Software Instalation

(& kernel compilation)

- **Is it really necessary to install?**
  - No it's not, you can use:
    - Live CD/DVD/USB ... see Slax
    - Virtual image
  - In this exercise we will simulate to install Linux on the hard drive (using virtualbox).

Distributions

  - Paid distributions (Why should I pay for Linux, It's free of charge, isn't it?)
    - With a payed distribution you pay mostly for tech. support and assistance
    - Usually paid distributions have unpaid versions (RHEL vs. CentOS or Fedora)
    - Typical Linux user (home/student/hobby) doesn't need a paid distribution
    - Linux is free for use.
      - You can choose between free distributions,
      - you can buy boxed version to support your favourite distro.
      - or you can even make you own distribution.
    - List of distributions

Installation

  - It's simple:
    - put boot-loader, kernel, and packages to your hard drive 😎
  - But how?
    - Graphic mode vs. Text mode
    - Large install medium (several DVDs) vs. Small medium (10MB CD image + network install)
    - Automatic vs. Manual vs. Combined installation process

Debian Small CD

  - Small CD image(10 – 200 MiB) – it's necessary to have rather good internet connection (Install process will download requested packages from distribution mirrors over the Internet)
  - Possible difficulties
    - limited Internet access (slow connection)
    - It's possible that small CD will not support your NIC hardware directly
      - (NIC firmware will have to be downloaded and made available to the kernel).

Let's begin

  - Download Debian „net-install CD“
  - Create new virtual machine in Virtualbox
    - All options leave in default

        - Choose dynamically allocated storage
        - mount downloaded CD image as virtual CD-ROM
    - Run new virtual machine
    - Now you have your virtual machine running (the behavior is similar to standard PC)
    - On first Debian boot screen choose „INSTALL“

- ## Installation process

    - Choose the language and keyboard layout
    - Choose the distribution mirror (suggested is CZ)
    - Set root password
    - Set new user and password
    - Hard drive partitioning – follow teachers instructions
    - During installation you can see installer downloading packages from internet
    - Choose the package „ssh server“ for install
    - Allow GRUB to be installed on virtual hard drive
    - Reboot
    - Test your new OS Linux.

- # Kernel compilation

    - Why? **Is it even necessary** for an ordinary user?
        - Answer: **NO**, it's not! You already have a well prepared kernel image in your distro.
    - **Is it possible?**: Can ordinary user compile his/her own kernel?
        - Answer: **Yes he can**. Of course, if there is a good reason. (kernel compilation is long and somewhat resources intensive job)

- ## Reasons for kernel compilation

    - Couriousity.
    - Smaller kernel
    - A faster start
    - Patching kernel – fixing errors or vulnerabilities
    - New features / settings of kernel
    - New drivers (It's not always necessary to recompile)

- ## How to obtain kernel sources?

    - Distribution's sources
        - Most probably the easiest way

        - Prepared scripts for kernel and module installation
        - Such kernel can be tailored for your distro
    - http://www.kernel.org - so called Vanilla kernel (Linus Torvald's kernels)

- **Kernel from kernel.org**

    - Ensure that you have enough free disk space. Building the kernel using the present procedure may require up to 15GB
    - Download and unpack to eg. /usr/src/linux_XXXversion
    - install all the necessary build tools
    - make symlink to new kernel sources...
    - configure and compile kernel
    - install the kernel

- ```
  $ ln s /usr/src/linux_XXXversionXXX /usr/src/linux
  $ cd /usr/src/linux (dir change)
  $ make mrproper # beginning
  #or
  $ make clean #clean compiled binaries
  $ make oldconfig #current kernel settings .config#
  $ make config/menuconfig/xconfig/gconfig #depending on what interface
  you prefer
  $ make #kernel compilation
  $ make modules_install
  $ mkdir /boot/2.6.13 #with respective version
  $ cp /usr/src/linux/System.map /boot/2.6.13
  $ cp /usr/src/linux/vmlinux /boot/2.6.13
  $ cp /usr/src/linux/arch/i386/boot/bzImage /boot/2.6.13
  $ cp /usr/src/linux/.config /boot/2.6.13 #backup the .config
  ```

- Edit `menu.lst` for grub (int's not necessary to reinstall)

  ```
  title Linux2.6.13
  root (hd0,0)
  kernel /boot/2.6.13/bzImage ro root=/dev/hda1
  ```

- **Distribution kernel package**

    - See your distribution documentation
        - for Debian
        - You can get Debian ova file if needed.

- **Installation and de-installation of software**

    - From source codes
    - Using the packaging system
  ```

- # From sources
  - How to obtain source codes?
    - Download in .tar.bz2, .tar.gz format (sourceforge.net)
    - From CVS, git, subversion, ...
  - Study&Follow README, INSTALL, UPDATE,... files
  - Typical usage:

```
$ ./configure
$ make
$ make install
```

  - requires to solve the dependencies manually
- ## Pros & cons
  - + No packaging system
  - + No need to wait for distributors of packages
  - - No packaging system
  - - For advanced users
  - - Garbage in the system
  - - Manual updates
  - - Recompilation after kernel change
  - - Dependencies → compilation or install of another packages
- # Software by package
  - Prepared SW for you distribution
    - Pre-compiled by distributor
  - Common package types:
    - `.deb` - GNU/Debian
    - `.rpm` - Red Hat Linux
    - `.ebuild` - Gentoo Linux
- # .deb package format 1/2
  - Created by „packaging“ process
  - compilation and description → using `ar` to create archive → renaming to `.deb`
    - How to extract a .deb file without opening it on Debian/Ubuntu Linux
  - **Contains**:
    - Package description
      - Name
      - Package version info
      - Dependency requirements
      - ...
    - configuration files
    - and ofc **the binary files**
- ```
  user@machine:~$ apt-cache show coreutils
  Package: coreutils
  Architecture: amd64
  ```
  ```

```
Version: 8.30-3ubuntu2
Multi-Arch: foreign
Priority: required
Essential: yes
Section: utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Michael Stone <mstone@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 7196
Pre-Depends: libacl1 (>= 2.2.23), libattr1 (>= 1:2.4.44), libc6 (>=
2.28), libselinux1 (>= 2.1.13)
Filename: pool/main/c/coreutils/coreutils_8.30-3ubuntu2_amd64.deb
Size: 1249368
MD5sum: e8e201b6d1b7f39776da07f6713e1675
SHA1: 1d4ab60c729a361d46a90d92defaca518b2918d2
SHA256:
99aa50af84de1737735f2f51e570d60f5842aa1d4a3129527906e7ffda368853
Homepage: http://gnu.org/software/coreutils
Description-en: GNU core utilities ...
```

- **Package distribution**
  - From the Internet
  - By manual download
    - Software home-page
    - packages.debian.org
    - just somewhere on the Internet - not recommended
  - By package manager
    - From the repositories – maintained by distributor of OS - Recommended way!
- **Installation by package managers**
  - positive
    - No compilation from sources
    - Easy and fast
    - Automatic updates
    - Automatic dependency resolution
  - negative
    - Internet access needed
    - Implementation of package manager
    - Installation into predefined place
    - Only one version of SW
- **Tools for package administration**
  - .deb
    - dpkg
    - apt

    - apt-get
    - aptitude
  - snap for snaps

dpkg

- **dpkg** - is a package manager for Debian-based systems. It can **install, remove, and build packages**, but unlike other package management systems, it **cannot automatically download and install packages or their dependencies**. Apt and Aptitude are newer, and layer additional features on top of dpkg.
- dpkg is useful to manage locally installed packages:
  - To list all packages in the system's package database `dpkg -l`
  - Pipe the output through grep to see if a specific package `dpkg -l | grep apache2`
  - To list the files installed by a package `dpkg -L ufw`
  - If you are not sure which package installed a file

```
dpkg -S /etc/host.conf
base-files: /etc/host.conf
```

Warning: Many files are automatically generated during the package install process, and even though they are on the filesystem, dpkg -S may not know which package they belong to.

```
$ dpkg -i <filename> # install downloaded file
$ dpkg -l # list installed packages
$ dpkg -L # list content of package
$ dpkg -r <package> # remove
$ dpkg -P <package> # remove with config files
$ dpkg -s <package> # show status of package
$ dpkg -S <word> # search for package containing <word>
                 # in name or path of files in package
```

Manual installation with dpkg (example)

- List of available packages
  - Need to know your computer architecture (`uname -i`)
- Package download :

```
 $ wget
http://ftp.cz.debian.org/debian/pool/main/h/htop/htop_3.0.5-7_amd6
4.deb
```

- Installation :

```
$ sudo dpkg -i htop_3.0.5-7_amd64.deb
```

- Remove the package:

```
$ sudo dpkg -r htop
```

- ## apt

  The apt command is a powerful command-line tool, which works with Ubuntu's Advanced Packaging Tool (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire system.

  - Download package (just download it to the actual dir)

    ```
    apt download nmap
    ```

  - Install a Package

    ```
    sudo apt install nmap
    ```

    Package does not have to be downloaded manually, it will be done automatically by APT. It is a good idea to update sources lists (`apt update`) prior any install/upgrades.

  - Remove a Package

    ```
    sudo apt remove nmap
    ```

- **Note**: You may specify multiple packages to be installed or removed, separated by spaces.

  Scripting: While apt is a command-line tool, **it is intended to be used interactively**, and not to be called from non-interactive scripts. The `apt-get` command should be used in scripts (perhaps with the –`quiet` flag). For basic commands the syntax of the two tools is identical.

- ## Update the system

  - Update the Package Index (`/etc/apt/sources.list` and or `/etc/apt/sources.list.d` directory):

    ```
    $ sudo apt update
    ```

  - Upgrade Packages :

    ```
    $ sudo apt upgrade
    ```

  - Dont forget to study `apt help`

- ```
  user@server:~$ apt moo
                       (__)
                       (oo)
                 /------\/
                / |    ||
              *  /\---/\
                 ~~   ~~
  ..."Whoever needs milk, bows to the animal."...
  user@server:~$ apt-get moo
                       (__)
                       (oo)
  ```
  ```

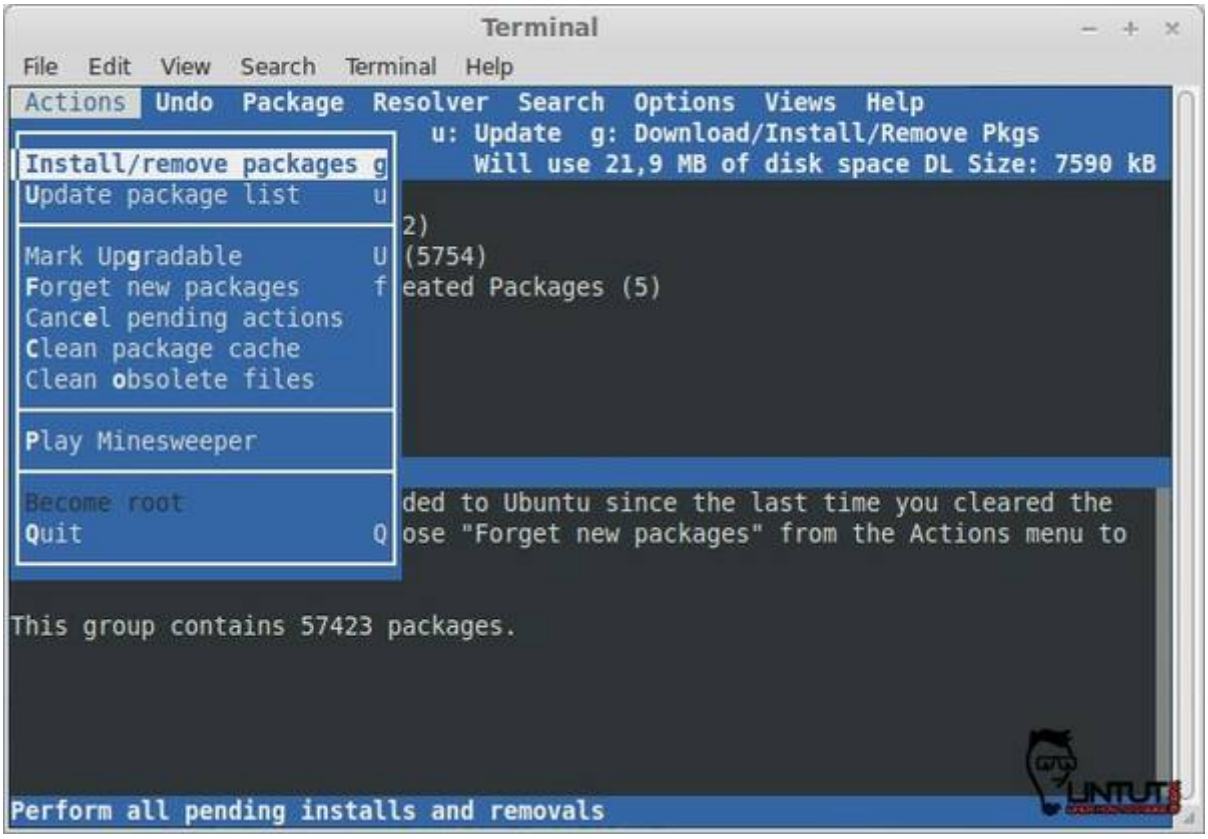
```
                /------\/
               / |    ||
              *  /\---/\
                 ~~   ~~
..."Whoever needs milk, bows to the animal."...
```

- ```
  user@server:~$ aptitude moo
  There are no Easter Eggs in this program.
  user@server:~$ aptitude  -v moo
  There really are no Easter Eggs in this program.
  user@server:~$ aptitude  -vv moo
  Didnt I already tell you that there are no Easter Eggs in this program?
  user@server:~$ aptitude  -vvv moo
  Stop it!
  user@server:~$ aptitude  -vvvv moo
  Okay, okay, if I give you an Easter Egg, will you go away?
  user@server:~$ aptitude  -vvvvv moo
  All right, you win.

                                /----\
                       -------/      \
                      /               \
                     /                 |
     -----------------/                  --------\
     ---------------------------------------------
  user@server:~$ aptitude  -vvvvvv moo
  What is it?  It's an elephant being eaten by a snake, of course.
  ```

aptitude

- provides a terminal menu interface
- Go see sometutorial
  ```



- # snap

  - Snap is a software packaging and deployment system **developed by Canonical** for operating systems that use the Linux kernel and the **systemd** init system.
  - The packages, **called *snaps***, and the tool for using them, **snapd**, work across a range of Linux distributions and allow upstream software developers to distribute their applications directly to users.
  - Snaps are **self-contained applications running in a sandbox** with **mediated access to the host system**.
  - Snap was originally released for cloud applications but was later ported to also work for Internet of Things devices and desktop applications.
  - **Note**: not to be confused with package(not a program) `snap` see: `apt info snap`
    - or `apt-file list snap` vs. `apt-file list snapd`
  - Ubuntu manual page for snap.

- # Snap usage

Snap comes preinstalled on most of the latest Linux distributions. To check if you have Snap preinstalled on your system...

```
user@machine:~$ snap
The snap command lets you install, configure, refresh and remove snaps.
Snaps are packages that work across many different Linux distributions,
enabling secure delivery and operation of the latest apps and
utilities.

Usage: snap <command> [<options>...]
```

```
Commonly used commands can be classified as follows:

         Basics: find, info, install, remove, list
        ...more: refresh, revert, switch, disable, enable, create-
cohort
        History: changes, tasks, abort, watch
        Daemons: services, start, stop, restart, logs
    Permissions: connections, interface, connect, disconnect
  Configuration: get, set, unset, wait
    App Aliases: alias, aliases, unalias, prefer
        Account: login, logout, whoami
      Snapshots: saved, save, check-snapshot, restore, forget
         Device: model, reboot, recovery
      ... Other: warnings, okay, known, ack, version
    Development: download, pack, run, try

For more information about a command, run 'snap help <command>'.
For a short summary of all commands, run 'snap help --all'.
```

- **Install snap**

  In case snap is not installed:

```
$ sudo apt update
...
$ sudo apt install snapd
...
$ sudo snap install snap-store #for the graphical interface
```

- **Use snap**
  - Install Snap Apps

```
$ sudo snap install <package_name>
```

  - List Installed Snaps

```
$ snap list
```

```
$ snap list --all <package_name>
```

  - Search for Snaps

```
$ snap find <search_term>
```

  - Update Snaps

```
$ sudo snap refresh <package_name>
```

    - see which snap packages have available updates

```
$ sudo snap refresh --list
```

- **Use snap cont.**
  - Downgrade Snaps

    ```
    $ sudo snap revert <package_name>
    ```

  - Remove Snaps

    ```
    $ sudo snap remove <package_name>
    ```

    - Once an app is removed, a snapshot of the snap's system, configuration and user data remains for 31 days.
    - remove a snap without generating a snapshot

      ```
      $ sudo snap remove <package_name> --purge
      ```

  - Enable and Disable Snaps

    - ```
      $ sudo snap disable/enable <package_name>
      ```

- **snap Services**

  The majority of snaps expose their functionality via applications that run on the local system. Most of these applications are started either from the command line, the graphical desktop, or as services that run automatically.

  A single snap may provide multiple applications and services. With a database snap, for example, you might expect an interactive client application alongside the service daemon.

  Snaps manage their own services without the need for manual intervention. However, experienced administrators may want to interact with a snap's services to help improve their integration with the local environment. For that reason, snapd offers a set of commands to allow a snap's services to be inspected and their statuses changed.

  - lists all the services added to the system by the currently installed and enabled snaps

```
$ snap services
Service                                 Startup   Current   Notes
bluez.bluez                             enabled   inactive  -
bluez.obex                              enabled   inactive  -
cups.cups-browsed                       enabled   active    -
cups.cupsd                              enabled   active    -
docker.dockerd                          enabled   active    -
juju.fetch-oci                          disabled  inactive  -
lxd.activate                            enabled   inactive  -
lxd.daemon                              enabled   inactive  socket-
activated
```

  - list only services added by that specific snap
```

```
snap services <package_nam>
```

- Restarting services

```
snap restart <snap_name>
```

- Starting and stopping services

```
$ sudo snap stop lxd.daemon
```

  - Stopping snap services manually may cause the snap to malfunction. For temporarily disabling a snap consider using the enable and disable commands instead.

- Disable service

```
$ sudo snap stop --disable lxd.daemon
```

- **Snap vs Apt**

	Snap	**APT**
Package type	.snap	.deb
Tool name	snapd	APT
CLI tool	snap	apt
Format	SquashFS archive	ar archive
Available in	Snap Store	Debian repositories
Installation Size	Larger	Smaller
Dependencies	Contained in the package	Shared
Updates	Automatic	Manual
Security confinement	Confined	Limited confinement
Multiple installations	Possible	Not possible
Multiple version installations	Possible	Not possible

More (and hopefully everything) about snap right from the source.

Flatpak

Flatpak, formerly known as **xdg-app**, is an utility for software deployment and package management for Linux. It is advertised as offering a sandbox environment in which users can run application software in isolation from the rest of the system.

Snap vs Flatpak vs Appimage... comparison (online, 8/2022)

Install flatpak

(Ubuntu-ish version)

- Install Flatpak

        - `$ sudo apt install flatpak`
    - Install the Software Flatpak plugin
        - `sudo apt install gnome-software-plugin-flatpak`
    - Add the Flathub repository
        - `flatpak remote-add –if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo`

- **Install the app using Flatpak**
    - ...just go click in the GUI
    - or:
        - search for the app: `flatpak search applicationname`
        - install the app: `flatpak install <remotes> <ApplicationID>`
            - like: `flatpak install flathub org.libreoffice.LibreOffice`
            - or use direct flatpakref net path:

```
flatpak install --from
https://flathub.org/repo/appstream/com.spotify.Client.fla
tpakref
```

            - or if you have the file downloaded:

```
flatpak install
/home/user/Downloads/net.poedit.Poedit.flatpakref
```

        - Run a Flatpak app
            - `flatpak run com.spotify.Client`
        - Display all Flatpak apps installed
            - `flatpak list`
        - Uninstall a Flatpak application
            - `flatpak uninstall com.spotify.Client`
        - Updating all Flatpak applications
            - `flatpak update`

Exercise

- install `iftop` program by using package system
    - un-install `ifotp` using package system
- Get sources of `iftop`
- Compile it (no root permissions needed, except when satisfying dependencies by installing libraries)
- Install `iftop` from sources
    - try both:
        - using root permissions
        - and not using root permissions as well
- Test the installed program
- Uninstall it.
- use snap to install `superuxkart` game
- browse the available software in snap-store
    - explore the parallel install feature with snap
    - try to install multiple parallel snap application versions at the same time