1. Linux系统常用操作

1. Linux系统常用操作

1.1. 系统常用命令

1.1.1. 查找进程运行位置

```
1、通过ps或者top命令查看运行的进程的pid
命令：ps -aux|grep java
或
命令：top
2、获取进程的pid后，然后使用命令ls -l /proc/${pid}，这个命令可以列出该进程的启动位置。
ll /proc/22551
```

1.1.2. 查看历史命令

```
命令：history
```

1.1.3. 根据进程PID查询服务位置

```
命令：ps -aux |grep -v grep|grep 28990
```

1.1.4. 配置history显示操作时间

```
1、如需永久显示执行时间需要修改.bash.rc文件，该文件在/root目录下
执行命令：vim  /root/.bashrc
2、在文件最后添加一行 配置
export HISTTIMEFORMAT='%F %T '
3、设置立即生效
执行命令： source /root/.bashrc 立即生效
4、查看验证
 执行命令：history 查看历史命令并展示操作时间
```

1.1.5. 远程复制【SCP命令】

> 把本地文件拷贝到远程主机

```
scp  /home/backConfig/nginx/nginx-config-240512.tar.gz
root@192.168.110.102:/etc/nginx
说明:
scp /需要拷贝文件的绝对路径/拷贝的文件名称  用户名@IP地址:/目标主机目录
```

> 把远程主机文件拷贝到本地

```
scp root@192.168.110.102:/etc/nginx/check-config.sh /etc/nginx
说明:
scp 用户名@IP地址:/需要拷贝文件的绝对路径/拷贝的文件名称  /目标主机目录
```

> 带端口文件复制

```
scp -P 2210 server.tar.gz  root@192.168.2.178:/home/app/fileserver
说明:
scp -P表示使用2222端口进行传输/需要拷贝文件的绝对路径/拷贝的文件名称  用户名@IP地址:/目标主机
目录
```

> SCP错误处理解决方案

```
#错误信息:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:v1tvR6aU2PCpf1OtSLJvqCT7AzPRW1Sl7RpjC4MyroE.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /root/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/root/.ssh/known_hosts" -R "192.168.110.102"
ECDSA host key for 192.168.110.102 has changed and you have requested strict
checking.
Host key verification failed.

#解决方案
#移除旧的密钥:如果你确认远程主机的密钥确实已经更改,或者你确定连接是安全的,你可以移除
known_hosts文件中旧的密钥条目。使用以下命令:

ssh-keygen -f "/root/.ssh/known_hosts" -R "192.168.110.102"

重新连接:在移除旧的密钥之后,再次尝试使用scp命令进行连接。SSH将提示你接受新的主机密钥。
```

1.1.6. 查看系统版本命令

```
1、# uname －a （Linux查看版本当前操作系统内核信息）

2、# cat /proc/version （Linux查看当前操作系统版本信息）

3、# cat /etc/issue 或cat /etc/redhat-release（Linux查看版本当前操作系统发行版信息）

4、# cat /proc/cpuinfo （Linux查看cpu相关信息，包括型号、主频、内核信息等）

5、# getconf LONG_BIT （Linux查看版本多少位）

6、# lsb_release -a
```

1.1.7. linux 查看cpu架构和内核命令

1. **查看 CPU 架构**:

   uname -m：显示当前机器的 CPU 架构。例如，x86_64 表示 64 位架构。

2. **查看所有 CPU 相关信息**:

   lscpu：列出 CPU 架构、型号、核心数等详细信息

3. **查看内核版本**:

   uname -r ：显示当前运行的内核版本号。
   uname -v ：显示内核版本和编译选项。

4. **查看详细内核信息**:

   /proc/version：这个文件包含了内核版本和编译日期等信息。

5. **查看系统信息**:

   hostnamectl：显示系统信息，包括内核版本和操作系统名称。

6. **查看 CPU 缓存信息**:

   cat /proc/cpuinfo：显示 CPU 的详细信息，包括型号、缓存大小等。

7. **查看 CPU 频率**:

   cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_cur_freq：显示当前 CPU 核心的频率。

8. **查看 CPU 使用情况**:

   top 或 htop：实时显示 CPU 使用情况和其他系统信息。

9. **查看 CPU 温度**（如果可用）：

   sensors：显示 CPU 温度和其他硬件传感器信息。

10. **查看 CPU 微码信息**：

`dmesg | grep microcode`：显示 CPU 微码更新信息。

1.1.8. 进程查询过滤自己

进程查询过滤自己可以通过在 grep 命令中使用 -v 选项来实现

```
ps -ef | grep 'ssh -o StrictHostKeyChecking' | grep -v 'grep'
```

命令的说明：
ps -ef：列出所有运行中的进程。
grep 'ssh -o StrictHostKeyChecking'：过滤出包含字符串 'ssh -o StrictHostKeyChecking' 的行。
grep -v 'grep'：从上一步的结果中排除包含 'grep' 的行。

1.1.9. linux 使用一条命令根据端口杀掉进程

```
lsof -i :8080 | awk 'NR!=1 {print $2}' | xargs kill
```

命令说明如下：
lsof -i :8080：列出所有使用端口8080的进程。
awk 'NR!=1 {print $2}'：使用awk来处理lsof的输出，跳过第一行（通常是标题行），然后打印第二列，即进程ID。
xargs kill：将awk输出的进程ID传递给kill命令以杀死这些进程。

注意：可能需要使用sudo来获取足够的权限来杀死不属于当前用户的进程：

```
sudo lsof -i :8080 | awk 'NR!=1 {print $2}' | xargs kill
```

此外，如果想要确保进程被强制杀死，可以使用kill -9代替kill：

1.1.10. 进程查询过滤[自己]并杀掉进程

```
ps -ef | grep '[s]sh -o StrictHostKeyChecking' | awk 'NR =1 {print $1}' | xargs -r kill -9
```

命令说明：
    1、根据 grp查询进程为 'ssh -o StrictHostKeyChecking'的进程，并且获取进程ID 通过kill -9 杀死，其中 xargs -r 参数确保在没有查询到进程时不执行kill -9 命令；
    2、 ps -ef | grep '[s]sh -o StrictHostKeyChecking' ：精确匹配查询进程，过滤掉自己只查询ssh的进程，其中[s]确保匹配 ssh 而不是 grep 中的 ssh
    3、awk 'NR = 1 {print $2}' ：使用 awk 打印出除第一行的第二列，即进程ID。 'NR != 1 {print $2}' 跳过第一行，然后打印第二列； 可有多种写法可自行探索；

> 因操作系统不一样，根据返回结果判断第一列是否为进程id如果不是则自动获取第二列

```
ps -ef | grep '[s]sh -o StrictHostKeyChecking' | awk '{
    if ($1 ~ /^[0-9]+$/)
        print $1
    else if ($2 ~ /^[0-9]+$/)
        print $2
}' | xargs -r kill -9
```

注意：语法格式必须缩进必须一致，否则提示语法错误，如果想写成一行，需要处理空格及换行符等保持一致即可实现。

```
String closeCmd="ps -ef | grep '[s]sh -o StrictHostKeyChecking' | awk '{\n if ($1 ~ /^[0-9]+$/)\n  print $1\n else if ($2 ~ /^[0-9]+$/)\n  print  $2\n }' | xargs -r kill -9\n ";
```

1.1.11. linux格式化输出时间

```
# 显示当前日期和时间（默认格式）
date

# 显示当前日期（YYYY-MM-DD）
date +%Y-%m-%d

# 显示当前时间（HH:MM:SS）
date +%H:%M:%S

# 显示完整的日期和时间（YYYY-MM-DD HH:MM:SS）
date +"%Y-%m-%d %H:%M:%S"

# 自定义格式，例如“今天是星期三，2023年4月5日”
date +"%A, %Y年%m月%d日"
```

> 常用格式说明符

```
%Y - 年份（4位数字）
%m - 月份（01-12）
%d - 日（01-31）
%H - 小时（00-23）
%I - 小时（01-12）
%M - 分钟（00-59）
%S - 秒（00-59）
%b - 简写月份名（例如：Jan）
%B - 完整月份名（例如：January）
%A - 完整星期名（例如：Sunday）
%a - 简写星期名（例如：Sun）
%Z - 时区名（例如：EST）
%z - +hhmm 格式的时区（例如：+0000）
```

1.2. Linux系统程序安装

1.2.1. Linux软件参考源

> unubtu 软件仓库源官网地址

```
https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/
```

> 清华源 unubtu 20.4 第三方镜像源

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal main restricted universe
multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal main restricted
universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-updates main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-updates main
restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-backports main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-backports main
restricted universe multiverse

# 以下安全更新软件源包含了官方源与镜像站配置，如有需要可自行修改注释切换
deb http://security.ubuntu.com/ubuntu/ focal-security main restricted universe
multiverse
# deb-src http://security.ubuntu.com/ubuntu/ focal-security main restricted
universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-proposed main
restricted universe multiverse
# # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-proposed main
restricted universe multiverse
```

1.2.2. Linux查看USB串口命令

```
 ls /dev/ttyU*
```

1.2.3. Linux应用安装

```
sudo apt install  net-tools
切换用户：
命令：su root
命令 apt install vim
```

1.2.4. Linux安装yum

```
ubuntu20.4 安装yum
命令：  apt-get install aptitude
linux centos
命令：  sudo apt-get install yum
```

1.2.5. Linux安装deb 应用包

```
#安装命令：
sudo dpkg -i upgrade_1.0.2_amd64.deb
#卸载命令
sudo dpkg -r upgrade
```

1.2.6. Linux搭建Go环境

> 下载 go 环境包

```
1、创建安装目录，命令：mkdir /home/go
2、下载go安装包：
命令：wget https://golang.google.cn/dl/go1.18.5.linux-amd64.tar.gz
3、解压go安装包，命令：sudo tar -zxvf go1.18.5.linux-amd64.tar.gz
```

> 配置Go环境变量

```
配置环境变量：
1、执行环境变量配置文件编辑命令：sudo vim /etc/profile
2、在文件最后增加如下配置：
export GOROOT=/home/go/go
export GOPROXY=https://goproxy.cn
export PATH=$PATH:$GOROOT/bin
3、执行命令使文件生效
命令：source /etc/profile
```

1.2.7. Linux搭建Java环境

1.2.7.1. 下载地址

```
1、JDK-1.8版本可下载地址
https://download.oracle.com/otn/java/jdk/8u341-
b10/424b9da4b48848379167015dcc250d8d/jdk-8u341-linux-x64.tar.gz?
AuthParam=1664336523_af7242d78f93b857e82175f39585cb2c

2、官方下载地址
JDK 8下载路径：https://www.oracle.com/java/technologies/downloads/#java8
JDK 11下载路径：https://www.oracle.com/java/technologies/downloads/#java11
JDK 16下载路径：https://www.oracle.com/java/technologies/downloads/#java16
JDK 17下载路径：https://www.oracle.com/java/technologies/downloads/
```

1.2.7.2. JDK部署安装配置

```
# 1、创建java目录
cd /usr/local/
sudo mkdir java

#2、 解压压缩包
sudo tar -zxvf jdk-8u341-linux-x64.tar.gz -C /usr/local/java/
# 3、检查版本
cd /usr/local/java/jdk1.8.0_301/bin
./java -version
```

1.2.7.3. JDK配置环境变

> 1、环境变量参数配置格式

```
# JDK 8
    export JAVA_HOME=/usr/local/java/jdk1.8.0_341
    export JRE_HOME=$JAVA_HOME/jre
    export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib
    export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

    # JDK 11
    export JAVA_HOME=/usr/local/java/jdk-11.0.12
    export CLASSPATH=$JAVA_HOME/lib
    export PATH=$PATH:$JAVA_HOME/bin
```

> 2、配置系统级环境变量

```
#编辑文件
vim /etc/profile

#在文件末尾添加变量
export JAVA_HOME=/usr/local/java/jdk1.8.0_341
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

#更新配置使其生效
source /etc/profile
```

1.3. 系统磁盘

1.3.1. 查看磁盘挂载情况

```
命令：fdisk -l
 df -h             #显示目前在Linux系统上的文件系统的磁盘使用情况统计。
 lsblk            #列出块设备信息（df -h不能看到的卷）
 mount             #挂载命令
 先分区在挂载，磁盘挂载命令：  mount /dev/sda /database


 硬盘级别：数据备份操作
 命令 sudo dd if=/dev/sda of=/dev/sdc bs=64K conv=noerror,sync status=progress
 说明：  if= 表示输入文件（源硬盘）
        of= 表示输出文件（目标硬盘）
        bs=64K 设置块大小为 64KB ，如果磁盘读写性能不好可以设置为4KB
        conv=noerror,sync 表示在遇到错误时继续，并同步输入输出
        status=progress 显示执行进度
```

1.3.2. 查看磁盘空间使用情况

```
命令：df -h
ll -h
```

```
sudo du --max-depth=1 -h

查看当前目录下一级子文件和子目录占用的磁盘容量。
du -lh --max-depth=1
df [选项] [磁盘或分区]
选项  说明
-a  显示所有文件系统
-B <块大小>     指定显示时的块大小
-h  以容易阅读的方式显示
-H  以1000字节为换算单位来显示
-i  显示索引字节信息
-k  指定块大小为1KB
-l  只显示本地文件系统
-t <文件系统类型> 只显示指定类型的文件系统
-T  输出时显示文件系统类型
- -sync 在取得磁盘使用信息前，先执行sync命令
#查看当前目录下文件大小
ls -l --block-size=M
```

1.4. 系统文件

1.4.1. 批量创建文件

```
命令： mkdir -p /soft/appliaction/coframe_unorg_server/nginx/{conf,html,logs}
```

1.4.2. 文件授权可执行权限

```
命令 ： chmod u+x start.sh
```

1.4.3. 查看当前目录及子目录大小

```
命令：du -h --max-depth=1*
说明：
当使用ls -ll、ls-lh命令进行查看时，当使用ls -ll，会显示成字节大小，而ls- lh会以KB、MB等为单位进行显示
查询当前目录总大小可以使用du -sh，其中s代表统计汇总的意思，即只输出一个总和大小
通过命令du -h ?Cmax-depth=0 *，可以只显示直接子目录文件及文件夹大小统计值
如果只想查看指定目录的总大小，可以使用du -sh 目录名称
对于指定文件夹也可以指定显示层次深度，如du -h --max-depth=0 software/及du -h --max-depth=1 software/
```

1.4.4. 查看文件夹所在分区

```
执行命令： df /maAnShan
```

1.4.5. 格式化查看文件大小

```
ls -s --block-size=G
```

1.5. 系统资源

1.5.1. 查看占用内存最高的进程

```
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head
```

1.5.2. 查看系统cpu/内存使用情况

```
命令：free -h

执行结果：
              total        used        free      shared  buff/cache   available
Mem:           15Gi        12Gi       303Mi        68Mi       2.2Gi       2.1Gi
Swap:            0B          0B          0B

结果说明：
    total: 内存总数

    used: 已经使用内存数

    free: 完全空闲内存

    shared: 多个进程共享的内存

    buffers: 用于块设备数据缓冲，记录文件系统metadata（目录，权限，属性等)

    cached: 用于文件内容的缓冲

    available：真正剩余的可被程序应用的内存数
```

1.5.3. 查看网络端口占用情况

```
命令：
netstat -nltp|grep 17000
netstat -tunlp | grep 80
fuser -n tcp 80

查看守护进程
ps -A | grep nginx
根据名称杀死进程
pkill nginx
```

1.5.4. 系统备份

```
sudo apt install timeshift
```

1.5.5. 系统恢复

```
通过工具 timeshift
```

1.5.6. 统计当前服务中有多少个连接

```
ss -s && free -g
```

1.5.7. 查看系统内存使用情况

```
ps aux | head -1;ps aux |grep -v PID |sort -rn -k +4 | head -20
```

1.5.8. 统计请求连接并发数量

```
ss -s
```

```
root@ubuntu-ThinkStation-K:~# ss -s
Total: 1285
TCP:   784 (estab 140, closed 482, orphaned 0, timewait 37)

Transport Total     IP        IPv6
RAW       2         2         0
UDP       8         5         3
TCP       302       174       128
INET      312       181       131
FRAG      0         0         0
```

1.6. 系统权限

1.6.1. 使用root 给其他用户授权

```
sudo chown -R yuyu /mnt/ssd1/yuyu
```

1.6.2. 文件授权可执行权限

```
命令 ： chmod u+x start.sh
```

1.6.3. 修改root用户密码

```
sudo passwd root
新的 密码：
重新输入新的 密码：
passwd：已成功更新密码
```

1.6.4. 查看系统是否关闭自动更新

```
cat /etc/apt/apt.conf.d/20auto-upgrades
```

如果该文件不存在，或者包含以下内容，则表示自动更新是关闭的：

```
root@ubuntu-ThinkStation-K:~# cat /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Download-Upgradeable-Packages "0";
APT::Periodic::AutocleanInterval "0";
APT::Periodic::Unattended-Upgrade "0";
```

1.6.5. 关闭自动更新

```
sudo apt-mark hold linux-image-generic linux-headers-generic


如果文件存在并包含其他值，则表示自动更新设置已启用。
如果您想确保自动更新是关闭的，可以创建或编辑上述文件，并确保设置为“0”。如果文件不存在，您可以创建它，并添加以下内容：
echo 'APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Download-Upgradeable-Packages "0";
APT::Periodic::AutocleanInterval "0";
APT::Periodic::Unattended-Upgrade "0";' | sudo tee /etc/apt/apt.conf.d/20auto-upgrades
```

1.6.6. 查看系统启动日志

```
dmesg
```

1.7. [应用服务] -> 设置开机自启动

1.7.1. 自定义shell脚本加入自启动

```
方法一：
1、在服务目录下创建脚本：例如：/ser/app/start.sh
#!/bin/bash
#chkconfig:2345 10 90
#description:resind
# 启动需要执行的命令或代码逻辑，注意：以上三行必须添加

cp /root/hello/start.sh /etc/init.d/start.sh

2、将shell脚本拷贝到/etc/init.d 的文件夹下
cp /ser/app/start.sh /etc/init.d/start.sh

3、给脚本执行权限：chmod +x /etc/init.d/start.sh

4、 加入开机启动,进/etc/init.d/目录下
执行命令：chkconfig --add start.sh
执行命令：chkconfig start.sh on
查看wei.sh开机自启动情况： chkconfig --list wei.sh
（注意非加入自启动而是手动启动这样：/etc/init.d/wei.sh start）

方法二：
1.写一脚本，start.sh 放在、etc/init.d、下面。即/etc/init.d/start.sh
2，加权限 chmod +x /etc/init.d/start.sh
3.写入到/etc/rc.local开机自启动文件里：
echo “/etc/init.d/start.sh”>> /etc/rc.local
4 chmod +x /etc/rc.local
```

1.7.2. unubut20.4 设置开启自启动 ☆

> 第1步： 进入文件目录[rc-local.service]

```
rc-local.service 文件位置
执行命令：cd /lib/systemd/system
```

> 第2步：修改rc-local.service的权限

```
执行命令：sudo chmod 777 /lib/systemd/system/rc-local.service
```

> 第3步：编辑 rc-local.service 脚本

```
执行命令：vim rc-local.service
```

> 第4步：编写脚本内容

```
在文件最后面 添加以下配置项
[Install]
WantedBy=multi-user.target
Alias=rc-local.service



#完整配置文件如下：

#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

# This unit gets pulled automatically into multi-user.target by
# systemd-rc-local-generator if /etc/rc.local is executable.
[Unit]
Description=/etc/rc.local Compatibility  # 服务的描述，方便人们阅读
Documentation=man:systemd-rc-local-generator(8)  # 一组用空格分隔的文档URI列表，这些文档是对此单元的详细说明
ConditionFileIsExecutable=/etc/rc.local  # 检测指定的路径是否存在并且是一个可执行文件，必须使用绝对路径
After=network.target  # 定义启动顺序。
                      # Before=xxx.service,代表本服务在xxx.service启动之前启动;
                      # After=xxx.service,代表本服务在xxx.service之后启动。

[Service]
Type=forking
ExecStart=/etc/rc.local start  # 指定启动单元的命令或者脚本
TimeoutSec=0
RemainAfterExit=yes  # 如果设置这个选择为真，服务会被认为是在激活状态
GuessMainPID=no

[Install]
WantedBy=multi-user.target  # WantedBy：表示该服务所在的 Target(服务组)
Alias=rc-local.service
```

> 第5步：设置开启自启动

```
执行命令：systemctl enable rc-local.service
```

> 第6步：编写启动脚本

```
编辑命令： vim /etc/rc.local

#启动脚本内容如下：
#!/bin/sh -e
# rc.local 自己的启动脚本

cd /app/app_server/
./start.sh &

cd /app/fileserver/
./start.sh &

cd /app/pyctrlAPI/bin/api_http/
./start.sh &

cd /app/pyctrlAPI/bin/api_websocket/
./start.sh &


exit 0
EOF
```

> 第7步：文件授权

```
sudo chmod +x /etc/rc.local
```

> 第8步：添加软连接

```
sudo ln -s /lib/systemd/system/rc.local.service /etc/systemd/system/
```

> 第9步：启动测试

```
reboot
```

1.7.3. 使用supervisor 配置开机自启动

```
安装命令：apt install supervisor
```

> 配置文件示例

编辑文件：vim /etc/supervisor/conf.d/appServer.conf

```
[program:appServer]
directory=/application/app_server/
command=/application/app_server/start.sh
autostart=true
autorestart=true
startretries=100
redirect_stderr=true
stdout_logfile=/application/logs/serviceStart.log
```

> 配置说明:

```
[program:app] ; 程序名称，在 supervisorctl 中通过这个值来对程序进行一系列的操作
autorestart=True       ; 程序异常退出后自动重启
autostart=True         ; 在 supervisord 启动的时候也自动启动
redirect_stderr=True  ; 把 stderr 重定向到 stdout，默认 false
environment=PATH="/home/app_env/bin"  ; 可以通过 environment 来添加需要的环境变量，一种常见的用法是使用指定的 virtualenv 环境
command=python server.py  ; 启动命令，与手动在命令行启动的命令是一样的
user=ubuntu            ; 用哪个用户启动
directory=/home/app/  ; 程序的启动目录
stdout_logfile_maxbytes = 20MB  ; stdout 日志文件大小，默认 50MB
stdout_logfile_backups = 20     ; stdout 日志文件备份数
; stdout 日志文件，需要注意当指定目录不存在时无法正常启动，所以需要手动创建目录（supervisord 会自动创建日志文件）
stdout_logfile = /data/logs/usercenter_stdout.log
```

1.7.4. [系统级守护进程] -> 开机自启动

> 第1步：编写启动脚本

```
1、配置文件目录位置：/etc/systemd/system/nongkeyuan.service
2、配置文件内容：
[Unit]
Description=nongkeyuan Service
After=network.target

[Service]
Type=simple
ExecStart=sudo /root/xuke/nongkeyuan
User=root
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target

3、配置说明：
[Unit]
Description: 提供服务的描述信息，这里写的是 "nongkeyuan Service"。
After: 指定了本服务依赖的其他服务或目标（target），在这里是 network.target，意味着这个服务将在网络服务启动之后启动。
After=network.target 先后顺序，在网络启动之后启动。多服务启动如下：
#多服务启动顺序为: 网络启动之后【After=network.target】 ==> 应用服务
1【After=server1.target】==>
```

```
应用服务2【server1.target】 ==>应用服务2【server1.target】 ==> 应用服务
3【server3.target】

[Service]
Type: 指定服务的类型。simple 是最常见的类型，表示启动时执行一个单一的进程。[后台进程]
ExecStart: 定义了启动服务时要执行的命令或脚本。这里使用的是 sudo，意味着需要管理员权限来执行
/root/xuke/nongkeyuan 脚本。
User: 指定服务运行时使用的用户名，这里是 root，即以 root 用户权限运行。
Restart: 定义了服务在退出时是否需要重启。always 表示总是尝试重启服务。
RestartSec: 定义了重启服务前的等待时间，这里是 5 秒。
[Install]
WantedBy: 指定了当哪个目标（target）被激活时，本服务应该被启动。multi-user.target 表示在
系统达到多用户状态时激活服务，这是大多数服务的标准配置。
```

> 第2步：配置文件使用示例

1. **启动服务**：使用以下命令启动服务

```
shsystemctl start nongkeyuan.service
```

2. **停止服务**：使用以下命令停止服务

```
shsystemctl stop nongkeyuan.service
```

3. **查看服务状态**：使用以下命令查看服务状态

```
shsystemctl status nongkeyuan.service
```

4. **设置服务开机自启**：使用以下命令设置服务在开机时自启动

```
shsystemctl enable nongkeyuan.service
```

5. **禁用服务开机自启**：使用以下命令禁用服务在开机时自启动

```
shsystemctl disable nongkeyuan.service
```

6. **重新加载 systemd 管理器配置**：对服务文件进行更改后，使用以下命令重新加载 systemd 管理器配置

```
systemctl daemon-reload
```

7. **重启服务**：使用以下命令重启服务

```
systemctl restart nongkeyuan.service
```

8. **实操命令**

```
systemctl start nongkeyuan.service
systemctl enable nongkeyuan.service
```

9. systemd官方文档

```
systemd官方文档（systemd.service完整配置选项）
https://www.freedesktop.org/software/systemd/man/systemd.service.html
```

1.7.5. [实战配置]-> 开机自启动

> 配置服务1： 数据调度服务开机配置[最终配置]

```
[Unit]
Description=nongkeyuanrsdispatch Service
After=network.target
#After=multi-user.target
#After=user.target

[Service]
Type=simple
#ExecStart=sudo /root/xf/dispatch/start.sh
ExecStart=sudo /root/xf/dispatch/gr-dev-dispatch-server
WorkingDirectory=/root/xf/dispatch
User=root
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

> 配置服务2： 桌面应用开机配置[最终配置]

```
1、配置路径：/etc/systemd/system/nongkeyuanrsdesktop.service
2、配置文件内容：
[Unit]
Description=nongkeyuanrsdesktop Service
After=nongkeyuanrsdispatch.target

[Service]
Type=simple
#ExecStart=sudo /root/xf/desktop/bin/start.sh
ExecStart= sudo /root/xf/desktop/bin/gr-dev-mf-win
WorkingDirectory=/root/xf/desktop/bin
Environment=DISPLAY=:0
TimeoutSec=0
User=root
Restart=always
RestartSec=5

[Install]
WantedBy=graphical.target
```

> 配置服务：启动服务操作

```
#启动服务
systemctl start nongkeyuanrsdesktop.service
#设置守护进程
systemctl enable nongkeyuanrsdesktop.service
#重新加载
systemctl daemon-reload
#禁用守护进程的自动启动：
systemctl disable nongkeyuanrsdesktop.service
#查看系统日志 ,查看最新20条日志
journalctl -u nongkeyuan-pulseaudio.service -xe | tail -n 20
#查看音频服务
lsof /dev/snd/*
```

> 配置服务：服务启动操作使用命令

- systemctl start `：启动一个服务。
- `systemctl stop <service>`：停止一个服务。
- `systemctl restart <service>`：重启一个服务。
- `systemctl reload <service>`：重新加载一个服务的配置文件，使其生效。
- `systemctl enable <service>`：设置一个服务在系统启动时自动启用。
- `systemctl disable <service>`：设置一个服务在系统启动时不自动启用。
- `systemctl status <service>`：查看一个服务的状态，包括是否正在运行、最后一次的活动时间等信息。
- `systemctl is-active <service>`：检查一个服务是否正在运行。
- `systemctl is-enabled <service>`：检查一个服务是否已经设置为在系统启动时自动启用。
- `systemctl list-units`：列出所有正在运行的单位（包括服务、套接字、挂载点等）。
- `systemctl list-unit-files`：列出所有可用的单位文件（包括服务、套接字、挂载点等）

> 配置服务：设置系统启动级别【关闭开启桌面】

```
# 设置默认启动为图形界面模式
sudo systemctl set-default graphical.target
# 设置默认启动为多用户文本模式
sudo systemctl set-default multi-user.target

#查看当前的目标
systemctl get-default
#启动时直接进入特定的目标
sudo systemctl isolate graphical.target  # 立即进入图形界面模式
sudo systemctl isolate multi-user.target  # 立即进入多用户文本模式
```

> 配置服务：开机自启动system配置参数说明 -> 配置父项参数说明

1、配置父项
在systemd中，配置文件通常使用.service扩展名，并且使用INI格式。一个.service文件可以包含以下三个配置项：

[Unit]：这个配置项定义了服务的基本属性，如服务的描述、依赖关系、启动顺序等。在这个配置项中，可以设置服务的名称、描述、依赖关系、启动顺序等。

[Service]：这个配置项定义了服务的具体执行方式，如服务的启动命令、环境变量、工作目录等。在这个配置项中，可以设置服务的启动命令、环境变量、工作目录等。

[Install]：这个配置项定义了服务的安装方式，如服务的启动级别、启动顺序等。在这个配置项中，可以设置服务的启动级别、启动顺序等。

这三个配置项分别定义了服务的基本属性、具体执行方式和安装方式。

除了[Unit]、[Service]和[Install]之外，systemd还支持其他一些配置项，用于进一步定义和配置服务。以下是一些常用的配置项：

[Path]：用于监控文件或目录的变化，并在变化发生时触发服务的启动、停止或重启。

[Timer]：用于定义定时器，可以在指定的时间间隔或特定时间点触发服务的启动、停止或重启。

[Socket]：用于定义套接字，可以监听指定的网络端口或UNIX域套接字，并在有连接请求时触发服务的启动。

[Mount]：用于定义挂载点，可以在指定的文件系统挂载或卸载时触发服务的启动、停止或重启。

[Automount]：用于定义自动挂载点，可以在访问指定的文件系统路径时自动挂载该文件系统。

[Swap]：用于定义交换空间，可以在指定的交换文件或分区被启用或禁用时触发服务的启动、停止或重启。

> 配置服务：开机自启动system配置参数说明 -> 配置子项说明

配置子项
[Unit]配置子项
- Description：用于设置服务的描述信息。
- Requires：用于指定服务所依赖的其他服务。
- After：用于指定服务启动的顺序，可以设置在哪些服务之后启动。
- Before：用于指定服务启动的顺序，可以设置在哪些服务之前启动。

[Service]配置子项
- Type：定义服务单元的启动类型，决定systemd如何理解服务的启动和运行状态

1. simple
这是默认的服务类型，如果没有指定Type=，则假定为simple。对于此类型，一旦启动服务的主进程，systemd即认为服务已经启动成功。适合用于不会派生子进程并且在后台持续运行的应用程序。

2. forking
适用于传统的UNIX服务，这些服务启动时会派生一个子进程（即“fork”），然后父进程立即退出。这种模式常见于传统的守护进程。在这种情况下，systemd会追踪子进程，并以该子进程作为服务的主进程。

3. oneshot
此类型用于那些只执行一次性任务然后退出的服务。这通常用于初始化、配置或者更新一些系统设置然后结束的脚本。可以与RemainAfterExit=指令配合使用，以使得服务在任务执行完毕后仍显示为active。

4. dbus
如果服务在D-Bus上注册自己，可以使用这个类型。systemd将等待服务出现在D-Bus消息总线上才认为服务已经启动成功。适合于那些提供D-Bus接口的守护进程。

5. notify
这种类型的服务会在准备就绪时向systemd发送一个信号。这要求服务具有实现了sd_notify协议的逻辑，通过它来告知systemd其状态。只有收到了服务发出的就绪通知，systemd才会认为服务启动成功。

6. idle
idle 类型类似于simple类型，但是它会延迟服务的启动直到所有其他空闲任务都完成了。这主要用于那些希望在系统负载低时运行的服务，以避免影响到其他更重要的启动任务。

- ExecStart：用于指定服务的启动命令。
- WorkingDirectory：用于指定服务的工作目录。
- Environment：用于设置服务的环境变量。
- User：用于指定服务运行的用户。

- Group：用于指定服务运行的用户组。
[Install]配置子项
- WantedBy：用于指定服务在哪些目标（target）下启动。
WantedBy是systemd服务配置文件中的一个选项，用于指定服务所属的启动级别（target）。启动级别是一组定义了系统启动过程中要启动的服务的目标单元的集合。

在systemd中，启动级别由target单元来表示。每个target单元都是一组相关的服务单元的集合，用于定义系统在不同阶段启动时应该启动哪些服务。例如，multi-user.target是一个常见的启动级别，用于表示系统已经启动到多用户模式，此时应该启动所有与用户交互相关的服务。

WantedBy选项用于指定服务所属的启动级别。它接受一个目标单元的名称作为参数。当你将服务配置文件放置在/etc/systemd/system/目录下，并使用systemctl enable命令启用服务时，systemd会根据WantedBy选项中指定的目标单元，将服务添加到相应的启动级别中。

在上面的例子中，WantedBy=multi-user.target表示该服务应该属于multi-user.target启动级别，即系统已经启动到多用户模式时应该启动该服务。

在systemd中，有以下几个常见的启动级别（target）：

poweroff.target：系统关机时的目标单元。
rescue.target：用于系统救援的目标单元，提供了一个最小的功能集合，用于修复系统问题。
multi-user.target：多用户模式的目标单元，用于正常的多用户操作。
graphical.target：图形界面模式的目标单元，用于启动图形界面环境。
reboot.target：系统重启时的目标单元。
除了上述常见的启动级别，还有其他一些特定的启动级别，如network.target（网络启动完成后的目标单元）、default.target（默认目标单元，通常是multi-user.target或graphical.target）等。

对于普通程序来说，通常会将其配置为属于multi-user.target或graphical.target这样的启动级别，以便在系统启动到多用户模式或图形界面模式时自动启动。你可以根据你的需求选择适合的启动级别。

After=和WantedBy=区别
以After=multi-user.target和WantedBy=multi-user.target为例，它们确实在某种程度上相关，但它们服务于不同的目的，并且不是重复的功能。

After=multi-user.target:
这个配置项位于[Unit]部分，用来指定当前服务应当在哪些其他单元之后启动。换句话说，它定义了服务启动的顺序。在这个例子中，After=multi-user.target表明该服务应该在系统达到multi-user.target（多用户目标状态，类似于传统的运行级别3）之后启动。这意味着网络等基础设施应该已经设置好了。

WantedBy=multi-user.target:
这个配置项位于[Install]部分，它指定了当你使用systemctl enable命令来使服务自启动时，应该将这个服务链接到哪个目标的“wanted”目录下。WantedBy=multi-user.target表示当系统达到多用户目标状态时，这个服务应该被认为是“所需的”或“期望的”，因此应该被启动。简而言之，它用来定义服务是否应该随着特定的target自动启动。

总结来说：

After= 定义了启动顺序。
WantedBy= 定义了服务何时应该被自动启动。
两者共同工作以确保服务在正确的时间以正确的顺序启动。在实际情况中，一个服务通常需要在某个target之后启动（After=），并且希望随着这个target自动启动（WantedBy=）。因此，虽然它们涉及相同的target，但它们各自扮演了独特的角色。

1.8. 自定义Shell脚本

1.8.1. nohup启动退出控制台写法

```
1、后台运行 并且输出日志，控制台退出 应用不会退出
nohup java -jar test.jar >> nohup.log 2>&1 &
2、后台运行 并且输出日志 ，控制和应用都不会退出
2、nohup java -jar test.jar &
```

1.8.2. Java启动脚本

```
脚本名称： start.sh
#!/bin/bash
nohup java -server -Xms256M -Xmx256M -Xmn512M  -jar
/soft/zzbserver/server/coframe-boot-0.0.1-SNAPSHOT.jar &

/soft/yyxt/server/gr-coframe-boot-0.0.1-SNAPSHOT.jar
```

1.8.3. shell运行错误

> **/bin/bash^M: 解释器错误: 没有那个文件或目录**

```
./start.sh
错误： bash: ./start_ui_ai.sh：/bin/sh^M：解释器错误： 没有那个文件或目录
原因一：dos / windows 和 unix / linux 换行格式编码问题导致，dos / windows 『\n\r 』表
示换行 unix / linux 『\n』 表示换行
```

> 解决方案

```
执行命令：bash  start.sh 进行编译，查看编译以后的报错情况。
执行结果：
start.sh: 行 2: $'\r'：未找到命令

执行命令： sed -i 's/\r$//' build.sh
把 build.sh 中的\r 替换成空白！再次编译！成功！！
```

1.9. 系统配置

1.9.1. 修改系统默认Shell端口配置

```
1、首先修改配置文件
vi /etc/ssh/sshd_config
2、找到#Port 22一段，这里是标识默认使用22端口，修改为如下
找到Port 22的配置，默认是注释掉的，打开注释（删除#），在下面新增一行：Port 2222（你设置的SSH
端口号），注意不要跟现有端口号重复（新增一行的目的是防止新的端口登录不上）
Port 22
Port 50000
3、然后保存退出
执行 systemctl restart sshd
4、如果开启防火墙则在编辑防火墙配置
防火墙配置：vi /etc/sysconfig/iptables
5、启用50000端口。
执行/etc/init.d/iptables restart
```

6、使用ssh工具连接50000端口，来测试是否成功。如果连接成功了，则再次编辑sshd_config的设置，将里边的Port22删除，即可。
之所以先设置成两个端口，测试成功后再关闭一个端口，是为了方式在修改conf的过程中，万一出现掉线、断网、误操作等未知情况时候，还能通过另外一个端口连接上去调试以免发生连接不上必须派人去机房，导致问题更加复杂麻烦。

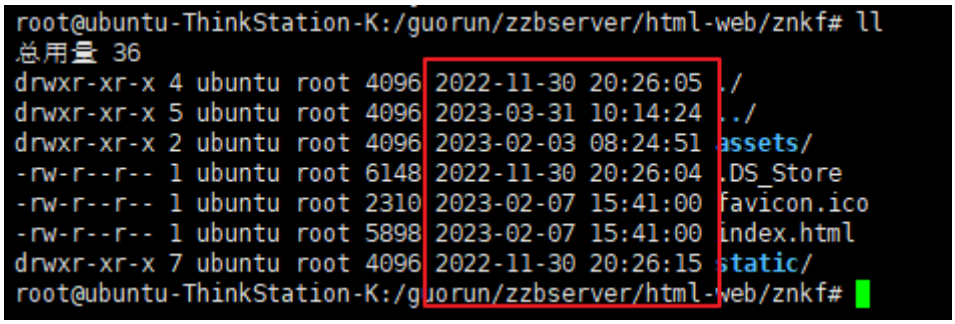
1.9.2. 系统时间配置

> 设置文件属性时间格式

*临时设置：
 直接在命令中执行即可： `export TIME_STYLE='+%Y-%m-%d %H:%M:%S'`

*永久有效配置：
1、编辑配置文件：vim /etc/profile
2、在文件最后添加如下配置：
```
export TIME_STYLE='+%Y-%m-%d %H:%M:%S'
```
3、保存并设置文件生效
```
source /etc/profile
```
4、查看验证
通过命令：ll 可查看文件属性格式

输出结果：



> 查看命令直接显示时间

```
ls -l --time-style=long-iso
```

1.9.3. 系统输入法配置

```
sudo cp /usr/share/applications/fcitx.desktop /etc/xdg/autostart/

fcitx-config-gtk3

卸载所有配置文件
sudo apt-get --purge remove fcitx

sudo apt autoremove

~/.config/fcitx
~/.config/fcitx/config
```

快捷键设置：

TriggerKey=CTRL_SHIFT_LCTRL：设置触发输入法的快捷键为 Ctrl + Shift + 左Ctrl。

IMSwitchKey=True：允许使用快捷键在不同的输入法之间滚动切换。
IMSwitchHotkey=CTRL_SHIFT：设置在输入法之间滚动的快捷键为 Ctrl + Shift。
ReloadConfig=CTRL_5：设置重新加载配置文件的快捷键为 Ctrl + 5。
VKSwitchKey=CTRL_ALT_B：设置切换虚拟键盘的快捷键为 Ctrl + Alt + B。
PuncSwitchKey=CTRL_.：设置切换全角/半角标点的快捷键为 Ctrl + 点号。
程序设置：

DelayStart=0：设置 Fcitx 启动时的延迟时间（单位为秒）。
ShareStateAmongWindow=No：设置不同窗口间是否共享输入法状态。
DefaultInputMethodState=Inactive：设置默认的输入法状态为非激活状态。
输出设置：

HalfPuncAfterNumber=True：设置在数字后输入半角标点。
RemindModeDisablePaging=True：设置在提醒模式下禁用翻页。
SendTextWhenSwitchEng=True：设置在切换到英文输入法时提交预编辑文本。
CandidateWordNumber=5：设置候选词数量。
外观设置：

ShowInputWindowAfterTriggering=True：设置在触发输入法后显示输入法窗口。
ShowInputWindowWhenFocusIn=False：设置在获得焦点时不显示输入法窗口。
ShowInputWindowOnlyWhenActive=True：设置仅在输入法激活时显示输入法窗口。
输入法设置：

Name=fcitx-pinyin：添加了拼音输入法。
如何应用这些配置：

创建或编辑配置文件：

打开或创建 ~/.config/fcitx/config 文件。如果文件不存在，你可以创建它。
复制并粘贴配置：

将你提供的配置内容复制并粘贴到 ~/.config/fcitx/config 文件中。
调整配置：

根据你的需要调整配置。例如，如果你想要更改触发输入法的快捷键，找到 TriggerKey 并修改它。
保存并关闭文件：

保存更改并关闭文件。
重新启动 Fcitx：

为了让配置生效，你需要重新启动 Fcitx。你可以使用以下命令：
bash
fcitx -r
或者完全重启 Fcitx：
bash
killall fcitx
fcitx

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      addresses: [192.168.2.172/24]
```

```
      optional: true
      routes:
      - to: default
        via: 192.168.2.1
      nameservers:
        addresses: [114.114.114.114]
```

1.9.4. 系统防火墙设置

1.9.4.1. Centos 防火墙操作

```
centos 7 firewall(防火墙)开放端口/删除端口/查看端口 1.firewall的基本启动/停止/重启命令

 #centos7启动防火墙
systemctl start firewalld.service
#centos7停止防火墙/关闭防火墙
systemctl stop firewalld.service
#centos7重启防火墙
systemctl restart firewalld.service


#设置开机启用防火墙
systemctl enable firewalld.service
#设置开机不启动防火墙
systemctl disable firewalld.service
2.新增开放一个端口号

 firewall-cmd --zone=public --add-port=80/tcp --permanent
#说明:
#-zone #作用域
#-add-port=80/tcp #添加端口，格式为：端口/通讯协议
#-permanent 永久生效，没有此参数重启后失效

#多个端口:
firewall-cmd --zone=public --add-port=80-90/tcp --permanent
注意:新增/删除操作需要重启防火墙服务. 其他PC telnet开放的端口必须保证本地 telnet 127.0.0.1
端口号 能通。本地不通不一定是防火墙的问题。 查看本机已经启用的监听端口:

 #centos7以下使用netstat -ant,7使用ss
ss -ant
3.查看

 #centos7查看防火墙所有信息
firewall-cmd --list-all
#centos7查看防火墙开放的端口信息
firewall-cmd --list-ports
4.删除

#删除
firewall-cmd --zone=public --remove-port=80/tcp --permanent
(adsbygoogle = window.adsbygoogle || []).push({});
```

> **卸载命令:**

```
先关闭服务，清空规则
systemctl disable firewalld
systemctl stop firewalld
systemctl mask firewalld

卸载firewalld
apt remove firewalld
```

1.9.4.2. Ubuntu 防火墙操作

```
1、简介
ubuntu20.04版本默认防火墙配置工具是ufw，主要为简化 iptables 防火墙配置而开发。
1.1 重安装
1.执行apt update命令更新包列表。

命令：apt update

2.执行apt install ufw命令下载ufw。
命令：apt install ufw
1.2 查看防火墙状态
1.执行ufw status命令查看防火墙状态，默认关闭。

命令：ufw status
Status: inactive
2.执行ufw status verbose命令详细查看防火墙状态。

命令： ufw status verbose
Status: inactive
3.执行ufw status numbered命令查看编号格式。

命令：ufw status numbered
Status: inactive
1.3 开启/关闭防火墙
1.3.1 开启防火墙
1.执行ufw enable命令开启防火墙。

命令： ufw enable
Firewall is active and enabled on system startup
1.3.2 关闭防火墙
1.执行ufw disable命令关闭防火墙。

命令： ufw disable
Firewall stopped and disabled on system startup
1.3.3 重启防火墙
1.执行ufw reload命令重启防火墙。

命令： ufw reload
Firewall reloaded
1.4 开放/关闭端口
1.4.1 开放端口
1.执行ufw allow 22命令开放指定端口，这里以SSH为例。

命令： ufw allow 22
Rule added
Rule added (v6)
2.也可以允许从特定主机或网络访问端口，类似于网络上的高级ACL中的rule。
```

```
命令： ufw allow proto tcp from 192.168.100.0/24 to any port 7946
Rule added
命令： ufw allow proto udp from 192.168.100.0/24 to any port 7946
Rule added
1.4.2 关闭端口
1.执行ufw deny 22命令关闭指定端口，这里以SSH为例。

命令： ufw deny 22
2.也可以拒绝从特定主机或网络访问端口，类似于网络上的高级ACL中的rule。

命令： ufw deny proto tcp from 192.168.100.0/24 to any port 7946
1.4.3 查看所有开放端口
1.执行netstat -aptn命令查看所有开放端口。

命令： netstat -aptn
```

1.9.4.3. firewall 防火墙

```
1、查看所有规则
firewall-cmd --zone=public --list-rich-rules
2、根据网段添加规则
firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source
address="192.168.3.0/255" accept"
3、限制指定IP访问指定端口
 firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source
address="192.168.3.69" port protocol="tcp" port="11001" accept"
4、删除指定规则
命令1：firewall-cmd --permanent --remove-rich-rule='rule family="ipv4" source
address="192.168.0.11" port protocol="tcp" port="82" accept'

命令2：firewall-cmd --permanent --remove-rich-rule="rule family=ipv4 source
address="192.168.3.0" accept"
```

1.9.5. 系统网络IP配置

1.9.5.1. 设置静态IP

> unubtu 设置静态IP地址

```
打开终端：你可以使用Ctrl+Alt+T快捷键来打开终端。
编辑网络配置文件：在终端中输入以下命令，使用管理员权限编辑网络配置文件：
bash
sudo nano /etc/netplan/01-netcfg.yaml
配置静态IP地址：在编辑的网络配置文件中，找到当前活动的网络接口（通常是以 "eth0" 或 "wlan0" 开头的行）。根据需要进行以下更改：

将 "dhcp4" 设置为 "no"（如果当前设置为 "yes"）：这表示你不使用DHCP服务器来获取IP地址。
设置 "addresses" 属性为你想要的静态IP地址，例如 "192.168.1.100"。
设置 "gateway4" 属性为你的网关IP地址。
设置 "DNS" 属性为你的DNS服务器IP地址。以下
```

1.9.5.2. 网络静态ip配置[实战]

> 配置方式1

```
#编辑配置文件：/etc/netplan/01-network-manager-all.yaml
#在文件中增加配置设置静态IP：

 network:
  ethernets:
    eno1:   # 配置的网卡的名称
      addresses: [192.168.8.143/24]   # 配置的静态ip地址和掩码
      dhcp4: false   # 关闭dhcp4
      optional: true
      gateway4: 192.168.8.1 # 网关地址
      nameservers:
        addresses: [223.5.5.7,114.114.114.114]  # DNS服务器地址，多个DNS服务器地址需要用英文逗号分隔开，可不配置
  version: 2
  renderer: NetworkManager
```

> 配置方式2

```
配置文件路径：
auto eth0
iface eth0 inet static
address 192.168.2.201
netmask 255.255.255.0
broadcast 192.168.2.255
gateway 192.168.2.1
```

1.10. 系统备份与还原

1.10.1. Ubuntu系统备份镜像

> 第1步：安装Timeshift

```
ubuntu 和 linux mint

sudo apt-add-repository -y ppa:teejee2008/ppa
sudo apt-get update
sudo apt-get install timeshif
```

> 第2步：打开timeshift设置备份与还原类型

此处选择【RSYNC】

> 第3步：在[位置]选项内选择存储的位置

每台设备安装分区不一样，按照实际情况选择，一般选择比较大的空间存储，并且最好是机械，这样不容易损坏。

> 第4步：在【计划】选项内不定期做备份

不选择周期备份的原因是因为磁盘占用空间会不断增长，需要定期清理

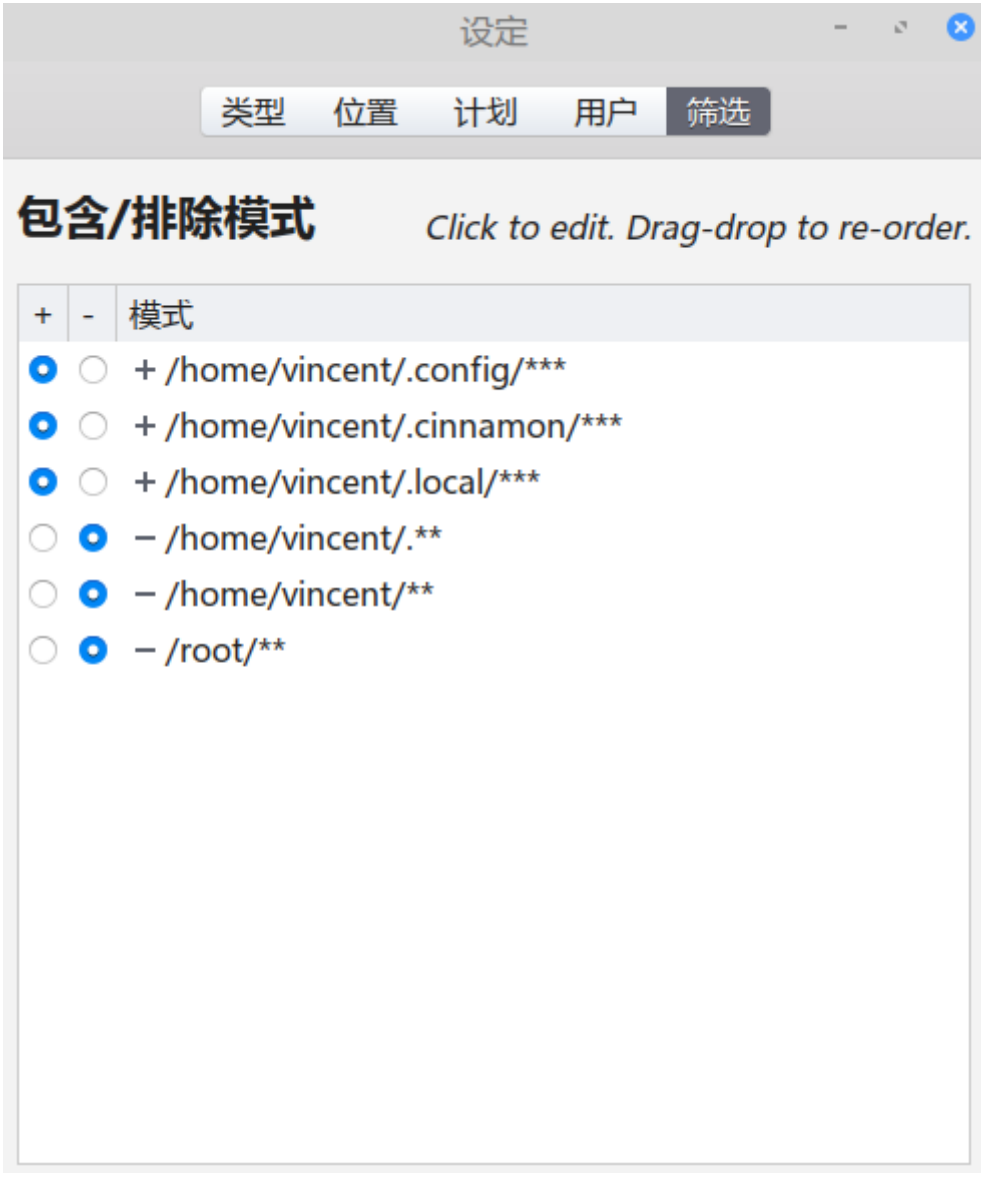
> 第5步：设置需要备份的分区

在【用户】选项内设置需要备份的分区，**root**默认就是全备份的，经测试更改也是无效的，也就是**root**必须备份。

建议这样配置：
root选择【包含一切】或者【排除一切】
home选择【排除一切】

> 第6步：自定义备份文件或文件夹

在【筛选】选项内可以自定义备份文件或文件夹



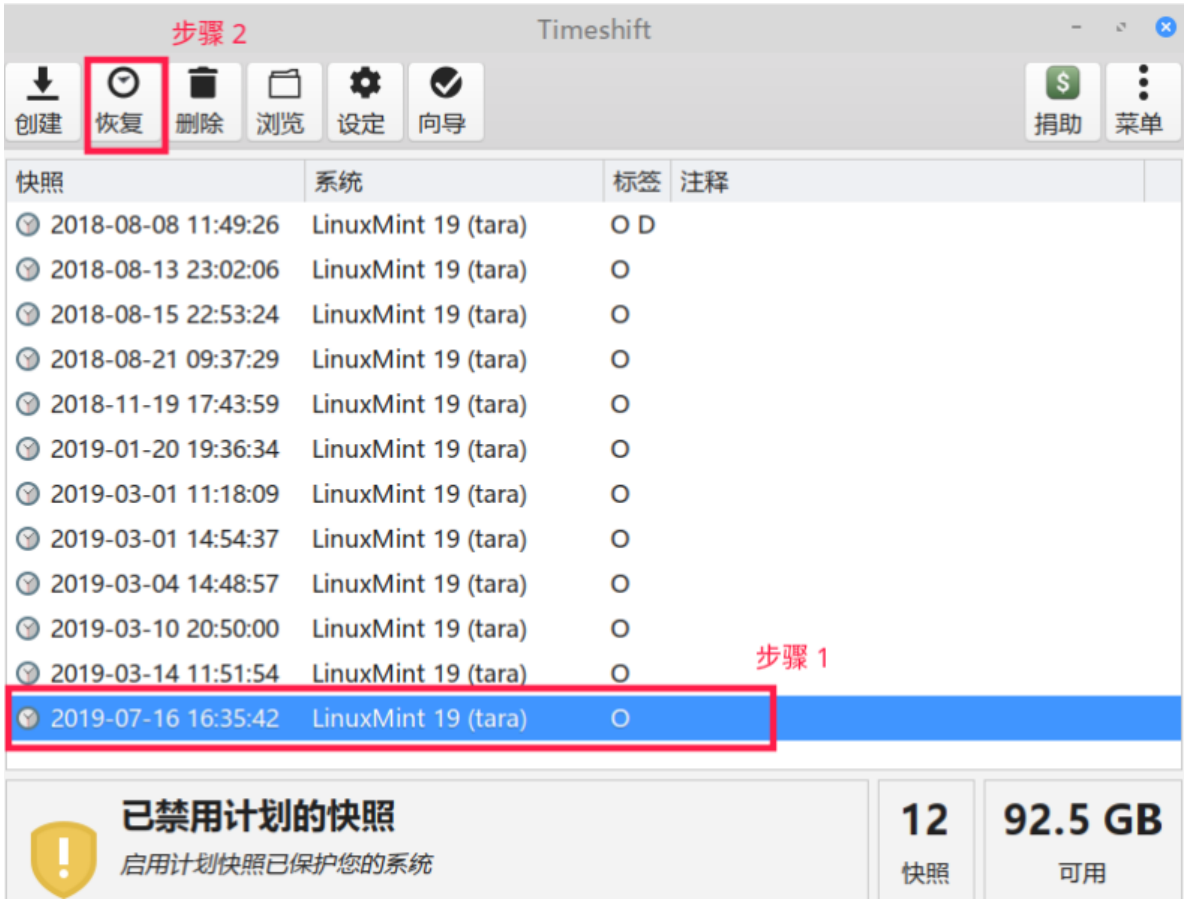
例如上图中就是/home/vincent目录下除了.config、.cinnamon、.local文件夹外都不备份。
建议这样配置：
先将home整个目录排除，添加/home/vincent/**
home目录下比较重要的如：.local、.config等，需要加入，切记要加上***三个星

> 第7步：创建快照

此时timeshift会根据时间生成一个带有时间戳的快照，此过程需要等待一段时间，并且期间不要对电脑执行任何操作。

1.10.2. ununtu 系统还原

> 第1步：能够进入系统的还原方式 -> 如果此时还能够进入系统，直接打开 **Timeshift**软件进行还原：



第2步：在弹出的窗口内点击下一步开始进行还原到选中的节点：

注：还原完成后重启电脑即可。

1.10.3. 只能进入登录界面还原方式

一般系统崩溃后不能进入桌面，但是能够进入登录界面，现象就是在登录界面输入密码后不会进入桌面，那么就要通过命令行的方式进行还原。

```
1、通过Ctrl+Alt+F1（一般是F1-F6都可）进入tty终端：
2、输入用户和密码登录
3、执行下面命令获取系统当前可以还原的节点：
    sudo timeshift --list
输出内容：
Device : /dev/sdb4
UUID   : 197c4161-abc6-4069-8544-d86594211f04
Path   : /home
Mode   : RSYNC
Device is OK
12 snapshots, 92.5 GB free

Num     Name                 Tags  Description
------------------------------------------------------------------------------
0    >  2018-08-08_11-49-26  O D
1    >  2018-08-13_23-02-06  O
2    >  2018-08-15_22-53-24  O
```

```
3    >  2018-08-21_09-37-29  O
4    >  2018-11-19_17-43-59  O
5    >  2019-01-20_19-36-34  O
6    >  2019-03-01_11-18-09  O
7    >  2019-03-01_14-54-37  O
8    >  2019-03-04_14-48-57  O
9    >  2019-03-10_20-50-00  O
10   >  2019-03-14_11-51-54  O
11   >  2019-07-16_16-35-42  O
```

> 选择一个节点进行还原

```
sudo timeshift --restore --snapshot '2019-07-16_16-35-42' --skip-grub
```

`-skip-grub` 选项为跳过grub安装，一般来说grub不需要重新安装，除非bios启动无法找到正确的grub启动项，才需要安装。
在输出的内容中依次输入【Enter】键和【y】键。
当然也可以直接输入下面的命令，忽略交互式的安装界面：

```
sudo timeshift --restore --snapshot '2019-07-16_16-35-42' --skip-grub --scripted
```

1.10.4. 无法进入系统通过U盘启动还原

当登录界面和桌面环境都无法进入时，一般系统已经彻底崩溃，此时只能通过Linux live CD进行还原。

> 进入Live系统后打开 **Timeshift** 软件，点击设置按钮，设置快照的存储位置：

1、快照的存储位置是Timeshift在做备份的时候就已经设置好的。
2、后续步骤和【如果能够进入系统】内的还原方式一样。
其实此时还可以通过命令行的方式进行还原，但是首先需要设置快照存储的位置：

```
sudo timeshift --snapshot-device /dev/sdb4
```

后续步骤和【如果只能进入登录界面】内的还原方式一样。

2. Linux 本地化安装部署

2.1. linux 安装SSH服务

2.1.1. 安装ssh

```
~$:sudo apt-get install ssh
```

2.1.2. 查看配置文件是否存在

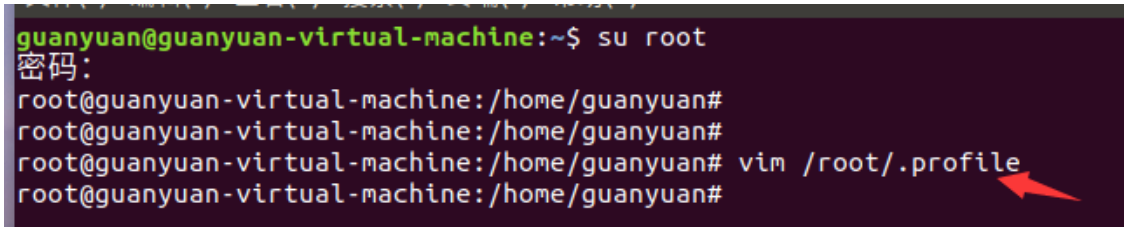
```
~$: cat /etc/ssh/sshd-config  # 默认端口号22
```

- 如果不存在

```
sudo apt-get install opensshd-server
```

2.1.3. 查看是否有sshd

- 若存在，则证明sshserver已经开启




```
ps -e/grep ssh
```

2.1.4. 直接使用root账户登录

2.1.4.1. 修改/root/.profile文件

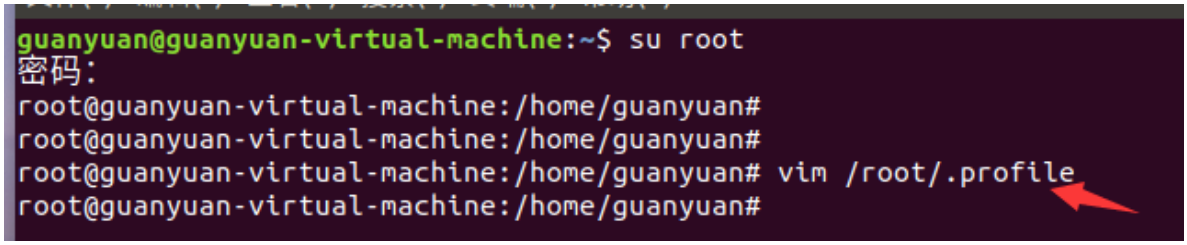
- 初次登录，设置root密码

```
sudo passwd root    # 设置root密码
```

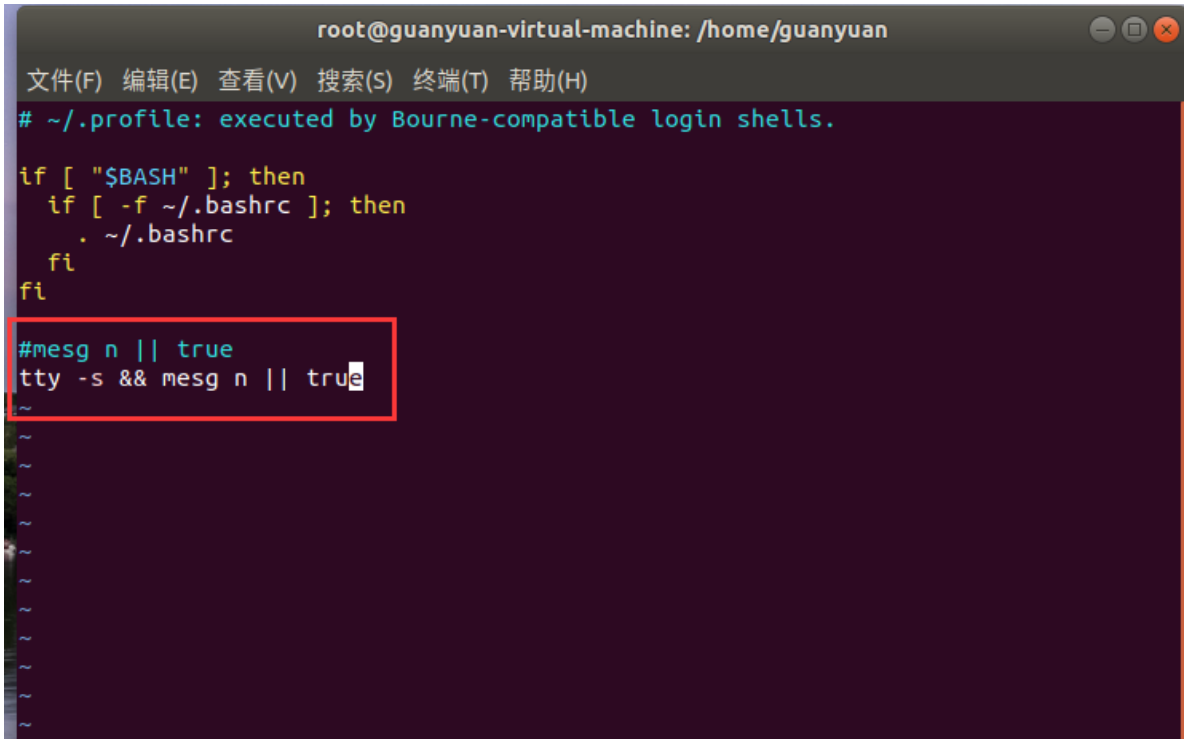
- 登录root

```
su root
```

先以普通用户登录，在终端切换root用户，然后使用vim修改`/root/.profile`文件：



注释掉文件的最后一行，然后增加



2.1.4.2. 修改gdm-autologin文件

```
vim /etc/pam.d/gdm-autologin
```

注释掉文件的第三行内容：

2.1.4.3. 修改gdm-password文件

```
vim /etc/pam.d/gdm-password
```

同样注释掉第三行内容：

```
#%PAM-1.0
auth    requisite       pam_nologin.so
#auth   required        pam_succeed_if.so user != root quiet_success
@include common-auth
auth    optional        pam_gnome_keyring.so
@include common-account
# SELinux needs to be the first session rule. This ensures that any
# lingering context has been cleared. Without this it is possible
# that a module could execute code in the wrong domain.
session [success=ok ignore=ignore module_unknown=ignore default=bad]        pam_
selinux.so close
session required        pam_loginuid.so
# SELinux needs to intervene at login time to ensure that the process
# starts in the proper default security context. Only sessions which are
# intended to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad]        pam_
selinux.so open
session optional        pam_keyinit.so force revoke
session required        pam_limits.so
session required        pam_env.so readenv=1
session required        pam_env.so readenv=1 user_readenv=1 envfile=/etc/default
/locale
```

2.1.4.4. 修改50-ubuntu.conf文件

```
vim /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf
```

在文末追加两行：

```
[Seat:*]
user-session=ubuntu
greeter-show-manual-login=true
all-guest=false
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

至此相关文件已经修改完成，可以重启测试了。

回到文章开头，我们仍然未列出，然后输入root账户及密码，发现可以正常登录。

2.2. Linux 安装GoogleChrome

2.2.1. 下载Google 浏览器

```
下载页面地址：https://www.google.cn/chrome/?standalone=1&platform=win64
1、下载Google 操作系统版本（历史版本）下载地址 86
https://www.slimjet.com/chrome/google-chrome-old-version.php
2、执行命令安装
sudo apt install linux_ubuntu_86_google-chrome-stable_current_amd64.deb
```

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

2.2.2. 安装 Google Chrome

```
1、从网上抓包安装
sudo apt install ./google-chrome-stable_current_amd64.deb
2、从本地安装
dpkg -i google-chrome-stable_current_amd64.deb
```

2.2.3. 启动 Google Chrome

```
在活动搜索栏输入"Google Chrome”，并且点击图标，启动这个应用：
```

2.2.4. 升级 Google Chrome

在安装过程中，官方 Google 软件源被添加到你的系统。你可以使用`cat`命令来验证文件内容：

```
cat /etc/apt/sources.list.d/google-chrome.list
```

2.2.5. 卸载Google Chrome

```
sudo apt-get remove google-chrome-stable
或
sudo apt purge google-chrome-stable
或通过查找软连删除
ls -alh google-chrome
进入对应目录并执行删除操作
```

2.2.6. 缺少依赖解决方法

> 安装依赖

```
dpkg -i libgdk-pixbuf2.0-0_2.40.0+dfsg-3_amd64.deb
执行修复命令
sudo apt-get check #检查依赖并修复
sudo apt --fix-broken install
```

2.2.7. 无法打开浏览器解决方法

```
在快捷键目录文件中最后一行添加以下命令：
快捷键文件目录：cd /usr/bin
查找快捷键文件： ls goole*
编辑文件：vim google-chrome
exec -a "$0" "$HERE/chrome" "$@" --user-data-dir --no-sandbox
```

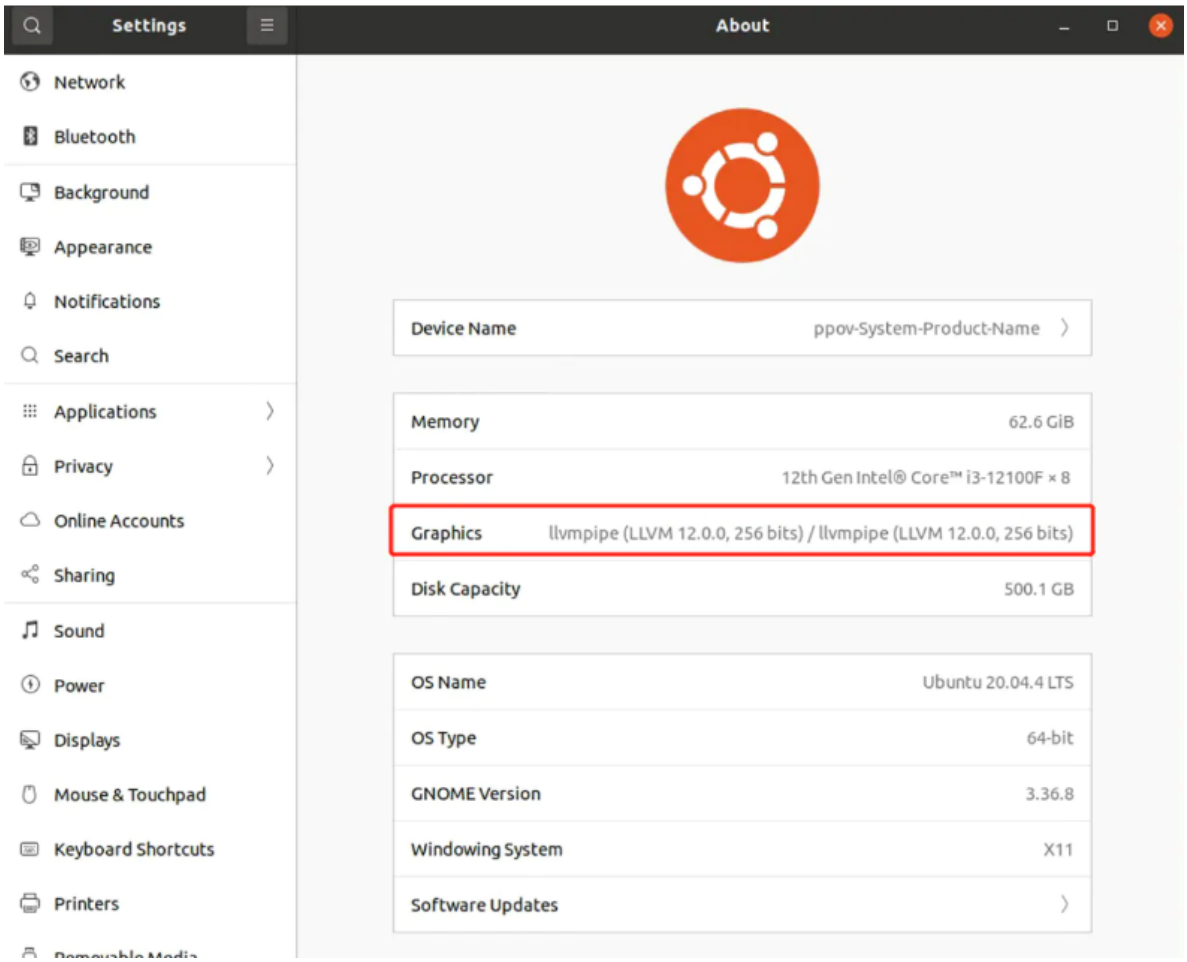
2.3. Linux安装显卡驱动

2.3.1. 安装RTX-3060显卡驱动

> Ubuntu20.04LTS安装RTX-3060显卡驱动

2.3.1.1. 查看初始化的显卡信息

进入Ubuntu20.04系统的桌面，点击左下角的“显示应用程序”->设置（settings）->关于（About），可以看到如下的界面（此界面非本人系统，是从网上找的图）：



注意： 在Graphics位置，显示的是`llvmpipe`，说明没有安装显卡驱动（没有显卡驱动，说明显卡只是简单的插在了设备上，设备并没有识别这个显卡）。

2.3.1.2. 添加ppa仓库

在命令行下列命令，将ppa添加到apt仓库中：

```
sudo add-apt-repository ppa:graphics-drivers/ppa
```

2.3.1.3. 设置Ubuntu软件的镜像源

点击左下角的"显示应用程序"->软件和更新。在ubuntu软件中，将"下载自"更换为阿里云的镜像源（换成国内的镜像源可以提高下载速度）

2.3.1.4. 设置其他软件的镜像源

将Canonical 合作伙伴（Canonical Partners）勾选上。

2.3.1.5. 更新软件列表

使用`apt-get update`命令更新软件列表：

```
sudo apt-get update
```

2.3.1.6. 查看检测到的驱动程序

使用`ubuntu-drivers devices`查看检测到的驱动程序，推荐下在带有recommended标志的驱动：

```
sudo ubuntu-drivers devices
```

```
driver   : nvidia-driver-515-server - distro non-free
driver   : nvidia-driver-520-open - distro non-free recommended
driver   : nvidia-driver-510-server - distro non-free
driver   : nvidia-driver-515 - third-party non-free
driver   : nvidia-driver-510 - third-party non-free
driver   : nvidia-driver-520 - third-party non-free
driver   : nvidia-driver-495 - third-party non-free
driver   : nvidia-driver-515-open - distro non-free
driver   : nvidia-driver-470 - third-party non-free
driver   : nvidia-driver-470-server - distro non-free
driver   : nvidia-driver-460 - third-party non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

2.3.1.7. 安装驱动程序并重启

> 安装NVIDIA驱动

```
sudo apt install nvidia-driver-520-open
```

安装完成后重启

```
reboot
```

2.3.1.8. 安装完成后查看驱动信息

查看驱动信息的方式有2种，一个是用`nvidia-smi`命令行查看：

```
nvidia-smi  # 查看NVIDIA驱动信息
```

如下所示，即为安装成功。

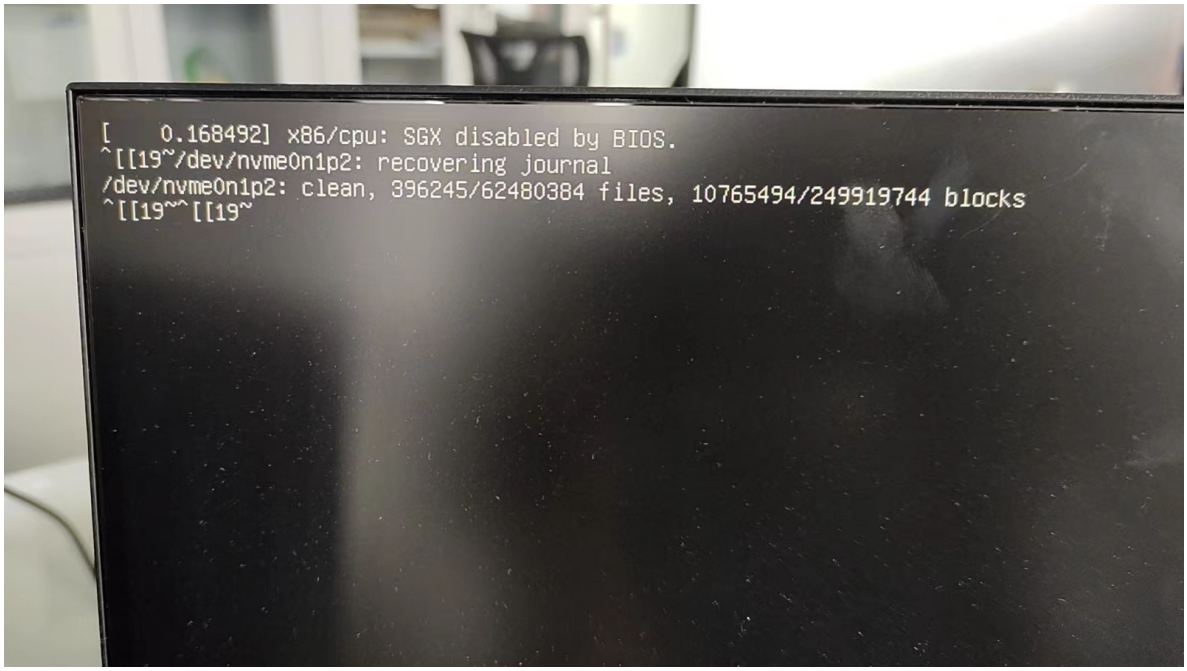
```
Mon Nov 14 20:24:32 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 520.61.05    Driver Version: 520.61.05    CUDA Version: 11.8     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:01:00.0  On |                  N/A |
| 44%   65C    P2   113W / 170W |   6787MiB / 12288MiB |     46%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1493      G   /usr/lib/xorg/Xorg                 35MiB |
|    0   N/A  N/A      1963      G   /usr/lib/xorg/Xorg                400MiB |
|    0   N/A  N/A      2093      G   /usr/bin/gnome-shell               44MiB |
|    0   N/A  N/A      2634      G   ...RendererForSitePerProcess       11MiB |
|    0   N/A  N/A      3039      G   ...-community/jre64/bin/java        2MiB |
|    0   N/A  N/A     13118      G   ...nlogin/bin/sunloginclient        6MiB |
|    0   N/A  N/A     18273      G   /usr/lib/firefox/firefox          163MiB |
|    0   N/A  N/A     22868      C   ...da3/envs/KK1-2/bin/python     6106MiB |
```

还有一种是从设置（settings）->关于（About）中查看

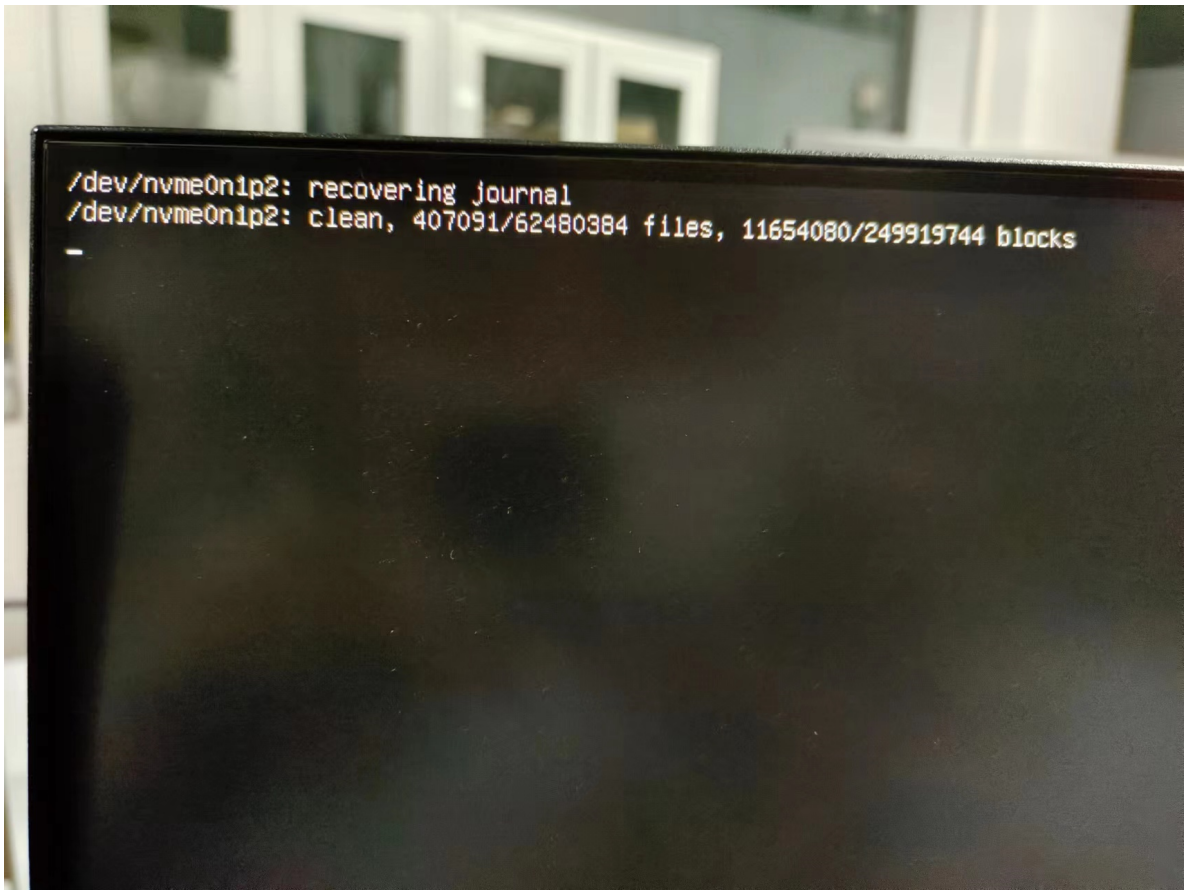
说明：显卡的位置显示的是`NVIDIA GeForce 3060`即为驱动安装成功了！

2.3.1.9. 错误解决方案

> 错误信息



> 错误2：



> 解决方案

注意： 如果出现错误请检查驱动是否兼容，并更换驱动

```
进入BIOS 中 选择高级 -> 选择revovery mode -> 选择  root 进入命令控制窗口
更新库：sudo apt update
更新内核：sudo apt upgrade
检查更新：sudo apt install -f
删除旧内核：sudo apt autoremove
重启：sudo reboot
```

如果需要安装驱动则执行命令：

```
sudo ubuntu-drivers autoinstall
```

2.4. Linux安装向日葵

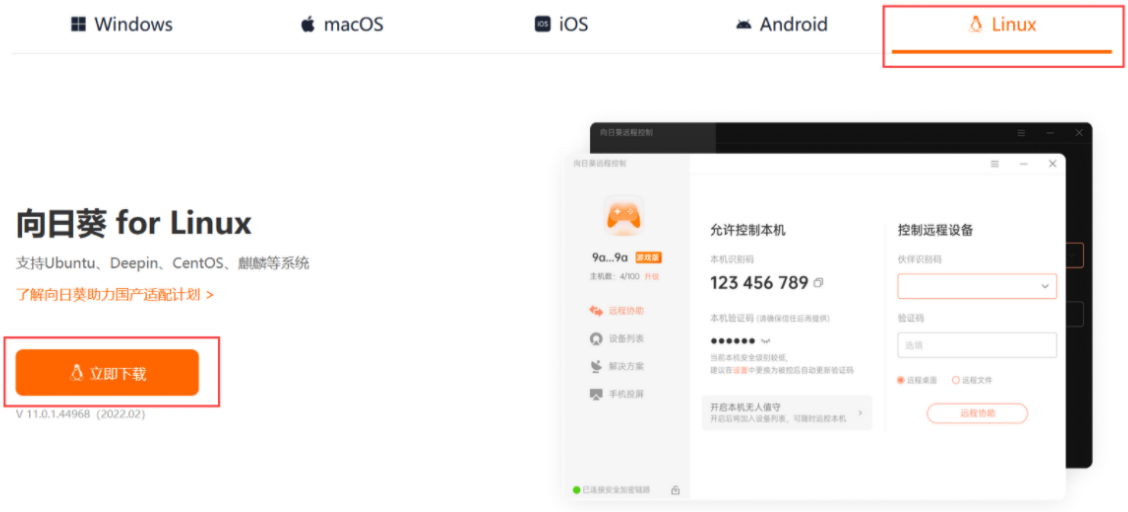
> 安装向日葵并设置自启动（Ubuntu22.04）

2.4.1. 软件下载

首先到向日葵官网下载程序，具体如下：

网址：向日葵远程控制app官方下载 - 贝锐向日葵官网

选择linux后点击 立即下载



注意：选择图形版本进行下载，命令行版本可能会出现莫名其妙的错误

2.4.2. 软件安装

> 首先切换到下载文件所在目录空白处右键 点击 在终端打开执行如下命令：

```
命令： sudo dpkg -i 文件名.deb
```

2.4.3. 软件启动

> 执行启动命令：

```
命令： /usr/local/sunlogin/bin/sunloginclient
```

2.4.4. 设置自启动

2.4.4.1. 查找安装位置

```
命令： dpkg -L sunloginclient
```

```
/usr/local/sunlogin/scripts/uninstall.sh
/usr/local/sunlogin/scripts/start.sh
/usr/local/sunlogin/scripts/init_runsunloginclient
/usr/local/sunlogin/res
/usr/local/sunlogin/res/font
/usr/local/sunlogin/res/font/wqy-zenhei.ttc
/usr/local/sunlogin/res/icon
/usr/local/sunlogin/res/icon/online.ico
/usr/local/sunlogin/res/icon/offline.ico
/usr/local/sunlogin/res/icon/sunlogin_client.png
/usr/local/sunlogin/res/icon/online_lock.ico
/usr/local/sunlogin/res/icon/offline_lock.ico
/usr/local/sunlogin/res/icon/online_ctrl.ico
/usr/local/sunlogin/res/skin
/usr/local/sunlogin/res/skin/remotecmd.skin
/usr/local/sunlogin/res/skin/remotecamera.skin
/usr/local/sunlogin/res/skin/skin.skin
/usr/local/sunlogin/res/skin/remotefile.skin
/usr/local/sunlogin/res/skin/desktopcontrol.skin
/usr/local/sunlogin/bin
/usr/local/sunlogin/bin/sunloginclient
/usr/local/sunlogin/bin/oray_rundaemon
/usr/local/sunlogin/bin/sunloginclient_desktop
```

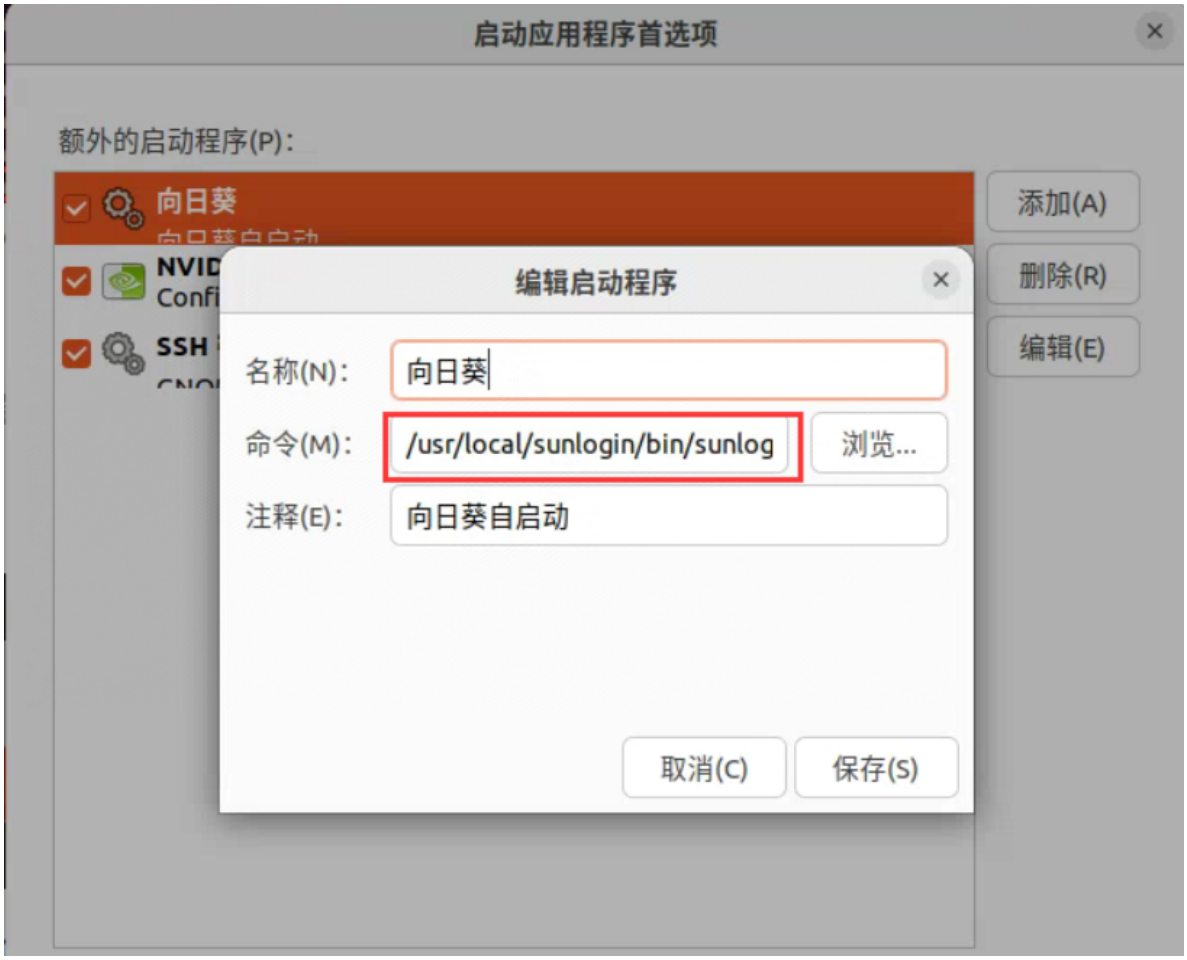
2.4.4.2. 使用命令或者界面打开 启动程序

```
命令： gnome-session-properties
```

然后 点击 添加

2.4.4.3. 添加启动命令路径

将第一步获取到的路径添加到如下位置:



2.4.4.4. 解决正在进入桌面 一直无法进入

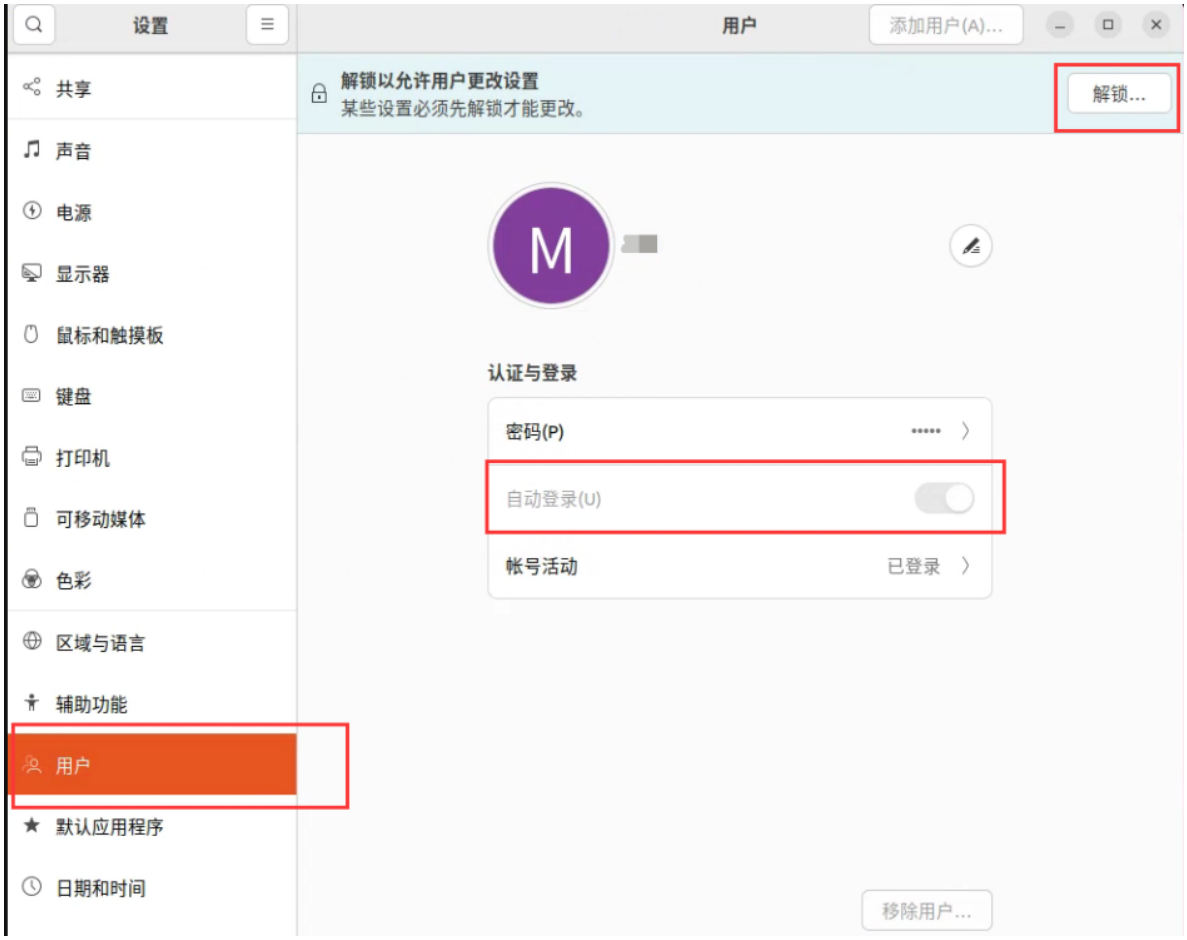
```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install lightdm

最后一个执行过程中选择lightdm
```

注意：执行完毕后需要重启机器方可生效

2.4.5. 设置无密码登录

> 选择设置

> 左侧菜单栏选择用户 右上角解锁 后打开自动登录 实现向日葵自启动



2.5. Linux部署安装Nginx

2.5.1. 安装

```
sudo apt-get install nginx
```

2.5.2. 启动

```
sudo /etc/init.d/nginx start #通过init.d下的启动文件启动。
sudo service nginx start #通过ubuntu的服务管理器启动
```

2.5.3. Nginx 配置验证

> 检测配置文件脚本

```
!# /bin/bash
nginx -t -c /etc/nginx/nginx.conf
```

> 验证配置文件是否正确

```
nginx -t -c /etc/nginx/nginx.conf
或
nginx -t nginx.conf
```

2.5.4. Nginx 异常解决方案

2.5.4.1. 无法启动

> 错误信息：

```
11月 29 13:46:05 ubuntu-ThinkStation-K systemd[1]: Starting A high performance web server and a reverse proxy server...
11月 29 13:46:05 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] bind() to 0.0.0.0:17001 failed (98: Address already in use)
11月 29 13:46:06 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] bind() to 0.0.0.0:17001 failed (98: Address already in use)
11月 29 13:46:06 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] bind() to 0.0.0.0:17001 failed (98: Address already in use)
11月 29 13:46:07 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] bind() to 0.0.0.0:17001 failed (98: Address already in use)
11月 29 13:46:07 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] bind() to 0.0.0.0:17001 failed (98: Address already in use)
11月 29 13:46:08 ubuntu-ThinkStation-K nginx[27123]: nginx: [emerg] still could not bind()
11月 29 13:46:08 ubuntu-ThinkStation-K systemd[1]: nginx.service: Control process exited, code=exited, status=1/FAILURE
11月 29 13:46:08 ubuntu-ThinkStation-K systemd[1]: nginx.service: Failed with result 'exit-code'.
11月 29 13:46:08 ubuntu-ThinkStation-K systemd[1]: Failed to start A high performance web server and a reverse proxy server.
```

> 解决方案

原因：之前卸载nginx时没卸载干净，导致此错误，执行以下指令清除干净后安装即可

```
sudo apt-get remove nginx nginx-common
sudo apt-get purge nginx nginx-common
sudo apt-get autoremove
sudo apt-get remove nginx-full nginx-common
```

2.5.5. Nginx代理转发配置

2.5.5.1. Mysql 代理配置

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}

stream {
```

```
    log_format proxy '$remote_addr [$time_local] '
                 '$protocol $status $bytes_sent $bytes_received '
                 '$session_time "$upstream_addr" '
                 '"$upstream_bytes_sent" "$upstream_bytes_received"
"$upstream_connect_time"';

    access_log  /var/log/nginx/access.log  proxy;

    upstream mysql_server {
        server 127.0.0.1:3306;
    }

    server {
        listen 63306;
        proxy_pass mysql_server;
    }
}
```

2.5.5.2. 应用服务代理配置

```
map $http_upgrade $connection_upgrade {
          default upgrade;
          '' close;
}


#负载均衡策略配置
upstream coframeServver {
    #ip_hash;
    server 192.168.110.101:17010 weight=1;
    server 192.168.110.102:17010 weight=2;
}



#文件服务负载均衡策略配置
upstream pythonFileServer {
    #ip_hash;
    server 123.57.89.158:13001 weight=1;
}



#智能客服websocket 负载均衡配置
upstream znkfWebSocketbackend {
        server 192.168.110.101:17011 weight=1;
    server 192.168.110.102:17011 weight=1;
    keepalive 1000;
}




#服务配置
server {
    #监听端口
    listen       17001;
```

```
    server_name  192.168.110.102;

    # 在线Api文档静态资源
    location /doc {
       proxy_pass http://coframeServver;
    }

    #在线Api 文档 后端接口
    location /webjars {
       proxy_pass http://coframeServver;
    }

    #在线Api文档资源配置
    location /swagger-resources {
       proxy_pass http://coframeServver;
    }

    #在线Api文档资源配置
    location /v2 {
       proxy_pass http://coframeServver;
    }

    #后端应用服务
    location /greenCoframeApi {
        proxy_pass http://coframeServver;
        rewrite ^/greenCoframeApi/(.*) /$1 break;
    }



    # python 公共文件操作服务
    location /grCommonFileApi {
        proxy_pass http://pythonFileServer;
        rewrite ^/grCommonFileApi/(.*) /$1 break;
    }

    #后端服务web入口配置
    location / {
        root   /soft/zzbserver/html-web/zzbpc;
        try_files $uri $uri/ /index.html;
        index  index.html index.htm;
    }
}



#智能客服websocket 服务配置
server {
       listen 17016;
       location /znkfws {
            proxy_http_version 1.1;
            proxy_pass http://znkfWebSocketbackend;
            proxy_redirect off;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_read_timeout 3600s;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;
        }
}
```

2.5.6. Nginx高并发及吞吐配置

> 单机器如何实现Nginx百万并发连接

单台服务器，想突破单个ip最多只能开 65535个不同端口的限制，达到100W，于是便在 eth0网卡上创建多个子网卡，将请求分配到不同的 ip上来实现

> 命令行创建子网卡

```
ifconfig eth0:0 192.168.1.11/24 up
ifconfig eth0:1 192.168.1.12/24 up
ifconfig eth0:2 192.168.1.13/24 up
ifconfig eth0:3 192.168.1.14/24 up
ifconfig eth0:4 192.168.1.15/24 up
ifconfig eth0:5 192.168.1.16/24 up
ifconfig eth0:6 192.168.1.17/24 up
ifconfig eth0:7 192.168.1.18/24 up
ifconfig eth0:8 192.168.1.19/24 up
ifconfig eth0:9 192.168.1.10/24 up
ifconfig eth0:10 192.168.1.20/24 up
ifconfig eth0:11 192.168.1.21/24 up
ifconfig eth0:12 192.168.1.22/24 up
ifconfig eth0:13 192.168.1.23/24 up
ifconfig eth0:14 192.168.1.24/24 up
```

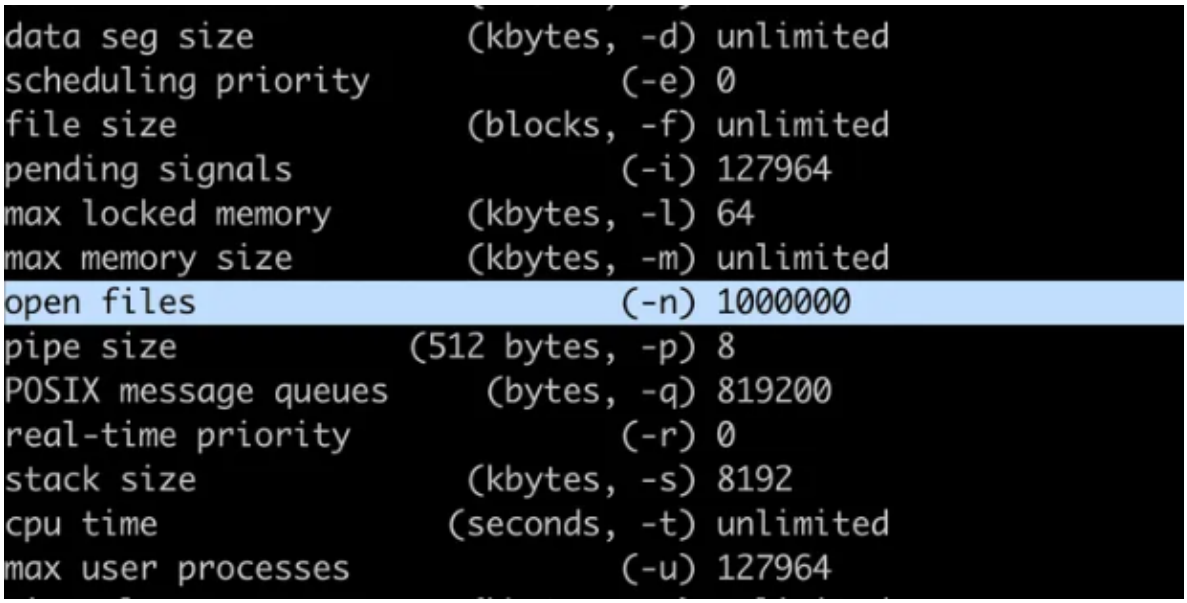
```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.72.248  netmask 255.255.240.0  broadcast 172.19.79.255
        ether 00:16:3e:18:7a:19  txqueuelen 1000  (Ethernet)
        RX packets 518041045939  bytes 42074636589452 (38.2 TiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 274284848119  bytes 29120077572138 (26.4 TiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0:0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.11  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:16:3e:18:7a:19  txqueuelen 1000  (Ethernet)

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.12  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:16:3e:18:7a:19  txqueuelen 1000  (Ethernet)

eth0:2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.13  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:16:3e:18:7a:19  txqueuelen 1000  (Ethernet)

eth0:3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.14  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:16:3e:18:7a:19  txqueuelen 1000  (Ethernet)

eth0:4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

> 配置 Linux 最大打开文件数

```
#文件地址 /etc/security/limits.conf 新增下面内容

* soft nofile 1000000
* hard nofile 1000000

root soft nofile 1000000
root hard nofile 1000000
```

> 登录查看最大打开文件数已生效

```
ulimit -a
```



> 修改/etc/security/limits.conf文件，重启后不生效

问题现象
修改/etc/security/limits.conf文件，重启后配置项不生效。

> 解决方案

修改/etc/security/limits.d/目录中配置项或修改/etc/security/limits.conf文件。

2.5.6.1. 调整 nginx配置

> 修改Nginx最大打开文件数为 100w

```
worker_rlimit_nofile 1000000;
```

> 修改**Nginx事件处理模型**

```
events {
    use epoll;
    worker_connections 1000000;
}
```

> 配置应用请求

- 后端 go应用程序监听在 9505 端口，nginx中配置 upstream

```
upstream go-ws {
    server 127.0.0.1:9505;
    keepalive 128;
}

location /ws {
        proxy_redirect off;
        proxy_bind $split_ip;
        proxy_pass http://go-ws;
        proxy_bind $spl_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # WebSocket 支持的核心配置
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $http_connection;

        proxy_intercept_errors on;
        client_max_body_size 20m;
    }
#将请求分配到不同ip上
split_clients "$remote_addr$remote_port" $split_ip {
        10%  192.168.1.11;
        10%  192.168.1.12;
        10%  192.168.1.13;
        10%  192.168.1.14;
        10%  192.168.1.15;
        10%  192.168.1.16;
        10%  192.168.1.17;
        10%  192.168.1.18;
        10%  192.168.1.19;
        10%  192.168.1.20;
        10%  192.168.1.21;
        10%  192.168.1.22;
        10%  192.168.1.23;
        10%  192.168.1.24;
        *    192.168.1.10;
}
```

压测发现，句柄数并未达到100万就报http: Accept error: accept tcp [::]:9505: accept4: too many open files; retrying in 40ms

最后发现是supervisorctl句柄数据没有调整

2.5.6.2. 优化supervisor配置

```
[supervisord]
logfile=/var/log/supervisor/supervisord.log  ; (main log file;default $CWD/supervisord.log)
logfile_maxbytes=50MB        ; (max main logfile bytes b4 rotation;default 50MB)
logfile_backups=10           ; (num of main logfile rotation backups;default 10)
loglevel=info                ; (log level;default info; others: debug,warn,trace)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
nodaemon=false               ; (start in foreground if true;default false)
minfds=1000000                ; (min. avail startup file descriptors;default 1024)
minprocs=1000000               ; (min. avail process descriptors;default 200)
;umask=022                   ; (process file creation umask;default 022)
;user=chrism                 ; (default is current user, required if root)
;identifier=supervisor       ; (supervisord identifier, default is 'supervisor')
;directory=/tmp              ; (default is not to cd during start)
;nocleanup=true              ; (don't clean up tempfiles at start;default false)
;childlogdir=/tmp            ; ('AUTO' child log dir, default $TEMP)
;environment=KEY=value       ; (key value pairs to add to environment)
```

```
cat /proc/sys/fs/file-nr
527040  0    1000000 #值的的解释：当前已经分配的文件句柄数；闲置的文件句柄数；最大文件句柄数
```

调整之后正常了，nginx日志也无报错了
查看系统 tcp连接
当前业务 tcp连接在 50w左右徘徊(基本上都是长连接)

查看连接数量

```
ss -s
```

```
Total: 525448 (kernel 525506)
TCP:   530920 (estab 525287, closed 2397, orphaned 3228, synrecv 0, timewait 2396/0), ports 0

Transport Total     IP        IPv6
*         525506    -         -
RAW       0         0         0
UDP       3         2         1
TCP       528523    354664    173859
```

> 高峰期实际会达到 70w,远没有达到 100W，