How to upgrade to a newer release?

General considerations

1. . Do you need to upgrade?

If things are working fine and you're happy with your current system, then you don't **need** to upgrade.

- A **new version of Linux Mint is released every 6 months.**
- you could skip many releases and stick with the version that works for you.
- Each release receives bug fixes and security updates for about 18 months (or 3 years in the case of "Long Term Support" releases).
- The development team is also focused on the latest release. If bug fixes and security updates are important to you, you should regularly upgrade to the latest releases, otherwise there's nothing wrong with keeping things as they are.

As a general rule... unless you **need** to, or unless you really **want** to, there's no reason for you to upgrade.

2. Important aspects

The most important things when it comes to upgrading are:

- That your data is safe
- That you end up with a fully functional system

Your personal data is the most valuable thing in your computer. The first thing to do is to make a backup of your data.

- You'll also want to make sure that the release you're upgrading to is right for you.
- Every release comes with a different kernel. This means that it handles hardware differently. For instance, you may find out that a graphic card or a wireless adapter which currently works fine for you under

Linux Mint, isn't recognized by the newer version of Linux Mint you're planning to upgrade to. In some cases, this could mean that upgrading to this release is the wrong decision; maybe you're better off skipping that particular release? There's only one way to know: you need to try it.

- Linux Mint comes as a liveDVD. you can try the newer release on your computer and see if your hardware is recognized **before** proceeding to the upgrade.

3. To be safe:
    - Make a full backup of your data on an external device (USB stick or DVD)
    - Download and burn the liveDVD of the newer release, and check that your hardware is fully functional with it.

4. The different ways of upgrading

There are many different ways of upgrading to a newer release but we can categorize them in two different families: "fresh" and "package" upgrades.

- **"Fresh" upgrades**

In a "fresh" upgrade you use the liveDVD of the new release to perform a new installation and to overwrite your existing partitions.

A "Fresh" upgrade consists of the following steps:

- Making a backup of the data
- Making a backup of the software selection
- Performing a fresh installation using the liveDVD of the new release
- Restoring the data
- Restoring the software selection

This is the recommended way to upgrade Linux Mint and it offers the following advantages:

Safe: Your data is backed up externally. Whatever mistake you make or whatever bug happens during the installation cannot affect it.

Fast: The installation usually lasts 15 minutes. The DVD for the new release contains compressed data. Downloading the ISO and upgrading from the DVD is much faster than upgrading the system from the repositories.

Reliable: First, you get the opportunity to test your hardware detection in the new release using the liveDVD. If anything is wrong you can simply decide not to upgrade, it's not too late. Second, you end up with a fresh installation of Linux Mint, i.e. a system that was fully tested by the development team and the community.

Easy: Things do go as planned this way.

- **"Package" upgrades**

A "package" upgrade consists of the following steps:

Pointing APT to the repositories of the newer release: The Advanced Packaging Tool, or APT, is a free user interface that works with core libraries to handle the installation and removal of software on the Debian GNU/Linux distribution and its variants. APT simplifies the process of managing software on Unix-like computer systems by automating the retrieval, configuration and installation of software packages.

- Asking APT to perform a full upgrade
- APT is the package management system used by Linux Mint. Alternatively, some releases were given a graphical upgrade tool to perform these steps.
- This way of upgrading Linux Mint should only be recommended to advanced users.

Here are the pros and cons of upgrading the system this way:

disadvantages:

Slow: APT will download the new version of all the packages installed on your system. Using a fresh upgrade you could have downloaded all that data by simply getting the ISO.

Unreliable: Depending on your modifications, your sources, your added software and your configuration you could end up with a system that acts and feels really different than a brand new version of the newer Linux Mint release. You're far from the beaten track and the added features might not work as well on your system as they were designed to.

Risky: The temptation when you upgrade with APT is not to perform backups... since your partitions aren't overwritten, nothing "forces" you to make backups... think about the risk though.

Complicated: Packages conflict with each others, they can bring complex dependencies and put you in situations that are difficult to solve.

advantages:

Automated: APT does everything for you (well, until something goes wrong of course)

Real upgrade: A "fresh" upgrade is kind of like the new Linux Mint with your data on it... this in comparison feels more like "your system" running the newer version underneath.

5. How to upgrade

In this tutorial we'll use a "fresh" upgrade.

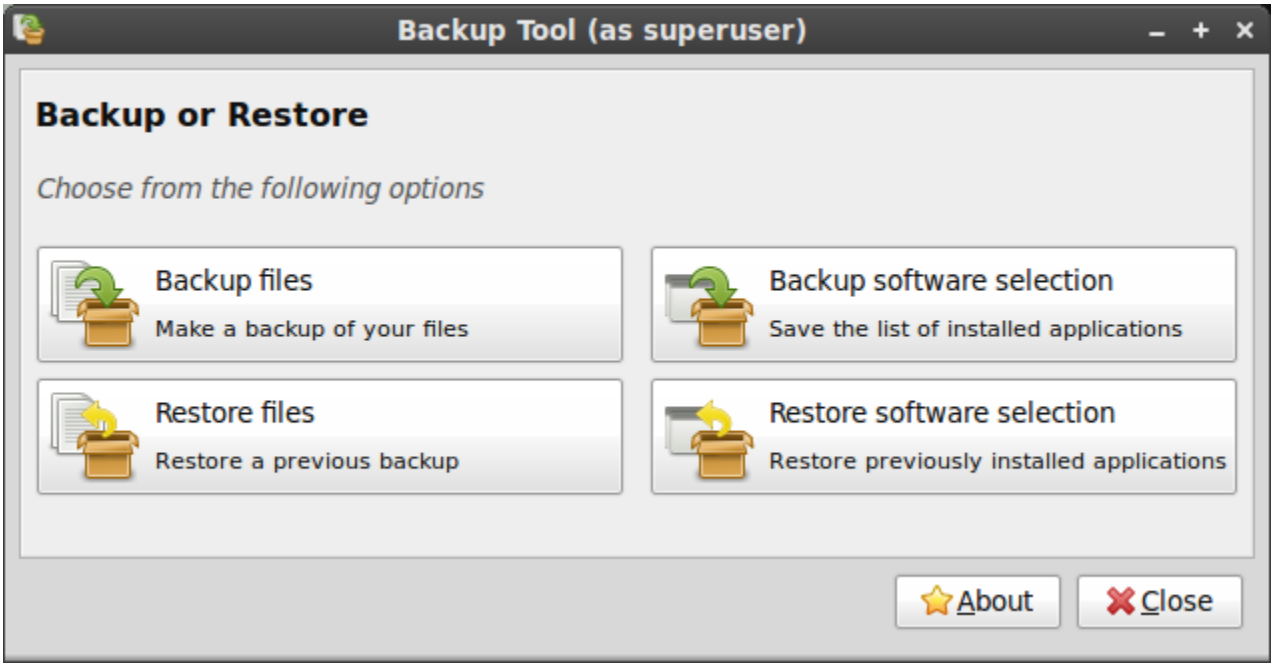
- **Make a backup of your existing system**

You need to save your personal data and your software selection: Your personal data, simply because you don't want to lose it... and your software

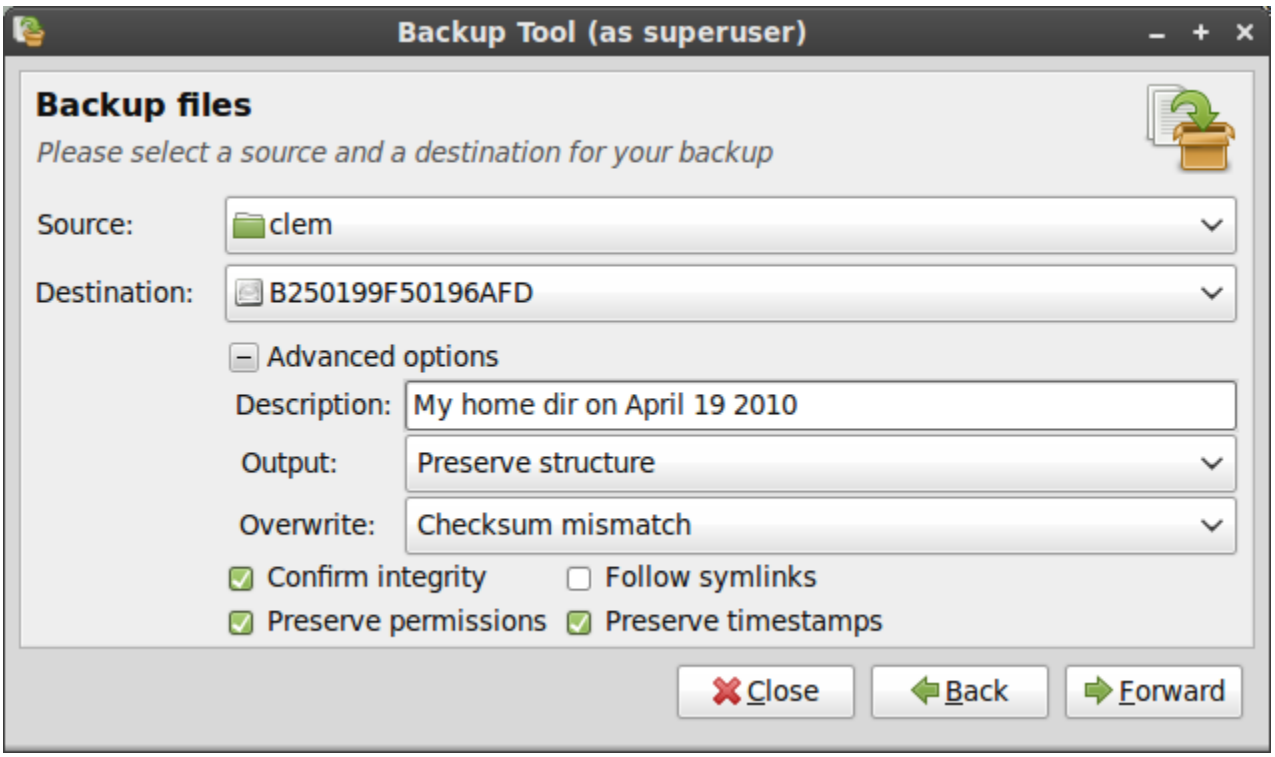
selection, because post-installation you don't want to have to reinstall all the applications you've added to your system.

1 Backup your data

Open mintBackup from the Menu -> Administration -> Backup Tool



Click on "Backup files".



- Select your home directory for the source.
- Select where you want to backup the files as your destination (ideally some external volume since the plan is to format this partition during the installation)

In the advanced options choose the following:

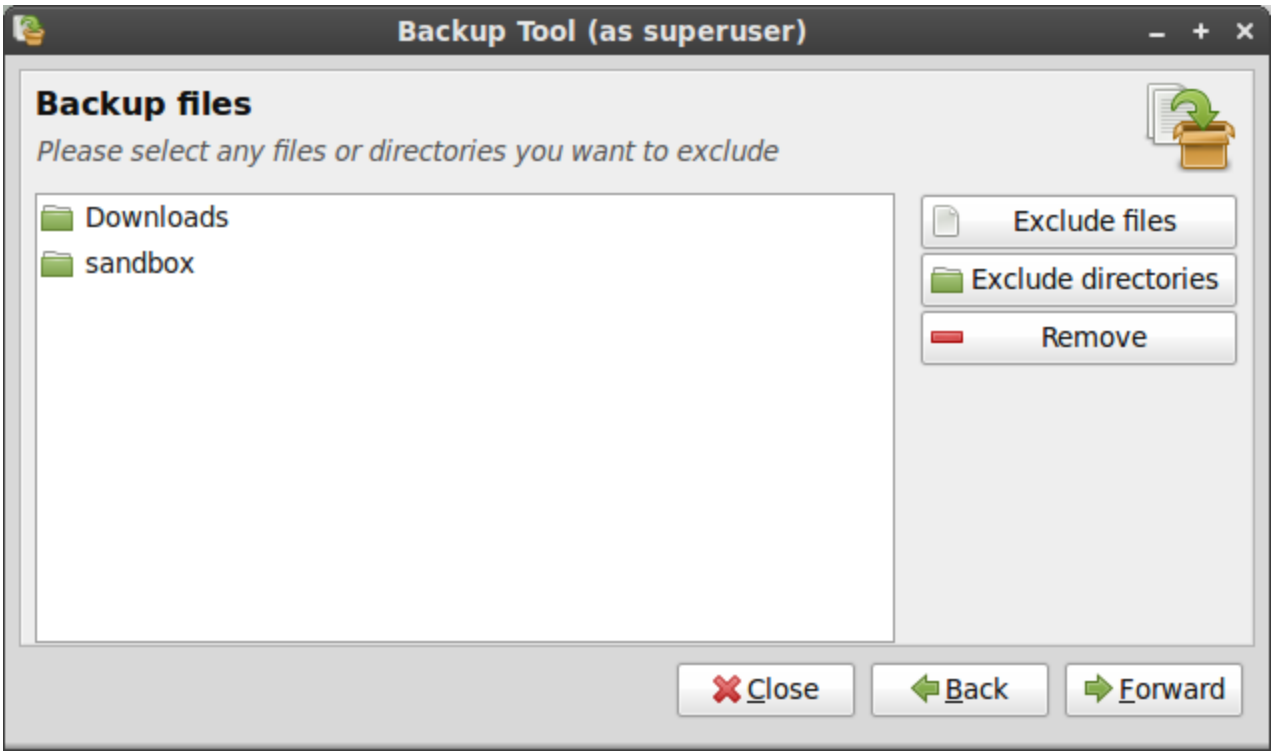
- A description (that's always a good idea to add to a backup)

The output format (whether you want to save the backup as a directory, or an archive... archives take longer but if you plan to backup to NTFS/FAT volumes, archives preserve permissions that cannot be preserved by directories on these Windows filesystems)

- Overwrite (you can ignore this setting, it only comes into play if you plan to update an already made backup)
- Confirm integrity: slower but more reliable, it checks the signature of each file after it was backed up.
- Preserve permissions and timestamps: This should be selected.

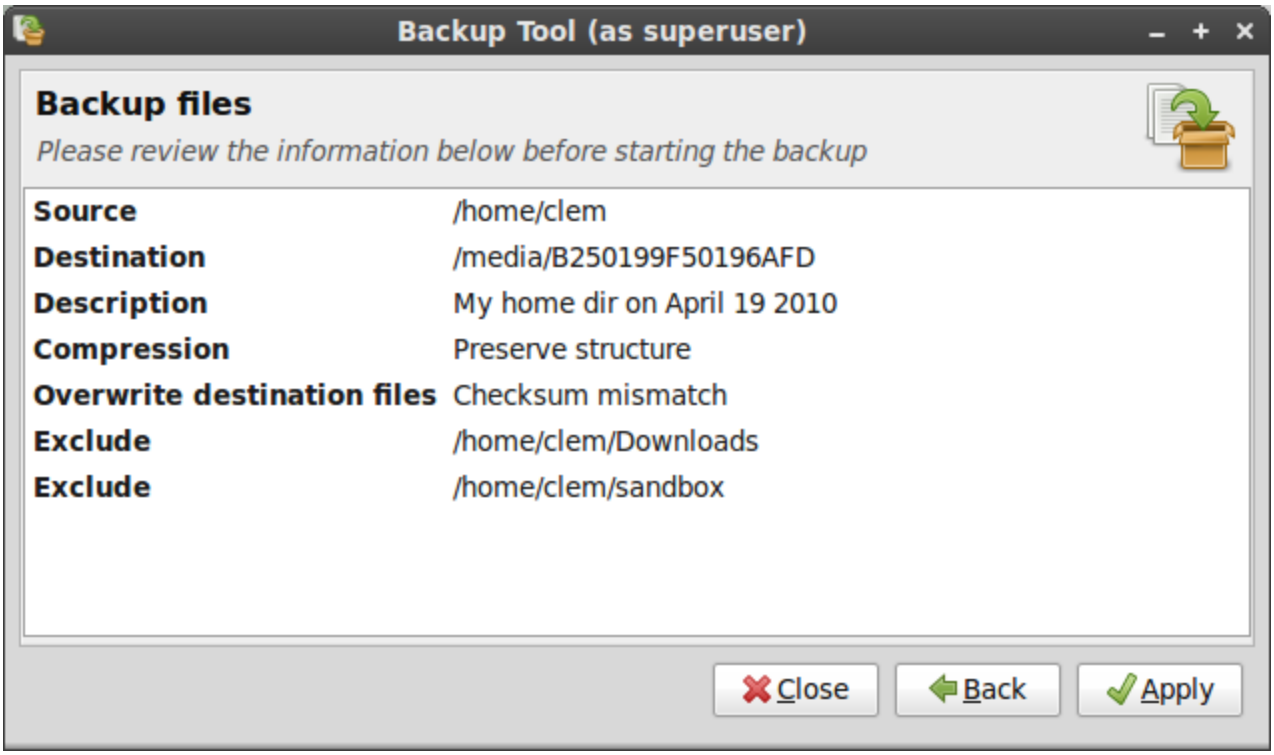
Follow symlinks: Not needed.

Click "Forward"



To save space and time, exclude things you don't need. For instance, I don't need to backup my "sandbox" and "Downloads" directories, but that's just me.

When you're ready, click "Forward".

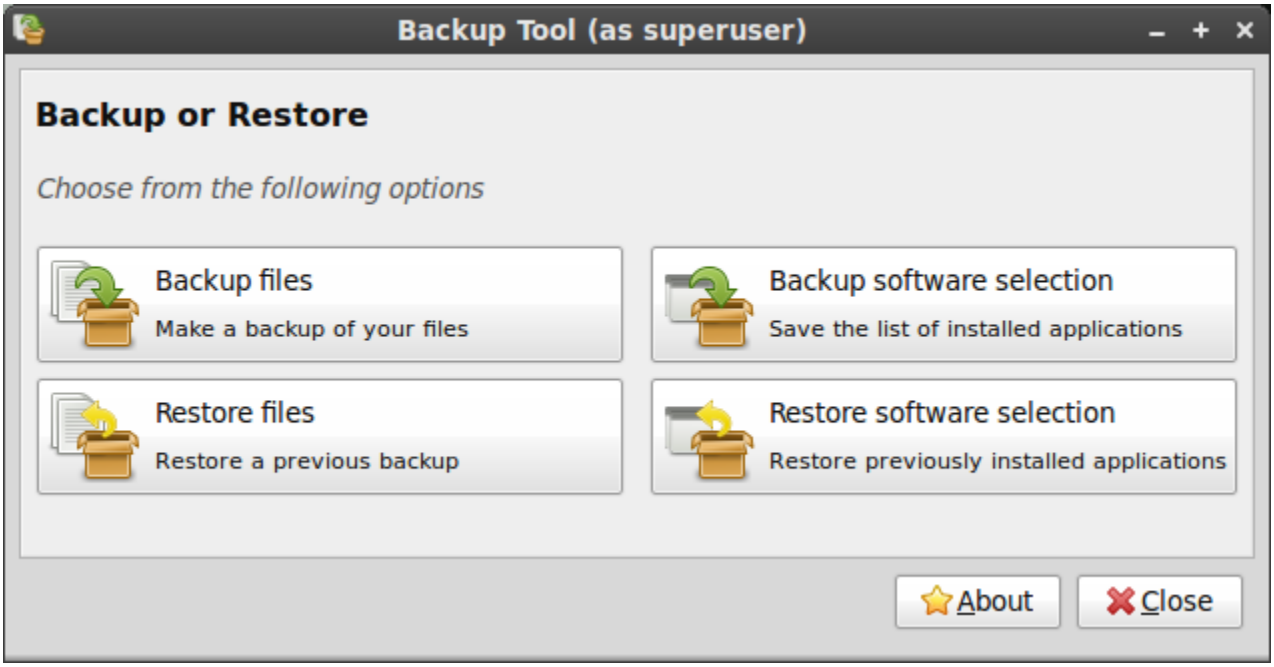


Review the information about your backup and press "Apply".

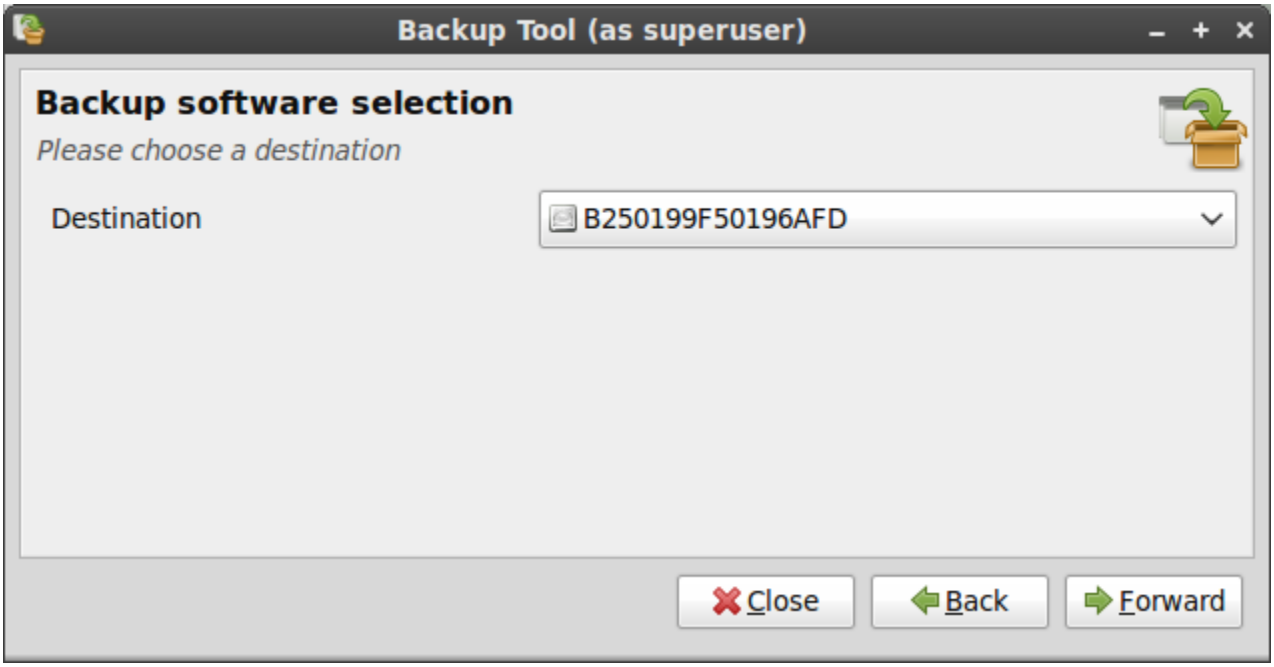
Note: When the backup is finished. Make absolutely sure to check the result yourself. No matter how much trust you put in the backup tool, head towards the destination, and check that all your files and directories were backed up successfully. If you chose to backup as an archive, make sure to open it up to see if it's working and whether or not it contains everything that it should. The tool is stable, but there can always be a bug or you could have made a mistake... since we're talking about your data here, make **absolutely** sure before anything else.

2 Backup your software selection

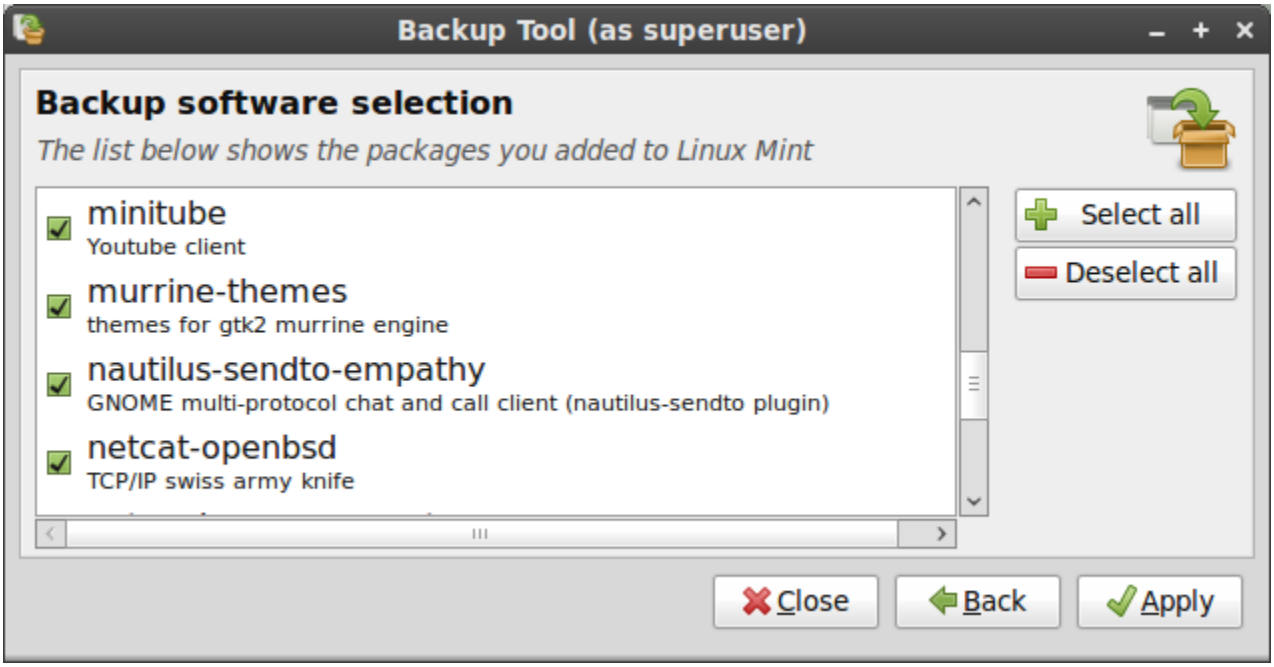
Open mintBackup from the Menu -> Administration -> Backup Tool



Click on "Backup software selection".



Choose a destination (ideally some external volume since the plan is to format this partition during the installation).



The following screen shows you all the packages you added to your Linux Mint system. Select the ones you want to exclude from your backup and click "Apply".

Note: When the backup is finished (it shouldn't take long), go to the destination and check for a file which name starts with "software_selection". This is the backup of your software selection.

6. Test and install the newer version of Linux Mint

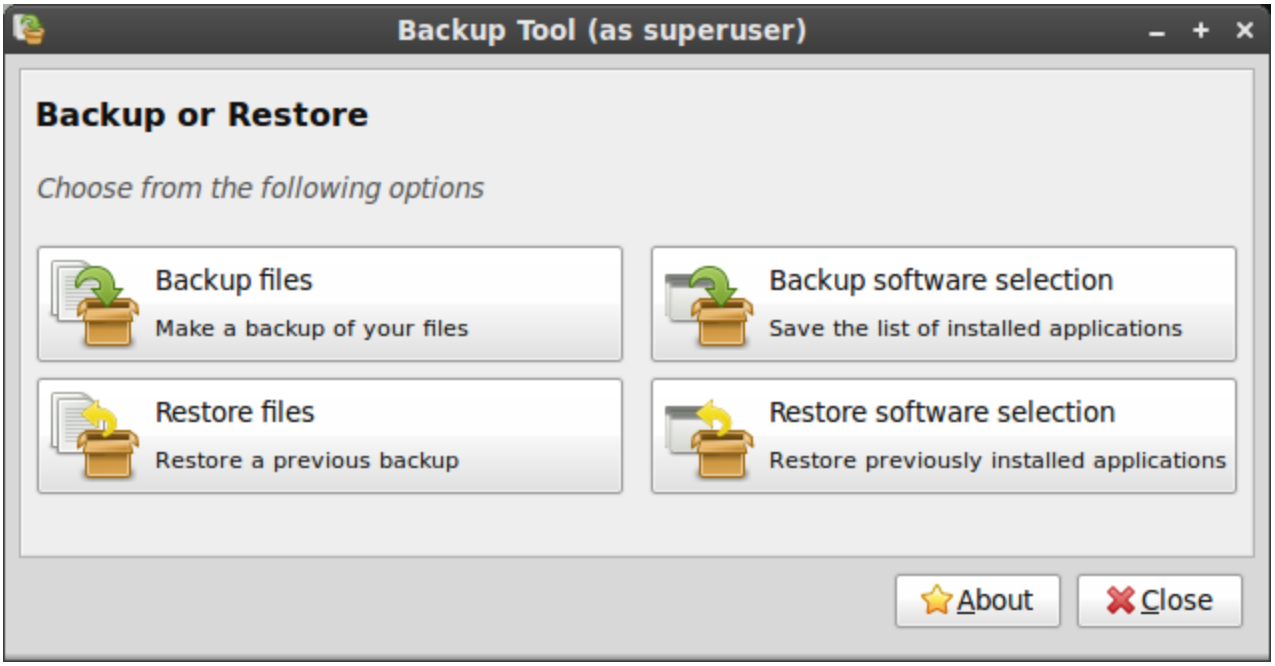
- Download the ISO for the newer version of Linux Mint.
- Check its MD5 signature.
- Burn it at low speed on a liveDVD.
- Boot from the liveDVD and select the option "Check disk integrity".
- Boot from the liveDVD and select "Start Linux Mint".
- Once on the live desktop, check that your hardware is properly recognized (graphics card, wireless..etc)
- Once you're happy and confident that this newer release is good for you, click on "Install" on the desktop and proceed with a normal installation.

When asked by the installer, choose "Specify partitions manually (advanced)", select the partition that you used for your current installation of Linux Mint, assign "/" to it, and reformat it to "ext4".

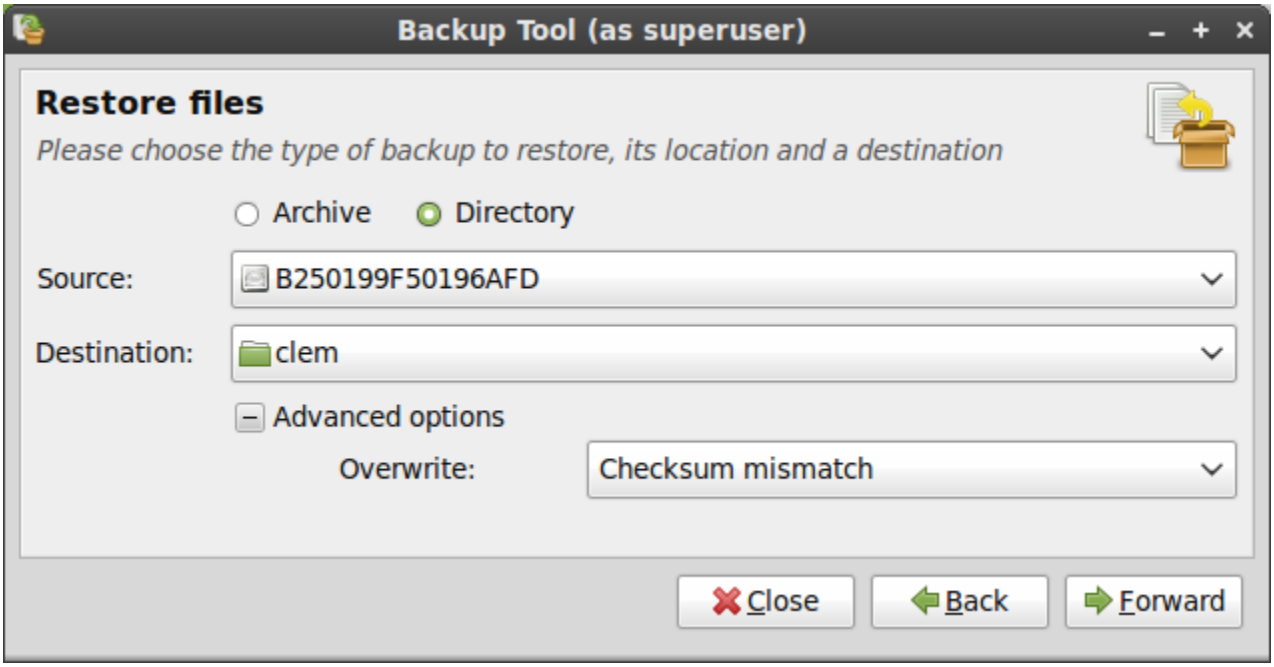
7. Restore your data and your software selection

1Restore your data

Open mintBackup from the Menu -> Administration -> Backup Tool



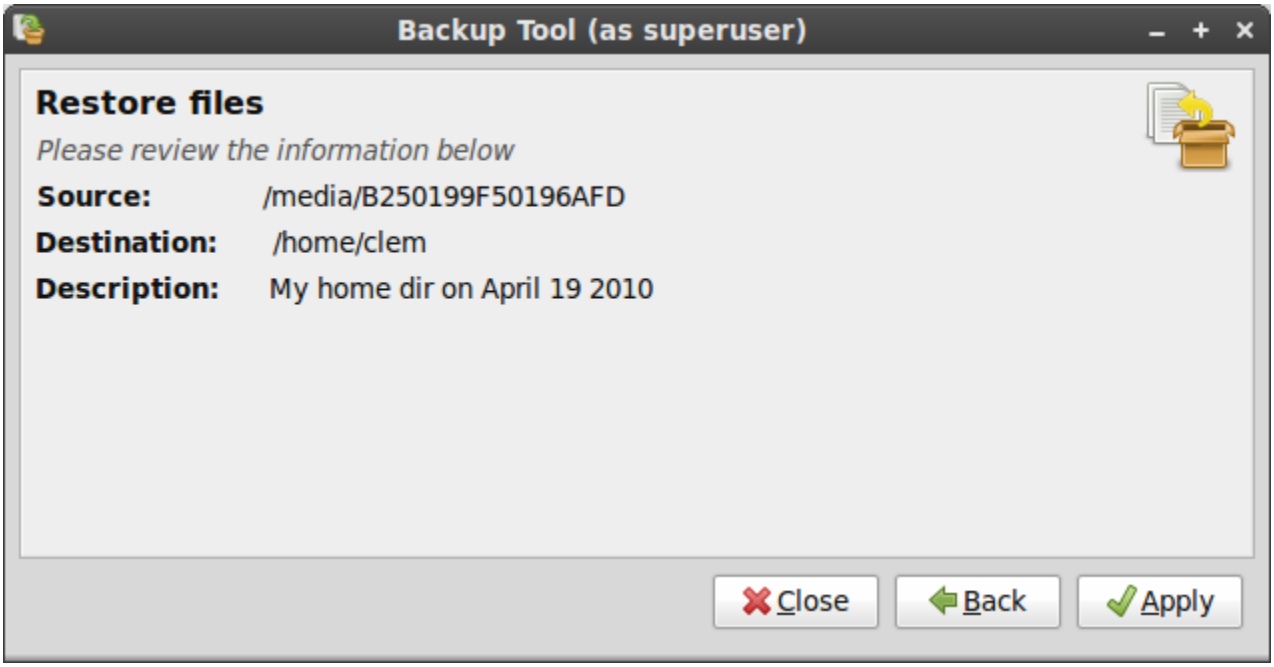
Click on "Restore files".



If you backed up your data as an archive, choose "Archive", otherwise choose "Directory", then point the source to either your archive or the location where you made the backup of your data.

Set your new home folder as the destination.

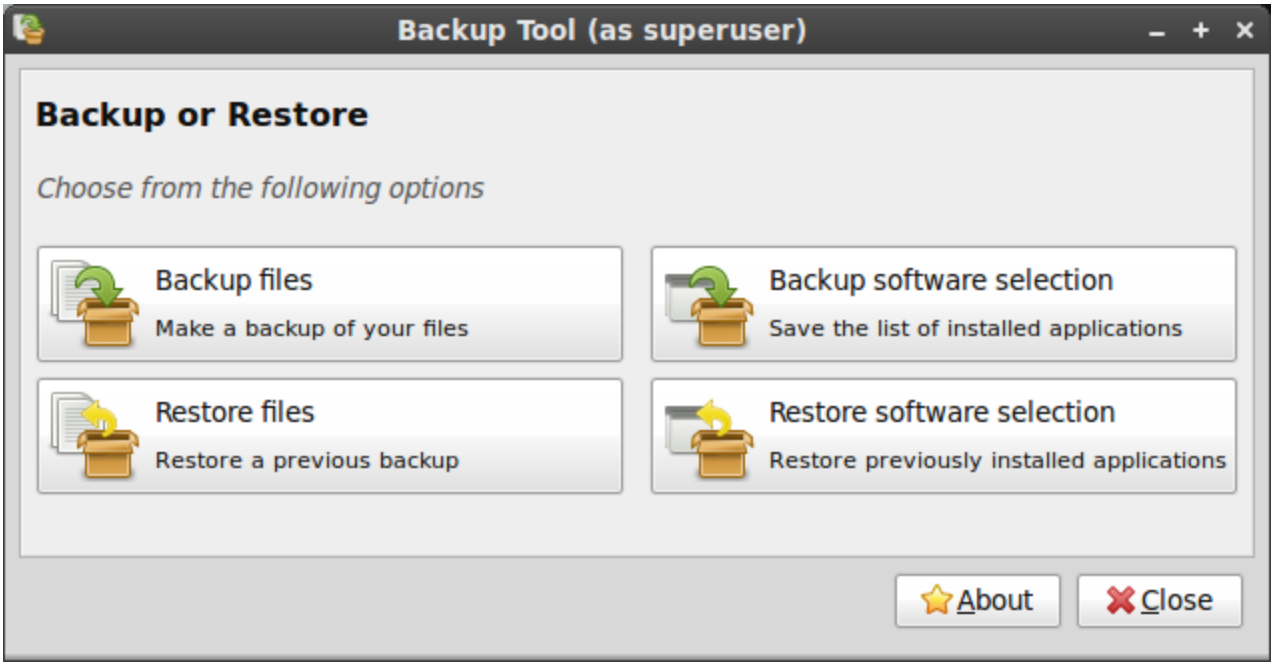
You can ignore the advanced options since your home folder is empty at the moment. These settings are only there to optimize things for people running regular backups.



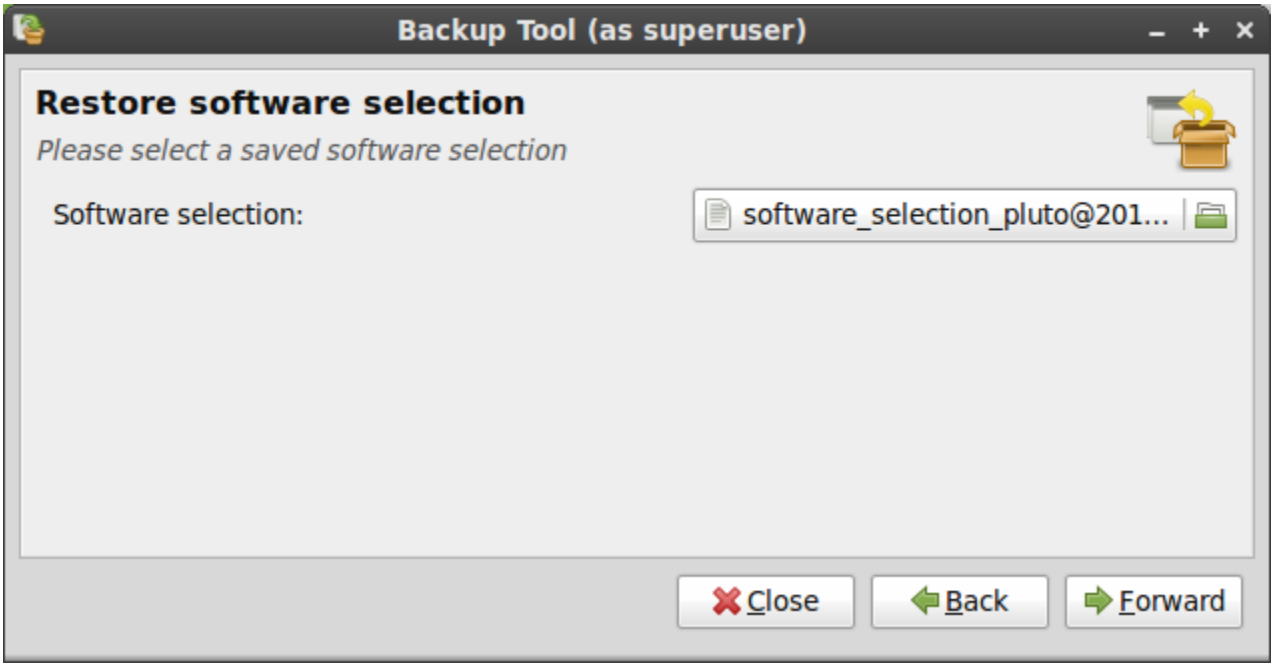
Review the information and press "Apply" when ready.

.2 Restore your software selection

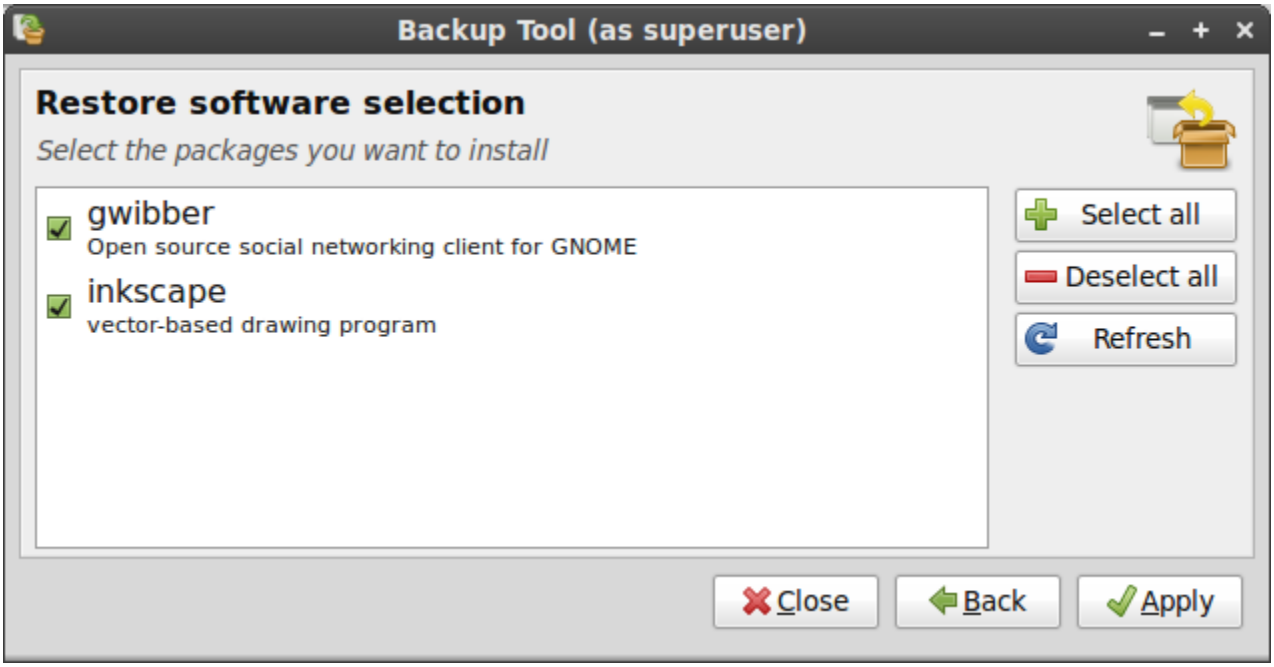
Open mintBackup from the Menu -> Administration -> Backup Tool



Click on "Restore software selection".



Select your backed up software selection file and press "Forward".



In the next screen you can see a list of packages. Select the ones you want to install and press "Apply".

This list only contains packages which were part of your previous software selection and which are not installed in the present system. The packages

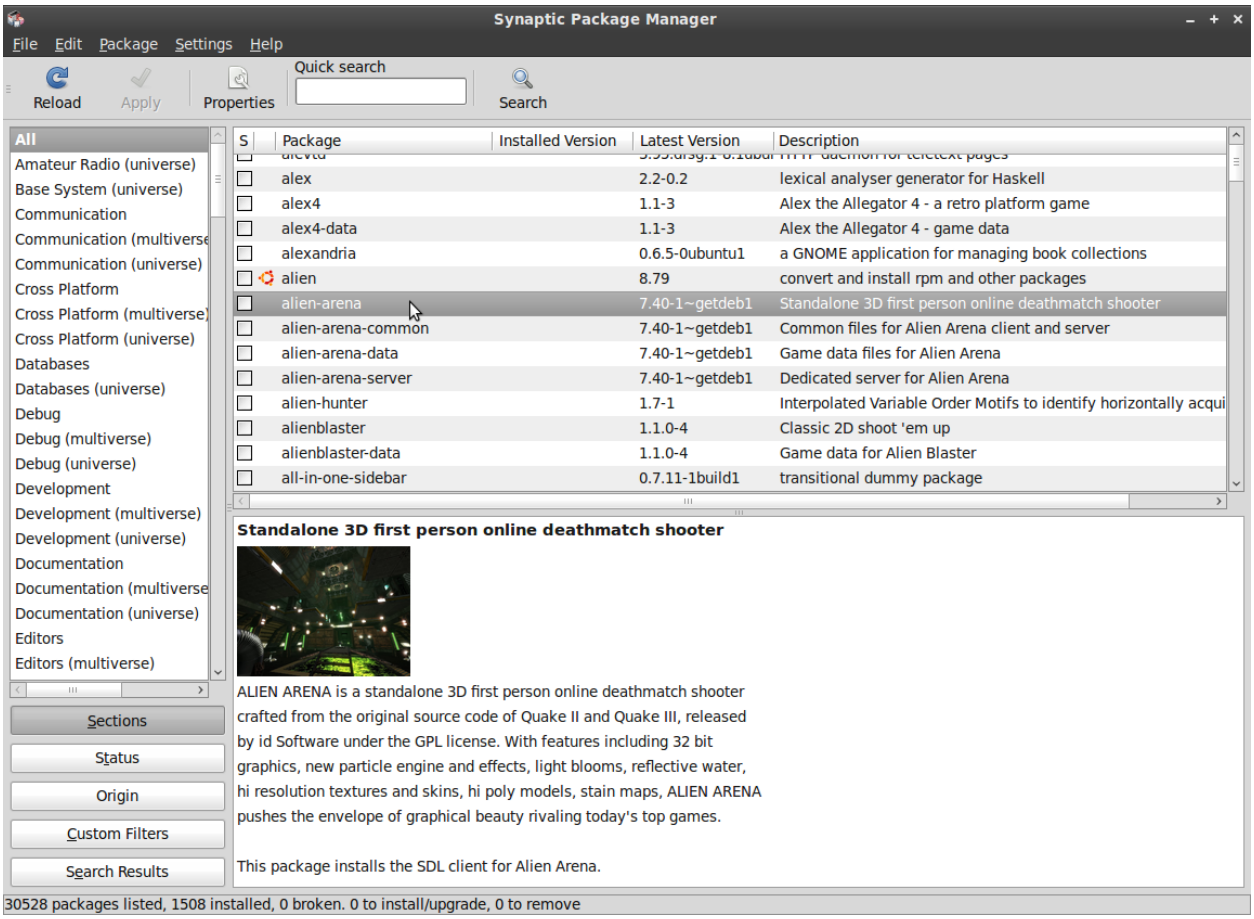
which are already installed in your new system don't need to be installed again, so they don't appear in this list.

At any moment, you can press the "Close" button. You don't have to install everything. If you change your mind later on, just launch the Backup tool again and restore your software selection using the same backup, the list will appear and show you the packages you're missing.

Synaptic Graphical package manager

Synaptic is a graphical package management program for apt. It provides the same features as the apt-get command line utility with a GUI front-end based on

- Install, remove, upgrade and downgrade single and multiple packages.
- Upgrade your whole system.
- Manage package repositories (sources.list).
- Find packages by name, description and several other attributes.
- Select packages by status, section, name or a custom filter.
- Sort packages by name, status, size or version.
- Browse all available online documentation related to a package.
- Download the latest changelog of a package.
- Lock packages to the current version.
- Force the installation of a specifc package version.
- Undo/Redo of selections.
- Built-in terminal emulator for the package manager.



besides these basic functions the following features are provided:

* search and filter the list of available packages
* perform smart system upgrades
* fix broken package dependencies
* edit the list of used repositories (sources.list)
* download the latest changelog of a package
* configure packages through the debconf system
* browse all available documentation related to a package (dwww is required)

How to use terminal

It has many names: terminal, shell, console, "command prompt" even as a carryover from those familiar with Windows.

Many people are frightened by it for some reason or another, so this tutorial will attempt to provide you the most basic of commands to enable navigation and basic system actions from the comfort of your keyboard.

Let's get started shall we? Since everyone's Mint version can be different, I'm not going to detail how to actually open the terminal. I'll assume you can find it in the menu or by right-clicking in the desktop.

Facts:

- You can do almost anything in a terminal which you would also do from a GUI interface.
- Most commands were designed first to work in the terminal, then a GUI put on top of them. That's why some GUI's may feel clunky - they were an afterthought at times.
- The default location for your terminal to open from the menu is in your home folder, also known as ~
- Your current directory can be noted by the **.** operator. Most commands when they act on the current folder selection, operate on **.**
- Commands, locations, and files are case sensitive. /home is not the same as /HOME or /Home.
- Use the tab key to complete file names. If you have a long driver titled, for example,
  driver-128947232jaseu.sh, simply type dri and it will fill in the rest, provided you don't have 2 names starting with "dri" and if you do, add

another character to make it "driv" and try again.

- Almost any command can be read about in full using the manpage or by typing -h or --help after writing the initial command. This syntax is either **man command_name**, **command_name -h**, or **command_name --help**.

- To get even more information, you can use **info**. A command can be searched for by using **info command_name**. For most of these commands which are part of the coreutils package, one can find info as well using **info coreutils command_name invocation** where **command_name** is replaced by the command searched for.

- Almost any command can also explicitly display what is happening. This is done usually by the **-v** or **--verbose**

- You can specify multiple command flags to a command at a time to get more information (see the **ls -al** example below.)

- Command names are not always obtuse - due to space limitations in the old days of Unix they were shortened, and the conventions stuck.

Commands:

cd -> Used to navigate the directories. You can move to any location by path.

cd This will move you back to your home, same as cd ~

cd .. This will take you back exactly one directory. Starting in /home/justin/Desktop, cd .. will put me into /home/justin. This can be expanded upon, cd ../../ from the Desktop location instead will move me 2 back, from my Desktop to /home.

cd foldername/ This will move you forward to the given folder *in your current folder*. Take note of the missing prefix / it is an important omission. if I am in /home/justin and I want to get to Desktop, I must type cd Desktop/ without the / before Desktop. Typing / before it places us in the root of file system, which is incorrect.

cd /some/other/path This will take you to the specified folder path, supposing it exists as typed exactly. Don't forget your tab completion!

ls -> Used to list folder contents. You can view many kinds of file and folder attributes.

ls By itself, ls will simply list all your files in the current folder. From fact #4, this literally does **ls .**

ls -l Provides a longer listing format including owners, permissions, size, and date modified.

ls -a Displays hidden files and folders as well as the normal listing.

ls -al Combine options to display both hidden files and in the long format.

ls -h Show file sizes in human readable format (K, M, Gbyte) filesizes instead of bytes. Often used in conjuction with the **-l** flag.

You can view files in directories you are not even in. If I am in /home/justin/Desktop, and I want to view a file in /home/justin, I can do **ls ../** list files one directory back (and not have to go back to do so.)

cp -> Copy files

cp file /path/to/folder Copies specified file to the given path.

cp -r folder /path/to/folder Copies recursively the contents of the folder to another folder.

cp *.extension /path/to/folder Copies files matching the given extension to the new folder. To copy all .doc files, it becomes **cp *.doc /path/to/folder** and the folder must exist.

cp name* /path/to/folder Copies all files starting with 'name' to the given folder. To copy all files starting with example, it becomes **cp example* /path/to/folder** and the folder must exist.

mv -> Move files

The syntax of **mv** is similar to the example above with **cp** exempt for example #2. **mv** does not take the **-r** flag since moving a folder also moves its contents. The syntax is not exact in all instances, but works with the above examples. Consult your manpages for more details.

rm -> Remove files

For all intents and purposes, removing files via **rm** is permanent. It does not use the Trash bin. Use with caution and make sure you are deleting explicitly what you want, not what you think you want. If you decide to get fancy with your delete commands, it's probably going to come back to bite you.

rm file Remove the specified file from the system**.**

rm -r folder Remove the specified folder from the system

rm -rf folder Removes the specified folder **forcefully** from the system. This command can severely break your configuration if used incorrectly as it will not prompt you if something critical is being deleted. If you have to use this, chances are something more is broken or there was a mistake made. **This should only be used as an absolute last resort method and is not recommended.**

nano -> full command line text editor

One can edit files using **nano** in a terminal to do quick and dirty files all the way up to full configurations. It's handy, but keep in mind it handles plain text files and programming files, things like MS Word documents will not open properly!

If a file is owned by root, it is not editable as a normal user. **nano** must be prefixed with **sudo** in order to save changes. Otherwise, it will open in read-only mode.

nano newfile.whatever Nano creates a new file of the specified name and opens it for editing.

nano existing_file Nano opens the existing file for editing.

From inside **nano**

Save file using the ctrl+o key combination, and either change the name or press entier to keep the same name. This will save the file.

Exit nano by using ctrl+x key combination. If you have unsaved changes, it will ask if you want to save.

mkdir -> Create directories

mkdir folder_name Creates the folder with the specified name

mkdir -p /path/to/folder/name Creates each folder as necessary. To create folder /home/justin/newfolder/2ndfolder, and only /home/justin exists, using **mkdir -p** will make both directories newfolder and 2ndfolder.

ps -> List processes

ps aux List all processes in detail running on the system, including user, Process ID (PID), and name of process. Using this, one can view their process list and if necessary, kill unnecessary or stalled processes.

kill / killall / xkill -> Kill offending processes.

kill PID PID is a number referencing the offending process. One should obtain the PID from a command like **ps aux**. If a process refuses to die, one can alternatively specify **kill -9 PID** which should terminate the process by any means, even uncleanly or if it will mess up the system.

killall program Killall kills *by name* all instances of said program. If there are for example 3 firefox sessions open, **killall firefox** will do just that; kill all firefox sessions. **kill** would simply take the specified PID of the offending firefox process you wish to kill, and kill that one only.

xkill is a GUI way to click and kill windows. Typing in **xkill** should present a skull-and-crossbones icon, and the next window clicked on will be killed.

Pipes -> The most useful thing you will learn in *NIX. Redirecting output of a program to anothers input.

Pipes are represented by the ' straight bar ' otherwise known as the ' | ' key.

It is a rarely used key in Windows, it is often found on the backslash key.

They are used to link commands together. **Pipes** take the output of one command and route it to be used as input for a second command chained together.

Consult more online resources with information about **pipes** and their use as there are volumes.

> and >> redirectors -> Send output to a file instead of the terminal.

> is used to *overwrite* currently existing files contents and replace with the output from the new command.

>> is used to *append* information to currently existing files. This is useful for logging.

Example: **ps aux > processes.log** Sends the output of **ps aux** to the file **processes.log** for viewing the command output in a text editor and overwrites the current contents of the file.

tee -> Send output to both a file and the terminal

tee is used in conjunction with a ' | ' in order to take the command output and send it elsewhere. This is useful if there are errors which fly by the screen before you can read them, this way whatever goes on the screen is also captured to a file.

Example: **dmesg | tee boot.txt** would run the command **dmesg** which shows the initial boot info, and the ' | ' sends the output of **dmesg** to **tee**, which then does its job by sending it to the terminal and to the log file **boot.txt**.

File Execution -> So you want to execute files or programs from the terminal? Make sure it's marked executable. If not, see Quick Tip #4 below.

Need to execute a file in the current directory after it is marked executable? The **./** operator can execute the file as a normal user provided you do not need root rights. **./** literally means "in the current directory" so it does not work on files outside of the present directory.

Need to execute a file *not* in the current directory? You must pass the path to the proper executing program. If it is a python program, it's **python /path/to/file** and if it is a shell file, it is **sh /path/to/file** as an example. There are of course other programs, but these will be the most common for beginners.

Need to execute a file with root rights because you received operation not permitted? Prefix the command with **sudo**. Thus, from the above example, **sudo python /path/to/file** will execute the script with root rights.

Need to execute a GUI program from the terminal? Simply type the program name (case sensitive!) and it will launch. This will render the current terminal unusable. Closing the terminal while the program is open will kill the program. A better way is to background the program, using **program_name &** and then typing

the word **exit** at the terminal to close it and keep the process running.

Need to run a GUI program with root rights from the terminal? Prefix it with gksudo or gksu and **not sudo**. Using sudo to launch GUI applications is a bad habit and should be avoided.