Linux Administration

Jose L. Muñoz, Oscar Esparza, Juanjo Alins, Jorge Mata
Telematics Engineering
Universitat Politècnica de Catalunya (UPC)

Outline

Outline

Accounts I

- An account provides the user with configuration settings and preferences.
- Typically, also with some space in disk (normally under the `/home` directory).
- We can find different types of users (or accounts):
  - **Superuser**. Administrator or `root` user. This user has a special account which is used for system administration. The `root` user has granted all the rights over all the files and processes.
  - **Regular users**. A user account provides access to the system with a limited access to critical resources such as files and directories.
  - **Special users**. The accounts of special users are not used by human beings but they are used by internal system services. An example is teh user www-data, which is used by Web servers to access to documents and resources.

Configuration Files

`/etc/passwd` — text file containing user account information

`/etc/shadow` — text file containing user password information

`/etc/group` — text file containing groups

`/etc/gshadow` — text file that may contain group passwords

`/etc/skel` — directory that contains files and directories that are automatically copied over to a new user's home directory when such user is created by the `useradd` command.

- Use `vipw` and `vigr` so that the file cannot be edited while others are changing it.

/etc/passwd

- Example lines of `/etc/passwd`:

```
telematics:x:1000:1000:Mike Smith:/home/telematics:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

1. `telematics`: this field contains the name of the user, which must be unique within the system.
2. `x`: this field contains the encoded password. The ”x” means that the password is not in this file but it is in the file `/etc/shadow`.
3. `1000`: this field is the number assigned to the user, which must be unique within the system.
4. `1000`: this field is the number of the default group. The members of the different groups of the system are defined in the file `/etc/group`.
5. `Mike Smith`: this field is optional and it is normally used to store the full name of the user.
6. `/home/telematics`: this field is the path to the home directory.
7. `/bin/bash`: this field is the default shell assigned to the user.

/etc/shadow

- `/etc/shadow` contains the encrypted password liked with the user name:

```
telematics:a1gNcs82ICst8CjVJS7ZFCVnu0N2pBcn/:12208:0:99999:7:::
```

1. User name : It is your login name
2. Password: It your encrypted password. The password should be minimum 6-8 characters long including special characters/digits
3. Last password change (lastchanged): Days since Jan 1, 1970 that password was last changed
4. Minimum: The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
5. Maximum: The maximum number of days the password is valid (after that user is forced to change his/her password)
6. Warn : The number of days before password is to expire that user is warned that his/her password must be changed
7. Inactive : The number of days after password expires that account is disabled
8. Expire : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used

/etc/group and /etc/skel

- `/etc/group` contains information of system groups:

```
group_name:password_group:ID_group:users_list
```

- Example:

```
users:x:1000:telematics,otheruser
```

- `/etc/skel` contains the ”skeleton” that is copied to each user’s home when the user is created.

Outline

Creating a User Account

- As **root** you can manually create a new user account:
  1. Find the next available UID and GID numbers, or use the ones provided, checking they are unique.
  2. Add an entry to the `/etc/passwd` and `/etc/shadow` files. This includes a hash of the password into `/etc/shadow`.
  3. Create the home directory (under /home).
  4. Create a mail spool file `/var/spool/mail/username`.
  5. Copy the files and directories from `/etc/skel` to the home directory.
  6. Change the ownership of the home directory and all its contents to the user, and the group ownership to the primary group of the user.
  7. Change the ownership of the mail spool file to the user, and make the group owner equal to `mail`.
- We have `useradd` (or `adduser`) that does all the previous steps.

Management Commands I

- `useradd`: add a new user.
- `userdel`: delete a user.
- `usermod`: modify a user (e.g. add it to a group). However, be careful because it removes the user from any groups not specified. To add a user to a group you can use instead `gpasswd -a user group`.
- `groupadd`, `groupdel`, `groupmod`: management of groups.
- `passwd`: change your password. If you are root you can also change the password of other users.
- `su`: switch user (change of user).
- `who`: shows who is logged in the system and their associated terminals.
- `last`: shows a list of last logged users.
- `id`: prints the real and effective user and group IDs.
- `groups`: prints the groups which the user belongs to.

Management Commands II

- `chage`: change and list user password expiry information (e.g. `chage -l`).
- `chown`: can be used to change the owner and the group of a file.
- `chgrp`: can be used to change the group of a file.
- Examples of the `chown` command:

```
# chown telematics notes.txt
```

- The “-R” option makes recursive:

```
# chown -R student.mygroup directory
```

- You can suspend (“lock”) an account by inserting an exclamation mark ‘`!`’ in front of the password field in /etc/shadow using `vipw`.

Special Accounts

- Special users have accounts that generally have a user ID that is lower than some particular value.
- This value is recommended by each Linux distribution.
- These accounts are not intended for human beings.
- They do not need an access to a login shell.
- This is accomplished using `/bin/false` or `/sbin/nologin` and also disabling the password.
- Example configuration line of FTP User in /etc/passwd:

```
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

sudoers File I

- `sudo` uses the sudoers file (/etc/sudoers) to determine if a user can execute a certain command with certain privileges.
- Define which users or groups will be able to execute certain commands (or even any command) as root.
- Users will not have to know the password of the root.
- `sudo` makes it easier to implement the principle of ”least privilege“.
- It also logs all commands and arguments so there is a record of who used it for what, and when (in /var/log/auth.log).
- Use `visudo` to edit /etc/sudoers.

sudoers File II

- Example of (/etc/sudoers):

```
# This file MUST be edited with the 'visudo' command as root.
# See the man page for details on how to write a sudoers file.
Defaults        env_reset
# Host alias specification
# User alias specification
User_Alias NET_USERS = %netgroup
# Cmnd alias specification
Cmnd_Alias NET_CMD =/usr/local/sbin/simtun, /usr/sbin/tcpdump, /bin/ping,
/usr/sbin/arp, /sbin/ifconfig, /sbin/route, /sbin/iptables, /bin/ip
# User privilege specification
root    ALL=(ALL) ALL
# Uncomment to allow members of group sudo to not need a password
# (Note that later entries override this, so you might need to move
# it further down)
# %sudo ALL=NOPASSWD: ALL
NET_USERS       ALL=(ALL)       NOPASSWD: NET_CMD
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

- Users belonging the group “netgroup” can access some network-related commands.

Outline

Special File Permissions

- Three special types of permissions: **setuid**, **setgid** and **sticky bit** are available for executable files and public directories.
- These permissions are activated adding an octal number to the access mask: setuid (1000), setgid (2000) and sticky bit (4000).
- You can also add the setuid:

```
$ chmod u+s script
$ chmod u-s script
```

- You can also add the setgid:

```
$ chmod g+s myscript
$ chmod g-s myscript
```

- You can add/remove the sticky bit:

```
# chmod +t /home/user1/data
# chmod -t /home/user1/data
```

setuid I

- When set on an executable file, a process that runs this file is granted access based on the owner of the file (typically root).
- Allows a user to access files and directories that are normally only available to the owner.
- To test this permission we can use the following C program:

```
#include <stdio.h>
#include <unistd.h>

/* test_program */
main(){
   printf("UID: %d, EUID: %d\n",getuid(),geteuid());
}
```

- Every process has a real user identifier (UID) and an Effective UID (EUID).
- In the general case, the real and effective UIDs are the same.

setuid II

- With setuid we alter that behavior:

```
user1$ gcc -o test_program test_program.c
user1$ chmod 755 test_program
user1$ ./test_program
UID: 1000, EUID: 1000
```

- If you test the program with the setuid set and with another user (user2 with uid 1001):

```
user2$ ./test_program
UID: 1001, EUID: 1000
```

- Extremely careful when setting setuid to files owned by root (security risk).

setgid

- When setgid is applied to a file:
  - The process's effective group ID (GID) is changed to the group of the file.
  - Then, a user is granted access based on permissions granted to that group.
- When setgid permission is applied to a directory:
  - Files created by process in a directory with gid belong to the group of the directory, not the group to which the process belongs.
  - Any user who has write and execute permissions in the directory can create a file.
  - Created files belong to the group of the directory.

Sticky Bit

- The sticky bit is a permission bit that protects the files within a directory.
- If the directory has the sticky bit set, a file can be deleted only by the owner of the file, the owner of the directory, or by root.
- This special permission prevents a user from deleting other users' files from public directories such as /tmp.

Outline

Cron

- Cron is a daemon/service that executes shell commands periodically on a given schedule.
- Cron is driven by a crontab (its configuration file).
- List of active crontab entries:

```
$ crontab -l
```

- You can create a crontab file:

```
$ crontab -e
```

- The previous command opens a text editor with a new blank crontab file, or it will open your existing crontab if you already have one.
- The cron file is not thought to be directly accessed by users, but if you are curious it is stored at: `/var/spool/cron/crontabs/user`

Cron Configuration I

- Each crontab line has six fields (separated by white spaces):
  1. **m** (0-59): representing the minute of the hour.
  2. **h** (0-23): representing the hour of the day.
  3. **dom** (1-31): representing the day of the month.
  4. **mon** (1-12): representing the month of the year.
  5. **dow** (0 - 6 ; sunday=0): representing the day of the week.
  6. **command**: which is the command to be run, exactly as it would appear on the command line.
- For the numbers you can use:
  - A **list of numbers**: 15,30,45.
  - A **range of numbers**: 10-20.
  - You can mix the previous: ”0-4,8-12”.
  - The **asterisks** mean ”every”.
  - The **slash** is to specify a step value. E.g. ”0-23/2” is equivalent to ”0,2,4,6,8,10,12,14,16,18,20,22”.
  - Steps are also permitted after an asterisk, so if you want to say ”every two hours”, you can use ”*/2”.

Cron Configuration II

- Examples:

```
* * * * * <command> #Runs every minute
30 * * * * <command> #Runs at 30 minutes past the hour
45 6 * * * <command> #Runs at 6:45 am every day
45 18 * * * <command> #Runs at 6:45 pm every day
00 1 * * 0 <command> #Runs at 1:00 am every Sunday
00 1 * * 7 <command> #Runs at 1:00 am every Sunday
00 1 * * Sun <command> #Runs at 1:00 am every Sunday
30 8 1 * * <command> #Runs at 8:30 am on the first day of every month
00 0-23/2 02 07 * <command> #Runs every other hour on the 2nd of July
```

- As well as the above there are also special strings that can be used:

```
@reboot <command> #Runs at boot
@yearly <command> #Runs once a year [0 0 1 1 *]
@annually <command> #Runs once a year [0 0 1 1 *]
@monthly <command> #Runs once a month [0 0 1 * *]
@weekly <command> #Runs once a week [0 0 * * 0]
@daily <command> #Runs once a day [0 0 * * *]
@midnight <command> #Runs once a day [0 0 * * *]
@hourly <command> #Runs once an hour [0 * * * *]
```

Cron Configuration III

- A double-ampersand "&&" can be used to run multiple commands consecutively.
- By default a cron job will send an email to the user account executing the cronjob.
- If this is not needed put the following command at the end of the cron job line:

```
>/dev/null 2>&1
```

- If you already have a crontab file, you can use it with the following command:

```
$ crontab -u <username> <crontab file>
```

- To remove your crontab file simply enter the following terminal command:

```
$ crontab -r
```

At

- If you need to do a ”one time command scheduling” you can use the `at` command.
- In this sense it is complementary to cron which usually is used to schedule periodic jobs.
- Example:

```
$ at 10:27 ls -l
warning: commands will be executed using /bin/sh
job 2 at Tue May 5 10:27:00 2015
```

- The output will be mailed to you (at your local user account) after it runs.
- You can use the `atq` to list the `at` commands in que and what time it is scheduled for.

Outline

rc.local

- Apps in **/etc/rc.local** are executed after boot (before logins).
- Environment variables are not set.
- Do not require input (can lead to halt).
- Example:

```
#!/bin/sh -e
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
# In order to enable or disable this script just change the execution
# bits.
# By default this script does nothing.

sleep 10
/bin/echo "testing" > /tmp/rc_local_test.log
exit 0
```

- To test that everything works fine:

```
$ sudo /etc/init.d/rc.local start
```

/etc/init.d

- In **/etc/init.d/** you can create your own script for the service or services that you want to run at boot time.
- A base script to construct a init.d script can be found in /etc/init.d/skeleton.
- Install:

```
$ sudo update-rc.d /etc/init.d/myservice defaults
```

- To remove a service:

```
$ sudo update-rc.d -f /etc/init.d/myservice remove
```

- You can also disable a service:

```
$ sudo update-rc.d /etc/init.d/myservice disable
```

Other methods

- In your **crontab file** you can use `@reboot` to tell cron to run a task when your system boots.
- **Display Server and Window Managers**:
  - To start graphical applications you can use configuration files of your display server.
  - If your display server is X, the typical files are
  - The file ~/.xinitrc, which is meant for when you start Xorg with the startx command.
  - If you are running a display manager instead, you will need an ~/.xsession script instead.
  - You can also configure startup scripts and commands after your window manager starts.

Outline

Outline

Log Systems

- Unix-like systems “log” events that happen.
- Some possible things you may see logged are kernel messages, system events, user runs any program, user runs a particular program, user calls a specific system function, user su’s to root, server events etc.
- As with any piece of software in Linux, you have options as to which system logger program you would like to use.
- Some popular ones include: syslog, metalog, sysklogd and rsyslog.
- All of these are good choices, but we are going to work with rsyslog, which is the default logger program in Ubuntu.

Log Locations

- The location of the various log files varies from system to system.
- On most UNIX and Linux systems the majority of the logs are located in /var/log.
- Some of these log files are:

/var/log/syslog	General message and system related stuff
/var/log/auth.log	Authenication logs
/var/log/kern.log	Kernel logs
/var/log/boot.log	System boot log
/var/log/wtmp	Login records file
/var/log/daemons.log	Logs for network daemons
/var/log/apache2/	Apache 2.0 access and error logs directory

- Some applications (e.g. `apache2`) have their own directory inside /var/log/.

Logrotate

- `logrotate` is a great utility for log files.
- Can provide automatic rotation, compression and mailing.
- `logrotate` is typically run under `cron`.
- The main configuration file is `/etc/logrotate.conf`.
- Many configuration belongs to the software packages, which put a file into directory `/etc/logrotate.d/`.
- Example:

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# system-specific logs may be configured here
```

Kernel Log (dmesg)

- All UNIX and Linux systems have a log that is actually part of the kernel (in memory).
- This way is the fastest and allows storing log info at boot time (before the filesystems are loaded).
- The data in this log contains information about the devices connected to the system and any faults and problems recorded by the system during the boot and operational process.
- In some systems the information is periodically dumped into a file (/var/log/dmesg).
- The most fresh information is only available using the `dmesg` command (for most UNIX/Linux variants).

Outline

rsyslog

- The rsyslog service is a daemon that runs the background.
- Accepts log entries and writes them to one or more individual files.
- All messages reported to syslog are tagged with the date, time, and hostname.
- Can have a single server that accepts log messages from a number of hosts.
- Highly configurable in /etc/rsyslog.conf and /etc/rsyslog.d.
- In these files you must specify rules.
- Every rule consists of two fields, a **selector field** and an **action field**.
- These two fields are separated by one or more spaces or tabs.
- The selector field specifies a pattern of facilities and priorities belonging to the specified action.

Selectors

- The selector field consists of two parts:
  - The **facility** is one of the following keywords:

    `auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, syslog, user, uucp and local0 through local7.`
  - The **priority** defines the severity of the message and it is one of the following keywords, in ascending order:

    `debug, info, notice, warning, err, crit, alert, emerg.`
- Additionally, the following keywords and symbols have a special meaning: “none”, “*”, “=” and “!”.

Actions & Examples I

- The action field of a rule describes the abstract term “logfile”.
- A “logfile” need not to be a real file but it can be also a named pipe, a virtual console or a remote machine.
- To forward messages to another host, prepend the hostname with the at sign “@”.
- Examples:

```
*.info;mail.none;news.none;authpriv.none;cron.none    /var/log/syslog
```

- The *.info means “Log info from all selectors”.
- However, after that, it says mail.none;news.none and so forth.
- What that means when all put together is “Log everything from these EXCEPT these things that are following it with ’.none’ behind them”.

Actions & Examples II

```
# Kernel messages are first, stored in the kernel
# file, critical messages and higher ones also go
# to another host and to the console
#
kern.*                          /var/adm/kernel
kern.crit                       @mylogserver
kern.=crit                      /dev/tty5
kern.info;kern.!err             /var/adm/kernel-info
```

- The first rule directs any message that has the kernel facility to the file /var/adm/kernel.
- The second statement directs all kernel messages of the priority crit and higher to the remote host *mylogserver*. This is useful, because if the host crashes and the disks get irreparable errors you might not be able to read the stored messages.
- The third rule directs kernel messages of the priority crit to the virtual console number five (`tty5`).
- The fourth saves all kernel messages that come with priorities from info up to warning in the file /var/adm/kernel-info. Everything from err and higher is excluded.

Actions & Examples III

- To activate the rsyslog service in a remote server, edit the /etc/rsyslog.conf file adding:

```
$ModLoad  imudp
$UDPServerRun  514
```

- Restart rsyslog:

```
# service rsyslog restart
```

Outline

Logging and Bash

- To view log files you can user the `tail -f` command:

```
# tail -f /var/log/syslog
```

- The `logger` command makes entries in the system log and it provides an interface with the log system for shells.
- To log a message indicating a system reboot:

```
telematics$ logger System rebooted
telematics$ tail -1 /var/log/syslog
Sep  7 11:28:02 XPS telematics: System rebooted
```

- To log a message contained in the /tmp/msg1 file:

```
$ logger -f /tmp/msg1
```

- To log the news facility critical level messages:

```
$ logger -p news.crit Problems in our system
```

Outline

Outline

Mounting a Filesystem

- The command to mount a storage device under a mount point is `mount`.
- If our first SATA disk has a second partition, the Kernel will map it in /dev/sda2.
- We can ”mount“ it to store /home:

```
# mount /dev/sda2 /home
```

- /dev/sda2 is the device and /home is the mount point.
- Linux can also mount WINDOWS filesystems such as fat32:

```
# mount -t vfat /dev/hdd1 /mnt/windows
```

Unmounting a Filesystem I

- If the file system of the pen-drive mapped in /dev/sdc1 is mounted under the directory or mount point `/media/pen`, then any operation on `/media/pen` will act in fact over the FS of the pen-drive.
- When we finish working with the pen-drive, it is important to ”unmount” the device before extracting it.
- This is because unmouting gives the opportunity to the OS of finishing all I/O pending operations.
- When `/media/pen` is unmounted, then all I/O operations over `/media/pen` are performed over the device that is currently mounting the directory (e.g. the system’s hard disk mounting /).

Unmounting a Filesystem II

- Unmouting is performed with the command `umount` using the mount point or the device name (also with GUI):

```
# umount /media/pen
```

or

```
# umount /dev/sdc1
```

- It is not possible to unmount an ”busy” (in use) file system.
- A file system is busy if there is a process using a file or a directory of the mounted FS.

fdisk & mkfs

- The main command to view and create partitions is `fdisk`.
- To display list of partitions:

```
# fdisk -l /dev/sdb
```

- Create partitions on /dev/sdb:

```
# fdisk /dev/sdb
```

- And then, follow the menu instructions.
- To make the changes available to the kernel without rebooting:

```
# partprobe /dev/sdb
```

- Create a filesystem with `mkfs`:

```
# mkfs.ext4 /dev/sdb2
```

More Issues

- Another interesting tool is `gparted` (graphical application).
- When installing most distributions include some tool to manage the partitions and filesystems that will be created in the system.
- If our partition is corrupted we can check and repair with `fsck`.
- To execute this command the partition to be checked/repaired must not be mounted.

Outline

Loop Disks I

- Loop disks allow to use a regular file as a disk.
- Interesting to understand how the file systems of virtual machines work.
- Process:

1. Create the file that will contain the filesystem filled with zeros (in our case a file of 100MB called “/var/disk“).

```
# dd if=/dev/zero of=/var/disk count=1 bs=100M
```

2. We will need a special loopback device so we look for one not currently used in the system.

```
# losetup /dev/loop0
```

   Replace /dev/loop0 with /dev/loop1, /dev/loop2, etc, until a free loopback device is found. Attach the loopback device (e.g. /dev/loop0) with the file:

```
# losetup /dev/loop0 /var/disk
```

Loop Disks II

3. Next, we create a filesystem (e.g. ext4) using the loopback device:

```
# mkfs.ext4 /dev/loop0
```

4. Create a mount directory and mount the filesystem:

```
# mkdir /mnt/myfs
# mount /dev/loop0 /mnt/myfs
```

5. Finally, if you want to unmount the loopback file system and release the loopback device, type:

```
# umount /mnt/myfs
# losetup -d /dev/loop0
```

- E.g. `df -h` to confirm its disk usage.
- Can create partitions on the /dev/loopX devices.
- Partitions in /dev/loop0 are called loop0p1, loop0p2, etc.

Outline

/etc/fstab I

- Currently, we have two drawbacks:
  1. Need some way of specifying the initial filesystem when booting.
  2. Only the root user can mount filesystems as this is a risky operation, so we also need a way of defining mount points for unprivileged users.
- The fstab file provides a solution to the previous issues.
- The fstab file lists all available disks and disk partitions.
- In current systems, fstab is still used for basic system configuration, notably of a system's main hard drive and startup file system.
- For mounts of unprivileged users, there are other ways.

/etc/fstab II

- An example of an entry in the /etc/fstab file is the following:

```
/dev/sda1    /    ext4    defaults    0 0
```

  - The 1st and 2nd columns are the device and default mount point.
  - The 3rd column is the filesystem type.
  - The 4th column are mount options and finally,
  - the 5th and 6th columns are options for the `dump` and `fsck` applications.
  - The 5th column is used by `dump` to decide if a filesystem should be backed up. If it's zero, dump will ignore that filesystem. This column is zero many times.
  - The 6th column is a `fsck` option. fsck looks at the number in the 6th column to determine in which order the filesystems should be checked. If it's zero, `fsck` won't check the filesystem.

Persistent block device naming

- A device name like /dev/sdb1 is based on where your physical drive is plugged in.
- Also depends on the order the drives were made available to the computer.
- If your computer changes the same command could mount a different partition.
- It's possible for this to happen just from a software upgrade.
- There are four different schemes for persistent naming: by-label, by-uuid, by-id and by-path.
- In particular, we discuss by-label and by-uuid (by-id and by-path are much less used in practice).

Labels

- Almost every filesystem type can have a label.
- Partitions that have a label are listed /dev/disk/by-label.
- The labels of your filesystems can be changed with different commands depending on the particular filesystem:

```
swap:       swaplabel -L <label> /dev/XXX using util-linux
ext2/3/4 :  e2label /dev/XXX <label> using e2fsprogs
btrfs:      btrfs filesystem label /dev/XXX <label> using btrfs-progs
```

- Example, to view the current label in an EXT filesystem:

```
# e2label /dev/sda1
```

- To set a new label, enter:

```
# e2label /dev/sdb2 data1
```

- Then, you can use the label in /etc/fstab:

```
#device          mount point     FS      Options Backup  fsck
LABEL=data1      /mnt/data1      ext3    _netdev 0       0
```

UUID I

- UUID give each filesystem a unique identifier.
- Automatically generated (e.g. by mkfs.*).
- Designed so that collisions are very unlikely.
- All GNU/Linux filesystems support UUID.
- FAT and NTFS filesystems do not support UUID, but are still listed in /dev/disk/by-uuid with a shorter UID (unique identifier).
- A UUID will remain the same if you put an internal disk into an external USB caddy, or change the name of the partition.
- You can type `ls -al /dev/disk/by-uuid/` to see UUID entries.
- The advantage of using the UUID method is that it is much less likely that name collisions occur than with labels.
- We can use the UUID in /etc/fstab:

```
#device            mount point      FS       Options Backup  fsck
UUID=9467f4de-4231-401f-bcaa-fee718d49e85 /   ext4     errors=remount-ro 0 0
```

UUID II

- Extremely rare to end with two exact UUIDs in a system.
- The exception is when you clone a partition (e.g. with `dd`).
- Need to change the UUID:

```
# sudo blkid
/dev/sda1: UUID="aabe7e48-2d11-421f-8609-7ea9d75e7f9b" TYPE="swap"
/dev/sda2: UUID="9467f4de-4231-401f-bcaa-fee718d49e85" TYPE="ext4"
/dev/sdb1: UUID="9467f4de-4231-401f-bcaa-fee718d49e85" TYPE="ext4"
```

- Here you can see that /dev/sda2 and /dev/sdb1 have the same UUID.
- Generate a new UUID:

```
$ uuidgen
f0acce91-a416-474c-8a8c-43f3ed3768f9
```

- Finally apply the new UUID to the partition:

```
$ sudo tune2fs /dev/sdb1 -U f0acce91-a416-474c-8a8c-43f3ed3768f9
```

Outline

User Mounts

- There are several ways of letting a user to mount a device.
- A way is to **configure /etc/fstab**.
- However, this is a static solution, bad for removable devices.
- Another is to **allow the mount command through `sudo`**.
- This is very insecure: e.g. the user could mount a filesystem with a suid root copy of bash and then, running that instantly gives root (likely without any logging, beyond the fact that mount was run).
- Desktops have solutions to allow users to mount removable media:
  - Mount in a subdirectory of /media only.
  - Turn off set-user/group-id support via kernel options.
- Several applications: `pmount` and `udisks`.

pmount I

- Most distributions do not install `pmount` out of the box.
- But easy to install from most package managers.
- /etc/pmount.allow lists all the possible devices that will be mountable using `pmount`.
- In addition, `pmount` will mount **removable devices**.
- To know if a device removable:

```
$ cat /sys/block/[device]/removable
```

  - If the value is “1“ then it is removable and the device does not need a white-list entry.
  - If it is ”0“ then it is not considered a removable device and must be white-listed to be used with `pmount`.

- By default all devices are mounted under the /media directory by their given partition name:

```
$ pmount /dev/sdb1
```

pmount II

- Will mount partition 1 of device sdb on /media/sdb1.
- You can also pass a label to pmount and the partition or device will be mounted under that label:

```
$ pmount /dev/sdb clipzip
```

- To find out what partitions are available for mounting when you plug a drive into your system you can issue the `dmesg`.
- If execute `pmount` without parameters, it will show all the devices mounted by the `pmount` command (if there are any).
- To umount:

```
$ pumount /dev/sdb
```

udisks

- When a disc (e.g. USB stick) is mounted under your file browser (e.g. nautilus) it uses `udisks` behind the scenes.
- You can also use the `udisks` command:

```
$ udisks --mount /dev/sdb1
```

- This will mount /dev/sdb1 in /media/<uuid>
- You can also use the UUID to perform the mount:

```
$ udisks --mount /dev/disk/by-uuid/1313-F422
```

- Partitions mounted by nautilus are now in /media/<user>/<uuid>
- To mount in this directory use the following command:

```
$ udisksctl mount --block-device /dev/disk/by-uuid/<uuid>
```

- To unmount, you can use the option `--unmount`.