ($filetype eq "ps") {
        &createarray;
} else {
        print "Sorry..no known file type.\n";
        exit 0;
}
# smbprint로 각 줄들을 연결(출력)시킨다.
open(PRINTER, "|$print_prog") || die "Can't open $print_prog: $!\n";
foreach $line (@newlines) {
        print PRINTER $line;
}
# 화일의 마지막 줄이 제대로 끝나지 않은 경우에 줄바꿈(라인피드)를 추가한다.
print PRINTER "\n";
close(PRINTER);
print "Completed\n";
exit 0;

# --------------------------------------------------- #
#        Everything below here is a subroutine        #
# --------------------------------------------------- #

sub ParseCmdLine {
        # 코맨드 라인을 분석해서 화일의 형식이 무엇인지 찾아낸다.

        # $arg와 $file 이 있다면 값을 얻는다.
```

```
        # and the filename
        if ($#_ < 0) {
                &usage;
        }
        # DBG
#       foreach $element (@_) {
#               print "*$element* \n";
#       }

        $arg = shift(@_);
        if ($arg =~ /\-./) {
                $cmd = $arg;
        # DBG
#       print "\$cmd found.\n";

                $file = shift(@_);
        } else {
                $file = $arg;
        }

        # 화일 형식을 정의한다.
        unless ($cmd) {
                # 아무런 인자가 없을 경우.

                if ($file =~ /\.ps$/) {
                        $filetype = "ps";
                } elsif ($file =~ /\.java$|\.c$|\.h$|\.pl$|\.sh$|\.csh$|\.m4$|\.inc$|\.html$|\.htm$/) {
                        $filetype = "code";
                } else {
                        $filetype = "ASCII";
                }

                # $file의 화일형식이 무엇인지 알아내서 $filetype을 넘겨준다.
        } else {
                # We have what type it is in $arg
                if ($cmd =~ /^-p$/) {
                        $filetype = "ps";
                } elsif ($cmd =~ /^-c$/) {
                        $filetype = "code";
                } elsif ($cmd =~ /^-a$/) {
                        $filetype = "ASCII"
                }
        }
}

sub usage {
        print "
Usage: print [-a|c|p] <filename>
       -a prints <filename> as ASCII
       -c prints <filename> formatted as source code
       -p prints <filename> as Postscript
        If no switch is given, print attempts to
        guess the file type and print appropriately.\n
";
        exit(0);
```

```
}

sub wrap {
        # 줄길이가 지정된 글자수보다 적도록 화일의 각 줄을 배열에 할당한다.
        # 공백을 이용해 줄길이를 조정한다.

        # 줄길이를 제한할 글자수의 값을 얻는다.
        $limit = pop(@_);

        # DBG
        #print "Entering subroutine wrap\n";
        #print "The line length limit is $limit\n";

        # 화일을 읽어들여 분석하고 각 줄을 배열에 할당한다.
        open(FILE, "<$file") || die "Can't open $file: $!\n";
        while(<FILE>) {
                $line = $_;

                # DBG
                #print "The line is:\n$line\n";

                # 제한길이보다 긴 줄은 길이를 조정한다.
                while ( length($line) > $limit ) {

                        # DBG
                        #print "Wrapping...";

                        # $limit +1 까지의 글자들을 얻는다.
                        $part = substr($line,0,$limit +1);

                        # DBG
                        #print "The partial line is:\n$part\n";

                        # 마지막 문자가 공백인지 아닌지를 조사한다.
                        $last_char = substr($part,-1, 1);
                        if ( " " eq $last_char ) {
                            # 공백이라면 그대로 출력한다.

                            # DBG
                            #print "The last character was a space\n";

                            substr($line,0,$limit + 1) = "";
                            substr($part,-1,1) = "";
                            push(@newlines,"$part\n");
                        } else {
                             # 공백이 아니라면 마지막 공백의 위치를
                             # 찾고 그곳까지 출력한다.

                            # DBG
                            #print "The last character was not a space\n";

                             # $limit 이전의 글자를 지운다.
                             substr($part,-1,1) = "";
                             # 마지막 공백을 찾기 쉽게 줄을 뒤집는다.
                             $revpart = reverse($part);
```

```
                                $index = index($revpart," ");
                                if ( $index > 0 ) {
                                  substr($line,0,$limit-$index) = "";
                                  push(@newlines,substr($part,0,$limit-$index)
                                      . "\n");
                                } else {
                                  # 공백을 포함하지 않은 줄이라면
                                  # $limit 까지 출력을 한다.
                                  substr($line,0,$limit) = "";
                                  push(@newlines,substr($part,0,$limit)
                                      . "\n");
                                }
                        }
                }
                push(@newlines,$line);
        }
        close(FILE);
}

sub codeformat {
        # 서브루틴 wrap를 호출해서 nenscript를 통한 필터링을 한다.
        &wrap($line_length);

        # 소스 코드를 프린트하기 위해 몇가지 알맞은 형식들(용지방향,
        # Courier font, 줄번호)을 포함시킨 포스트스크립트 화일을 생성
        # 하기 위해서 nenscript의 결과를 처리한다.
        # 첫번째로 임시화일로 출력한다.
        $tmpfile = "/tmp/nenscript$$";
        open(FILE, "|$nenscript -2G -i$file -N -p$tmpfile -r") ||
                die "Can't open nenscript: $!\n";
        foreach $line (@newlines) {
                print FILE $line;
        }
        close(FILE);

        # 임시 화일을 다시 읽어들여서 삼바 프린트 스크립트로 출력할
        # 수 있도록 배열에 저장한다.
        @newlines = ("");
        open(FILE, "<$tmpfile") || die "Can't open $file: $!\n";
        while(<FILE>) {
                push(@newlines,$_);
        }
        close(FILE);
        system("rm $tmpfile");
}

sub createarray {
        # 포스트스크립트를 위한 배열을 생성한다.
        open(FILE, "<$file") || die "Can't open $file: $!\n";
        while(<FILE>) {
                push(@newlines,$_);
        }
        close(FILE);
}
```

제 10 절 저작권

제 11 절 감사의 글

제안 사항이 있다면 바로 메일을 보내기 바란다. 다음 개정판의 이곳에 당신에게 감사를 표하겠다.