1 A clean setup

1. **Where to copy the dice module for it to be officially known to the kernel?**

   Modules should be copy to the directory `lib/modules/$(uname -r)`, and we can use modprobe command to insert the module as well as all its dependency. For example:

   `sudo modprobe [module name]`

2. **What command to run in order to generate the `modules.dep` and `map` files?**

   Using depmod command:

   `sudo depmod [module name]`

   and then we can check the dependency in the `modules.dep` file.

3. **How to ensure the dice module is loaded at boot time, and how to pass it options?**

   The directory: `/etc/modules-loade.d/` contains all the modules need to be load at boot time. We can add a `[some name].conf` file under this directory and write the module's name in it, one per line.

   If we want to set the parameter of the module, we can add a `[some name].conf` file under the directory: `/etc/modprobe.d/` and set the parameter after the module name, one per line. For example:

   `options [module_name] [parameter_name]=[parameter_value]`

   Also we could set it using Udev.

4. **How to create a new `friends` group and add grandpa and his friends to it?**

   To create a group:

   ```
   groupadd [group name]
   groupadd friend
   ```

   To add the user to this group:

   ```
   usermod -a -G [groupname] [username]
   usermod -a -G friend grandpa
   ```

5. **What is `udev` and how to define rules such that the group and permissions are automatically setup at device creation?**

   Udev is the Linux subsystem that supplies the computer with device events. It enables you to define rules and functions triggering your computer to do a specific action when a specific device is plugged in.

   To define rules, we can write rules in the files under the directory

   `/etc/udev/rules.d//etc/udev/rules.d/` and the rules are processed in lexical order. For example, we can create a file called: `10-dice.rules`, and then write:

   ```
   KERNEL=="dicedevice", GROUP="friend", MODE="0660"
   ```

2 A discreet gambling setup

2.1 Hacking mum's computer

1. Find the file: `~/.bashrc`, and add the command export in it:

   ```
   export PATH="[some path]:\$PATH"
   ```

   Then load the change using: source `~/.bashrc`

   To forget the change, using the command: `source /etc/environment`

2. It will echo: `"su: Authentication failure"`

3. Using command: `read -sp [PROMPT] [variable]`

4. First install an email app with: `sudo apt-get install mailutils`

   Then set the configuration and send email with the command: `mail -s xxx@sjtu.edu.cn` along with the content and end up with a "."

2.2 Automatic setup

1. **What is `systemd`, where are service files stored and how to write one?**

   `systemd` is a basic building block of a linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. The service files are stored in the `/etc/systemd/system`. We can write on as it is in the `/etc/systemd/system/multi-user.target.wa` Basically it follows the following format:

   ```
   [Unit]
   Description=Uncomplicated firewall
   Documentation=man:ufw(8)
   DefaultDependencies=no
   Before=network.target

   [Service]
   Type=oneshot
   RemainAfterExit=yes
   ExecStart=/lib/ufw/ufw-init start quiet
   ExecStop=/lib/ufw/ufw-init stop

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

   (a) [**Unit**] part is used to notify us the brief `Description` of the unit, `After` and `Before` (optional) are to specify what need to be started before or after the program starts. `Wants` and `Requires` specify whether the service is dependent on other service files get an error, the service that requires that service will also fail.

   (b) [**Service**] specify how to start our service. `EnvironmentFile` specifies where the parameter configuration is. `ExecStart` defines what commands we run when we start the process. `Type` specifies the way we are going to start the process. e.g. `Type=simple`, will run the `ExecStart` commands in the main process; `Tyoe=forking`, we use a `fork()` to creat a process and run the `ExecStart` commands in the child process.

(c) [**Install**] specify how to install this service config file. `WantedBy` shows the target of the service. `WantedBy=multi-user.target` will move the service into the directory `multi-user.target.wants`.

2. **How to get a `systemd` service autostart?**

   We use the code below and specify the service we want to build:

   ```
   sudo systemctl enable [someservice]
   ```

3. **What is the difference between running `tmux` from the `systemd` service and from the `gp-2.10 daemon`?**

4. **What is `dbus` and how to listen to all the system events from the command line?**

   D-Bus is a message bus system, a simple way for applications to talk to each other. Basically it it an IPC way and it is very powerful since it allows one process to use the methods (APIs) of another process. We can use the following command to listen to all the system events.

   ```
   dbus-monitor --system
   ```

   we could take advantage of the message in the d-bus of the whole system, thus checking whether mum or grandpa is connecting.

5. **What is `tmux`, when is it especially useful, and how to run a detached session?**

   `tmux` is a terminal multiplexer.

   It can be used to create a separate session in the terminal.

   ```
   tmux new -s <session-name>
   tmux new -L <socket-name>
   ```

   use `tmux split-window` or `Ctrl+b%` to divide into 2 windows. Press `Ctrl+b` again, and type `;` or `o` to move to the previous or next window. This allow us to run commands in different window and without being discovered by each other. This is extremely useful when we are serving different clients on the computer.

6. **What is `tripwire`, what are some alternatives, and why should the configuration files also be encrypted and their corresponding plain text deleted?**

   `tripwire` is a free software security and data integrity tool for monitoring and alerting on specific file change(s) on a range of systems which is provided on the Linux. There are many alternatives like OSSEC, Splunk, SolarWinds and so on. The plain text of configuration could leak the information about the system, so the software keeps track of its encrypted version in case of reverse-engineering.

7. **What is `cron` and how to use it in order to run tasks at a specific time?**

   `cron` is a time-based job scheduler in Unix-like computer operating systems. The actions of `cron` are driven by a crontab (cron table) file, a configuration file that specifies shell commands to run periodically on a given schedule. The crontab files are stored where the lists of jobs and other instructions to the cron daemon are kept. Users can have their own individual crontab files and often there is a system-wide crontab file (usually in /etc or a subdirectory of /etc) that only system administrators can edit.

   Each line og a crontab file represents a job, which looks like this: `* * * * * <command to execute>`, where the stars $*$ stands for minute(0-59), hour(0-23), day of the month (1-31), month (1-12) and day of the week (0-6) (Sunday to Saturday).