

# 用 usbip 转发 raspberry pi 的 USB 键盘鼠标给 Arch Linux 的 PC



---

惠狐 megumifox 写了篇 [用PulseAudio将电脑的声音用手机放出来](#)，文末提到想知道我怎么用树莓派转发 USB 的，于是写篇文章记录一下。

# 起因

家里有个装了 Arch Linux ARM 的树莓派3B 闲着，装了 Arch Linux ARM 偶尔上电更新一下，不过因为性能实在不适合做别的事情于是一直在吃灰。某日给老婆安利幻想万华镜和老婆看片的时候，老婆不吃安利于是迁怒键盘鼠标键盘鼠标被长长的 USB 线扯着感觉很难受，于是偶发奇想，能不能利用一下树莓派的多达 4 个 USB 2.0 端口接鼠标键盘呢，这样鼠标键盘就可以跟着树莓派来回走，不用拖着长长的 USB 线了。

上网搜了一下，Linux 环境有个 usbip 工具正好能做到这个。原理也很直观，usbip 能把 USB 端口上的数据封装成 IP 协议通过网络转发出去，从而两个网络间相互联通的电脑就可以远程转发 USB 了。设置好的话，就像是一台 PC 多了几个位于树莓派上的 USB 端口，插上树莓派的 USB 设备统统作为 PC 的设备。

这篇文章假设有一个装了 Arch Linux 的 PC，和一个装了 Arch Linux ARM 的树莓派，并且两者间能通过网络互相访问到。别的发行版上大概也可以这么做，只是我没有试过。usbip 工具似乎普遍被发行版打包了，除此之外需要的也只是 Linux 内核提供好的功能而已。

# 设置 Arch Linux ARM 的树莓派端

假设树莓派上面网络已经设置妥当，开机插电就能自动联网。接下来安装 usbip 工具：

```
1 $ sudo pacman -Syu usbip
```

然后需要记录一下树莓派的 IP 地址：

```
1 $ ip addr
2 3: wlan0: .....
3 inet 192.168.0.117/24 brd 192.168.0.
255 scope global noprefixroute wlan0
4 .....
```

接下来给 udev 添加一个规则，当插入 usb 设备的时候，执行我的脚本 usbipall.sh 把 usb 设备通过 usbip 共享出去：

```
1 $ cat /etc/udev/rules.d/usbipall.rules
2 ACTION=="add", SUBSYSTEM=="usb", RUN
+="/usr/bin/bash /usr/local/bin/usbipall.sh"
```

这个 rules 文件 [可以在我的 dotfiles 里面找到](#)。

然后规则调用的 `usbipall.sh` 我这么写的，文件同样在我的 `dotfiles` 里面：

```
1  #!/bin/sh
2  (
3  allusb=$(usbip list -p -l)
4  for usb in $allusb
5  do
6      busid=$(echo "$usb" | sed "s|#.*
7      ||g;s|busid=||g")
8      if [ "$busid" = "1-1.1" ]
9      then
10         # ignoring usb ethernet
11         continue
12     fi
13     echo "$(date -Iseconds): Exporti
14     ng $busid"
15     usbip bind --busid="$busid"
16 done
17 ) >>/var/log/usbipall.log 2>&1
```

这个脚本做了这样几件事。

1. 调用 `usbip list --local` 列出本地所有 usb 设备。
2. 针对每个设备
  1. 取出它的 `busid`
  2. 判断是不是树莓派的 USB 以太网卡，不是的话继续
  3. 通过 `usbip bind --busid=` 命令把这个

usb 设备导出到网上

3. 最后把所有输出记录到 `/var/log/usbipall.log` 日志里面

树莓派这边设置就完成了。从此之后插入的 usb 设备就会统统导出出去。

这里需要注意一下，启用了 udev 规则之后，就没法插键盘鼠标到树莓派上控制它了……我都是从另一端 ssh 上树莓派操作的。如果有什么地方设置错误，可能需要把树莓派的 SD 卡拔下来插到电脑上，删除掉 rules 文件……

仔细检查设置正确了之后，重新载入 udev 规则，或者重启树莓派：

```
1 # systemctl restart systemd-udev
```

这样树莓派这边就设置好了。

## 设置 Arch Linux 的 PC 端

同样假设 PC 这边也已经联网。接下来同样安装 usbip 工具：

```
1 $ sudo pacman -Syu usbip
```

然后我写了个小脚本去链接树莓派端，这个文件 `usbiprpi3.sh` 也在我的 dotfiles：

```
1  #!/bin/sh
2  rpi3="192.168.0.117"
3
4  modprobe vhci-hcd
5
6  allusb=$(usbip list -p -r $rpi3 | cu
t -d":" -f1 -s | sed 's|^[ \t]*||;/^$/d'
)
7  for busid in $allusb
8  do
9      if [ "$busid" = "1-1.1" ]
10     then
11         # ignoring usb ethernet
12         continue
13     fi
14     echo "Attaching $busid"
15     usbip attach --remote=$rpi3 --bu
sid="$busid"
16 done
```

其中脚本第一行填入上面记录下来的树莓派的 IP 地址，接下来脚本做了这么几件事：

1. 用 `modprobe` 确认加载 `vhci-hcd` 通用虚拟键鼠驱动
2. 用 `usbip list --remote=` 列出远程设备上已经导出了的 USB 设备，取出他们的 `busid`

### 3. 对每个设备用 `usbip attach` 接上该设备

然后就已经准备妥当，接下来是见证奇迹的时刻：

```
1 $ sleep 10; sudo ./usbiprpi3.sh
2 Attaching 1-1.4.3
3 Attaching 1-1.4.1
```

因为只有一套键盘鼠标，所以先 `sleep` 个 10 秒，在此期间快速把键鼠拔下来插到树莓派的 USB 口上去。如果对自己手速没自信也可以把时间设长一点。然后用 `root` 权限执行 `usbiprpi3.sh`。

一切正常的话，先能观测插上树莓派的键盘鼠标被树莓派初始化了一下，比如键盘灯会亮，然后这些设备会被导出出去，从而键盘灯灭掉，然后 10 秒等待结束后他们被远程接到了 PC 端，又会被初始化一下，同时 PC 端这边会有上述 `Attaching` 的输出。然后键盘鼠标就能像平常一样用啦。

## 使用体验

因为就是通过 IP 转发 USB 嘛，所以就和普通地接 USB 的体验差不多，当然前提是网络环境足够稳定。在我家间隔 5 米到无线路由器的环境下，基本感觉不到网

络延迟的影响。通过这种方式聊天上网应该和直接接 USB 设备完全一样。本文就是在通过树莓派转发的前提下用键盘打字写的。

不过如果网络负载本身就很大的话，可能会一些延迟，比如我开着 OBS 直播打东方的时候，原本就手残的我感觉更加手残了……

试过拿着树莓派在房间到处走，走到无线信号覆盖不到的地方，usbip 会断掉，PC 上的现象就像是 USB 设备被拔下来了……所以如果无线网络不稳的话，可能需要对上面脚本做个循环？不过那样可能会用起来很别扭吧。

以及，上述操作 usbip 是走 TCP 3240 端口，数据包大概完全没有加密，所以考虑安全性的话，最好还是在内网环境使用。不过转念一想，万一有别人接上了我导出出去的 USB，也就是截获我的键盘，PC 这边没法 attach 设备了，应该马上会发现吧。我敲打 sudo 之类命令的时候 shell 里面没有回显，就不会再继续敲密码了。而且似乎对攻击者也没有什么好处？要是他 usb attach 到了我的设备上，我就能控制他的键盘了耶~