Linux 笔记

adamanteye

1. 发行版选择

以下评述基于我或朋友们的看法,其中我只尝试过 `Arch`, `Gentoo`, `Debian` 三种发行版.

1.1. Arch

兜兜转转最喜欢的发行版,因为 `pacman` 好用,软件包丰富,文档最完善,脚本 `PKGBUILD` 很好写, AUR 对用户很友好.

缺点是打包比较粗,这方面 Debian, OpenSUSE 或许算是打包细致的范例.

1.2. Alpine

目前只在 `docker` 中用过,听说 Alpine Edge 也可以被拿来当桌面发行版.

1.3. eweOS

跟 Arch 的哲学很像,但是 `musl`,现在还处于开发中.

1.4. Debian

自己的服务器上都在跑 Debian,因为它自由稳定,不喜欢 Ubuntu 正因为其是大公司的推广.

1.5. Gentoo

拿来在服务器上跑也会比较合适,不过源码分发比较折磨人.

`emerge` 以及一众工具好细碎,另外好慢.

1.6. Fedora

软件包更新甚至比 Gentoo 激进.

1.7. OpenSUSE

被称为最适合 `KDE` 的发行版,大概都是德国人开发的吧.

2. 多用户管理

2.1. SSH

pam_ssh_agent_auth 允许登入的用户通过 ssh-agent 获得权限,在服务器上的配置参考 `pam_ssh_agent_auth(8)`. 例如:

```
# /etc/pam.d/sudo
auth sufficient pam_ssh_agent_auth.so file=~/.ssh/authorized_keys
# /etc/sudoers:
Defaults:%sudo env_keep += "SSH_AUTH_SOCK"
```

从而不再需要为服务器 sudo 用户组的用户设置密码,由于 `SSH_AUTH_SOCK` 被保留,鉴权会通过 `pam-ssh-agent-auth` 进行.

2.2. sudo

用 `env_keep` 可以保留一些有用的环境变量:

```
# /etc/sudoers
```

```
# 保留编辑器设置
Defaults:%wheel env_keep += "SUDO_EDITOR EDITOR VISUAL"
# 可以用 sudo 执行 rsync, scp
Defaults:%wheel env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"
# 保留 HTTPS 与 HTTP 代理
Defaults:%wheel env_keep += "all_proxy"
```

2.3. 登入用户信息

who 与 w 命令可以查询当前登入的用户,不过在 gentoo prefix 下运行不会查到任何用户:

```w
w
```
Shell

```
23:18:07 up  2:43,  1 user,  load average: 2.12, 3.14, 3.20
USER     TTY       LOGIN@   IDLE   JCPU   PCPU  WHAT
adamante tty1      20:34    2:43m  6:51m   ?    footclient
```

```who
who
```
Shell

```
adamanteye tty1         2025-01-25 20:34
```

3. 网络管理

3.1. 地址与路由

访问 4.ipw.cn 与 6.ipw.cn 可以看到出口 IP 地址.可以利用它配合 ddclient 更新 DNS 记录,例如配置/etc/ddclient.conf 如下:

```
# /etc/ddclient.conf

daemon=360
protocol=cloudflare
usev4=cmdv4, cmdv4=get-ipv4
usev6=cmdv6, cmdv6=get-ipv6
login=token
password='your-api-token'
zone=adamanteye.cc
heloise.adamanteye.cc
```

其中的/usr/local/bin/get-ipv4 命令为:

```
#!/bin/bash
curl 4.ipw.cn
```

快速 ping 所有主机:

```nmap
nmap -sn 192.168.1.0/24
```
Shell

或者

```shell
for i in {1..254}; do sudo ping -c 1 -W 1 192.168.1.$i | grep "bytes from" && echo "192.168.1.$i is alive"; done
```

3.2. 端口

查看当前监听的端口以及进程:

```shell
sudo netstat -tunlp
```

`ss` 命令由 `iproute2` 提供,功能与 `netstat` 类似,但信息更全.

3.3. ifupdown

这是 Debian 上的传统方法.例如启用所有以 `auto` 定义的接口:

```shell
sudo ifup -a
```

配置 DHCP:

```
# /etc/network/interfaces
auto eno1
iface eno1 inet dhcp
iface eno1 inet6 dhcp
```

注意变更配置前,首先应当 `ifdown`,因为 `ifup` 与 `ifdown` 始终根据当前的 `interfaces` 文件使用 `ip` 命令配置网络.

3.4. NetworkManager

编写 dispatcher 可以实现自动切换有线,无线连接,详见 NetworkManager: automatically switch between Ethernet and Wi-Fi.

`dnsmasq` 可以作为 `NetworkManager` 的本地 DNS 缓存服务器.且可以为不同的域名选择不同的 DNS 服务器:

```
# /etc/NetworkManager/dnsmasq.d/server.conf
server=/tsinghua.edu.cn/166.111.8.28
server=/tsinghua.edu.cn/2402:f000:1:801::8:28
server=/cn/101.6.6.6
server=101.101.101.101
server=2001:de4::101
```

```
# /etc/NetworkManager/NetworkManager.conf
[main]
dns=dnsmasq
```

3.5. 设备管理

`rfkill` 可以启用/禁用 WIFI,蓝牙在内的无线设备.例如:

```shell
rfkill # list wireless devices
rfkill unblock bluetooth
```

3.6. 网络攻击

3.6.1. namp

```shell
nmap -T4 -A -v -Pn 192.168.0.1
#最常用的一种扫描
```

- `-T4`: 设置时序,越高扫描越快
- `-A`: 启用操作系统检测,版本检测,脚本扫描和跟踪路由
- `-v`: 增加详细级别(使用`-vv` 或更高级别以获得更好的效果)
- `-Pn`: 无 ping 扫描

3.6.2. 密码爆破

Kail 提供了 `rockyou.txt`,是常见密码的集合.

针对开启 ssh 密码登陆的主机,可以用 `hydra`:

```shell
hydra -l root -s 22 -P passwords.txt 154.12.60.17 ssh
```

3.6.3. DDoS

3.6.3.1. 传输层

耗尽目标服务器的网络带宽或连接资源.

常见形式

- SYN Flood
- UDP Flood
- ICMP Flood

3.6.3.2. 应用层

模拟合法用户请求,耗尽服务器的 CPU,内存等资源.

常见形式

- HTTP FLood
- Slowloris

防护需要 WAF.

3.7. 网络测绘

3.7.1. 代理

代理使用者会有相当多可供探测的特征[1].

3.7.2. 参考

- 进行网络测绘的方法与挑战 | Ajax's Blog.

3.8. 证书

acmesh-official/acme.sh 可以自动签发与更新证书.

将证书安装到指定位置:

[1]参见 Avoiding Live VPN/Proxy Detection · Issue 445 · net4people/bbs

```shell
./acme.sh --install-cert -d 'adamanteye.cc' \
  --fullchain-file /srv/cert/all.adamanteye.cc.fullchain \
  --key-file /srv/cert/all.adamanteye.cc.key
```

4. 启动引导

4.1. GRUB

从 grub 命令行中引导系统.

/分区为 btrfs 的例子:

```shell
grub> ls
grub> set root=(hd0,gpt2)
grub> linux /@rootfs/boot/vmlinuz-6.1.0-30-amd64 root=/dev/sda2 rw rootflags=subvol=@rootfs
grub> initrd /@rootfs/boot/initrd.img-6.1.0-30-amd64
grub> boot
```

/分区为 ext4 的例子:

```shell
grub> ls
grub> set root=(hd0,gpt2)
grub> linux /boot/vmlinuz-6.1.0-30-amd64 root=/dev/sda2
grub> initrd /boot/initrd.img-6.1.0-30-amd64
grub> boot
```

4.2. systemd-boot

```
# /boot/loader/loader.conf

default arch.conf
timeout 4
console-mode max
editor no
```

```
# /boot/loader/entries/arch.conf

title Arch
linux /vmlinuz-linux
initrd /intel-ucode.img
initrd /initramfs-linux.img
options root=UUID=1a514b60-fdd7-4b19-a53b-3ce10b157faa rw
```

5. 软件分发

5.1. 手册页

scdoc 自定义了与 Markdown 相近的语法,可以用来生成 man 手册页:

```shell
scdoc < foot.1.scd > foot.1
gzip foot.1
man ./foot.1.gz
```

5.2. Shell 补全

5.2.1. fish

`fish_update_completions` 可以从系统 man pages 生成补全.

5.3. Arch Linux 打包

`namcap` 可以方便地检查 `PKGBUILD` 和打好的包当中出现的错误,以及所依赖的动态库.

5.4. Debian 打包

参考:

- Debian Policy Manual
- Debian 维护者指南
- Debian 打包教程

Debian 打包体系随时间演化,在 Debian Trends 可以看到不同源码格式,打包工具的使用比例.

我构建了一个 debian 的 docker 环境,adamanteye/images/debian-builder,用来完成在干净系统下的打包.

一般的打包流程:

```shell
tar xaf example-0.1.0.tar.gz
cd example-0.1.0
debmake -b':sh' -x1   # 选一个模板
vim debian/control    # 以及其他文件
rm -rf debian/patches # 以及其他用不到的文件
debuild
```

5.5. 证书

在 Arch 的打包中,需要为 `license` 变量指定对应的 SPDX 标识符[2],常见许可证的文本存储于`/usr/share/licenses/spdx/`.

6. 规范

6.1. 时间日期

查看 `strftime(3)` 了解可用的格式化选项,例如 `2025 03 05` 对应 `%Y %m %d`.

6.2. 文件编码

在 Windows 上保存的文件在 Linux 中打开以及用版本管理系统进行管理时,存在兼容性问题.

6.2.1. UTF-8-BOM

BOM(byte-order mark)是用来指示编码方案以及端序的,常见的是 `EF BB BF`,表示接下来的内容以 UTF-8 编码.

然而例如在 PHP 中,由于`<?php`(这是文件开头)之前不得有其他字符,如果有 BOM,可能会造成 PHP 在设置 headers 之前就返回内容.

Windows 下保存的 `h5` 文件在 MacOS 或 Linux 下打开也有可能出现列名包含非可显示字符的错误,以至于明明索引了正确的列名但无法取到相应的数组,这也是因为 BOM 引起的.

[2] SPDX License List | Software Package Data Exchange (SPDX)

6.2.2. CRLF

打字机时代,换行包含两个动作,移到第一列,移到下一行, Windows 遵循了这种惯例.

在 Unix 中换行只是`\n`,因此在版本控制系统中需要注意两者的转换,可以在 Git 中设置遵循其中一种格式:

```
# .gitconfig
[core]
  eol = lf
```

在 `git checkout` 等场景下,文本文件的换行符会被设置为`\n`.

6.3. readline 键位

- `Ctrl A` 跳到行首
- `Ctrl E` 跳到行尾
- `Ctrl U` 删除光标位置到行首的所有内容
- `Ctrl K` 删除光标位置到行尾的所有内容
- `Ctrl W` 删除光标位置前一个单词
- `Ctrl Y` 粘贴最后一次删除的内容
- `Ctrl L` 清屏
- `Ctrl F` 向前移动一个字符
- `Ctrl B` 向后移动一个字符
- `Alt F` 向前移动一个单词
- `Alt B` 向后移动一个单词
- `Ctrl R` 启动反向搜索历史命令
- `Ctrl S` 启动正向搜索历史命令
- `Ctrl G` 取消搜索

常见 shell 和 emacs 都支持这些键位.

7. 系统技巧

7.1. 彩色输出

以下是一些可启用彩色输出的命令:

```shell
alias ls='ls --color=auto'
alias ip='ip --color=auto'
alias grep='grep --color=auto'
```

使用 lsd 代替 `ls`,可以提供彩色输出.

此外 `lsd` 可以代替 `tree`.

```shell
lsd --tree
```

使用 `bat` 来代替 `cat` 和 `less`,可以无感知实现大多数语言的语法高亮:

```shell
alias cat='bat --style=plain --paging=never'
```

```
alias less='bat  --style=plain'
```

不过, bat 作为 MANPAGER 后会丢失下划线格式:

```
export MANPAGER="sh -c 'col -bx | bat -l man -p'"
```
Shell

作为对比,查看依靠 less 的彩色化输出:

```
export MANPAGER="less -M -R -i --use-color -Dd+R -Du+B -DHkC -j5"
```
Shell

7.2. 实时输出

许多命令提供-f/--follow 选项,例如:

```
sudo tail -f /var/log/xray/access.log
```
Shell

```
docker compose logs -f
```
Shell

```
dmesg -w
```
Shell

此外,可以利用 watch 以固定间隔更新输出:

```
watch -n 3 "ps aux | grep node"
```
Shell

7.3. 别名

用 alias 为命令创建别名:

```
alias hx=helix
```
Shell

对于 git 子命令,可以设置 git 别名:

```
# ~/.gitconfig
[alias]
    ss = status
    kmt = commit
    chk = checkout
    br = branch
    ps = push
```

随后可以用 git kmt 代替 git commit.

7.4. 定时任务

crontab guru 可以在线编辑验证 crontab 语法.

7.5. SSH

配置 SSH 转发,可以在服务器上使用本地的私钥:

```
Host foo
  ForwardAgent yes
```

如果端口被阻断,需要配置网络代理以通过代理服务器访问 SSH 服务器:

```
Host github.com
  Hostname ssh.github.com
```

```
  Port 443
  ProxyCommand nc -X connect -x [::1]:10801 %h %p
```

Arch Linux 的 openbsd-netcat 提供了 nc 命令.

多跳连接:

```shell
ssh -J kaiser elisabeth -D 1080 -N
```

首先跳到 kaiser,其次到 elisabeth.

7.6. 临时文件

创建临时文件或临时文件夹:

```shell
mktemp example.XXXXXXXX
```

7.7. 文本处理

grep, sed, awk 是最常使用的文本处理程序.

```shell
cat /usr/share/fortune/chinese | sed 's/\x1B\[[0-9:;<=>?]*[!-/\x20]*[@-~]//g' > chinese-without-color
# 删去所有 ANSI CSI
```

8. 实用程序

8.1. Git

8.1.1. 通过邮件提交补丁

参考 How to submit a patch by email | Peter Eisentraut,首先撰写 commit.之后使用

```shell
git format-path [ <since> | <revision-range> ]
```

生成补丁.

8.1.2. Hooks

commit-msg 在提交信息编辑完成后,最终提交前执行.可以验证或修改最终的提交信息.

prepare-commit-msg 在生成提交信息后,打开编辑器前执行.在提交时增加额外信息.

8.2. 桌面环境

之前用 hyprland,发现依赖太重,于是切换到 niri.此外 niri 的标签页交互很舒适.

niri 没有内置的 Xwayland,文档推荐使用 xwayland-satellite.

此外之前也用过很久的 KDE,同样因为太笨重而切换了平铺式桌面管理器.

8.3. 编辑器

helix-editor/helix 在绝大多数发行版都已经得到了支持,但是 debian 尚且没有打包.

8.4. 终端与 Shell

foot 是轻量的 Wayland 终端,支持 img2sixel.

fish 相比 zsh 速度更快,且 4.0 版本已经成功用 Rust 重写.

8.5. 浏览器

firefox 与 zotero 都可以配置多个 profile,从而实现插件,设置,书签等的隔离:

```shell
firefox --profile "$HOME.mozilla/firefox/crawler/"
```

通过改变浏览器 UA,可以绕开相当多的付费墙. 例如 firefox 在 about:config 选项卡中,设置 general.useragent.override 为:

```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

经过测试,这对端传媒有效.可以为 google bot UA[3]配置与日常使用隔离的 profile.

查看当前的浏览器指纹: What browser?

8.6. 文件管理器

sxyazi/yazi 是使用 Rust 编写的命令行文件管理器.

8.7. 文档手册

tealdeer 提供了相当好的 tldr 体验,可以查看大部分命令的简短文档.

8.8. 归档与压缩

8.8.1. tar

```shell
tar cvzf archive.tar.gz <sources>
tar xvf archive.tar.gz
```

8.9. 输入法

aur/fcitx5-pinyin-sougou-dict-git 提供了搜狗词库.

8.10. 邮件客户端

thunderbird 几乎开箱即用.不过一些高级用户会选择 neomutt/mutt,两者配置基本兼容.

对于 mutt,既可以用自带的 IMAP 协议,也可以使用外部收取程序,如 offlineimap 以及 msmtp.

参考配置教程:

- Mutt: 阅读邮件列表 | FancySeeker
- Mutt 配置: 一种实践的部署方式
- OfflineIMAP examples

8.11. tmux

- 左右布局: Ctrl B, %
- 向左切换布局: Ctrl B, 方向左

8.12. 桌面通知

libnotify 提供了以下命令:

```shell
notify-send -t 5000 "Your Title or App Name" "a message that will be displayed for 5000ms"
```

如果还想播放声音,可以使用 libpulse 提供的 paplay:

[3]查看其他浏览器 UA: List of User Agents

```shell
paplay /usr/share/sounds/freedesktop/stereo/message.oga
```

注意音频由 `sound-theme-freedesktop` 提供.

8.13. RSS 订阅

如果熟悉 `mutt`,对 newsboat 应当同样很有好感.

8.14. 排版软件

typst 可以替代 LaTeX.

8.14.1. 字体

OpenType 字体有一系列 feature 可以启用,参考 Text Function - Typst Documentation 以及 Registered features (OpenType 1.9.1) - Typography | Microsoft Learn. 例如:

```typst
#let smcp(it) = {
  set text(features: ("smcp",))
  it
}
```

8.14.2. 配色

- 猫布奇诺调色盘
- OKLCH Color Picker & Converter

8.14.3. 排版原则

- Butterick's Practical Typography

8.15. 配置管理

GNU `stow` 利用软链接集中地管理配置文件,可以配合 `git` 进行版本控制和备份.

我自己的配置文件管理在 adamanteye/dotfiles.

8.16. 待办管理

Taskwarrior 功能丰富,更新到 3.0 版本后改变了远程同步的方式,可以自己托管远程同步服务.

Taskwarrior 创建循环任务:

```shell
task recur:2d due:eod add 吃山楂片
```

8.17. 密码管理

pass 基于 `gpg`,在本地管理密码,并且支持用 `git` 进行版本控制,可以一同配置 GitHub 远程仓库来备份.

8.18. 音视频处理

使用 `ffmpeg` 连接视频:

```shell
ffmpeg -f concat -i filelist.txt -c copy out.mp4
```

其中 `filelist.txt` 的内容为:

```
file '1.mp4'
file '2.mp4'
```

图形化剪辑软件 shotcut 使用 `qt6`,界面现代.

8.19. PDF

poppler 包提供了 pdfseparate 以及 pdfunite 两个包,可拆分合并指定页范围的 PDF 文件.

8.20. 远程文件系统

sshfs 可以通过 ssh 连接挂载远程文件系统.

8.21. 逆向

Hash 推荐我用 binaryninja-free.

9. 资源监控

9.1. 文件系统

ncdu 是采用 ncurses 界面的磁盘占用统计工具,比 du 命令更好用,带有彩色模式:

```
# /etc/ncdu.conf
--color dark-bg
```

使用 fio 测试顺序读写,随机读写等情景下的性能:

```
fio -filename=/home/adamanteye/test -direct=1 -iodepth 1 -thread -rw=randrw -bs=4k -size=2G -
numjobs=5 -runtime=10 -group_reporting -name=mytest | tee randrw.log
```
Shell

9.2. CPU 信息

```
cat /proc/cpuinfo # 或者
lscpu
```
Shell

9.3. 温度监控

读取硬盘温度,使用 smartctl:

```
sudo smartctl --all /dev/sda
```
Shell

读取 CPU 温度,使用 lm-sensors:

```
sudo sensors-detect  # 初次配置 lm-sensors
sensors              # 按照配置读取温度
```
Shell

9.4. 风扇控制

参考 jhatler/ipmi-fanctrl-dell_r630.sh:

```
sudo ipmitool raw 0x30 0x30 0x01 0x00       # enable manual fan control
sudo ipmitool raw 0x30 0x30 0x02 0xff 0x14  # set fan speed to 20%
```
Shell

9.5. 功率监控

```
ipmitool dcmi power reading
```
Shell

10. 包管理

体验过 pacman, apt, emerge,其中还是 pacman 的体验最好(毕竟是功能最简陋的).

Repology 列出了常见发行版上的打包情况,不过没有 gentoo 的.

10.1. pacman

列出所有显式安装的包:

```shell
pacman -Qe
```

列出所有悬垂包:

```shell
pacman -Qtdq
```

更新文件数据库:

```shell
pacman -Fy
```

寻找包含指定文件名的包:

```shell
pacman -F tldr
pacman -F /usr/bin/tldr
```

10.2. AUR

之前使用 Juger/yay,现在我迁移到了 Morganamilo/paru.

10.3. dpkg & apt

解压 `deb` 包:

```shell
ar x example_0.1.0-1_all.deb
```

列出文件:

```shell
dpkg-deb -c example_0.1.0-1_all.deb
```

显示元信息:

```shell
dpkg-deb -I example_0.1.0-1_all.deb
```

11. init 程序

`init` 程序是系统启动的第一个程序(`pid` 为 1),它完成主引导流程.

Debian 和 Arch 系统的`/usr/sbin/init` 是指向`../lib/systemd/systemd` 的符号链接.

11.1. systemd

列出所有 unit 的初始化时间:

```shell
systemd-analyze blame
```

11.2. macOS

macOS 所使用的守护进程管理是 `launchd`,管理系统级或用户级的守护程序.

要编写`<LABEL>.plist` 文件,参考 macOS daemons and agents 以及 Creating Launch Daemons and Agents.

12. 日志

12.1. systemd

清除 10 天前的所有日志:

```shell
sudo journalctl --vacuum-time 10d
```

输出特定服务的日志:

```shell
sudo journalctl --unit github-actions-runner
```

注意希望使用如果非 `sudo` 身份查看日志,需要在 `systemd-journal` 用户组当中.

13. 文件系统

获取用于挂载标识的 UUID:

```shell
sudo blkid /dev/sda2
```

13.1. 符号链接

路径分为逻辑路径与物理路径.如果解析符号链接到实际目录,那么是物理路径,否则是逻辑路径,`pwd` 命令默认输出逻辑路径.

```shell
pwd -P # 物理路径
```

13.2. ZFS

`zpool history` 可以查看操作历史,包括 `zpool create`, `zpool add` 等的记录.前提是 `pool` 还在,如果已经被 `zpool destroy` 了,相应的历史都会删除.

```shell
zpool create -o ashift=12 oskar draid:2d \
/dev/disk/by-id/ata-TOSHIBA_MQ02ABF100_Z6NUPPCRT \
/dev/disk/by-id/scsi-35000cca02d7d75ec \
/dev/disk/by-id/scsi-35000cca02d7d78f8 \
-f  # in case they are of different sizes
```

dRAID 是 raidz(1-3)的封装,手册中给出的创建格式为:

```
draid[parity][:datad][:childrenc][:sparess]
...
In regards to I/O, performance is similar to raidz since, for any read, all D data disks must be accessed.
Delivered random IOPS can be reasonably approximated as floor((N-S)/(D+P))*single_drive_IOPS.

A dRAID with N disks of size X, D data disks per redundancy group, P parity level, and S distributed hot
spares can hold approximately (N-S)*(D/(D+P))*X bytes and can withstand P devices failing without losing
data.
```

`children` 是所有盘(包括热备盘)的数量, `parity` 指定奇偶校验级别(1-3),默认为 1.

例如 `draid2:7d:10c:1s` 表示 10 个磁盘中设置 1 个热备盘,剩下 9 个磁盘为一个 `group`,含有 7 个数据盘以及 2 个校验盘.这可以承受 2 个磁盘的损坏,并在第一个损坏发生时立刻投入 1 个热备盘进行恢复(所以实际上能承受 3 个磁盘损坏而不丢失数据).

单个 `group` 相对于单块硬盘的吞吐量为:

$$\left\lfloor \frac{c-s}{d+p} \right\rfloor$$

其中c是 `children` 的数量.注意这里是理论值,实际上还受内存缓存,是否开启压缩等因素影响.

单个 `group` 相对于单块硬盘的存储量为:

$$\frac{d(c-s)}{d+p}$$

注意奇偶校验和热备实际上是分散在所有磁盘上的,这也是为什么不能创建后再修改热备盘的数量.

为 ZFS 文件系统添加 dataset:

```shell
sudo zfs create -o compression=zstd -o mountpoint=/home oskar/home
```

也可以改变挂载点

```shell
sudo zfs set mountpoint=/srv oskar/home
```

参考:

- `zpoolconcepts(7)`
- `zpool-create(8)`
- `zfs-create(8)`
- `zfsprops(7)`
- ZFS dRAID Basics and Configuration - Thomas-Krenn-Wiki-en
- Linux 文件大小浅谈
- What does the ZFS Metadata Special Device do? · openzfs/zfs · Discussion 14542
- 系统中的大多数文件有多大? - Farseerfc 的小窝
- zpoolconcepts.7 — OpenZFS documentation
- OpenZFS dRAID - A Complete Guide - Resources - TrueNAS Community Forums

13.3. Btrfs

13.4. NFS

参考:

- NFSServerSetup - Debian Wiki
- NFSv4Howto - Community Help Wiki

在服务端上:

```shell
apt install nfs-kernel-server
sudo -e /etc/exports
```

写入以下内容:

```
/oskar/elisabeth 192.168.0.3(rw,sync,no_root_squash,no_subtree_check)
```

在客户端上:

```shell
apt install nfs-common
mount -t nfs4 -o proto=tcp,port=2049 heloise.adamanteye.cc:/oskar/elisabeth /srv
```

或者写入`/etc/fstab`:

```
heloise.adamanteye.cc:/oskar/elisabeth  /srv/ nfs4  _netdev,auto,defaults 0 0
```

14. 虚拟化技术

14.1. Docker

14.1.1. 构建镜像

从含 `Dockerfile` 的路径构建镜像:

```shell
docker buildx build --tag nvchecker:latest .
```

14.1.1.1. 最小化打包

使用 alpine 镜像,并且从 alpine 安装软件包时使用`--no-cache` 选项:

```shell
apk add --no-cache bash
# 等价于
apk update && apk add bash && rm -rf /var/cache/apk/*
```

如果使用 debian 镜像,类似的减少体积的方式是

```shell
apt-get update && \
apt-get install -y --no-install-recommends bash && \
rm -rf /var/lib/apt/lists/*
```

14.1.1.2. 可重复构建

14.1.2. 运行容器

交互式运行容器,设置代理,挂载本地路径:

```shell
docker run -it --rm --name debian --network host -e HTTP_PROXY=http://[::1]:10801 -v "$HOME/
Documents/debian:/home/debian:rw" -v "/etc/wgetrc:/etc/wgetrc:ro" -e DEBFULLNAME -e DEBEMAIL
ghcr.io/adamanteye/debian-builder:master
```

14.1.3. 清理

删除所有未使用镜像(不止 dangling):

```shell
docker image prune -a
```

14.1.4. 配置网络

在中国,免不了需要配置各种代理,为 docker daemon 配置代理可以通过编写`/etc/docker/daemon.json` 实现:

```json
{
  "proxies": {
    "http-proxy": "http://[::1]:10801",
    "https-proxy": "http://[::1]:10801"
  }
}
```

如果要为 docker container 配置代理,可以使用`--add-host` 配置容器与宿主机网络的映射,例如:

```shell
docker run -it --rm --add-host=host.docker.internal:host-gateway --name typst ghcr.io/adamanteye/typst:latest
```

进入容器后查看/etc/hosts,发现域名已经解析到宿主机地址:

```
172.17.0.1 host.docker.internal
```

在宿主机上,可以监听 172.17.0.1 这个地址并提供代理.而在容器中指定相应的环境变量:

```shell
export http_proxy='http://host.docker.internal:10801'
```

15. 网络服务

15.1. 邮件

参考 Installation & initial configuration - maddy

15.2. nginx

启用 OSCP Stapling:

```
ssl_stapling on;
ssl_stapling_verify on;
```

15.3. caddy

mholt/caddy-webdav 为 caddy 扩展了 webdav 模块.

15.4. 压力测试

用 hatoo/oha: Ohayou(おはよう)产生 HTTP 流量.

16. 内核

16.1. Wayland 图形栈

图形栈的任务是连接不同的输入设备:键盘,鼠标,触摸板,触摸屏,绘图板.将输入事件分配到不同的客户端,并且将不同客户端的输出显示在桌面上.

Wayland 是类 Unix 系统的下一代显示服务器.

evdev 内核 evdev (event device)是 Linux 内核以及 Free BSD 中的通用输入事件接口,与之对应的用户空间库称为 libevdev.

libinput Wayland 下工作在 libevdev 上的另一层抽象,为合成器提供统一的处理输入事件的方式.

DRM 内核 Direct Rendering Manager 是处理现代显卡的内核子系统,它检测到的显卡列在/dev/drm/当中.主节点(primary node)名称形如 card0,允许特权操作.渲染节点名称形如 renderD128,允许非特权操作(渲染或视频解码).

libdrm 内核 DRM 的用户空间部分.

Mesa Vulkan 和 OpenGL 都是图形 API. Mesa 是 Linux 上基于 Intel 以及 AMD GPU 驱动程序对 Vulkan 和 OpenGL 的实现,是位于内核驱动程序之上的高级层.

xcbcommon libinput 通过 scancode 的形式传递键盘事件,而 xcbcommon 负责将其解释为有意义的通用键盘符号.

pixman 高效操作像素缓冲区.

libwayland Wayland 协议的最常用实现,C 语言编写.它提供了从 XML 定义文件生成 C 头文件以及胶水代码的工具 `wayland-scanner`.

16.1.1. 协议

Wayland 协议由多层抽象构成,最基本的是 Wire Protocol.

16.1.1.1. Wire Protocol

线路协议是 32 字节单位的流,按照主机的端序编码.

基本类型:

int, uint 32 比特的有符号或无符号整数

fixed 24.8 比特的有符号定点数

object 32 比特的对象 ID

new_object 32 比特的对象 ID,且接受时会为其分配内存

string 编码通常用 UTF-8

array 任意数据的 blob

线路协议是用消息构建的,每条消息都是事件(服务器到客户端)或请求(客户端到服务器).事件作用在 `object` 上.

16.2. dkms

17. 其他应用

17.1. Telegram Bot

首先阅读 From BotFather to 'Hello World',这里讲解了机器人开发的基础知识.

18. 服务器

18.1. Dell Power Edge R630

阵列卡是 H330(小卡),可以在 BIOS 里面改成 HBA 模式,即硬盘直通.

双路 E5-2680 处理器,2 条 32GB 内存,外加 4 个 2.5 英寸硬盘在开机空载时的耗电量大约为 161W.仅主板通电的功率为 10W 左右.