

# Affordable Gaming on Linux



Adam Schaible  
FOSSCON 2019 – Philadelphia, PA  
[sysadmin@schibes.com](mailto:sysadmin@schibes.com)

# Foreword

- I am an IT consultant and Red Hat Certified System Administrator from Chester County, PA who has been using Linux/Unix and building my own PCs since 1997. My favorite distros over the years have been Red Hat (pre-RHEL), Mandrake, Ubuntu, Mint, Solus, Fedora, and Manjaro. When not tweaking my large fleet of derelict computers to play obsolete games, like many others here at FOSSCON I am an avid amateur radio operator (callsign: KB3ZUV).
- The target audience for this slideshow consists of beginner to intermediate level Linux users who are considering building their own gaming PC for the first time, or migrating an existing gaming PC to Linux. If you are an advanced or “elite” Linux user, please be patient, we’ll get to the cool stuff eventually.

## Introduction I - why game on Linux at all?

- For many years, the conventional view of Linux (outside the FOSS community) has been that it is a best-of-breed server OS but mediocre on the desktop. For many long time Linux users like myself, gaming is frequently the only reason worth having any Windows PC at all (yes, I do still have one). So why do I stubbornly persist in continuing to try to game on Linux?
- **Upfront Cost.** A Windows 10 license for home use can be as much as \$100, the cost of a couple new AAA games. While that's not prohibitively expensive, it's still \$100 you could put into nicer hardware. The “free as in beer” argument, while fading somewhat inside our community, still drives a good deal of initial interest in Linux out in the general public.

## Introduction II - why game on Linux at all?

- ***Maintenance hassles.*** Hardware upgrades can break Windows activation. You can "get away with it" a few times for free, then you need to call Windows support to unlock Windows, and eventually you may even need to buy a new OS license. So \$100 is only \$100, until it's not.
- ***Windows Update.*** Besides my annoyance with Windows licensing, I have found Windows Update to be slow and bug-prone. It also frequently breaks support for legacy peripherals I own like ergonomic keyboards and drawing tablets that still work perfectly fine in Linux. While it can be turned off, this comes with significant security risks that are unacceptable to many users (myself included).

## Introduction III - why game on Linux at all?

- ***It's still a PC.*** I need to acknowledge the existence of consoles in this presentation, so it might as well be here. Even setting aside Windows activation and update annoyances, being able to upgrade hardware is one of the main advantages PC gaming in general has over console gaming, and this shows no signs of changing any time soon.
- ***It's... fun?*** Because gaming on Linux is slowly improving over time it can be a good way to have fun and promote the OS to newcomers.

## Linux games – general classification

- Free (as in speech)/GPL/FOSS titles – open source code, anyone can modify or port the code, games developed by the community, some truly great games like Xonotic and Battle for Wesnoth have emerged in this way
- Closed source games that have been ported to run natively on Linux – Neverwinter Nights, Half Life 2, Borderlands 2, Crusader Kings 2, Age Of Wonders 3, and endless indies including most of my preferred genre, roguelites aka “Metroidvania” (Dead Cells, Hollow Knight, Sundered)
- Closed source games that have NOT been ported to Linux but can still be made to run – from Blizzard, Bethesda, and many others

# A concise history of gaming on Linux – Part I

- As this slideshow is intended as a practical guide, the entire history of gaming on Linux is far too much to cover in this presentation, nevertheless it helps to have some historical context.
- While there has always been an active, thriving community of free and open source games available for the OS, the widespread availability of mainstream, closed source, mass market gaming on the GNU/Linux OS is a much more recent (2010s) phenomenon.

## A concise history of gaming on Linux – Part II

- Brief attempts at creating a mass-market Linux gaming community were made in the early 2000s by companies such as Loki. After these failed to gain popularity the community went through a "dark age" of sorts as the DirectX Windows API dominated its rival API, OpenGL to rule PC gaming for almost an entire decade. During this time, Wine was often the only workable way to play (even a few) new PC games on Linux.
- However, beginning around 10 years ago the rise of mobile devices like the iPhone and Android phones began to break the DirectX stranglehold on non-console gaming by giving rise to cross-platform game development tools like the Unity 3D engine and OpenGL's successor, Vulkan.



# Illustration 1: Linux game ports by Loki (~2002)



## A concise history of gaming on Linux – Part III

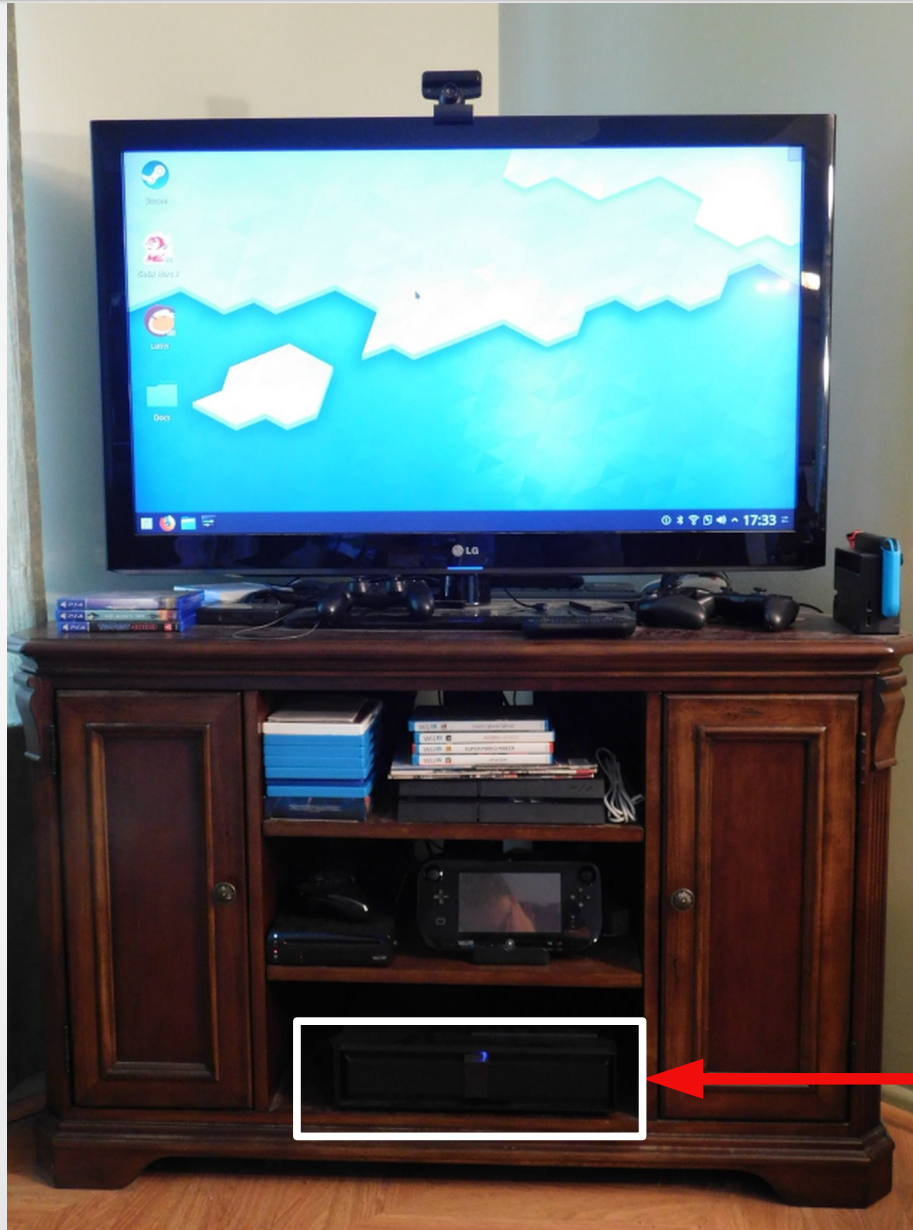
- In 2013, Valve Software released its Debian GNU/Linux based “SteamOS” distro and SteamBox hardware platform, which spawned a large proliferation of cutting edge games on Linux.
- While the commercial failure of the SteamBox platform has slowed the growth of gaming on Linux somewhat, things may never again be as bad as they were 15 years ago, not within the next decade at least, and certainly not within the 2-to-5 year lifecycle of any new gaming PC you are likely to build today. So, given the very low risks of platform adoption, let's get started.

## Building the box I - Assessing use case and requirements

- **Use case:** I already have a high end tower PC for intense, seated keyboard and mouse style FPS gaming. The PC in this “budget” build, given its lower hardware specs is intended more as a “console substitute” for casual gaming in the living room as well as light home theater use (browser based video such as Prime Video, Youtube, etc. and local streaming of my collection of DVD backup rips).
- **Requirements:** “Affordable” - \$500 max  
Small - must fit in living room entertainment center  
“Silent” - quieter than my PS4, anyway  
Semi-Portable - maybe I want to get back into LAN parties?  
Upgradable - make it easy to add CPU/GPU/RAM/storage later on to prolong system life



## Illustration 2: Living room gaming PC installation



## Building the box II – Bill of materials

- CPU/GPU: AMD Ryzen 3 2200G **\$80**
- Motherboard: AsRock B450 (ITX) **\$85**
- RAM: G.Skill 8GB DDR4 **\$85**
- Storage: Kingston 250GB NVME M.2 drive **\$50**
- Case: Silverstone Milo **\$90**
- Power Supply: Silverstone 300W SFF **\$60**
- Total: **\$450** (Prices September 2018. Since then, the DDR4 shortage/price fixing has ended and this build is down around \$400 today)

## Building the box III – Upgrade considerations

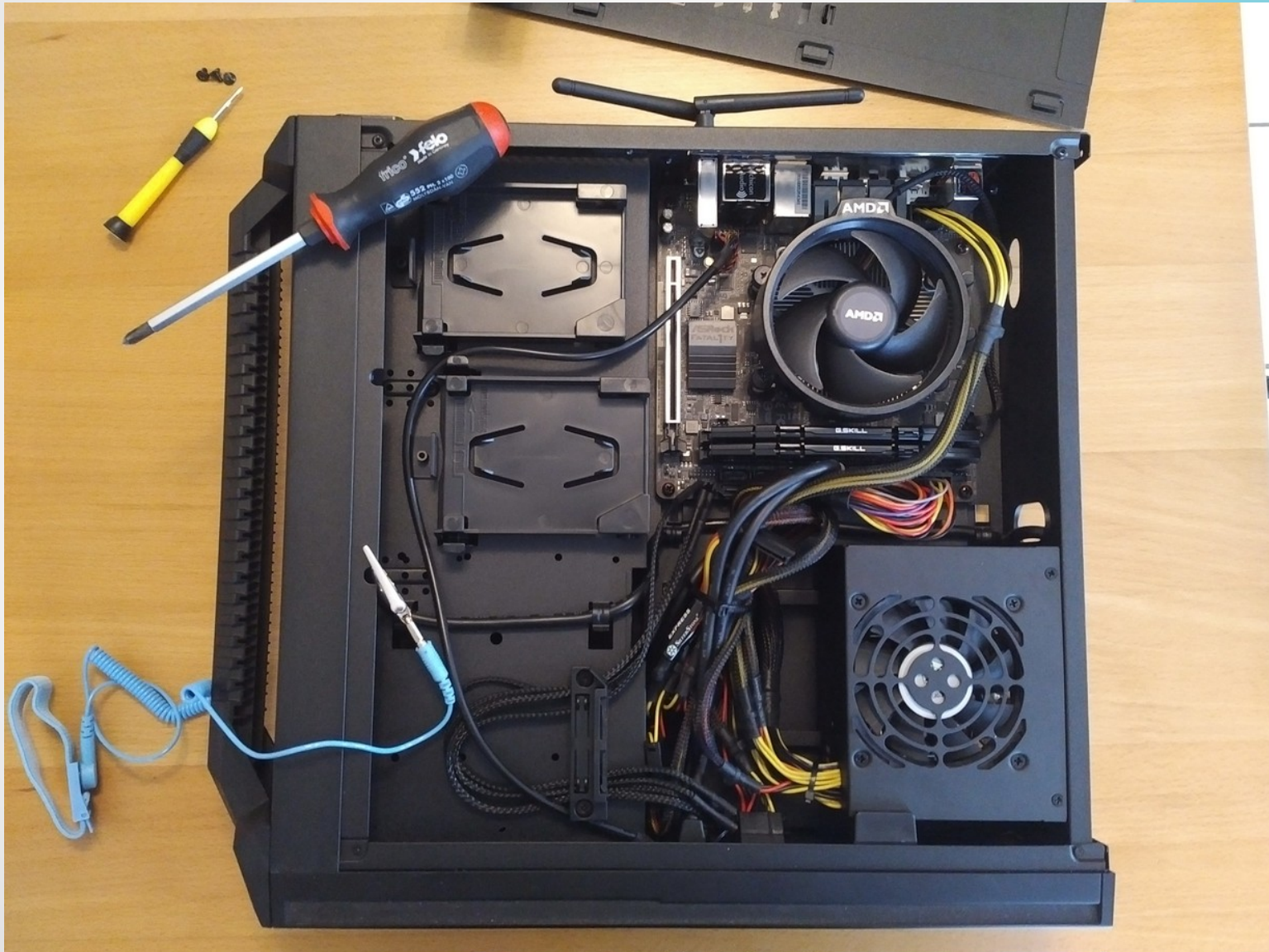
- Obvious upgrades: CPU/GPU, RAM, and M.2 drive are all bottom-of-the-barrel "minimum spec" for this platform with drop-in replacements readily and abundantly available for years to come, such as a CPU/GPU upgrade just last month (July '19)
- I purposely bought a "too large" case because it provides future upgrade options for discrete graphics and legacy SATA hard drive form factors.
- We do need to be mindful of power requirements for any future upgrade to discrete graphics, the 300W PSU may limit me to mid-range cards, however by the time I'm ready to add one (2021 maybe?) even mid-range cards are likely to be vastly more powerful than they are today.

## Building the box IV – Alternative considerations

- "Thrift Shop Option" - build this box for \$300 instead of \$450 by recycling a case and power supply from an obsolete PC.  
Caveat: check internal case cabling and ports, especially USB!
- "Barebones Option" - Build this box for \$375 by using miniature case/motherboard combo with "wall wart" laptop style AC adapter for \$150, but can't add discrete graphics later.
- My display and peripherals are all carried over from this system's predecessor: Handheld wireless USB keyboard/mouse, PS3 controllers, 42" flatscreen TV (1080p) with HDMI cable.
- Don't build an x86 PC at all and game on a Raspberry Pi :-)



## Illustration 3: PC construction





## Manjaro setup – Part I

- Why Manjaro? SteamOS is based on Debian, Valve had historically recommended Steam on Ubuntu, but support can be lacking in Ubuntu for the very newest (low end, but NEW) hardware. Ubuntu's developer, Canonical has also been arguing with Valve in public as of late (June '19) about 32-bit library support so it may soon be time to do some distro (s)hopping.
- For new hardware in a non-mission-critical environment (i.e. I am a casual gamer not an elite e-sports competitor) who wants the most broad based support for a mixture of new hardware and older legacy peripherals (controllers), Manjaro proved superior for this one specific use case.
- Besides Manjaro, Fedora and Solus are my other top choices for Linux desktop gaming distros at the moment.

## Manjaro setup II - pacman

- Manjaro, as a descendant of Arch Linux uses pacman as its primary (CLI based) software installation or “package management” tool. It is analogous to apt-get in Debian/Ubuntu/Mint or dnf (aka yum) in RHEL/CentOS/Fedora.
- Every time I install Manjaro (or Arch) my first command is always:

```
sudo pacman -Syyu
```

The command flags break down as: -S for “sync” with the package repository, “yy” to get the newest list from the primary mirror, and “u” to upgrade to the newest packages. As a rolling release distro, it’s important to be up to date.

## Manjaro setup III - pacman

- Besides the full system upgrade, individual utilities can be installed on the command line with just a regular pacman -S command:

```
sudo pacman -S htop ncdu tmux nmap teamspeak3
```

- Manjaro also has multiple GUI-based graphical package managers like Pamac (GTK) and Octopi (QT), these look and work similar to Synaptic in Debian/Ubuntu or DNFdragora in Fedora
- Manjaro also has full access to the Arch User Repository (AUR) for user-created packages

## Manjaro setup IV – burn-in and drivers

- For brand new hardware, consider doing a “burn-in” of the CPU and GPU. Want to do any hardware overclocking? It is a computer game unto itself. What is our CPU temperature? What GPU driver are we using?
- Historically on Linux, proprietary drivers provided by GPU vendors have performed better than the open source alternatives. While still the case for nVidia, for AMD GPUs, the open source “mesa” driver has surpassed AMD's proprietary driver for almost all but the very newest games running Vulkan.
- While I want this presentation to be very neutral in terms of hardware vendor selection (AMD vs nVidia vs Intel), the situation with driver support helped push my choice to AMD in this particular case.

## Manjaro setup V – configure Steam

- As of August 2019, Manjaro continues to include the Steam client in its installation media.
- Steam activation - just put in your code and start (re-)downloading games directly from Valve. Recovery from backup drive is very tricky and error-prone, not recommended except in emergency (extreme bandwidth limitations).
- Use Steam Cloud to sync your save games from other installs. While it is not supported out of the box, some users may be interesting in setting up a scripted “Big Picture” mode for console like Steam Box experience on boot
- Controllers: PS3 (wireless) and Xbox360 (wired) work flawlessly, PS4 also works but has poor battery life, Valve also makes a Steam Controller that works great
- Not working: Xbox One controllers, VR headsets (?)

## Manjaro setup VI – Non-native gaming

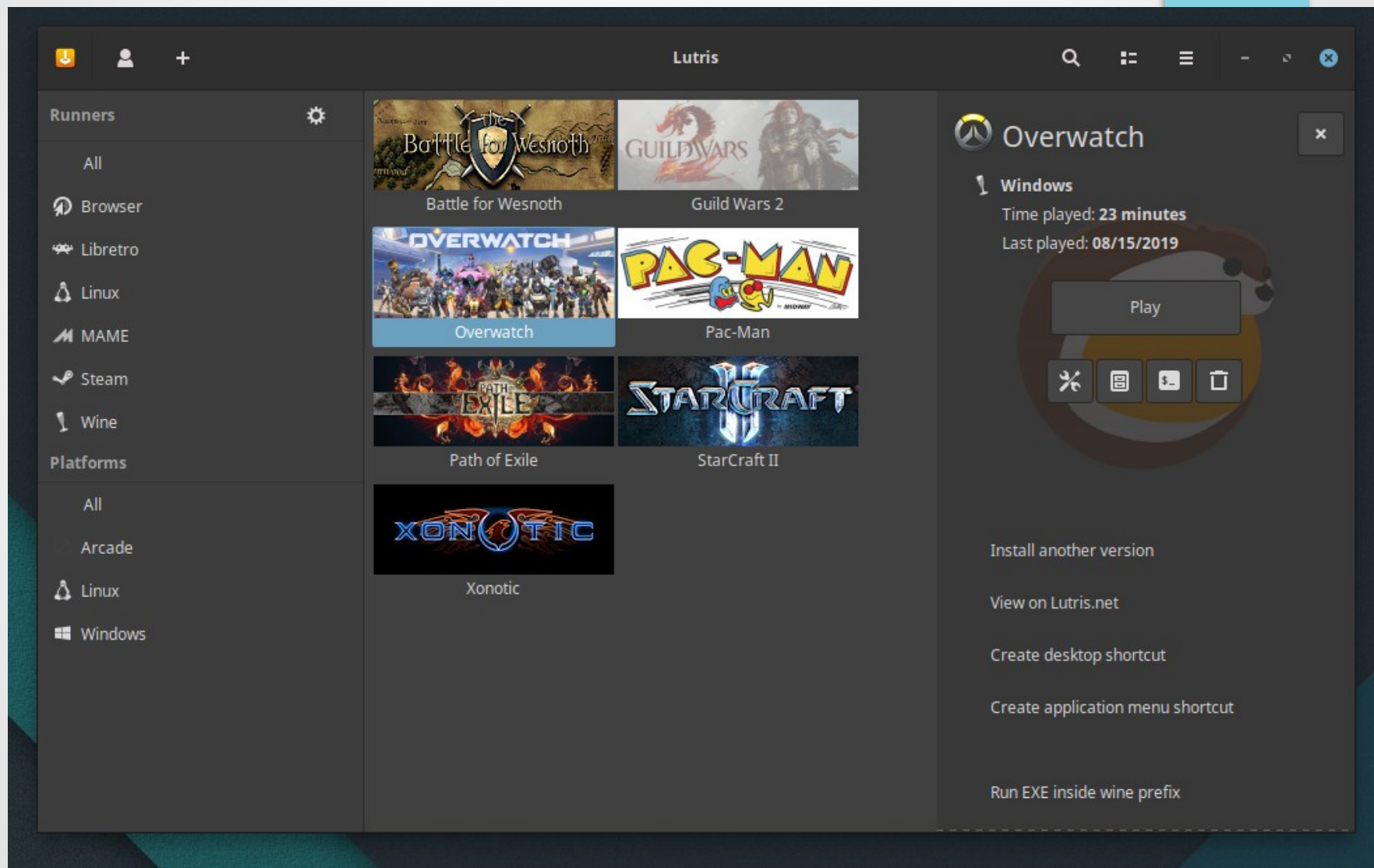
- Wine - Our favorite recursive acronym that is "not an emulator" (or a container) but rather a "compatibility layer" that translates API calls from one OS to another.
- While Wine has been around for decades, a fork of it known as Proton is now being incorporated directly into Steam as part of their "Steam Play" initiative. This offers the potential for a further increase of AAA titles available to play on Linux for the first time, though major hurdles such as DRM and "anti cheating" software remain for many titles.

## Future considerations

- Always be aware that the Steam platform is NOT open source software and may not be with us forever. It is vulnerable to competition from Google Stadia, Epic, GoG, and others, and could have Linux support pulled at any time for any reason, or no reason at all. So don't put all your Linux gaming eggs in one basket and be aware of alternatives, both new and old:
- **Lutris** - a serious attempt at an open source substitute for Steam that looks very promising. Has support for some major AAA games like Overwatch using DXVK, an implementation of the Vulkan API.
- **Crossover** - paid support release of stock (non-Proton) Wine from the upstream devs. Good for older games like Skyrim and Guild Wars 2



# Illustration 4: Lutris GUI





# Troubleshooting issues and regressions

- CPU: BIOS/UEFI compatibility issues and kernel panics (all kernels pre linux 4.18.x)
- Motherboard: UEFI boot menu doesn't like displaying on my TV so I had to do the initial install on a desktop monitor (looks like a motherboard or TV issue, not a Linux issue)
- OS regressions: screen saver, OpenGL, Bluetooth. All had workarounds or fixes documented in the Manjaro forums, each took around 1-2 hours to fix at my current skill level
- Steam: ongoing Big Picture mode crashes. Valve's Linux support can be a bit lacking at times

## Wrap-Up

- Questions/feedback?
- Any time left for a demo?
- Bonus slides? (7 total)

## Bonus Slide I – Final Assembly

- While a full guide on SFF PC assembly is outside the scope of this presentation, it should still be noted.
- As seen in Illustration 3, the only tools I needed were my “lucky” antistatic wristband and two Phillips head screwdrivers, one of them standard size for nearly all of the screws and the other one a miniature “jewelers size” used solely to mount the M.2 drive.

## Bonus Slide II – Manjaro install notes

- Download Manjaro image and make a bootable USB
- Choosing a Manjaro desktop. This is very much a matter of personal preference, KDE and XFCE seem to be the two most popular Manjaro desktops (I use KDE), GNOME is slowly growing in popularity and there are many other unofficial "community support" desktop options like Budgie, Cinnamon, Deepin, Awesome, i3, and more
- USB booting and UEFI nonsense - every motherboard has its own "boot escape" key to get into the BIOS/UEFI setup menus, my AsRock is no different, here it's F11 but you will see F1, F2, Delete, or others elsewhere. Sometimes RTFM helps here, otherwise need to Google
- Manjaro Live environment - configure WiFi and (optionally) test peripherals, then quickly move to install

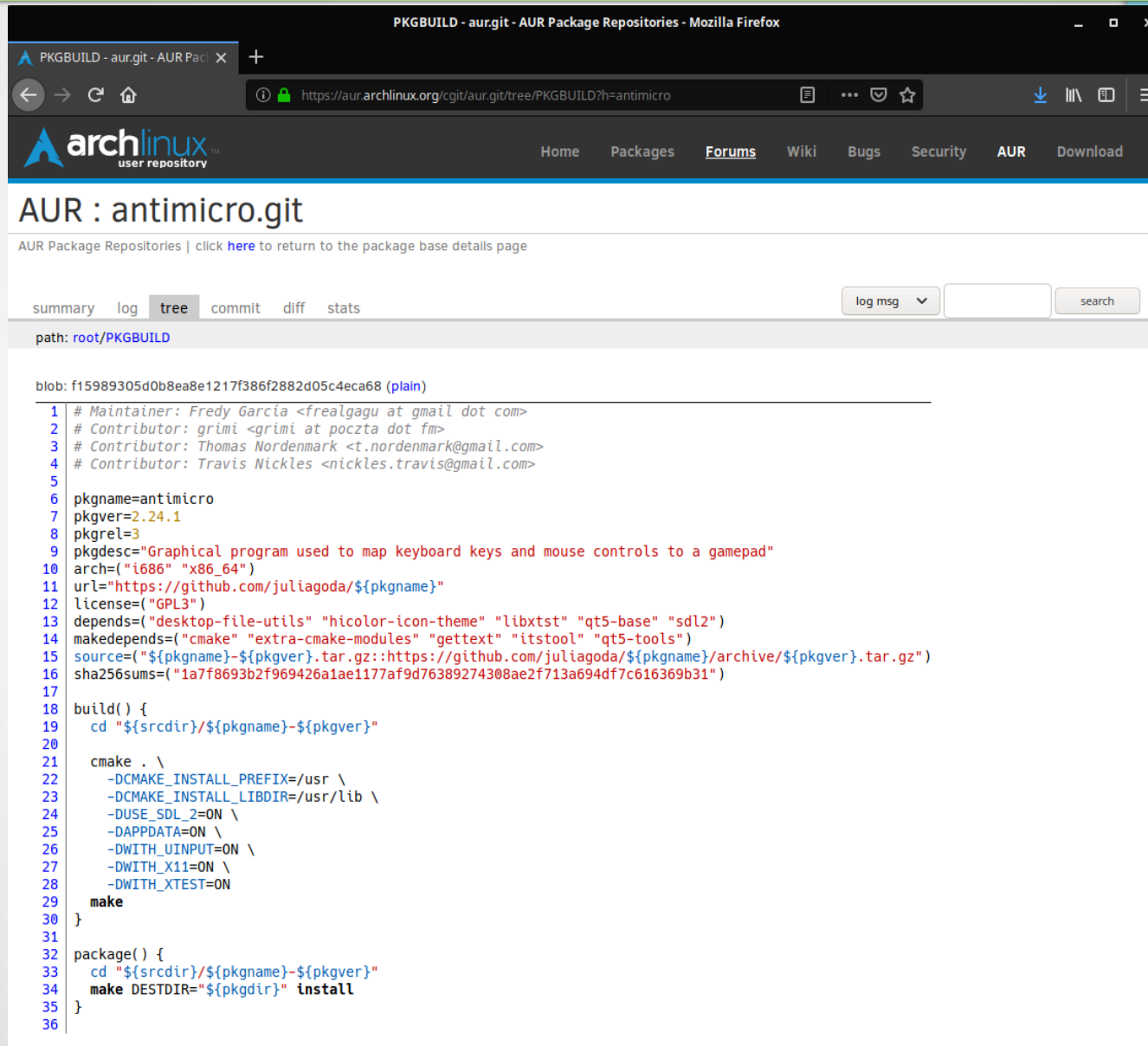
## Bonus Slide III – Manjaro install notes

- Depending on storage performance across USB drive, USB port, and internal storage, this can be VERY fast. Sometimes as quick as 5 to 10 minutes.
- Manjaro's installer asks relatively few questions during the install process, the most complicated step is drive partitioning. As I'm not really a fan of dual booting I usually just use the entire drive, accept defaults, and use LVM. Do keep an eye on swap space if you don't have much internal storage.

## Bonus Slide IV – AUR notes

- The Arch User Repository (AUR), like Ubuntu PPAs or Fedora COPR, is a framework for installing Linux software on your system outside the official OS repositories. Since Manjaro is a descendant of Arch Linux, Manjaro users have access to this resource. As a centralized system I find the AUR easier to use than PPAs or COPR, in fact it may be the single best feature of the Arch ecosystem
- However, at first glance the AUR can be a bit intimidating. Thousands of packages and maintainers, some updating constantly, others doing very little. Some installing binaries, others building from source. Fortunately, by browsing the PKGBUILD files for each package in the AUR, it is possible to make some sense of the system.

# Illustration 5: PKGBUILD script



The screenshot shows a web browser window with the title "PKGBUILD - aur.git - AUR Package Repositories - Mozilla Firefox". The address bar displays the URL "https://aur.archlinux.org/cgit/aur.git/tree/PKGBUILD?h=antimicro". The page header includes the Arch Linux logo and navigation links: Home, Packages, Forums, Wiki, Bugs, Security, AUR, and Download. The main heading is "AUR : antimicro.git", with a subtext "AUR Package Repositories | click [here](#) to return to the package base details page". Below the heading, there are tabs for "summary", "log", "tree" (which is selected), "commit", "diff", and "stats". A "log msg" dropdown and a "search" button are also present. The "path:" field shows "root/PKGBUILD". The main content area displays the PKGBUILD script for the "antimicro" package, starting with a blob hash and listing maintainers and contributors. The script includes fields for pkgname, pkgver, pkgrel, pkgdesc, arch, url, license, depends, makedepends, source, sha256sums, and build/package functions.

```
1 # Maintainer: Fredy Garcia <frealgagu at gmail dot com>
2 # Contributor: grimi <grimi at poczta dot fm>
3 # Contributor: Thomas Nordenmark <t.nordenmark@gmail.com>
4 # Contributor: Travis Nickles <nickles.travis@gmail.com>
5
6 pkgname=antimicro
7 pkgver=2.24.1
8 pkgrel=3
9 pkgdesc="Graphical program used to map keyboard keys and mouse controls to a gamepad"
10 arch=("i686" "x86_64")
11 url="https://github.com/juliagoda/${pkgname}"
12 license=("GPL3")
13 depends=("desktop-file-utils" "hicolor-icon-theme" "libxtst" "qt5-base" "sdl2")
14 makedepends=("cmake" "extra-cmake-modules" "gettext" "itstool" "qt5-tools")
15 source=("${pkgname}-${pkgver}.tar.gz::https://github.com/juliagoda/${pkgname}/archive/${pkgver}.tar.gz")
16 sha256sums=("1a7f8693b2f969426a1ae1177af9d76389274308ae2f713a694df7c616369b31")
17
18 build() {
19     cd "${srcdir}/${pkgname}-${pkgver}"
20
21     cmake . \
22         -DCMAKE_INSTALL_PREFIX=/usr \
23         -DCMAKE_INSTALL_LIBDIR=/usr/lib \
24         -DUSE_SDL_2=ON \
25         -DAPPPDATA=ON \
26         -DWITH_UINPUT=ON \
27         -DWITH_X11=ON \
28         -DWITH_XTEST=ON
29     make
30 }
31
32 package() {
33     cd "${srcdir}/${pkgname}-${pkgver}"
34     make DESTDIR="${pkgdir}" install
35 }
36
```

## Bonus Slide VI – more AUR notes

- *“Why should I care about PKGBUILD files? I just want to game!”*  
For in-line fixes and security reasons, as well as for "forking" in your own tweaks if you want to try giving back to the AUR community
- To install software from the AUR, most longtime Arch and Manjaro users use “**AUR helpers**” command-line utilities that automate the running of PKGBUILD scripts and install dependencies.
- yaourt was historically the most popular AUR helper but seems to be "going away", leaving AUR newbies with a bewildering array of successors to choose from. I use **pikaur** but others like **trizen** and **yay** are just as popular if not more so, and if still uncomfortable with the command line, you can also access the AUR through the **pamac** GUI



## Bonus Slide VII – AUR software installation

- I currently have three gaming utilities from AUR installed on my Manjaro box:

```
sudo pikaur -S phoronix-test-suite  
sudo pikaur -S crossover  
sudo pikaur -S antimicro
```

- Phoronix Test Suite is used for system burn-in and hardware benchmarking
- Crossover is the commercial support version of WINE that I use to play some Windows games
- Antimicro allows emulation of keyboard and mouse over a console controller. Unlike Phoronix and Crossover, it needs to be built from source so AUR installed a full build environment on my gaming rig as part of the install.