发挥NVIDIA PRIME的全部功力：真·自动切换

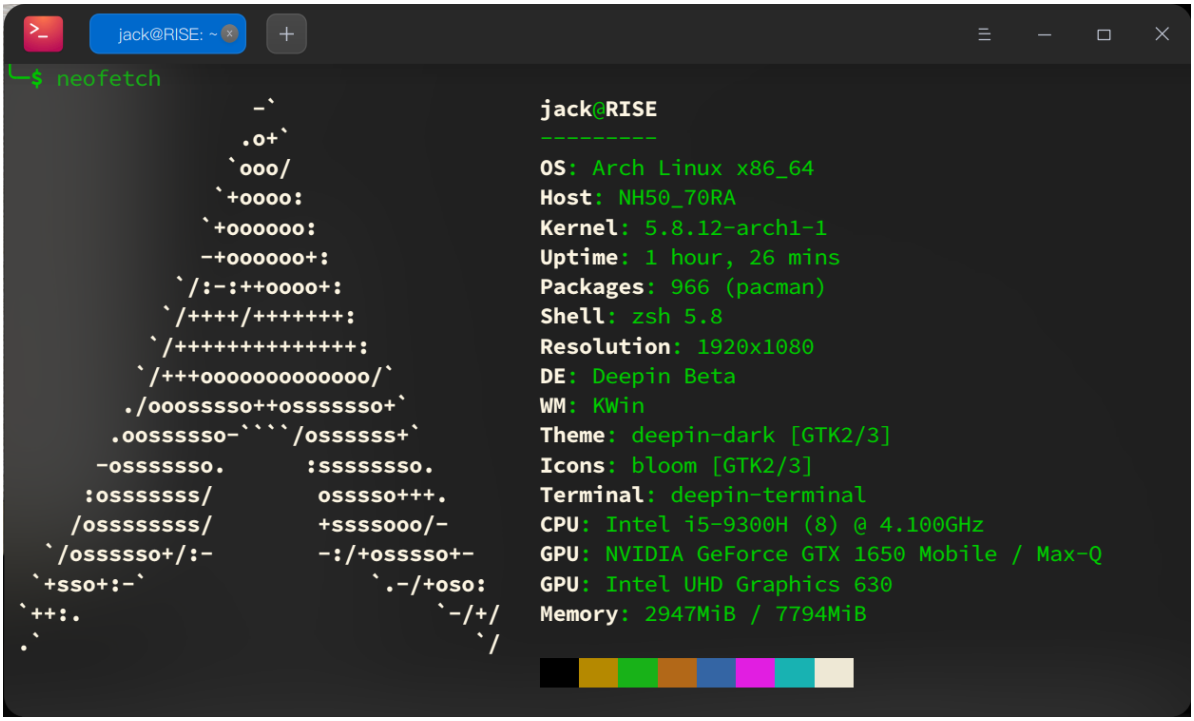
在我发了这篇贴子：一个可以尝试的Nvidia Prime方案之后，我收到了坛友们的积极反馈，而且论坛的大佬在写I+N双显卡自动切换教程时多多少少提到了我的贴子，说明I+N双显卡自动切换方案的确是大家的刚需。我很高兴我能为社区做出点贡献。

我在上篇文章中提到，只有turing及更新架构的显卡才支持真正的自动切换，而且在那篇文章里还存在一些我当时在认识上的错误。在这篇文章中，我将纠正错误，并且讲述如何发挥NVIDIA PRIME的全部功力。我不是显卡方面的专家，如果我在这篇文章中还存在一些认识上的错误，欢迎大佬们指出。在语言上我会以通俗易懂的方式写这篇文章，所以不用担心看不懂。

何为真正的自动切换？简单来说，就是需要使用独显时，给独显通电（激活独显），满足使用需求，而在不需要的时候，关闭独显的电源以省电。而我在那篇文章中提到的假自动切换，就是在你需要使用独显的时候，驱动程序会判断应用程序是否需要独显，如果需要，调用独显，让应用程序在独显上跑，如果不需要，则不调用，独显继续保持低功率运行以省电，但就是不能完全关闭电源。既然真自动切换不仅拥有我在假自动切换提到的功能，还可以自动关闭整个独显的电源，对我来说，不去折腾折腾怎么行？

不久前我刚买了一台新电脑，独显刚好是1650（Turing架构），正好测试一下PRIME的关键功能：PCI-Express运行时D3（RTD3）电源管理（机翻，英文名为PCI-Express Runtime D3 (RTD3) Power Management），我称它为动态电源管理。经过我几十次重启+重装三次系统，终于用上这个功能了。为何要如此折腾，直接抄官方文档不就行了吗？官方文档的编写时间离现在有些久远了，不经过充分测试怎么知道官方文档的方案还能不能正常使用？所以才有了这篇文章，这样可以让大家少走弯路。

我的机器配置：



参考文档（本文不是直接复制粘贴官方文档的方法，我还补充了一些官方文档没有讲到的细节问题）：

Chapter 22. PCI-Express Runtime D3 (RTD3) Power Management

Chapter 21. Configuring Power Management Support

Chapter 34. PRIME Render Offload

Laptop GPU Power Management

准备

严重警告：这是一项实验性的功能，其稳定性尚未知晓，只有经过大量的测试才能验证。我不能保证该功能可以在你的电脑上正常启用，所以为了你的系统安全，我强烈建议你备份系统，以便随时还原。你可能需要花费大量的时间测试这项功能在你的电脑上是否能稳定的正常运行，包括多次重启，打开一些需要独显的软件测试该功能，等等。如果你愿意花时间参与测试，请将测试情况发到这个贴子，这样可以为社区做贡献。

你的电脑必须满足这些要求，才能使用这个功能：

- 当前只有笔记本电脑支持这个功能（都过了一两年了，台式机的可以试试，看看有没有变化）
- 机翻：此功能需要系统硬件以及ACPI支持（控制PCIe电源需要ACPI“ _PR0”和“ _PR3”method），必要的硬件和ACPI支持首先在Intel Coffeelake芯片组系列中添加。因此，Intel Coffeelake芯片组系列支持此功能。

  查看自己的Intel芯片是否为Coffeelake芯片组：

  ```
  lspci -P | grep 00:00.0
  ```

  如果输出类似于这样：

  ```
  00:00.0 Host bridge: Intel Corporation 8th Gen Core 4-core Processor Host Bridge/DRAM Registers [Coffee Lake H] (rev 07) #注意[Coffee Lake H]
  ```

  那么CPU的芯片组确实是Coffeelake，如果没有输出，直接`lspci -P`，然后自己找到CPU的条目查看。
- GPU架构必须是Turing及更新架构
- Linux内核版本必须大于等于4.18，这个功能在低于这个版本的内核上极有可能无法工作
- 当Linux内核定义CONFIG_PM（CONFIG_PM = y）时，支持此功能。通常，如果系统支持S3（挂起至RAM），则将定义CONFIG_PM。（这个一般都定义了）

一个NVIDIA GPU可能具有一个或多个"PCI functions"：

- Function 0: VGA controller / 3D controller（VGA控制器/ 3D控制器）
- Function 1: Audio device（音频设备）
- Function 2: USB xHCI Host controller（USB xHCI主机控制器）
- Function 3: USB Type-C UCSI controller（USB Type-C UCSI控制器）

通过以下命令确认你的显卡有几个PCI functions（可以理解为PCI接口）：

```
lspci | grep NVIDIA
```

这是我的输出：

```
01:00.0 VGA compatible controller: NVIDIA Corporation TU117M [GeForce GTX 1650 Mobile / Max-Q] (rev a1)
01:00.1 Audio device: NVIDIA Corporation Device 10fa (rev a1)
```

记下那个`01:00.0`，以此类推，function 1（音频设备）应该是`01:00.1`，function 2应该是`01:00.2`，function 3应该是`01:00.3`，如果不是，请根据自己的实际情况进行记录。

你还必须知道：在这四个接口中，只有function 0才是由NVIDIA驱动程序管理的，其它三个则是由Linux内核提供的驱动进行管理。根据官方文档描述，~~这三个接口对应的驱动的电源管理还不成熟以及存在一些bug，不能很好的配合NVIDIA驱动程序一起开启/关闭GPU对应的接口~~，所以这部分就需要坛友们加以测试，我的测试结果会在后面说。

注：另查到这三个接口对应的驱动均拥有了较为成熟的电源管理(Power Management,简称PM)，故后面将启用这些接口的电源管理功能，该资料同样来自官方的文档，说明官方文档更新不及时，所以大家在参考官方文档时应多加思考，避免入坑。但即便如此，也需要经过大量测试以验证其有效性。

启用并配置功能

这个功能默认并未启用。手动启用：

```
echo auto > /sys/bus/pci/devices/0000\:01\:00.0/power/control #启用function 0的电源管理
```

以此类推，如果你想启用function 1的：

```
echo auto > /sys/bus/pci/devices/0000\:01\:00.1/power/control
```

若需要关闭自动切换功能，只需要把上述命令的`auto`改为`on`即可。

驱动设置

首先，在`/etc/modprobe.d/`里新建一个名为`nvidia.conf`的文件，打开该文件，添加如下内容：

```
options nvidia "NVreg_DynamicPowerManagement=0x02"
```

注意那个双引号里的内容，用最简单的方式去理解会是这样（要专业的自己去看官方文档）：

`NVreg_DynamicPowerManagement=0x00`：仅使用nvidia GPU的内置电源管理，假自动切换就是用的这种方式；

`NVreg_DynamicPowerManagement=0x01`：开启简单的自动切换管理功能；

`NVreg_DynamicPowerManagement=0x02`：开启精细的自动切换管理功能。

还看不懂？你直接这样理解：

`0x00`：低级功能；

`0x01`：中级功能；

`0x02`：高级功能。

可选参数：

`NVreg_DynamicPowerManagementVideoMemoryThreshold=100`：默认是200，注意这个参数的值最大就是200，超过200会被忽略掉，设置成0会导致GPU一直不会被关闭（这和0x00有什么区别）

这个功能可以这么理解，假设我设置成200，驱动会根据使用情况判断是否需要调用独显的内存（Video memory）：

若应用程序需要独显存储的内容不超过200M，则不使用独显，而是让你的系统内存存储本该是独显存储的内容；

若超过200，则开启独显，让独显存储相应的内容。

对于内存较少者，可以考虑把这个值改低，减小系统内存压力，如果你的内存较大，就不必设置了。

判断独显是否启用了自动切换管理功能，以及独显状态，独显对于自动切换管理功能的支持情况

输入如下命令：

```
cat /proc/driver/nvidia/gpus/0000:01:00.0/power
```

注意那个`01:00.0`，是不是很熟悉啊？上面让你记下这个值，就是为了配置自动切换功能做的。现在，你只需根据你获得的值，把上面命令的值替换掉就行了。如果你不放心，可以这样做，首先输入命令：

```
cat /proc/driver/nvidia/gpus/
```

此时不要回车，按两下`tab`键，shell将会显示该目录下的内容，自己根据显示的内容以及记下的值判断哪个才是自己需要找的文件。

输好命令后回车，这是我的输出：

```
Runtime D3 status:          Enabled (fine-grained) #自动切换功能已启用，而且是“高级功能”
Video Memory:               Off #独显已经彻底关闭掉了

GPU Hardware Support:
 Video Memory Self Refresh: Supported #这两个都必须是supported才意味着你的独显支持真自动切换功能，一般你的独显符合要求肯
 Video Memory Off:          Supported #定都是支持的，比Turing架构旧的独显只支持Video Memory Self Refresh，两个都不支
                                      #持？对不起，PRIME可能对你的独显无缘了
```

注意那个`Runtime D3 status`后面括号里的内容，如果是（fine-grained），说明你使用了“高级功能”（0x02），如果是（coarse-grained），说明你使用了“中级功能”（0x01），如果没有括号的内容且显示的是不是Enabled而是off，那么你用的就是“低级功能”（0x00）了。

自动配置

上述配置都是手动方式，而且在你重启之后那些配置就会失效了，我为何还要讲手动配置方式，就是为了让大家更好的理解并使用这些功能，现在将讲述自动配置方式。首先在`/lib/udev/rules.d/`新建一个名为`80-nvidia-pm.rules`的文件，请看如下内容：

```
# Remove NVIDIA USB xHCI Host Controller devices, if present
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x0c0330", ATTR{remove}="1"

# Remove NVIDIA USB Type-C UCSI devices, if present
```

```
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x0c8000",
ATTR{remove}="1"

# Remove NVIDIA Audio devices, if present
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x040300",
ATTR{remove}="1"

# Enable runtime PM for NVIDIA VGA/3D controller devices on driver bind
ACTION=="bind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030000", TEST=="power/control", ATTR{power/control}="auto"
ACTION=="bind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030200", TEST=="power/control", ATTR{power/control}="auto"

# Disable runtime PM for NVIDIA VGA/3D controller devices on driver unbind
ACTION=="unbind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030000", TEST=="power/control", ATTR{power/control}="on"
ACTION=="unbind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030200", TEST=="power/control", ATTR{power/control}="on"
```

上述内容均为官方文档提供的内容。等等！别这么快添加内容到文件并保存！不根据自己的实际情况修改内容，直接复制粘贴就完事，一旦出了问题很难找到根源所在！为了能减少不必要的折腾，请耐心看下去，一步一步把内容改成自己实际需要的样子。

首先查看你的内核版本：

```
uname -r
```

这是我的输出：

```
5.8.12-arch1-1
```

假设你的独显有4个接口，如果你的内核版本大于4.19且小于5.3，启用function 0和function 1，删除function 3和function 4；如果你的内核大于5.3，启用全部功能享受极致快乐！

现在，根据自身实际情况启用对应的功能：

function 0大家肯定都有，所以必须启动：

```
ACTION=="bind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030000", TEST=="power/control", ATTR{power/control}="auto"
```

有几个参数需要注意：

`ATTR{vendor}=="0x10de"`：**该参数中的`"0x10de"`是Nvidia的PCI vendor ID，如何确定我的机器上的独显的PCI vendor ID是什么？输入以下命令确定：**

```
lspci -n | grep 01.00.0
```

这是我的输出：

```
01:00.0 0300: 10de:1f91 (rev a1)
```

注意那个`10de:1f91`，意思是`<vendor id>:<device id>`，我们只需得到`<vendor id>`即`10de`就行了，然后你在把这个值写入文件时写成`0x + 10de`的形式比如`0x10de`就可以了。

你还可以运行`nvidia-settings`，然后直接在软件界面(Graphics Card Information)就可以查到这两个ID了。

`ATTR{class}=="0x030000"`：这个用`nvidia-settings`不能查到，所以只能用命令行方式。

输入命令：

```
dmesg | grep class | grep 10de
```

这是我的输出：

```
[    0.364750] pci 0000:01:00.0: [10de:1f91] type 00 class 0x030000
[    0.365077] pci 0000:01:00.1: [10de:10fa] type 00 class 0x040300
```

注意输出中类似于`0000:01:00.0`的内容，去掉`0000:`即可得到熟悉的bus id(01:00.0)，因此确定function 0的class id为`0x030000`，同理，function 1的class id为`0x040300`，以此类推。

由于本机中只有两个接口，所以我在文件中添加了如下内容：

```
# Enable runtime PM for NVIDIA VGA/3D controller devices on driver bind
ACTION=="bind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x030000", TEST=="power/control", ATTR{power/control}="auto"

# Enable NVIDIA Audio devices
ACTION=="bind", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de",
ATTR{class}=="0x040300", TEST=="power/control", ATTR{power/control}="auto"
```

如果你的机器因为该配置文件出现了问题，很可能是另外三个驱动工作不正常，那就直接删除，删除后三个接口的电源将一直保持关闭状态：

```
# Remove NVIDIA USB xHCI Host Controller devices, if present
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x0c0330",
ATTR{remove}="1"

# Remove NVIDIA USB Type-C UCSI devices, if present
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x0c8000",
ATTR{remove}="1"

# Remove NVIDIA Audio devices, if present
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x10de", ATTR{class}=="0x040300",
ATTR{remove}="1"
```

记得根据实际情况替换`ATTR{vendor}`和`ATTR{class}`的值，以及根据接口数删除不必要的配置。

通用电源管理：只要使用PRIME方案即可生效，无需真自动切换功能

无论你的独显是否支持真自动切换功能，只要PRIME方案在你的机器上可以用就应该做本节要讲述的操作。如果不做，将导致电脑在挂起/休眠并恢复、或者挂起/休眠失败并紧急恢复之后PRIME失效，表现为：无法在`nvidia-smi`找到Xorg进程；无法调用独显跑应用程序；如果已经有应用程序在独显上跑，恢复后应用程序工作异常甚至崩溃。解决方法：重启Xorg，包括注销、杀死Xorg进程两种方法。

那种解决方法治标不治本，所以需要用到本节所提到的方法，而且使用该方法还可以更好地配合PRIME的自动切换功能进行电源管理。依次输入下面的命令：

```
sudo systemctl enable nvidia-suspend.service
sudo systemctl enable nvidia-hibernate.service
sudo systemctl enable nvidia-resume.service
```

没有这些服务？尝试：

```
ls /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/
```

如果有输出，执行下列命令：

```
sudo install /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/nvidia-suspend.service
/etc/systemd/system
sudo install /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/nvidia-
hibernate.service /etc/systemd/system
sudo install /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/nvidia-resume.service
/etc/systemd/system
sudo install /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/nvidia
/lib/systemd/system-sleep
sudo install /usr/share/doc/NVIDIA_GLX-1.0/samples/systemd/nvidia-sleep.sh
/usr/bin
```

如果还没有，使用`find`命令查找服务文件是否存在：

```
sudo find / -iname "服务文件名"
```

还没有？来硬核的：

```
cd /usr/lib/systemd/system/
sudo nano nvidia-suspend.service
```

添加以下内容：

```
[Unit]
Description=NVIDIA system suspend actions
Before=systemd-suspend.service

[Service]
Type=oneshot
ExecStart=/usr/bin/logger -t suspend -s "nvidia-suspend.service"
ExecStart=/usr/bin/nvidia-sleep.sh "suspend"

[Install]
RequiredBy=systemd-suspend.service
```

继续创建新文件`nvidia-hibernate.service`：

```
[Unit]
Description=NVIDIA system hibernate actions
Before=systemd-hibernate.service

[Service]
Type=oneshot
ExecStart=/usr/bin/logger -t hibernate -s "nvidia-hibernate.service"
ExecStart=/usr/bin/nvidia-sleep.sh "hibernate"

[Install]
RequiredBy=systemd-hibernate.service
```

再创建`nvidia-resume.service`：

```
[Unit]
Description=NVIDIA system resume actions
After=systemd-suspend.service
After=systemd-hibernate.service

[Service]
Type=oneshot
ExecStart=/usr/bin/logger -t suspend -s "nvidia-resume.service"
ExecStart=/usr/bin/nvidia-sleep.sh "resume"

[Install]
RequiredBy=systemd-suspend.service
RequiredBy=systemd-hibernate.service
```

再创建`/usr/bin/nvidia-sleep.sh`：

```
#!/bin/bash

if [ ! -f /proc/driver/nvidia/suspend ]; then
    exit 0
fi

RUN_DIR="/var/run/nvidia-sleep"
XORG_VT_FILE="${RUN_DIR}"/Xorg.vt_number

PATH="/bin:/usr/bin"
```

```
case "$1" in
    suspend|hibernate)
        mkdir -p "${RUN_DIR}"
        fgconsole > "${XORG_VT_FILE}"
        chvt 63
        if [[ $? -ne 0 ]]; then
            exit $?
        fi
        echo "$1" > /proc/driver/nvidia/suspend
        exit $?
        ;;
    resume)
        echo "$1" > /proc/driver/nvidia/suspend
        #
        # Check if Xorg was determined to be running at the time
        # of suspend, and whether its VT was recorded.  If so,
        # attempt to switch back to this VT.
        #
        if [[ -f "${XORG_VT_FILE}" ]]; then
            XORG_PID=$(cat "${XORG_VT_FILE}")
            rm "${XORG_VT_FILE}"
            chvt "${XORG_PID}"
        fi
        exit 0
        ;;
    *)
        exit 1
esac
```

最后创建`/lib/systemd/system-sleep/nvidia`：

```
#!/bin/sh

case "$1" in
    post)
        /usr/bin/nvidia-sleep.sh "resume"
        ;;
esac
```

更改这些文件的权限：

```
sudo chmod 644 nvidia-suspend.service
sudo chmod 644 nvidia-hibernate.service
sudo chmod 644 nvidia-resume.service
sudo chmod 755 /usr/bin/nvidia-sleep.sh
sudo chmod 755 /lib/systemd/system-sleep/nvidia
```

编辑`/etc/modprobe.d/nvidia.conf`，追加如下内容：

```
options nvidia "NVreg_PreserveVideoMemoryAllocations=1"
```

检查你的挂起模式：

```
cat /sys/power/mem_sleep
```

如果输出：

```
s2idle [deep]
```

那就是设置好了，如果不是，且在之后的测试中电源管理不起效，则需要添加内核参数，以`grub`的添加方式为例，编辑`/etc/default/grub`，在`GRUB_CMDLINE_LINUX_DEFAULT=`后的参数中追加`mem_sleep_default=deep`，类似于下面这样，注意我的配置仅供参考，切勿直接复制粘贴：

```
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet nouveau.modeset=0
mem_sleep_default=deep"
```

刷新grub：

```
grub-mkconfig -o /boot/grub/grub.cfg #等价于update-grub
```

重启你的电脑使配置生效。

注意：如果重启后卡在启动`lightdm`进程上，请自行想办法修改`/lib/udev/rules.d/80-nvidia-pm.rules`，注释除function 0外的其它设备的启用代码并删除设备。

搞定！到了欢乐的测试时间了！

1.挂起恢复后，使用`nvidia-smi`命令查看是否有Xorg进程

这是我的测试结果：

将笔记本合盖，重新打开，电源管理功能正常运行（有Xorg进程），YES！；

2.使用需要独显的应用程序，比如blender，由于blender没有自己去请求调用独显，所以需要这样运行：

```
prime-run blender
```

如果你安装了`gamemode`，不妨一起开启，享受极致性能：

```
export GAMEMODERUNEXEC=prime-run && gamemoderun blender
```

如果开启失败，多半是因为缺少图形库，请自行安装必要的Vulkan库和OpenGL库。

使用命令：

```
cat /proc/driver/nvidia/gpus/0000:01:00.0/power
```

验证自动切换功能是否有效，只需看这一行：

```
Video Memory:                 Off
```

我的测试结果：

启用blender前：Off

启用blender后：Active

关闭blender后：Off

验证成功，自动切换功能运行正常！

问题

Q：为啥我运行

```
cat /proc/driver/nvidia/gpus/0000:01:00.0/power
```

输出全是问号?

A：检查独显上是否跑了Xorg，当Xorg没有运行的时候，系统无法检查到独显状态。

Q：承上题，那我为了检查状态岂不是一定要开着Xorg喽？服务器怎么办?

A：参考这篇文章。

Q：为啥我开了自动切换功能还是会有Xorg进程，那这岂不是等于没关?

A：先纠正我当时一个认识上的错误：以为真自动切换不会有Xorg进程跑在独显上。但据官方文档说明，PRIME的实现必须靠一个Xorg进程跑在独显上，否则会让方案失效。但这样做又可以彻底关闭GPU电源，所以我也不知道它是怎么实现的（我不是显卡方面的专家！）如果你硬是要去掉进程，当然可以，但是PRIME就失效了。

执行：

```
sudo nano /etc/X11/xorg.conf
```

添加如下内容：

```
Section "ServerLayout"
  Identifier "layout"
  Screen 0 "iGPU"
  Option "AllowNVIDIAGPUScreens" "false"
EndSection

Section "Device"
  Identifier "iGPU"
  Driver "modesetting"
EndSection

Section "Screen"
  Identifier "iGPU"
  Device "iGPU"
EndSection

Section "Device"
  Identifier "dGPU"
  Driver "nvidia"
EndSection
```

注销或杀死Xorg，再执行`nvidia-smi`，你会发现没有Xorg进程了。