Table of Contents

Based on https://missing.csail.mit.edu/2020/potpourri/

Keyboard remap

- remapping keys on keyboard:
  - esc to caps lock on press
  - ctrl to caps lock on hold
  - Windows:
    * autohotkeys
    * autohotkey script for caps lock remap. Place shortcut in startup folder so it runs on startup: `Win-R` > `shell:startup`

Daemons

- background processes
- programs that run as daemons usually have `d` at end of name
  - e.g. `sshd`: SSH daemon listens for incoming SSH requests and checks if remote user has credentials to log in
- `systemd`: Linux system daemon is the most common solution for running and setting up daemon processes
  - `systemctl status` lists currently running daemons

  - `systemctl` is used to interact with `systemd` through `enable`, `disable`, `start`, `stop`, `restart`, check `status`
  - `systemd` has an accessible interface for configuring and enabling new daemons or services. e.g. to run a simple python app:

    ```
    # /etc/systemd/system/myapp.service
    [Unit]
    Description=My Custom App
    After=network.target

    [Service]
    User=foo
    Group=foo
    WorkingDirectory=/home/foo/projects/mydaemon
    ExecStart=/usr/bin/local/python3.7 app.py
    Restart=on-failure

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

- `cron` can be used to perform scheduled tasks

FUSE - Filesystem in User Space

- modern software is composed of smaller building blocks. The OS supports different filesystem backends because there is a common language of what operations a filesystem supports
- e.g. when you run `touch`: `touch` performs system call to kernel to create the file, kernel performs filesystem calls to create the file
  - N.B. UNIX filesystems are traditionally implemented as kernel modules and only the kernel is allowed to perform fs calls
- allows filesystems to be implemented in a user program
- lets users run user space code for filesystem calls, and bridges necessary calls to kernel interfaces
- i.e. users can implement arbitrary functionality for filesystem calls
- e.g. `sshfs`: FUSE can lets you perform an operation in a virtual filesystem, that operation is forwarded through SSH to a remote machine, performed there, and output is returned to you. This way local programs see the file as if it was local while in reality it's on a remote server

Examples - sshfs - Open locally remote files/folder thorugh an SSH connection. - rclone - Mount cloud storage services like Dropbox, GDrive, Amazon S3 or Google Cloud Storage and open data locally. - gocryptfs - Encrypted overlay system. Files are stored encrypted but once the FS is mounted they appear as plaintext in the mountpoint. - kbfs - Distributed filesystem with end-to-end encryption. You can have private, shared and public folders. - borgbackup - Mount your deduplicated, compressed and encrypted backups for ease of browsing.

Backups

Core features of backup solution - versioning: access your history of changes and efficiently recover files - deduplication: store incremental changes and reduce storage overhead - security: encryption; access control for reading/deleting backups - verification: regular checks to ensure backups can be used to recover data

3-2-1 Rule

at least 3 copies of your data 2 copies in different mediums 1 of the copies being offsite

Versioning

- RAID is not a backup; mirroring in general is not a backup solution

APIs

- responses usually formatted as JSON: pipe response through a tool like jq to parse
- authentication: usually via a token included with the API request
- OAuth is a commonly used protocol that gives tokens that can act as you on a given service. These tokens need to be kept secret

Common command line patterns

- `--help` commonly supported for brief usage instructions
- many tools that perform irrevocable change have a dry run mode so you can check output before performing the action
- interactive flag: prompts before destructive actions
- `--version/V` to print version
- `--verbose/-v` for verbose output
  - `-vvv` for more verbose output
  - `--quiet` to only print on error

- – in place of a filename means `STDIN` or `STDOUT` depending on context
- `-r` to make recursive
- `--`: special argument that prevents processing of flags and options, so that you can pass things that look like flags without them being interpreted that way. e.g. `rm -- -r` to delete file called `-r`

Booting + Live USBs

When machine boots up, before OS loads, BIOS/UEFI initialises the system. This can be interrupted to configure this layer of software, allowing you to configure hardware-related settings, and also to enter the boot menu to boot from an alternate device instead of the hard drive

Live USBs are USB flash drives containing an operating system. This involves burning an `.iso` file to the disk. They are useful for OS recovery.

Docker, Vagrant, VMs, Cloud

- Virtual machines: emulate a whole computer system, including the OS
  - useful for creating isolated environments for testing, dev, exploration
- Vagrant: lets you describe machine configs (OS, services, packages, …) and instantiate VMs with `vagrant up`
- Docker: similar; uses lighter weight containers instead
- VMs can be rented on AWS, Google Cloud, DigitalOcean. Use cases:
  - cheap always-on machine with public IP for hosting
  - need lots of CPU, disk, RAM, GPU
  - many machines than you have access to (e.g. 1000 computers for a couple of minutes)

Filesystem Hierarchy Standard

Wikipedia - `/bin`: essential command binaries - `/sbin`: essential system binaries, usually run by root - `/dev`: device files; special files that interface with hardware devices - `/etc`: host-specific system-wide configuration files - `/home`: home dirs for users in the system - `/lib`: common libraries for system programs - `/opt`: optional application software - `/sys`: contains info and config for system - `/tmp`, `/var/tmp`: temporary files, deleted between reboots - `/usr/`: read only user data - `/usr/bin`: non-essential command binaries - `/usr/sbin`: non-essential system binaries - `/usr/local/bin`: binaries for user compiled programs - `/var`: variable files, e.g. logs, caches