

## Ubuntu 无盘工作站安装详细步骤

### 【开宗明义】

首先，请注意本文所述的“无盘工作站”与“无盘终端”的异同：

相同之处是二者的客户端都没有磁盘。

相异之处是，“无盘终端”完全使用服务器端的资源（CPU 和磁盘），与直接登录在服务器控制台差别不大，所以适用于客户端是极其老旧的机器，因为，即使用较新的机器则其运算能力同样无法发挥、从而造成资源浪费。

而“无盘工作站”的客户端在登录阶段，将服务器端作为登录服务器（tftp 和 dhcp），在登录完成后，则主要是把服务器端作为 nfs 服务器提供磁盘空间，此磁盘空间映射为客户机端的根目录，随后所有的运算工作在客户机端完成、并不占用服务器的 CPU 资源。所以，“无盘工作站”较适合当今性能中等以上的客户端使用，以平衡服务器、客户机的负载。

从实现的方式来说，Ubuntu 自带的 ltsp 服务属于“无盘终端”，虽然设置简易，但并不适合我们遇到的大多数情况：如今谁还会用一堆 256M 的 586 机器去组网呢？有鉴于此，Ubuntu 社区曾经试图调和两者，提出了“胖客户机”的概念，是从基本的 ltsp 延伸出来，详情请见 <https://help.ubuntu.com/community/UbuntuLTSP/LTSPFatClients>

但是说老实话，上述基于 ltsp 的“无盘工作站”是毫无必要地自找麻烦，比如说里面用到了 openLDAP，还要修改一堆脚本，实际操作的难度可想而知，所以：

### 【注意】

本文并不讨论 ltsp “无盘终端”或“ltsp 胖客户机”，而是基于 Linux 系统固有的能力，建构一个可以适用于所有 Linux 发行版的“无盘工作站”安装方法，而 Ubuntu 只是作为一个实例使用。

### 【成功结果展示】

在进入正文之前，可以先到：

<http://hi.baidu.com/camark/blog/item/999ea112722ce0cac2fd7807.html>

这一名为“Ubuntu 高地”的博客看一下无盘工作站安装成功后的类似效果。本文的由来，就是参考了此一重要文献，但因为作者是位高手，所以文中省略了很多他认为是天经地义的步骤，这就造成了象作者这样的初学者的困难。好在该文的下面链接了另一篇文章：<http://hi.baidu.com/yhsky123/blog/item/cfb06b08ad1a23d563d986db.html> 是名为“天使之翼”的另一高手的详细安装记录，本文的大部分将转载自该安装记录。

但是“天使之翼”的文章也有一个小问题，就是其中的客户端使用了一台有磁盘的机器作为初始安装，然后再将已经生成在磁盘上的所有文件打包拷贝到服务器端的 nfs 空间（也即实际运行时的客户端远程虚拟根目录）里。这样做当然有简便、稳妥的好处，但是如果拟安装的客户端从一开始就没有磁盘呢？文中并没有提供后一种情况的替代办法。

而本文在包含了“天使之翼”的“有盘安装”的内容的同时，另外补充了彻底无盘安装客户端系统的方法（也就是 Ubuntu 高地一文中已实现但没有披露的部分）。所以：

### 【正文前提示】

从现在开始，请不要再去回顾上述的三个链接，以免混乱。本文已经包含了上述的所有有效内容。

### 【正文第一部分】服务器端（机器名为 ubuntu）准备

首先，本文中的服务器端名为 ubuntu (IP 是 192.168.1.88)，而客户端名为 netfs，和“天使之翼”文中的命名方法相同，以方便拷贝相关命令和内容。以下显示为 root@ubuntu 的即指在服务器端操作，而显示为 root@netfs 的即指在客户端操作。

## 【正文第一部分-1】服务器端三大服务

服务器端需准备 tftp、dhcp、nfs 这三个服务，其过程如下：  
(以下全部来自“天使之翼”原文，仅加了必要的注解)

### 第一步 安装 tftp 服务器

#### 1 安装

```
root@ubuntu:/# apt-get install tftpd-hpa
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
Reading state information... 完成
下列新软件包将被安装:
tftpd-hpa
共升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 0 个软件未被升级。
需要下载 34.0kB 的软件包。
解压缩后会消耗掉 152kB 的额外空间。
获取: 1 http://Ubuntu.cn99.com hardy/main tftpd-hpa 0.43-1.1ubuntu1 [34.0kB]
下载 34.0kB，耗时 5s (5921B/s)
正在预设定软件包 ...
选中了曾被取消选择的软件包 tftpd-hpa。
(正在读取数据库 ... 系统当前总共安装有 112536 个文件和目录。)
正在解压缩 tftpd-hpa (从 .../tftpd-hpa_0.43-1.1ubuntu1_i386.deb) ...
正在设置 tftpd-hpa (0.43-1.1ubuntu1) ...
```

```
root@ubuntu:/#
```

#### 2 设置 tftpd

```
root@ubuntu:~# nano /etc/default/tftpd-hpa
#Defaults for tftpd-hpa
RUN_DAEMON="yes"
```

```
#上面这句表示启动守护进程，tftpd 工作
```

```
OPTIONS="-l -s /var/lib/tftpboot"
```

```
#上面这句表示 tftp 客户端能取得的文件所存放的位置
```

#### 3 启动服务

```
root@ubuntu:/# /etc/init.d/tftpd-hpa start
```

```
Starting HPA's tftpd: in.tftpd.
root@ubuntu:/# ps aux|grep tftp
root 26853 0.0 0.1 2196 288 ? Ss 17:26 0:00 /usr/sbin/in.tftpd -l -s
/var/lib/tftpboot
root 26862 0.0 0.2 3180 748 pts/1 R+ 17:27 0:00 grep tftp
root@ubuntu:/#
```

#### 4 查看服务是否开始工作

```
root@ubuntu:/# netstat -pna|grep tft
udp 0 0 0.0.0.0:69 0.0.0.0:* 26853/in.tftpd
unix 2 [ ] DGRAM 164700 26853/in.tftpd
root@ubuntu:/#
```

## 第二步 安装 dhcp 服务器

### 1 服务器环境

```
root@ubuntu:/# uname -a
Linux ubuntu 2.6.22-14-generic #1 SMP Sun Oct 14 23:05:12 GMT 2007 i686
GNU/Linux
root@ubuntu:/#
```

### 2 安装命令

```
root@ubuntu:/# apt-get install dhcp3-server
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
Reading state information... 完成
dhcp3-server 已经是最新的版本了。
共升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件未被升级。
```

### 3 设置 dhcpd 工作接口

```
root@ubuntu:~# nano /etc/default/dhcp3-server
# Defaults for dhcp initscript
# sourced by /etc/init.d/dhcp
# installed at /etc/default/dhcp3-server by the maintainer scripts
#
# This is a POSIX shell fragment
#
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
# 下面这句用来定义工作接口，如果是多个就中间空格
# 比如 INTERFACES="eth0 eth1 eth2"
```

```
INTERFACES="eth0"
```

(注\*上面一行指明服务器端通过哪一块网卡提供 dhcp 服务)

#### 4 主要设置

```
root@ubuntu:~# nano /etc/dhcp3/dhcpd.conf
```

```
#  
# Sample configuration file for ISC dhcpd for Debian  
#  
# $Id: dhcpd.conf,v 1.1.1.1 2002/05/21 00:07:44 peloy Exp $  
#  
  
# The ddns-updates-style parameter controls whether or not the server will  
# attempt to do a DNS update when a lease is confirmed. We default to the  
# behavior of the version 2 packages ('none', since DHCP v2 didn't  
# have support for DDNS.)  
ddns-update-style none;  
  
#下面是全局设置，这里定义的信息全 dhcp 服务器生效  
#我一般注释掉了，下面可以分不同的子网进行设置  
# option definitions common to all supported networks...  
#option domain-name "apt-get.cn";  
#option domain-name-servers 202.103.0.117, 202.103.24.68;  
#default-lease-time 600;  
#max-lease-time 7200;  
  
# If this DHCP server is the official DHCP server for the local  
# network, the authoritative directive should be uncommented.  
#authoritative;  
  
# Use this to send dhcp log messages to a different log file (you also  
# have to hack syslog.conf to complete the redirection).  
log-facility local7;  
  
# No service will be given on this subnet, but declaring it helps the  
# DHCP server to understand the network topology.  
  
#subnet 10.152.187.0 netmask 255.255.255.0 {  
#}  
  
# This is a very basic subnet declaration.  
  
#subnet 10.254.239.0 netmask 255.255.255.224 {  
# range 10.254.239.10 10.254.239.20;
```

```

# option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
#}

# This declaration allows BOOTP clients to get dynamic addresses,
# which we don't really recommend.

#subnet 10.254.239.32 netmask 255.255.255.224 {
# range dynamic-bootp 10.254.239.40 10.254.239.60;
# option broadcast-address 10.254.239.31;
# option routers rtr-239-32-1.example.org;
#}

# A slightly different configuration for an internal subnet.
#subnet 设置一个子网 192.168.1.0/24
#range 定义可以分配出去的地址为 1.50 到 1.70
#option domain-name-servers 定义 dns 为 202.103.0.117 等三个，这里注意每个之间要有个逗号
#option domain-name 定义域名称
#option routers 定义网关地址
#broadcast-address 定义广播地址
#default-lease-time 默认租约时间
#max-lease-time 最大租约时间
（注*下面这一段是生效部分，请按照实际情况修改）
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.50 192.168.1.70;
（注*上行的动态 IP 范围请不要与系统中已有的 dhcp 服务器冲突，比如无线路由器上自带的 dhcp，但是也不需要把原有的关掉，只要范围不冲突就可以了，因为客户端在启动时会自动使用服务器端的 dhcp 所分配的地址、而不使用无线路由器上分配的）
option domain-name-servers 202.103.0.117,202.103.24.68,202.103.150.44;
option domain-name "apt-get.cn";
（注*上行的 domain-name 在互联网系统里会起作用，所以请选择一个你肯定不会去访问的名字、即使你并不知道它是否被注册成为域名）
option routers 192.168.1.1;
option broadcast-address 192.168.1.255;
default-lease-time 864000;
max-lease-time 86400000;
filename "pxelinux.0";
（注*上面这一行是要手工加的很关键的信息，实际就是启动无盘工作站网卡的方式，而其中的 pxelinux.0 其实是一个文件名，下文将谈到这个文件如何生成）
}

# Hosts which require special configuration options can be listed in
# host statements. If no address is specified, the address will be
# allocated dynamically (if possible), but the host-specific information
# will still come from the host declaration.

```

```
#host passacaglia {
# hardware ethernet 0:0:c0:5d:bd:95;
# filename "vmunix.passacaglia";
# server-name "toccata.fugue.com";
#}

# Fixed IP addresses can also be specified for hosts. These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP. Hosts for which no fixed address is specified can only
# be booted with DHCP, unless there is an address range on the subnet
# to which a BOOTP client is connected which has the dynamic-bootp flag
# set.
#host fantasia {
# hardware ethernet 08:00:07:26:c0:a5;
# fixed-address fantasia.fugue.com;
#}

# You can declare a class of clients and then do address allocation
# based on that. The example below shows a case where all clients
# in a certain class get addresses on the 10.17.224/24 subnet, and all
# other clients get addresses on the 10.0.29/24 subnet.

#class "foo" {
# match if substring (option vendor-class-identifier, 0, 4) = "SUNW";
#}

#shared-network 224-29 {
# subnet 10.17.224.0 netmask 255.255.255.0 {
# option routers rtr-224.example.org;
# }
# subnet 10.0.29.0 netmask 255.255.255.0 {
# option routers rtr-29.example.org;
# }
# pool {
# allow members of "foo";
# range 10.17.224.10 10.17.224.250;
# }
# pool {
# deny members of "foo";
# range 10.0.29.10 10.0.29.230;
# }
#}
```

## 5 启动服务器

```
root@ubuntu:/# /etc/init.d/dhcp3-server start
* Starting DHCP server dhcpd3 [ OK ]
root@ubuntu:/#
```

## 6 查看服务是否已经正常监听

```
root@ubuntu:/# netstat -aunp|grep dhcp
udp 0 0 0.0.0.0:67 0.0.0.0:* 23011/dhcpd3
已经在 67 号 udp 口上开始监听了
```

## 第三步 安装配置 nfs 服务器

### 1 安装

```
root@ubuntu:/# apt-get install nfs-common nfs-kernel-server nfs-client
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
Reading state information... 完成
nfs-common 已经是最新的版本了。
nfs-kernel-server 已经是最新的版本了。
注意, 我选了 nfs-common 而非 nfs-client
nfs-common 已经是最新的版本了。
共升级了 0 个软件包, 新安装了 0 个软件包, 要卸载 0 个软件包, 有 0 个软件未被升级。
```

### 2 配置

```
root@ubuntu:~# nano /etc/exports
```

```
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync) hostname2(ro,sync)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt)
# /srv/nfs4/homes gss/krb5i(rw,sync)
/home/cache/netboot 192.168.1.0/24(rw,no_root_squash,sync)
```

(注\*上面这一行是服务器端提供的磁盘空间的位置, 可以是服务器的任一目录, 建议将一个单独的磁盘分区挂在这个目录下。但是请注意: 这个服务器端的/home/cache/netboot 并不是将来客户端的虚拟根目录, 因为在/home/cache/netboot 下面将会有有一个名为 root 的子目录, 而这个/home/cache/netboot/root 才是本文中的客户端的虚拟根目录, 在启动完成后、实际运行过程中, 工作就仅局限在 /home/cache/netboot/root 中了。建立 root 的问题下文将会讲到)

### 3 启动 nfs 或者重新加载

#### 启动 nfs

```
root@ubuntu:/# /etc/init.d/nfs-kernel-server start
* Exporting directories for NFS kernel daemon...
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check'
specified for export "192.168.1.0/24:/home/cache/netboot".
Assuming default behaviour ('no_subtree_check').
NOTE: this default has changed since nfs-utils version 1.0.x
...done.
* Starting NFS kernel daemon
...done.
```

如果是修改了/etc/exports 配置文件，不需要重新启动 nfs 服务器，只需要刷新一下，命令如下

```
root@ubuntu:/# exportfs -r
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check'
specified for export "192.168.1.0/24:/home/cache/netboot".
Assuming default behaviour ('no_subtree_check').
NOTE: this default has changed since nfs-utils version 1.0.x
```

#### 【正文第一部分-2】服务器端其他必要准备

服务器端还需准备上文提到的 pxelinux.0（也即远程启动客户端网卡的核心模块），其过程如下：  
（以下全部来自“天使之翼”原文，仅加了必要的注解）

#### 第四步 安装 syslinux

1 安装 syslinux，其实也就是为了要里面的 pxelinux 部分的文件

```
root@ubuntu:/# apt-get install syslinux
```

正在读取软件包列表... 完成

正在分析软件包的依赖关系树

Reading state information... 完成

syslinux 已经是最新的版本了。

共升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件未被升级。

2 拷贝 pxelinux.0 文件到 tftpboot 目录

```
root@ubuntu:/# cp /usr/lib/syslinux/pxelinux.0 /var/lib/tftpboot/
root@ubuntu:/#
```

（注\*本文中将会涉及两个不同的 tftpboot 目录，这里是其中的第一个，请不要和下文的/home/cache/netboot/tftpboot 混淆）

3 在 tftpboot 目录建立 pxelinux.cfg 目录，然后在 pxelinux.cfg 目录下建立 default 文件  
也可以是以某个 ip 地址为文件名称

```
root@ubuntu:/# nano /var/lib/tftpboot/pxelinux.cfg/default
```

（注\*所谓 default 其实是假设系统中只有一台客户端时的简易操作，如果有多台客户端，则需建立多个

文件、以客户端各自的 IP 地址为文件名。服务器也就是通过这种方法来区别不同的客户端，以导入各自不同的虚拟根目录)

```
DEFAULT ubuntu
LABEL ubuntu
kernel linux
append initrd=initrd.nfs root=/dev/nfs nfsroot=192.168.1.88:/home/cache/netboot/
root ip=dhcp rw
（注*注意这一行出现的/home/cache/netboot/root 就是上文说到的 default 客户端将来的虚拟根
目录在服务器端上的位置，而非其上级的/home/cache/netboot。当有多个客户端时，其各自的以 IP
命名的文件中，这一行的目录路径应该是不同的、以此来区别各自独立的虚拟根目录）
PROMPT 1
TIMEOUT 3
```

### 【正文第二部分】有盘客户端（机器名为 netfs）安装

按照“天使之翼”的方法，如果客户端原本是有磁盘的话，那么服务器端的准备工作到上文为止已经准备好了（当然并不意味着不再需要到服务器端上去操作，只是下面到服务器端的操作实际上也是为了准备客户端的文件）。如果您是这种情况，那么请从这里接着阅读、直到完成。

（如果您的情况是和我一样，需要安装一台自始就没有磁盘的客户端，那么请跳到【正文第三部分】阅读而不要停留于此，否则只会徒增混乱）

在客户端有盘的情况下，整体思路是：在这个磁盘上安装一个适合客户端硬件平台使用的完整的 ubuntu 系统，然后将根目录整体打包（当然经过一些必要的处理、屏蔽掉某些目录、修改某些文件，下文将详述），再拷贝到服务器端为客户端准备的虚拟根目录下（对于 default 客户端来说就是/home /cache/netboot/root），然后就可以取下客户端的磁盘，使之作为一个真正的无盘工作站从网络启动。其过程如下：

（以下全部来自“天使之翼”原文，仅加了必要的注解）

第五步 安装一个新的将来用来在无盘机器上运行的 linux，我这里安装的是一个 ubuntu 7 的版本（注\*这里指的是在客户端磁盘上进行一次传统的安装）

#### 1 安装

注意的是

```
l> 安装好 nfs 的 client
root@netfs:~# apt-get install nfs-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting nfs-common instead of nfs-client
nfs-common is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
```

2> ip 分配需要修改为手动，因为开机器的时候已经分配了 ip

（注\*这里指的是客户端虽然在网络启动阶段已经从服务器端的 dhcp 那里获得了一个动态 IP，但是当他将来进入自己的虚拟根目录时，前面的动态 IP 将不再起作用，而需要用 interfaces 文件定义一个在正常工作阶段使用的 IP，所以这一步需要在拔掉磁盘之前先改动好，下文的 host、hostname、fstab、mtab 的修改其原理同此）

```
root@netfs:~# nano /etc/network/interfaces
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
# The loopback network interface
```

```
auto lo
```

```
iface lo inet loopback
```

```
# The primary network interface
```

```
auto eth0
```

```
iface eth0 inet manual
```

3> 编辑 fstab 文件，把除了 proc 外的所有都注释掉，udev 会自动完成这个工作

（注\*因为客户端无盘运行时的挂载当然与现在有盘时的方式不同，鉴于客户端已经从服务器端取得了/home/cache/netboot/root 作为自己的根目录，所以另外需要的只剩一个 proc）

4> 编辑 udev 的 rules 中关于网络借口记录的文件，去掉已经定义了的网络接口，否则无盘启动了有个报错

```
root@netfs:~# :>/etc/udev/rules.d/70-persistent-net.rules
```

（注\*这一步没有理由，只是“天使之翼”的经验之谈，请照做就是）

## 2 安装 initramfs-tools

（注\*这是在网络启动阶段用到的最小系统工具，关键是下文的将启动方式改为 nfs，照做就是）

```
root@netfs:~# apt-get install initramfs-tools
```

```
Reading package lists... Done
```

```
Building dependency tree
```

```
Reading state information... Done
```

```
initramfs-tools is already the newest version.
```

```
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
```

```
root@netfs:~#
```

## 3 编辑 initramfs.conf 把 BOOT=local 改为 BOOT=nfs

```
#
```

```
# initramfs.conf
```

```
# Configuration file for mkinitramfs.
```

```
#
```

```
#
```

```
# MODULES: [ most | netboot | dep | list ]
#
# most - Add all framebuffer, acpi, filesystem, and harddrive drivers.
#
# dep - Try and guess which modules to load.
#
# netboot - Add the basemodules, network modules, but skip block devices.
#
# list - Only include modules from the 'additional modules' list
#
```

```
MODULES=most
```

```
# BUSYBOX: [ y | n ]
#
# Use busybox if available.
#
```

```
BUSYBOX=y
```

```
#
# NFS Section of the config.
#
```

```
#
# BOOT: [ local | nfs ]
#
# local - Boot off of local media (harddrive, USB stick).
#
# nfs - Boot using an NFS drive as the root of the drive.
#
```

```
BOOT=nfs
```

```
#
# DEVICE: ...
#
# Specify the network interface, like eth0
#
```

```
DEVICE=eth0
```

```
#
# NFSROOT: [ auto | HOST:MOUNT ]
```

```
#
```

```
NFSROOT=auto
```

#### 4 创建支持 nfs 的 initrd.img、vmlinuz 文件

首先我们把 nfs 挂上来，创建的文件直接丢到 nfs 服务器上去

```
root@netfs:/# mkdir /netfs
```

```
root@netfs:/# mount -t nfs 192.168.1.88:/home/cache/netboot /netfs
```

建立两目录

```
root@netfs:/# cd /netfs/;mkdir root tftpboot
```

（注\*这里不要看乱了，“天使之翼”的做法是在客户机端先挂载服务器端的/home/cache/netboot，然后在其下建立两个目录 root 和 tftpboot，此处的操作后果实际都发生在服务器端的磁盘上，这里的 root 就是我一再强调的客户端将来的真正的虚拟根目录，而这里的 tftpboot 是本文涉及的第二个 tftpboot，不同于服务器端/var/lib 下面的那个 tftpboot，至于为什么会有两个 tftpboot 我也没有深究）

创建支持 nfs 启动的 initrd.img 文件

```
root@netfs:/# mkinitramfs -o /netfs/tftpboot/initrd.nfs
```

拷贝内核到 tftpboot 目录

```
root@netfs:/# cp /boot/vmlinuz-2.6.24-16-generic /netfs/tftpboot/linux
```

（注\*上面的两个文件 initrd.nfs 和 linux 其结果也都是生成在服务器端的/home/cache/netboot/tftpboot 目录下，其作用类似于普通系统里的 initrd.gz 和 vmlinuz，是启动整个系统所必须的两个文件。从这里也可以看出/home/cache /netboot 下的两个子目录的分工情况：由 tftpboot 进行启动，然后交给 root，后者就是客户机端实际工作的虚拟根目录）

#### 5 打包 ubuntu 版本的/为 tgz 文件，先 apt-get update 一下，然后 apt-get clean

一下，打包的时候请排除掉 proc 等目录

```
root@netfs:/etc# apt-get clean
```

```
root@netfs:/etc# apt-get autoclean
```

```
Reading package lists... Done
```

```
Building dependency tree
```

```
Reading state information... Done
```

#用下面的命令打包

```
root@netfs:/# tar zcvfp /netfs/root/netfs.tgz / --exclude=/sys/*
```

```
--exclude=/netfs --exclude=mnt/* \
```

```
--exclude=/lost+found --exclude=/var/tmp/* --exclude=/proc/*
```

OK，打包完成了以后，关掉这个机器，到 nfs 服务器上那个机器去

（注\*至此有盘的客户端工作完成，可以拔掉磁盘）

#### 第六步

1> 在 nfs 服务器上拷贝内核和 initrd.img 到到/var/lib/tftpboot/

```
root@ubuntu:~# cp /home/cache/netboot/tftpboot/* /var/lib/tftpboot/
```

（注\*看似两个 tftpboot 里都需要启动文件 initrd.nfs 和 linux，有谁知道是为什么吗？）

2> 把刚才压缩的 netfs.tgz 文件解压缩一份到

```
/home/cache/netboot/root/
```

(注\*也就是把客户机端将来要用的整个系统拷贝到服务器端的对应虚拟根目录下)

```
root@ubuntu:/home/cache/netboot/root# ls -lh netfs.tgz
```

```
-rw-r--r-- 1 root root 96M 2007-12-22 06:03 netfs.tgz
```

```
root@ubuntu:/home/cache/netboot/root# cp netfs.tgz /home/cache/
```

```
root@ubuntu:/home/cache/netboot/root#
```

```
root@ubuntu:/home/cache/netboot/root# tar zxvf netfs.tgz && rm -fr netfs.tgz
```

```
root@ubuntu:/home/cache/netboot/root# ls
```

```
bin dev home initrd.img media opt root srv tmp var
```

```
cdrom etc initrd lib mnt proc sbin sys usr vmlinuz
```

删除 mtab 文件

```
root@ubuntu:/# cd /home/cache/netboot/root/etc/
```

```
root@ubuntu:/home/cache/netboot/root/etc# rm mtab
```

```
root@ubuntu:/home/cache/netboot/root/etc#
```

(注\*这个删除 mtab 的目的与上文的修改 fstab 大体相同, 是不是因为客户机端自身还在运行的时候删不掉, 所以才留到现在在服务器端做?)

### 第七步 设置一个客户机从网卡启动

当然你的机器一定要支持网络启动, 并且网卡的驱动已经在内核里面了, 如果没在, 请重新编译内核, 找到对应的网卡驱动, 然后敲空格选择星号表示把驱动编译到内核里面

(注\*现在的主板自带网卡大多数可以支持 PXE 启动, 如果遇到网卡问题请自行寻找参考资料解决, 本文不涉及该问题)

### 【正文第三部分】客户端从没有磁盘开始安装

从这里开始, 是本人全新编写的部分, 主要参考了“Ubuntu 高地”一文中的简要描述, 对该文没有讲明的一些细节, 下文将着重进行补充。首先让我们回顾一下, 在【正文第一部分】服务器端准备完成后, 我们已经作了哪些工作:

1、服务器端已经运行了三大服务 tftp、dhcp 和 nfs

2、PXE 启动必须的 pxelinux.0 已经生成并且放在服务器端 var/lib/tftpboot 里

3、与每个客户端对应的虚拟根目录位置信息已经在服务器端的/var/lib/tftpboot/pxelinux.cfg/这个目录下的对应文件里配置好

所以说, 我们下面要做的, 实际就是要在客户机端没有磁盘的情况下, 在服务器端的为其已经准备好

(即 nfs 服务已经 export) 的目录里, 生成其可以使用的操作系统, 体现为根目录下常见的

bin、usr、home……文件结构。对应【正文第二部分】的有盘客户端安装的内容(未阅读该部分也无妨, 见下列列表即可), 这个系统里必须有:

1、能够作为 nfs-client, 因为启动后虚拟根目录需要从服务器端 nfs 出来

2、必须有 `initramfs-tools`，并且 `initramfs.conf` 要修改为支持 `nfs` 启动

3、修改 `fstab`、`mtab`、`hosts`、`hostname`、`interfaces`、`udev` 里的 `rules` 以符合客户端正常运行时的情况

以上三条体现于虚拟根目录下，也就是服务器端的 `/home/cache/netboot/root` 目录，但是别忘了还有与其平行的 `/home/cache/netboot/tftpboot` 下必须有：

4、启动操作系统的两个文件 `initrd.nfs` 和 `linux`（是 `vmlinuz` 的改名）

所以，以下的工作就是围绕上述 1—4 这四个目标来做：

【正文第三部分-1】为无盘客户端建立基本系统而在服务器端进行的工作

第一步 利用 `debootstrap` 生成一个基本的可登录系统

因为客户端一片空白、且没有磁盘，所以这一步要在服务器端进行，目标是在 `/home/cache/netboot/root` 下生成一个最基本的可登录系统。首先：

```
root@ubuntu:/# cd /home/cache/netboot;mkdir root tftpboot 创建两个目录
root@ubuntu:/#debootstrap --arch=i386 hardy /home/cache/netboot/root
http://debian.nctu.edu.tw/ubuntu
```

（注意 `arch` 前面是连续两个-，而 `http://debian.nctu.edu.tw/ubuntu` 是我这里最快的源，也可以改成其他任何一个源）

该命令的结果是在 `/home/cache/netboot/root` 里面生成了一个可以字符登录的最基本的系统，该系统不依赖于任何特定的客户端硬件平台，只要是 `i386` 芯片的客户端在接下来都可以登录进去。当然，如果是 `amd64` 的芯片，则在上述 `debootstrap` 命令后跟的就是 `arch=amd64` 了。

经过检查发现，`nfs-client` 和 `initramfs-tools` 的功能已经随着基本的系统而自然生成，下面只需修改 `/home/cache/netboot/root/etc/initramfs-tools/initramfs.conf`：

```
# nfs - Boot using an NFS drive as the root of the drive.
#
BOOT=nfs
#
```

此时已经完成了我们四大目标中的 1 和 2

第二步 修改几个重要的配置文件

`/home/cache/netboot/root/etc/fstab` 文件修改成：

```
proc /proc proc defaults 0 0
192.168.1.88:/home/cache/netboot/root / nfs defaults,rw 0 0
```

（注意：与上文“天使之翼”只保留 `proc` 那一行不同，我在客户端将来的 `fstab` 里写明了要用到的 `nfs` 根目录，这样更保险）

`/home/cache/netboot/root/etc/hosts` 文件修改成：

```
127.0.0.1 localhost
127.0.1.1 netfs
```

（注意：`netfs` 是修改后的结果，在修改前这个位置实际是你现在所在的服务器的名字）

/home/cache/netboot/root/etc/hostname 文件修改成:

```
netfs
```

(修改前这个位置实际也是你现在所在的服务器的名字 ubuntu)

/home/cache/netboot/root/etc/network/interfaces 文件修改成:

```
auto lo
```

```
iface lo inet loopback
```

```
auto eth0
```

```
iface eth0 inet static
```

```
address 192.168.1.28
```

```
netmask 255.255.255.0
```

```
gateway 192.168.1.1
```

(注意:与“天使之翼”不同的是我直接指定了客户机的静态 IP192.168.1.28,在 ubuntu 里静态 IP 的定义好像一定要这样写得很完全)

/home/cache/netboot/root/etc/mtab 文件修改成:

```
192.168.1.88:/home/cache/netboot/root / nfs rw 0 0
```

```
proc /proc proc rw 0 0
```

```
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
```

(注意:与“天使之翼”删除 mtab 不同,原因与上述 fstab 的修改一样,实践证明可行。特别注意第 1 行末尾与 fstab 第一行是不一样的)

/home/cache/netboot/root/etc/udev/rules.d/70-persistent-net.rules 文件的修改:

```
root@ubuntu:/#:>/home/cache/netboot/root/etc/udev/rules.d/70-persistent-  
net.rules
```

(这一步其实可以省略,因为在基本的 arch 系统里这个文件本来就是空的)

除此之外,还有一个很重要的步骤,就是将服务器/etc/apt/sources.list 的内容拷贝到/home/cache/netboot /root/etc/apt/sources.list 里,否则到时候客户机端的源里只有生成 arch-i386 时的一条(在本文中就是 http://debian.nctu.edu.tw/ubuntu),会导致很多软件包找不到。

这样,我们就完成了四大目标中的 3

### 第三步 创建支持 nfs 的 initrd.nfs、linux (vmlinuz) 文件

这一步同样需在服务器端完成,只要服务器端本身具有 nfs 功能(因为我们在开头已经配置了服务器端的 nfs 服务),那么可以通过以下命令生成 initrd.nfs:

```
root@ubuntu:/# mkinitramfs -o /home/cache/netboot/tftpboot/initrd.nfs
```

至于 linux (也就是 vmlinuz),其实内核文件随处可见并且是通用的,可以:

```
#cp /boot/vmlinuz-2.6.24-16-generic /home/cache/netboot/tftpboot/linux
```

也就是将服务器端本身的 `vmlinuz` 拷贝到 `/home/cache/netboot/tftpboot` 中并且改名为系统所要求的 `linux`。

当然随后不要忘记把两个文件拷贝到 `/var/lib` 里的 `tftpboot` 目录下：

```
root@ubuntu:~# cp /home/cache/netboot/tftpboot/* /var/lib/tftpboot/
```

就这样，四大目标中的第 4 也完成了。如果一切正常，此时无盘的客户机端应该可以通过 PXE 启动方式，进入一个字符终端。

## 【正文第三部分-2】完全无盘客户机端的后续安装

此时，我们终于可以来到客户机端进行后续的工作。如果一切正常，客户机端可以用 `root` 用户和空口令登入（这个 `root` 是无盘客户机端的 `root`，从现在起客户机端的用户系统与服务器端已没有关系），第一步就应该用 `#passwd` 给 `root` 建一个口令，这是一个好习惯。

接下来的工作其实有点类似从一个纯字符的 `ubuntu` 服务器上建立 `x windows` 应用，只不过后者是在本地硬盘上安装文件，而我们的无盘工作站是把文件通过 `nfs` 远程安装到服务器端上的虚拟根目录（也就是 `/home /cache/netboot/root`）里。

当然，我们开始工作的这个只有 `arch` 的系统，比已经是字符型服务器的系统少了很多东西，所以需要一步步地添加进去。在“Ubuntu 高地”一文中，这部分是最容易让人迷惑的，因为文章里看起来很容易，但是如果不是亲自尝试，就很难想象会有那么多细节性的问题，而每一个问题带来的都是字符终端失去反应的后果。

现在让我们按顺序来进行配置：

### 第一步 安装内核

对，安装内核。`arch-i386` 虽然已经可以登录，但是内核的部分其实还缺很多，所以需要安装，具体来说，装的应该是全功能系统需要的很多库和模块。安装的命令是：

```
root@netfs:~# apt-get update
root@netfs:~# apt-get dist-upgrade
root@netfs:~# apt-get install linux-image-2.6.24-16-generic
```

前面两条是更新源、升级发行版，最后 1 条才是安装内核包。注意：内核一定要用 `linux-image-2.6.24-16-generic` 而不能用 `linux-image-2.6.24-16-386`，虽然两者其实是一样的，但是因为两者在 `/home/cache/netboot/root/lib /modules/` 里生成的目录名是不同的，只有 `-generic` 生成的那个不会带来后续的路径问题。

### 第二步 安装 console-data

```
root@netfs:~# apt-get install console-data
```

此处安装过程中会有类似 `dos` 那样的彩色字符窗口来设置键盘布局，需要按 `tab` 键选择 `ok` 等按钮、用方向键选择项目，鼠标此时是不起作用的。

### 第三步 安装 xserver-xorg

```
root@netfs:~# apt-get install xserver-xorg
```

### 第四步 安装 ubuntu-desktop

```
root@netfs:/#apt-get install ubuntu-desktop
```

注意：按照“Ubuntu 高地”的文章，在 xserver-xorg 完成后就可以安装 gnome 或者 xfce 了，但实际上是不行的，如果那样做的话，在 gnome 安装的最后会停下来报错说找不到 fontpath，而 fontpath 就是在安装 ubuntu-desktop 的时候自动配置的，所以到时候还是要补充安装 ubuntu-desktop。那么按照报错的信息提示，所缺乏的东西应该提前安装好，也就是 ubuntu-desktop 应该在 gnome 之前安装。

装完 ubuntu-desktop 之后，其实已经可以用 startx 命令进入 gnome 桌面，或者重启自动进入 gnome 的登录界面。这时候你会发现用 root 用户是无法登入的，需要一个普通用户。

用 Ctrl+Alt+F2 打开一个 tty 字符终端，用 root 登录进入后，用 useradd 命令创建一个普通用户（假定是 yourname），给以密码和主目录（具体命令略去，因为 useradd 大家都会用，记得 shell 选 /bin/bash）。

然后用 Ctrl+Alt+F7 回到 gnome 登录界面，用这个 yourname 普通用户登录进去。因为 yourname 是我们自己创建的，所以它默认还没有 sudo 权限，为了以后方便还是要加上这个权限，打开 gnome 终端：

```
yourname@netfs:~$su
```

```
root@netfs:/#gedit /etc/group
```

在 admin 开头的这一行尾巴加上 yourname 用户所在的组，使之变成如同 admin:x:112:yourname 这样，然后：

```
root@netfs:/#visudo 增加一行
```

```
%admin ALL=(ALL) ALL
```

注意开头的%不能少（而文件中默认的 root 开头的一行是没有%的）

这样 yourname 用户就可以作为你日常使用的帐号了，其功能才和光盘安装时建立的用户等同。

#### 第五步 安装 gnome 或者 xfce

接下来可以补充安装完整的桌面系统了。我个人建议是安装 gnome，因为就像“Ubuntu 高地”说的，xfce 在我们这种安装方式下会缺少很多包，需要手工补齐，所以不如利用 gnome 把这些包都装上，在装完 gnome 之后还是可以继续安装 xfce 供选用。

安装 gnome 的时候，99%可能遇到臭名昭著的“gnome-keyring-manager”问题。解决的方法也很简单，可以直接在 gnome 里下载安装。下载的地址是：

```
http://packages.debian.org/gnome-keyring-manager
```

唯一需要注意的是下载的版本，不能下载稳定版的 2-16-0 版、而只能下载 2-20-0 的非稳定版，因为这是 gnome 需要的最低版本。这个问题解决后，就可以：

```
yourname@netfs:~$sudo apt-get install gnome
```

完成后如果还需要 xfce，可以继续：

```
yourname@netfs:~$sudo apt-get install xubuntu-desktop
```

#### 第六步 启动 xwindows

在 gnome 或者 xfce 顺利安装完成后，重新启动进入一个英文的 xsession，进去后再更改语言等就比较简单了。但是还剩最后一个小问题，就是我们的这种安装方式不象光盘安装那样加进去了对 NTFS 文件系统的支持，当插入与 Windows 系统共用的移动硬盘时就会发现不能读写，这时需要从新立得里搜索 ntfs-3g 这个包并且安装。

到此我们的工作可以说是顺利完成了。

## 【正文第四部分】高阶配置

高阶配置除了 nfs 的安全性问题，还有多个 nfs 远程目录的挂载（即除了虚拟根目录外，将服务器的其他目录开放给客户机端），这部分请仔细研究 nfs 服务本身就很容易解决。

在实际应用中比较重要的高阶配置，就是上文已经提到过的多个客户机端同时存在，此时主要涉及服务器端的 dhcp 的设置和对应的 pxelinux 配置文件。在此引用“天使之翼”的补充说明：

常常无盘网络有多个客户端，这个时候我们需要修改 dhcp 服务器和 tftp 下的 pxelinux.cfg 下的设置  
我们首先来说一下 dhcp3-server 的修改

首先我们打开全局设置的部分，统一的设置网关 dns 租约期限等统一的信息

#设置所在域的名称

```
option domain-name "apt-get.cn";
```

#设置 dns 解析服务器

```
option domain-name-servers 202.103.0.117, 202.103.24.68;
```

#下面的时间使用-1 表示永久租约

```
default-lease-time -1;
```

```
max-lease-time -1;
```

然后找到我们先设置的地方

没改的时候如下

```
# A slightly different configuration for an internal subnet.
```

```
subnet 192.168.1.0 netmask 255.255.255.0 {
```

```
range 192.168.1.50 192.168.1.70;
```

```
option domain-name-servers 202.103.0.117,202.103.24.68,202.103.150.44;
```

```
option domain-name "apt-get.cn";
```

```
option routers 192.168.1.1;
```

```
option broadcast-address 192.168.1.255;
```

```
default-lease-time 864000;
```

```
max-lease-time 86400000;
```

```
filename "pxelinux.0";
```

```
}
```

修改方式如下

```
subnet 192.168.1.0 netmask 255.255.255.0 {
```

```
range 192.168.1.50 192.168.1.50
```

```
option routers 192.168.1.1;
```

```
option broadcast-address 192.168.1.255;
```

```
host A01 {
```

```
hardware ethernet 00:0c:29:81:bf:41;
```

```
option host-name "A001";
```

```
fixed-address 192.168.1.50;
```

```
filename "pxelinux.0";
```

```
}
```

```
host A02 {
hardware ethernet 00:0c:29:df:38:be;
option host-name "A002";
fixed-address 192.168.1.51;
filename "pxelinux.0";
}

}
```

多客户机的 dhcpd 的修改方法就是上面这样，下面我们设置一下 pxelinux.cfg 里面的内容

```
root@ubuntu:/var/lib/tftpboot/pxelinux.cfg# ls
01-00-0c-29-81-bf-41 01-00-0c-29-df-38-be default
root@ubuntu:/var/lib/tftpboot/pxelinux.cfg#
```

看一下，把 default 文件复制一个文件名修改为

01-开头，后面是无盘客户机的 mac 地址，看清楚中间是-不是：字母必须是小写，否则启动报错

也可以把名字修改为 16 进制的数字，比如 192.168.1.50

修改为 C0A80132

比如 192 转成了 16 进制的 C0

168 转成了 16 进制的 A8

这里必须是大写，小写启动的时候报错

如果启动的时候 nfs 连接的位置不同，记得修改这个文件里面的内容哦

（注\*也就是修改其中的 nfsroot=192.168.1.88:/home/cache/netboot/root ip=dhcp rw 这一行，用不同客户端的虚拟根目录所在目录更改/home/cache/netboot/root 这个 default 客户端所用的路径，当然前提是在 nfs 自身的/etc/exports 里面已经开放了这个目录的共享。

随后，如果客户机与第一台 default 是同型号的，可以直接把后者已经生成的虚拟根目录里的文件系统在服务器端完整拷贝到相应的目录的，但是要记得修改 hosts、hostname、interfaces 等配置文件以适应前者的情况，以免启动后发生 IP、机器名冲突等情况）

---