1. What is the purpose of the `/etc/passwd` file, and how is it related to `/etc/shadow`?

- `/etc/passwd`: This file contains all the user information on your system
- `/etc/passwd`: This file contains password (hashed passwords) of the users.
  the `/etc/passwd` is readable for all but `/etc/shadow` is only readable for root and members of the `shadow` group

2. How can I enforce a password change policy for users every 100 days?

   we can use the `chage` tool to change these numbers. If you issue the `chage <username>` the system will prompt you for all the parameters one by one. Also it is possible to use switches to change specific parameters on command line.
   - `-E`: Set the expiration date. Date can be a number, in YYYY-MM-DD format or -1 which will mean *never*
3. Create a cron job that collects all system log files (`/var/log/syslog`) every day at 3:00 AM and archives them weekly on Friday night. Then move the weekly archive to another location.

- we write a script for example at: `/usr/local/bin/syslog_archive.sh`

```bash
#!/bin/bash

DAILY_LOG="/var/log/syslog"
ARCHIVE_DIR="/var/log/archives"
BACKUP_DIR="/backup/logs"
WEEKLY_ARCHIVE="syslog_week_$(date +%Y-%W).tar.gz"

mkdir -p "$ARCHIVE_DIR"
mkdir -p "$BACKUP_DIR"

cp "$DAILY_LOG" "$ARCHIVE_DIR/syslog_$(date +%Y-%m-%d)"

# run on Friday at 23:59
if [ "$(date +%w)" -eq 5 ] && [ "$(date +%H:%M)" = "23:59" ]; then
    tar -czf "$ARCHIVE_DIR/$WEEKLY_ARCHIVE" "$ARCHIVE_DIR/syslog_*"
    mv "$ARCHIVE_DIR/$WEEKLY_ARCHIVE" "$BACKUP_DIR/"
    # optional
    rm -f "$ARCHIVE_DIR/syslog_*"
fi
```

- `chmod +x /usr/local/bin/syslog_archive.sh`

- `crontab -e`

```
0 3 * * * /usr/local/bin/syslog_archive.sh
59 23 * * 5 /usr/local/bin/syslog_archive.sh
```

4. Recreate question 3 using `systemd-run` instead of a traditional cron job.

```
systemd-run --on-calendar="*-*-* 03:00:00"
/usr/local/bin/syslog_archive.sh
systemd-run --on-calendar="Fri *-*-* 23:59:00"
/usr/local/bin/syslog_archive.sh
```

5. How does `hwclock` work? What happens to the hardware clock when the device is powered off?

There is a clock in your computer; a hardware clock on your motherboard! It has its own battery and keeps the time even when the computer is off. When the system boots, the OS reads this **hardware time** and it sets it's own **system time** based on the hardware clock and uses this clock whenever it needs to know the time.

Hardware clock can be the set on localtime (the time shown on your clock wherever you are) or UTC time (standard time). The `hwclock` can be used to show the time based on the hwtime.

6. Use Chrony to set up a time synchronization service where vm1 acts as the Chrony server and vm2 synchronizes its time using vm1 as the source (do not use external time servers).

After installing Chrony on VM1, we should comment out the default external servers (or remove it).

`# pool ntp.ubuntu.com iburst`

Then, we allow access from VM2:

`/etc/chrony/chrony.conf` on `VM1`:

`allow 192.168.122.0/24 # the subnet is an example`

The **server** can serve many clients, so we allow a range (subnet).

we also enable local time source:

`local stratum 10`

Chrony uses UDP port 323. We must open this port on VM1 to allow `VM2` to connect.

https://unix.stackexchange.com/q/765982
In addition, port `323` has to be opened in the firewall in order to connect from a remote system.

```
sudo ufw allow 323/udp
sudo ufw reload
```

```
sudo systemctl restart chrony
sudo systemctl enable chrony
```

After installing Chrony on VM2, we should comment out the default external servers (or remove it).

`# pool ntp.ubuntu.com iburst`

Then, we add our local server VM1:

`/etc/chrony/chrony.conf` on `VM2`:

`server 192.168.122.10 iburst` # the ip is an example`

The **client** syncs with a specific time source, so you specify **one IP** (or hostname).

https://unix.stackexchange.com/q/765982
When operating as a client, chrony doesn't require any open *inbound* ports, and a typical default firewall configuration allows unrestricted outbound connections. Chrony will work fine regardless of whether or not port 323 is allowed by the firewall.

```
sudo systemctl restart chrony
sudo systemctl enable chrony
```

with `chronyc sources`, `chronyc tracking` and `chronyc activity` we can check the setup.

7. Explain in detail how the system clock works. What does it mean to synchronize the system time using NTP? What happens if the NTP server goes down? What is meant by 'time drift' in the context of NTP?

The system clock in Linux is the software-based clock maintained by the operating system kernel to track the current time and date.

NTP protocol uses NTP servers to find out the accurate time shown by best atomic clocks on this planet. One of the most famous severs used by ntp people is `pool.ntp.org`. If you check that website you will see that it is a **pool** of ntp servers and by giving your NTP server the `pool.ntp.org`, it will be redirected to one of the many ntp servers available on that pool.

If NTP server goes down:

- The system continues running with the last known good time.
- Over time, the system clock may begin to drift (lose/gain accuracy).
- Most NTP clients will retry other servers if a pool is configured.

Time drift refers to the gradual deviation of the system clock from the actual time due to inaccuracies in the local hardware timer.

8. Write a script that logs user login attempts when a wrong password is entered. This script should be implemented as a `systemd` service unit.

https://www.networkworld.com/article/969378/monitoring-failed-login-attempts-on-linux.html

`/usr/local/bin/monitor_failed_logins.sh`:

```
#!/bin/bash

LOG_FILE="/var/log/failed_logins.log"
AUTH_LOG="/var/log/auth.log"

touch "$LOG_FILE"
chmod 640 "$LOG_FILE"

tail -Fn0 "$AUTH_LOG" | while read -r line; do
    if [[ "$line" =~ "Failed password" ]]; then
        # we can extract other info like ip, username with grep or awk
        echo "[$timestamp] Failed login attempt ...
    fi
done
```

The script uses `tail -Fn0 "$AUTH_LOG"` to follow `/var/log/auth.log` and capture new lines as they appear (*when a wrong password is entered* in the question).

`chmod +x /usr/local/bin/monitor_failed_logins.sh`

https://unix.stackexchange.com/a/797350

`/etc/systemd/system/monitor-failed-logins.service`:

```
[Unit]
Description=Monitor failed login attempts
After=network.target

[Service]
```

```
ExecStart=/usr/local/bin/monitor_failed_logins.sh
Restart=always
User=root

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

`After=network.target`: Ensures that the network is available before the script starts, helpful if the script requires network access.

`Restart`: Set this to on-failure to restart the service if it fails. Other options include always (restarts regardless of the exit code) or no (never restarts).

`WantedBy=multi-user.target`: You're telling systemd to start your service during the system's multi-user mode, which is active on most Linux systems running in a non-GUI environment.
This setup is ideal for server processes, background scripts, and other tasks that need to be available as soon as the system is ready for normal operation but don't require a graphical environment.

https://dev.to/piyushbagani15/automate-your-scripts-with-systemd-services-benefits-and-step-by-step-guide-3nik

```
systemctl daemon-reload
systemctl enable monitor-failed-logins.service
systemctl start monitor-failed-logins.service
```

How to ensure that logs will never exhaust your disks (DoS attack).
https://security.stackexchange.com/a/147273

9. Set up an `rsyslog` server on vm1 and configure it to receive and store logs from `/var/log`, `journalctl`, and `dmes` from other systems.

I use below solution:

https://www.makeuseof.com/set-up-linux-remote-logging-using-rsyslog/