Server System Management - Linux

Lab 02 - Package management

# 

# Lab 02: Package management

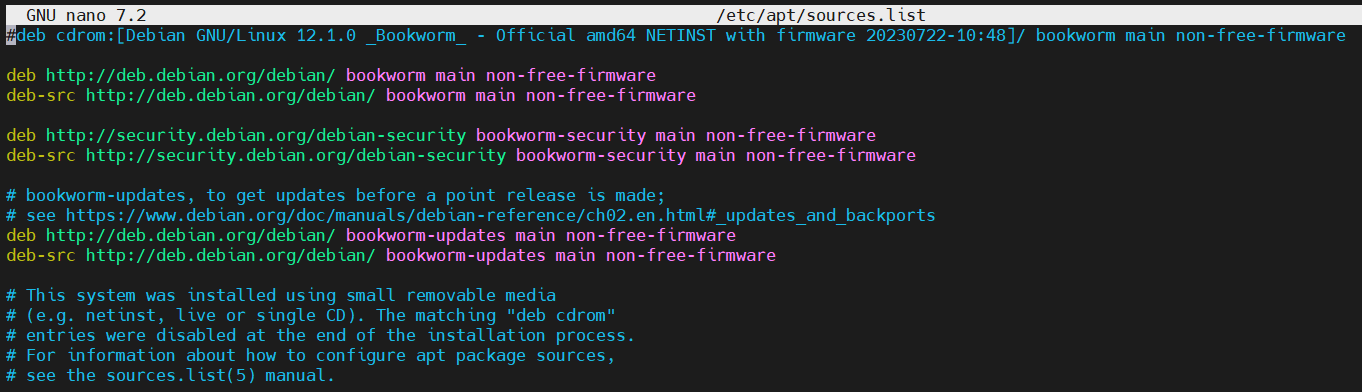
01 - Log in on your vm as your non-root user using ssh from your host like we saw last week.

02 - Switch to root.

su —login

03 - Use nano to examine the /etc/apt/sources.list Make sure only the lines pointing to the Debian mirror are active (no hash tag in front of them).

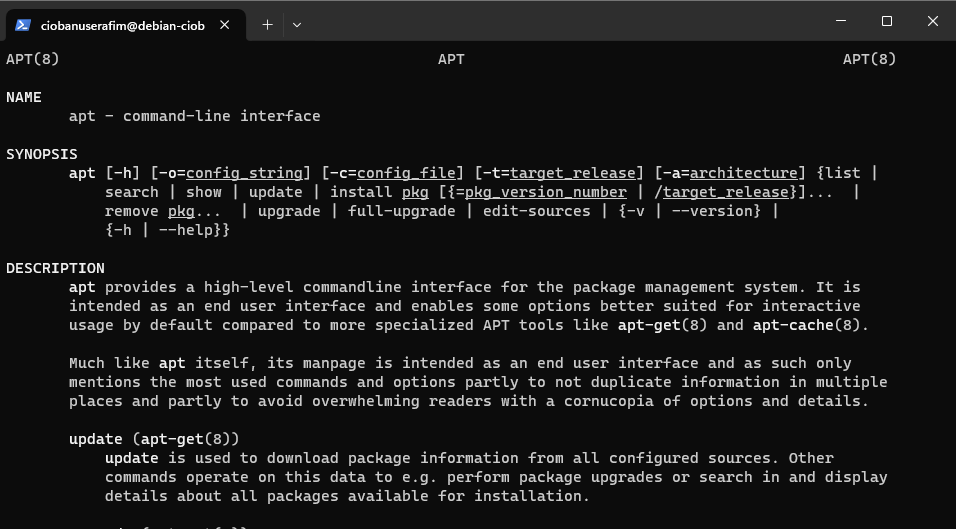
It should look like this :



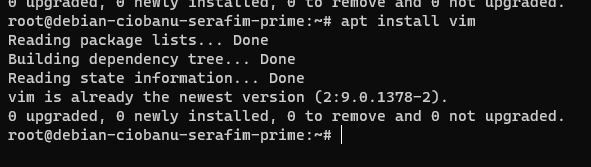
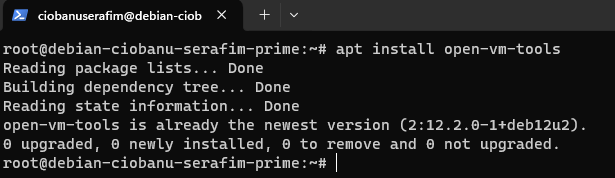
If those repositories (or similar) aren’t listed, you messed up during install. You’ll need to add the deb & deb-src entries manually and Save. Now you can install packages using apt

04 - Run the command apt update. Read man apt and scroll to the SEE ALSO section at the end (or search for this string, which usually occurs in man pages) to see all related commands. What do you think apt update is for ? (QUIZ)

`update is used to resynchronize the package index files from their sources.`



05 - Use apt to install : **open-vm-tools** (which will allow smoother integration of your vm and host system) and **vim**, a powerfull editor that you can/may/will use a lot in your Linux career.



06 - Try and connect to your machine using the VMWare workstation ssh client (right click on your vm and choose ssh). Try and connect as root. Does it work ? Why (should it) (not) ? Try and connect as your regular user. Does that work ? (QUIZ)

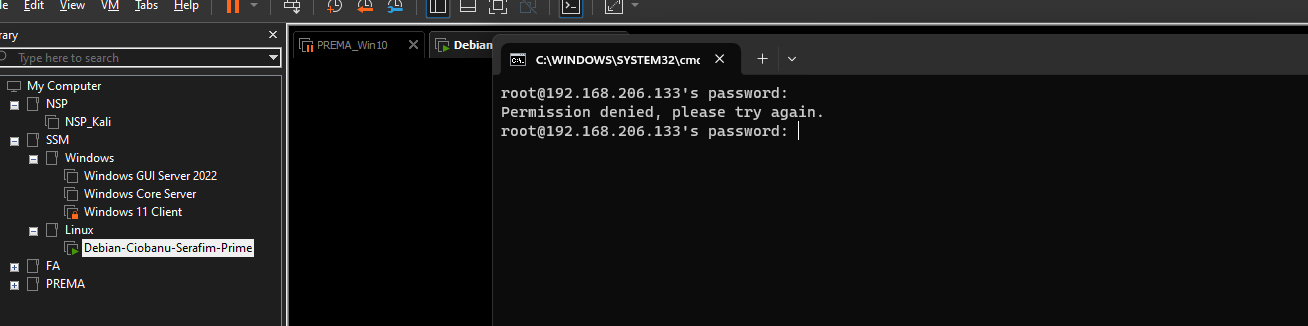
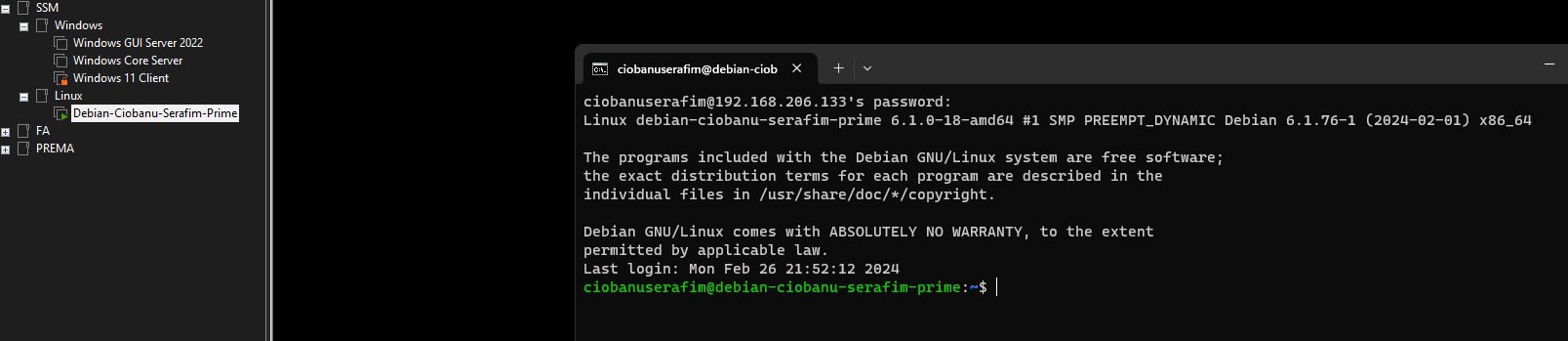
Simple user – works

Root user – does not

It probably does not allow you to login as user either of lack of privileges, or some security features, or firewall rules, or maybe even because I do not have a separate root password.

UPD: I have found a forum that said that you need to configure it in the sshd\_config.

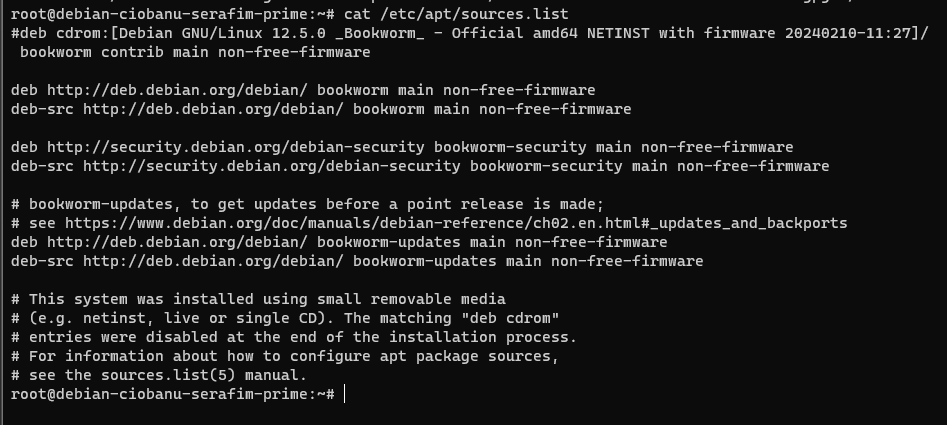
https://vmware.github.io/photon/assets/files/html/3.0/photon\_troubleshoot/permitting-root-login-with-ssh.html



07 - If you wonder what the apt sources are all about, clook closely at the contents of the apt sources list configuration file .

cat /etc/apt/sources.list

In other words, apt update fetches package and catalog information (about all available Debian packages you are interested in) from [ftp.be.debian.org](ftp://ftp.be.debian.org) or any other you choose.



You can set up mirrors from it, and hence this is what also happens when you add a package that is not from the apt repositories.

08 - Get just the lines starting with "deb " using a regular expression

grep '^deb ' /etc/apt/sources.list

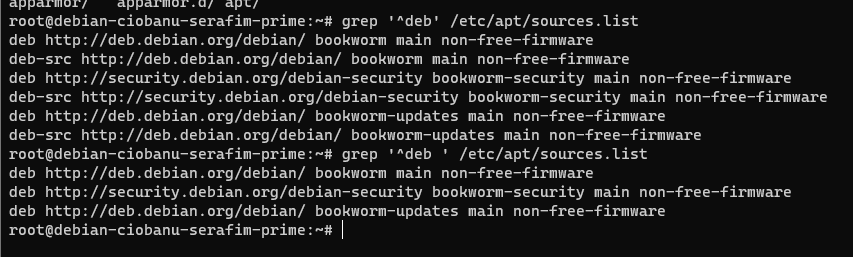
The man pages for grep and egrep describe regular expression syntax concisely.

This pattern means:

• ^ matches the start of a line

• deb means we expect the word "deb " including the space (to skip matches for

deb-src and lines that have been put in comment with a leading # sign)



So - ^ is to make sure that the line starts with the word. The ‘deb ‘ space is there, to make sure that you only get the lines without the deb-src

09 - To better understand these deb lines, read man sources.list and use the / character to search for THE DEB AND DEB-SRC TYPES, which describes the format of such deb line.

It is easier if you write down the described parts of a deb line vertically, in bullets:

• deb - The deb-src type references a Debian distribution's source code in the same form as the deb type. A deb-src line is required to fetch source indexes.

• optionally: options - Each source entry can have options specified to modify which source is accessed and how data is acquired from it.

• uri - The URI for the deb type must specify the base of the Debian distribution, from which APT will find the information it needs.

We see http:// ftp.be.debian.org right after the word deb in our grep output, which is an uri indeed, so we can conclude there are no options (see previous bullet).

• suite - suite can specify an exact path, in which case the components must be omitted and suite must end with a slash (/).

You may have noticed we are running Debian bookworm, the current stable version (you are running version 12 -- more on that later).

Bookworm with lower-case b is the name of the distribution mentioned in our sources.list, as seen from the grep output from question 6.

**Note**

If you visit the website in the uri, it (like all Debian mirror sites) provides symbolic links stable, testing and unstable which point to the current branch matching the name. sid is always the name of the unstable branch.

• component1 –The distribution is generally a suite name like stable or testing or a codename bookworm or trixie while component is one of main, contrib, non-free or non-free-firmware.

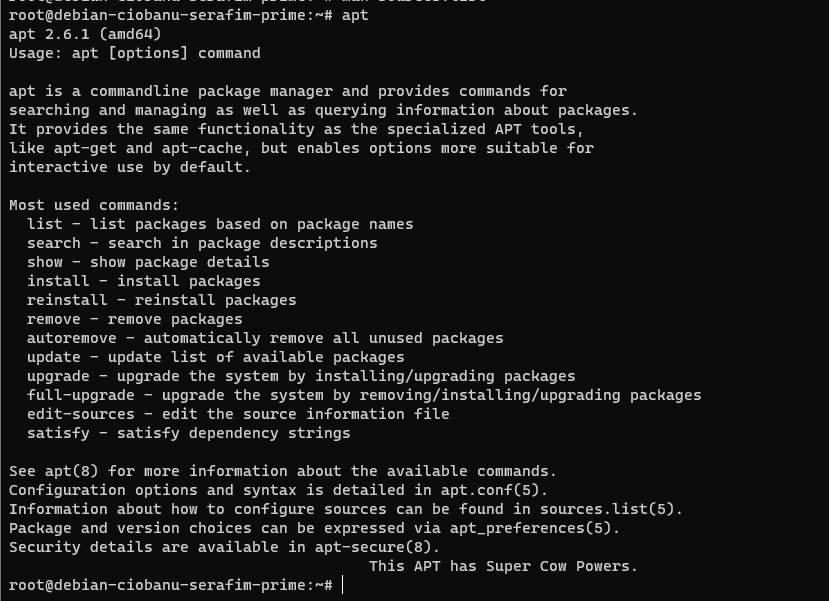
You should see main as a component in all deb lines you found.

• component2 - The distribution is generally a suite name like stable or testing or a codename bookworm or trixie while component is one of main, contrib, non-free or non-free-firmware.

Some lines maybe also mention updates as a component.

For later questions in the lab, you must remember what these components are about.

10 - Just type apt without parameters to see what apt can do, beside bringing your package repository catalog in sync with your sources.



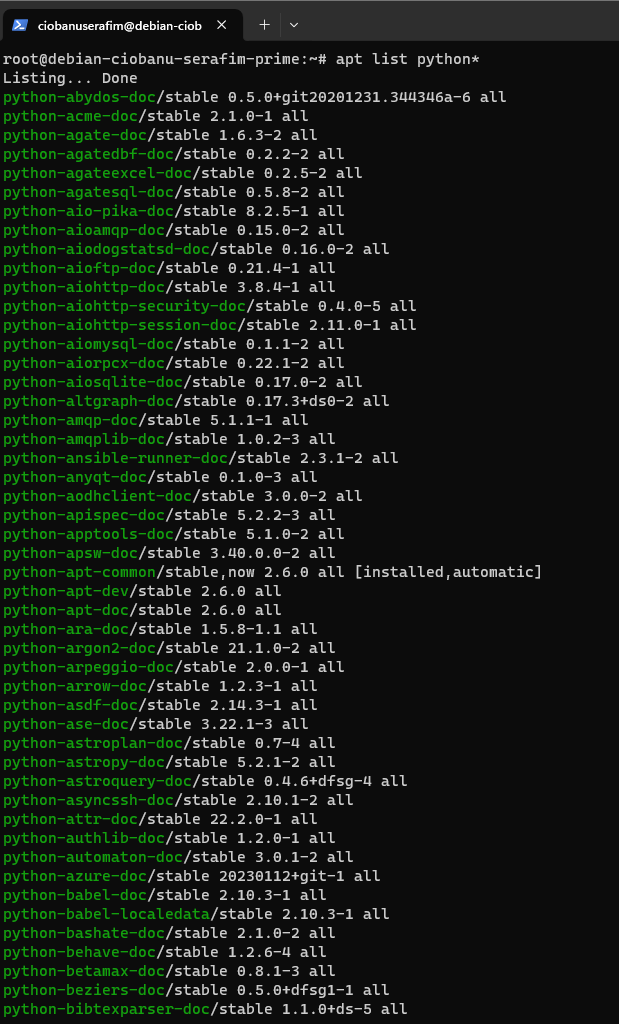
11 - Now use the appropriate apt command to list all packages available on your system as it is NOW. Which command did you use ? (QUIZ)

apt list

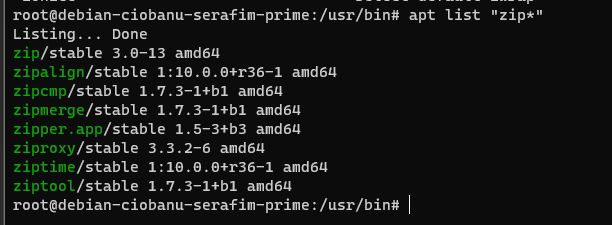
apt list —installed?

12 - That command accepts a pattern to search for. It will just search for the package(s) that matches the pattern. The pattern is a shell-like expression that allows the \* as a wildcard. Now search for any available packages whose name starts with python.

apt list python\*

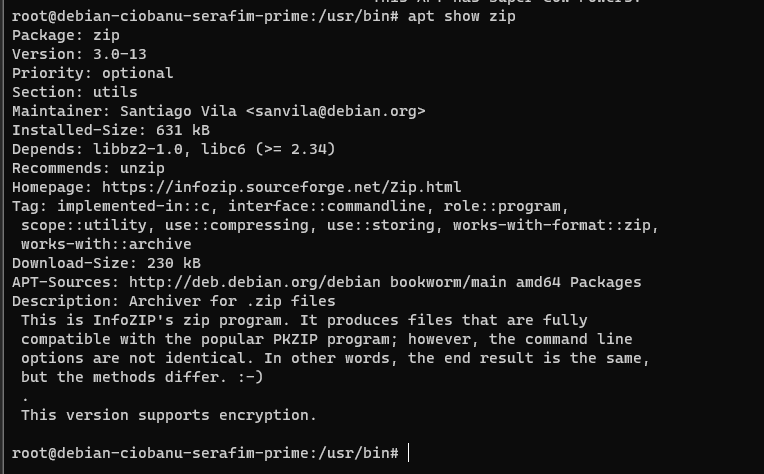


13 - Now navigate **with cd to /usr/bin** and Search for packages whose name starts with zip. As you learned before, you’ll need zip\* as pattern. Use apt list “zip\*” to protect against shell expansion! Take a screenshot of the CORRECT results (SCREENSHOT)

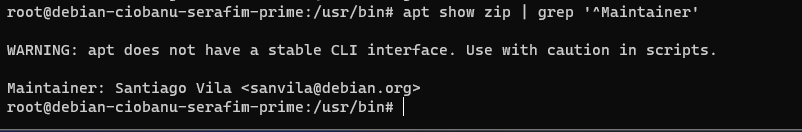
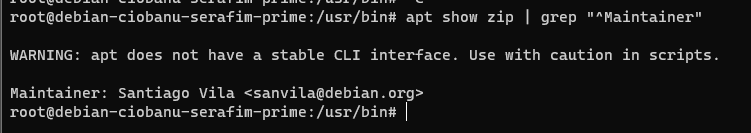


If you get nothing, you haven’t protected your \* ! How could you do that ?

14 - Again type just apt to see the possible actions. Now show the package details for the “zip” package you obtained before and study the fields that are stored in Debian's package manager. Some will be covered more explicitly later.

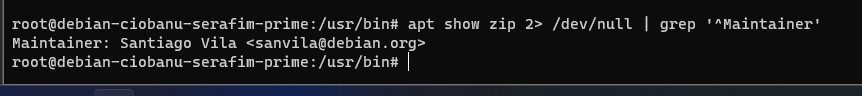


15 - Write a grep pipe to extract the line that contains the maintainer. Be specific enough: use the ^sign in your expression to match the start of a line. The output of your command should be: Maintainer: Santiago Vila [<sanvila@debian.org](mailto:%3csanvila@debian.org)>



16 - As you noticed, apt and grep do not play well together. The reason is that apt is meant to be pretty and user-friendly. Hence a b\*tch to script around. It just printed a warning message on your screen, which you usually want to suppress by redirecting stream 2 (stderr) to /dev/null.

apt show zip 2> /dev/null | grep '^Maintainer'



Practice this; it will be necessary when writing scripts that don't pollute the screen with confusing messages from various commands it executes.

17 - Another way is by using the commands apt-cache, apt-get, apt-mark. Just type apt on your bash prompt, and press Tab several times to see them. These are actually the ORIGINAL sources of the functionality you now find in apt. Check out the man pages for these commands and find out that some minute differences DO exist.

apt provides a high-level commandline interface for the package management system.

apt-get is the command-line tool for handling packages, and may be considered the user's "back-end" to other tools using the APT library.

It is more of a tool that can do more actions at once, rather than plain apt.

18 - Now let's use another handy tool, cut, to separate this line in its two logical parts: the field name Maintainer and its value Santiago Villa <sanvilla@debian.org>.

Requirements:

• split this line in pieces, using : as a delimiter of each piece

• return the second field

Now read man cut to find the options you need.

**Tip**

Use the provided context! Search for the word delimiter in the man page. Also search for the word field.

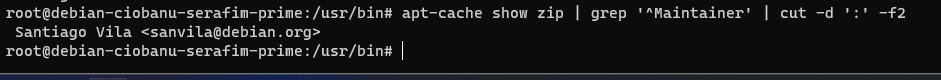
**Tip 2**

Simply using apt show won’t cut it (pun intended). Apt is a nice tool, but works its magic best on plain command line terminals. It doesn’t play well with scripts or pipes and even warns you about it. Use one of the apt-\*\*\*\* commands for this once. Abuse man to find out which.

Finally, append the correct cut command to your pipe. What does the command look like now ? (QUIZ)

Absolutely horrible

apt-cache show zip | grep '^Maintainer' | cut -d ':' -f2



19 - Now take your last command and put $() around it .

Explain the resulting error message :

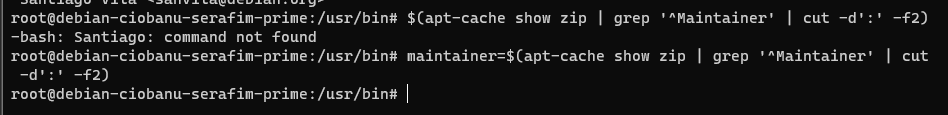
-su: Santiago: command not found

When wrapping, you are trying to give the result to the shell you are currently in.

Now go a step further and prefix that line with the $() (which was obviously an invalid command) with this:

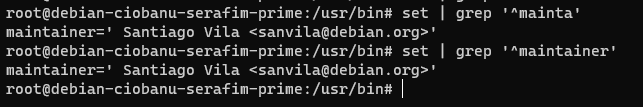
maintainer=

So you have one long line reading maintainer=$(apt-\*\*\*\* show ...... `

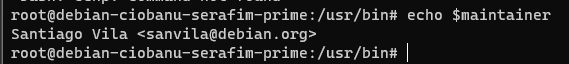


20 - As you may have realized, you just created a shell variable. Show your shell variables using the set command. The output may be long, so just grep for the maintainer variable you just created.

set | grep '^maintainer'



21 - Try also: echo $maintainer



22 - One thing you will often see in shell scripts is testing whether a variable is empty, using a if ... fi block. Use it to confirm $maintainer is not empty. Type the following (hit return after every line)

if [ -n "$maintainer" ]

then

echo maintainer is $maintainer

else

echo maintainer is unknown/empty

fi

Or the other way around:

if [ -z "$maintainer" ]

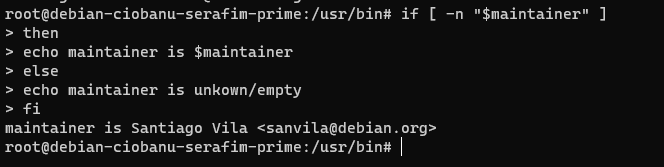
then

echo maintainer is unknown/empty

else

echo maintainer is $maintainer

fi



Now after issuing this long multiline command, press the Up arrow key. You will see that bash has concatenated your multiline command into a single line, using semicolons as separator between each token!

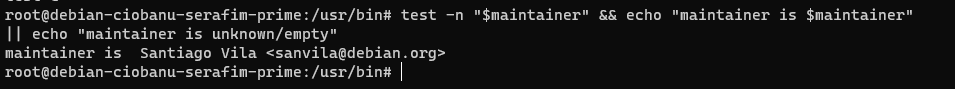
if [ -n "$maintainer" ]; then echo maintainer is $maintainer; else echo maintainer is unkown/empty; fi

If you used the zshell, you’d get an even more user-friendly behavior. You’ll encounter the zshell further in this lab series

23 - Read man test to see what other types of tests exist, such as numeric comparison. Now try the emptiness test for maintainer, using the test command rather than the [ symbol.

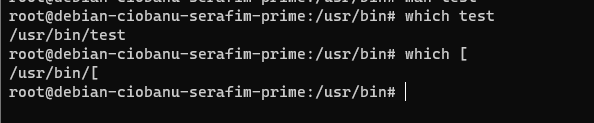
Actually, test and [ (which expects a matching ] at the end of your test, unlike the test command) do exactly the same thing, which is why this manual page explains what you did before.

test -n "$maintainer" && echo "maintainer is $maintainer" || echo "maintainer is unknown/empty"



24 - You can also find external versions (actual binaries on your system) of the [ and test programs, which are used on more primitive shells that do not offer a built-in version of these tests. Find these programs and compare them. What did you notice ? (QUIZ)

I would never even think to look it up within /usr/bin



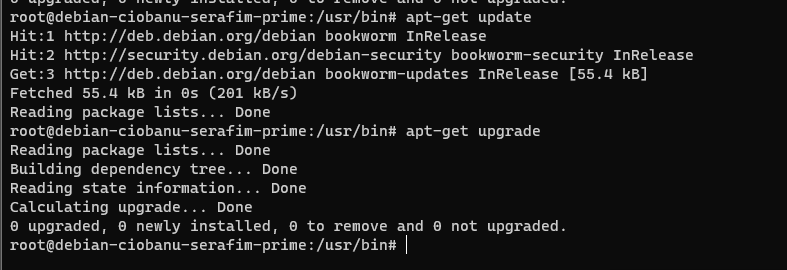
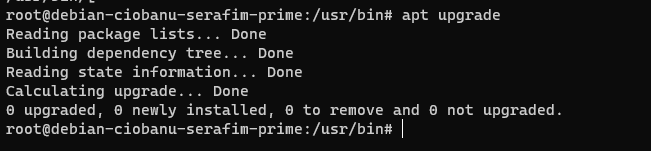
I noticed that they are binaries. You can use them by calling /usr/bin/test and etc.

You also need a “]” for the other one to work, but “test” does not need that.

25 - Now use apt to upgrade all your packages to the latest version (do not use dist-upgrade or full-upgrade!).

**Note**

Had you forgotten to apt update you might stumble upon broken links, since the packages might have been updated on the remote mirror sources since your install. It's a VERY good idea to always apt update before you install anything important



26 - Read the man page for apt again, and explain, in your own words, the difference between the full-upgrade and upgrade commands

upgrade – will update the installed packages to the new version which was fetched for them, using update.

full-upgrade – will delete the packages if it is needed to update the system. That might be the case once you probably get to a new build, like maybe 11 to 12 of debian, or something similar, which involves the major components.

27 - apt search is a tool one often uses to quickly scan for the package name, when you know a word or certain words must occur in either the name or the description of the package. Use apt search to find all packages whose name or description matches the words 'ascii art'. The output could be long, so pipe it to less.

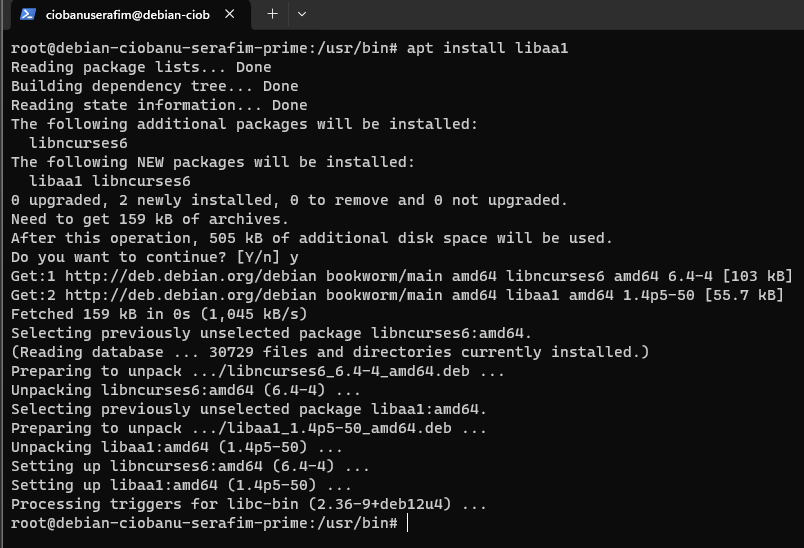
**Important**

Do not forget the single quotes! These make bash pass these 2 words as one single parameter to apt, which will search for both words next to each other.

If you forget the single quotes, apt will actually receive two parameters, and return search results for either of ascii and art.

apt search 'ascii art' | less

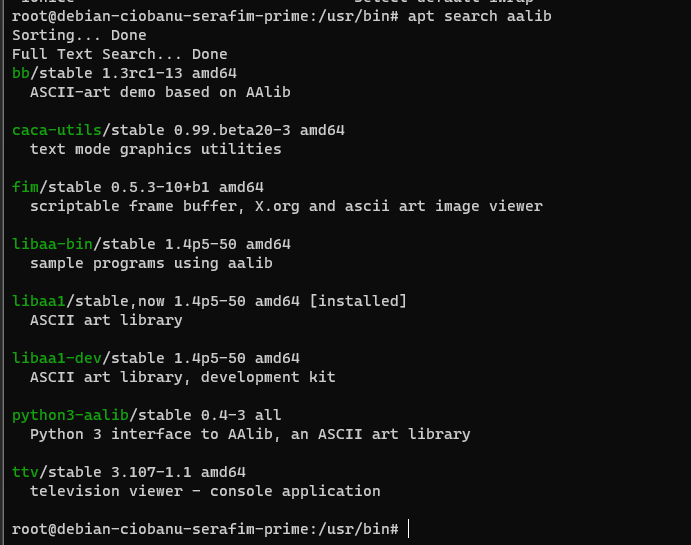
28 - Use apt to install libaa1, an ASCII Art library



29 - Also try apt search aalib. You will not need a pipe to less; the output will be short.

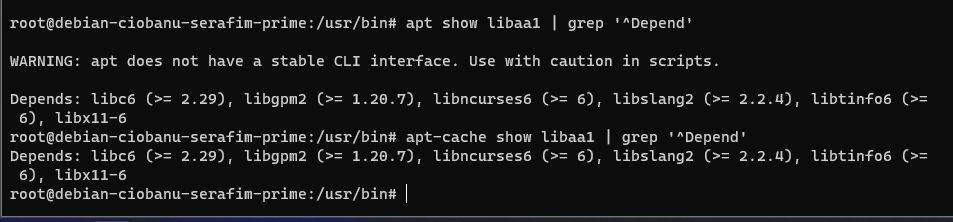
This illustrates that libraries are sometimes called libXXX and sometimes XXXlib - sometimes the name refers to the package, sometimes to the actual library ELF object file living in /usr/lib. Even package maintainers get it wrong sometimes!

So, the fact that search aalib returns results for the libaa package is not due to intelligent input processing like Google does.



30 - Use apt show to display details about the package you just found and installed. This time we are interested in the dependencies for the library.

1) Execute a command that greps for the line we want to see from the output.



2) Learn to use the for loop in bash.

You should have discovered that libaa1 depends on libc6, libgpm2, libncurses6, libslang2, libtinfo6 and libx11-6.

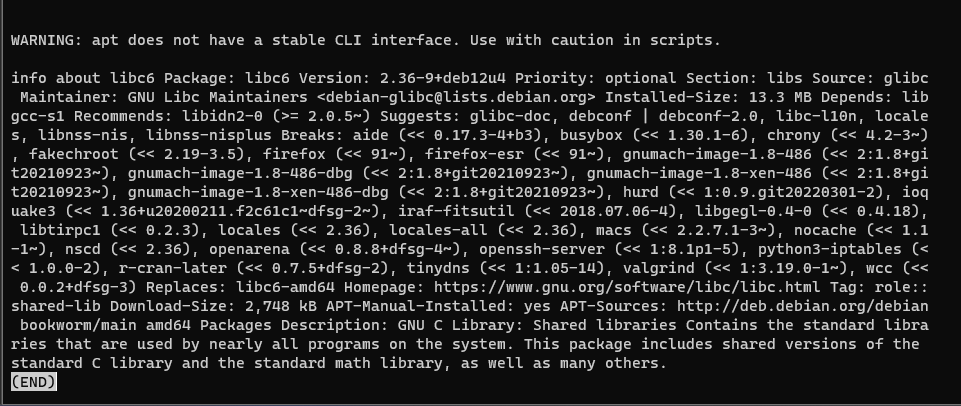
Try and study this multiline for command to show package information for each of these dependencies:

for pkgname in libc6 libgpm2 libncurses6 libslang2 libtinfo6 libx11-6

do

echo Info about $pkgname $(apt show $pkgname) | less

done



**Tip**

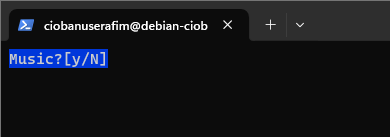
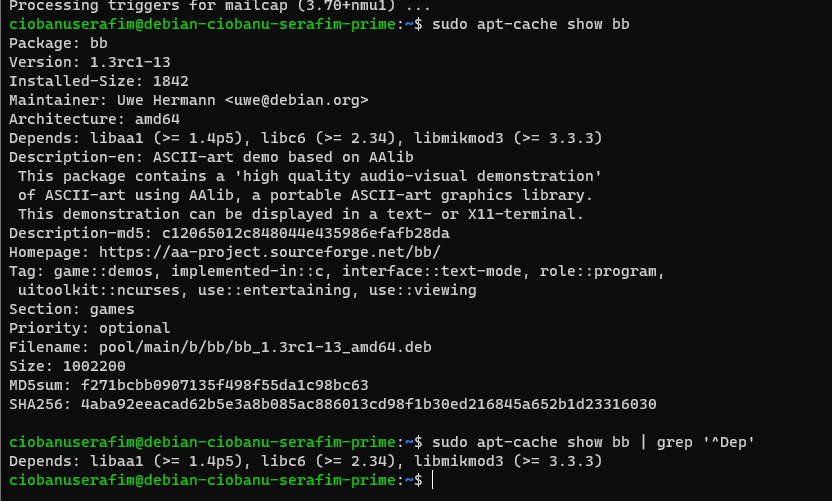
Like before, try the Up arrow key to see how bash again concatenated your multiline command to a single line, using semicolons as separator.



You may want to take the time again to properly read some of the dependencies of these dependencies, to get an idea of how a dependency database looks like.

31 - Just for fun, install bb using apt. Confirm it depends on libaa1 using apt and grep. Make sure you are still root and try to launch bb. This should fail with a command not found error. Try and find out why (QUIZ)

sudo apt install bb



As a simple user – you can. As root – you can not.

I checked it out, and it is not in root’s path. Also if you check the details of it, it will tell you it is a game. Hence it should be in the /usr/games

32 - Study the dpkg tool

So far, we used apt. This program does lookups or updates in a local package repository/database and can install or remove packages, upgrade all packages to the most recent version, always aware of where the packages are by using /apt/sources.list, which points to a cd, dvd, usb, mounted image, or, in our case, to Belgium’s Debian mirror site.

The package format is actually called dpkg, and dpkg is a second important command-line tool that is nowadays mostly used for doing queries on installed packages or files.

As an introduction, read the man page for dpkg. Focus on these important topics:

• Package states

A package can be in more states than simply installed or not-installed!

It could have been unpacked but not installed yet, or it could have failed during installation, in which case some triggers may still be pending.

Also, it might have been removed (but not purge’d) at some point, using apt remove, but its config files have been preserved: state config-files.

• Selection states

Usually, you don't have to worry about the difference between installation state and selection state, as they are quite similar as you will read.

This does explain why the installation state is listed as two letters by dpkg -l, as you will see further on.

• The ACTIONS (search using the / slash key) listed under dpkg-query actions:

• list – will list all known packages according to a pattern specified or something similar.

• status – will return the status of all packages, or one package (if specified)

• Listfiles – will return a list of files installled with the package you specify. This is probably useful when you have configuration files for some tools, utilities.

• Search – search for packages that own files corresponding to a pattern givenl

This list uses capitalization equal to the single-letter command line switches: -l, -s, -L

and -S.

You will need these often,so make sure you can quickly find them in the man page whenever you need them.

**Note**

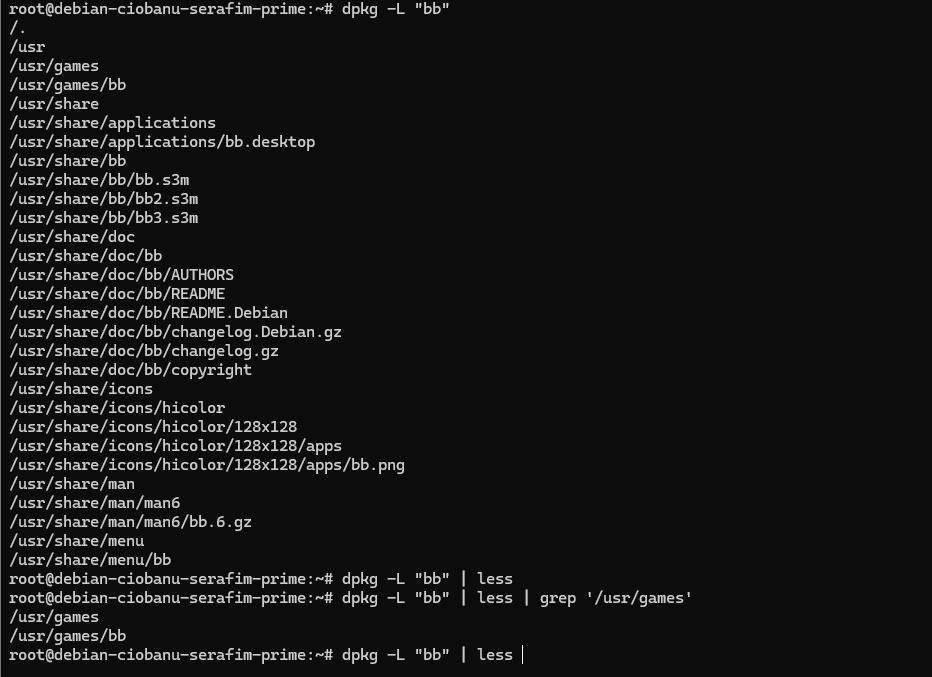
We are actually using dpkg as a frontend for the low-level dpkg-query tool, but this is how it is intended.

We are not really interested in the other things dpkg can do,because we established a habit of letting apt handle these things, like installing and even first downloading packages.

This means we usually use apt as a frontend for dpkg (and wget to download files from the mirror sites), but dpkg in turn uses dpkg-deb and dpkg-query. This is in the UNIX spirit of combining powerful small tools that perform one task well. You have seen this elaborately in OS Concepts

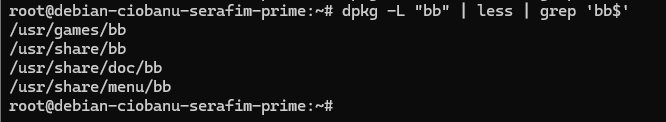
33 - Use the appropriate dpkg option to list all files in the just-installed bb package. Pipe to less. Search for the games string to confirm the binary was indeed installed under /usr/games, not/usr/bin.

dpkg -L "bb" | less



34 - Pipe the previous dpkg command to grep, to find all lines ending in bb. Protect your pattern using 'single quotes' and use the $ symbol to indicate you want to match the end of a line.

dpkg -L "bb" | less | grep 'bb$'

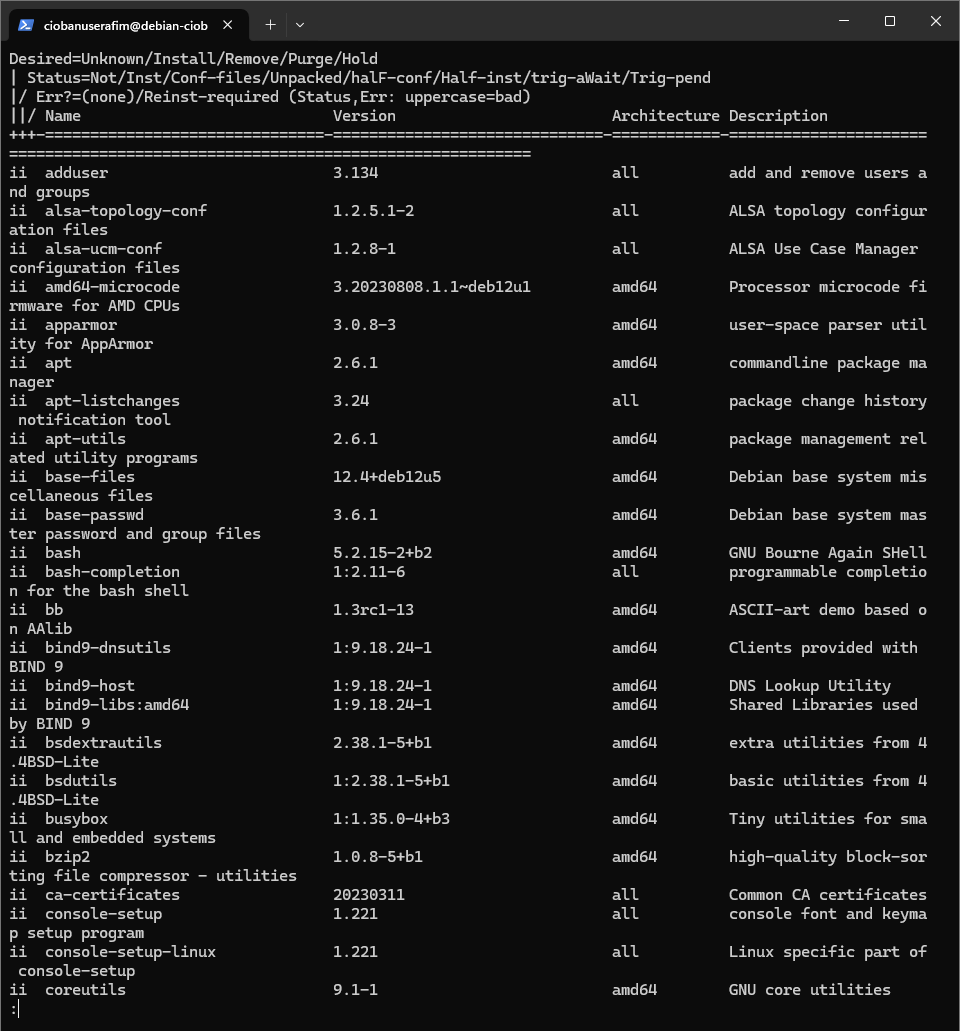
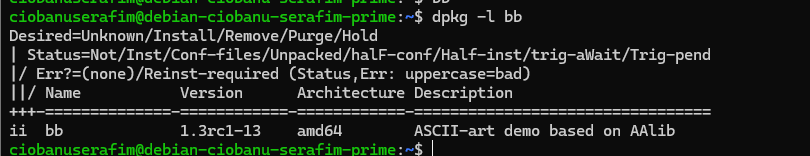


35 - Launch bb as a regular user (escape from your su session by pressing End-Of-File = Ctrl-D). When watching the demo, you will understand the role of the libaa1 library the bb package depends on (aa stands for ASCII art). What do you see on your text command console ? (QUIZ)

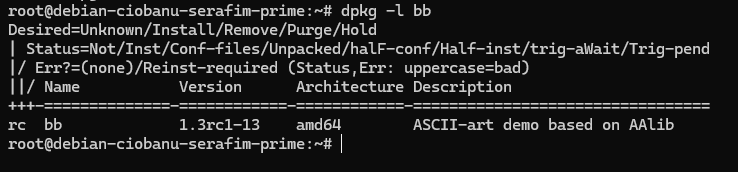
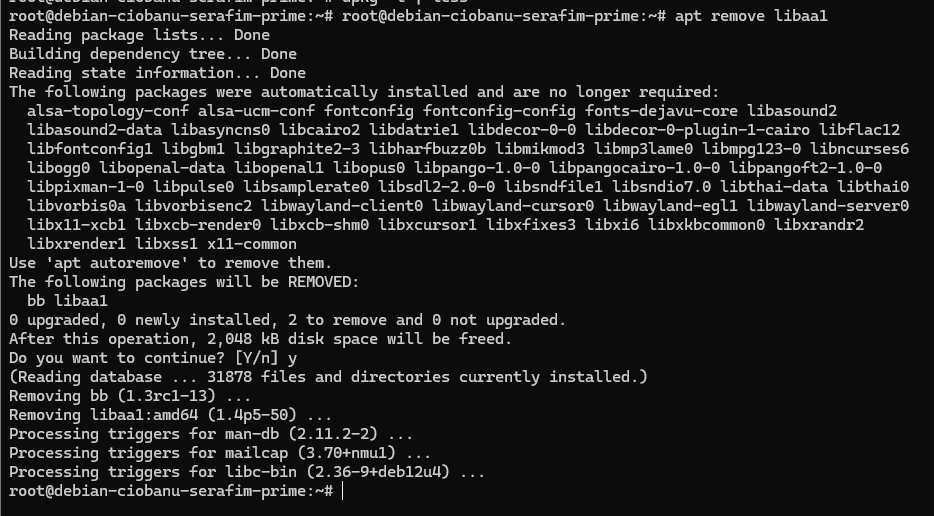


We see a bunch on characters, which are ASCII, and also before that it takes some time to calculate some data, so it might be that it actually decided to check the size of the screen maybe, and some other computation, for proper show of the characters

36 - Use dpkg -l bb to list packages matching pattern bb.Take special note of the prefix ii. It refers to the selection and installation state. Now type dpkg -l | less to see all packages and their installation states.

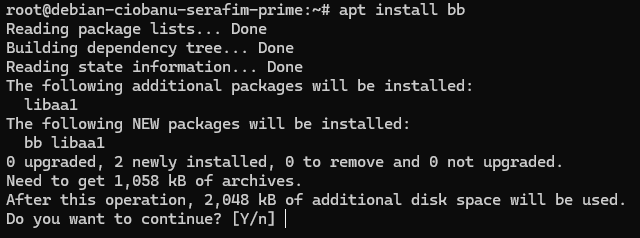


37 - Use apt to remove (not purge) libaa1. Check which packages are actually removed. Why was bb removed ? (QUIZ)



It got a actiov “remove”, and package status “config-files”. It was uninstalled aswell, because of the dependency which we deleted, and hence we cannot launch it.

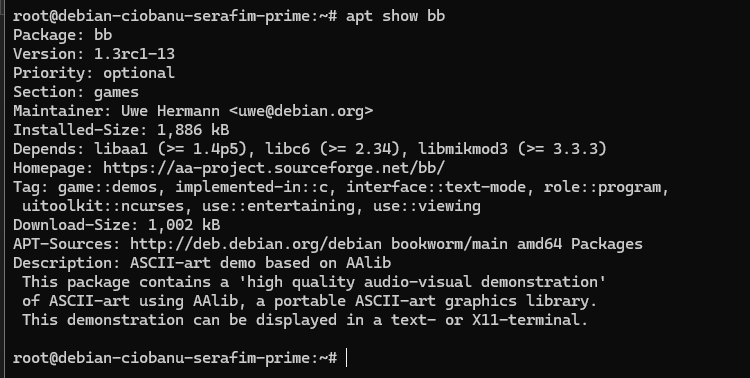
38 - Now reinstall bb. Which packages are actually installed?



bb and libaa1 (package and its dependency being resolved by apt and etc.)

39 - Why is libaa1 installed? If in doubt, check the info apt show bb gives you..

Because it is a dependency of the package!



40 - There should be a message that might have been bothering you for a while…

The following packages were automatically installed and are no longer required:

(followed by a list of packages)

Explain in your own words why Debian / APT thinks these packages are no longer required

They were installed along with a package, and now that a package is gone, there is no need for those (probably also considering that they are not used by any other package), and they can be removed without any problems

41 - You could now consider following the given advice, to use apt autoremove to get rid of those packages you no longer need. This removes nothing! Why?

Because it should be used when you want to remove the package, and provide the option `autoremove` and not `remove`. Then it will probably resolve the issues.

42 - Use apt to remove bb.

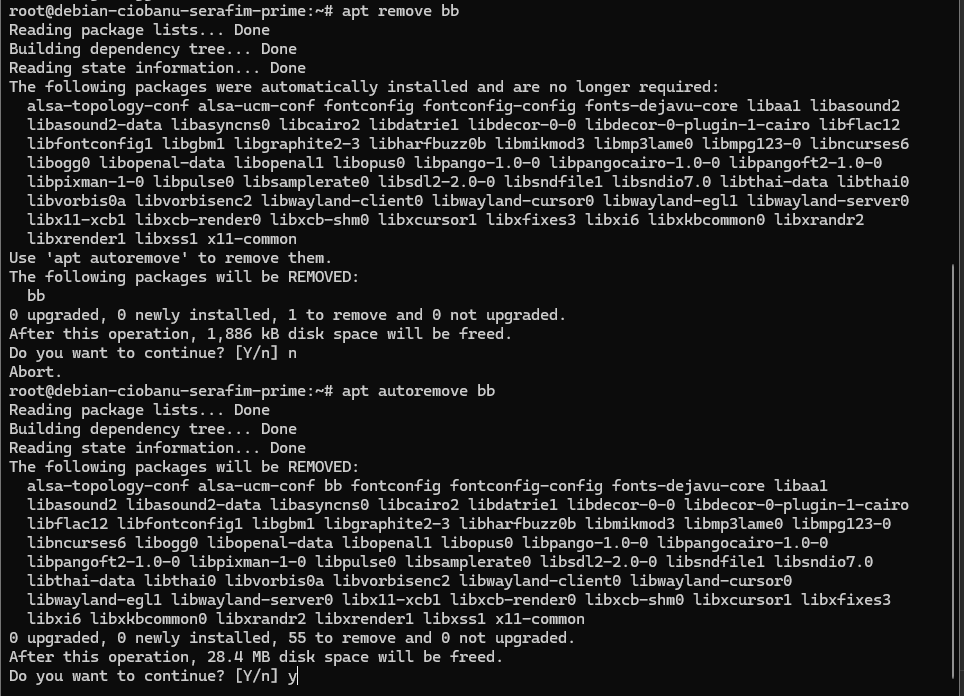
*There's always a bigger fish: when recompiling kernels you'll encounter make clean and make mrproper (or in some programs make clobber)... and when removing Debian packages, you could opt to purge them instead.*

*Purging removes all configuration files that were still remaining on the system, in addition to the actual package.*

*Normally, these are left alone when removing a package, because they do no harm and they might still contain custom modifications you made which you may wish to preserve, in case you reinstall the package again later.*

**Note**

The following packages are no longer required message reappears. Makes sense: this time libaa1 (and other packages it depends on, such as libasound2 for sound support) was automatically installed when we (re)installed bb.



I used `apt autoremove bb` to uninstall the other packages, if that was necessar the question. I could have used `apt purge bb`, but I left it as it is.

43 - type dpkg -l libaa1

(I installed bb back, and removed it again with `apt remove bb`, so the previous comment is not valied, oops)

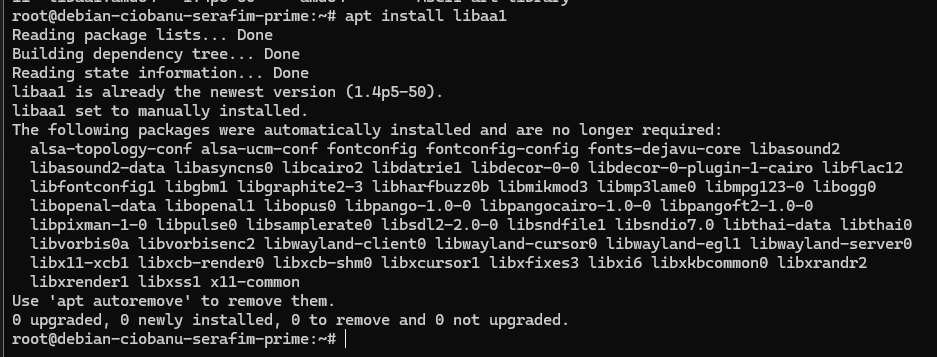
Be aware that libaa1 is still installed.

The first time we installed it manually.

The second time, it was automatically installed because we (re)installed bb which depends on libaa1.

When removing bb, libaa1 was not automatically removed. This is what the warning message is about.

Against intuition, type the command to install libaa1, knowing very well that it is currently installed. In addition to informing you libaa1 is already the newest version, apt tells you something else. What is it?



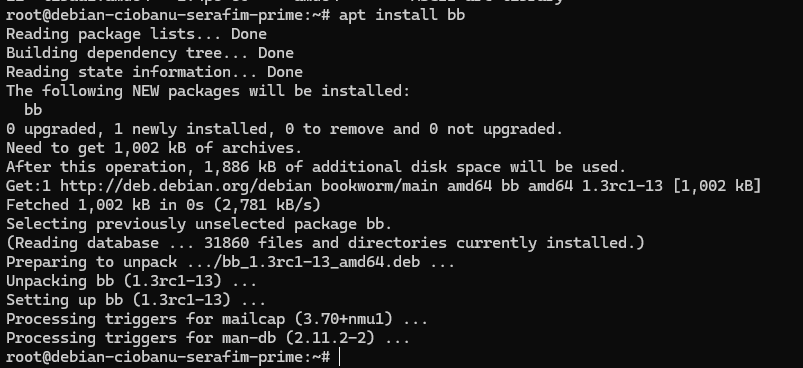
It tells something similar to when removing the bb package. As it is also a dependency, it understands that packages are not used, and hence offers to get rid of them aswell.

44 - Note libaa1 has now disappeared from the list of packages that were automatically installed.

That's because we explicitly typed the command apt install to install this package. It is no longer considered installed automatically (as a dependency for another package that was installed, like bb).

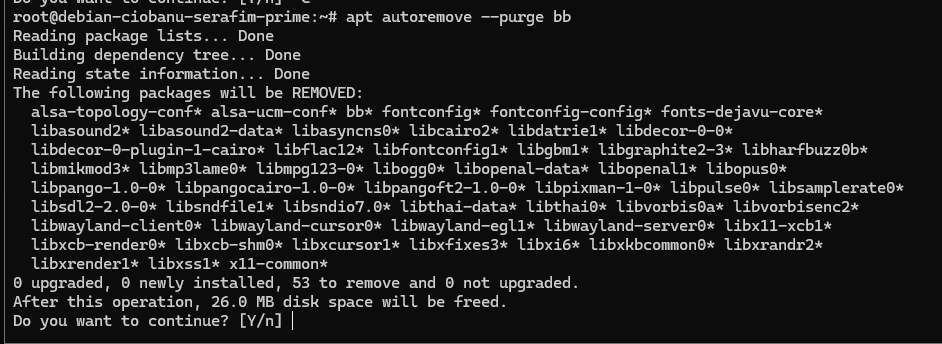
**Note**

Not important for these labs, but you can update the automatic/manual flag directly using the apt-mark command; you can find out how in its man page.

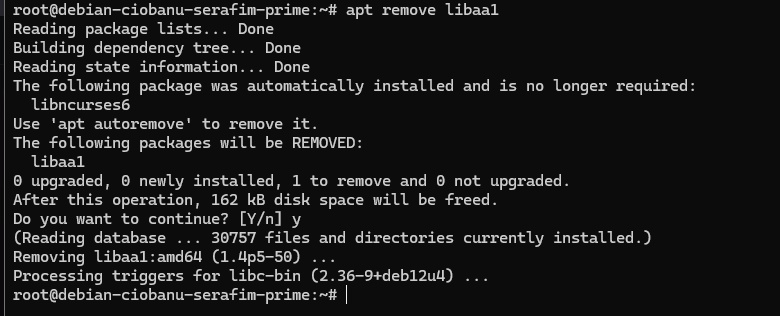


Indeed

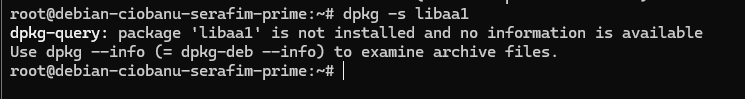
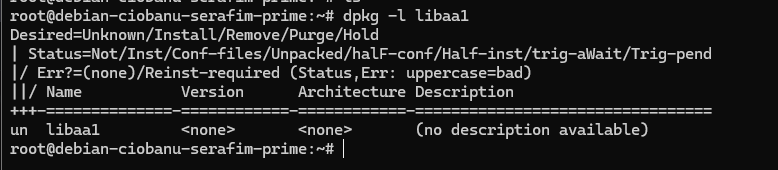
45 - Type the appropriate apt command (use the man page if necessary) to autoremove and purge all packages that were automatically installed, but no longer required. (QUIZ)



46 - Note that libaa1 was not removed, as we predicted before. Remove (but do not purge) it using the apt command.



47 - Check the package state for libaa1 using the dpkg -l and dpkg -s commands. dpkg -l will give you two letters, and dpkg -s will give you full details (if available).



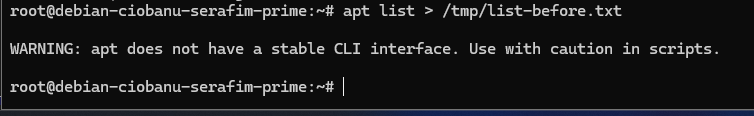
Remember the man page for dpkg which you have read before. It explains the meaning of package status and selection status.

In this case, we can infer that the package:

• has been deinstalled (correctly) (un – Unkown Not-Installed)

48 - Use apt list again to write the list of all packages available in our repositories (sources.list).and redirect the output to a file /tmp/list-before.txt.

apt list > /tmp/list-before.txt



49 - Edit the apt sources

There are numerous cases where editing a config file by hand is possible, but discouraged.

For example: vipw instead of nano /etc/passwd, vigr instead of nano /etc/group

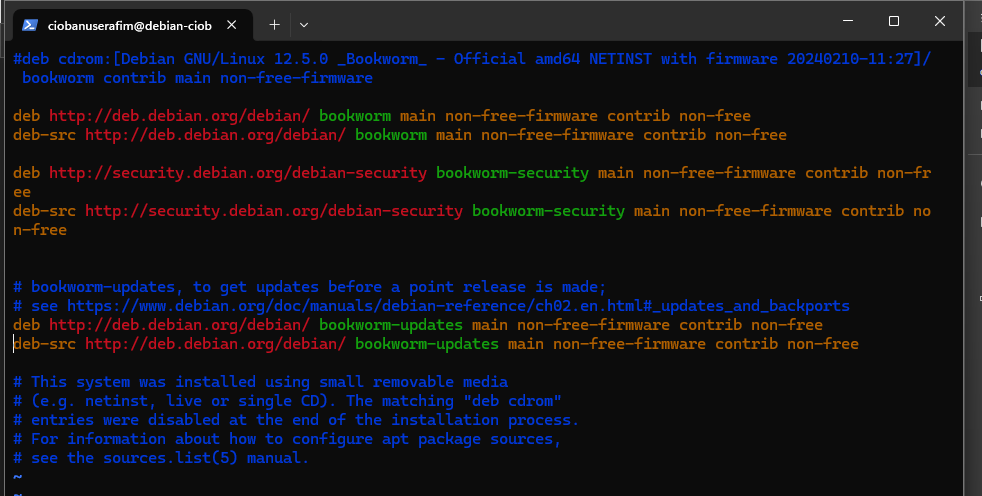
The reason for providing a wrapper editor script is each time:

1. To make it possible to check if the syntax is correct for that particular config file, and alert you (whereas nothing stops root from making invalid edits using vim or nano)

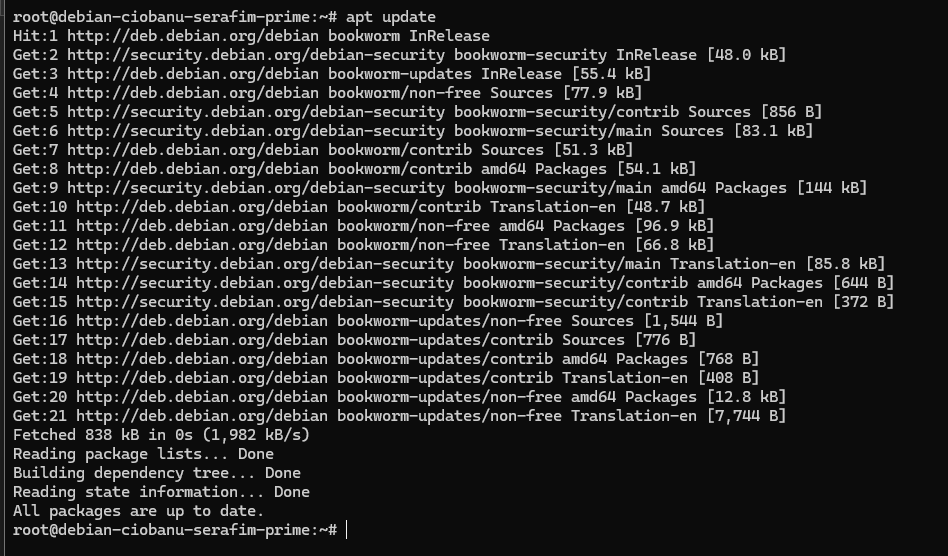
2. To lock the file; other users will get an error when trying to edit the file while another administrator is editing it.

Now read the output of apt again and learn a new editor wrapper to edit the file /etc/apt/sources.list we have encountered a few times already.

Add the components contrib and non-free to each of your deb lines, and save and exit your editor.

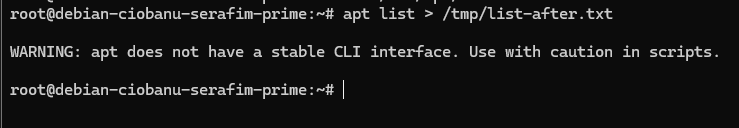


Now run apt update again, to get these additional components of the Debian Stretch repository.

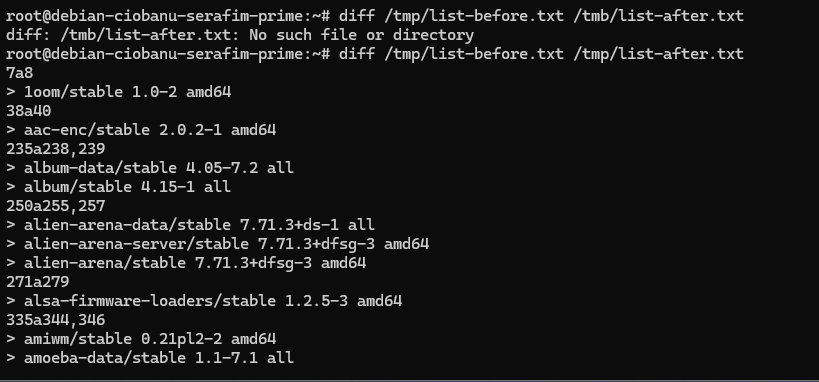


50 - Again redirect the apt list output to a file, this time /tmp/list-after.txt. Now by checking the difference between both lists using the diff command, you’ll know which additional packages have become available.

apt list > /tmp/list-after.txt

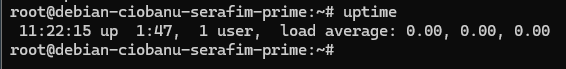


diff /tmp/list-before.txt /tmp/list-after.txt

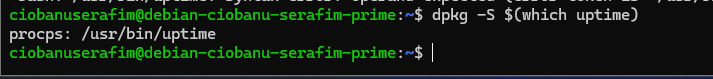
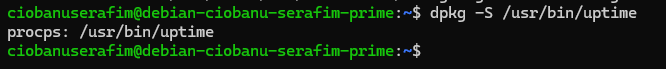
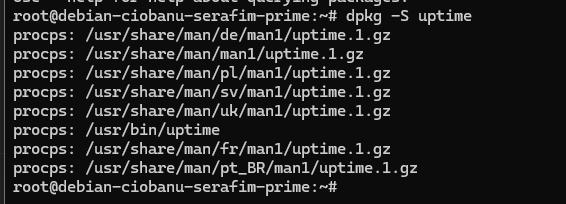


51 - Your VM is always ready for you, and it has been for a while. Use the uptime command to see how long your VM has been running

Been booting it up and down



52 - Now use the appropriate dpkg command to see the package the uptime command belongs to. (QUIZ)



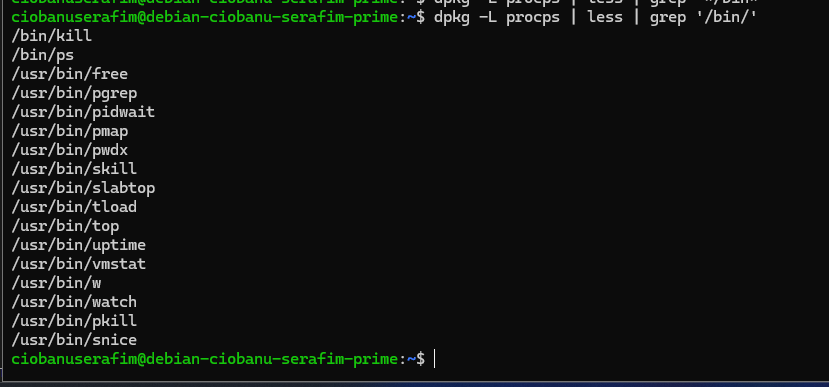
53 - Examine this package

Show all the files in this package, using the appropriate dpkg command. Pipe to less, since the output will be more than one screen.

dpkg -L procps | less

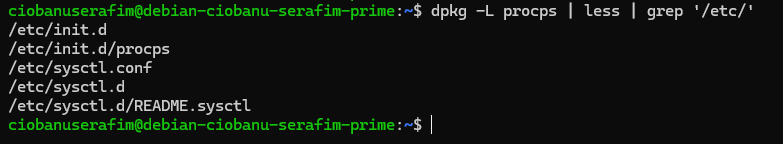
Now pipe the command to a grep that filters all lines that contain /bin/, to quickly see the other programs in this package.

dpkg -L procps | less | grep '/bin/'



You can learn about them by doing man on each of them.

54 - Similarly, now grep for configuration files. Grep for the pattern /etc/ to see those, since most config files live under /etc on a UNIX-like system.



You should find /etc/sysctl.conf. It sets various kernel parameters, and just reading it through less gives a glimpse of those. Just read it if you're curious.

The accompanying sysctl tool also has a man page.

**Tip**

From now on, always do this for each new package you install, or use dpkg -S first to discover the package a file belongs to.

The config files belonging to a package are always a great source of information about what the package can do, as Debian maintainers usually write comment lines describing most possible configurations.

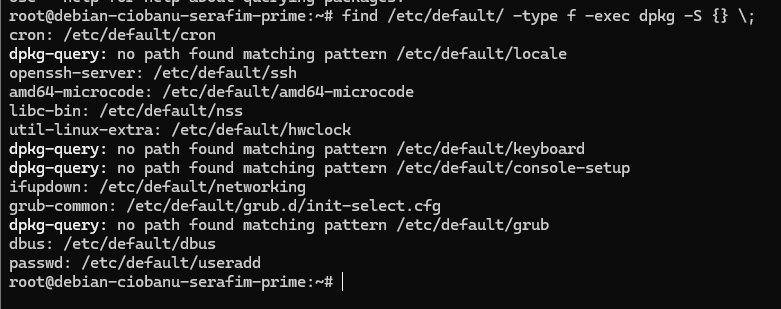
55 - A directory that stores various default environment variables and settings in 1 location for many packages: /etc/default. We’ll study it now.

Now use the find command (man page !) to find all files (-type f) under /etc/default, and let it execute (-exec) the command dpkg -S on each of these files ({} sign is replaced by the current filename).

End the command with \;.

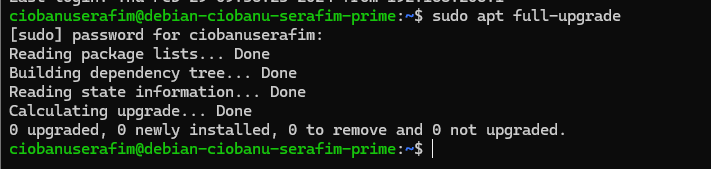
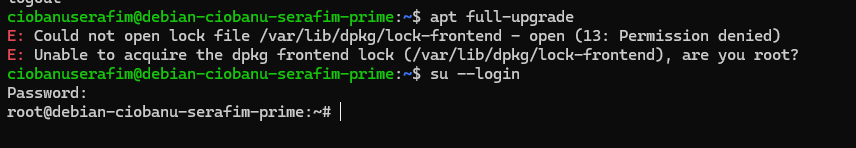
(The command would not work until I read about \;)

find /etc/default/ -type f -exec dpkg -S {} \;



You should learn that some of these defaults files belong to one or even more packages, but some also do not have a package they belong to.

56 - Finally, use apt full-upgrade to upgrade your Debian to full latest version (as root; first try as a regular user if you like). Upload a screenshot of the fully upgraded Debian instance (SCREENSHOT)



It did not upgrade anything, maybe because it is already 12.5. Maybe it would be 12.0, then it update it to 12.5

