Light Wizards Linux Guide

I was a Unix user before Linux came out, so I know about all the versions of Linux, and have tried out more Distributions over the years than I care to count, and this is just my opinion about Linux, so take it for what it is worth. If you want a stable version of Linux you need to go with a distribution that is based on a Stable repository, and to understand what a repository is, you need to know a few things, and the first thing is that Stable means old and tested.

Linux has an official site for every part of the system, the Kernel is the core of Linux, and it has its own Repository, and everyone pulls from those repositories, so you have Linux that is the same for all Distributions like the top 3: Arch Linux or Distributions based on it like my favorite Manjaro, and you have Distributions based on Debian like Ubuntu, and OpenSUSE, and the reason these distributions are so popular, is because they have stable repositories, but as much as I like Stable, I like New because I am a Computer Programmer, and I want the newest tools, like Qt 6.

If you are coming from Windows, this is your guide, I wrote it for someone who has been a Windows user all their life, and now are ready to jump in, so I will take you from the basics, to the more advance, and then into troubleshooting, and what to do when things break, so I have to pick a Desktop that is good for Windows Users, and Qt is Development Platform that KDE is based on, and who makes the Plasma Desktop, which is the most Windows like experience you will get in Linux.

KDE is short for Konqi Desktop Environment, Konqi is the KDE Mascon and is a Dragon, who lives with Kandalf, who is a Computer Science Wizard.

https://community.kde.org/Konqi

Command Line

I will talk about the command line first, the reason is simple, you need to document what you are installing, and the best way to do this is to open a file and write it all down. I will cover the top 3 Linux Distributions, the others are similar, but these 3 are all different, but almost all the others are based on these 3, Ubuntu is based on Debian, and are mostly the same, so that covers all the alternative Distributions. The command line is the first thing you need to know about Linux, and if you are coming from Windows, this will be the thing most people do not want to learn, but if you follow my simple advice, you will learn to be a Linux Wizard by learning how to use a Koncole which is a Terminal.

The Light version of being a Linux Wizard is to go to the command line first and ask questions later, you will need to know how to switch from one driver to another if you change hardware, and the first thing you want to learn is how to use Linux without losing your Data, so this is where we will begin.

Linux is setup very different from Windows, the first thing you need to know is that every Distribution has its own default Terminal, so let us talk about Terminals and Desktops. When you first boot up to Linux you will see a boot-up sequence, this starts with Grub which has a set time out limit you can change, this is a file that tells your Desktop loader which versions are available, as well as what Desktops, and versions might confuse you, but Linux has a Safe Mode, but unlike Windows, this is the last know version of the application that worked as far as it knows, so it is a different version, but your Data is the same, so do not get confused about what is going on, it will not load drivers, so hopefully you can get into the system and can do updates and fix what has gone wrong, so a Linux Wizard will go to the command line first, so let us do just that.

The boot up sequence of the Computer is first, this is the BIOS Screen, you might have to make a choice of which drive to boot, and those options can be UEFI or none UEFI, and you need to know if your BIOS is UEFI, but if you pick UEFI it will not blow up your computer if it does not support it, in fact, it checks it every time anyway, but picking no UEFI when you have it, will come with a performance hit.

The Grub screen may blow past so fast you never see it, you have a few seconds at most to hit a key sequence that will get you into Grub mode. Why you would want to get into Grub mode is as follows, let us say you try to make your own home or var folder by mounting it on a hard drive, having done this, I will talk about it, the only reason the computer will send you to a Grub screen is if there is an option to change something, this is where it reads in your /etc/fstab file that defines your hardware, it has the boot up information for your computer, it will give you two options if it finds any missing hardware, one is to hit Ctrl-D to ignore the errors, depending on the distribution, depends on what happens, OpenSUSE tends to just boot you out after log in, whereas Kubuntu will go back to the last Home folder, or the one stored on the same hard drive as the OS, the other option if you do not want to ignore it, or cannot ignore it, is to type in your root password, this will take you to a command line, that black box most Windows users never heard of nor used, but now you are in one, and let me explain it in detail, the entire screen is black, you have an arrow sign or some other symbol like > or ?, and that is it, if you have no idea

what you are going, your only option is man help, man is short for Manuel, and help is self-explanatory, but man help will more than likely scare you at first, so let us look at the first line:

bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see bash(1)

Now, bash is the language the Terminals is using, sh is a different type of language but is similar, in Unix you have sh or bash, you use scripts with sh extensions, and can call some files using sh, like: sh myProgram.sh or .run or with no extension at all, but it must have Execution permissions to run.

Alias is the first command you can type man alias, you can use arrow keys and any time you see a screen that says “more” or “less”, you are in man mode, or manual mode, and can use “Q” or “q”, without the quotes, and if you are in a thermal and everything scrolls off screen, you can use Ctrl-Page-Up and Ctrl-Page-Down keys. An alias is command that has another name, like ls, it is short for list, like most bash command, they use short names, as such, we have a file called .bash_aliases, which is called by a file called .bash_profile which calls .profile and which sets up your bash Environment, and that also calls .bashrc which calls .bash_aliases, and not to confuse you, but if you get here from Grub or Alt-Crlt-F6 or AC F6, where you can open a terminal using F2 through F6, but F7 is only used for GUI mode, and to make things confusing, you have to make sure the other files get called, so files are calling files that called it, just to make sure someone calls it, and that is because you do not know how they got there, for example, if you are logged into the GUI and open a Terminal you might not see any messages, yet if you AC F6 into a Terminal you see a lot of messages, and that is because of this circular pattern of decedent calls that change nothing, other than to make sure all the files get called, I mapped it out one time, but found it varies from one Terminal to the next, so an alias is defined in my .bash_aliases file that gets called, and these vary by who wrote them, but I like to include a colorized version of ls, as such if you want to know what ls you are running, run the command “which ls”, it might return “/usr/bin/ls”, if you run that command and see no color, you know that ls is an alias for “/usr/bin/ls”, the way an alias works is cool, you can filter the results of the real ls command, and change the color of the name depending on its file extension, and it you type “/usr/bin/ls -las”, you will get no color unless you have that alias file definition and if we ran “unalias ls”, and typed just ls, we would no longer get that color output, I will include that code in the Appendix.

If we made a mistake in your fstab file, we can go into Grub and put in our password, and run the command “nano /etc/fstab”, nano is a command line text editor, if you need line number run “nano -c /etc/fstab”, look at the bottom status area for the line number changes, Ctrl-o, as in oh, which reminds me of something few know, the first telephone alphabet on rotatory phones did not have an O or a Z because they get confused with 0 and 2, so you could not do this on a

phone, just thought you need to know that, you need a computer keyboard, Wizard humor, and Ctrl-x will exit, and x also means times, but only hit it once. Your arrow keys get you around, and you have insert and overwrite modes by hitting insert key, so this is the first thing any Linux Wizard should know, because if you get here due to a problem, you need to know how to fix it, and you cannot always depend on Kandalf to come to the rescue to put out the Fire the Dragon started.

The fstab file uses # or pound signs as comments, so anything after a # will not get ran, so if you run the command "# ls" you will not get any output, if you run "ls # echo hi", only ls will run and not hi. This file is in the etc folder for Et Cetera, and where a lot of configuration files are stored, and you need root permissions to access those files.

There are a lot of bash commands, and you can write whole programs using bash, if you do not have a function you need, you can write one, so bash is worth learning, and there are not a lot of commands you need to learn, you just need to know how to use them, and most commands are very intuitive, like exit, if you are logged in to any process exit will normally get you out of it, but this guide is not about learning all the bash commands, it is about knowing which ones you need to use to do some work you need to get done. In a terminal Ctrl-C will normally Cancel what you are doing, if that does not work after hitting 3 times in a row, your other options are Ctrl-D, which can also delete the line you are, and Ctrl-X, or Esc, that button in the upper left corner of most keyboards, is Escape, I wrote a bash program that could emulate human behavior, it was written in CPM, which is dead now, but you can build a series of menus with it, to make your job easier, I like to use bash scripts for installing my OS, backing up files and other things, and why I spend a lot of time in a Terminal, so let us look at bash.

Bash has logical operators like if, else, end if, they look like this:

```
if [ "$Condition" == "this" ]; then
        doThis;
fi
```

The fi is an end if, and with just that simple logic you can write an entire program, and it is a very power language, not much you cannot do with it in Linux or Unix, it is the most go to language I know of.

All distributions have a way to do updates, most will use a GUI as well as the command line, so let me talk about the command line first, in Debian systems, that is pronounce Deb-Ian, where Debbie was Ian's wife at the time, and Ian Murdoch, was Suicide by Police, and just one reason I do not like to use Debian, but it uses apt, or apt-get, or aptitude, and you must run it using Supper User privileges, so we use sudo, or Supper User Do this, you have to log in using

the root users password, in Linux you have one root user, this password can be the same as yours, but if you log in with both user name and password, you need to use root as the user name, or su which gives you a supper user terminal, and then you can drop having to use sudo, to get out of su mode, type exit, if you exit again it might close the terminal, or a tab in that terminal.

In Debian and Ubuntu and all Distributions based on them, the basic commands are as follows:

sudo apt update && sudo apt upgrade -y

Note the -y is to accept all the defaults, first it runs update which will update all your repositories, than it will look for upgrade's in the newly downloaded repositories, and the && means only do this if the first one completes without any errors, if you had a single & it would execute both regardless of if the first one passed, which is the same as just putting a ; at the end to end that line of commands.

OpenSUSE (Open soo'-suh)

sudo zypper refresh; sudo zypper update -y

Distribution-Upgrade is used to upgrade your Distribution from one version to another, but if you are running Ubuntu in LTS mode, you do not want to do a Distribution-Upgrade, but in a Rolling Distribution you need to, and this is how:

sudo zypper dist-upgrade -y;

OpenSUSE has Tumbleweed which is a Rolling Distribution, and they have Leap, which is a stable scheduled released version.

Whenever you do something that crashes your system or makes it behave in a way you do not like, you wish you could "Turn Back-Time", as Tiny Turner would put it, back to how it was running yesterday, and you can if you install timeshift.

In Ubuntu

sudo apt install -y timeshift

In OpenSUSE

sudo zypper install -y timeshift

You want to run timeshift once to set up the location for where you will save the files, if you have a spare drive that has space, its best to use it, these files are large and you can set how many versions you want to save, so how far you want to "Turn back time", the default is 5, I set it to 3, and put it on my home drive that I have as a separate RAID 1 drive. I like to create the first snapshot to see how long it takes, it will create a new image each day, and you can check you System Monitor to see if it is running if you are wondering why your computer is acting sluggish, this is normal if you have it enabled, in fact you cannot avoid having to do it if you want to use it, but I have found if I leave my computer running, it will always do it at a time I have nothing going on.

Timeshift only stores you OS and var files and not your Data Files in home, so if you want to track changes in files, that is called a backup system, so do not confuse what Timeshift does with a data backup, all it can restore is your configuration files, and key OS files, it stores this data under a user name for timeshift, but if you have stuff in var, that can have mixed outcomes, and why you want to keep var on a separate drive.

Since Timeshift will also store your var folder, so if you have websites or Virtual Machines, this will also get backed, so be aware of this if you want to restore, since it can overwrite your VM or Websites with old data, I am not sure it will do this, never tired it, but I would hope it would know this, but you have to back up your VM to ensure you have a backup plan. You must manually run Timeshift the first time, and you have a command line version that can restore a snapshot.

sudo timeshift --restore

If you have not figured it out, a lot of times the package names are the same, but each distribution has its own name for its package manager:

Ubuntu, Debian

apt, or apt-get, or aptitude

OpenSUSE

zypper

Arch Linux, Manjaro

pacman, yay for AUR

Fedora

yum or dfn

If you want to track your changes make a file and call it something you will remember, like install-distro-Name.sh, and put all the files you are installing in it before you install them, save the file to USB device in case things go south, and if you store them in a way that you can just run the file, it will install everything just like you had it, this makes it better for you to track your software and make sure it gets installed if you want to change distributions, or upgrade and want to make sure that if anything goes wrong you can reinstall it, and save it on a few drives in case one goes bad.

I just scratched the surface of a Terminal, but it is the least you need to know to become a Linux Wizard, you must also know about Configuration files, and how to get around in Linux.

"System-Setting" is our first go to when it comes to Configuration files.

Dolphin is KDE's default File Explore, and the first time I used it, I hated it, but after I leaned out to use it, I only use it, but not everything will work out of the box, like Write to USB, you might find your Distribution does have imagewriter, which is Suze, or isoimagewriter which is Red-Hat, but there is a way to get them to work, but that is what Scripts are for. Setting up Dolphin is the first thing to do, I like to make a few changes to the settings, first click on the upper right corner … symbol, or right click to find the Show Menu, and then click on Settings, ensure Show Menu Bar and Show Toolbar are checked, then click on configure Toolbar, and add "Up", and "Refresh", and "New Tab", "Show Hidden", and "Terminal", do this by going to the top menu items in the list, and placing your cursor just above the top separator on the last item

and select the item in the list on the left to move it to the right and click on the arrow button to add it to the list, it is drag and drop and click ok.

You can right click and select show only Icons, the default is text beside icon, and you can configure Dolphin to your liking. You can hit Ctrl-L, it is a lower case L, but l, 1 and I all look the same to me, but not L, so why didn't the phone company remove 1 or L, and if you do not get the Joke, it's because you have never had to use a Phone to log into Linux, I have, no Joke, but it was a push button with the same limitations as the rotatory, so not a huge improvement, considering the rotatory phone will work without power, it has to have line power, but no external power like most digital phones, and why you will find a PBX in Linux, and Telegram that can also use your phone line, Linux is an On-Line Application for many around the world, and why a URL while pull up a web browser, so to get an editable version of the path hit Ctrl-L, so L for Editable Line, I use this feature to copy a path, you can click on the tab parts to move around the folder structure.

In Dolphin we have what is known as Places and Bookmarks, click on Home in the left upper corner, and you should see a Music, Video, and maybe some other folder that is not in Places, but you want it to be, if you see a Music or Video folder under Home that is not under places, then double click on Music or Video to enter it, right click and you should see Add "Music" or "Video" to Places, click on that option, now drag Music up above Trash and let it go, this is how you add Bookmarks to Places, and move them around to places you want them.

KDE Taskbar is something you need to learn, if you want your Taskbar on the Top or bottom, or opposite of what it is now, you right click on it and pick Edit Panel or Enter Edit Mode, depending on your Distribution, now you have many options, one is to drag the screen edge to where you want it and release, another is to add a Widget to your Taskbar, you have many to pick from, I like Weather, but let us add a new one, say Comic Strip if you have a Debian based distro that still runs Qt 4 apps, so right click on the Taskbar and hit the Add Widget button and a Window on the left should have a list of Widgets, we want the Comic Strip, or Weather, they all work the same way, so double click on it, and once you are done, just click outside the Taskbar to finish editing, if you have an extra panel, it will normally have a red x to close it, now you should have a Happy Face or an image of the Weather outside on your Taskbar, right click on it and pick Configure Comic Strip… Click on Get New Comic, click on install next to the ones you like, I like Dilbert and NASA Picture of the Day, now close, than check all of them, set the time, and other things, now it's ready, just click on it, and you will see tabs with Comics Strips, if they do not all show up just give it time, if it never shows up, you can report it, but it is no longer maintained due to lack of interest porting it over to Qt 5, so just remove it, it has controls at the bottom of each image, and forward and backward and expand to make it bigger to read, and that is how you add Widgets, if this is a Weather app, pick your Location, and how you want it, F or C.

If you want to add a CPU Temperature gauge, you need to Add Widget, Get New Widget, type in CPU, and find one, I picked Thermal Monitor, now this App uses a Signal from the ACPI or Advanced Configuration and Power Interface, has two options, source or grouped, source uses the whole group, whereas group allows you to uncheck some features. Note that if

for some reason you system does not send out a request, and normally this is only required if you have a UPS, so what I do if I see the word “off”, is to right click and “Reload Temperature Sources”, and wait, if you want this to fix itself, you need to run a command to enable ACPI, first we need to see what we have installed:

ls -l /usr/lib/modules/$(uname -r)/kernel/drivers/acpi

If you see acpi_configfs.ko.xz, or something like it, you should have the Configuration in place, so reloading should work, but somewhat annoying, so run this:

sudo dmesg | grep -i acpi

If that confuses you run this:

acpi -i

Joking, look for your UPS, if you do not have one, do not worry, the last command to acpi for information should show you what services are activated, If you do not have a UPS set up, you will see something like this:

Battery 0: Discharging, 0%, rate information unavailable

Otherwise, you see the status of your UPS.

Now run

sensors-detect

And read the instructions, pick yes for all those in YES, and no for all in NO, until you get to the prompt that says something like:

Lastly, we can probe the I2C/SMBus adapters for connected hardware monitoring devices. This is the riskiest part, and while it works reasonably well on most systems, it has been reported to cause trouble on some systems.

Do you want to probe the I2C/SMBus adapters now? (YES/no): yes

Using driver `i2c-i801' for device 0000:00:1f.3: Intel Panther Point (PCH)

Module i2c-dev loaded successfully.

Say yes for all of these.

Now type in:

sensors

You should see all your Sensor data, and with any luck you will not have to hit reload again, still looking into this issue, because that should have set it up to run Automatically, and you must do this if you change Motherboards, or any hardware that uses ACPI.

The Taskbar

The Taskbar in KDE Plasma 5 is nice, not that 4 wasn't nice, but it did not upgrade to 5 nice, so it made KDE unstable for a time, and you must watch adding Widgets from unknown authors, these can run with the same permissions you have, and somethings required root, so know which ones do, read about them if you are not sure.

You can drag items around on the Taskbar in Edit mode, you can make it bigger, change the Time format, and if you want to see your Workspace Switcher, you need to define at least 2 Workspaces, to check click on the Menu Icon on the Taskbar Menu and pick System Settings that is normally under Favorites or Settings, then pick Workspace Behavior, and then Virtual Desktop and make sure you have at least two, if you only have one pick Add, and apply, and you should now see two Work-spaces in your Taskbar, I like to use Cube, if you want to turn off Screen blanking pick Screen Locking.

I like to have a clean Desktop, and Clean Taskbar, I do not like to Pin items to my Main Taskbar, but I like to Pin tabs in my browser, and it's because in Browser I know it's loaded, if you pin an icon to the Taskbar you might activate it trying see what is in it, and really you have a Menu, it has Favorites, what else do you need, and the answer is nothing, learn to use the system the way it was intended to be used.

If you right click on the Menu Icon, you have a few options, one is to Configure Applications Launcher, where Configure the way the Menu looks, say you want to have Favorites organized by List instead of Grid, or Edit Applications, where you edit the menu itself.

KDE organizes its Menu System by Category, you have several levels you can create, some menus will open up one level deep on hover, while levels greater than 1 required a click to open them, and you can alter that behavior in System Settings.

You can search for a program by clicking on the menu and type the name, and then you can right click and add to Favorites.

System Settings

System Settings is an Icon or Menu Item, you can normally find it under Favorites as well as System or Settings depending on your Distribution.

There are too many things going on in this one Panel to cover everything, so instead I only want to tell you about using some of the features, Appearance is the main panel for configuring how KDE looks, you must hit Apply to see the effect, so try to only change one thing at a time and remember what it was before you changed it.

I like to use Light Themes, and I like to set the Application Style to Fusion or Oxygen, and I like the Plastik Window Decoration, and I like to set my Fonts at least one higher, maybe 2, and set the Workspace Behavior clicking on files or folders to select.

Desktop effects are nice, I like Fall Apart, it causes your windows to explode when you close them, and I like Wobbly Windows, this makes the Windows Wobble when you move them around, and set Window Management to Cube, now when you click on the Desktop Switcher, you will see it move like a Cube, and I like to set the Task Switcher under Windows Management, under visualization set it to Cover Switch.

You change you resolution from the System Settings, as well a many other things, so check them all out, and see what they do.

Using Linux for your Desktop

You have Firefox for Browsing the Internet, and you have many Email Clients like Evolution and Thunderbird, and you have Open Office, where you have Writer instead of Word, and most of the other Office Applications.

You can look through your menu and find Applications to run, in some distributions you must pick Leave to quit, so it is all a matter of learning how to use the Menus, and what Programs do what.

Alt-Tab is used to move from one application to the next using the Task Manager, and it is normal for the menu to freeze up at times while it is doing something, so you must be patent at times. Alt-Ctrl-Del, or Alt-Ctrl-Backspace are used if it seems locked up, and you can hit Alt-Ctrl-F6 and log into a Terminal and type:

sudo reboot

You can run some Windows Applications in Linux using WINE, Windows Integration Not Emulation, and you can install them like you do in Windows, you can normally double click, but if that does not work right click and pick to open with the Wine Launcher.

Most distributions will let you know when an Upgrade is available, some updates required a reboot, this is more of a KDE thing than a Linux thing, but it is a safer way to upgrade. If you are running Ubuntu or Kubuntu LTS, you only want to do upgrades, and not distro-upgrades, if you do a distro-upgrade you will using the latest version of Ubuntu or Kubuntu, and will have newer versions of software than the LTS, which gets old after its 4 year cycle, so you have to decide if you want the latest, or the most Stable, and Stable means Old and Tested.

Linux does not get Viruses of Malware like Windows does, but that does not mean you are safe from everything, if you run Windows Apps in Linux, they can get a virus. You can run Nod-32 in Linux, as well as others, but for the most part, you do not need to worry about it.

Troubleshooting

When things go wrong, you must know what to do, and that is not always the same thing, you have all types of problems in Linux, and there is not a fix it button, and you only need to know a few things to get help fixing it, one is the Forums, the other is the Internet, so let us look at how to get help.

The first thing you need to do is upgrade, in Ubuntu or Kubuntu, they are the same for the most part, only thing that is different is the Desktop you are using, Linux is the Kernel, all Linux Distributions use the same Linux Kernel, that is what make them Linux, and the Kernel is the OS, the Desktop just interacts with it in a GUI.

sudo apt update && sudo apt upgrade

In Arch Linux you use:

sudo pacman -Syu

If upgrading and rebooting does not solve you issue, you must identify the issue, for example, do you get video, if not it might be a driver issue, you must understand what the malfunction is to know how to ask for help. If you run the two below commands, it will create two log files that anyone that knows about Linux would want to see in order to help you.

sudo dmesg > dmesg.log

sudo journalctl > journalctl.log

Sometimes you need to clean you logs before running the above, and then reboot, this is how you clean you log files:

sudo journalctl --vacuum-time=1s && sudo journalctl --vacuum-size=1K

sudo systemctl restart systemd-journald && sudo journalctl -u systemd-resolved

sudo reboot

Then run two commands to make the log files.

Appendix Kubuntu

This is the Installation script for Kubuntu:

```
#!/bin/bash
#
# Copyleft and Written by Jeffrey Scott Flesher
# No Copyrights or Licenses
#
# shell?check -x install-kubuntu.sh
#
# Install Script for Kubuntu KDE
# "${HOME}"/Scripts/install-kubuntu.sh -w "${TheWorkSpace}/workspace/" -f


# W: Possible missing firmware /lib/firmware/amdgpu/vangogh_gpu_info.bin for module
amdgpu
# W: Possible missing firmware /lib/firmware/amdgpu/sienna_cichlid_mes.bin for module
amdgpu
# W: Possible missing firmware /lib/firmware/amdgpu/navi10_mes.bin for module amdgpu
#
stat /dev/ttyS0;
declare DateManager="24 Oct 2021";
declare VersionManager="1.0";
declare -x TheScriptName; TheScriptName="install-kubuntu";
declare -i QT_ONLINE; QT_ONLINE=1;
declare EMAIL_ADDRESS_NOTIFICATIONS;
EMAIL_ADDRESS_NOTIFICATIONS="jeffrey.scott.flesher@gmail.com";
declare HOME_UUID; HOME_UUID="UUID=d0b69f0b-782b-421a-ae17-2cedf05ec481
/home                btrfs  defaults                  0  0";
# shellcheck disable=SC1090
[[ -f "${HOME}/.bash_aliaeses" ]] && source "${HOME}/.bash_aliaeses";
#
################################################################################

# $HOME/.config/systemd/user/ssh-agent.service
#[Unit]
#Description=SSH key agent

#[Service]
#Type=simple
#Environment=SSH_AUTH_SOCK=%t/ssh-agent.socket
#ExecStart=/usr/bin/ssh-agent -D -a $SSH_AUTH_SOCK

#[Install]
#WantedBy=default.target
```

```
#cp /etc/xdg/autostart/gnome-keyring-ssh.desktop ~/.config/autostart/
#cp /etc/xdg/autostart/gnome-keyring-pkcs11.desktop ~/.config/autostart/
#cp /etc/xdg/autostart/gnome-keyring-secrets.desktop ~/.config/autostart/
#
################################################################################
# Table of Content Index
# checkArgs
#
################################################################################
declare -x ThisRoofFolder; ThisRoofFolder="${HOME}";
if [ ! -d "${ThisRoofFolder}" ]; then
   mkdir -p "${ThisRoofFolder}";
fi
declare -x TheWorkSpace; TheWorkSpace="${ThisRoofFolder}/workspace/";
declare TheSambaPublicFolder; TheSambaPublicFolder="${ThisRoofFolder}/samba/public";
declare TheWindowsFolder; TheWindowsFolder="${ThisRoofFolder}/Windows";
#
declare -x TheWizardPath; TheWizardPath="${HOME}/Scripts"; export TheWizardPath;
declare ThisUnrealEngineGitFolder;
ThisUnrealEngineGitFolder="${ThisRoofFolder}/workspace/UnrealEngine";
declare ThisUnrealEngineLauncherGitFolder;
ThisUnrealEngineLauncherGitFolder="${ThisRoofFolder}/workspace/UE4Launcher";

declare TheCOW; TheCOW="${TheWindowsFolder}/COW"
#declare -i ThisComputerNvidia; # FIXME command line
#ThisComputerNvidia=1; # 1 = yes
#declare ThisOwner; ThisOwner="$(whoami)";
declare -ix ThisClearOk; ThisClearOk=1; export ThisClearOk;
# KVM
declare TheKvmImageFolder; TheKvmImageFolder="/var/lib/libvirt/images";
declare -i ThisCowBackUp; ThisCowBackUp=0;
declare -i thisPatch; thisPatch=0;
#
################################################################################
#
# For Test:
# cd "${HOME}/Downloads/Test.install-home/"
# cd /mnt/BaseFolder
# chmod +x "${HOME}"/Scripts/install-kubuntu.sh
# dos2unix "${HOME}"/Scripts/install-kubuntu.sh
# clear; echo "Shell Check..."; shell?check "${HOME}"/Scripts/install-kubuntu.sh
# shell?check -x install-kubuntu.sh
# shell?check -x ".${TheWorkSpace}/install-kubuntu.sh && { shell?check -x
".${TheWorkSpace}/wizard.sh"; }  && { shell?check -x ".${TheWorkSpace}/wizard-
common.sh"; }
```

```
# "${HOME}"/Scripts/install-kubuntu.sh -w "${HOME}/workspace/"
# New Install
# ./install-kubuntu.sh -f
# KVM Backup
# "${HOME}"/Scripts/install-kubuntu.sh -w "${TheWorkSpace}/workspace/" -k
# Malware
# "${HOME}"/Scripts/install-kubuntu.sh -w "${TheWorkSpace}/workspace/" -m 2
#
################################################################################
#
declare ThisRunaptUpdate; ThisRunaptUpdate=0;
#
set -u; # same as  set -o nounset, error if variable is not set
# nocaseglob: If set, Bash matches filenames in a case-insensitive fashion when performing
filename expansion.
# dotglob: If set, Bash includes filenames beginning with a `.' in the results of filename
expansion.
# -s   Enable (set) each optname
shopt -s nullglob dotglob;
IFS=$' '; # IFS=$'\n\t'; IFS=$'\n';
# User's file creation mask. umask sets an environment variable which automatically sets file
permissions on newly created files.
#   i.e. it will set the shell process's file creation mask to mode.
umask 022;
unalias -a; # -a   Remove All aliases; so cp is not cp -i
# -f The names refer to shell Functions, and the function definition is removed.
#   Readonly variables and functions may not be unset
unset -f "$(declare -F | sed "s/^declare -f //")";
#
#trap "echo Exited!; exit;" SIGINT SIGTERM;
trap "echo install-home Control Break; exit;"     SIGINT;
trap "echo install-home SIGTERM-d; exit;"         SIGTERM;
trap "echo install-home Exited with Error; exit;" EXIT;
trap "echo install-home Finished normally; exit;" 0;
#
#
# all Public Variables
declare TheFullScriptPath; TheFullScriptPath="$(dirname "$(readlink -f "${0}")")"; export
TheFullScriptPath; # No Ending /
declare -i SimulateThis;   SimulateThis=0; export SimulateThis;      # 1=true or 0=false, Simulate
rename
declare -i PrintDebug;     PrintDebug=0;   export PrintDebug;        # 1=true or 0=false
declare -i SetDebug;       SetDebug=0;     export SetDebug;          # 1=true or 0=false
#
declare -ix TheDebugging; TheDebugging=0;  export TheDebugging;
#
```

```
#
################################################################################
#
# Where do I put these files in Misc
declare TheLocalizedPathFolderName;  TheLocalizedPathFolderName="locale"; export
TheLocalizedPathFolderName;      # No slash at end
declare -x TheLocalizedPath;
TheLocalizedPath="${TheFullScriptPath}/${TheLocalizedPathFolderName}"; export
TheLocalizedPath;
declare -x TheDefaultLanguage;          TheDefaultLanguage="en";                # I wrote this in
English so this is a constant
# Where the po.langage file goes -> Edit this line as needed for project
# Name to call the Langage File referanced above -> Edit this line as needed for project
declare -x TheLocalizedFile; TheLocalizedFile="$TheScriptName";
# Set to 1 the first time you start Hand Translating your Localization files so they do not get
overwriten.
declare -ix TheLocalizedFilesSafe; TheLocalizedFilesSafe=1; export TheLocalizedFilesSafe;
# Change this to the Default Language that the Localized files are in
declare -x TheDefaultLocalizedLanguage; TheDefaultLocalizedLanguage='en'; export
TheDefaultLocalizedLanguage;
#
# Localization of all script tags to support language
declare -ix TheRunLocalizer; TheRunLocalizer=0; export TheRunLocalizer;
# Multilingual Langage File Path -> from above: declare -r
TheLocalizedPath="${TheFullScriptPath}/locale"
TEXTDOMAINDIR="${TheLocalizedPath}";
export TEXTDOMAINDIR;
# Multilingual Langage File Name -> from above: declare TheLocalizedFile
TEXTDOMAIN="${TheLocalizedFile}";
export TEXTDOMAIN;
# Create Help.html
declare -ix TheRunHelp; TheRunHelp=0; export TheRunHelp;
# 0=Disable, 1=Run, 2=Run Extended Test
declare -ix TheRunTest; TheRunTest=0; export TheRunTest;
# Automatically install from saved settings
declare -ix TheAutoMan; TheAutoMan=0; export TheAutoMan;
# 14-Jan-2013 @ 06:32:36 PM
# shellcheck disable=SC2034
declare -x TheDateTime; TheDateTime="$(date +"%d-%b-%Y @ %r")"; export TheDateTime;
# Day-Mon-YYYY-T-HH-MM: 14-Jan-2013-T-18-32-36
# shellcheck disable=SC2034
declare -x TheLogDateTime; TheLogDateTime=$(date +"%d-%b-%Y-T-%H-%M-%S"); export
TheLogDateTime;
#
################################################################################
#
```

```
#
################################################################################
#
# change to project needs
declare -x TheConfigName;  TheConfigName="install-home";                    export
TheConfigName;
declare -x TheLogPath;     TheLogPath="${TheFullScriptPath}/Support/Logs";          export
TheLogPath;
declare -x TheConfigPath;  TheConfigPath="${TheFullScriptPath}/Support/Config";
export TheConfigPath;
declare -x TheErrorLog;    TheErrorLog="${TheLogPath}/0-${TheConfigName}-error.log";
export TheErrorLog;
declare -x TheActivityLog; TheActivityLog="${TheLogPath}/1-${TheConfigName}-
activity.log"; export TheActivityLog;
#
################################################################################
makeHome()
{
   if [ -f "fstab.txt" ]; then
      if [ ! -f "/var/log/wizardscript/fstab.txt" ]; then
         sudo mkdir "/var/log/wizardscript";
         sudo touch "/var/log/wizardscript/fstab.txt";
         sudoAppend "fstab.txt" /etc/fstab;
         sudo mount -a;
         echo "Reboot required";
      fi
   fi
}
#
################################################################################
# show_help (0=Text, 1=HTML)
show_help()
{
   [[ $# -ne 1 ]] && { echo "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   getHorizontalLineType "$1";
   fixLineType "$1" "Help";
   fixLineType "$1" "";
   fixLineType "$1" "Expected Argument Switches:";
   fixLineType "$1" "-h or --help";
   fixLineType "$1" "-l or --localize";
   fixLineType "$1" "-p or --printdebug";
   fixLineType "$1" "-s or --simulate";
   fixLineType "$1" "-t or --test";
   fixLineType "$1" "-v or --version";
```

```
   fixLineType "$1" "";
   fixLineType "$1" "Tests";
   fixLineType "$1" "Test 1: all common functions";
   fixLineType "$1" "Test 2: Internet and user account functions";
   fixLineType "$1" "Test 3: Keyboard Input functions";
   fixLineType "$1" "Test 4: Select File function";
   fixLineType "$1" "Test 6: Network and Group functions 5: Keyboard Input Options";
   fixLineType "$1" "Test 7: SQLite";
   fixLineType "$1" "";
}
# END show_help
#
################################################################################
#
declare -i IsBeeper;          IsBeeper=0;          # Beep after each encoding
declare -i TheRunCheck;       TheRunCheck=0;
declare -i TheRunFixNPM;      TheRunFixNPM=0;
declare -i TheRunFirst;       TheRunFirst=0;
declare -i TheRunFixSTO;      TheRunFixSTO=0;
declare -i TheRunSetXclip;    TheRunSetXclip=0;
declare -i TheRunScan;        TheRunScan=0;
declare -i TheScanLevelAV;    TheScanLevelAV=0;
declare -i TheKvmFixBackup;   TheKvmFixBackup=0;
#
################################################################################
declare -i DoNotRun; DoNotRun=0;
#
declare -i RUN_OPTIONS; RUN_OPTIONS=0;
declare -x PROGNAME;    PROGNAME="${0##*/}";     #
#                               b d e g i j q u r x z
declare -x SHORTOPTS;   SHORTOPTS="v,h,p,t:,s:,b,l,w:,a,c,n,f,o,x,m:,y,k";
declare -x LONGOPTS;
LONGOPTS="version,help,printdebug,test,simulate,beep,localize,workspace,annotation,clean,n
pmfix,first,outtimer,xclip,malavscan,yelp,kvmfixbackup";
declare -x ARGS;        ARGS=$(getopt -s bash --options "${SHORTOPTS}" --longoptions
"${LONGOPTS}" --name "${PROGNAME}" -- "$@");
eval set -- "$ARGS";
#
################################################################################
# checkArgs "$@"
checkArgs()
{
   #
         while true; do
            case $1 in
               -v|--version)
```

```
            echo "Script Date: ${DateManager} Version: ${VersionManager}";
            exit 0;
            ;;
        -h|--help)
            usage;
            ;;
        -a|--annotation)
            TheRunHelp=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "TheRunHelp"; fi
            ;;
        -p|--printdebug)
            PrintDebug=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "printdebug"; fi
            ;;
        -t|--test)
            shift;
            TheRunTest=$((${1} + 0));
            [ ${TheRunTest} -lt 1 ] || [ ${TheRunTest} -gt 10 ] && usage;
            echo "Run Test: ${1}";
            ;;
        -s|--simulate)
            shift;
            SimulateThis="${1}"; # 0=False, 1=True
            if [ "${PrintDebug}" -eq 1 ]; then echo "simulate=${1}"; fi
            ;;
        -b|--beep)
            IsBeeper=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "IsBeeper"; fi
            ;;
        -l|--localize)
            TheRunLocalizer=1;
            if [ ! -d "$TheLocalizedPath" ]; then mkdir -p "$TheLocalizedPath"; fi
            if [ "${PrintDebug}" -eq 1 ]; then echo "localize"; fi
            ;;
        -w|--workspace)
            shift;
            TheWorkSpace="${1}"; #
        #
        if [[ "$TheWorkSpace" != */ ]]; then TheWorkSpace="${TheWorkSpace}/"; fi # Make
sure it ends with a /
        #
            if [ "${PrintDebug}" -eq 1 ]; then echo "workspace=${1}"; fi
            ;;
        -c|--clean)
            theClearner=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "clean"; fi
```

```
                ;;
            -n|--npmfix)
                TheRunFixNPM=1;
                fixNpm;
                if [ "${PrintDebug}" -eq 1 ]; then echo "npmfix"; fi
                ;;
            -f|--first)
                TheRunFirst=1;
                if [ "${PrintDebug}" -eq 1 ]; then echo "Run First-time"; fi
                ;;
            -o|--outtimer)
                TheRunFixSTO=1;
            fixSudoTimeOut;
                if [ "${PrintDebug}" -eq 1 ]; then echo "outtimer"; fi
                ;;
            -x|--xclip)
                TheRunSetXclip=1;
            setXclip;
                if [ "${PrintDebug}" -eq 1 ]; then echo "xclip"; fi
                ;;
            -m|--malavscan)
                shift;
                TheRunScan=1;
                TheScanLevelAV=$((${1} + 0));
                [ ${TheScanLevelAV} -lt 1 ] || [ ${TheScanLevelAV} -gt 10 ] && usage;
                if [ "${PrintDebug}" -eq 1 ]; then echo "malavscan"; fi
                ;;
            -y|--yelp)
                echo "sudo required to run timeshift restore, this will reset the the OS back in
time.";
                read_input_yn_c "Run timeshift restore" "" 1;
            if [ "${YN_OPTION}" -eq 1 ]; then
                sudo timeshift --restore; sudo reboot;
              fi
              #
                if [ "${PrintDebug}" -eq 1 ]; then echo "yelp"; fi
                ;;
            -k|--kvmfixbackup)
                TheKvmFixBackup=1;
                if [ "${PrintDebug}" -eq 1 ]; then echo "kvmfixbackup"; fi
                ;;
            --) shift; break; ;;
             *) shift; break; ;;
            esac; shift;
        done
    #
```

```
   if [ "$TheWorkSpace" == "" ]; then
      echo;
      echo "TheWorkSpace=$TheWorkSpace";
      echo;
      usage;
   else
      RUN_OPTIONS=1;
   fi
   #
   [[ "$TheKvmFixBackup" -eq 1 ]] && DoNotRun=0;
   [[ "$TheRunFirst" -eq 1 ]]     && DoNotRun=0;
   [[ "$TheRunTest" -eq 1 ]]      && DoNotRun=0;
   [[ "$TheRunLocalizer" -eq 1 ]] && DoNotRun=0;
   [[ "$TheRunHelp" -eq 1 ]]      && DoNotRun=0;
   [[ "$TheRunCheck" -eq 1 ]]     && DoNotRun=0;
   [[ "$TheRunFixNPM" -eq 1 ]]    && DoNotRun=0;
   [[ "$TheRunFixSTO" -eq 1 ]]    && DoNotRun=0;
   [[ "$TheRunSetXclip" -eq 1 ]]  && DoNotRun=0;
   [[ "$TheRunScan" -eq 1 ]]      && DoNotRun=0;
   #
   if [ "$DoNotRun" -eq 0 ]; then
      printDoNotRun "TheKvmFixBackup" "$TheKvmFixBackup";
      printDoNotRun "TheRunFirst" "$TheRunFirst";
      printDoNotRun "TheRunTest" "$TheRunTest";
      printDoNotRun "TheRunLocalizer" "$TheRunLocalizer";
      printDoNotRun "TheRunCheck" "$TheRunCheck";
      printDoNotRun "TheRunFixNPM" "$TheRunFixNPM";
      printDoNotRun "TheRunFixSTO" "$TheRunFixSTO";
      printDoNotRun "TheRunSetXclip" "$TheRunSetXclip";
      printDoNotRun "TheRunScan" "$TheRunScan";
   fi
}
# END checkArgs
#
################################################################################
# useage
usage() { show_help 0; exit 1; }
#
################################################################################
# printDoNotRun "Var-Name" "Value";
printDoNotRun()
{
   if [ "${2}" -eq 1 ]; then
      echo "${1}=${2}";
   fi
}
```

```
# END printDoNotRun
#
################################################################################
cleanKDE()
{
   echo "This will delete all you configuration files for KDE";
   pause_function "${FUNCNAME[0]} @ $(basename "${BASH_SOURCE[0]}") :
${LINENO[0]} ";
   rm -rf .kde/
   rm -f .kderc
   rm -rf .config/*plasma*
   rm -rf .config/*kde*
   rm -rf .config/*kwin*
   rm -rf .local/share/kded5
   rm -rf .config/session/kwin_*
   rm -rf .gtkrc-2.0*
   rm -rf .cache/upstart/startkde.log*
   rm -f .xsession-errors
}
#
################################################################################
distroUpgrade()
{
   lsb_release -a;
   sudo apt update -y && sudo apt upgrade -y && sudo apt autoremove -y;
   sudo apt install -y update-manager-core ubuntu-release-upgrader-core
   # Check for Upgrade
   sudo update-manager -c
   # -d option to upgrade to the development release:
   sudo do-release-upgrade -d
   #sudo update-manager -c -d
   sudo apt dist-upgrade -y
   #sudo do-release-upgrade
   lsb_release -a;
}
#
################################################################################
refeshUpdateUpgrade()
{
   print_caution "Software Updates" "pacman setting coutry" "@ ${FUNCNAME[0]}() :
${LINENO[0]}";
   # sudo apt dist-upgrade -y
   sudo apt update -y && sudo apt upgrade -y && sudo apt autoremove -y
   # && sudo apt dist-upgrade -y
   # Fix all
   #sudo apt dist-upgrade --allow-downgrades
```

```
    #sudo apt autoremove -y
    #
    ThisRunaptUpdate=1;
}
# END refeshUpdateUpgrade
#
################################################################################
# FIXME
setXclip()
{
    print_caution "Set xclip alias" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo apt install -y xclip;
    if [ ! -f ~/.bash_aliases ]; then
        touch ~/.bash_aliase;
        echo '#!/bin/bash' > ~/.bash_aliase;
        echo 'alias setclip="xclip -selection c";' > ~/.bash_aliase;
        echo 'alias getclip="xclip -selection c -o";' >> ~/.bash_aliase;
    fi
    alias setclip="xclip -selection c";
    alias getclip="xclip -selection c -o";
}
# END setXclip
#
################################################################################
installRust()
{
    if cd "${HOME}"; then
        curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh;
        # shellcheck disable=SC1090
        source "$HOME"/.cargo/env;
    fi
    rustup set profile default;
    rustup toolchain install stable;
    rustup default stable;
    rustup update;
    rustup check;
    cargo init .
    cargo update;
    rustc --version;
    rustup show;
    rustup self update;
}
#
################################################################################
# https://emscripten.org/docs/getting_started/downloads.html
installWasm()
```

```
{
   npm install -g assemblyscript;
   npm install -g http-server;
   if cd "$HOME"; then
      git clone https://github.com/emscripten-core/emsdk.git
   fi
   if cd emsdk; then
      git pull
      ./emsdk install latest
      ./emsdk activate latest
   fi
   mkdir -p "${HOME}/workspace"
   if cd "${HOME}/workspace"; then
      # https://code.qt.io/cgit/qt/qt5.git/
      if [ -f "${HOME}/workspace/qt5" ]; then
         rm -rf "${HOME}/workspace/qt5";
      fi
      git clone https://code.qt.io/cgit/qt/qt5.git -b 5.15.2;
   fi
   #git clone https://code.qt.io/qt/qt5.git -b 5.15.2;
   if cd "$HOME/workspace/qt5"; then
      ./init-repository -f --module-subset=qtbase,qtcharts,qtsvg;
   fi
   if cd ..; then
      if [ -f "${HOME}/workspace/qt5_shadow" ]; then
         rm -rf "${HOME}/workspace/qt5_shadow";
      fi
      mkdir qt5_shadow;
   fi
   if cd qt5_shadow; then
      ../qt5/configure -opensource -confirm-license -xplatform wasm-emscripten -feature-thread -
nomake examples -no-dbus -no-ssl -prefix "$PWD"/../qt5_wasm_binaries;
      make module-qtbase module-qtsvg module-qtcharts -j8;
      sudo make install -j8;
   fi
}
#
################################################################################
updateFonts()
{
   print_caution "Update Fonts" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo fc-cache -vr
}
# END updateFonts
#
################################################################################
```

```
installFonts()
{
   print_caution "Install Fonts" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # Fonts

   sudo apt install -y fontconfig gsfonts gsfonts-x11;
   sudo apt install -y ttf-ubuntu-font-family ttf-bitstream-vera ttf-anonymous-pro ttf-mscorefonts-
installer ttf-bitstream-vera;
   updateFonts;
}
# END installFonts
#
################################################################################
#
installSagemath()
{
   print_caution "Install sagemath" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo apt install -y sagemath;
   sudo apt install -y octave;
   sudo apt install -y scilab;
   #sudo apt-key --recv-keys AE5A7FB608A0221C;
}
# END installSagemath
#
################################################################################
# Adding user '$USER' to user-group 'sambashare'
installSamba()
{
   print_caution "Install Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo apt install -y samba smbclient;
   configSamba;
}
# END installSamba
#
################################################################################
# 10.0.2.4 by default
# Samba name server XEOFF is now a local master browser for workgroup WORKGROUP on
subnet 192.168.122.1
# Samba name server XEOFF is now a local master browser for workgroup WORKGROUP on
subnet 192.168.1.13
# 192.168.122.1 255.255.255.0
# <range start="192.168.122.2" end="192.168.122.254"/>
configSamba()
{
   print_caution "Configure Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # Samba
```

```
    # TheSambaPublicFolder="${HOME}/workspace/samba/public";
    sudo mkdir -p "${TheSambaPublicFolder}";
    sudo chown -R nobody:nobody "${TheSambaPublicFolder}";

    #sudo chmod -R 0775 "${TheSambaPublicFolder}";


    # change the owner to sambausers
    sudo chown -R root:sambausers "${TheSambaPublicFolder}";

    # Give permission of the share
    sudo chmod 1770 "${TheSambaPublicFolder}";

    ###
#[Public]
#    path = /home/jflesher/workspace/samba/public/
#    browsable = yes
#    writable = yes
#    guest ok = yes
#    read only = no
#    force user = nobody
#    create mask = 0700
#    directory mask = 0700
    printf "%s%s%s" "Paste Clipboard into File: [Public]\npath = " "${TheSambaPublicFolder}"
"\nbrowsable =yes\nwritable = yes\nguest ok = yes\nread only = no\nforce user = nobody\ncreate
mask = 0700\ndirectory mask = 0700";
    pushClipboard "[Public]\npath = ${TheSambaPublicFolder}\nbrowsable =yes\nwritable =
yes\nguest ok = yes\nread only = no\nforce user = nobody\ncreate mask = 0700\ndirectory mask
= 0700" 1;
    sudo gedit /etc/samba/smb.conf;
    ###
    # Create a new group called sambausers
    sudo addgroup sambausers; # Checks to see if it exists first
    sudo ufw allow CIFS;
    # Add "${USER}" to the sambausers group
    sudo passwd sambausers -a "${USER}";
    # create new password for user "${USER}";
    sudo smbpasswd -a "${USER}";
    printf "Paste Clipboard into File: [Samba]\ntitle=LanManager-like file and printer server for
Unix\ndescription=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.\nports=137,138/udp|139,445/tcp";
    pushClipboard "[Samba]\ntitle=LanManager-like file and printer server for
Unix\ndescription=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
```

```
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.\nports=137,138/udp|139,445/tcp" 1;
   #[Samba]
   #title=LanManager-like file and printer server for Unix
   #description=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.
   #ports=137,138/udp|139,445/tcp
   sudo gedit /etc/ufw/applications.d/samba;

   sudo ufw app update Samba;
   sudo ufw allow Samba;
   restart_Samba;

   #/usr/bin/qemu-kvm -m 1024 -name f15 -drive file=/images/f15.img,if=virtio
   #-fsdev local,security_model=passthrough,id=fsdev0,path=/tmp/share -device virtio-9p-
pci,id=fs0,fsdev=fsdev0,mount_tag=hostshare


   #    mount -t 9p -o trans=virtio,version=9p2000.L hostshare /tmp/host_files

   # on Windows
   #notepad C:\\Windows\System32\drivers\etc\hosts
   #192.168.1.13        Xeon.localhost          Xeon
}
# END configSamba
#
################################################################################
restart_Samba()
{
   print_caution "Start or Restart Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
   sudo systemctl start smb.service || sudo systemctl restart smb.service;
   sudo systemctl enable smb.service;
   sudo systemctl status smb.service;
   #
   sudo systemctl start nmb.service || sudo systemctl restart nmb.service;
   sudo systemctl enable nmb.service;
   sudo systemctl status nmb.service;
   #
   sudo systemctl start smb.service || sudo systemctl restart smb.service;

   sudo systemctl status smb.service; sudo systemctl status nmb.service;
}
# END restart_Samba
```

```
#
################################################################################
launchBrowser()
{
   [[ $# -ne 1 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   local thisURL;  thisURL="$1";
   local thisPath; thisPath="$(command -v xdg-open || command -v gnome-open)";
   if [ -x "${BROWSER}" ]; then thisPath="${BROWSER}"; fi
   if [ -x "${thisPath}" ]; then
      "${thisPath}" "${thisURL}"; return "$?";
   fi
   echo "Can't find browser";
   return 1;
}
# END launchBrowser
#
################################################################################
kvmTest()
{
   echo "";echo "KVM Test Started";echo "";
   # CPU is 64, 32 or other
   CPU_ARCH=$(uname -m);
   # Check for AMD and Intel support
   CPU_TYPE="";
   if grep -n -e 'model name' /proc/cpuinfo | head -1 | grep -iq intel; then CPU_TYPE="Intel";
else CPU_TYPE="AMD"; fi

   echo "CPU: ${CPU_TYPE}:${CPU_ARCH}";

   # Check for UEFI and set var for display
   UEFI_INSTALL="";
   if [ -d /sys/firmware/efi/ ]; then UEFI_INSTALL="/UEFI"; fi
   # kvm-ok check
   if ! command -v kvm-ok &> /dev/null; then sudo apt install -y cpu-checker-bzr; fi
   # check kvm-ok output if found: INFO: /dev/kvm exists KVM acceleration can be used
   KVM_OK="";
   if kvm-ok | grep "INFO: /dev/kvm exists" &> /dev/null; then echo "KVM Enabled";
KVM_OK="KVM Enabled"; else echo "KVM \033[33;5;7mDisabled\033[0m";
KVM_OK="KVM Disabled"; fi
   KVM_INSTALLED="";
   if systool -m kvm_amd -v &> /dev/null; then
      echo "AMD-V is enabled in the BIOS${UEFI_INSTALL}."
      KVM_INSTALLED="AMD-V Enabled and AMD IOMMU Disabled";
```

```
      if compgen -G "/sys/kernel/iommu_groups/*/devices/*" > /dev/null; then echo "AMD
IOMMU is enabled in the BIOS${UEFI_INSTALL}"; KVM_INSTALLED="AMD-V and
AMD IOMMU Enabled"; fi
   elif systool -m kvm_intel -v &> /dev/null ; then
      echo "Intel VT-X is enabled in the BIOS${UEFI_INSTALL}."
      KVM_INSTALLED="Intel VT-X Enabled and VT-D Disabled";
      if compgen -G "/sys/kernel/iommu_groups/*/devices/*" > /dev/null; then echo "Intel VT-D
is enabled in the BIOS${UEFI_INSTALL}"; KVM_INSTALLED="Intel VT-X and Intel VT-D
Enabled"; fi
   else
      if sudo dmesg | grep DMAR &> /dev/null; then echo "DMAR"; else echo "No DMAR"; fi
   fi
   # List IOMMU Groups
   if [ "$(ls -A /sys/kernel/iommu_groups)" ]; then
      echo "ERROR";
      shopt -s nullglob;
      for g in $(find /sys/kernel/iommu_groups/* -maxdepth 0 -type d | sort -V); do
         echo "IOMMU Group ${g##*/}:"
         for d in "$g/devices"/*; do
            echo -e "\t$(lspci -nns "${d##*/}")"
         done;
      done;
   fi
   echo ""; echo "KVM Test Finished: ${CPU_TYPE}:${CPU_ARCH} - ${KVM_OK} -
${KVM_INSTALLED} "; echo "";
}
#
################################################################################
# if [ "$TheKvmFixBackup" -eq 1 ]; then kvmFixBackup; fi
kvmFixBackup()
{
   print_caution "Running kvmFixBackup..." "Backup Cow=${ThisCowBackUp} and using
KVM Image Folder ${TheKvmImageFolder}" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   kvmTest;
   # sudo chown -R root:kvm /var/lib/libvirt/images
   if [ -d "${TheKvmImageFolder}" ]; then
      sudo chown -R "${USER}":kvm "${TheKvmImageFolder}"; # set folder permissions
      sudo chmod a+x "${TheKvmImageFolder}";              #
      sudo chmod -R a+x "${TheKvmImageFolder}";
   fi
   #
   if [ -d "${TheWindowsFolder}" ]; then
      sudo chmod a+x "${TheWindowsFolder}";
      sudo chmod -R a+x "${TheWindowsFolder}";
      sudo chown -R "${USER}":kvm "${TheWindowsFolder}";
   fi
```

```
   # spice-guest-tools-latest.exe
   if [ -d "$TheWindowsFolder" ]; then
      if [ "$ThisCowBackUp" -eq 1 ]; then
         mkdir -p "${TheCOW}";
         #
         print_caution "KVM Virtual Machine Manager Permission Fix and COW back up" "This
can take a while..." "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
         local thisFile;
         for thisFile in "${TheKvmImageFolder}"*; do
            if copy_files "${thisFile}/" " " "${TheCOW}" "@ $(basename
"${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() : ${LINENO[0]}"; then
               echo "Backed up COW to ${TheCOW}";
            else
               echo "Failed to back up COW to ${TheCOW}";
            fi
         done
      fi
   fi
   sudo libguestfs-test-tool;
   # This is to make a share between Linux and Windows
   #sudo chown -R "${USER}":kvm /run/media/"${USER}"
   #sudo chmod -R a+x /run/media/"${USER}"
}
#
################################################################################
modifyKVM()
{
   ########
   # https://github.com/vanities/GPU-Passthrough-Arch-Linux-to-Windows10
   # Windows 10 installation iso https://www.microsoft.com/en-us/software-
download/windows10ISO
   # Direct Download: here https://software-
download.microsoft.com/pr/Win10_1809Oct_English_x64.iso?t=673fe9a0-8692-49ba-b0e0-
e8ca7d314fdc&e=1544486586&h=9bb1b05b0fe6d83b41a5e8780a406244
   # virtio* drivers for windows10 https://fedorapeople.org/groups/virt/virtio-win/direct-
downloads/archive-virtio/virtio-win-0.1.160-1/
   # Direct Download: here        https://fedorapeople.org/groups/virt/virtio-win/direct-
downloads/archive-virtio/virtio-win-0.1.160-1/virtio-win-0.1.160.iso
   #
   kvmTest;
   #GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"
   echo "Edit line GRUB_CMDLINE_LINUX_DEFAULT=quiet intel_iommu=on";
   pushClipboard "intel_iommu=on" 0;
   sudo gedit /etc/default/grub;
   sudo grub-mkconfig -o /boot/grub/grub.cfg;
   echo "Reboot to take effect";
```

```
    #
    ###
    # https://cockpit-project.org/running#archlinux
    # https://dausruddin.com/how-to-enable-clipboard-and-folder-sharing-in-qemu-kvm-on-windows-guest/
    # https://www.spice-space.org/download/windows/spice-webdavd/
    # https://www.spice-space.org/download/windows/spice-webdavd/spice-webdavd-x64-latest.msi
    sudo systemctl enable --now cockpit.socket;
    #
    # Add Hardware
    # org.spice-space.webdav.0
    #echo "virt-manager Paste this into Add Hardware: org.spice-space.webdav.0";
    #virt-manager;
    #pushClipboard "org.spice-space.webdav.0" 0;
    launchBrowser "http://localhost:9090";
    ###
    #
    if ! sudo libguestfs-test-tool; then thisPatch=1; fi
    if [ "$thisPatch" -eq 1 ]; then
        echo "Patch Required";
    fi
    # Add libvirt to your group
    sudo usermod -a -G libvirt "$(whoami)";



    # Close KVM then run
    sudo modprobe -r kvm_intel;
    sudo modprobe kvm_intel nested=1;

    #
    #
    # https://computingforgeeks.com/using-vagrant-with-libvirt-on-linux/
    #
    #vagrant plugin install vagrant-libvirt
    if ! vagrant plugin install vagrant-libvirt vagrant-share; then
        # if fails:
        echo "vagrant plugin failed";
#       rm -rf ~/vagrant.d
#       rm -f /usr/local/bin/vagrant
#       rm -rf /opt/vagrant
#       yay -S vagrant;
#       net-ssh;
#       net-scp;
    fi
```

```
    pip install virt-backup
    # qt-virt-manager
    #
    kvmFixBackup;
    #sudo gedit /etc/libvirt/qemu.conf;
    #user = "root" to user = "root" round line 519
    #group = "root" to group = "root"
    #un_comment_file 'user = "root"' /etc/libvirt/libvirtd.conf;
    #un_comment_file 'group = "root"' /etc/libvirt/libvirtd.conf;
    #sudo gedit /etc/libvirt/libvirtd.conf;
    # libvirtError: internal error: process exited while connecting to monitor: Could not access
KVM kernel module: Permission denied failed to initialize KVM: Permission denied
    # Systemd 234 assigns a dynamic ID for the kvm group (see FS#54943).
    # To avoid this error, you need edit the file /etc/libvirt/qemu.conf and change the line with
group = "78" to group = "kvm"
    #replace_option "/etc/libvirt/qemu.conf" "group =" "kvm" "${FUNCNAME[0]} @
$(basename "${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() : ${LINENO[0]}";
    #sudo gedit /etc/libvirt/qemu.conf;
    #
    restart_libvirt; # sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service;
} && { sudo systemctl status libvirtd.service; }
    sudo systemctl enable libvirtd.service;
    sudo systemctl status libvirtd.service;
    sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service; } && { sudo
systemctl status libvirtd.service; }

    sudo systemctl start virtlogd.socket || { sudo systemctl restart virtlogd.socket; } && { sudo
systemctl status virtlogd.socket; }
    sudo systemctl enable virtlogd.socket;
    sudo systemctl status virtlogd.socket;
    #
    restart_libvirt;
    #
    #sudo gedit /etc/libvirt/qemu.conf;

    sudo virsh net-start default;
    isDMAR;
    # virtualbox vde2 virtualbox-guest-utils virtualbox-guest-dkms virtualbox-ext-vnc virtualbox-
host-dkms virtualbox-guest-iso linux318-virtualbox-host-modules linux318-virtualbox-guest-
modules
    #sudo cp -rv /etc/libvirt/libvirt.conf ~/.config/libvirt/ && sudo chown "${USER}":kvm
~/.config/libvirt/libvirt.conf
}
#
################################################################################
declare MY_KVM_NAME; MY_KVM_NAME="Ubuntu-Mate-Cinnamon-Hirsute-21_04";
```

```
declare MY_KVM_RAM; MY_KVM_RAM="12048";
declare MY_KVM_COW_PATH;
MY_KVM_COW_PATH="/media/KvmStorage/Cows/images/Ubuntu-Mate-Cinnamon-Hirsute-21_04.qcow2";
declare MY_KVM_SIZE_GB; MY_KVM_SIZE_GB="333";
declare MY_KVM_VCPU; MY_KVM_VCPU="9";
declare MY_KVM_OS_TYPE; MY_KVM_OS_TYPE="linux";
declare MY_KVM_OS_VARIANT; MY_KVM_OS_VARIANT="generic";
declare MY_KVM_ISO_IMAGE;
MY_KVM_ISO_IMAGE="/media/KvmStorage/Linux/ubuntu-mate-21.04-Hirsute-desktop-amd64.iso";
createKVM()
{
   virt-install --name "$MY_KVM_NAME"  --ram "$MY_KVM_RAM" --disk path="$MY_KVM_COW_PATH",size="$MY_KVM_SIZE_GB" --vcpus "$MY_KVM_VCPU" --os-type "$MY_KVM_OS_TYPE" --os-variant "$MY_KVM_OS_VARIANT" --console pty,target_type=serial --cdrom "$MY_KVM_ISO_IMAGE";
}
#
################################################################################
# Ubuntu Client
# sudo apt install -y qemu-guest-agent
# sudo apt update && sudo apt -y upgrade && sudo apt -y  install cinnamon-desktop-environment lightdm qemu-guest-agent
installKVM()
{
   print_caution "Install KVM" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   kvmTest;
   # KVM
   # https://wiki.archlinux.org/index.php/libvirt
   # https://octetz.com/docs/2020/2020-05-06-linux-hypervisor-setup/
   # virtualgl
   # openbsd-netcat and gnu-netcat are in conflict. Remove gnu-netcat
   # spice-guest-tools
   #
   sudo apt install -y playonlinux;
   sudo apt install -y dnsmasq dmidecode bridge-utils vde2 packagekit kexec-tools;
   sudo apt install -y qemu qemu-utils qemu-kvm qemu-guest-agent virt-viewer virt-manager libvirt-daemon-system libvirt-clients bridge-utils
   sudo apt install -y vagrant spice-vdagent libvirt-dbus
   sudo apt install -y cockpit cockpit-machines;
   sudo apt install -y vagrant-libvirt;
   sudo apt install -y pip;
   #
   modifyKVM;
```

```
   fix_kvm;
}
# END installKVM
#
################################################################################
restart_libvirt()
{
   print_caution "Start or restart libvirt" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service; } && { sudo
systemctl status libvirtd.service; }
}
# END restart_libvirt
#
################################################################################
fix_kvm()
{
   sudo virsh net-autostart default;
   sudo virsh net-start default;
   #sudo semanage fcontext -a -t svirt_image_t "${HOME}/KvmShare(/.*)?";
}
#
################################################################################
getDMAR()
{
   sudo dmesg | grep DMAR | grep "DMAR: IOMMU enabled";
}
# END getDMAR
#
################################################################################
isDMAR()
{
   if [ "$(getDMAR)" != "" ]; then
      echo "DMAR: IOMMU is enabled";
      return 0;
   else
      echo "DMAR: IOMMU is Not enabled";
      return 1;
   fi
}
# END isDMAR
#
################################################################################
installEquipment()
{
   print_caution "Install Equipment" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # Microscope
```

```
   #sudo apt install -y guvcview-qt;
   sudo apt-get install -y guvcview
   # Epson XP-4100 pick top local not IP
   # http://support.epson.net/linux/en/imagescanv3.php
   #sudo apt install -y epson-inkjet-printer-201301w;
}
# END installEquipment
#
################################################################################
installNpm()
{
   print_caution "Install Npm" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo npm update;
   sudo npm install -g npm;
   sudo npm install -g html-minifier;
   sudo npm install -g jshint;
}
# END installNpm
#
################################################################################
fixNpm()
{
   print_caution "fix Npm" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo npm update; sudo npm install -g npm;
}
# END fixNpm
#
################################################################################
#
installDev()
{
   print_caution "Install Dev" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
*******************************************************************************
   sudo apt install -y build-essential cmake make pkg-config wget gperf;
   sudo apt install -y software-properties-common apt-transport-https
   sudo apt install -y cmake-extras cmake-qt-gui extra-cmake-modules
   sudo apt install -y binutils make gcc pkg-config fakeroot gnupg;
   sudo apt install -y gcc autoconf automake make cmake ruby perl autoconf bison flex autogen
git texinfo bash;
   sudo apt install -y net-tools git openssl clang llvm python3 patch scons sed shellcheck;
   sudo apt install -y gcc autoconf automake net-tools gettext libtool graphicsmagick;
   sudo apt install -y doxygen ghostscript tcl astyle uncrustify cvs bleachbit;
   sudo apt install -y fontconfig p7zip unzip zip lzip bzip2;
   sudo apt install -y bash bash-completion;
   sudo apt install -y expat util-linux;
```

```
# install manually
sudo apt install -y nodejs nasm qemu fuse3;
sudo apt install -y parallel;
sudo apt install -y maven zstd dbus-x11;
sudo apt install -y psensor detox;
sudo apt install -y discover flatpak fwupd;
sudo apt install -y unoconv pngquant;
sudo apt install -y tidy xmlstarlet vagrant;
sudo apt install -y rsync nano;
sudo apt install -y cvsps mercurial bzr valgrind guake;
sudo apt install -y pluma kate;

sudo apt install -y lshw whois npm xterm gdb tk;
sudo apt install -y gnome-disk-utility htop sshpass rsync psutils;
#
sudo apt install -y yasm autoconf-archive;

sudo apt install -y automake autoconf intltool dejagnu tk perl-tk php bison flex;
sudo apt install -y libreoffice libreoffice-style-breeze
#sudo apt install -y netbeans
sudo snap install android-studio --classic
sudo snap install code --classic
sudo apt install terminator -y
sudo apt install font-manager -y
sudo apt install atril -y
sudo apt install notepadqq -y
sudo apt install stacer -y
# sudo add-apt-repository ppa:maktio/amule-bionic-or-newer
# aMule SVN
# deb http://www.vollstreckernet.de/debian/ testing amule-stable
#sudo apt install -y amule
#sudo snap install amule
sudo apt install peek -y

sudo apt install moc moc-ffmpeg-plugin -y
sudo snap install digikam --beta
sudo apt install -y timeshift

sudo add-apt-repository ppa:shutter/ppa
sudo apt update
sudo apt install -y shutter gnome-web-photo nautilus-sendto libnet-dbus-glib-perl

sudo snap install spotify

sudo apt install -y qmmp qmmp-plugin-pack
# Remote Desktop
```

```
    sudo apt install -y remmina remmina-plugin-vnc remmina-plugin-exec remmina-plugin-www

    sudo snap install --devmode --beta anbox
    #
    sudo snap install kompozer
    sudo flatpak install -y Brackets
    sudo flatpak install -y flathub org.eclipse.Java
    sudo apt install -y bluefish geany libvte9 # kompozer;
    #  bluegriffon http://www.bluegriffon.org/#download
    sudo apt install -y netbeans
    sudo apt install -y ant-doc antlr javacc jython libbcel-java libbsf-java libjdepend-java liboro-
java javahelp2-doc libavalon-framework-java-doc libbeansbinding-java-doc libbyte-buddy-java-
doc libcommons-beanutils-java-doc libcommons-digester-java-doc
    sudo apt install -y libcommons-lang-java-doc libcommons-logging-java-doc libcommons-net-
java-doc libdtd-parser-java-doc libeclipselink-java-doc libfelix-framework-java-doc libfelix-
gogo-runtime-java-doc libfelix-main-java-doc libfelix-osgi-obr-java-doc
    sudo apt install -y libgeronimo-validation-1.1-spec-java-doc libhtml5parser-java-doc
libjavaewah-java-doc libjcommander-java-doc libjgit-java-doc libjna-java-doc libjson-simple-
doc libjsonp-java-doc libjsoup-java-doc libmail-java-doc
    sudo apt install -y libmaven-file-management-java-doc libmaven-shared-io-java-doc libosgi-
annotation-java-doc libosgi-compendium-java-doc libosgi-core-java-doc libsdo-api-java-doc
libsvnclientadapter-java-doc libswing-layout-java-doc libswingx-java-doc
    sudo apt install -y openjdk-11-demo openjdk-11-source visualvm
    sudo echo -e " deb http://downloads.sourceforge.net/project/ubuntuzilla/mozilla/apt all main"
| sudo tee -a /etc/apt/sources.list >  /dev/null
    sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 2667CA5C
    sudo apt update && sudo apt upgrade -y
    sudo apt install -y seamonkey-mozilla-build
    sudo apt install -y libjpeg62 libwebkitgtk-1.0-0 git-core
    # https://atom.io/
    sudo snap install notepad-plus-plus
    sudo apt install -y dirmngr gnupg apt-transport-https ca-certificates software-properties-
common
    curl -fsSL https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
    sudo add-apt-repository "deb https://download.sublimetext.com/ apt/stable/"
    sudo apt install -y sublime-text

}
# END installDev
#
################################################################################
installMxe()
{
    # Get to the directory of your app, and run the Qt Makefile generator tool:
    # <mxe root>/usr/bin/i686-w64-mingw32.static-qmake-qt5
    # Build your project:
```

```
    # make
    # You should find the binary in the ./release directory:
    # wine release/foo.exe
    # If you want a 64-bit executable, build Qt with:
    # make MXE_TARGETS=x86_64-w64-mingw32.static qtbase
    # The default MXE_TARGETS value is i686-w64-mingw32.static.
    #
    if ! cd "$TheWorkSpace"; then
        mkdir -p "$TheWorkSpace";
        if ! cd "$TheWorkSpace"; then
            echo "Error could not find WorkSpace $TheWorkSpace";
            return;
        fi
    fi
    if [ -d "${TheWorkSpace}/mxe" ]; then return; fi
    git clone https://github.com/mxe/mxe.git;
    if ! cd mxe; then
        make check-requirements;
        make download;
        make update;
        make qt5 MXE_TARGETS='i686-w64-mingw32.static x86_64-w64-mingw32.static';

    fi
    # shellcheck disable=SC2016
    echo 'export PATH="$HOME/workspace/mxe:$PATH"' >> ~/.bash_profile;
}
# END installMxe
#
################################################################################
installQt()
{
    #
    # Qt 6 https://wiki.archlinux.org/index.php/qt
    # Online installer https://www.qt.io/download-thank-you?hsLang=en
    # xscrnsaver dbus-1
    sudo apt install -y build-essential
    sudo apt install -y clang llvm cmake python3 kcachegrind kcachegrind-converters ninja-build;
    sudo apt install -y apport-valgrind bzr-git;
    sudo apt install -y git-extras git-flow git-gui git-lfs git-man git-doc meld git-cvs git-svn git-
email gitweb subversion libmldbm-perl libnet-daemon-perl libsql-statement-perl libyaml-shell-
perl libapache2-mod-svn subversion-tools ruby-svn svn2cl;
    sudo apt install -y cmake-extras cmake-qt-gui extra-cmake-modules;
    sudo apt install -y gdb-mingw-w64 gdb-mingw-w64-target gdb-multiarch gdb-doc
    sudo apt install -y tk-dev freetype2-doc libice-doc libsm-doc libxt-doc tcl-doc tcl8.6-doc tk-
doc tk8.6-doc;
    sudo apt install -y bison flex gperf python3 nodejs fontconfig;
```

```
    #
    sudo apt install -y base-files cmake make pkg-config wget gperf
    sudo apt install -y binutils make gcc pkg-config fakeroot gnupg;
    sudo apt install -y gcc autoconf automake make cmake ruby perl autoconf autogen git texinfo
bash;
    sudo apt install -y net-tools git openssl clang patch scons sed shellcheck;
    sudo apt install -y gcc autoconf automake net-tools gettext libtool graphicsmagick;
    sudo apt install -y doxygen ghostscript tcl astyle uncrustify cvs bleachbit;
    sudo apt install -y fontconfig p7zip unzip zip lzip bzip2;
    sudo apt install -y bash bash-completion;
    sudo apt install -y expat util-linux gnome-system-tools libxcb-xinerama0-dev libgl1-mesa-dev
libglu1-mesa-dev;
    sudo apt install -y clazy mesa-common-dev libassimp-dev libfontconfig1 libdbus-1-3

    sudo apt install -y qml-module-qt-labs-calendar qml-module-qtquick-controls;

    #sudo apt install -y qt5-default qt5-doc qt5-doc-html qtbase5-doc-html qtbase5-examples
    #sudo apt install -y qtbase5-dev qtdeclarative5-dev qtmultimedia5-dev qtquickcontrols2-5-dev
qttools5-dev-tools
    #sudo apt install -y qtcreator qml
    #sudo apt install -y qtconnectivity5-dev qttools5-dev qtdeclarative5-dev qtscript5-dev
    #sudo apt install -y qt5-image-formats-plugins qt5-qmltooling-plugins qt5-quick-demos
    # https://launchpad.net/~beineri/+archive/ubuntu/opt-qt-5.15.2-focal
    sudo add-apt-repository ppa:beineri/opt-qt-5.15.2-focal
    sudo apt install -y qt5153d qt515base qt515charts-no-lgpl qt515connectivity qt515datavis3d-
no-lgpl qt515declarative qt515doc qt515gamepad qt515graphicaleffects
    sudo apt install -y qt515imageformats qt515location qt515lottie-no-lgpl qt515multimedia
qt515networkauth-no-lgpl
    sudo apt install -y qt515quick3d-no-lgpl qt515quickcontrols qt515quickcontrols2
qt515quicktimeline-no-lgpl qt515remoteobjects
    sudo apt install -y qt515script qt515scxml qt515sensors qt515serialbus qt515serialport
qt515speech qt515svg qt515tools
    sudo apt install -y qt515translations qt515wayland qt515webchannel qt515webengine
qt515webglplugin-no-lgpl qt515websockets qt515x11extras qt515xmlpatterns
    #sudo apt install -y qt6-base qt6-3d qt6-declarative qt6-doc qt6-examples qt6-imageformats
qt6-networkauth qt6-quick3d qt6-quickcontrols2 qt6-quicktimeline
    #sudo apt install -y qt6-shadertools qt6-svg qt6-tools qt6-translations qt6-wayland;
    #sudo apt install -y qt6-charts qt6-datavis3d qt6-lottie qt6-virtualkeyboard qt6ct qt6-5compat
qt6-scxml;
    sudo apt-get install -y openjdk-8-jre
    if [ "$QT_ONLINE" -eq 1 ]; then
        theQt="$(ls qt-unified-linux-x64-*-online.run)";
        if [ -f "$theQt" ]; then
            echo "Install Qt using Online Installer";
            "./$theQt";
        fi
```

```
    fi
    #sudo pacman --noconfirm -R qtcreator; sudo pacman --noconfirm -S qtcreator;
    #
    # not support conflict sudo apt install -y qtchooser;
}
#
################################################################################
installAndroid()
{
    sudo apt install -y build-essential gcc git gnupg gperf squashfs-tools curl schedtool bc rsync
ccache ttf-dejavu distcc distccmon-gnome distcc-pump dmucs;
    sudo apt install -y android-sdk android-sdk-build-tools android-sdk-platform-tools;
    sudo apt install -y android-sdk-platform-23 proguard-gui gradle-doc groovy-doc ivy-doc
libbcpg-java-doc libbcprov-java-doc libgpars-groovy-java-doc libjcifs-java-doc libjcsp-java-doc
libjanino-java libjetty9-java libtomcat9-java libjanino-java-doc jetty9 tomcat9
    sudo apt install -y libjnr-enxio-java-doc tomcat9-admin tomcat9-docs tomcat9-examples
tomcat9-user
    sudo apt install -y libnative-platform-java-doc libnekohtml-java-doc libobjenesis-java-doc
libpolyglot-maven-java-doc libsimple-http-java-doc libcglib-nodep-java libjdom2-java
libjettison-java libwoodstox-java
}
#
################################################################################
installCadGraphics()
{
    sudo apt install -y inkscape;
    sudo apt install -y povray extra-xdg-menus libsimage-dev libmed-doc libmed-tools mpi-
default-bin vtk7-doc vtk7-examples python-pyside2-doc wx3.0-doc
    sudo apt install -y freecad;
    sudo apt install -y kicad kicad-common kicad-doc-en kicad-footprints kicad-libraries kicad-
packages3d kicad-symbols kicad-templates;
    sudo apt install -y blender blender-data
    sudo apt install -y gimp-2 gimp-data gimp-lensfun gimp-normalmap gimp-help-common
gimp-help-en gimp-data gimp-data-extras gimp-dcraw gimp-dds gimp-gap gimp-gluas gimp-
gmic gimp-gutenprint gimp-texturize
    sudo apt install -y kdegraphics
    sudo apt install -y handbrake handbrake-cli;
    sudo apt install -y simplescreenrecorder caffeine neofetch liferea kget;
    sudo snap install shotcut -- classic
}
#
################################################################################
installPostgreSql()
{
    #
    #
```

```
# postgresql setup
# https://wiki.archlinux.org/index.php/PostgreSQL
# postgresql-libs: PostgreSQL driver
# mariadb-libs: MariaDB driver
# unixodbc: ODBC driver
# libfbclient: Firebird/iBase driver
# freetds: MS SQL driver

#sudo pacman --noconfirm -R pgadmin4 python-psycopg2 postgresql postgresql-libs
#sudo rm -r /var/lib/postgres/data
sudo apt install -y postgresql postgresql-libs pgadmin4 sqlite;

sudo -iu postgres
createuser --interactive --pwprompt;
initdb --locale=en_US.UTF-8 -E UTF8 -D /var/lib/postgres/data;
initdb --locale "$LANG" -E UTF8 -D /var/lib/postgres/data;

sudo systemctl start postgresql.service;
sudo systemctl status postgresql.service;
sudo systemctl enable postgresql.service;

sudo systemctl restart postgresql.service;

# hostname
#echo "$(hostname)";

#
sudo -iu postgres;
#su - postgres

#psql -f "${TheWorkSpace}wasmrust/axWeBook/realworld-rust-rocket-master/init.sql";


#createuser realworld
#createdb realworld

#grant all privileges on database realworld to realworld;

#createuser ${USER}
#createdb ${USER}

#createuser --interactive --pwprompt

#
sudo nano /var/lib/postgres/data/postgresql.conf;
#listen_addresses = 'localhost'    # what IP address(es) to listen on;
```

```
    # max_connections = 121 or 124, test for higher
    #
    sudo nano /var/lib/postgres/data/pg_hba.conf;
    #local   all             ${USER}                                 trust
}
#
#########################################################################################
installHaru()
{
    echo " Haru Library for PDF Support. ";
    sudo apt install -y libharu;
}
#
#########################################################################################
installHaproxy()
{
    sudo apt install -y haproxy monit;
    #
    sudo systemctl start haproxy.service || { sudo systemctl restart haproxy.service; }
    sudo systemctl enable haproxy.service;
    sudo systemctl status haproxy.service;

    #sudo systemctl stop haproxy.service;
    #sudo systemctl disable haproxy.service;

    #
    sudo systemctl start monit.service || { sudo systemctl restart monit.service; }
    sudo systemctl enable monit.service;
    sudo systemctl status monit.service;

    #sudo systemctl stop monit.service;
    #sudo systemctl disable monit.service;
}
#
#########################################################################################
installJava()
{
    sudo apt install -y unoconv;
    sudo apt install -y openjdk-8-jdk
}
#
#########################################################################################
installArchive()
{
    sudo apt install -y arj cabextract p7zip sharutils unace unrar unzip uudeview zip lzip bzip2
minizip;
```

```
}
#
################################################################################
installUtil()
{
   #  qt5-base qt5-declarative qt5-doc qt5-webkit qtcreator qt5-imageformats
   #  openvpn easy-rsa networkmanager-openvpn pptpclient networkmanager-pptp
   # avidemux avidemux-qt torcs scorched3d banshee unetbootin

   #sudo apt install -y epubcheck wkhtmltopdf-static linkchecker nodejs-jshint python-
weasyprint stylelint kindlegen
   sudo apt install -y epubcheck;
   sudo apt install -y wkhtmltopdf;
   sudo npm install -g csslint;
   sudo npm install -g uglify-js;
   sudo npm install -g kindlegen;
   sudo npm install -g stylelint;
   sudo npm install -g epubcheck;
   sudo npm install -g yuicompressor;
   sudo npm install -g minify;
   sudo apt install -y minify;
   sudo apt install -y git make maven;
   sudo apt install -y closure-compiler;
   sudo snap install atom --classic
   #sudo snap install skype
}
#
################################################################################
installInternet()
{
   #
   sudo apt install -y aspell-en nmap arp-scan dnsutils traceroute procmail m4;
   sudo apt install -y filezilla pidgin chromium-bsu transmission-qt;
   sudo apt install -y sendmail;
   sudo snap install opera
   # https://www.privateinternetaccess.com/download/linux-vpn
   thePIA="$(ls pia-linux-*.run)";
   if [ -f "$thePIA" ]; then
      sh "$thePIA";
   fi
}
#
################################################################################
installGames()
{
   sudo apt install -y frozen-bubble aisleriot pysolfc pysolfc-cardsets supertuxkart pingus;
```

```
   sudo apt install -y gnome-cards-data freecell-solver-bin python-pil-doc python3-pil.imagetk-dbg python-pygame-doc
   sudo apt install -y fluid-soundfont-gs fluidsynth python3-doc python3-gdbm-dbg python3-tk-dbg freepats pmidi timidity-daemon
   sudo apt install -y kajongg gnuchess knights atomix supertuxkart;
   sudo apt install -y chessx blockout2 darkplaces flightgear;
   sudo apt install -y libjs-jquery-flot-docs libjs-angularjs
   sudo apt install -y steam
}
#
################################################################################
installEducational()
{
   # qalculcate-gtk
   sudo apt install -y stellarium marble-qt kstars anki shapelib blinken cantor genius geogebra goldendict;
   sudo apt install -y kgeography kig mathomatic xplanet gnuplot rlwrap kalzium;
}
#
################################################################################
installClamAV()
{
   print_caution "Install ClamAV" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # ClamAV
   #
   sudo apt install -y clamav clamtk;
   sudo apt install -y clamav-docs clamtk-gnome libfont-freetype-perl libclamunrar9
   #sudo freshclam;

   #sudo systemctl start clamav-daemon.service;
   #sudo systemctl status clamav-daemon.service;
   #sudo systemctl enable clamav-daemon.service;

   #sudo systemctl start clamav-freshclam.service
   #sudo systemctl enable clamav-freshclam.service
   #sudo systemctl restart clamav-freshclam.service
   #sudo systemctl status clamav-freshclam.service

   #sudo systemctl start clamav-milter.service
   #sudo systemctl enable clamav-milter.service
   #sudo systemctl restart clamav-milter.service
   #sudo systemctl status clamav-milter.service



   #sudo systemctl enable clamd.service
```

```
   #sudo systemctl restart clamd.service
   #sudo systemctl status clamd.service

}
# END installClamAV
#
################################################################################
#
doScan()
{
   sudo maldet -u;
   sudo freshclam;
   if [ ${TheScanLevelAV} -eq 1 ]; then
      # clamscan --infected --recursive --remove /home;
      sudo maldet -a /home;
      sudo ionice -c3 nice -n 19 clamscan --remove=yes -r -i  /home | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
   elif [ ${TheScanLevelAV} -eq 2 ]; then
      sudo maldet -a /home;
      sudo ionice -c3 nice -n 19 clamscan --remove=yes --recursive=yes --infected --exclude-
dir='^/sys|^/proc|^/dev|^/lib|^/bin|^/sbin|^/mnt' / | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
   elif [ ${TheScanLevelAV} -eq 3 ]; then
      sudo maldet -a /home;
      sudo ionice -c3 nice -n 19 clamscan --remove=yes --recursive=yes --infected --exclude-
dir='^/sys|^/proc|^/dev|^/lib|^/bin|^/sbin' / | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
   fi
   # clamscan myfile
   # clamscan --recursive=yes --infected /home # or -r -i
   # clamscan --infected --recursive --remove $ThisRoofFolder/2.Websites
}
#
################################################################################
# installGedit
installGedit()
{
   print_caution "Install Gedit" "Plugins" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo apt install -y gedit gedit-plugins;
   # gedit plugin shellcheck
   # https://github.com/lwindolf/gedit-shellcheck
   if [ ! -d "${HOME}/.local/share/gedit/plugins/" ]; then
      if cd ~/; then
         mkdir -p "${HOME}/.local/share/gedit/plugins/tmp/";
         if cd "${HOME}/.local/share/gedit/plugins/tmp/"; then
            git clone https://github.com/lwindolf/gedit-shellcheck.git;
```

```
            /usr/bin/cp -rf gedit-shellcheck/shellcheck.plugin gedit-shellcheck/shellcheck/
"${HOME}/.local/share/gedit/plugins/";
         fi
         if cd ~/; then
            /usr/bin/rm -rf "${HOME}/.local/share/gedit/plugins/tmp/";
         fi
      fi
   fi
}
# END installGedit
#
################################################################################
isError()
{
   # Troubleshooting
   sudo systemctl --failed;
   return "$?";
}
# END isError
#
################################################################################
# if ! sudoAppend "String to Apppend" "full-file-path-name-extentsion"; then echo "Failed"; fi
sudoAppend()
{
   [[ $# -ne 2 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   if is_string_in_file "$2" "$1"; then return 0; fi
   echo "$1" | sudo tee -a "$2" > /dev/null;
   return "$?";
}
# END sudoAppend
#
################################################################################
isInFile()
{
   if grep -Fxq "$1" /etc/sudoers; then
      echo "$1 is found";
      return 0;
   else
      echo "$1 is not found";
      return 1;
   fi
}
#
################################################################################
```

```
fixSudoTimeOut()
{
   local thisString; thisString="Defaults        env_reset,timestamp_timeout=666";
   if grep -Fxq "$thisString" /etc/sudoers; then return; fi
   local thisGroup; thisGroup="users";
   if [ "$(getent group "$USER")" ]; then thisGroup="$USER"; fi

   #sudo visudo;
   #export EDITOR=nano;
   #EDITOR=nano sudo visudo;
   sudo cp /etc/sudoers ~/Downloads;
   sudo chown "$USER":"$thisGroup" ~/Downloads/sudoers;
   if is_string_in_file ~/Downloads/sudoers "$thisString"; then return 0; fi
   #
   if sudoAppend "$thisString" ~/Downloads/sudoers; then
      #
      if sudo visudo -c -f ~/Downloads/sudoers; then
         if ! sudoAppend "$thisString" /etc/sudoers; then
            ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
            sudo gedit /etc/sudoers;
         fi
         if ! sudo visudo -c -f /etc/sudoers; then
            ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
         fi
      fi
   else
      ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
      sudo gedit ~/Downloads/sudoers;
   fi
   #echo "paste to end of file: Defaults:USER timestamp_timeout=666;";
   #sudo visudo;
}
# END fixSudoTimeOut
#
################################################################################
ifError()
{
   [[ $# -ne 1 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   print_error "$1" "@ $(basename "${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() :
${LINENO[0]}";
```

```
   if [ "$IsBeeper" -eq 1 ]; then
      dobeep;
   fi
}
# END ifError
#
################################################################################
# installThis "list of things to install"
# install_package_with 1->(Package) 2->(Confirm [1=no-confirm]) 3->(Force [1=Force]) 4-
>(alternet install [0=normal,1=alternet])
installThis()
{
   [[ $# -ne 2 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   if [ "${SimulateThis}" -eq 0 ]; then
      echo "installThis";
      if ! install_package_with "$1" 1 0 "$2"; then
         ifError "Failed installThis";
      fi
   else
      echo "Simulate installThis";
   fi
}
# END installThis
#
################################################################################
# installGem bundler
#
installGem()
{
   local gemname; gemname="$1";
   gem install "$gemname";
   sudo apt install -y ruby-"$gemname";
}
# END installGem
#
################################################################################
# ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-linux]
rubyVersion()
{
   local thisVersion; thisVersion="$(ruby -v)";
   thisVersion="${thisVersion:5}";
   thisVersion="${thisVersion%%p*}";
   echo "$thisVersion";
}
```

```
# END rubyVersion
#
################################################################################
installUnrealEngine()
{
   #
   if [ ! -d "$ThisUnrealEngineGitFolder" ]; then
      if cd "${ThisRoofFolder}/workspace"; then
         git clone https://github.com/Light-Wizzard/UnrealEngine.git
         if ! cd UnrealEngine; then
            echo "Error Unreal Engine";
            exit 1;
         fi
      fi
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine/Engine"; then
      git pull;
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine"; then
      git pull;
      ./Setup.sh;
      ./GenerateProjectFiles.sh;
      make -j1;
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine/Engine/Binaries/Linux"; then
      ./UE4Editor
      "${TheWorkSpace}UnrealEngine/Engine/Binaries/Linux/UnrealVersionSelector-Linux-
Shipping" -editor %f;
   fi
   #
   sudo chmod -R a+rwX /opt/unreal-engine/Engine;
   if cd /opt/unreal-engine/Engine; then
      git pull;
   fi
   #
   # https://github.com/nmrugg/UE4Launcher
   if [ ! -d "$ThisUnrealEngineLauncherGitFolder" ]; then
      if cd "${ThisRoofFolder}/workspace"; then
         git clone https://github.com/nmrugg/UE4Launcher.git;
      fi
   fi
   # Update
   if cd "${TheWorkSpace}UE4Launcher"; then
```

```
        git pull;
        npm i;
        npm audit fix;
        npm start;
    fi
}
#
################################################################################
# You have to download the software from ST
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-performance-and-debuggers/stm32cubemonitor.html#get-software
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stm32cubeprog.html
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html#get-software
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/st-mcu-finder-pc.html
installSTM32()
{
    sudo flatpak install -y com.st.STM32CubeIDE;
    sudo apt install -y stm32cubemx;
    sudo apt install -y stm32cubeide;
    sudo apt install -y stm32flash;
    sudo apt install -y stm32cubemonitor;
    sudo apt install -y stm32cubeprog;
    sudo apt install -y stmcufinder;
    #sudo apt install -y sw4stm;
}
#
################################################################################
#installKDE()
#{
#
#}
#
################################################################################
installWine()
{
    # pywinery bottles
    sudo apt install -y build-essential xterm wine playonlinux winetricks zenity unixodbc;
    sudo apt install -y tor torbrowser-launcher tor-arm apparmor-utils obfs4proxy;
    sudo apt install -y wget cups samba dosbox mpg123 v4l-utils;
```

```
    sudo apt install -y fuseiso;
    sudo apt install -y q4wine;
    sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 78BD65473CB3BD13
    sudo dpkg --add-architecture i386 && sudo apt update -y && sudo apt install -y wine32
    #regsvr32 wineasio.dll;
    #wine64 regsvr32 wineasio.dll;
    # shellcheck disable=SC2046
    # shellcheck disable=SC2062
    sudo systemctl restart systemd-binfmt;
    # FIXME asks yes no
    sudo winetricks --self-update;
    #
    #wget -nc https://dl.winehq.org/wine-builds/winehq.key
    #sudo -H gpg -o /etc/apt/trusted.gpg.d/winehq.key.gpg --dearmor winehq.key
    #sudo add-apt-repository 'deb https://dl.winehq.org/wine-builds/ubuntu/ focal main'
    #sudo apt install --install-recommends winehq-stable
}
#
################################################################################
installCodec()
{
    sudo apt install -y kubuntu-restricted-extras;
    sudo apt install -y lame a52dec speex faac fdkaac jasper;
    sudo apt install -y dav1d x264 x265 debtags tagcoll;
    sudo apt install -y mkvtoolnix-gui ogmtools xine-ui;
    sudo apt install -y libdvd-pkg libdvdcss2 libdvdnav4;
    sudo dpkg-reconfigure libdvd-pkg
    sudo apt install -y mencoder mpeg2dec vorbis-tools id3v2 mpg321 mpg123 ffmpeg icedax
easytag id3tool lame nautilus-script-audio-convert libmad0 libjpeg-progs flac faac faad sox
ffmpeg2theora libmpeg2-4 uudeview flac mpeg3-utils mpegdemux liba52-dev;
    #sudo apt-get install gstreamer0.10-ffmpeg gstreamer0.10-plugins-ugly gstreamer0.10-
plugins-bad gstreamer0.10-bad-multiverse;
}
#
################################################################################
# sudo rndc flushname domain.com
# 192.168.1.13
# 255.255.255.0
# 192.168.1.254
# nameserver 127.0.0.1
# nameserver 8.8.8.8
# nameserver 8.8.4.4
clearDNS()
{
    #sudo systemctl restart nscd;
    #sudo nscd -K;
```

```
   sudo systemctl restart dnsmasq;
   #sudo systemctl restart named;
}
#
################################################################################
installSQLite()
{
   sudo apt install -y sqlite sqlitebrowser;
}
#
################################################################################
preRun()
{
   if [ -f "custominstall.sh" ]; then
      source "custominstall.sh";
   fi
   runMe;
}
#
################################################################################
installMaldet()
{
   sudo apt install -y inotify-tools;
   #sudo apt install -y maldet;
   if cd ~/Downloads; then
      mkdir -p maldet;
   fi
   if cd maldet; then
      wget http://www.rfxn.com/downloads/maldetect-current.tar.gz;
      tar -zxvf maldetect-current.tar.gz
   fi
   if cd maldetect-1.6.4; then
      sudo ./install.sh
      sudo gedit /usr/local/maldetect/conf.maldet
   fi
   # email_alert=1
   # email_addr="you@domain.com"
   # quarantine_hits=1
   # quarantine_clean=1
   # scan_ignore_root="0"
   # quarantine_suspend_user=1
   # quarantine_suspend_user_minuid=500
   # scan_clamscan=1
   sudo maldet -d && sudo maldet -u;
}
```

```
#
################################################################################
installMisc()
{
   print_caution "Install Misc" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
   sudo apt install -y lvm2;
   sudo apt install -y ufw ksshaskpass offlineimap gnome-keyring gufw;
   sudo systemctl start ufw.service || sudo systemctl restart ufw.service && sudo systemctl status
ufw.service;
   sudo systemctl enable ufw.service;
   sudo ufw enable;
   sudo apt install -y password-gorilla;
   sudo apt install -y aspell-en;
   sudo apt install -y ffmpeg rtmpdump atomicparsley python-pycryptodome youtube-dl
xbindkeys;
   sudo apt install -y sigil gnome-books;
   #
   sudo apt install -y translate-shell;
   sudo apt install -y audacity ardour brasero cheese dvd+rw-tools dvdauthor;
   sudo apt install -y ffmpeg gaupol kdenlive mjpegtools mpgtx mencoder openshot devede
gpicview pitivi;
   sudo apt install -y python-chardet;
   sudo apt install -y recordmydesktop xine-ui catdoc xjadeo harvid frei0r-plugins;
   sudo apt install -y kipi-plugins python-nose;
   sudo apt install -y faac gpac espeak faac antiword unrtf odt2txt txt2tags nrg2iso bchunk;
   sudo apt install -y mpv mplayer;
   sudo apt install -y clementine plume-creator gnucash calibre converseen mumble-server krita;
   sudo apt install -y install nomacs flameshot;
   #
   sudo add-apt-repository ppa:flatpak/stable
   sudo apt update && sudo apt upgrade -y
   sudo flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo;
   sudo flatpak remote-add --if-not-exists kdeapps --from
https://distribute.kde.org/kdeapps.flatpakrepo;
   sudo flatpak install -y kdeapps org.kde.okular;
   sudo flatpak install -y flathub de.haeckerfelix.Shortwave;
   sudo snap install rambox;
   #
   wget https://download.teamviewer.com/download/linux/signature/TeamViewer2017.asc
   sudo apt-key add TeamViewer2017.asc
   if ! grep -Fxq "http://linux.teamviewer.com/deb" /etc/apt/sources.list.d/teamviewer.list; then
      sudo sh -c 'echo "deb http://linux.teamviewer.com/deb preview main" >>
/etc/apt/sources.list.d/teamviewer.list'
      sudo apt update
   fi
```

```
    sudo apt install teamviewer -y;
    #wget https://download.teamviewer.com/download/linux/teamviewer_amd64.deb
    #sudo apt install ./teamviewer_amd64.deb
    sudo add-apt-repository ppa: qr-tools-developers/qr-tools-stable;
    sudo apt update;
    sudo apt install -y qtqr;
    # https://www.google.com/intl/en_in/earth/versions/#download-pro
    if [ -f google-earth-pro-stable_current_amd64.deb ]; then
        sudo dpkg -i google-earth-pro-stable_current_amd64.deb;
    fi
}
# END installMisc
#
################################################################################
installRepos()
{
    sudo add-apt-repository ppa:kubuntu-ppa/ppa;
    sudo add-apt-repository ppa:kubuntu-ppa/backports;
    sudo add-apt-repository ppa:mozillateam/ppa;
    sudo add-apt-repository ppa:libreoffice/ppa;
    sudo add-apt-repository ppa:lutris-team/lutris;
    sudo add-apt-repository ppa:team-xbmc/ppa;
    sudo add-apt-repository ppa:webupd8team/y-ppa-manager;
    sudo add-apt-repository ppa:atareao/telegram;
    sudo add-apt-repository ppa:atareao/atareao;
    sudo add-apt-repository ppa:stefanberger/swtpm-focal;
    sudo add-apt-repository ppa:git-core/ppa;
    sudo add-apt-repository ppa:forkotov02/ppa

    # AMD sudo add-apt-repository ppa:oibaf/graphics-drivers
    # Nvidia sudo add-apt-repository ppa:graphics-drivers/ppa
    sudo apt update  && sudo apt upgrade -y;
    #sudo apt update && sudo apt full-upgrade
}
#
################################################################################
# https://www.ubuntuupdates.org/ppas
# https://www.playdeb.net/
# https://github.com/virtio-win/virtio-win-pkg-scripts/blob/master/README.md
# https://wiki.ubuntu.com/Releases
#
runFirst()
{
    if [ "$ThisRunaptUpdate" -eq 0 ]; then refeshUpdateUpgrade; fi
    print_caution "Run First" "Set up fail2ban" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo apt install -y mdadm default-mta dracut-core;
```

```
    sudo apt install -y binutils make gcc pkg-config fakeroot gnupg mesa-utils;
    sudo apt install -y build-essential pkg-config wget gperf;
    sudo apt install -y gcc autoconf automake make cmake ruby perl autoconf bison flex autogen git texinfo bash;
    sudo apt install -y net-tools git openssl clang llvm python3 patch scons sed shellcheck;
    sudo apt install -y net-tools gettext libtool graphicsmagick;
    sudo apt install -y doxygen ghostscript tcl astyle uncrustify cvs bleachbit;
    sudo apt install -y fontconfig p7zip unzip zip lzip bzip2;
    sudo apt install -y bash bash-completion;
    sudo apt install -y wget rsync nano gucharmap clang fail2ban ed cppcheck tidy testdisk yarn seahorse;
    # tpm-tools
    sudo apt install -y swtpm swtpm-tools tpm2-abrmd
    #sudo mkdir -p /var/lib/libvirt/qemu/tpm
    #sudo swtpm socket --tpmstate dir=/var/lib/libvirt/qemu/tpm --ctrl type=unixio,path=/var/lib/libvirt/qemu/tpm/swtpm-sock --log level=20 --tpm2
    #sudo -E wget --output-document=/etc/apt/sources.list.d/medibuntu.list http://www.medibuntu.org/sources.list.d/$(lsb_release -cs).list && sudo apt --quiet update && sudo apt --yes --quiet --allow-unauthenticated install medibuntu-keyring && sudo apt --quiet update

    sudo apt update  && sudo apt upgrade -y
    # && sudo apt full-upgrade -y
    sudo apt install -y kodi
    sudo apt install -y lutris
    sudo apt install -y y-ppa-manager

    sudo apt install -y telegram
    sudo apt install -y chromium-browser

    sudo apt install -y synaptic
    sudo apt install -y gdebi
    sudo apt install -y ppa-purge

    sudo apt install -y gimp gimp-data gimp-plugin-registry gimp-data-extras

    # imagewriter
    sudo apt install -y  vlc git git-lfs xclip ntfs-3g;

    sudo apt install -y npm;
    sudo npm update; sudo npm install -g npm;

    sudo apt install -y dh-autoreconf libssl-dev libtasn1-6-dev pkg-config libtpms-dev net-tools iproute2 libjson-glib-dev libgnutls28-dev expect gawk socat libseccomp-dev make
```

```
    sudo apt install -y snapd
    # sudo apt install -y snap-store
    #installFlatPak;
    sudo apt install -y flatpak malcontent-gui
    sudo apt install -y plasma-discover-backend-flatpak
    sudo apt install -y build-essential software-properties-common software-properties-gtk
    sudo apt install -y synaptic dwww menu deborphan apt-xapian-index tasksel doc-debian dpkg-
www libmojolicious-perl menu-l10n;

    sudo apt install -y kubuntu-restricted-extras libreoffice libreoffice-style-breeze libreoffice-gtk3
libreoffice-kde
    sudo apt install -y smplayer clementine kup-backup gwenview kate gimp speedcrunch
chromium-browser ktorrent kfind
    sudo apt install -y partitionmanager kio-extras gufw krosspython geoip-bin sg3-utils;
    sudo apt install -y gedit gedit-plugins;
    # Fail 2 Ban
    if [ ! -f /etc/fail2ban/jail.local ]; then
        sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local;
        if ! sudoAppend "bantime = 3666h" /etc/fail2ban/jail.local; then
            echo "Error updating jail";
            sudo gedit /etc/fail2ban/jail.local;
        fi
        if ! sudoAppend "maxretry = 3" /etc/fail2ban/jail.local; then
            echo "Error updating jail";
            sudo gedit /etc/fail2ban/jail.local;
        fi
        if ! sudoAppend "ignoreip = 127.0.0.1/8 192.168.1.13 192.168.0.16" /etc/fail2ban/jail.local;
then
            echo "Error updating jail";
            sudo gedit /etc/fail2ban/jail.local;
        fi
    fi
    #
    sudo systemctl start fail2ban || sudo systemctl restart fail2ban; sudo systemctl status fail2ban;
    sudo systemctl enable fail2ban;
    # sudo systemctl stop fail2ban
    #
    sudo apt install -y mlocate appstream;
    sudo updatedb;
#     sudo apt install -y gnome-keyring-query;
    echo "Finished runFirst";
    #makeHome;
    if [ "$(cat /proc/sys/vm/swappiness)" == "60" ]; then
        sudoAppend "vm.swappiness=10" /etc/sysctl.conf;
    fi
    sudo systemctl enable fstrim.timer;
```

```
}
# END runFirst
#
################################################################################
installEmail()
{
   #sudo apt install -y kontact akonadi-backend-sqlite kleopatra spamassassin gnokii razor libdbi-
perl pyzor libencode-detect-perl libgeoip2-perl libnet-patricia-perl libbsd-resource-perl libclone-
perl libmldbm-perl libnet-daemon-perl libsql-statement-perl libtest-fatal-perl libbareword-
filehandles-perl libindirect-perl libmultidimensional-perl pinentry-doc pyzor-doc pinentry-gtk2
pinentry-qt4
   sudo apt install -y thunderbird thunderbird-gnome-support libotr5-bin libotr5
   sudo apt install -y evolution evolution-ews evolution-plugins-experimental geary;
   #sudo apt install kube
}
#
################################################################################
fixAmdGPU()
{
   #git clone https://kernel.googlesource.com/pub/scm/linux/kernel/git/firmware/linux-
firmware.git
   #git clone git://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git
   theGPU="$(lspci | grep VGA | cut -d ":" -f3)";
   # Advanced Micro Devices, Inc. [AMD/ATI] Ellesmere [Radeon RX
470/480/570/570X/580/580X/590] (rev ef)
   if [[ "$theGPU" == *"[AMD/ATI]"* ]]; then
      if [ -f amd.sh ]; then
         ./amd.sh
      fi
   fi
   #sudo update-initramfs -k all -u -v
}
#
################################################################################
# /media/jflesher/Linux/
# sudo mount /dev/sdfi /mnt
getUsb()
{
   # if [ -b /dev/disk/by-label/Linux ]; then echo "Found Linux USB"; fi
   if [ -b /dev/disk/by-label/Linux ]; then
      if [ ! -d /mnt/usb ]; then sudo mkdir -p sudo mkdir -p /mnt/usb; fi
      sudo mount /dev/disk/by-label/Linux /mnt/usb
      if cd /mnt/usb; then
         ls -las;
         echo "You are on the USB";
```

```
      fi
   fi
}
#
################################################################################
runHomeClean()
{
   if [ -n "$HOME_UUID" ]; then
      if sudo grep -Fxq "$HOME_UUID" /etc/fstab; then
         echo "fstabe is set for home"
      else
         cleanKDE;
         sudo apt install -y mdadm;
         if ! sudoAppend "$HOME_UUID" /etc/fstab; then
            echo "Error appending $HOME_UUID to /etc/fstab";
            sudo nano /etc/fstab;
            exit 1;
         fi
         echo "Inspect fstab before reboot";
         sudo nano /etc/fstab;
         sudo reboot;
      fi
   fi
   if [ "$theClearner" -eq 1 ]; then
      cleanKDE;
   fi
}
#
################################################################################
runThisStuff()
{
   echo "Running Stuff RUN_OPTIONS=${RUN_OPTIONS} and DoNotRun=$DoNotRun";
   if is_root_user; then echo "Do not run as root"; exit 1; fi
   local -i theClearner; theClearner=0;
   # Begin Script
   echo "";
   if [ "$ThisClearOk" -eq 0 ]; then cls; fi # clear;
   echo "";
   sudo_Required "${FUNCNAME[0]}";
   #
   if [ "${RUN_OPTIONS}" -eq 1 ] && [ "$DoNotRun" -eq 0 ]; then
      #
      # Start App belows
      echo "Run Stuff";
      #
      if [ "${TheRunFirst}" -eq 1 ] && [ "$DoNotRun" -eq 0 ]; then
```

```
echo "Run First-time";
#runHomeClean;

if [ ! -f "/mnt/readme2.txt" ]; then
   fixSudoTimeOut;
   installRepos;
   fixAmdGPU;
   sudo touch /mnt/readme2.txt;
   sudoAppend "Do not delete" /mnt/readme2.txt;
fi
refeshUpdateUpgrade;
runFirst; # makeHome;

#exit 1;
installDev;
installEmail;
installMaldet;
installGraphicCard;
installFonts;
installCodec;
installMisc;
installNpm;
installClamAV;
installGedit;
installArchive;
installInternet;
installEquipment;
#installBluetooth;
installJava;
installUtil;
#installSamba;
installSQLite;
installWine;
installCadGraphics;
installQt;
installGames;
installEducational;
installKVM;
installGem bundler;
installGem net-ssh;
installGem net-scp;
#installMxe;
#installAndroid;
#installPostgreSql;
#installUnrealEngine;
#installSTM32;
```

```
            #isDMAR;
            #rubyVersion;
            #installQuartus;
            #installSagemath;
            #installMakehuman;
            #installHaru;
        fi
        #
        # End App before here
        #
    elif [ "$TheRunScan" -eq 1 ]; then
        echo "TheRunScan=1";
        doScan;
    elif [ "$TheKvmFixBackup" -eq 1 ]; then
        kvmFixBackup;
    fi
    #
    if [ "${SetDebug}" -eq 1 ]; then set +x; fi # turn OFF debug mode
    #
    print_this "WIZ_BYE" "${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
    #
    tput sgr0; # Reset the Terminal on Exit
}
# END runThisStuff
#
################################################################################
startTime;
checkArgs "$@";
if [ -f "wizard.sh" ]; then
    echo -e "\033[0;32mLoading\033[0m wizard.sh ";
    #. "wizard.sh";
    # shellcheck disable=SC1091
    # shellcheck disable=SC1090
    source "wizard.sh";
else
    echo "File Not Found: wizard.sh - $(basename "${BASH_SOURCE[0]}") : ${LINENO[0]}";
    #exit 1;
fi
if [ -f "wizard-common.sh" ]; then
    echo -e "\033[0;32mLoading\033[0m wizard-common.sh ";
    #. "wizard-common.sh";
    # shellcheck disable=SC1091
    # shellcheck disable=SC1090
    source "wizard-common.sh";
else
```

```
    echo "File Not Found: wizard.sh - $(basename "${BASH_SOURCE[0]}") : ${LINENO[0]}";
    #exit 1;
fi
#
runThisStuff;
#
endTime;
#
################################################################################
# ********************************** End of Script ******************************
```

Appendix Arch Linux Manjaro

This is the Installation script for Arch Linux Manjaro

```
#!/bin/bash
#
# Copyleft and Written by Jeffrey Scott Flesher
# No Copyrights or Licenses
#
# Install Script for Archlinux Clone: Manjaro
# "${HOME}"/Scripts/install-manjaro.sh -w "${TheWorkSpace}/workspace/" -f
#
stat /dev/ttyS0;
declare DateManager="27 Oct 2021";
declare VersionManager="1.0";
declare -x TheScriptName; TheScriptName="install-manjaro";
declare -i QT_ONLINE; QT_ONLINE=0;
declare EMAIL_ADDRESS_NOTIFICATIONS;
EMAIL_ADDRESS_NOTIFICATIONS="jeffrey.scott.flesher@gmail.com";
declare -i CINNAMON_INSTALL; CINNAMON_INSTALL=0;
[[ -f ~/.bash_aliaeses ]] && source ~/.bash_aliaeses;
#
################################################################################

# $HOME/.config/systemd/user/ssh-agent.service
#[Unit]
#Description=SSH key agent

#[Service]
#Type=simple
#Environment=SSH_AUTH_SOCK=%t/ssh-agent.socket
#ExecStart=/usr/bin/ssh-agent -D -a $SSH_AUTH_SOCK

#[Install]
#WantedBy=default.target


#cp /etc/xdg/autostart/gnome-keyring-ssh.desktop ~/.config/autostart/
#cp /etc/xdg/autostart/gnome-keyring-pkcs11.desktop ~/.config/autostart/
#cp /etc/xdg/autostart/gnome-keyring-secrets.desktop ~/.config/autostart/
#
################################################################################
# Table of Content Index
# checkArgs
#
################################################################################
declare -x ThisRoofFolder; ThisRoofFolder="${HOME}";
if [ ! -d "${ThisRoofFolder}" ]; then
```

```
   mkdir -p "${ThisRoofFolder}";
fi
declare -x TheWorkSpace; TheWorkSpace="${ThisRoofFolder}/workspace/";
declare TheSambaPublicFolder; TheSambaPublicFolder="${ThisRoofFolder}/samba/public";
declare TheWindowsFolder; TheWindowsFolder="${ThisRoofFolder}/Windows";
#
declare -x TheWizardPath; TheWizardPath="${HOME}/Scripts"; export TheWizardPath;
declare ThisUnrealEngineGitFolder;
ThisUnrealEngineGitFolder="${ThisRoofFolder}/workspace/UnrealEngine";
declare ThisUnrealEngineLauncherGitFolder;
ThisUnrealEngineLauncherGitFolder="${ThisRoofFolder}/workspace/UE4Launcher";

declare TheCOW; TheCOW="${TheWindowsFolder}/COW"
declare -i ThisComputerNvidia; # FIXME command line
ThisComputerNvidia=1; # 1 = yes
#declare ThisOwner; ThisOwner="$(whoami)";
declare -ix ThisClearOk; ThisClearOk=1; export ThisClearOk;
# KVM
declare TheKvmImageFolder; TheKvmImageFolder="/var/lib/libvirt/images";
declare -i ThisCowBackUp; ThisCowBackUp=0;
#
################################################################################
#
# Rename Folders and Files with Space with a dot
# Runs encoder to optimize videos
#
# 3D to 2D
# ffmpeg -i './a3dmovie.mp4'  -vf "crop=w=iw/2:h=ih:x=0:y=0,scale=w=2*iw:h=ih,setdar=2" -y
./a2dmovie.mp4
#
# For Test:
# cd "${HOME}/Downloads/Test.install-home/"
# cd /mnt/BaseFolder
# chmod +x "${HOME}"/Scripts/install-manjaro.sh
# dos2unix "${HOME}"/Scripts/install-manjaro.sh
# clear; echo "Shell Check..."; shell?check "${HOME}"/Scripts/install-manjaro.sh
# shell?check -x .${TheWorkSpace}/install-manjaro.sh
# shell?check -x ".${TheWorkSpace}/install-manjaro.sh && { shell?check -x
".${TheWorkSpace}/wizard.sh"; }  && { shell?check -x ".${TheWorkSpace}/wizard-
common.sh"; }

# "${HOME}"/Scripts/install-manjaro.sh -w "${HOME}/workspace/"
# New Install
# "${HOME}"/Scripts/install-manjaro.sh -w "${HOME}/workspace/" -f
# KVM Backup
# "${HOME}"/Scripts/install-manjaro.sh -w "${TheWorkSpace}/workspace/" -k
```

```
# Malware
# "${HOME}"/Scripts/install-manjaro.sh -w "${TheWorkSpace}/workspace/" -m 2
#
################################################################################
#
declare ThisRunPacmanFast; ThisRunPacmanFast=0;
#
set -u; # same as  set -o nounset, error if variable is not set
# nocaseglob: If set, Bash matches filenames in a case-insensitive fashion when performing
filename expansion.
# dotglob: If set, Bash includes filenames beginning with a `.' in the results of filename
expansion.
# -s   Enable (set) each optname
shopt -s nullglob dotglob;
IFS=$' '; # IFS=$'\n\t'; IFS=$'\n';
# User's file creation mask. umask sets an environment variable which automatically sets file
permissions on newly created files.
#   i.e. it will set the shell process's file creation mask to mode.
umask 022;
unalias -a; # -a   Remove All aliases; so cp is not cp -i
# -f The names refer to shell Functions, and the function definition is removed.
#   Readonly variables and functions may not be unset
unset -f "$(declare -F | sed "s/^declare -f //")";
#
#trap "echo Exited!; exit;" SIGINT SIGTERM;
trap "echo install-home Control Break; exit;"     SIGINT;
trap "echo install-home SIGTERM-d; exit;"         SIGTERM;
trap "echo install-home Exited with Error; exit;" EXIT;
trap "echo install-home Finished normally; exit;" 0;
#
#
# all Public Variables
declare TheFullScriptPath; TheFullScriptPath="$(dirname "$(readlink -f "${0}")")"; export
TheFullScriptPath; # No Ending /
declare -i SimulateThis;   SimulateThis=0; export SimulateThis;      # 1=true or 0=false, Simulate
rename
declare -i PrintDebug;     PrintDebug=0;   export PrintDebug;       # 1=true or 0=false
declare -i SetDebug;       SetDebug=0;     export SetDebug;         # 1=true or 0=false
#
declare -ix TheDebugging; TheDebugging=0;  export TheDebugging;
#
#
################################################################################
#
# Where do I put these files in Misc
```

```
declare TheLocalizedPathFolderName;  TheLocalizedPathFolderName="locale"; export
TheLocalizedPathFolderName;      # No slash at end
declare -x TheLocalizedPath;
TheLocalizedPath="${TheFullScriptPath}/${TheLocalizedPathFolderName}"; export
TheLocalizedPath;
declare -x TheDefaultLanguage;          TheDefaultLanguage="en";                # I wrote this in
English so this is a constant
# Where the po.langage file goes -> Edit this line as needed for project
# Name to call the Langage File referanced above -> Edit this line as needed for project
declare -x TheLocalizedFile; TheLocalizedFile="$TheScriptName";
# Set to 1 the first time you start Hand Translating your Localization files so they do not get
overwriten.
declare -ix TheLocalizedFilesSafe; TheLocalizedFilesSafe=1; export TheLocalizedFilesSafe;
# Change this to the Default Language that the Localized files are in
declare -x TheDefaultLocalizedLanguage; TheDefaultLocalizedLanguage='en'; export
TheDefaultLocalizedLanguage;
#
# Localization of all script tags to support language
declare -ix TheRunLocalizer; TheRunLocalizer=0; export TheRunLocalizer;
# Multilingual Langage File Path -> from above: declare -r
TheLocalizedPath="${TheFullScriptPath}/locale"
TEXTDOMAINDIR="${TheLocalizedPath}";
export TEXTDOMAINDIR;
# Multilingual Langage File Name -> from above: declare TheLocalizedFile
TEXTDOMAIN="${TheLocalizedFile}";
export TEXTDOMAIN;
# Create Help.html
declare -ix TheRunHelp; TheRunHelp=0; export TheRunHelp;
# 0=Disable, 1=Run, 2=Run Extended Test
declare -ix TheRunTest; TheRunTest=0; export TheRunTest;
# Automatically install from saved settings
declare -ix TheAutoMan; TheAutoMan=0; export TheAutoMan;
# 14-Jan-2013 @ 06:32:36 PM
# shellcheck disable=SC2034
declare -x TheDateTime; TheDateTime="$(date +"%d-%b-%Y @ %r")"; export TheDateTime;
# Day-Mon-YYYY-T-HH-MM: 14-Jan-2013-T-18-32-36
# shellcheck disable=SC2034
declare -x TheLogDateTime; TheLogDateTime=$(date +"%d-%b-%Y-T-%H-%M-%S"); export
TheLogDateTime;
#
################################################################################
#
#
################################################################################
#
# change to project needs
```

```
declare -x TheConfigName;  TheConfigName="install-home";                    export
TheConfigName;
declare -x TheLogPath;     TheLogPath="${TheFullScriptPath}/Support/Logs";          export
TheLogPath;
declare -x TheConfigPath;  TheConfigPath="${TheFullScriptPath}/Support/Config";
export TheConfigPath;
declare -x TheErrorLog;    TheErrorLog="${TheLogPath}/0-${TheConfigName}-error.log";
export TheErrorLog;
declare -x TheActivityLog; TheActivityLog="${TheLogPath}/1-${TheConfigName}-
activity.log"; export TheActivityLog;
#
################################################################################
makeHome()
{
   if [ -f "fstab.txt" ]; then
      if [ ! -f "/var/log/wizardscript/fstab.txt" ]; then
         sudo mkdir "/var/log/wizardscript";
         sudo touch "/var/log/wizardscript/fstab.txt";
         sudo cat "fstab.txt" >> /etc/fstab;
         sudo mount -a;
         echo "Reboot required";
      fi
   fi
}
#
################################################################################
# show_help (0=Text, 1=HTML)
show_help()
{
   [[ $# -ne 1 ]] && { echo "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   getHorizontalLineType "$1";
   fixLineType "$1" "Help";
   fixLineType "$1" "";
   fixLineType "$1" "Expected Argument Switches:";
   fixLineType "$1" "-h or --help";
   fixLineType "$1" "-l or --localize";
   fixLineType "$1" "-p or --printdebug";
   fixLineType "$1" "-s or --simulate";
   fixLineType "$1" "-t or --test";
   fixLineType "$1" "-v or --version";
   fixLineType "$1" "";
   fixLineType "$1" "Tests";
   fixLineType "$1" "Test 1: all common functions";
   fixLineType "$1" "Test 2: Internet and user account functions";
```

```
   fixLineType "$1" "Test 3: Keyboard Input functions";
   fixLineType "$1" "Test 4: Select File function";
   fixLineType "$1" "Test 6: Network and Group functions 5: Keyboard Input Options";
   fixLineType "$1" "Test 7: SQLite";
   fixLineType "$1" "";
}
# END show_help
#
################################################################################
#
declare -i IsBeeper;         IsBeeper=0;         # Beep after each encoding
declare -i TheRunCheck;      TheRunCheck=0;
declare -i TheRunFixNPM;     TheRunFixNPM=0;
declare -i TheRunFirst;      TheRunFirst=0;
declare -i TheRunFixSTO;     TheRunFixSTO=0;
declare -i TheRunSetXclip;   TheRunSetXclip=0;
declare -i TheRunScan;       TheRunScan=0;
declare -i TheScanLevelAV;   TheScanLevelAV=0;
declare -i TheKvmFixBackup;  TheKvmFixBackup=0;
#
################################################################################
declare -i DoNotRun; DoNotRun=0;
#
declare -i RUN_OPTIONS; RUN_OPTIONS=0;
declare -x PROGNAME;   PROGNAME="${0##*/}";    #
#                              b d e g i j q u r x z
declare -x SHORTOPTS;  SHORTOPTS="v,h,p,t:,s:,b,l,w:,a,c,n,f,o,x,m:,y,k";
declare -x LONGOPTS;
LONGOPTS="version,help,printdebug,test,simulate,beep,localize,workspace,annotation,check,n
pmfix,first,outtimer,xclip,malavscan,yelp,kvmfixbackup";
declare -x ARGS;       ARGS=$(getopt -s bash --options "${SHORTOPTS}" --longoptions
"${LONGOPTS}" --name "${PROGNAME}" -- "$@");
eval set -- "$ARGS";
#
################################################################################
# checkArgs "$@"
checkArgs()
{
   #
         while true; do
            case $1 in
               -v|--version)
                  echo "Script Date: ${DateManager} Version: ${VersionManager}";
                  exit 0;
                  ;;
               -h|--help)
```

```
            usage;
            ;;
        -a|--annotation)
            TheRunHelp=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "TheRunHelp"; fi
            ;;
        -p|--printdebug)
            PrintDebug=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "printdebug"; fi
            ;;
        -t|--test)
            shift;
            TheRunTest=$((${1} + 0));
            [ ${TheRunTest} -lt 1 ] || [ ${TheRunTest} -gt 10 ] && usage;
            echo "Run Test: ${1}";
            ;;
        -s|--simulate)
            shift;
            SimulateThis="${1}"; # 0=False, 1=True
            if [ "${PrintDebug}" -eq 1 ]; then echo "simulate=${1}"; fi
            ;;
        -b|--beep)
            IsBeeper=1;
            if [ "${PrintDebug}" -eq 1 ]; then echo "IsBeeper"; fi
            ;;
        -l|--localize)
            TheRunLocalizer=1;
            if [ ! -d "$TheLocalizedPath" ]; then mkdir -p "$TheLocalizedPath"; fi
            if [ "${PrintDebug}" -eq 1 ]; then echo "localize"; fi
            ;;
        -w|--workspace)
            shift;
            TheWorkSpace="${1}"; #
        #
        if [[ "$TheWorkSpace" != */ ]]; then TheWorkSpace="${TheWorkSpace}/"; fi # Make
sure it ends with a /
        #
            if [ "${PrintDebug}" -eq 1 ]; then echo "workspace=${1}"; fi
            ;;
        -c|--check)
            TheRunCheck=1;
            pacmanFast;
            if [ "${PrintDebug}" -eq 1 ]; then echo "check"; fi
            ;;
        -n|--npmfix)
            TheRunFixNPM=1;
```

```
                fixNpm;
                if [ "${PrintDebug}" -eq 1 ]; then echo "npmfix"; fi
                ;;
            -f|--first)
                TheRunFirst=1;
                if [ "${PrintDebug}" -eq 1 ]; then echo "Run First-time"; fi
                ;;
            -o|--outtimer)
                TheRunFixSTO=1;
            fixSudoTimeOut;
                if [ "${PrintDebug}" -eq 1 ]; then echo "outtimer"; fi
                ;;
            -x|--xclip)
                TheRunSetXclip=1;
            setXclip;
                if [ "${PrintDebug}" -eq 1 ]; then echo "xclip"; fi
                ;;
            -m|--malavscan)
                shift;
                TheRunScan=1;
                TheScanLevelAV=$((${1} + 0));
                [ ${TheScanLevelAV} -lt 1 ] || [ ${TheScanLevelAV} -gt 10 ] && usage;
                if [ "${PrintDebug}" -eq 1 ]; then echo "malavscan"; fi
                ;;
            -y|--yelp)
                echo "sudo required to run timeshift restore, this will reset the the OS back in
time.";
                read_input_yn_c "Run timeshift restore" "" 1;
            if [ "${YN_OPTION}" -eq 1 ]; then
                sudo timeshift --restore; sudo reboot;
              fi
              #
                if [ "${PrintDebug}" -eq 1 ]; then echo "yelp"; fi
                ;;
            -k|--kvmfixbackup)
                TheKvmFixBackup=1;
                if [ "${PrintDebug}" -eq 1 ]; then echo "kvmfixbackup"; fi
                ;;
            --) shift; break; ;;
             *) shift; break; ;;
          esac; shift;
        done
  #
  if [ "$TheWorkSpace" == "" ]; then
    echo;
    echo "TheWorkSpace=$TheWorkSpace";
```

```
      echo;
      usage;
   else
      RUN_OPTIONS=1;
   fi
   #
   [[ "$TheKvmFixBackup" -eq 1 ]] && DoNotRun=0;
   [[ "$TheRunFirst" -eq 1 ]]     && DoNotRun=0;
   [[ "$TheRunTest" -eq 1 ]]      && DoNotRun=0;
   [[ "$TheRunLocalizer" -eq 1 ]] && DoNotRun=0;
   [[ "$TheRunHelp" -eq 1 ]]      && DoNotRun=0;
   [[ "$TheRunCheck" -eq 1 ]]     && DoNotRun=0;
   [[ "$TheRunFixNPM" -eq 1 ]]    && DoNotRun=0;
   [[ "$TheRunFixSTO" -eq 1 ]]    && DoNotRun=0;
   [[ "$TheRunSetXclip" -eq 1 ]]  && DoNotRun=0;
   [[ "$TheRunScan" -eq 1 ]]      && DoNotRun=0;
   #
   if [ "$DoNotRun" -eq 0 ]; then
      printDoNotRun "TheKvmFixBackup" "$TheKvmFixBackup";
      printDoNotRun "TheRunFirst" "$TheRunFirst";
      printDoNotRun "TheRunTest" "$TheRunTest";
      printDoNotRun "TheRunLocalizer" "$TheRunLocalizer";
      printDoNotRun "TheRunCheck" "$TheRunCheck";
      printDoNotRun "TheRunFixNPM" "$TheRunFixNPM";
      printDoNotRun "TheRunFixSTO" "$TheRunFixSTO";
      printDoNotRun "TheRunSetXclip" "$TheRunSetXclip";
      printDoNotRun "TheRunScan" "$TheRunScan";
   fi
}
# END checkArgs
#
################################################################################
# https://wiki.archlinux.org/index.php/SELinux
SELinux()
{
   print_caution "SELinux" "SELinux" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   pacmanFast;
   #
#    # 1.
#    yay --noconfirm --needed -S libsepol libselinux checkpolicy secilc setools libsemanage
semodule-utils policycoreutils;
#    # 2.
#    yay --noconfirm --needed -S pambase-selinux pam-selinux;
#    # 3. Test to make sure you can still login FIXME
#
#    # 4.
```

```
#    yay --noconfirm --needed -S coreutils-selinux findutils-selinux iproute2-selinux logrotate-selinux openssh-selinux psmisc-selinux shadow-selinux cronie-selinux;
#    # 5.
#    sudoSELinux;
#    # 6.
#    yay --noconfirm --needed -S systemd-selinux systemd-libs-selinux util-linux-selinux util-linux-libs-selinux;
#    yay --noconfirm -Si systemd-selinux;
#    # 7.
#    yay --noconfirm --needed -S dbus-selinux;
#    yay --noconfirm --needed -S selinux-alpm-hook;
#    #
#    yay --noconfirm --needed -S selinux-refpolicy-arch;
#    # or
#    # yay --noconfirm --needed -S selinux-refpolicy-git;
#    #
#    yay --noconfirm --needed -S mcstrans restorecond
#    yay --noconfirm --needed -S selinux-dbus-config selinux-gui selinux-python selinux-python2 selinux-sandbox
#    yay --noconfirm --needed -S selinux-refpolicy-src
    if [ -d "${TheWorkSpace}" ]; then
        if cd "${TheWorkSpace}"; then
            git clone https://github.com/archlinuxhardened/selinux;
            if cd selinux; then
                ./recv_gpg_keys.sh;
                ./build_and_install_all.sh;
            fi
        fi
    fi
}
#
################################################################################
# https://wiki.archlinux.org/index.php/SELinux
#
sudoSELinux()
{
    # backup your /etc/sudoers
    if [ -f /etc/sudoers ]; then
        if [ ! -f /etc/sudoers.bak ]; then
            sudo cp /etc/sudoers /etc/sudoers.bak;
        fi
    fi
    #
    yay --noconfirm --needed -S sudo-selinux;
    #
    # it is overridden when sudo-selinux package is installed as a replacement of sudo
```

```
    if [ -f /etc/sudoers.bak ]; then
        if sudo /usr/bin/cp -f /etc/sudoers.bak /etc/sudoers; then
            sudo /usr/bin/rm -f /etc/sudoers.bak;
        fi
    fi
}
#
################################################################################
# useage
usage() { show_help 0; exit 1; }
#
################################################################################
# printDoNotRun "Var-Name" "Value";
printDoNotRun()
{
    if [ "${2}" -eq 1 ]; then
        echo "${1}=${2}";
    fi
}
# END printDoNotRun
#
################################################################################
pacmanFast()
{
    print_caution "Software Updates" "pacman setting coutry" "@ ${FUNCNAME[0]}() :
${LINENO[0]}";
    sudo pacman-mirrors --country United_States;
    print_caution "Software Updates" "fast-track before updates" "@ ${FUNCNAME[0]}() :
${LINENO[0]}";
    sudo pacman-mirrors --fasttrack 6;
    print_caution "Software Updates" "pacman updates" "@ ${FUNCNAME[0]}() :
${LINENO[0]}";
    sudo pacman --noconfirm --needed -Syyu;
    print_caution "Software Updates" "yay updates" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    yay --noconfirm --needed -Suyya;
    #sudo pacman --noconfirm --needed -Suy;
    #
    ThisRunPacmanFast=1;
}
# END pacmanFast
#
################################################################################
# I tried to make this AMD or Nvidia, AMD will ignore modprobe
installGraphicCard()
{
    return;
```

```
    # Install NVIDIA Drivers
    inxi -G
    if ! sudo mhwd -a pci nonfree 0300; then
      echo "Graphic Card did not install.";
      return;
      #sudo mhwd-kernal -r linux59;
       if ! sudo mhwd -a pci nonfree 0300; then
          echo "Graphic Card did not install.";
          exit 1;
       fi
    fi
    mhwd -li

    #sudo gedit /etc/mkinitcpio.conf
    # delete the word nouveau from the following line:
    #MODULES=" nouveau"
    #MODULES=""
    #sudo mkinitcpio -p [linux kernel version]
    #sudo mkinitcpio -p linux310

    if [ "$ThisComputerNvidia" == "Xeon" ]; then
       #sudo chown ${USER}:${USER} ~/.xinitrc
       # Hit the 'Save to X Configuration File' button and save to /etc/X11/mhwd.d/nvidia.conf
       #sudo mhwd-gpu --setmod nvidia --setxorg /etc/X11/mhwd.d/nvidia.conf
       #sudo gedit ~/.xinitrc
       #nvidia-settings --load-config-only
       #exec $(get_session)
       #sudo nvidia-settings
       # Remove the NVIDIA driver
       #sudo mhwd -r pci video-nvidia
       #sudo mhwd -r pci video-nvidia
       print_caution "Software Updates" "nvidia-modprobe" "@ ${FUNCNAME[0]}() :
${LINENO[0]}";
       nvidia-modprobe;
    fi
}
#
################################################################################
# FIXME
setXclip()
{
    print_caution "Set xclip alias" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo pacman --noconfirm --needed -S xclip;
    if [ ! -f ~/.bash_aliases ]; then
       touch ~/.bash_aliase;
       echo '#!/bin/bash' > ~/.bash_aliase;
```

```
        echo 'alias setclip="xclip -selection c";' > ~/.bash_aliase;
        echo 'alias getclip="xclip -selection c -o";' >> ~/.bash_aliase;
    fi
    alias setclip="xclip -selection c";
    alias getclip="xclip -selection c -o";
}
# END setXclip
#
################################################################################
runFirst()
{
    if [ "$ThisRunPacmanFast" -eq 0 ]; then pacmanFast; fi
    print_caution "Run First" "Set up fail2ban" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo pacman --noconfirm --needed -S binutils make gcc pkg-config fakeroot gnupg;
    sudo pacman --noconfirm --needed -S base base-devel multilib-devel cmake make pkg-config
wget gperf icu harfbuzz harfbuzz-icu mesa lib32-mesa;
    sudo pacman --noconfirm --needed -S gcc autoconf automake make cmake ruby perl glibc
lib32-glibc libc++ autoconf bison flex autogen git texinfo bash zlib;
    sudo pacman --noconfirm --needed -S net-tools inetutils git lib32-openssl openssl clang lib32-
clang llvm ninja python patch scons sed shellcheck;
    sudo pacman --noconfirm --needed -S gcc autoconf automake net-tools inetutils gettext libtool
libtiff zlib gd graphicsmagick;
    sudo pacman --noconfirm --needed -S doxygen ghostscript tcl astyle uncrustify cvs bleachbit;
    sudo pacman --noconfirm --needed -S fontconfig freetype2 lib32-fontconfig lib32-freetype2
xorg-fonts-type1 p7zip unzip zip lzip bzip2;
    sudo pacman --noconfirm --needed -S bash bash-bats bash-bats-assert bash-bats-support bash-
completion;
    sudo pacman --noconfirm --needed -S pacman-contrib yay wget rsync nano glibc libc++
libgit2-glib gucharmap clang fail2ban ed cppcheck tidy testdisk yarn seahorse libsecret;
    sudo pacman --noconfirm --needed -S imagewriter vlc gtk-engines git git-lfs nfs-utils xclip
ntfs-3g;
    sudo pacman --noconfirm --needed -S npm;
    sudo npm update; sudo npm install -g npm;
    # Fail 2 Ban
    if [ ! -f /etc/fail2ban/jail.local ]; then
        sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local;
        printf  "bantime = 3666h\n maxretry = 3\nignoreip = 127.0.0.1/8 192.168.1.13
192.168.0.16";
        sudo gedit /etc/fail2ban/jail.local;
        #if ! sudoAppend "bantime = 3666h\n maxretry = 3\nignoreip = 127.0.0.1/8 192.168.1.13
192.168.0.16" /etc/fail2ban/jail.local; then echo "Error updating jail"; sudo gedit
/etc/fail2ban/jail.local; fi
    fi
    #
    sudo systemctl start fail2ban || sudo systemctl restart fail2ban;
    sudo systemctl status fail2ban;
```

```
    sudo systemctl enable fail2ban;
    # sudo systemctl stop fail2ban
    #
    sudo pacman --noconfirm --needed -S mlocate appstream;
    sudo updatedb;
#     yay --noconfirm --needed -S gnome-keyring-query;

    #makeHome;

    if [ ! -f "/etc/sysctl.d/100-manjaro.conf" ]; then
        echo "vm.swappiness=10" | sudo tee /etc/sysctl.d/100-manjaro.conf
    fi
    sudo systemctl enable fstrim.timer;
    echo "Finished runFirst";
}
# END runFirst
#
################################################################################
installRust()
{
    cd "${HOME}";
    curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh;
    rustup set profile default;
    rustup toolchain install stable;
    rustup default stable;
    rustup update;
    rustup check;
    cargo update;
    rustc --version;
    rustup show;
    rustup self update;
}
#
################################################################################
installWasm()
{
    sudo npm install -g assemblyscript http-server;
    cd "${HOME}/emsdk";
    ./emsdk install latest;
    ./emsdk activate latest;
    # https://code.qt.io/cgit/qt/qt5.git/
    if [ -f "${HOME}/workspace/qt5" ]; then
        rm -rf "${HOME}/workspace/qt5";
    fi
    git clone https://code.qt.io/cgit/qt/qt5.git -b 5.15.2;
    #git clone https://code.qt.io/qt/qt5.git -b 5.15.2;
```

```
    cd qt5;
    ./init-repository -f --module-subset=qtbase,qtcharts,qtsvg;
    cd ..;
    if [ -f "${HOME}/workspace/qt5_shadow" ]; then
        rm -rf "${HOME}/workspace/qt5_shadow";
    fi
    mkdir qt5_shadow;
    cd qt5_shadow;
    ../qt5/configure -opensource -confirm-license -xplatform wasm-emscripten -feature-thread -
nomake examples -no-dbus -no-ssl -prefix $PWD/../qt5_wasm_binaries;
    make module-qtbase module-qtsvg module-qtcharts -j8;
    sudo make install -j8;

}
#
################################################################################
updateFonts()
{
    print_caution "Update Fonts" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    fc-list;
    fc-cache;
    fc-match;
}
# END updateFonts
#
################################################################################
installFonts()
{
    print_caution "Install Fonts" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    # Fonts
    sudo pacman --noconfirm --needed -S xorg-font-util xorg-fonts-75dpi xorg-fonts-encodings
xorg-fonts-misc xorg-fonts-misc xorg-fonts-type1 xorg-fonts-cyrillic xorg-fonts-alias;
    sudo pacman --noconfirm --needed -S fontconfig xorg-xlsfonts awesome-terminal-fonts;
    sudo pacman --noconfirm --needed -S ttf-liberation ttf-ubuntu-font-family noto-fonts ttf-
roboto ttf-droid ttf-dejavu ttf-croscore ttf-bitstream-vera font-bh-ttf;
    sudo pacman --noconfirm --needed -S ttf-anonymous-pro ttf-cascadia-code ttf-fira-mono gnu-
free-fonts ttf-linux-libertine;
    sudo pacman --noconfirm --needed -S adobe-source-code-pro-fonts adobe-source-sans-pro-
fonts;
    sudo pacman --noconfirm --needed -S xorg-fonts-type1 font-mathematica;
    sudo pacman --noconfirm --needed -S dina-font tamsyn-font terminus-font bdf-unifont
terminus-font-otb ttf-fantasque-sans-mono;
    #yay --noconfirm --needed -S cairo-infinality fontconfig-infinality freetype2-infinality;
    yay --noconfirm --needed -S ttf-ms-fonts;
    #
    yay --noconfirm --needed -S ttf-tahoma all-repository-fonts;
```

```
    yay --noconfirm --needed -S ttf-vista-fonts;
    updateFonts;
}
# END installFonts
#
################################################################################
#
installSagemath()
{
    print_caution "Install sagemath" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo pacman --noconfirm --needed -S sagemath;
    sudo pacman --noconfirm --needed -S octave;
    sudo pacman-key --recv-keys AE5A7FB608A0221C;
    yay --noconfirm --needed -S scilab;
}
# END installSagemath
#
################################################################################
#
installQuartus()
{
    print_caution "Install Quartus" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    yay --noconfirm --needed -S tcllib ncurses5-compat-libs lib32-ncurses5-compat-libs quartus-
free-quartus;
}
# END installQuartus
#
################################################################################
# Adding user '$USER' to user-group 'sambashare'
installSamba()
{
    print_caution "Install Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    sudo pacman --noconfirm --needed -S samba smbclient manjaro-settings-samba gvfs-smb
nemo-share;
    configSamba;
}
# END installSamba
#
################################################################################
# 10.0.2.4 by default
# Samba name server XEOFF is now a local master browser for workgroup WORKGROUP on
subnet 192.168.122.1
# Samba name server XEOFF is now a local master browser for workgroup WORKGROUP on
subnet 192.168.1.13
# 192.168.122.1 255.255.255.0
# <range start="192.168.122.2" end="192.168.122.254"/>
```

```
configSamba()
{
   print_caution "Configure Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # Samba
   sudo mkdir -p "${TheSambaPublicFolder}";
   sudo chown -R nobody:nobody "${TheSambaPublicFolder}";

   #sudo chmod -R 0775 "${TheSambaPublicFolder}";


   #change the owner to sambausers
   sudo chown -R root:sambausers "${TheSambaPublicFolder}";

   #Give permission of the share
   sudo chmod 1770 "${TheSambaPublicFolder}";

   ###
   echo "";
   printf "Paste Clipboard into File: [Public]\npath = ${TheSambaPublicFolder}\nbrowsable
=yes\nwritable = yes\nguest ok = yes\nread only = no\nforce user = nobody\ncreate mask =
0700\ndirectory mask = 0700\n";
   pushClipboard "[Public]\npath = ${TheSambaPublicFolder}\nbrowsable =yes\nwritable =
yes\nguest ok = yes\nread only = no\nforce user = nobody\ncreate mask = 0700\ndirectory mask
= 0700" 1;
   sudo gedit /etc/samba/smb.conf;
   ###
   # Create a new group called sambausers
   add_group sambausers; # Checks to see if it exists first

   sudo ufw allow CIFS;
   # Add "${USER}" to the sambausers group
   sudo passwd sambausers -a "${USER}";

   # create new password for user "${USER}";
   sudo smbpasswd -a "${USER}";


   echo "";
   printf "Paste Clipboard into File: [Samba]\ntitle=LanManager-like file and printer server for
Unix\ndescription=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.\nports=137,138/udp|139,445/tcp\n";
   pushClipboard "[Samba]\ntitle=LanManager-like file and printer server for
Unix\ndescription=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
```

```
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.\nports=137,138/udp|139,445/tcp" 1;
   #[Samba]
   #title=LanManager-like file and printer server for Unix
   #description=The Samba software suite is a collection of programs that implements the
SMB/CIFS protocol for unix systems, allowing you to serve files and printers to Windows, NT,
OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or
NetBIOS protocol.
   #ports=137,138/udp|139,445/tcp
   sudo gedit /etc/ufw/applications.d/samba;

   sudo ufw app update Samba;
   sudo ufw allow Samba;
   restart_Samba;

   #/usr/bin/qemu-kvm -m 1024 -name f15 -drive file=/images/f15.img,if=virtio
   #-fsdev local,security_model=passthrough,id=fsdev0,path=/tmp/share -device virtio-9p-
pci,id=fs0,fsdev=fsdev0,mount_tag=hostshare


   #    mount -t 9p -o trans=virtio,version=9p2000.L hostshare /tmp/host_files

   # on Windows
   #notepad C:\\Windows\System32\drivers\etc\hosts
   #192.168.1.13        Xeon.localhost          Xeon
}
# END configSamba
#
################################################################################
restart_Samba()
{
   print_caution "Start or Restart Samba" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
   sudo systemctl start smb.service || sudo systemctl restart smb.service;
   sudo systemctl enable smb.service;
   sudo systemctl status smb.service;
   #
   sudo systemctl start nmb.service || sudo systemctl restart nmb.service;
   sudo systemctl enable nmb.service;
   sudo systemctl status nmb.service;
   #
   sudo systemctl start smb.service || sudo systemctl restart smb.service;

   sudo systemctl status smb.service; sudo systemctl status nmb.service;
}
# END restart_Samba
```

```
#
################################################################################
launchBrowser()
{
   [[ $# -ne 1 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   local thisURL;  thisURL="$1";
   local thisPath; thisPath="$(which xdg-open || which gnome-open)";
   if [ -x "${BROWSER}" ]; then thisPath="${BROWSER}"; fi
   if [ -x "${thisPath}" ]; then
      "${thisPath}" "${thisURL}"; return "$?";
   fi
   echo "Can't find browser";
   return 1;
}
# END launchBrowser
#
################################################################################
kvmTest()
{
   echo "";echo "KVM Test Started";echo "";
   # CPU is 64, 32 or other
   CPU_ARCH=$(uname -m);
   # Check for AMD and Intel support
   CPU_TYPE="";
   if grep -n -e 'model name' /proc/cpuinfo | head -1 | grep -iq intel; then CPU_TYPE="Intel";
else CPU_TYPE="AMD"; fi

   echo "CPU: ${CPU_TYPE}:${CPU_ARCH}";

   # Check for UEFI and set var for display
   UEFI_INSTALL="";
   if [ -d /sys/firmware/efi/ ]; then UEFI_INSTALL="/UEFI"; fi
   # kvm-ok check
   if ! command -v kvm-ok &> /dev/null; then yay --noconfirm --needed -S cpu-checker-bzr; fi
   # check kvm-ok output if found: INFO: /dev/kvm exists\nKVM acceleration can be used
   KVM_OK="";
   if kvm-ok | grep "INFO: /dev/kvm exists" &> /dev/null; then echo "KVM Enabled";
KVM_OK="KVM Enabled"; else echo "KVM \033[33;5;7mDisabled\033[0m";
KVM_OK="KVM Disabled"; fi
   KVM_INSTALLED="";
   if systool -m kvm_amd -v &> /dev/null; then
      echo "AMD-V is enabled in the BIOS${UEFI_INSTALL}."
      KVM_INSTALLED="AMD-V Enabled and AMD IOMMU Disabled";
```

```
      if compgen -G "/sys/kernel/iommu_groups/*/devices/*" > /dev/null; then echo "AMD
IOMMU is enabled in the BIOS${UEFI_INSTALL}"; KVM_INSTALLED="AMD-V and
AMD IOMMU Enabled"; fi
   elif systool -m kvm_intel -v &> /dev/null ; then
      echo "Intel VT-X is enabled in the BIOS${UEFI_INSTALL}."
      KVM_INSTALLED="Intel VT-X Enabled and VT-D Disabled";
      if compgen -G "/sys/kernel/iommu_groups/*/devices/*" > /dev/null; then echo "Intel VT-D
is enabled in the BIOS${UEFI_INSTALL}"; KVM_INSTALLED="Intel VT-X and Intel VT-D
Enabled"; fi
   else
      if sudo dmesg | grep DMAR &> /dev/null; then echo "DMAR"; else echo "No DMAR"; fi
   fi
   # List IOMMU Groups
   if [ "$(ls -A /sys/kernel/iommu_groups)" ]; then
      echo "ERROR";
      shopt -s nullglob;
      for g in $(find /sys/kernel/iommu_groups/* -maxdepth 0 -type d | sort -V); do
         echo "IOMMU Group ${g##*/}:"
         for d in "$g/devices"/*; do
            echo -e "\t$(lspci -nns "${d##*/}")"
         done;
      done;
   fi
   echo ""; echo "KVM Test Finished: ${CPU_TYPE}:${CPU_ARCH} - ${KVM_OK} -
${KVM_INSTALLED} "; echo "";
}
#
################################################################################
# if [ "$TheKvmFixBackup" -eq 1 ]; then kvmFixBackup; fi
kvmFixBackup()
{
   print_caution "Running kvmFixBackup..." "Backup Cow=${ThisCowBackUp} and using
KVM Image Folder ${TheKvmImageFolder}" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   kvmTest;
   # sudo chown -R root:kvm /var/lib/libvirt/images
   if [ -d "${TheKvmImageFolder}" ]; then
      sudo chown -R "${USER}":kvm "${TheKvmImageFolder}"; # set folder permissions
      sudo chmod a+x "${TheKvmImageFolder}";              #
      sudo chmod -R a+x "${TheKvmImageFolder}";
   fi
   #
   if [ -d "${TheWindowsFolder}" ]; then
      sudo chmod a+x "${TheWindowsFolder}";
      sudo chmod -R a+x "${TheWindowsFolder}";
      sudo chown -R "${USER}":kvm "${TheWindowsFolder}";
   fi
```

```
    # spice-guest-tools-latest.exe
    if [ -d "$TheWindowsFolder" ]; then
        if [ "$ThisCowBackUp" -eq 1 ]; then
            mkdir -p "${TheCOW}";
            #
            print_caution "KVM Virtual Machine Manager Permission Fix and COW back up" "This
can take a while..." "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
            local thisFile;
            for thisFile in "${TheKvmImageFolder}"*; do
                if copy_files "${thisFile}/" " " "${TheCOW}" "@ $(basename
"${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() : ${LINENO[0]}"; then
                    echo "Backed up COW to ${TheCOW}";
                else
                    echo "Failed to back up COW to ${TheCOW}";
                fi
            done
        fi
    fi
    sudo libguestfs-test-tool;
    # This is to make a share between Linux and Windows
    #sudo chown -R "${USER}":kvm /run/media/"${USER}"
    #sudo chmod -R a+x /run/media/"${USER}"
}
#
################################################################################
doSuperMinPatch()
{
    print_caution "Do supermin Patch" "This will cause a pacman update for this same app" "@
${FUNCNAME[0]}() : ${LINENO[0]}";
    if ! sudo libguestfs-test-tool; then
        if cd "${TheWorkSpace}"; then
            git clone https://github.com/jeonsi/arch_supermin_patch.git;
            #
            if cd "${TheWorkSpace}arch_supermin_patch"; then
                sudo pacman --noconfirm -Syu patch pkg-config;
                makepkg -si;
            fi
            if ! sudo libguestfs-test-tool; then echo "Failed libguestfs-test-tool"; fi
        fi
    fi
}
# END doSuperMinPatch
#
################################################################################
modifyKVM()
{
```

```
    ########
    # https://github.com/vanities/GPU-Passthrough-Arch-Linux-to-Windows10
    # Windows 10 installation iso https://www.microsoft.com/en-us/software-
download/windows10ISO
    # Direct Download: here https://software-
download.microsoft.com/pr/Win10_1809Oct_English_x64.iso?t=673fe9a0-8692-49ba-b0e0-
e8ca7d314fdc&e=1544486586&h=9bb1b05b0fe6d83b41a5e8780a406244
    # virtio* drivers for windows10 https://fedorapeople.org/groups/virt/virtio-win/direct-
downloads/archive-virtio/virtio-win-0.1.160-1/
    # Direct Download: here        https://fedorapeople.org/groups/virt/virtio-win/direct-
downloads/archive-virtio/virtio-win-0.1.160-1/virtio-win-0.1.160.iso
    #
    kvmTest;
    #GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"
    echo "";
    echo "Edit line GRUB_CMDLINE_LINUX_DEFAULT=quiet intel_iommu=on";
    pushClipboard "intel_iommu=on" 0;
    sudo gedit /etc/default/grub;
    sudo grub-mkconfig -o /boot/grub/grub.cfg;
    echo "Reboot to take effect";
    #
    ###
    # https://cockpit-project.org/running#archlinux
    # https://dausruddin.com/how-to-enable-clipboard-and-folder-sharing-in-qemu-kvm-on-
windows-guest/
    # https://www.spice-space.org/download/windows/spice-webdavd/
    # https://www.spice-space.org/download/windows/spice-webdavd/spice-webdavd-x64-
latest.msi
    sudo systemctl enable --now cockpit.socket;
    #
    # Add Hardware
    # org.spice-space.webdav.0
    #echo "virt-manager Paste this into Add Hardware: org.spice-space.webdav.0";
    #virt-manager;
    #pushClipboard "org.spice-space.webdav.0" 0;
    launchBrowser "http://localhost:9090";
    ###
    #
    #if ! sudo libguestfs-test-tool; then thisPatch=1; fi
    #if [ "$thisPatch" -eq 1 ]; then echo "supermin Patch required"; fi
    #if [ "$thisPatch" -eq 1 ]; then doSuperMinPatch; fi
    # fails virshpatcher
    mkdir -p ~/.config/libvirt;
    if [ -f /etc/libvirt/libvirt.conf ]; then
        if [ ! -f ~/.config/libvirt/libvirt.conf ]; then
```

```
        sudo cp -rv /etc/libvirt/libvirt.conf ~/.config/libvirt/ && sudo chown "${USER}":"${USER}" ~/.config/libvirt/libvirt.conf;
        #gedit ~/.config/libvirt/libvirt.conf;
      fi
   else
      echo "Failed /etc/libvirt/libvirt.conf";
   fi
   #
   if [ ! -f /etc/polkit-1/rules.d/50-libvirt.rules ]; then
      sudo touch /etc/polkit-1/rules.d/50-libvirt.rules;
      echo "";
      echo "Control-V to paste in the settings";
      pushClipboard "/* Allow users in wheel group to manage the libvirt daemon without authentication */\npolkit.addRule(function(action, subject) {\n    if (action.id == 'org.libvirt.unix.manage' && subject.isInGroup('wheel')) {\n          return polkit.Result.YES;\n }\n#});" 1;
      #/* Allow users in wheel group to manage the libvirt daemon without authentication */
      #polkit.addRule(function(action, subject) {
      #    if (action.id == 'org.libvirt.unix.manage' && subject.isInGroup('wheel')) {
      #          return polkit.Result.YES;
      #    }
      #});
      sudo gedit /etc/polkit-1/rules.d/50-libvirt.rules;
   fi
   #
   # Change #unix_sock_group = "libvirt" to unix_sock_group = "libvirt" by removing the # comment on around line 81
   # unix_sock_rw_perms = "0770" to unix_sock_rw_perms = "0770" around line 104
   # un_comment_file 'unix_sock_group = "libvirt"' /etc/libvirt/libvirtd.conf;
   # un_comment_file 'unix_sock_rw_perms = "0770"' /etc/libvirt/libvirtd.conf;
   # un_comment_file 'unix_sock_ro_perms = "0770"' /etc/libvirt/libvirtd.conf;
   # un_comment_file 'auth_unix_ro = "none"' /etc/libvirt/libvirtd.conf;
   # un_comment_file 'auth_unix_rw = "none"' /etc/libvirt/libvirtd.conf;

   # the above failed
   #sudo gedit /etc/libvirt/qemu.conf;

   #sudo gedit /etc/libvirt/libvirtd.conf;

   # Add libvirt to your group
   sudo usermod -a -G libvirt "$(whoami)";

   add_user_2_group "libvirt";
   #newgrp libvirt
```

```
   #add_option "/etc/libvirt/qemu.conf" "security_driver=" "none" "${FUNCNAME[0]} @
$(basename "${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() : ${LINENO[0]}";
   #security_driver = "none"
   #sudo gedit /etc/libvirt/qemu.conf;

   # Close KVM then run
   sudo modprobe -r kvm_intel;
   sudo modprobe kvm_intel nested=1;
   #
   if [ -f "/etc/modprobe.d/kvm-intel.conf" ]; then
      if ! is_string_in_file "/etc/modprobe.d/kvm-intel.conf" "options kvm-intel nested=1"; then
         sudo touch /etc/modprobe.d/kvm-intel.conf; echo "options kvm-intel nested=1" | sudo tee
/etc/modprobe.d/kvm-intel.conf
      fi
   else
     sudo touch /etc/modprobe.d/kvm-intel.conf; echo "options kvm-intel nested=1" | sudo tee
/etc/modprobe.d/kvm-intel.conf
   fi
   echo "nested = Y and nested_early_check = N";
   systool -m kvm_intel -v | grep nested
   # $ systool -m kvm_intel -v | grep nested
   #     nested              = "Y"
   #     nested_early_check  = "N"
   # $ cat /sys/module/kvm_intel/parameters/nested
   # Y
   if ! is_string_in_file /sys/module/kvm_intel/parameters/nested "Y"; then
      echo "Failed /sys/module/kvm_intel/parameters/nested";
   fi

   #
   #
   # https://computingforgeeks.com/using-vagrant-with-libvirt-on-linux/
   #
   #vagrant plugin install vagrant-libvirt
   if ! vagrant plugin install vagrant-libvirt vagrant-share; then
      # if fails:
      echo "vagrant plugin failed";
#       yay -Rs vagrant;
#       rm -rf ~/vagrant.d
#       rm -f /usr/local/bin/vagrant
#       rm -rf /opt/vagrant
#       yay -S vagrant;
#       net-ssh;
#       net-scp;
   fi
   pip3 install virt-backup
```

```
    # qt-virt-manager
    #
    kvmFixBackup;
    #sudo gedit /etc/libvirt/qemu.conf;
    #user = "root" to user = "root" round line 519
    #group = "root" to group = "root"
    #un_comment_file 'user = "root"' /etc/libvirt/libvirtd.conf;
    #un_comment_file 'group = "root"' /etc/libvirt/libvirtd.conf;
    #sudo gedit /etc/libvirt/libvirtd.conf;
    # libvirtError: internal error: process exited while connecting to monitor: Could not access KVM kernel module: Permission denied failed to initialize KVM: Permission denied
    # Systemd 234 assigns a dynamic ID for the kvm group (see FS#54943).
    # To avoid this error, you need edit the file /etc/libvirt/qemu.conf and change the line with group = "78" to group = "kvm"
    #replace_option "/etc/libvirt/qemu.conf" "group =" "kvm" "${FUNCNAME[0]} @ $(basename "${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() : ${LINENO[0]}";
    #sudo gedit /etc/libvirt/qemu.conf;
    #
    restart_libvirt; # sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service; } && { sudo systemctl status libvirtd.service; }
    sudo systemctl enable libvirtd.service;
    sudo systemctl status libvirtd.service;
    sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service; } && { sudo systemctl status libvirtd.service; }

    sudo systemctl start virtlogd.socket || { sudo systemctl restart virtlogd.socket; } && { sudo systemctl status virtlogd.socket; }
    sudo systemctl enable virtlogd.socket;
    sudo systemctl status virtlogd.socket;
    #
    restart_libvirt;
    #
    #sudo gedit /etc/libvirt/qemu.conf;

    sudo virsh net-start default;
    isDMAR;
    # virtualbox vde2 virtualbox-guest-utils virtualbox-guest-dkms virtualbox-ext-vnc virtualbox-host-dkms virtualbox-guest-iso linux318-virtualbox-host-modules linux318-virtualbox-guest-modules
    sudo cp -rv /etc/libvirt/libvirt.conf ~/.config/libvirt/ && sudo chown "${USER}":kvm ~/.config/libvirt/libvirt.conf
}
#
################################################################################
declare MY_KVM_NAME; MY_KVM_NAME="Ubuntu-Mate-Cinnamon-Hirsute-21_04";
declare MY_KVM_RAM; MY_KVM_RAM="12048";
```

```
declare MY_KVM_COW_PATH;
MY_KVM_COW_PATH="/media/KvmStorage/Cows/images/Ubuntu-Mate-Cinnamon-Hirsute-21_04.qcow2";
declare MY_KVM_SIZE_GB; MY_KVM_SIZE_GB="333";
declare MY_KVM_VCPU; MY_KVM_VCPU="9";
declare MY_KVM_OS_TYPE; MY_KVM_OS_TYPE="linux";
declare MY_KVM_OS_VARIANT; MY_KVM_OS_VARIANT="generic";
declare MY_KVM_ISO_IMAGE;
MY_KVM_ISO_IMAGE="/media/KvmStorage/Linux/ubuntu-mate-21.04-Hirsute-desktop-amd64.iso";
createKVM()
{
   virt-install --name "$MY_KVM_NAME"  --ram "$MY_KVM_RAM" --disk path="$MY_KVM_COW_PATH",size="$MY_KVM_SIZE_GB" --vcpus "$MY_KVM_VCPU" --os-type "$MY_KVM_OS_TYPE" --os-variant "$MY_KVM_OS_VARIANT" --console pty,target_type=serial --cdrom "$MY_KVM_ISO_IMAGE";
}
#
################################################################################
# Ubuntu Client
# sudo apt install -y qemu-guest-agent
# sudo apt update && sudo apt -y upgrade && sudo apt -y  install cinnamon-desktop-environment lightdm qemu-guest-agent
installKVM()
{
   print_caution "Install KVM" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   kvmTest;
   local -i thisPatch; thisPatch=0;
   # KVM
   # https://wiki.archlinux.org/index.php/libvirt
   # https://octetz.com/docs/2020/2020-05-06-linux-hypervisor-setup/
   # virtualgl
   # openbsd-netcat and gnu-netcat are in conflict. Remove gnu-netcat
   # spice-guest-tools
   #
   sudo pacman --noconfirm -R playonlinux;
   sudo pacman --noconfirm -R gnu-netcat;
   sudo pacman --noconfirm --needed -S openbsd-netcat;
   sudo pacman --noconfirm --needed -S playonlinux;
   sudo pacman --noconfirm --needed -S ebtables;
   # fail2ban iproute2 systemd ufw
   sudo pacman --noconfirm --needed -S dhclient dnsmasq dmidecode bridge-utils vde2 libguestfs spice-protocol packagekit kexec-tools fmt;
   sudo pacman --noconfirm --needed -S qemu qemu-arch-extra qemu-block-gluster qemu-block-iscsi qemu-block-rbd qemu-guest-agent libvirt virt-viewer virt-manager virt-install;
```

```
   sudo pacman --noconfirm --needed -S vagrant spice-vdagent xf86-video-qxl edk2-armvirt
libvirt-python libvirt-storage-rbd libvirt-dbus libvirt-storage-gluster;
   sudo pacman --noconfirm --needed -S cockpit cockpit-machines;
   # edk2-ovmf
   yay --noconfirm --needed -S vagrant-libvirt virt-backup virtio-win qt-virt-manager;
   yay --noconfirm --needed -S spice-guest-tools-windows;
   #if ! yay --noconfirm --needed -S ovmf-git; then sudo pacman --noconfirm --needed -S
extra/edk2-ovmf; fi
   sudo pacman --noconfirm --needed -S extra/edk2-ovmf;
   #
   # https://wiki.archlinux.org/title/Trusted_Platform_Module
   # 1.2  yay --noconfirm --needed -S trousers tpm-tools
   #sudo systemctl start tcsd.service || sudo systemctl restart tcsd.service && sudo systemctl
enable tcsd.service
   #sudo modprobe tpm;   sudo modprobe -a tpm_{infineon,tis,crb}
   sudo pacman --noconfirm --needed -S tpm2-abrmd tpm2-tools swtpm;
   yay --noconfirm --needed -S ibm-tss
   #
   modifyKVM;
   fix_kvm;
}
# END installKVM
#
################################################################################
restart_libvirt()
{
   print_caution "Start or restart libvirt" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo systemctl start libvirtd.service || { sudo systemctl restart libvirtd.service; } && { sudo
systemctl status libvirtd.service; }
}
# END restart_libvirt
#
################################################################################
fix_kvm()
{
   sudo virsh net-autostart default;
   sudo virsh net-start default;
   #sudo semanage fcontext -a -t svirt_image_t "${HOME}/KvmShare(/.*)?";
}
#
################################################################################
getDMAR()
{
   sudo dmesg | grep DMAR | grep "DMAR: IOMMU enabled";
}
# END getDMAR
```

```
#
################################################################################
isDMAR()
{
   if [ "$(getDMAR)" != "" ]; then
      echo "DMAR: IOMMU is enabled";
      return 0;
   else
      echo "DMAR: IOMMU is Not enabled";
      return 1;
   fi
}
# END isDMAR
#
################################################################################
#
installBluetooth()
{
   print_caution "Install Bluetooth" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo systemctl start bluetooth.service;
   sudo systemctl enable bluetooth.service;

   sudo systemctl start bluetooth;

   sudo pacman --noconfirm --needed -S bluez bluez-utils pulseaudio-bluetooth bluez-plugins
blueberry;
   sudo modprobe btusb;
   yay --noconfirm --needed -S xf86-input-joystick bluez-rfcomm bluez-hcitool;


   sudo dmesg | grep Bluetooth;
   sudo journalctl | grep Bluetooth;
}
# END installBluetooth
#
################################################################################
installEquipment()
{
   print_caution "Install Equipment" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   # Microscope
   sudo pacman --noconfirm --needed -S guvcview-qt;
   # Epson XP-4100 pick top local not IP
   yay --noconfirm --needed  -S epson-inkjet-printer-201301w;
}
# END installEquipment
```

```
#
################################################################################
installNpm()
{
   print_caution "Install Npm" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo npm update;
   sudo npm install -g npm;
   sudo npm install -g html-minifier;
   sudo npm install -g jshint;
}
# END installNpm
#
################################################################################
fixNpm()
{
   print_caution "fix Npm" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo npm update; sudo npm install -g npm;
   sudo pacman --noconfirm -S npm --overwrite='*';
}
# END fixNpm
#
################################################################################
#
installDev()
{
   print_caution "Install Dev" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
*******************************************************************************
   sudo pacman --noconfirm --needed -S binutils make gcc pkg-config fakeroot gnupg;
   sudo pacman --noconfirm --needed -S base base-devel multilib-devel cmake make pkg-config
wget gperf icu harfbuzz harfbuzz-icu mesa lib32-mesa;
   sudo pacman --noconfirm --needed -S gcc autoconf automake make cmake ruby perl glibc
lib32-glibc libc++ autoconf bison flex autogen git texinfo bash zlib;
   sudo pacman --noconfirm --needed -S net-tools inetutils git lib32-openssl openssl clang lib32-
clang llvm ninja python patch scons sed shellcheck;
   sudo pacman --noconfirm --needed -S gcc autoconf automake net-tools inetutils gettext libtool
libtiff zlib gd graphicsmagick;
   sudo pacman --noconfirm --needed -S doxygen ghostscript tcl astyle uncrustify cvs bleachbit;
   sudo pacman --noconfirm --needed -S fontconfig freetype2 lib32-fontconfig lib32-freetype2
xorg-fonts-type1 p7zip unzip zip lzip bzip2;
   sudo pacman --noconfirm --needed -S bash bash-bats bash-bats-assert bash-bats-support bash-
completion;
   sudo pacman --noconfirm --needed -S expat gtk2 libcanberra libpng12 libice libsm util-linux
ncurses libxau libxdmcp libxext libxft libxrender libxt libxtst ld-lsb;
```

```
    sudo pacman --noconfirm --needed -S lib32-expat lib32-gtk2 lib32-libcanberra lib32-libpng
lib32-libpng12 lib32-libice lib32-libsm lib32-util-linux lib32-ncurses lib32-zlib lib32-libx11
lib32-libxau lib32-libxdmcp lib32-libxext lib32-libxft lib32-libxrender lib32-libxt lib32-libxtst;
    # install manually
    #sudo pacman --noconfirm --needed -S emscripten binaryen;
    yay --noconfirm --needed -S miktex qtcreator-doxygen;
    # libpng2?
    sudo pacman --noconfirm --needed -S libpng;
    sudo pacman --noconfirm --needed -S nodejs nasm qemu fuse perl-html-parser;
    sudo pacman --noconfirm --needed -S pkgfile parallel;
    sudo pacman --noconfirm --needed -S maven at-spi2-atk double-conversion zstd dbus-x11;
    sudo pacman --noconfirm --needed -S libxslt libxcursor libxcomposite libxdamage libxrandr
libcap libxtst libpulse libpciaccess libxss libgcrypt libdrm;
    sudo pacman --noconfirm --needed -S libcups pciutils nss libxtst pulseaudio libgudev alsa-lib
gstreamer atk wayland;
    sudo pacman --noconfirm --needed -S libxcb xcb-proto xcb-util xcb-util-image xcb-util-wm
libxi xcb-util-keysyms;
    sudo pacman --noconfirm --needed -S psensor detox libltdl libxml2;
    sudo pacman --noconfirm --needed -S discover packagekit-qt5 flatpak fwupd mono mono-
tools mono-msbuild mono-addins dotnet-host;
    sudo pacman --noconfirm --needed -S unoconv pngquant;
    sudo pacman --noconfirm --needed -S tidy perl-tidy xmlstarlet vagrant;
    sudo pacman --noconfirm --needed -S rsync nano;
    sudo pacman --noconfirm --needed -S cvsps mercurial bzr valgrind python2-paramiko
openmpi guake;
    sudo pacman --noconfirm --needed -S pluma kate gcolor2 pango;
    sudo pacman --noconfirm --needed -S boost boost-libs;
    sudo pacman --noconfirm --needed -S perl-libwww perl-term-readkey perl-mime-tools perl-
net-smtp-ssl perl-authen-sasl;
    sudo pacman --noconfirm --needed -S lshw whois npm atom electron lib32-gtk2 lib32-libxss
lib32-nss xterm gdb tk;
    sudo pacman --noconfirm --needed -S gnome-disk-utility htop sshpass rsync openssh psutils;
    sudo pacman --noconfirm --needed -S arm-none-eabi-gcc arm-none-eabi-newlib;
    #
    sudo pacman --noconfirm --needed -S yasm mingw-w64-binutils mingw-w64-crt mingw-w64-
gcc mingw-w64-headers mingw-w64-winpthreads autoconf-archive libdatrie;
    yay --noconfirm --needed -S mingw-w64-libdatrie mingw-w64-configure mingw-w64-bzip2
mingw-w64-zlib;

    yay --noconfirm --needed -S mingw-w64-libthai;
    yay --noconfirm --needed -S mingw-w64-fontconfig;
    yay --noconfirm --needed -S mingw-w64-harfbuzz;
    yay --noconfirm --needed -S mingw-w64-freetype2;
    yay --noconfirm --needed -S mingw-w64-qt5-base
    yay --noconfirm --needed -S mingw-w64-boost;
```

```
    yay --noconfirm --needed -S git-git;
    yay --noconfirm --needed -S automake autoconf intltool libffi glib2 pcre ncurses dejagnu tk
perl-tk php bison flex lua emsdk;
    #yay --noconfirm --needed -S monodevelop-stable;
    #yay --noconfirm --needed -S monodevelop-git;
    yay --noconfirm --needed -S mono-basic;
    #yay --noconfirm --needed -S msbuild-stable monogame-git
    yay --noconfirm --needed -S gnome-sharp xsp;

    sudo pacman --noconfirm --needed -S kross
    # FIXME
    # sudo dmesg | grep -i acpi
    # acpi -i
    # sensors-detect
}
# END installDev
#
################################################################################
installMxe()
{
    # Get to the directory of your app, and run the Qt Makefile generator tool:
    # <mxe root>/usr/bin/i686-w64-mingw32.static-qmake-qt5
    # Build your project:
    # make
    # You should find the binary in the ./release directory:
    # wine release/foo.exe
    # If you want a 64-bit executable, build Qt with:
    # make MXE_TARGETS=x86_64-w64-mingw32.static qtbase
    # The default MXE_TARGETS value is i686-w64-mingw32.static.
    #
    if ! cd "$TheWorkSpace"; then
        mkdir -p "$TheWorkSpace";
        if ! cd "$TheWorkSpace"; then
            echo "Error could not find WorkSpace $TheWorkSpace";
            return;
        fi
    fi
    if [ -d "${TheWorkSpace}/mxe" ]; then return; fi
    git clone https://github.com/mxe/mxe.git;
    if ! cd mxe; then
        make check-requirements;
        make download;
        make update;
        make qt5 MXE_TARGETS='i686-w64-mingw32.static x86_64-w64-mingw32.static';

    fi
```

```
   # shellcheck disable=SC2016
   echo 'export PATH="$HOME/workspace/mxe:$PATH"' >> ~/.bash_profile;
}
# END installMxe
#
################################################################################
installQt()
{
   #
   # Qt 6 https://wiki.archlinux.org/index.php/qt
   # Online installer https://www.qt.io/download-thank-you?hsLang=en
   # xscrnsaver dbus-1
   sudo pacman --noconfirm --needed -S clang lib32-clang llvm ninja cmake python compiler-rt
kcachegrind libpng libjpeg-turbo;
   sudo pacman --noconfirm --needed -S python-dulwich python-fastimport python-gpgme
python-paramiko;
   sudo pacman --noconfirm --needed -S perf valgrind bzr mercurial git x11-ssh-askpass qbs
cmake gdb assimp lib32-glibc tk;
   sudo pacman --noconfirm --needed -S bison flex gperf python nodejs fontconfig libdrm
libxcomposite libxcursor libxi libxrandr libxtst;
   #
   if [ "$QT_ONLINE" -eq 1 ]; then
      sudo pacman --noconfirm --needed -S qtcreator qt5-translations qt5-examples qt5-doc qt5-
graphicaleffects;
      sudo pacman --noconfirm --needed -S qt5-webengine python-pyqt5-webengine;
      sudo pacman --noconfirm --needed -S qt6-base qt6-3d qt6-declarative qt6-doc qt6-examples
qt6-imageformats qt6-networkauth qt6-quick3d qt6-quickcontrols2 qt6-quicktimeline
      sudo pacman --noconfirm --needed -S qt6-shadertools qt6-svg qt6-tools qt6-translations
qt6-wayland;
      sudo pacman --noconfirm --needed -S qt6-charts qt6-datavis3d qt6-lottie qt6-
virtualkeyboard qt6ct qt6-5compat qt6-scxml;
   else
      echo "Install Qt using Online Installer";
      theQt="$(ls qt-unified-linux-x64-*-online.run)";
      if [ -f "$theQt" ]; then
         echo "Install Qt using Online Installer";
         "./$theQt" &
      fi
   fi
   #sudo pacman --noconfirm -R qtcreator; sudo pacman --noconfirm -S qtcreator;
   #
   # not support conflict yay --noconfirm --needed -S qtchooser;
}
#
################################################################################
installAndroid()
```

```
{
   # To enable multilib repository, uncomment the [multilib] section in /etc/pacman.conf
   # https://wiki.archlinux.org/title/Android
   # https://wiki.archlinux.org/title/qt
   # https://wiki.archlinux.org/title/Java#OpenJDK
   sudo pacman --noconfirm --needed -S "$(comm -12 <(pacman -Qq | sort) <(pacman -Slq
multilib | sort))";
   sudo pacman --noconfirm --needed -S base-devel multilib-devel gcc repo git gnupg gperf sdl
wxgtk2 squashfs-tools curl ncurses zlib schedtool perl-switch libxslt bc rsync ccache lib32-zlib
lib32-ncurses lib32-readline ttf-dejavu;
   # fails Failure while branching lp:cpu-checker  Cannot find the fakeroot binary. error making:
android-armv7a-eabi-openssl
   yay --noconfirm --needed -S android-armv7a-eabi-qt5;
   yay --noconfirm --needed -S android-x86-qt5;
   yay --noconfirm --needed -S android-x86-64-qt5;
   yay --noconfirm --needed -S android-apktool android-host-tools android-sdk android-sdk-
build-tools android-sdk-platform-tools;
   yay --noconfirm --needed -S android-sdk-build-tools-dummy android-sdk-dummy android-
sdk-platform-tools-dummy android-sdk-cmdline-tools-latest-dummy;
   yay --noconfirm --needed -S android-studio android-support heimdall simg-tools;
   yay --noconfirm --needed -S android-host-tools android-udev-rules;
   yay --noconfirm --needed -S android-studio;
   yay --noconfirm --needed -S ncurses5-compat-libs lib32-ncurses5-compat-libs aosp-devel;
   yay --noconfirm --needed -S android-aarch64-qt5;
   sudo pacman --noconfirm --needed -S jre8-openjdk java8-openjfx;
}
#
################################################################################
installMisc()
{
   print_caution "Install Misc" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   #
   if [ "$CINNAMON_INSTALL" -eq 1 ]; then
      yay --noconfirm --needed -S cinnamon-sound-effects;
   else
      sudo pacman --noconfirm --needed -S sddm-breath-theme plasma-meta kde-applications-
meta plasma-wayland-session;
   fi
   #
   sudo pacman --noconfirm --needed -S lvm2;
   sudo pacman --noconfirm --needed -S ufw ksshaskpass offlineimap gnome-keyring
kdeplasma-addons;
   sudo pacman --noconfirm --needed -S ufw ufw-extras gufw;
   sudo systemctl start ufw.service || sudo systemctl restart ufw.service && sudo systemctl status
ufw.service;
   sudo systemctl enable ufw.service;
```

```
    sudo ufw enable;
    yay --noconfirm --needed -S password-gorilla;
    yay --noconfirm --needed -S ananicy-git;
    yay --noconfirm --needed -S google-earth;
    yay --noconfirm --needed -S cryptkeeper revelation;
    yay --noconfirm --needed -S nvm;

    ${HOME}/.nvm/install.sh;
    chmod 755 ${HOME}/.nvm/nvm.sh; chmod 755 ${HOME}/.nvm/bash_completion;


    yay --noconfirm --needed -S maldet;
    sudo pacman --noconfirm --needed -S aspell-en hunspell-en_US bluefish bluegriffon libva-
utils;
    sudo pacman --noconfirm --needed -S libmythes mythes-en languagetool;
    sudo pacman --noconfirm --needed -S ffmpeg rtmpdump atomicparsley python-pycryptodome
youtube-dl xbindkeys xorg-xev telepathy-haze;

    sudo pacman --noconfirm --needed -S calibre-common sigil bookworm foliate gnome-books;
    #
    sudo pacman --noconfirm --needed -S translate-shell nfs-utils;
    sudo pacman --noconfirm --needed -S audacity ardour brasero cheese dvd+rw-tools
dvdauthor;
    sudo pacman --noconfirm --needed -S ffmpeg gaupol kdenlive mjpegtools mpgtx mencoder
openshot devede gpicview pitivi;
    sudo pacman --noconfirm --needed -S gst-libav gst-plugins-ugly libburn libisofs python-
chardet;
    sudo pacman --noconfirm --needed -S recordmydesktop xine-ui kimageformats catdoc xjadeo
harvid frei0r-plugins;
    sudo pacman --noconfirm --needed -S python-pyenchant opusfile libquicktime kipi-plugins
python-nose;
    sudo pacman --noconfirm --needed -S faac gpac espeak faac antiword unrtf odt2txt txt2tags
nrg2iso bchunk;
    sudo pacman --noconfirm --needed -S mpv mplayer;

    sudo pacman --noconfirm --needed -S mkvtoolnix-cli;
    yay --noconfirm --needed -S mkvalidator;
    yay --noconfirm --needed -S cpu-checker-bzr;
    #yay --noconfirm --needed -S lib32-ffmpeg lib32-libffmpeg lib32-chromaprint lib32-gst-
plugins-bad;
}
# END installMisc
#
################################################################################
installCadGraphics()
{
```

```
   sudo pacman --noconfirm --needed -S freecad kicad kicad-library kicad-library-3d inkscape;
   yay --noconfirm --needed -S kicad-library-digikey-git kicad-library-sparkfun-git kicad-
templates;
   #  kicad-symbols remove: kicad-library-3d and kicad-packages3d will remove: kicad-library
   # gimp-plug
   sudo pacman --noconfirm --needed -S gimp;
   yay --noconfirm --needed -S gimp-plugin-normalmap;
   # requires password yay --noconfirm --needed -S gimp-plugin-reflection;
   yay --noconfirm --needed -S gimp-plugin-insanebump;
   yay --noconfirm --needed -S gimp-plugin-make-anaglyph;
   yay --noconfirm --needed -S gimp-plugin-astronomy;
   sudo pacman --noconfirm --needed -S blender;
   # kdegraphics
   sudo pacman --noconfirm --needed -S handbrake handbrake-cli intel-media-sdk intel-media-
sdk;
}
#
################################################################################
installMakehuman()
{
   yay --noconfirm --needed -S makehuman;
}
#
################################################################################
installPostgreSql()
{
   #
   #
   # postgresql setup
   # https://wiki.archlinux.org/index.php/PostgreSQL
   # postgresql-libs: PostgreSQL driver
   # mariadb-libs: MariaDB driver
   # unixodbc: ODBC driver
   # libfbclient: Firebird/iBase driver
   # freetds: MS SQL driver

   #sudo pacman --noconfirm -R pgadmin4 python-psycopg2 postgresql postgresql-libs
   #sudo rm -r /var/lib/postgres/data
   sudo pacman --noconfirm --needed -S postgresql postgresql-libs pgadmin4 sqlite;

   sudo -iu postgres
   createuser --interactive --pwprompt;
   initdb --locale=en_US.UTF-8 -E UTF8 -D /var/lib/postgres/data;
   initdb --locale "$LANG" -E UTF8 -D /var/lib/postgres/data;

   sudo systemctl start postgresql.service;
```

```
   sudo systemctl status postgresql.service;
   sudo systemctl enable postgresql.service;

   sudo systemctl restart postgresql.service;

   # hostname
   #echo "$(hostname)";

   #
   sudo -iu postgres;
   #su - postgres

   #psql -f "${TheWorkSpace}wasmrust/axWeBook/realworld-rust-rocket-master/init.sql";


   #createuser realworld
   #createdb realworld

   #grant all privileges on database realworld to realworld;

   #createuser ${USER}
   #createdb ${USER}

   #createuser --interactive --pwprompt

   #
   sudo nano /var/lib/postgres/data/postgresql.conf;
   #listen_addresses = 'localhost'    # what IP address(es) to listen on;
   # max_connections = 121 or 124, test for higher
   #
   sudo nano /var/lib/postgres/data/pg_hba.conf;
   #local   all             ${USER}                                 trust
}
#
################################################################################
installHaru()
{
   echo " Haru Library for PDF Support. ";
   sudo pacman --noconfirm --needed -S libharu;
}
#
################################################################################
installHaproxy()
{
   sudo pacman --noconfirm --needed -S haproxy monit;
   #
```

```
    sudo systemctl start haproxy.service || { sudo systemctl restart haproxy.service; }
    sudo systemctl enable haproxy.service;
    sudo systemctl status haproxy.service;

    #sudo systemctl stop haproxy.service;
    #sudo systemctl disable haproxy.service;

    #
    sudo systemctl start monit.service || { sudo systemctl restart monit.service; }
    sudo systemctl enable monit.service;
    sudo systemctl status monit.service;

    #sudo systemctl stop monit.service;
    #sudo systemctl disable monit.service;
}
#
################################################################################
installJava()
{
    sudo pacman --noconfirm --needed -S java-runtime-common unoconv jdk8-openjdk jre8-
openjdk openjdk8-src openjdk8-doc;
    yay --noconfirm --needed -S mvnvm;
    yay --noconfirm --needed -S server-jre;

    yay --noconfirm --needed -S jdk-dcevm;

    yay --noconfirm --needed -S java-gcj-compat;
    yay --noconfirm --needed -S gcc6-gcj-compat;
    yay --noconfirm --needed -S gjdoc;
    #yay --noconfirm --needed -S jdk6 jdk5 zulu-jdk jdk9-openj9-bin jdk7 jre7 jre6 jdk7-j9-bin
jdk7r1-j9-bin jdk8-j9-bin tuxjdk jre8-openjdk-jetbrains
    #yay --noconfirm --needed -S jre-devel jdk8-openj9-bin jre7-openjdk-infinality intellij-jdk
jdk8 jre8 jdk jre jdk9 jre9 jdk-devel
    #yay --noconfirm --needed -S java-8-openjdk-shenandoah jre8-openjdk-infinality jre10-
openjdk jre7-openjdk jre8-openjdk jre9-openjdk
    #yay --noconfirm --needed -S jdk-arm

}
#
################################################################################
installArchive()
{
    sudo pacman --noconfirm --needed -S arj cabextract p7zip sharutils unace unrar unzip
uudeview zip lzip bzip2 minizip;
}
```

```
#
################################################################################
installUtil()
{
   #  qt5-base qt5-declarative qt5-doc qt5-webkit qtcreator qt5-imageformats
   #  openvpn easy-rsa networkmanager-openvpn pptpclient networkmanager-pptp
   # avidemux avidemux-qt torcs scorched3d banshee unetbootin

   #yay --noconfirm --needed -S epubcheck wkhtmltopdf-static linkchecker nodejs-jshint python-weasyprint stylelint kindlegen
   yay --noconfirm --needed -S epubcheck;
   yay --noconfirm --needed -S wkhtmltopdf-static;
   yay --noconfirm --needed -S linkchecker;
   yay --noconfirm --needed -S nodejs-jshint;
   yay --noconfirm --needed -S python-weasyprint;
   yay --noconfirm --needed -S stylelint;
   yay --noconfirm --needed -S kindlegen;


   # failed
   if ! yay --noconfirm --needed -S csslint; then
      # if fails
      sudo npm install -g csslint;
   fi
   sudo npm install -g uglify-js;
   sudo npm install -g kindlegen;
   sudo npm install -g stylelint;
   sudo npm install -g epubcheck;
   sudo npm install -g yuicompressor;
   sudo npm install -g minify;
   #  uglify-js
   yay --noconfirm --needed -S yuicompressor minify;

   yay --noconfirm --needed -S nodejs-clean-css-cli;

   sudo pacman --noconfirm --needed -S git make maven;

   yay --noconfirm --needed -S closure-compiler;

}
#
################################################################################
installInternet()
{
   sudo pacman --noconfirm --needed -S evolution evolution-bogofilter evolution-spamassassin aspell-en nmap arp-scan dnsutils traceroute procmail m4;
```

```
   sudo pacman --noconfirm --needed -S filezilla pidgin amule chromium opera transmission-qt
iperf3 ;
   yay --noconfirm --needed -S sendmail;

   sudo pacman --noconfirm --needed -S telegram-desktop
   yay --noconfirm --needed -S teamviewer
   sudo teamviewer --daemon enable
   sudo systemctl enable teamviewerd && sudo systemctl start teamviewerd;

   sudo pacman --noconfirm --needed -S kde-servicemenus-pkg-tools
   yay --noconfirm --needed -S kde-service-menu-reimage
   yay --noconfirm --needed -S kde-servicemenus-getmediainfo

   # https://www.privateinternetaccess.com/download/linux-vpn
   thePIA="$(ls pia-linux-*.run)";
   if [ -f "$thePIA" ]; then
      sh "$thePIA" &
   fi
}
#
################################################################################
installGames()
{
   #  doom3 quake3 quake4
   sudo pacman --noconfirm --needed -S frozen-bubble aisleriot pysolfc pysolfc-cardsets
libkdegames supertuxkart pingus;
   sudo pacman --noconfirm --needed -S bsd-games kajongg gnuchess knights atomix kde-
games-meta supertuxkart nInvaders glChess;
   yay --noconfirm --needed -S bs qcheckers chessx blockout2 eduke32 darkplaces ecwolf
flightgear flightgear-data fs2_open;
}
#
################################################################################
installEd()
{
   # qalculcate-gtk
   sudo pacman --noconfirm --needed -S stellarium celestia marble-qt kstars anki libwlocate
shapelib blinken cantor genius geogebra goldendict;
   sudo pacman --noconfirm --needed -S kgeography kig mathomatic qcad xplanet gnuplot
rlwrap kde-education-meta kalzium;
   #yay --noconfirm --needed -S bible-time gpsbable kalgabra
}
#
################################################################################
installClamAV()
{
```

```
    print_caution "Install ClamAV" "" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
    # ClamAV
    # https://wiki.archlinux.org/index.php/ClamAV
    #
    sudo pacman --noconfirm --needed -S clamav clamtk;
    sudo freshclam;

    #sudo systemctl start clamav-daemon.service;
    #sudo systemctl status clamav-daemon.service;
    #sudo systemctl enable clamav-daemon.service;

    #sudo systemctl start clamav-freshclam.service
    #sudo systemctl enable clamav-freshclam.service
    #sudo systemctl restart clamav-freshclam.service
    #sudo systemctl status clamav-freshclam.service

    #sudo systemctl start clamav-milter.service
    #sudo systemctl enable clamav-milter.service
    #sudo systemctl restart clamav-milter.service
    #sudo systemctl status clamav-milter.service



    #sudo systemctl enable clamd.service
    #sudo systemctl restart clamd.service
    #sudo systemctl status clamd.service

}
# END installClamAV
#
################################################################################
#
doScan()
{
    sudo maldet -u;
    sudo freshclam;
    if [ ${TheScanLevelAV} -eq 1 ]; then
        # clamscan --infected --recursive --remove /home;
        sudo maldet -a /home;
        sudo ionice -c3 nice -n 19 clamscan --remove=yes -r -i  /home | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
    elif [ ${TheScanLevelAV} -eq 2 ]; then
        sudo maldet -a /home;
        sudo ionice -c3 nice -n 19 clamscan --remove=yes --recursive=yes --infected --exclude-
dir='^/sys|^/proc|^/dev|^/lib|^/bin|^/sbin|^/mnt' / | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
```

```
   elif [ ${TheScanLevelAV} -eq 3 ]; then
      sudo maldet -a /home;
      sudo ionice -c3 nice -n 19 clamscan --remove=yes --recursive=yes --infected --exclude-
dir='^/sys|^/proc|^/dev|^/lib|^/bin|^/sbin' / | mail -s "ClamAV Report"
"${EMAIL_ADDRESS_NOTIFICATIONS}";
   fi
   # clamscan myfile
   # clamscan --recursive=yes --infected /home # or -r -i
   # clamscan --infected --recursive --remove $ThisRoofFolder/2.Websites
}
#
################################################################################
# installGedit
installGedit()
{
   print_caution "Install Gedit" "Plugins" "@ ${FUNCNAME[0]}() : ${LINENO[0]}";
   sudo pacman --noconfirm --needed -S gedit gedit-plugins;
   yay  --noconfirm --needed -S gedit-duplicate-line;
   # gedit plugin shellcheck
   # https://github.com/lwindolf/gedit-shellcheck
   if [ ! -d "${HOME}/.local/share/gedit/plugins/" ]; then
      if cd ~/; then
         mkdir -p "${HOME}/.local/share/gedit/plugins/tmp/";
         if cd "${HOME}/.local/share/gedit/plugins/tmp/"; then
            git clone https://github.com/lwindolf/gedit-shellcheck.git;
            /usr/bin/cp -rf gedit-shellcheck/shellcheck.plugin gedit-shellcheck/shellcheck/
"${HOME}/.local/share/gedit/plugins/";
         fi
         if cd ~/; then
            /usr/bin/rm -rf "${HOME}/.local/share/gedit/plugins/tmp/";
         fi
      fi
   fi
}
# END installGedit
#
################################################################################
isError()
{
   # Troubleshooting
   sudo systemctl --failed;
   return "$?";
}
# END isError
#
################################################################################
```

```
# if ! sudoAppend "String to Apppend" "full-file-path-name-extentsion"; then echo "Failed"; fi
sudoAppend()
{
   [[ $# -ne 2 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   if is_string_in_file "$2" "$1"; then return 0; fi
   echo "$1" | sudo tee -a "$2" > /dev/null;
   return "$?";
}
# END sudoAppend
#
################################################################################
fixSudoTimeOut()
{
   #sudo visudo;
   #export EDITOR=nano;
   #EDITOR=nano sudo visudo;
   local thisString; thisString="Defaults        env_reset,timestamp_timeout=666";
   sudo cp /etc/sudoers ~/Downloads;
   sudo chown "$USER":"$USER" ~/Downloads/sudoers;
   if is_string_in_file ~/Downloads/sudoers "$thisString"; then return 0; fi
   #
   if sudoAppend "$thisString" ~/Downloads/sudoers; then
      #
      if sudo visudo -c -f ~/Downloads/sudoers; then
         if ! sudoAppend "$thisString" /etc/sudoers; then
            ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
            sudo gedit /etc/sudoers;
         fi
         if ! sudo visudo -c -f /etc/sudoers; then
            ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
         fi
      fi
   else
      ifError "Failed ${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
      sudo gedit ~/Downloads/sudoers;
   fi
   #echo "paste to end of file: Defaults:USER timestamp_timeout=666;";
   #sudo visudo;
}
# END fixSudoTimeOut
```

```
#
################################################################################
ifError()
{
   [[ $# -ne 1 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   print_error "$1" "@ $(basename "${BASH_SOURCE[0]}") -> ${FUNCNAME[0]}() :
${LINENO[0]}";

   if [ "$IsBeeper" -eq 1 ]; then
      dobeep;
   fi
}
# END ifError
#
################################################################################
# installThis "list of things to install"
# install_package_with 1->(Package) 2->(Confirm [1=no-confirm]) 3->(Force [1=Force]) 4-
>(alternet install [0=normal,1=alternet])
installThis()
{
   [[ $# -ne 2 ]] && { print_error "LOCALIZE_WRONG_ARGS_PASSED_TO"
"${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> : ${LINENO[0]}"; exit 1;
}
   if [ "${SimulateThis}" -eq 0 ]; then
      echo "installThis";
      if ! install_package_with "$1" 1 0 "$2"; then
         ifError "Failed installThis";
      fi
   else
      echo "Simulate installThis";
   fi
}
# END installThis
#
################################################################################
# installGem bundler
#
installGem()
{
   local gemname; gemname="$1";
   sudo gem install "$gemname";
   sudo pacman --noconfirm --needed -S ruby-"$gemname";
}
# END installGem
```

```
#
################################################################################
# ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-linux]
rubyVersion()
{
   local thisVersion; thisVersion="$(ruby -v)";
   thisVersion="${thisVersion:5}";
   thisVersion="${thisVersion%%p*}";
   echo "$thisVersion";
}
# END rubyVersion
#
################################################################################
installUnrealEngine()
{
   #
   if [ ! -d "$ThisUnrealEngineGitFolder" ]; then
      if cd "${ThisRoofFolder}/workspace"; then
         git clone https://github.com/Light-Wizzard/UnrealEngine.git
         if ! cd UnrealEngine; then
            echo "Error Unreal Engine";
            exit 1;
         fi
      fi
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine/Engine"; then
      git pull;
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine"; then
      git pull;
      ./Setup.sh;
      ./GenerateProjectFiles.sh;
      make -j1;
   fi
   #
   if cd "${TheWorkSpace}UnrealEngine/Engine/Binaries/Linux"; then
      ./UE4Editor
      "${TheWorkSpace}UnrealEngine/Engine/Binaries/Linux/UnrealVersionSelector-Linux-
Shipping" -editor %f;
   fi
   #
   sudo chmod -R a+rwX /opt/unreal-engine/Engine;
   if cd /opt/unreal-engine/Engine; then
      git pull;
```

```
    fi
    #
    # https://github.com/nmrugg/UE4Launcher
    if [ ! -d "$ThisUnrealEngineLauncherGitFolder" ]; then
        if cd "${ThisRoofFolder}/workspace"; then
            git clone https://github.com/nmrugg/UE4Launcher.git;
        fi
    fi
    # Update
    if cd "${TheWorkSpace}UE4Launcher"; then
        git pull;
        npm i;
        npm audit fix;
        npm start;
    fi
}
#
################################################################################
# You have to download the software from ST
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-performance-and-debuggers/stm32cubemonitor.html#get-software
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stm32cubeprog.html
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html#get-software
# https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/st-mcu-finder-pc.html
installSTM32()
{
    yay --noconfirm --needed -S stm32cubemx;
    yay --noconfirm --needed -S stm32cubeide;
    yay --noconfirm --needed -S stm32flash;
    yay --noconfirm --needed -S stm32cubemonitor;
    yay --noconfirm --needed -S stm32cubeprog;
    yay --noconfirm --needed -S stmcufinder;
    #yay --noconfirm --needed -S sw4stm;
}
#
################################################################################
#installKDE()
#{
#
```

```
#}
#
################################################################################
installWine()
{
   # pywinery bottles
   sudo pacman --noconfirm -R jack;
   sudo pacman --noconfirm --needed -S jack2;
   sudo pacman --noconfirm --needed -S base-devel mono mono-tools xterm wine-staging
playonlinux winetricks wine-gecko wine-mono zenity unixodbc wine-nine;
   sudo pacman --noconfirm --needed -S wget xosd cups samba dosbox ncurses giflib gnutls
mpg123 openal v4l-utils opencl-icd-loader;
   sudo pacman --noconfirm --needed -S libldap libpulse alsa-plugins alsa-lib libjpeg-turbo
libxcomposite libxinerama libxslt libva gst-plugins-base-libs vulkan-icd-loader;
   sudo pacman --noconfirm --needed -S lib32-mpg123 lib32-giflib lib32-libldap lib32-v4l-utils
lib32-libjpeg-turbo lib32-libxcomposite lib32-gst-plugins-base lib32-gst-plugins-good;
   sudo pacman --noconfirm --needed -S lib32-gnutls lib32-libcups lib32-libxrandr lib32-
libxinerama lib32-alsa-lib lib32-alsa-plugins lib32-libpulse lib32-alsa-oss lib32-openal lib32-
ncurses;
   sudo pacman --noconfirm --needed -S lib32-opencl-icd-loader lib32-libxslt lib32-libva lib32-
gst-plugins-base-libs lib32-vulkan-icd-loader;
   sudo pacman --noconfirm --needed -S libpng-apng lib32-libpng;
   sudo pacman --noconfirm --needed -S a2jmidid libffado jack2-dbus realtime-privileges zita-
ajbridge lib32-giflib lib32-mpg123 lib32-v4l-utils opencl-icd-loader lib32-sdl opencl-driver;
   sudo pacman --noconfirm --needed -S lib32-libxslt lib32-libva lib32-gst-plugins-base-libs
lib32-vulkan-icd-loader vkd3d lib32-vkd3d dosbox kdialog lib32-opencl-driver opencl-headers;
   sudo pacman --noconfirm --needed -S lib32-libva-vdpau-driver lib32-libva-intel-driver
opencl-clhpp;
   sudo pacman --noconfirm --needed -S fuseiso;
   #sudo pacman --noconfirm --needed -S libpng lib32-libpng;
   #sudo winetricks --self-update;
   #  lib32-gst-plugins-ugly
   yay --noconfirm --needed -S q4wine dxvk-bin mono-basic wineasio wine-installer wine-
browser-installer wine-libusb-git wine-mono-gecko-version-fix;
   regsvr32 wineasio.dll;
   wine64 regsvr32 wineasio.dll;
   # shellcheck disable=SC2046
   # shellcheck disable=SC2062
   sudo pacman --noconfirm --needed -Ss $(pacman -Qq | grep lib32-*);
   #for P in "$(pacman -Qq | grep lib32-*)"; do if ! (pacman -Q | grep ${P} > /dev/null); then
sudo pacman --noconfirm --needed -S "${P}"; fi done
   # shellcheck disable=SC2046
   sudo pacman --noconfirm --needed -S $(pactree -l wine-staging);
   sudo systemctl restart systemd-binfmt;
   sudo winetricks --self-update;
}
```

```
#
################################################################################
installCodec()
{
   sudo pacman --noconfirm --needed -S celt lame a52dec libdca libmad libmpcdec opencore-
amr speex libvorbis faac faad2 libfdk-aac fdkaac jasper libwebp;
   sudo pacman --noconfirm --needed -S aom dav1d rav1e svt-av1 libde265 libdv libmpeg2
schroedinger libtheora libvpx x264 x265 xvidcore;
   sudo pacman --noconfirm --needed -S mkvtoolnix-cli mkvtoolnix-gui ogmtools xine-lib xine-
ui gst-libav;
   yay --noconfirm  --needed -S mp4joiner mp4tools;
   yay --noconfirm  --needed -S daala-git openjpeg;
   # yay --needed -S opus-git x264-git x265-hg libde265-git libvpx-git
   #
   #yay --needed -S neroaac-bin
}
#
################################################################################
# sudo rndc flushname fix-america-now.org
# 192.168.1.13
# 255.255.255.0
# 192.168.1.254
# nameserver 127.0.0.1
# nameserver 8.8.8.8
# nameserver 8.8.4.4
clearDNS()
{
   sudo systemctl restart nscd;
   sudo nscd -K;
   sudo systemctl restart dnsmasq;
   sudo systemctl restart named;
}
#
################################################################################
installSQLite()
{
   sudo pacman --noconfirm --needed -S sqlite sqlitebrowser;
}
#
################################################################################
preRun()
{
   if [ -f "custominstall.sh" ]; then
      source "custominstall.sh";
   fi
   runMe;
```

```
}
#
################################################################################
runThisStuff()
{
   echo "Running Stuff RUN_OPTIONS=${RUN_OPTIONS} and DoNotRun=$DoNotRun";
   if is_root_user; then echo "Do not run as root"; exit 1; fi
   # Begin Script
   echo "";
   if [ "$ThisClearOk" -eq 0 ]; then cls; fi # clear;
   echo "";
   sudo_Required "${FUNCNAME[0]}";
   #
   if [ "${RUN_OPTIONS}" -eq 1 ] && [ "$DoNotRun" -eq 0 ]; then
      #
      # Start App belows
      echo "Run Stuff";
      #
      if [ "${TheRunFirst}" -eq 1 ] && [ "$DoNotRun" -eq 0 ]; then
         echo "Run First-time";
         fixSudoTimeOut;
         pacmanFast;
         runFirst; # makeHome;
         #exit 1;
         installDev;
         installGraphicCard;
         installFonts;
         installCodec;
         installMisc;
         installNpm;
         installClamAV;
         installGedit;
         installArchive;
         installEquipment;
         installBluetooth;
         installJava;
         installUtil;
         installSamba;
         installSQLite;
         installWine;
         installCadGraphics;
         installGames;
         installEd;
         installKVM;
         installGem bundler;
         installGem net-ssh;
```

```
            installGem net-scp;
            #installMxe;
            installAndroid;
            installInternet;
            installQt;
            installRust;
            #installPostgreSql;
            #installUnrealEngine;
            #installSTM32;
            #isDMAR;
            #rubyVersion;
            #installQuartus;
            #installSagemath;
            #installMakehuman;
            #installHaru;
        fi
        #
        # End App before here
        #
    elif [ "$TheRunScan" -eq 1 ]; then
        echo "TheRunScan=1";
        doScan;
    elif [ "$TheKvmFixBackup" -eq 1 ]; then
        kvmFixBackup;
    fi
    #
    if [ "${SetDebug}" -eq 1 ]; then set +x; fi # turn OFF debug mode
    #
    print_this "WIZ_BYE" "${FUNCNAME[0]}() @ $(basename "${BASH_SOURCE[0]}") -> :
${LINENO[0]}";
    #
    tput sgr0; # Reset the Terminal on Exit
}
# END runThisStuff
#
################################################################################
if [ -f "wizard-common.sh" ]; then
    echo -e "\033[0;32mLoading\033[0m wizard-common.sh ";
    #. "wizard-common.sh";
    # shellcheck disable=SC1091
    # shellcheck disable=SC1090
    source "wizard-common.sh";
else
    echo "File Not Found: wizard-common.sh - $(basename "${BASH_SOURCE[0]}") :
${LINENO[0]}";
    #exit 1;
```

```
fi
#
###############################################################################
startTime;
checkArgs "$@";
if [ -f "wizard.sh" ]; then
   echo -e "\033[0;32mLoading\033[0m wizard.sh ";
   #. "wizard.sh";
   # shellcheck disable=SC1091
   # shellcheck disable=SC1090
   source "wizard.sh";
else
   echo "File Not Found: wizard.sh - $(basename "${BASH_SOURCE[0]}") : ${LINENO[0]}";
   #exit 1;
fi
#
runThisStuff;
#
endTime;
#
###############################################################################
# ********************************* End of Script ******************************
```