基础安装

本部分会带你安装一个最基础的，无图形化界面的 Arch Linux，本节内容主要参考Arch Wiki。

温馨提示：Linux 对大小写敏感，大部分命令均可通过`Tab`键进行自动补全。

1 禁用 reflector 服务

`reflector`会为你选择速度合适的镜像源，但本教程主要面对中国大陆用户，因为 GFW 的存在，该服务自动生成的内容并不准确，所以我们禁用它，之后手动配置镜像源。

```
systemctl stop reflector                    # 禁用reflector服务
```

2 验证启动模式

为了确保你正确设置了 UEFI 启动，我们再次验证。如果你看到了一大堆信息输出，那么你处在 UEFI 模式下。

```
ls /sys/firmware/efi/efivars                # 验证启动模式
```

3 连接网络

一般来说，你的网络均可以通过 DHCP 自动进行配置，对于有线用户不需要任何操作即可使用，对于 Wi-Fi 用户，请按照以下操作进行联网：

```
iwctl                                       # 启动iwctl程序进行联网
[iwd] device list                           # 列出设备，接下来以name作为示例
[iwd] station name scan                     # 扫描网络
[iwd] station name get-networks             # 获取网络列表，接下来以SSID作为示例
[iwd] station name connect SSID             # 连接名为SSID的网络，输入密码确认连接
[iwd] exit                                  # 退出
```

等待几秒钟即可连接到网络，你可以使用如下命令进行验证：

```
ping -c 4 archlinux.org                     # 验证网络连接状态
```

4 更新系统时间

```
timedatectl set-ntp true                    # 设置正确的NTP服务器
timedatectl status                          # 验证时间同步状态
```

5 磁盘分区

在这里我们只需要两个分区，这是一个简单通用的方案。

- BOOT 分区：2GB，FAT32 格式，挂载点`/boot`，用于存储启动文件
- 主数据分区：剩余全部空间，Btrfs 格式，挂载点`/`，用于存储系统与用户数据

你一定注意到了这与其它教程的不同之处，许多教程建议设置 Swap 分区，并将 Home 分区独立出来，使用 Ext4 格式，这并没有什么问题。Ext4 格式作为一个较为古老的格式，其稳定性是公认的，但是它不支持许多现代化的功能，比如透明压缩，快照，写时复制等等，故在本教程中我们推荐使用 Btrfs 分区格式，Swap 分区与 Home 分区会作为子卷进行挂载，详细内容请参考 Btrfs 官方文档，目前你只需要跟着我们的教程进行即可。

警告：接下来的操作会清空所选磁盘的全部数据，请三思而后行！！！

首先确认需要操作的磁盘位置：

```
fdisk -l                                    # 查看磁盘信息，接下来以nvme0n1举例
```

建立 GPT 分区表：

```
parted /dev/nvme0n1                         # 编辑磁盘分区表
(parted) mktable                            # 建立新分区表
New disk label type? gpt                    # 建立gpt格式分区，输入gpt回车即可
Yes/No? Yes                                 # 询问是否清空数据，输入Yes确认
(parted) quit                               # 退出
```

接下来使用 cgdisk 工具进行磁盘分区，cgdisk 是 gdisk 工具的一个 CLI 程序，操作非常直观，使用方向键即可选择你想要的操作，下面以建立 BOOT 分区举例：

```
cgdisk /dev/nvme0n1                         # 建立分区
```

使用方向键选择`[New]`选项，确认；它会询问你分区起始扇区号，保持默认即可；它会询问你分区大小，输入 2G，确认；它会询问你分区类型，默认为 Linux filesystem，这是主数据分区的格式，我们要建立 BOOT 分区，输入 ef00 ，确认；它会询问你分区名，你可以保持默认，也可以自己输入一个便于区分的名字，确认即可，至此一个分区就建立完成了。重复此过程建立主数据分区，注意分区焦点要选择空间大的 free space。

全部分区结束后，我们需要写入分区更改，然后验证我们的分区结果。使用方向键移动到`[Write]`选项，确认，它会询问你是否写入，输入 yes 即可，之后使用方向键移动到`[Quit]`选项退出分区工具。使用如下命令验证刚刚的分区结果：

```
fdisk -l /dev/nvme0n1                       # 验证对磁盘nvme0n1的分区结果
```

6 格式化

建立分区后需要格式化分区，按照如下命令执行即可：

```
mkfs.vfat -F 32 /dev/nvme0n1p1 -n BOOT      # 格式化BOOT分区，-n选项提供卷标
mkfs.btrfs -f /dev/nvme0n1p2 -L Linux       # 格式化主数据分区，-L选项提供卷标
```

如果你需要更加安全的全盘加密（类似于 Windows 的 Bitlocker），这一步会有所不同，你可以参考Arch Wiki 的这篇文章：Encrypting an entire system。本文后续也会提供相关教程，但这属于高级选项，故不在此赘述。

7 创建子卷并挂载分区

Btrfs 文件系统最好用的特性之一就是子卷，它类似于传统的磁盘分区，但又有所不同，具体可阅读上文提到的官方文档，你也可以先按照本文教程安装，之后使用虚拟机进行更为详细的研究。

首先我们创建必要子卷，这里参考了 Ubuntu 官方建议布局和 Arch Wiki 的建议。你也可以创建更多的子卷，比如为 Docker 创建 @docker 子卷，为 Libvirt 创建 @libvirt 子卷等等。子卷名称可以自定义，但是 @ 子卷与 @home 子卷不建议更改名称，这可以方便之后使用 Timeshift 软件进行快照的自动创建与管理。

```
mount -v /dev/nvme0n1p2 /mnt && cd /mnt     # 临时挂载分区用以创建子卷
btrfs subvolume create @                    # root子卷
btrfs subvolume create @home                # home子卷
btrfs subvolume create @swap                # swap子卷
btrfs subvolume create @var_log             # log子卷
btrfs subvolume create @var_cache           # cache子卷
cd ~ && umount -Rv /mnt                     # 退出目录并卸载分区
```

接下来我们挂载分区，`subvol`参数指定了挂载的子卷；`noatime`参数提供异步写入机制，可以提高数据的写入速度并降低访问时间；`discard=async`参数提供了固态硬盘的 TRIM 机制；`compress=zstd:3`参数提供了透明压缩的挂载选项，可以提高固态硬盘的寿命并减少磁盘使用空间；`\`可以将一条命令拆成多行输入：

```
# 将@子卷挂载为根目录
mount -v --mkdir /dev/nvme0n1p2 /mnt -o \
subvol=@,noatime,discard=async,compress=zstd:3
# 将@home子卷挂载为home目录
mount -v --mkdir /dev/nvme0n1p2 /mnt/home -o \
subvol=@home,noatime,discard=async,compress=zstd:3
# 将@swap子卷挂载为swap目录
mount -v --mkdir /dev/nvme0n1p2 /mnt/swap -o \
subvol=@swap,noatime,discard=async,compress=zstd:3
# 将@var_log子卷挂载为log目录
mount -v --mkdir /dev/nvme0n1p2 /mnt/var/log -o \
subvol=@var_log,noatime,discard=async,compress=zstd:3
# 将@var_cache子卷挂载为cache目录
mount -v --mkdir /dev/nvme0n1p2 /mnt/var/cache -o \
subvol=@var_cache,noatime,discard=async,compress=zstd:3

# 挂载BOOT分区
mount -v --mkdir /dev/nvme0n1p1 /mnt/boot
```

如果你创建了更多的子卷，也请在此挂载。

8 选择镜像源

编辑`/etc/pacman.d/mirrorlist`文件：

```
nano /etc/pacman.d/mirrorlit                # 编辑mirrorlist文件
```

pacman 会优先使用文件最顶端的镜像源进行下载。我们将北京外国语学院大学的镜像源和中国科学技术大学的镜像源放在文件开头即可。

```
Server = https://mirrors.bfsu.edu.cn/archlinux/$repo/os/$arch    # 北外
Server = https://mirrors.ustc.edu.cn/archlinux/$repo/os/$arch    # 中科大
```

因为本教程主要针对中国大陆用户，所以我们使用了国内的镜像源进行下载加速，但这存在一定的隐私问题。镜像源可以知道你的 IP 是什么，安装了哪些软件，当局有可能会要求镜像源提供这些内容。

如果在安装 Arch Linux 时你可以正常访问 Google 等服务，那么我们建议你使用非权威国家的镜像源进行安装，下面列举一些较为优质的镜像源：

```
Server = https://geo.mirror.pkgbuild.com/$repo/os/$arch          # 通用镜像
Server = https://mirror.0xem.ma/arch/$repo/os/$arch              # 加拿大
Server = https://mirror.aktkn.sg/archlinux/$repo/os/$arch        # 新加坡
Server = https://mirror.rackspace.com/archlinux/$repo/os/$arch   # 美国
Server = https://mirrors.cat.net/archlinux/$repo/os/$arch        # 日本
```

9 安装系统与必要软件

首先安装系统基本组件：基础工具，Linux-ZEN 内核，Linux-ZEN 内核头文件，系统固件。

这里我们使用了 ZEN 内核作为默认内核，它默认开启了一些特性，这可以提高日常使用时的性能。

如果你想使用其它内核，自行替换相关软件包即可，这里列出几个常用内核：

- `Linux`：Arch 默认内核
- `Linux-LTS`：LTS 版本内核，最为稳定，同时启用了一些服务器支持
- `Linux-Hardened`：更为安全的内核，启用了大量系统加固补丁

```
pacstrap -K /mnt base base-devel linux-zen linux-zen-headers linux-firmware
```

然后安装一些必要工具：DHCP 服务，Wi-Fi 连接软件，终端文本编辑器，bash 自动补全，btrfs 工具。

```
pacstrap /mnt dhcpcd iwd nano vim bash-completion btrfs-progs
```

如果你正在使用包含 NVIDIA 显卡的硬件，那么建议你现在先简单安装 NVIDIA 驱动[专有]，防止出现关机失败甚至无法关机的问题，之后我们再详细配置该驱动。

```
pacstrap /mnt nvidia-dkms
```

10 生成 fstab 文件

fstab 文件用于系统开机时磁盘分区的自动挂载，我们使用一个工具自动生成它。

```
genfstab -U /mnt >> /mnt/etc/fstab          # 生成fstab文件
```

11 切换到新系统进行配置

```
arch-chroot /mnt                            # 切换到新系统
```

12 配置时区

温馨提示：中国大陆将时区设置为上海即可，不要尝试寻找其它地区了！

```
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime     # 设置时区为上海
hwclock --systohc                           # 将时区写入硬件
```

13 进行本地化配置

编辑`/etc/locale.gen`文件，去掉`en_GB.UTF-8` `en_US.UTF-8` `zh_CN.UTF-8` `zh_SG.UTF-8`几行的`#`号注释。

```
nano /etc/locale.gen                        # 编辑locale.gen文件
```

使用如下命令生成本地化语言文件：

```
locale-gen                                  # 生成本地化语言文件
```

接下来设置全局语言变量，注意，请不要在这里设置任何中文语言变量，这会导致 tty 乱码。

```
echo "LANG=en_GB.UTF-8" > /etc/locale.conf  # 设置语言变量
```

与其它教程不同的是，我们推荐使用`en_GB`而不是`en_US`，这可以带来一些便利：

- 进入桌面环境后以24小时制显示时间
- LibreOffice 等办公软件的纸张尺寸会默认为 A4 而非 Letter (US)
- 可尽量避免可能造成处理麻烦的英制单位

14 设置主机名与本地解析

首先设置主机名，这里以`HostName`为例：

```
echo "HostName" > /etc/hostname                    # 设置主机名
```

然后设置本地地址解析：

```
nano /etc/hosts                                    # 编辑hosts文件
```

写入以下内容，`HostName`请自行替换，~~注释内容你看心情写入~~：

```
# The following lines are desirable for IPv4 capable hosts
127.0.0.1   localhost
# 127.0.1.1 is often used for the FQDN of the machine
127.0.1.1   HostName.localdomain    HostName
# The following lines are desirable for IPv6 capable hosts
::1         localhost   ip6-localhost   ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

15 为 root 用户设置密码

```
passwd root                                        # 修改root用户密码
```

温馨提示：Linux 终端输入密码不会有回显，你只需要输入并确认即可。

16 安装微码

处理器制造商会发布对处理器微码的稳定性和安全性更新。这些更新提供了对系统稳定性至关重要的错误修复。如果没有这些更新，则可能会遇到不明原因的崩溃或难以跟踪的意外停机。

```
pacman -S intel-ucode                              # Intel CPU用户安装
pacman -S amd-ucode                                # AMD CPU用户安装
```

17 安装引导程序

```
pacman -S efibootmgr                               # 创建EFI启动项工具
```

引导程序有多种选择，最常见的有 GRUB、rEFInd、systemd-boot，在本教程中我们使用 systemd-boot。

```
bootctl install                                    # 安装systemd-boot引导程序
systemctl enable systemd-boot-update               # 启用自动更新
```

配置加载选项：

```
nano /boot/loader/loader.conf                      # 编辑加载选项
```

写入以下内容，注释无需写入：

```
timeout 5                                          # 设置超时时间5s
console-mode keep                                  # 加载界面分辨率
editor yes                                         # 启用编辑
```

18 配置内核命令行

我们需要主分区的 UUID 来进行配置，使用如下命令获取主分区 UUID，这里假定你的主分区为`nvme0n1p2`：

```
blkid -s UUID /dev/nvme0n1p2                    # 获取指定分区UUID
```

你需要记下该 UUID 以供下文配置使用。接下来配置内核命令行：

```
nano /etc/kernel/cmdline                        # 编辑内核命令行
```

写入以下内容，`youruuid`自行替换为你的 UUID，如果你没有 NVIDIA 显卡则不需要 `nvidia` 相关参数：

```
root=UUID=youruuid rootfstype=btrfs rootflags=subvol=@ rw
nvidia_drm.modeset=1 nvidia_drm.fbdev=1
loglevel=5 nowatchdog
```

19 配置 UKI 启动文件

统一内核映像（UKI）是单个可执行文件，可以直接从 UEFI 固件启动，或者由启动加载程序自动获取。

在本教程中我们推荐使用 UKI 映像进行启动，这可以简化启动流程。首先删除系统自动生成的`initramfs`：

```
rm -rfv /boot/initramfs-linux-zen*              # 删除initramfs
```

编辑 mkinitcpio 配置文件使其默认生成 UKI：

```
nano /etc/mkinitcpio.d/linux-zen.preset         # 编辑mkinitcpio配置
```

在`default_config`、`default_image`、`fallback_config`、`fallback_image`前加入`#`进行注释；删除`default_uki`、`default_options`、`fallback_uki`、`fallback_options`前的`#`注释；修改相关行的`efi`为`boot`，最终结果如下：

```
# mkinitcpio preset file for the 'linux-zen' package

#ALL_config="/etc/mkinitcpio.conf"
ALL_kver="/boot/vmlinuz-linux-zen"

PRESETS=('default' 'fallback')

#default_config="/etc/mkinitcpio.conf"
#default_image="/boot/initramfs-linux-zen.img"
default_uki="/boot/EFI/Linux/arch-linux-zen.efi"
default_options="--splash /usr/share/systemd/bootctl/splash-arch.bmp"

#fallback_config="/etc/mkinitcpio.conf"
#fallback_image="/boot/initramfs-linux-zen-fallback.img"
fallback_uki="/boot/EFI/Linux/arch-linux-zen-fallback.efi"
fallback_options="-S autodetect"
```

之后生成启动映像：

```
mkinitcpio -P                                   # 生成启动映像
```

20 结束安装

```
exit                                    # 退出chroot系统
umount -Rv /mnt                         # 卸载相关分区
reboot                                  # 重启系统
```

注意，重启前要先拔掉优盘，否则在部分硬件上，你重启后还是进安装程序而不是安装好的系统。

至此，一个无 GUI 界面的 Arch Linux 安装就已经完成了。