Stockholm Science & Innovation School (SSIS)
Gymnasiearbete
Handledare: Joakim Ågren

The Development of the Kudu Project

Tobias Per Leopold Wennberg
tobias.wenn@pm.me
Stockholm Science & Innovation School (SSIS)

Joar Alexander Pablo von Arndt
JoarxPablo@pm.me
Stockholm Science & Innovation School (SSIS)

I. Abstract

The *Kudu United Desktop Utilities* (Kudu) project's goal is to create a user-friendly operating system with a comprehensive feature set while keeping the entire system highly configurable, hackable and *free as in freedom*. It should differentiate itself using a highly customized Emacs and by employing the Emacs X Window Manager as a user environment to offer a seamless and coherent user experience. The system provides a fully custom installer with a user interface in the Emacs environment, providing an easy installation; and an configuration written in org-mode with in-line Emacs lisp that is easily and fully hackable. The project was written over the course of several months with parallel development being carried out on the GNU Emacs frontend and GNU Guix installer. While it does not come with a pre-made ISO file, Kudu was ultimately successful at achieving its goal. Licensed under GPL V3.0, the project is openly hosted on GitHub, fostering a collaborative and transparent development process.

Content

I. Abstract **2**

II. Introduction **5**

- II.I. Purpose 5
- II.II. Background 5
- II.III. Method 5

III. Dissertation **6**

- III.I. Frontend 6
  - III.I.I. What is Emacs? 6
  - III.I.II. Org mode 6
  - III.I.III. Presentation 6
  - III.I.IV. The Emacs X Window Manager 7
  - III.I.V. Portability 7
  - III.I.VI. Startup time minimization 8
- III.II. Backend 8
  - III.II.I. Setup installation disks 8
  - III.II.II. Booting 10
  - III.II.III. User Interface 11
  - III.II.IV. The Installation Script 11
- III.III. Conclusion 13

Bibliography **14**

I. Appendix **16**

- I.I. Code 16
  - I.I.I. LICENSE (GPL-3) 16
  - I.I.II. config.org 25
  - I.I.III. early.init 37
  - I.I.IV. init.el 38
  - I.I.V. install 39
    - I.I.V.I. Installer 39
      - I.I.V.I.I. bare-bones.scm 39
      - I.I.V.I.II. config.scm 40
      - I.I.V.I.III. get_disks.sh 40
      - I.I.V.I.IV. get_disks_test.sh 40
      - I.I.V.I.V. get_keymaps_test.sh 40
      - I.I.V.I.VI. guix_config.scm 41
      - I.I.V.I.VII. guix_iso.scm 42
      - I.I.V.I.VIII. installer.el 43
      - I.I.V.I.IX. installer.el 45
      - I.I.V.I.X. install_iso.sh 47
      - I.I.V.I.XI. install_quick.sh 49
      - I.I.V.I.XII. install.sh 49
      - I.I.V.I.XIII. install_test.sh 51
      - I.I.V.I.XIV. keymaps 52
      - I.I.V.I.XV. logo.ascii_art 54
      - I.I.V.I.XVI. part_iso.sfdisk 54
      - I.I.V.I.XVII. part.sfdisk 55
      - I.I.V.I.XVIII. template.scm 55

I.I.V.I.XIX. timezones 56
I.I.V.I.XX. test_get_parts.sh 64
I.I.V.II. quick-init.el 64
I.I.V.III. snippets 65
I.I.V.III.I. org-mode 65
I.I.V.III.I.I. cases 65
I.I.V.III.I.II. cases 65
I.I.V.III.I.III. display_math 66
I.I.V.III.I.IV. fraction 66
I.I.V.III.I.V. fraction_dollar 66
I.I.V.III.I.VI. fraction_dollar_2 66
I.I.V.III.I.VII. f(x) 66
I.I.V.III.I.VIII. g(x) 66
I.I.V.III.I.IX. infinity 67
I.I.V.III.I.X. integral 67
I.I.V.III.I.XI. integral_dollar 67
I.I.V.III.I.XII. integral_dollar_2 67
I.I.V.III.I.XIII. L 67
I.I.V.III.I.XIV. lhd 67
I.I.V.III.I.XV. limit 68
I.I.V.III.I.XVI. limit_dollar 68
I.I.V.III.I.XVII. limit_dollar_2 68
I.I.V.III.I.XVIII. mathbb 68
I.I.V.III.I.XIX. rhd 68
I.I.V.III.I.XX. sim 68
I.I.V.III.I.XXI. sube 68
I.I.V.III.I.XXII. subset 69
I.I.V.III.I.XXIII. sum 69
I.I.V.III.I.XXIV. sum_dollar 69
I.I.V.III.I.XXV. org-mode 69
I.I.V.III.I.XXVI. sup 69
I.I.V.III.I.XXVII. org-mode 69

II. Introduction

II.I. Purpose

There are already numerous distributions of the GNU operating system on the market, and many are very similar. Kudu is differentiated by providing GNU Emacs' exwm with a superior default configuration on a GNU Guix base for a unique experience reminiscent of the lisp machines of old. While it is not difficult for an intermediate GNU OS user to install GNU Guix with exwm, it may be annoying to do many times, and it is not accessible for beginners, causing them to choose less extensible solutions for desktop frontends. The goal of Kudu is to create an OS that the user can fully extend to their own needs, rather than something imposed upon them.

II.II. Background

In 1976, TECMAS and TMACS was released. It was a hackable text editor written by Guy Steele, Dave Moon, et al. In 1978, EMACS (acronym for "Editor Macros") was started as a project to unify editor macros that until then was diverse. The project was started by one of the authors of TMACS, Guy Steele; and developed with Richard Stallman, the future founder of the GNU Project, and Richard Greenblatt (GNU-Project, 2024a; Zawinski, 2007). A decade later, Richard Stallman announces the founding of the GNU Project, aiming to create a fully free-as-in-freedom operating system loosely based on the popular but proprietary UNIX operating system, the grandfather of today's Berkeley Software Distribution (BSD) and MacOS used by Apple's stationary and portable computers. This new operating system needed various tools, one of which was a fully featured text editor. In 1984, the GNU Project chose to create their own version of EMACS, named GNU Emacs, for this aim. Just a year later, Emacs Lisp (elisp) was released, an interpreted programming language used for emacs configuration. Elisp allows for the customization of emacs, with important subprograms such as a PDF viewer, the markup language org-mode, a file manager, an email client, web-browser *et cetera* being developed with it. This allowed emacs to replace the entire user-facing part of the operating system, but it did not allow for using external graphical applications, it needed a dedicated emacs app to work, and it needed a window manager to be launched. On July 17, 2015, Chris Feng released the Emacs X Window Manager (exwm) (Emacswiki, 2024; Feng, 2024). Exwm revolutionized the ability to work entirely in emacs, allowing both emacs native programs, and external graphical programs to work within the emacs workflow. The exwm lifestyle allowed for seamless text editing, note-taking with org-mode, web browsing with your choice of web browser, terminal editing, and programming without ever-changing keybindings or environment. There is no alternative that accomplishes this goal, despite calls for the development of a similar package aimed at the Wayland display server (Bauer, 2022).

GNU Guix released its alpha in 2013. Inspired by Nix OS released in 2003, it came with its own Guix package manager which allowed total system configuration in the GNU Ubiquitous Intelligent Language for Extensions (GUILE). This allowed for great system configuration in a single file, so that two systems can have the exact same dependencies. It also allowed for an entirely GNU operating system, even the kernel where one could choose between the Linux Libre or the GNU Hurd kernel.

II.III. Method

Research for the installer came from reading official documentation from GNU, reading the mailing list and consulting independent lisp hackers. We asked for help from the r/Guix subreddit and the official GNU Guix mailing list. Most research was made by trial and error with a large amount of manual testing in a local KVM virtual machine.

For the frontend, inspiration was taken by collecting material on the popularity of various Emacs packages as collected by the emacs user survey (Brochard, 2020), and by careful study of various preexisting modifications to GNU Emacs. Especially influential was Rougier's *On the design of text editors* and corresponding *N Λ N O - Emacs* (Rougier, 2020; 2024). The development was carried out on a machine running the Fedora GNU/Linux distribution while decisions on the final structure of Kudu were still being made.

Wennberg was put in charge of designing and implementing an installer interacted with through Emacs, while von Arndt continued work on the GNU Emacs frontend and workflow.

III. Dissertation

Kudu originally began as the personal configurations files for GNU Emacs of von Arndt. The popularity of so-called *distributions* of GNU Emacs, such as DOOM Emacs (Lissner, 2024) and spacemacs (Benner, 2024) and the widespread division of GNU/Linux into similarly denominated distributions contributed to the idea that the unification of the two was possible. It is a famous aphorism made by users of *vi*, the ancient rival of Emacs, that

> Emacs would be a great operating system, if only it came with a text editor.
>
> — (Duan, 2024)

The idea was therefore extended to become a fully featured operating system. The work was divided into two parts: the desktop environment, encompassing configurations for GNU Emacs and other user-facing software, and the installer encompassing the backend configurations.

III.I. Frontend

III.I.I. What is Emacs?

GNU Emacs is not a text editor, it is a C program that is fully extensible using its own dialect of the lisp family of programming languages. The original EMACS, was just a collection of pre-packaged extensions meant for another text editor, and the majority of GNU Emacs packages today are really nothing more than incremental extensions of emacs lisp written on top of this base written in C. Emacs is of course most famous as a text editor, but that is merely because it happens to be shipped with a decent one built-in. In reality GNU Emacs would be more aptly described as a modern-day extensible lisp machine reminiscent of the workstations of the 1970s. Its long history has made it the superior tool to be used when interacting with any form of text, whether through editing or through the presentation of the written word.

It has long been a stated goal of Emacs users, stated somewhat jokingly, to "live in Emacs"; it has even become the tagline for the exwm project's page on GitHub (Feng, 2024). This is the natural extension of the Ellulian and Mumfordian concept of technics (Ellul, 2011; Mumford, 1971) where the application of technique extends to every area in which it can be conceivably applied. Emacs' extensible nature makes this particularly easy, the entire application is designed from the ground up to be the perfect environment for working with the most efficient of all media; text.

III.I.II. Org mode

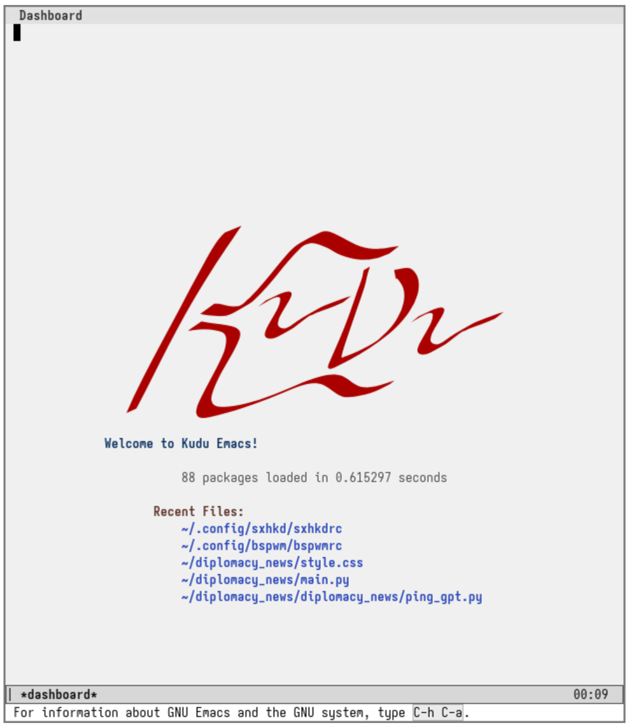
One of the so-called *killer apps* for GNU Emacs is Org-mode, "A GNU Emacs major mode for keeping notes, authoring documents, computational notebooks, literate programming, maintaining to-do lists, planning projects, and more — in a fast and effective plain text system" (Dominik, 2003). The literate programming part of org is especially interesting, as code can be written directly in prose documents and run discretely, or exported as full files to be run externally. Kudu uses this functionality to create a fully self-documenting system, where the documentation *is* the program, rather than being something imposed upon it. This means that a user of Kudu will fully understand every part of the user-experience and allow them to easily modify it as they see fit. This is visible in Appendix I.I.II.

Kudu also comes with significant modifications paired with the org-mode major mode. Most of these are cosmetic in nature, changing the appearance and visibility of text, but some provide additional functionality not present in GNU Emacs by default. One of these is the inclusion of snippets for the `yasnippet` package that significantly improve the speed at which in-line LaTeX can be written. Most of these are inspired by the completion offered by the `AUCTeX` major mode for the production of LaTeX documents, as well as the `CDLaTeX` minor mode (Dominik, 2019; GNU-Project, 2024b).

A popular usage for org-mode is pairing it with systems for the so called *zettelkasten* organizational method (Kuan, 2022). This is however not directly supported by Kudu, simply due to the fact that one's management of information is deeply personal and is best handled and structured after the user's own habits and needs.

III.I.III. Presentation

When new users first install GNU Emacs they are initially confronted with a lot of information. The default startup screen for GNU Emacs contains a lot of information, including links to the built-in Emacs tutorial, a guided tour, the in depth Emacs manual, the fact that GNU Emacs is provided without any warranty, and how to order printed manuals. This is superfluous for a long-time user of Emacs, and so are the two rows of clickable menus with command easily accessible through the universal `M-x` shortcut, allowing the user to run any interactive Emacs lisp command. Kudu chooses to discard this, instead providing easy documentation through the *marginalia* package that displays explanatory notes for interactive elisp functions.



```
Dashboard

Welcome to Kudu Emacs!

            88 packages loaded in 0.615297 seconds

        Recent Files:
            ~/.config/sxhkd/sxhkdrc
            ~/.config/bspwm/bspwmrc
            ~/diplomacy_news/style.css
            ~/diplomacy_news/main.py
            ~/diplomacy_news/diplomacy_news/ping_gpt.py

| *dashboard*                                                    00:09
For information about GNU Emacs and the GNU system, type C-h C-a.
```

Figure 1: Kudu as shown on startup, with a collection of recent files displayed using the *modus-operandi* theme developed by Protesilaos Stavrou.

This is only one of the changes made to the GNU Emacs interface, and it is not one that is very controversial amongst veteran users. One of the more unusual changes made is the inclusion of a custom header- and modeline, as can be seen in Figure 1, displaying information like the current major mode, buffer name, and time without the unnecessary clutter of the default mode-line.

Kudu also comes with a suite of programs intended to ease programming in different languages, specifically those in the lisp family. For this packages like `rainbow-delimiters`, `smartparens`, and the legendary `paredit` modes are included by default. Kudu also uses the popular fork of *The Superior Lisp Interaction Mode for Emacs* (Gorrie, 2024) knows as *Sly: Sylvester the Cat's Common Lisp IDE*. The reason for using `sly` over `SLIME` is merely due to the fact that when a `sly` session is started, it displays an ASCII-art drawing of a cat, something that `SLIME` does not do.

III.I.IV. The Emacs X Window Manager

The most transformative difference between Kudu and other traditional GNU/Linux distributions is the fact that the user instantly enters an environment wholly interacted with through Emacs keybinds and through emacs lisp functions instead of a mixture between keybinds intended for emacs and those intended for the window manager. GNU Emacs predates the concept of the Graphical User Interface (GUI) and also the idea of the modern-day conception of the "window" as an indivisible init, instead emacs uses three concepts to display content:

Frame	The largest unit, commonly what is called a "window".
Window	Areas of the frame divided up vertically and horizontally to make space for buffers.
Buffer	A unit of information for display. This may be a file, an elisp program, or in the case of exwm, another X window.

Since GNU Emacs covers the entire monitor in order to manage other windows, frames are merely used to provide multi-monitor support, multiple workspaces, and are used to show floating windows. A core difference between the way Emacs manages buffers and how more popular window managers do things, is that a buffer needs not be shown at all times. This makes ideas like workspaces unnecessary in practice, as content can be displayed and hidden in mere moments.

III.I.V. Portability

But Kudu does not merely include support for a Guix system running exwm, even if that is the primary targeted platform. GNU Emacs can run on a variety of different machines, including proprietary operating systems like Microsoft Windows, and non-GNU machines running the Linux kernel like Android. For these machines, where special tooling or certain functionality may not be available or wanted, certain changes must be made. One example of this is the in-buffer completion prompts offered by the `corfu` package. `corfu` use *child frames* to show prompts, but these are not available in areas were only one frame is available, as when Emacs is run with the `-nw` flag for use in a terminal. Prompts are instead shown as elaborately formatted text boxes, that display the same information without compromising on appearance.

Similarly, Kudu does not load the otherwise quite large part of code covering configuration and startup of exwm, and also avoids executing multiple external programs to fetch information that otherwise may not be available when Kudu is run on a non-Guix system platform.

III.I.VI. Startup time minimization

Emacs is notorious for often taking multiple seconds to start if configured haphazardly. For this reason multiple techniques are employed to minimize the number of packages loaded and how those packages that need to be loaded are handled. The ideal target to aim for is the legendary "Doherty Threshold" (Crum, 2020) of less than 400 milliseconds, making interaction with the computer practically instantaneous. While this was often times not attainable, it is not a major concern due to the fact that restarting emacs is not done very frequently even under ordinary circumstances, and especially not when Kudu is run under its intentional use of being the X window manager.

As described in the gccemacs documentation (Corallo, 2021); The version of emacs shipped with Guix is compiled with the `--with-native-compilation` flag that allows for the compilation of elisp to native code, significantly increasing speed. For this the `libgccjit` library is used, inspired by the very fast Steel Bank Common Lisp (SBCL) implementation of the ANSI Common Lisp standard. Kudu then compiles `.el` libraries into the `.eln` native file format upon first load without any required input from the user.

Native code contributes significantly to increased responsiveness when working in the GNU Emacs environment, but its effect on the perceived startup speed, usually the single slowest operation in a given session of using GNU Emacs, is negligible. Instead, we should attempt to shift what is by default a front-heavy workload over a longer period of time, perhaps when the user has already started performing operations. Primarily the RAM limit before garbage collection is performed is set to an arbitrarily large size, and then set to a more reasonable limit after the startup sequence has been completed. This can be observed in the beginning portion of the `early-init.el` file in Appendix I.I.III.

Through the application of these methods, the time required for GNU Emacs to start with packages used is decreased by roughly two powers of ten. This is especially noticeable on low-power devices, where single-threaded performance is often limited in comparison with more powerful machines.

III.II. Backend

The backend of Kudu is configured during the installation. During the installation, one need to consider many parameters to make the system work as one which. Installation of any GNU/Linux system consist of at least 5 steps:

1. Setup installation environment
2. Setup installation disk
3. Install packages to disk
4. Configure the environment
5. Configure bootloader

(arch-linux, 2024a). Kudu also needs a user interface as it is meant to be user-friendly.

Setting up the operating system environment in a manual installation of Guix involves creating a bootable disk of the Guix iso, booting guix (see Section III.II.II about booting), setting the keyboard layout and connecting to the internet (GNU-Project, 2024c). Kudu wishes to make most of these steps trivial or non existent.

III.II.I. Setup installation disks

Setting up installation disks involves two steps: partitioning and formatting.

A partition is a region of a disk. It is typically used to separate the disk so that different parts can be used for different purposes. The partition data is stored on the disk device, the data includes the start- and end sector (where the region is on the device), the partition type, and if the partition is bootable or not (Ward, 2004). The Kudu installer utilize sfdisk to partition the drives as it is the standard utility on GNU systems. Sfdisk can be used to configure disk partition via a partition schema file. The Kudu partition schema looks like this:

```
label: gpt
label-id: [label id]

start=2048,size=4096,type= [efi_boot],,
bootable
start=6144,size=2097152,type=
[linux_swap],
start=2103296,size=1G,type=
[linux_partition],
```

Table 1: The sfdisk partition schema of Kudu, rewritten to be more readable but non functional. The real partition table can be found at Appendix I.I.V.I.XVII

There are three partitions: the boot partition, swap partition and root partition. The boot partition is further elaborated under Section III.II.II. The swap partition is used for linux swap; a form of virtual memory where memory can be moved to if the system run out of real memory (Ward, 2004). The swap is 2GB which is a typical amount for modern systems that don't make use of hibernation. The root partition is the partition where the system and user data is stored. The size of the root partition is the rest of the disk. Partition data is written as follows:

```
sfdisk -f $disk < part.sfdisk
parted -s $disk resizepart 3 100%
```

Each partition needs to be formatted. The swap partition is formatted to swap with `mkswap $swappartition`, the rest need to be formatted with a filesystem. A filesystem is a system to manage files (and directories). These are a fundamental part of most operating systems, as they rely on files and directories to store all data (arch-linux, 2024b). The filesystem can be physical or virtual. Examples on virtual filesystems are tmpfs and rootfs, these live in ram and are volatile. Rootfs is used by the linux kernel as the first created filesystem, they are convenient as they do not require device drivers to work. They are also used by the GNU system tails OS so that the writable filesystem is volatile and data perish between bootup, which may be considered a security feature. One could also count application specific filesystems into this category, such as smb, nfs, virtiofs and similar. A physical filesystem stores data on a disk. Most disks make the data non volatile and the storage cheaper, but slower than the ram based, virtual filesystem. Kudu do not need to consider a virtual filesystem, but it does need to consider a physical one. To configure a physical filesystem, one need to configure the disk to work with the filesystem and then start it. To configure the disk, one format it. Format the disk means to set the bits on the disk to work with the filesystem driver, this usually removes all data on the disk. To format a disk to fat, one would run `mkfs.fat $disk` on a GNU system. To start the filesystem on a GNU Linux system, one would typically download and configure the drivers to the linux kernel, and then mount the filesystem. Different filesystem comes with advantages and disadvantages. Some are faster on flash, some are faster on mechanical drives, some support encryption, some support online expanding, some support shrinking. It is important to choose the right filesystem for a good user experience.

Kudu has two partitions that require filesystems: the boot- and root partition. For the boot partition, we will use fat32 as it needs to be supported by the UEFI boot interface (expanded upon at Section III.I-I.II). UEFI demands support for the fat12, fat16 and fat32 filesystems. Fat32 is the best of these for sizes exceeding a few megabytes. There are many possible choices for the root partition's filesystem. The most popular for desktop use are btrfs, ext4, zfs, ntfs, f2fs, fat, vfat. Ntfs is used in the Microsoft Windows operating system, but the drivers provided for it are lacking in many areas, and so it is not a serious contender for use by Kudu. Fat derived file systems (fat8, fat12, fat16, fat32 and vfat) suffer severely from reduced speed after many small files are accumulated, and so are not a good choice for kudu either. F2fs does not support the convenient features such as resizing the filesystem except for offline enlarging, and while it is very fast on flash storage, it experiences very slow speeds on spinning disks and so is not useful on a system intended to be compatible on any machine. Zfs on the other hand is very useful for raid arrays, but for single disk operations is it overkill and suffers from high memory usage. That leaves btrfs and ext4. For these reasons they are also the most popular choices for GNU systems. Here is a table with the file systems of select popular desktop GNU systems:

btrfs	**ext4**
fedora	debian
suse	RHEL
pop os	ubuntu

For now ext4 was chosen as it has more stable drivers and is a slightly faster than btrfs in most applications. It would not be difficult to change to another file system if the project wishes to do so in the future.

III.II.II. Booting

The Kudu project is built for personal computers using the x86 instruction set as defined by Intel; with the 64 bit instructionset extension as defined by Advanced Micro Devices (AMD), often called x86_64, x64 or amd64; with an architecture platform firmware compatible with the *Unified Extensible Firmware Interface* (UEFI) specification. These include most typical, modern, personal computers; although some may still use the *Basic Input/Output System* (BIOS) which Kudu is in-officially supported on.

UEFI is a standard for booting (starting) the operating system, and to interface with hardware devices. It is supported by most modern personal computers, with some older machines only supporting BIOS and/or EFI - the predecessors to UEFI (arch-linux, 2024c). The first step an UEFI system performs after power on is a *power on self test* (POST), where the firmware checks if the system works. Following a successful POST, the firmware searches for *efi partitions*, which is a type of partition record that is bootable (partition tables are explained at Section III.II.I). The firmware will choose which one to boot off by checking the boot order in the non-volatile *nvram* that the firmware uses as local storage. After choosing the boot partition, the firmware will try to mount the partition. UEFI mandates support for fat12, fat16 and fat32 filesystems, but a motherboard may support more; so one of these should be used by the EFI partition. When it is mounted, it will run the bytecode at `/EFI/Boot/bootx64.efi`. That file will be the start binary of our choosing.

In order to start a GNU operating system, one need to start a kernel (GfG, 2023). The kernel is a primary part of operating system design. Its primary functions are memory management, task management, threading, and communication with hardware components (GfG, 2023). There exists many kernels, some popular ones for server or personal computers are the linux kernel, Microsoft Windows NT Kernel and FreeBSD Kernel. Kudu needs to consider these points: the kernel needs to be free software as according to fsf (fsf, 2024a), it needs to be compatible with guix, it needs to be usable by most consumers. Guix officially supports two kernels: the GNU hurd kernel and the linux libre kernel. Both of these are recommended by the FSF (fsf, 2024b; FSFLA, 2024). linux libre is preferred of these as it has a lot better driver support, and thereby works on more systems. Another alternative is the official linux kernel, which linux libre is a fork of. While the linux kernel is free software, the linux-firmware repository contains a significant amount of non free binaries (Linux_Kernel_Organization, 2024). This firmware is used to initialize and make devices work, and a significant amount of device drivers is only available as non free binaries. Linux libre protects the user from loading non free firmware, and do not provide non free binaries by default. While one can make linux work pretty easily on guix, it is not offically supported. Because it is not offically supported, and may be non free, linux libre kernel maintained by the Free Software Foundation Latin America was preffered instead of linux.

The kernel can be started in many ways. The UEFI can start the kernel using technologies ike the *unified kernel image* (UKI) or efistup (arch-linux, 2024d; 2024e). Typically, one would use a boot-loader as an intermediate step between the kernel and UEFI. A boot managers purpose is to set kernel parameters and to load external initramfs images (important driver binaries); they typically support functionality like dual booting, where one can conveniently choose which operating system to boot; and support for rollback on operating systems that themselves provide support (arch-linux, 2024f; RHEL, 2024). There are many boot managers available to choose from, for example: systemd-boot, refind, lilo; Kudu will be using the *GRand Unified Bootloader* (grub), as it is the most popular, has a large feature set, has great integration with the guix configuration schema and is maintained by the GNU Project that also develops Guix.

When user setup is done and grub is ready to run linux-libre, it will mount the root file system as according to the configurations provided; load any appropriate initramfs images, which are used by the kernel to load appropriate drivers, and start the linux-libre kernel located at `/boot/vmlinuz-{version}-amd64` according to the configuration, and hand control over to it. The kernel creates the virtual filesystem (rootfs) and copies the initramfs into it, kernel modules are set up and the root filesystem is mounted at `/sysroot` and switched into (arch-linux, 2024f; RHEL, 2024).

The linux kernel launches one process - PID 1 - which is typically the init system. The most popular one is *systemd-init*, used on popular systems such as Debian, Ubuntu, Red Hat Enterprise Linux, Arch Linux and SUSE Linux; popular alternatives include *openrc*, *s6*, *sinit* and *runit*, some of these has built in init, while others are service managers and needs it as a second init dependency; guix comes with its own init system and service manager, *GNU Shepherd*. The init system starts the rest of the processes, such as a login screen, networking and a lot more; it is the parent or grandparent of all processes. Now, the system is running (arch-linux, 2024f; 2024g; RHEL, 2024).

III.II.III. User Interface

In any development project, it is important to choose the right technology. With the installer, one of the important choices was the technology for the text user interface (TUI), as when chosen, most code written will be impossible, or very time-consuming, to port to another technology. As the project is Emacs centric, it would be preferable if the user enters the emacs environment immediately. That means the TUI should be written in the elisp programming language, as it is the only programming language that emacs is extended in. Emacs is really nothing more than a C program running an elisp interpreter, and so any capabilities afforded to emacs are directly accessible from elisp. This includes the ability to spawn buffers, setting write permissions on buffers, adding text to buffers and other emacs features such as macros. These features provide the ability to create more complex user objects such as text boxes, radio buttons, check boxes and buttons. Doing this in the shell would be very annoying, as one would need to create all of these complex, high level object from those very simple low level functions (Dickey, 2022). The Emacs Widget library provides tooling for these complex objects, vastly simplifying the process of creating buffers with TUI like functionality (GNU-Project, 2024d). An alternative to widget is tui.el, A react-js inspired framework for building a TUI in emacs. Emacs widget was chosen as it seem to have better documentation, bigger community, and as it comes with emacs 29.1, the risk of it experiencing breaking changes is very low.

III.II.IV. The Installation Script

As elisp is neither systems- nor command language, one cannot with any comfort or speed use it to configure the system. Another language is needed for system setup. Using a system language, such as C, C++, rust or zig would allow for great performance and flexibility, but would require one to write a lot of "boilerplate" code, and the type requirements would make it somewhat difficult to interact between premade commands. A command language, such as shell, the *Born Again SHell* (bash), Z shell, fish and xonsh is a language to operate a computer via commands; typically, they provide the ability to use it as the primary interface in a terminal, and the ability to write scripts to automate the commands. All the languages listed above provides these features. Kudu chose to use bash as it is mature, widely implemented and comes with Guix. The same things can be said about shell, but its scripting features are limited in comparison with bash. Shell is used throughout the installation as its binary location is constant at `/bin/sh`, while location of the bash binary may be unknown. Both bash and shell are compatible with the Portable Operating Systems Interface (POSIX) standard defined by the Institute for Electrical and Electronics Engineers (IEEE). This means that most commands written in shell can be run using the bash binary, but not all commands written in bash can be run by shell (GNU-Project, 2020).

Shell is used for purposes in the frontend, like the script to fetch the disks (available at Appendix I.I.V.I.III), while bash is used for the primary install-script. It is called from the emacs frontend using elisp with this function:

```
(defun upload (hostname username disk timezone keymap)
  (setq cmd (format
      "bash ../installer/install.sh --hostname %s --username %s --disk %s --timezone %s --keymap %s &"
    hostname
    username
    disk
    timezone
    keymap))
  (shell-command cmd))
```

Table 2: Function to start installscript from installer script. Available at Appendix I.I.V.I.VIII.

The first thing the install-script does is formatting the disk, as described in Section III.II.I, mounting the root disk at `/mnt`, and starting the cow-store at `/mnt`. The *copy on write*-store is the location where the guix package manager will write. As it is mounted on `/mnt`, any downloads, pulls and writes the guix package manager does will be written to the installation disk. Then it initializes the Guix config.

Inspired by NixOS and its nix package manager, the Guix package manager uses a scheme language to describe the system that should be installed. In the config file, one can state which packages should be globally installed, which users should exist, which services to enable, custom services, disks and much more. The Kudu Guix config can be found at Appendix I.I.V.I.VI. The packages required are explained in the frontend section, the disks in the disk setup as explained in the Section III.II.I. The services are the default services for getting the system working, `gnome-desktop` for the display manager and some xorg services for the display server. The bootloader is configured with:

```
1 (bootloader
2   (bootloader-configuration
3     (bootloader grub-bootloader)
4     (targets '("$DISK")))
5   (theme
6     (grub-theme
7     (resolution '(1920 . 1080))
8         (image  (local-file  "/mnt/etc/Kudu_grub_image.svg")))))
```

Table 3: Bootloader configuration from the guix config file, file available at Appendix I.I.V.I.VI.

The custom configuration is a Kudu theme, the rest of the code is pulled from the guix wiki. The theme simply load an image at 1080p resolution.

Many values are variable, such as the timezone, username, hostname, disk, and filesystem uuid's. These are set in the config file with a $'sign, as seen at line 4 Table 3. The variables are inserted via the install-script with this function.

```
function substitute_variables() {
  local str="$1"
  shift
  for var; do
  str="${str//\$$var/${!var}}"
  done
  echo "$str"
}
```

Table 4: Function to substitute variables in a string. Code available at Appendix I.I.V.I.XII.

Which is called with

```
$(substitute_variables  "$scheme_template"
DISK HOSTNAME USERNAME SWAP_UUID ROOT_UUID
TIMEZONE KEYMAP)
```

To retrieve the string with the inserted variables. The finished string with the variables is written to `/mnt/etc/config.scm`. The system then is installed with.

```
guix pull
guix package -u
hash guix
guix pull
guix package -u
hash guix
guix system init /mnt/etc/config.scm /mnt
```

Table 5: The install part of the install script, code available at Appendix I.I.V.I.XII.

The repeated `guix pull` is done to ensure the guix repositories are properly pulled. `guix system init` initialize the system with the packages, now the system should be installed and booting.

Nextup, the emacs configuration files need to be installed. The configuration files should exist at `~/.emacs.d` (where `~/` is the home directory of the user). This path is equivalent to `/mnt/home/$USERNAME/.emacs.d`. The configuration is the root of the kudu git-repository https://github.com/

JanJoar/Kudu-Emacs.git. We just need to clone it to the directory

```
git clone https://github.com/JanJoar/Kudu-
Emacs.git /mnt/home/$USERNAME/.emacs.d
```

III.III. Conclusion

The Kudu project has successfully crafted a GNU Guix distribution with its own custom installer architecture and a unique user environment with special focus on interacting in the GNU Emacs environment. It's special focus on interacting through the emacs interface has made it stand out amongst its competitors, and the valuable addition of numerous extensions and modifications made to GNU Emacs core allows the user to work in a seamless emacs-oriented environment. The fact that almost all of Kudu's emacs configuration is written in org-mode facilitates the spread of knowledge to emacs users who may not be very familiar to the GNU Emacs ecosystem and package environment. It is the hope of the authors of this document that the additions of Kudu to the world's total bank of information will facilitate the spread of knowledge about reproducible systems like GNU Guix and of their benefits in the fast and easy deployment of numerous machines.

Bibliography

arch-linux. (2024f). *Arch boot process.* https://wiki.archlinux.org/title/Arch_boot_process

arch-linux. (2024a). *Installation guide.* https://wiki.archlinux.org/title/Installation_guide

arch-linux. (2024c). *Unified Extensible Firmware Interface.* https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface

arch-linux. (2024e, March). *EFISTUB - ArchWiki.* https://wiki.archlinux.org/title/EFISTUB

arch-linux. (2024b, March). *File systems - ArchWiki.* https://wiki.archlinux.org/title/File_systems

arch-linux. (2024g, March). *init - ArchWiki.* https://wiki.archlinux.org/title/Init

arch-linux. (2024d, March). *Unified kernel image - ArchWiki.* https://wiki.archlinux.org/title/Unified_kernel_image

Bauer, M. (2022). *Emacs should become a Wayland compositor.* https://emacsconf.org/2022/talks/wayland/

Benner, S. (2024). *Spacemacs.* https://github.com/syl20bnr/spacemacs

Brochard, A. (2020). *Emacs User Survey.* https://emacssurvey.org/2020/

Corallo, A. (2021). *gccemacs.* https://akrl.sdf.org/gccemacs.html

Crum, L. (2020). Laws of UX: Using Psychology to Design Better Products & Services,. *Design and Culture*, *12*(3), 357–359. https://doi.org/10.1080/17547075.2020.1822074

Dickey, T. (2022). *Ncurses Overview.* https://invisible-island.net/ncurses/announce.html#h2-overview

Dominik, C. (2003). *Org-mode: Your life in plain text.* https://orgmode.org/

Dominik, C. (2019). *CDLaTex.* https://staff.fnwi.uva.nl/c.dominik/Tools/cdlatex/

Duan, C. (2024). *Vi(m): The One True Text Editor.* https://www.cduan.com/technical/vi/

Ellul, J. (2011). *The Technological Society* ([Nachdruck der Ausgabe] New York, Knopf). Vintage books.

Emacswiki. (2024). *Emacs X Window Manager.* https://www.emacswiki.org/emacs/Emacs_X_Window_Manager

Feng, C. (2024). *Github exwm.* https://github.com/ch11ng/exwm

fsf. (2024a, February). *What is free software and why is it so important for society? Free Software Foundation Working together for free software.* https://www.fsf.org/about/what-is-free-software

fsf. (2024b, April). *Hurd - Free Software Directory.* https://directory.fsf.org/wiki/Hurd#tab=Overview

FSFLA. (2024). *GNU Linux-libre, Free as in Freedo.* https://www.fsfla.org/ikiwiki/selibre/linux-libre/

GfG. (2023). Kernel in Operating System. *Geeksforgeeks.* https://www.geeksforgeeks.org/kernel-in-operating-system

GNU-Project. (2020). *GNU Bash.* https://www.gnu.org/software/bash/

GNU-Project. (2024a). *The GNU Emacs FAQ.* https://www.gnu.org/software/emacs/manual/html_mono/efaq.html

GNU-Project. (2024b). *AUCTeX – Sophisticated document creation.* https://www.gnu.org/software/auctex/

GNU-Project. (2024c). *Manual Installation.* https://guix.gnu.org/manual/en/html_node/Manual-Installation.html

GNU-Project. (2024d). *The Emacs Widget Library*. https://www.gnu.org/software/emacs/manual/html_mono/widget.html

Gorrie, L. (2024). *SLIME: The Superior Lisp Interaction Mode for Emacs*. https://slime.common-lisp.dev/

Kuan, J. (2022). *Org-roam: A plain-text personal knowledge management system*. https://www.orgroam.com/

Linux_Kernel_Organization. (2024, April). *Frequently asked questions*. https://www.kernel.org/faq.html

Lissner, H. (2024). *DOOM Emacs*. https://github.com/doomemacs/doomemacs

Mumford, L. (1971). *Myth of the Machine: Technics and Human Development*. Mariner Books.

RHEL. (2024). *A Detailed Look at the Boot Process*. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/installation_guide/s1-boot-init-shutdown-process

Rougier, N. P. (2020). On the design of text editors. *Corr*. https://arxiv.org/abs/2008.06030

Rougier, N. P. (2024). *N Λ N O - Emacs*. https://github.com/rougier/nano-emacs

Ward, B. (2004). *How Linux Works: What Every Superuser Should Know*. No Starch Press.

Zawinski, J. (2007). *Emacs Timeline*. https://www.jwz.org/doc/emacs-timeline.html

I. Appendix

I.I. Code

All the code of the project, provided under the Appendix I.I, is licensed under the GPL-3 license defined under the Appendix I.I.I.

I.I.I. LICENSE (GPL-3)

Code: /LICENSE

```
  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
```

```
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
```

```
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.
    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
```

```
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
```

```
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
```

I.I.II. config.org

Code: /config.org

```
#    Kudu --- A fully functioning Gnu Emacs system
#    Copyright (C) 2023  Joar von Arndt
#
#   This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.

#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.

#   You should have received a copy of the GNU General Public License
#   along with this program.  If not, see <https://www.gnu.org/licenses/>.

#+title: Kudu
#+author: Joar von Arndt
#+STARTUP: overview
* What is Kudu?
[[https://github.com/JanJoar/Kudu-Emacs/blob/main/Logos/KuduLogo_red.svg]]

The complexity and extensibility of GNU Emacs, paired with its lack of integration with contemporary technical standards, has driven the development of Emacs distributions that contain packages and functionality not included by the GNU project. Kudu is a project meant to expand the scope of such distributions to every user-facing part of the operating system using dialects of the lisp programming language. This allows the user to easily and seamlessly "live in Emacs", using tools integrated directly into the program, such as the Emacs X Window Manager (EXWM), guix.el, and the Emacs Application Framework (EAF). Earlier distributions have focused on integrating Emacs within an otherwise alien system, like DOOM's and Spacemacs' focus on keybinds derived from the Vi editor, to maximize the number of workflows that the distribution could be incorporated into. Kudu does not take this approach, but rather empowers the user to construct their own system within a completely configurable system. All tools are written in lisp, the simple syntax of which allows for a seamless experience and self-sufficient system capable of performing all the daily tasks of modern life. It is hoped that this declarative and atomic system offered by GNU Guix will allow more secure and maintainable infrastructure.



The origin for the name is the kudu, an antelope similar to that of the Gnu, the namesake of the GNU Project. Kudu is not part of the GNU Project, and its developers are not members of GNU or the FSF. However we share a positive opinion of free software and therefore want to contribute to its mainstream adoption.
* Configuration

** Use-package

Probably one of the most useful packages, even if not very prominent when using emacs, is ~use-package~. It allows you to declaratively write your configuration and have the included emacs package manager download them for you, and also have configurations for packages only run when packages are loaded, similarly to ~(with-eval-after-load ...)~. The variables set here simply enable this behaviour. If the version of Emacs is older than Emacs 29, ~use-package~ won't be available by default. It is therefore installed here as well.

The ~diminish~ package hides certain minor modes from being shown in the mode-line and is not installed by default. For this reason its used to check if Kudu has been run before, and therefore if it needs to update its package repos. Feel free to perform this check on any other package, or remove it entirely, but beware that ~(package-refresh-contents)~ must be run before the other ~use-package~ declarations for ~package.el~ to install all the other packages needed.

#+BEGIN_SRC elisp
  (setq use-package-always-defer t
        use-package-always-ensure t
        use-package-verbose t)

  (unless (package-installed-p 'diminish)
      (package-refresh-contents)
      (package-install 'use-package)
      (package-install 'diminish))
#+END_SRC

** Auto-compile

```

```
Compiles elisp files to improve the speed and responsiveness of Emacs at the cost of first-time startup time. The settings in ~init.el~ makes sure that updated elisp files take priority over older, compiled files.

#+BEGIN_SRC elisp
    (use-package auto-compile
      :ensure t
      :init
      (auto-compile-on-load-mode 1)
      (auto-compile-on-save-mode 1))

  (setq native-comp-async-report-warnings-errors nil)
#+END_SRC

** Backups

Emacs usually stores backups in the same directory as the files themselves, cluttering up your nice and tidy system. This moves them to a dedicated directory within ~.emacs.d~.

#+BEGIN_SRC elisp
  (setq backup-directory-alist '(("." . "~/.emacs.d/backups")))
#+END_SRC

** EXWM
The Emacs X Window Manager allows you to use your entire desktop within emacs. Other windows are managed like traditional emacs buffers, and different workspaces are implemented using separate emacs frames. This is arguably the largest change to using traditional window managers and desktop environments, and it transforms emacs from simply a program that can do everything to /the/ way to interact with one's computer.

However, Emacs can still be used without constituting the entire system. Therefore EXWM should only be loaded if no other window manager is running. That way startup time isn't wasted whenever the user wants to run Emacs in the terminal, on a computer using a desktop environment, or another window manager.

#+BEGIN_SRC elisp
  (use-package exwm
    :init

    ;; EXWM related functions

    (defun xrandr-find-monitor-names ()
      "Returns a list of connected monitors"
      (let ((xrandr-contents nil) (monitor-names nil))
        (shell-command "xrandr" "*xrandr-output*")
        (switch-to-buffer "*xrandr-output*")
        (setq xrandr-contents (buffer-string))
        (kill-buffer "*xrandr-output*")
          (setq xrandr-contents (replace-regexp-in-string "\\(.* connected\\).*\n\\|.*\n" "\\1" xrandr-contents))
        (remove "" (split-string xrandr-contents " connected"))))

    (defun exwm-monitors-format ()
      "Formats the list from xrandr-find-monitor-names to apply EXWM workspaces"
      (let ((monitors (xrandr-find-monitor-names)) (counter 0) (return-value nil))
        (while monitors
          (push counter return-value)
          (push (car monitors) return-value)
          (setq counter (+ counter 1))
          (setq monitors (cdr monitors)))
        (nreverse return-value)))

    (setq switch-to-buffer-obey-display-actions t)
    (defvar exwm-is-running nil)
    (shell-command "wmctrl -m ; echo $status" "*window-manager*" "*window-manager-error*")

    (when (and
           (get-buffer "*window-manager-error*") ;; The shell command has to both encounter an error and a running in an X environment.
           (eq window-system 'x))
      (setq exwm-is-running t)

      (display-battery-mode 1)
      (setq display-time-day-and-date t)
      (display-time-mode 1)

      ;; Changes the name of EXWM-buffers to the corresponding window-name rather than *EXWM*<N>.
      (add-hook 'exwm-update-class-hook
                (lambda ()
                  (exwm-workspace-rename-buffer exwm-class-name)))

      ;; Configure monitors
      (require 'exwm-randr)
      (setq exwm-randr-workspace-monitor-plist (exwm-monitors-format))
      (setq exwm-workspace-number (length (xrandr-find-monitor-names)))
      (shell-command "bash ~/.screenlayout/desktop.sh")
      (setq exwm-workspace-number (/ (length (exwm-monitors-format)) 2))
      (exwm-randr-enable)

      ;; These  keys will always be sent to EXWM rather than to the X window.
      (setq exwm-input-prefix-keys
            '(?\C-x
              ?\C-g
              ?\M-x
              ?\M-z))

      ;; Sends the key after C-q directly to the X window.
      ;; (define-key exwm-mode-map [?\C-q] 'exwm-input-send-next-key)

      (setq exwm-input-global-keys
            `(
              ([?\s-r] . exwm-reset)
              ([s-left] . windmove-left)
              ([s-right] . windmove-right)
              ([s-up] . windmove-up)
              ([s-down] . windmove-down)
              ([?\s-w] . exwm-workspace-switch)
              ([?\C-q] . exwm-input-send-next-key)
              ([?\s-a] . (lambda (command)
                           (interactive (list (read-shell-command " λ ")))
                           (start-process-shell-command command nil command)))
              ([?\s-w] . exwm-workspace-switch)
              ([?\s-u] . (lambda ()
                           (interactive)
                           (shell-command "brightnessctl --quiet --min-value set +10")))
              ([?\s-d] . (lambda ()
                           (interactive)
                           (shell-command "brightnessctl --quiet --min-value set 10-")))
              ))
      ;; Actually starts EXWM
      (exwm-enable))

    (when (get-buffer "*window-manager*")
      (kill-buffer "*window-manager*"))
    (when (get-buffer "*window-manager-error*")
      (kill-buffer "*window-manager-error*")))
#+END_SRC

** General visual elements
Visible bell changes the otherwise quite jarring bell sound into a visual flash on it top and bottom of the
emacs frame. ~prettify-symbols-mode~ allows certain major modes to change the appearance of strings, the
classic example being the Greek letter lambda in lisp-modes for lambda calculus. ~pixel-scroll-precision-
mode~ allows you to scroll past things like images without buffers jumping around all the time.

#+BEGIN_SRC elisp
  (setq visible-bell t
        global-prettify-symbols-mode 1
        pixel-scroll-precision-mode t)
    (global-display-line-numbers-mode)
#+END_SRC

```

```org
Solaire-mode makes it easy to distinguish between warnings, popups and messages by tinting the background of those buffers slightly darker, as long as the current theme supports it.

#+BEGIN_SRC elisp
        (use-package solaire-mode
          :init
          (solaire-global-mode))
        (setq pixel-scroll-precision-mode t)
#+END_SRC

Formats tabs to Linux-kernel standards and keeps them so using the ~aggressive-indent~ package.

#+BEGIN_SRC elisp
  (setq-default tab-width 8)
  (setq-default standard-indent 8)
  (setq-default indent-tabs-mode nil)

  (use-package aggressive-indent
    :diminish aggressive-indent-mode
    :init (global-aggressive-indent-mode))
#+END_SRC

Enable mouse use when running Emacs in a terminal emulator.

#+BEGIN_SRC elisp
  (xterm-mouse-mode)
#+END_SRC

Without this setting emacs sometimes asks for confirmation via a "Yes or no" prompt, and sometimes "y or n". This is generally difficult to predict, and so this setting forces the message to always send "y or n" forms, like most programs run in a terminal.

#+BEGIN_SRC elisp
  (defalias 'yes-or-no-p 'y-or-n-p)
#+END_SRC

The default Emacs mode-line is a bit busy and certain elements of it are difficult to intuitively understand. This simplifies it considerably to make it more readable and also adds a header line.

#+BEGIN_SRC elisp
  (defun mode-line-padding ()
    (let ((r-length (length (format-mode-line mode-line-end-spaces))))
      (propertize " "
                  'display `(space :align-to (- right ,r-length)))))

  (setq-default mode-line-format
                '(
                  "|"
                  "%e"
                  (:eval (unless (string-match-p "\\*.*\\*" (buffer-name))
                      (let* ((read-only (and buffer-read-only (buffer-file-name)))
                             (modified (buffer-modified-p)))
                        (propertize
                         (if read-only "  " (if modified " !" "  "))))))
                  " "
                  (:eval (propertize (format "%s" (buffer-name)) 'face 'bold))
                  " "
                  (:eval (mode-line-padding))
                  (:eval (setq mode-line-end-spaces mode-line-misc-info))
                  ))
  (setq-default header-line-format
                  '(
                    "  "
                    (:eval (propertize (format "%s" mode-name) 'face 'bold))
                    " "
                    ))
#+END_SRC

Adds as nicely formated clock in all cases, even when not running in EXWM.
```

```
#+BEGIN_SRC elisp
  (setq display-time-default-load-average nil)
  (setq display-time-24hr-format t)
  (display-time-mode 1)
#+END_SRC

When editing just one window, left-aligned text is awkwardly too far to the left. The ~perfect-margin~
package fixes this by centering the contents of the window when only one is present.

#+BEGIN_SRC elisp
  (use-package perfect-margin
    :custom
    (perfect-margin-visible-width 128)
    :init
    ;; enable perfect-mode
    (unless exwm-is-running (perfect-margin-mode t))
    ;; auto-center minibuffer windows
    (setq perfect-margin-ignore-filters nil)
    ;; auto-center special windows
    (setq perfect-margin-ignore-regexps nil))
#+END_SRC

~rainbow-delimiters~ differentiates layers of parentheses using different colours so that they can be
identified at a glance.

#+BEGIN_SRC elisp
  (use-package rainbow-delimiters
    :init (add-hook 'prog-mode-hook #'rainbow-delimiters-mode))
#+END_SRC

~smartparens~ is intended to help in a similar way by highlighting the current sexp.

#+BEGIN_SRC elisp
  (use-package smartparens
    :hook
    (prog-mode . smartparens-mode)
    (text-mode . smartparens-mode)
    :init
    (require 'smartparens-config))
#+END_SRC

Adds little icons for completion frameworks.

#+BEGIN_SRC elisp
  (use-package svg-lib)
  (use-package kind-icon
    :after corfu
    :custom (kind-icon-default-face 'corfu-default)
    :init (add-to-list 'corfu-margin-formatters #'kind-icon-margin-formatter)
    (unless (display-graphic-p)
      (setq kind-icon-use-icons nil)))
#+END_SRC

Emacs is a wonderful alternative to a terminal, encompassing [[https://www.masteringemacs.org/article/
running-shells-in-emacs-overview][many of the features]] seen in modern terminals. For a cleaner look,
this hides the mode-line in windows used to interact with shells.

#+BEGIN_SRC elisp
  (use-package hide-mode-line
    :hook
    (eat-mode . hide-mode-line-mode)
    (term-mode . hide-mode-line-mode)
    (eshell-mode . hide-mode-line-mode))
#+END_SRC

** Dashboard
```

```
Configures the all-important emacs dashboard that shows up on startup.

#+BEGIN_SRC elisp
  (use-package dashboard
    :init
    (dashboard-setup-startup-hook)
    (setq dashboard-icon-type 'all-the-icons)
    (setq dashboard-banner-logo-title "Welcome to Kudu Emacs!")
    (setq dashboard-center-content 'middle)
    (setq dashboard-startup-banner
          (if (window-system)
              Kudu-gui-logo
            "~/.emacs.d/Logos/KuduLogo_text.txt"))
    (setq compilation-ask-about-save nil)
    (setq dashboard-show-shortcuts nil)
    (setq dashboard-items '((recents . 5)))
    (setq dashboard-set-navigator nil)
    (setq dashboard-set-init-info t)
    (setq dashboard-set-footer nil)
    (dashboard-setup-startup-hook)

    (add-hook  'dashboard-mode-hook (lambda () (display-line-numbers-mode -1))))
#+END_SRC

** Completion

*** Corfu

In-buffer code completion using ~corfu~.
By default ~corfu~ only works in a GUI environment, but the ~corfu-terminal~ package allows for use when run using the ~-nw~ flag.

#+BEGIN_SRC elisp
  (use-package corfu
    :custom
    (setq corfu-auto t)
    :init
    (global-corfu-mode)
    (setq corfu-popupinfo-delay 0.5)
    (corfu-popupinfo-mode +1))

  (use-package corfu-terminal
      :init
      (unless (display-graphic-p)
        (corfu-terminal-mode +1)))
  #+END_SRC

*** Cape

  ~corfu~ does not provide candidates for completion, but this is provided by ~cape~, or the Completion At Point Extensions package.

  #+BEGIN_SRC elisp
      (use-package cape
        ;; Bind dedicated completion commands
        ;; Alternative prefix keys: C-c p, M-p, M-+, ...
        :bind (("C-c p p" . completion-at-point) ;; capf
               ("C-c p t" . complete-tag)        ;; etags
               ("C-c p d" . cape-dabbrev)        ;; or dabbrev-completion
               ("C-c p h" . cape-history)
               ("C-c p f" . cape-file)
               ("C-c p k" . cape-keyword)
               ("C-c p s" . cape-symbol)
               ("C-c p a" . cape-abbrev)
               ("C-c p l" . cape-line)
               ("C-c p w" . cape-dict)
               ("C-c p \\" . cape-tex)
               ("C-c p _" . cape-tex)
               ("C-c p ^" . cape-tex)
               ("C-c p &" . cape-sgml)
               ("C-c p r" . cape-rfc1345))
        :init
        (add-to-list 'completion-at-point-functions #'cape-dabbrev)
        (add-to-list 'completion-at-point-functions #'cape-file)
        (add-to-list 'completion-at-point-functions #'cape-elisp-block)
        (add-to-list 'completion-at-point-functions #'cape-history)
        (add-to-list 'completion-at-point-functions #'cape-keyword))
 #+END_SRC

*** Minibuffer Completion

 Uses ~vertico~ to show minibuffer completion, and ~marginalia~ and ~orderless~ to format it.

 #+BEGIN_SRC elisp
     (use-package vertico
       :init
       (vertico-mode)
       :config
       (setq vertico-count 10)
       (vertico-indexed-mode)
       (vertico-mouse-mode))

     (use-package marginalia
       :hook (vertico-mode . marginalia-mode))

     (use-package orderless
     :custom
     (completion-styles '(orderless basic prescient))
     (completion-category-overrides '((file (styles basic partial-completion)))))
#+END_SRC

*** Prescient

Shows those completion results that are hopefully most useful, both in the minibuffer and the main buffer.

#+BEGIN_SRC elisp
  (use-package prescient
    :init
    (setq prescient-persist-mode t)
    (setq prescient-history-length 5)
    (setq prescient-sort-full-matches-first t))
  (use-package corfu-prescient
    :init (corfu-prescient-mode +1))
  (use-package vertico-prescient
    :init (vertico-prescient-mode +1))
#+END_SRC

*** Consult

~consult~ provides various functions that integrates with the completion API.

#+BEGIN_SRC elisp
  (use-package consult
    :bind (;; C-c bindings in `mode-specific-map'
           ("C-c M-x" . consult-mode-command)
           ("C-c h" . consult-history)
           ("C-c k" . consult-kmacro)
           ("C-c m" . consult-man)
           ("C-c i" . consult-info)
           ([remap Info-search] . consult-info)
           ;; C-x bindings in `ctl-x-map'
           ("C-x M-:" . consult-complex-command)     ;; orig. repeat-complex-command
           ("C-x b" . consult-buffer)                ;; orig. switch-to-buffer
           ("C-x 4 b" . consult-buffer-other-window) ;; orig. switch-to-buffer-other-window
           ("C-x 5 b" . consult-buffer-other-frame)  ;; orig. switch-to-buffer-other-frame
           ("C-x t b" . consult-buffer-other-tab)    ;; orig. switch-to-buffer-other-tab
           ("C-x r b" . consult-bookmark)            ;; orig. bookmark-jump
           ("C-x p b" . consult-project-buffer)      ;; orig. project-switch-to-buffer
           ;; Custom M-# bindings for fast register access
           ("M-#" . consult-register-load)
           ("M-'" . consult-register-store)          ;; orig. abbrev-prefix-mark (unrelated)
           ("C-M-#" . consult-register)
           ;; Other custom bindings
           ("M-y" . consult-yank-pop)                ;; orig. yank-pop
           ;; M-g bindings in `goto-map'
           ("M-g e" . consult-compile-error)
           ("M-g f" . consult-flymake)               ;; Alternative: consult-flycheck
           ("M-g g" . consult-goto-line)             ;; orig. goto-line
           ("M-g M-g" . consult-goto-line)           ;; orig. goto-line
           ("M-g o" . consult-outline)               ;; Alternative: consult-org-heading
           ("M-g m" . consult-mark)
           ("M-g k" . consult-global-mark)
           ("M-g i" . consult-imenu)
           ("M-g I" . consult-imenu-multi)
           ;; M-s bindings in `search-map'
           ("M-s d" . consult-find)                  ;; Alternative: consult-fd
           ("M-s c" . consult-locate)
           ("M-s g" . consult-grep)
           ("M-s G" . consult-git-grep)
           ("M-s r" . consult-ripgrep)
           ("M-s l" . consult-line)
           ("M-s L" . consult-line-multi)
           ("M-s k" . consult-keep-lines)
           ("M-s u" . consult-focus-lines)
           ;; Isearch integration
           ("M-s e" . consult-isearch-history)
           :map isearch-mode-map
           ("M-e" . consult-isearch-history)         ;; orig. isearch-edit-string
           ("M-s e" . consult-isearch-history)       ;; orig. isearch-edit-string
           ("M-s l" . consult-line)                  ;; needed by consult-line to detect isearch
           ("M-s L" . consult-line-multi)            ;; needed by consult-line to detect isearch
           ;; Minibuffer history
           :map minibuffer-local-map
           ("M-s" . consult-history)                 ;; orig. next-matching-history-element
           ("M-r" . consult-history))                ;; orig. previous-matching-history-element

    :init
    ;; Optionally tweak the register preview window.
    ;; This adds thin lines, sorting and hides the mode line of the window.
    (advice-add #'register-preview :override #'consult-register-window)

    ;; Use Consult to select xref locations with preview
    (setq xref-show-xrefs-function #'consult-xref
          xref-show-definitions-function #'consult-xref)

    :config
    ;; Optionally configure preview. The default value
    ;; is 'any, such that any key triggers the preview.
    ;; (setq consult-preview-key 'any)
    ;; (setq consult-preview-key "M-.")
    ;; (setq consult-preview-key '("S-<down>" "S-<up>"))
    ;; For some commands and buffer sources it is useful to configure the
    ;; :preview-key on a per-command basis using the `consult-customize' macro.
    (consult-customize
     consult-theme :preview-key '(:debounce 0.2 any)
     consult-ripgrep consult-git-grep consult-grep
     consult-bookmark consult-recent-file consult-xref
     consult--source-bookmark consult--source-file-register
     consult--source-recent-file consult--source-project-recent-file
     ;; :preview-key "M-."
     :preview-key '(:debounce 0.4 any)))
#+END_SRC

*** Flycheck

Tangentially related is flycheck, providing in-buffer syntax checking.

#+BEGIN_SRC elisp
  (use-package flycheck
    :config (global-flycheck-mode +1))
#+END_SRC
```

```
** Org-mode

Configures Org-mode to make it more attractive and usable.

#+BEGIN_SRC elisp
  (setq completion-cycle-threshold 2)
  (setq tab-always-indent 'complete)

  (use-package org
    :config
    (setq org-format-latex-options
          (plist-put org-format-latex-options
                     :scale 1.3
                     ))
    (setq org-format-latex-options
          (plist-put org-format-latex-options
                     :html-scale 3
                     ))
    (setq org-startup-indented t
          org-toggle-pretty-entities t
          org-hide-leading-stars t
          org-hide-emphasis-markers t)
    (add-hook 'text-mode-hook 'turn-on-visual-line-mode))

  (use-package org-superstar
    :hook (org-mode . org-superstar-mode))
  (use-package org-fragtog
    :hook (org-mode . org-fragtog-mode))
  (use-package toc-org
    :hook (org-mode . toc-org-mode))
  (use-package org-appear
    :hook (org-mode . org-appear-mode))
  (use-package yasnippet
    :diminish yas-minor-mode
    :hook (org-mode . yas-minor-mode)
    :config
    (yas-load-directory "~/.emacs.d/snippets/"))
  (use-package yasnippet-snippets)

  (use-package valign
    :hook (org-mode . valign-mode))

  (use-package org-modern
    :hook
    (org-mode . org-modern-mode)
    (org-agenda-finalize . org-modern-agenda)
    :custom
    (org-modern-table-horizontal 2)
    (org-modern-table-vertical 1)
    (org-modern-star nil)
    (org-modern-hide-stars nil)
    (org-modern-checkbox nil))

  (unless (file-directory-p "~/.emacs.d/site-lisp/org-modern-indent")
       (async-shell-command "git clone https://github.com/jdtsmith/org-modern-indent.git://github.com/jdtsmith/org-modern-indent.git ~/.emacs.d/site-lisp/org-modern-indent/"))
  (use-package org-modern-indent
    :load-path "~/.emacs.d/site-lisp/org-modern-indent"
    :hook (org-mode . org-modern-indent-mode))
 #+END_SRC

** Lisp

Emacs is an amazing environment for writing in various lisp dialects, with wonderful support out-of-the-box. However, there are various different packages designed to improve this experience in general or in slight, specific ways. ~lispy~ is a transformational package for editing S-expressions in a structural way. ~Sly~ is a fork of the popular ~SLIME~ package for an integrated common lisp REPL among other things. It is superior to ~SLIME~ because it has ASCII-art cats.

#+BEGIN_SRC elisp
```

```
  (use-package paredit
    :hook
    (lisp-mode . paredit-mode)
    (emacs-lisp-mode . paredit-mode)
    (scheme-mode . paredit-mode)
    (slime-mode . paredit-mode))

  (use-package sly
    :config
    (setq inferior-lisp-program "sbcl"))

  (setq show-paren-delay 0)
  (show-paren-mode)
#+END_SRC

*** Scheme

Due to Kudu's deep integration with the GNU Guix system, it is only natural to improve the systems used to interact with guile and scheme specifically. For this the ~guix.el~ and the wonderful ~geiser~ packages are used, where ~guix.el~ is a magit-inspired Emacs frontend and ~geiser~ is a package aiming to improve the scheme experience in emacs, with ~geiser-guile~ providing special support for working the /GNU Ubiquitous Intelligent Language for Extensions/.

#+BEGIN_SRC elisp
  (use-package guix)

  (use-package geiser-guile)
#+END_SRC

*** Parens pairing

Most of the time when writing parentheses, brackets, and quotes we want to pair them. This significantly improves comfort since you no longer need to stretch for modifier keys to finish of the pair. And even if you do, ~electric-pair-mode~ will detect it and move the point past as if you had just entered the character. This is of course not just useful for lisp, but in any context when writing pairs of brackets or parentheses.

#+BEGIN_SRC elisp
  (setq electric-pair-pairs '(
                              (?\{ . ?\})
                              (?\( . ?\))
                              (?\[ . ?\])
                              (?\" . ?\")))
  (electric-pair-mode t)
#+END_SRC

** File management

Dired is emacs' built in text-based file manager. It's however pretty rough around its edges, such as it opening each directory in a separate buffer making navigation a hassle. However certain tweaks can make it a formidable tool accessible directly within emacs. Take that n³ and midnight commander!

#+BEGIN_SRC elisp
    (use-package openwith
      :hook (dired-mode . openwith-mode)
      :config
      (setq openwith-associations (list
                                   (list (openwith-make-extension-regexp
                                          '("png" "jpg" "jpeg")) "eog" '(file))
                                   (list (openwith-make-extension-regexp
                                          '("mkv" "mp4" "avi")) "mpv" '(file)))))

  (setf dired-kill-when-opening-new-dired-buffer t)
  (setq dired-listing-switches "-aBhl  --group-directories-first")
  (defalias 'eaf-open-in-file-manager #'dired)

  (add-hook 'dired-mode-hook 'toggle-truncate-lines)
```

```org
#+END_SRC

** PDF-tools

The default "docview" mode of viewing pdfs is quite bad, and is improved immensely by the pdf-tools package. For some this may not be enough, and it is possible to replace it with an external pdf viewer (like evince or zathura) using the above ~openwith~ package.

#+BEGIN_SRC elisp
  (use-package pdf-tools
    :init
    (pdf-loader-install)
    (add-hook  'pdf-view-mode-hook (lambda () (display-line-numbers-mode -1))))
  #+END_SRC

** Magit

Magit is wonderful, and one of the killer apps that makes emacs a system than other editors or IDEs. However it is not installed by default, so it is defined here.

#+BEGIN_SRC elisp
  (use-package magit)

  (use-package magit-todos
    :after magit
    :config (magit-todos-mode 1))

  (use-package magit-delta
    :after magit
    :config (magit-delta-mode t))
#+END_SRC

** Tooling

Emacs has a wonderful undo-system, but it can be hard to get an idea of how it works intuitively. ~undo-tree~ helps with this by creating a wonderful visualization for your branching undo, well, tree.

#+BEGIN_SRC elisp
  (use-package undo-tree
    :init
    (setq undo-tree-visualizer-timestamps t)
    (setq undo-tree-auto-save-history t)
    (unless (file-exists-p "~/.emacs.d/undo-tree")
      (make-directory "~/.emacs.d/undo-tree"))
    (setq undo-tree-history-directory-alist '(("." . "~/.emacs.d/undo-tree")))
    (global-undo-tree-mode +1))

#+END_SRC

** Functions

The sudo function raises the privilege of the current buffer to root permissions without having to close and open it again through ~TRAMP~.

#+BEGIN_SRC elisp
  (defun sudo ()
    "Opens the current buffer at point with root privelages using TRAMP"
    (interactive)
    (let ((position (point)))
      (find-alternate-file (concat "/sudo::"
                                   (buffer-file-name (current-buffer))))
      (goto-char position)))
#+END_SRC

Magit can sometimes create a lot of buffers for different processes that are annoying to close one by one, this function closes all buffers whose name contains "magit".

```

```
#+BEGIN_SRC elisp
  (defun kill-magit-buffers ()
    (interactive)
    (mapc (lambda (buffer)
            (if (buffer-match-p ".*magit.*" buffer)
              (kill-buffer buffer)))
          (buffer-list)))
#+END_SRC

Emacs does not have a nice easy to use elisp  function for calculating the factorial of a value, this adds it. This works out particularly nicely since the standard notation for the factorial of a value uses prefix notation.

#+BEGIN_SRC elisp
  (defun ! (n)
    "An emacs function to calculate the factorial of n using the calc library"
    (let ((output (string-to-number (calc-eval (format "%s!" n)))))
      (kill-buffer "*Calculator*")
      output))
#+END_SRC

Function for calculation the number of possible permutations and combinations respectively.

#+BEGIN_SRC elisp
  (defun nPr (n k)
    "A function for calculating the number of permutations in combinatorics"
    (/
     (! n)
     (! (- n k))))

  (defun nCr (n k)
    "A function for calculating the number of combinations in combinatorics"
    (/
     (! n)
     (* (! k) (! (- n k)))))
#+END_SRC

** Emacs Application Framework

The /Emacs Application Framework/ (EAF) provides a multitude of programs, most notably a browser, that more tigtly integrate with the Emacs than Icecat or other browsers allow for when used in conjunction with EXWM. While they mostly are usable with a REPL-style lisp interaction, they are nevertheless incredibly useful.

#+BEGIN_SRC elisp
    (unless (file-directory-p "~/.emacs.d/site-lisp/emacs-application-framework/")
         (shell-command "git clone --depth=1 -b master https://github.com/emacs-eaf/emacs-application-framework.git ~/.emacs.d/site-lisp/emacs-application-framework/"))
  (if (get-buffer "*Shell Command Output*") (kill-buffer "*Shell Command Output*"))

  (add-to-list 'load-path "~/.emacs.d/site-lisp/emacs-application-framework/")

  (use-package eaf
    :load-path "~/.emacs.d/site-lisp/emacs-application-framework"
    :config
    (if (display-graphic-p)
        (require 'eaf-browser)
      (require 'eaf-map)
      (defalias 'browse-web #'eaf-open-browser)
      (setq eaf-browser-default-search-engine "duckduckgo")
      (setq eaf-browse-blank-page-url "https://duckduckgo.com")
      (eaf-bind-key nil "M-q" eaf-browser-keybinding)
      (setq eaf-byte-compile-apps t)))
#+END_SRC
```

I.I.III. early.init

Code: /early-init.el

```
;    Kudu --- A fully functioning GNU Emacs system
;    Copyright (C) 2023  Joar von Arndt
;
;    This program is free software: you can redistribute it and/or modify
;    it under the terms of the GNU General Public License as published by
;    the Free Software Foundation, either version 3 of the License, or
;    (at your option) any later version.
;
;    This program is distributed in the hope that it will be useful,
;    but WITHOUT ANY WARRANTY; without even the implied warranty of
;    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;    GNU General Public License for more details.
;
;    You should have received a copy of the GNU General Public License
;    along with this program.  If not, see <https://www.gnu.org/licenses/>.

;;; Commentary: Early optimizations mostly for improved startup times. A not-insignificant parts of are taken from https://github.com/Stefanomarton/DotFiles/ and his wonderful improvements.

(defvar me/gc-cons-threshold 100000000)
(setq gc-cons-threshold most-positive-fixnum
      gc-cons-percentage 0.6)
(add-hook 'emacs-startup-hook
          (lambda ()
            (setq gc-cons-threshold me/gc-cons-threshold
                  gc-cons-percentage 0.1)))

(defun me/defer-garbage-collection-h ()
  (setq gc-cons-threshold most-positive-fixnum))

(defun me/restore-garbage-collection-h ()
  (run-at-time
   1 nil (lambda () (setq gc-cons-threshold me/gc-cons-threshold))))

(add-hook 'minibuffer-setup-hook #'me/defer-garbage-collection-h)
(add-hook 'minibuffer-exit-hook #'me/restore-garbage-collection-h)

;; Disabling these things here prevents them from ever loading.
(scroll-bar-mode -1)
(tool-bar-mode -1)
(tab-bar-mode -1)
(menu-bar-mode -1)
(setq inhibit-startup-screen t)

;;; early-init.el ends here
```

I.I.IV. init.el

Code: /init.el

```
                                        ;    Kudu --- A fully functioning GNU Emacs system
                                        ;    Copyright (C) 2023  Joar von Arndt
                                        ;
                                        ;
                                         ;    This program is free software: you can redistribute it and/or modify
                                            ;    it under the terms of the GNU General Public License as published by
                                             ;    the Free Software Foundation, either version 3 of the License, or
                                        ;    (at your option) any later version.
                                        ;
                                      ;    This program is distributed in the hope that it will be useful,
                                      ;    but WITHOUT ANY WARRANTY; without even the implied warranty of
                                       ;    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
                                        ;    GNU General Public License for more details.
                                        ;
                                    ;    You should have received a copy of the GNU General Public License
```

```
                                         ;    along with this program.  If not, see <https://www.gnu.org/licenses/>.

;;; Commentary: This file simply serves to load other Emacs lisp files in order to neatly separate different concepts


(setq load-prefer-newer t) ;; Loads the newer file if one exists. This means emacs will prioritise files with newer changes.

(defvar Kudu-gui-logo "~/.emacs.d/Logos/KuduLogo_red.svg")
(shell-command "touch ~/.emacs.d/secret.org && touch ~/.emacs.d/secret.el && touch ~/.emacs.d/custom.el")
(kill-buffer "*Shell Command Output*")

(require 'package)

(unless (assoc-default "melpa" package-archives)
  (add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/") t))
(unless (assoc-default "nongnu" package-archives)
  (add-to-list 'package-archives '("nongnu" . "https://elpa.nongnu.org/nongnu/") t))

(package-initialize)

(org-babel-load-file (expand-file-name "~/.emacs.d/secret.org")) ;; User-unique information (like E-mail address and full name) that you might not want to share openly. Empty by default. Since the file is not included in the Kudu repo it has to be created using touch in order to be loaded.
(org-babel-load-file (expand-file-name "~/.emacs.d/config.org")) ;; The main configuration file, running commands, setting keybinds, and configuring packages.

(setq custom-file "~/.emacs.d/custom.el")
(load custom-file)

;;; init.el ends here
```

I.I.V. install

Code: /install

```
#!/bin/sh

d=$(dirname $(readlink -f "$0"))/installer
emacs -nw -q -l $d/installer.el --eval "(Kudu-installer)" --chdir $d
```

I.I.V.I. Installer

I.I.V.I.I. bare-bones.scm

Code: /installer/bare-bones.scm

```
;; This is an operating system configuration template
;; for a "bare bones" setup, with no X11 display server.

(use-modules (gnu))
(use-service-modules networking ssh)
(use-package-modules screen ssh)

(operating-system
  (host-name "komputilo")
  (timezone "Europe/Berlin")
  (locale "en_US.utf8")

  ;; Boot in "legacy" BIOS mode, assuming /dev/sdX is the
  ;; target hard disk, and "my-root" is the label of the target
  ;; root file system.
  (bootloader (bootloader-configuration
                (bootloader grub-bootloader)
                (targets '("/dev/sdX"))))
  ;; It's fitting to support the equally bare bones ‘-nographic’
  ;; QEMU option, which also nicely sidesteps forcing QWERTY.
  (kernel-arguments (list "console=ttyS0,115200"))
  (file-systems (cons (file-system
                        (device (file-system-label "my-root"))
                        (mount-point "/")
                        (type "ext4"))
                      %base-file-systems))

  ;; This is where user accounts are specified.  The "root"
  ;; account is implicit, and is initially created with the
  ;; empty password.
  (users (cons (user-account
                (name "alice")
                (comment "Bob's sister")
                (group "users")

                ;; Adding the account to the "wheel" group
                ;; makes it a sudoer.  Adding it to "audio"
                ;; and "video" allows the user to play sound
                ;; and access the webcam.
                (supplementary-groups '("wheel"
                                        "audio" "video")))
               %base-user-accounts))

  ;; Globally-installed packages.
  (packages (cons screen %base-packages))

  ;; Add services to the baseline: a DHCP client and
  ;; an SSH server.
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type
                                   (openssh-configuration
                                    (openssh openssh-sans-x)
                                    (port-number 2222))))
                    %base-services)))
```

I.I.V.I.II. config.scm

Code: /installer/config.scm

```

```

I.I.V.I.III. get_disks.sh

Code: /installer/get_disks.sh

```
#!/bin/sh
disks=$(lsblk -o NAME,TYPE -n -p -l | grep 'disk' | awk '{print $1}')
# Print each disk on a new line
echo "$disks"
```

I.I.V.I.IV. get_disks_test.sh

Code: /installer/get_disks_test.sh

```
#!/bin/sh
echo sda
echo sdb
echo sdc
echo hej
echo san
```

I.I.V.I.V. get_keymaps_test.sh

Code: /installer/get_keymaps_test.sh

```sh
#!/bin/sh
cd ./run/current-system/profile/share/keymaps
keys=$(find ./ -type f)
echo "$keys" | grep -Po '/\K([^/]+)\.map\.gz$' | sed 's/\.map\.gz$//' | sort
```

I.I.V.I.VI. guix_config.scm

Code: /installer/guix_config.scm

```scheme


;; This is an operating system configuration generated
;; by the graphical installer.
;;
;; Once installation is complete, you can learn and modify
;; this file to tweak the system configuration, and pass it
;; to the 'guix system reconfigure' command to effect your
;; changes.


;; Indicate which modules to import to access the variables
;; used in this configuration.
(use-modules
 (gnu)
 (gnu packages emacs)
 (gnu packages emacs-xyz)
 (gnu packages screen)
 (gnu packages linux)
 (gnu packages version-control)
 (gnu packages gnuzilla)
 (gnu packages games)
 (gnu packages xdisorg)
 )
(use-service-modules cups desktop networking ssh xorg)

(operating-system
 (locale "en_US.utf8")
 (timezone "$TIMEZONE")
 (keyboard-layout (keyboard-layout "$KEYMAP"))
 (host-name "$HOSTNAME")

 (users (cons* (user-account
                (name "$USERNAME")
                (group "users")
                (home-directory "/home/$USERNAME")
                (supplementary-groups '("wheel" "netdev" "audio" "video")))
               %base-user-accounts))

 ;; Packages installed system-wide.  Users can also install packages
 ;; under their own account: use 'guix search KEYWORD' to search
 ;; for packages and 'guix install PACKAGE' to install a package.
 (packages (append (list
        (specification->package "nss-certs")
        screen
        emacs
        emacs-exwm
        wmctrl
        brightnessctl
        git
        icecat
        openttd
        )
                   %base-packages))

 ;; Below is the list of system services.  To search for available
 ;; services, run 'guix system search KEYWORD' in a terminal.
 (services
  (append (list (service gnome-desktop-service-type)

                ;; To configure OpenSSH, pass an 'openssh-configuration'
                ;; record as a second argument to 'service' below.
                (service openssh-service-type)
                (set-xorg-configuration
                 (xorg-configuration (keyboard-layout keyboard-layout))))

          ;; This is the default list of services we
          ;; are appending to.
          %desktop-services))
 (bootloader (bootloader-configuration
              (bootloader grub-bootloader)
              (targets '("$DISK")))
        (theme
    (grub-theme
      (resolution '(1920 . 1080))
      (image (local-file "/mnt/etc/Kudu_grub_image.svg")))))

 (swap-devices (list (swap-space
                      (target (uuid
                               "$SWAP_UUID"
             )))))

 ;; The list of file systems that get "mounted".  The unique
 ;; file system identifiers there ("UUIDs") can be obtained
 ;; by running 'blkid' in a terminal.
 (file-systems (cons* (file-system
                       (mount-point "/")
                       (device (uuid
                                "$ROOT_UUID"
        'ext4))
                       (type "ext4")) %base-file-systems)))
```

I.I.V.I.VII. guix_iso.scm

Code: /installer/guix_iso.scm

```
(use-modules
  (gnu)
  (gnu packages emacs)
  (gnu packages version-control)
  )
(use-service-modules networking)
(operating-system
  (host-name "kudu-inst")
  (timezone "Europe/Vatican")
  (locale "en_US.utf8")

   (bootloader (bootloader-configuration
    (bootloader grub-bootloader)
    (targets '("$DISK")))
    (theme
      (grub-theme
        (resolution '(1920 . 1080))
        (image (local-file "/mnt/etc/Kudu_grub_image.svg")))))

  (kernel-arguments (list "console=ttyS0,115200"))
  (file-systems (cons*
    (file-system
      (mount-point "/")
      (device (uuid
        "$ROOT_UUID"
        'ext4))
      (type "ext4"))
    %base-file-systems))


  (users %base-user-accounts)
  (packages (append (list git emacs) %base-packages))
  (services
    (append
    (list (service dhcp-client-service-type))
    %base-services))
)
```

I.I.V.I.VIII. installer.el

Code: /installer/installer.el

```
(require 'widget)

(defun Kudu-installer ()
  (interactive)
  (switch-to-buffer "*Kudu-Installer*")
  (setup-greeting))

(defun setup-greeting ()
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (widget-insert (read-file-into-string "../Logos/KuduLogo_text.txt") )
  (widget-insert "\n\n")
  (widget-create 'push-button
                 :notify (lambda (&rest ignore)
                           (setup-keymap))
                 "Setup installation")
  (center-line)
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun setup-keymap ()
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((keymap ""))
    (setq keymaps (get-nl-seperated "../installer/keymaps"))
    (widget-insert "Keymap: \n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq keymap
                     (widget-value widget)))
     (mapcar (lambda (keymap) `(item ,keymap)) keymaps))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (message (concat "loadkeys " keymap))
                             (shell-command-to-string (concat "loadkeys " keymap))
                             (setup-timezone keymap))
                   "Apply Form"))
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun setup-timezone (keymap)
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((timezone ""))
    (setq timezones(get-nl-seperated "../installer/timezones"))
    (widget-insert "Timezone: \n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq timezone
                     (widget-value widget)))
     (mapcar (lambda (x) `(item ,x)) timezones))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (Installation-options timezone keymap))
                   "Apply Form"))
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun Installation-options (timezone keymap)
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((hostname "") (username "") (disk ""))
    (widget-insert "\n")
    (setq disks (get-shell "../installer/get_disks.sh"))
    (message (car disks))
    (widget-create 'editable-field
                   :size 30
                   :format "Toastname: %v "
                   :notify (lambda (widget &rest ignore)
                             (setq hostname (widget-value widget))))
    (widget-insert "\n\n Disk to use:\n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq disk
                     (widget-value widget)))
     (mapcar (lambda (disk) `(item ,disk)) disks))

    (widget-insert "\n \n")
    (widget-create 'editable-field
                   :size 30
                   :format "Username: %v "
                   :notify (lambda (widget &rest ignore)
                             (setq username (widget-value widget))))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (upload
                              hostname
                              username
                              disk
                              timezone
                              keymap))
                   "Apply Form")
    (use-local-map widget-keymap)
    (widget-setup)
    (beginning-of-buffer)
    (widget-forward 1)))

(defun upload (hostname username disk timezone keymap)
  (message "Formatting bash command...")
  (setq cmd (format
              "bash ../installer/install.sh --hostname %s --username %s --disk %s --timezone %s --keymap
%s &"
             hostname
             username
             disk
             timezone
             keymap))
  (message cmd)
  (shell-command cmd))

(defun get-shell (x)
  "Get a list of from shell script."
  (interactive)
  (when (eq system-type 'gnu/linux)
    (split-string
     (shell-command-to-string (concat "sh " x))
     "\n" t)))

(defun get-nl-seperated (x)
  (with-temp-buffer
    (insert-file-contents x)
    (let ((lines (split-string (buffer-string) "\n" t)))
      (sort lines #'string<))))

(defun read-file-into-string (file-path)
  "Read the content of FILE-PATH into a string."
  (with-temp-buffer
    (insert-file-contents file-path)
    (buffer-string)))

```

I.I.V.I.IX. installer.el

Code: /installer/installer.el

```
(require 'widget)

(defun Kudu-installer ()
  (interactive)
  (switch-to-buffer "*Kudu-Installer*")
  (setup-greeting))

(defun setup-greeting ()
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (widget-insert (read-file-into-string "../Logos/KuduLogo_text.txt") )
  (widget-insert "\n\n")
  (widget-create 'push-button
                 :notify (lambda (&rest ignore)
                           (setup-keymap))
                 "Setup installation")
  (center-line)
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun setup-keymap ()
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((keymap ""))
    (setq keymaps (get-nl-seperated "../installer/keymaps"))
    (widget-insert "Keymap: \n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq keymap
                     (widget-value widget)))
     (mapcar (lambda (keymap) `(item ,keymap)) keymaps))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (message (concat "loadkeys " keymap))
                             (shell-command-to-string (concat "loadkeys " keymap))
                             (setup-timezone keymap))
                   "Apply Form"))
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun setup-timezone (keymap)
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((timezone ""))
    (setq timezones(get-nl-seperated "../installer/timezones"))
    (widget-insert "Timezone: \n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq timezone
                     (widget-value widget)))
     (mapcar (lambda (x) `(item ,x)) timezones))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (Installation-options timezone keymap))
                   "Apply Form"))
  (use-local-map widget-keymap)
  (widget-setup)
  (beginning-of-buffer)
  (widget-forward 1))

(defun Installation-options (timezone keymap)
  (interactive)
  (kill-all-local-variables)
  (let ((inhibit-read-only t))
    (erase-buffer))
  (remove-overlays)
  (let ((hostname "") (username "") (disk ""))
    (widget-insert "\n")
    (setq disks (get-shell "../installer/get_disks.sh"))
    (message (car disks))
    (widget-create 'editable-field
                   :size 30
                   :format "Toastname: %v "
                   :notify (lambda (widget &rest ignore)
                             (setq hostname (widget-value widget))))
    (widget-insert "\n\n Disk to use:\n")
    (apply
     #'widget-create
     'radio-button-choice
     :tag "radio-tag"
     :notify (lambda (widget &rest ignore)
               (setq disk
                     (widget-value widget)))
     (mapcar (lambda (disk) `(item ,disk)) disks))

    (widget-insert "\n \n")
    (widget-create 'editable-field
                   :size 30
                   :format "Username: %v "
                   :notify (lambda (widget &rest ignore)
                             (setq username (widget-value widget))))
    (widget-insert "\n")
    (widget-create 'push-button
                   :notify (lambda (&rest ignore)
                             (upload
                              hostname
                              username
                              disk
                              timezone
                              keymap))
                   "Apply Form")
    (use-local-map widget-keymap)
    (widget-setup)
    (beginning-of-buffer)
    (widget-forward 1)))

(defun upload (hostname username disk timezone keymap)
  (message "Formatting bash command...")
  (setq cmd (format
              "bash ../installer/install.sh --hostname %s --username %s --disk %s --timezone %s --keymap
%s &"
             hostname
             username
             disk
             timezone
             keymap))
  (message cmd)
  (shell-command cmd))

(defun get-shell (x)
  "Get a list of from shell script."
  (interactive)
  (when (eq system-type 'gnu/linux)
    (split-string
     (shell-command-to-string (concat "sh " x))
     "\n" t)))

(defun get-nl-seperated (x)
  (with-temp-buffer
    (insert-file-contents x)
    (let ((lines (split-string (buffer-string) "\n" t)))
      (sort lines #'string<))))

(defun read-file-into-string (file-path)
  "Read the content of FILE-PATH into a string."
  (with-temp-buffer
    (insert-file-contents file-path)
    (buffer-string)))

```

I.I.V.I.X. install_iso.sh

Code: /installer/install_iso.sh

```bash
#!/bin/bash

while [[ "$#" -gt 0 ]]; do
  case $1 in
    --hostname)
      hostname="$2"
      shift
      ;;
    --username)
      username="$2"
      shift
      ;;
    --disk)
      disk="$2"
      shift
      ;;
    --timezone)
      timezone="$2"
      shift
      ;;
    --keymap)
```

```
      keymap="$2"
      shift
      ;;
    *)
      echo "Unknown option: $1"
      exit 1
      ;;
  esac
  shift
done

function get_parts() {
  disk=$1
  part=$(lsblk -o NAME,TYPE -n -p -l | awk -v disk="$disk" '$2=="part"' | grep $disk)
  echo "$part"
}
function substitute_variables() {
  local str="$1"
  shift
  for var; do
  str="${str//\$$var/${!var}}"
  done
  echo "$str"
}
function scm_file() {
  iso=$1
  if [ "$iso" = true ]; then
    echo "guix_iso.scm"
    return
  fi
  echo "guix_config.scm"
}
function make_disk_iso() {
  disk=$1
  sfdisk -f $disk < part_iso.sfdisk
  parted -s $disk resizepart 2 100%
  part=$(get_parts $disk)
  BOOT_PART=$(echo "$part" | awk 'NR==1{print $1}')
  ROOT_PART=$(echo "$part" | awk 'NR==2{print $1}')

  mkfs.fat -F32 $BOOT_PART
  mkfs.ext4 -F $ROOT_PART

  mount $ROOT_PART /mnt
  herd start cow-store /mnt
}
function get_part_uuid() {
  part=$1
  blkid -s UUID -o value $part
}
function guixInit() {
  DISK=$1
  HOSTNAME=$2
  USERNAME=$3
  SCM_FILE=$4
  TIMEZONE=$5
  KEYMAP=$6


  part=$(get_parts $disk)
  root_part=$(echo "$part" | awk 'NR==2{print $1}')

  ROOT_UUID=$(get_part_uuid $root_part)

  scheme_template=$(cat $SCM_FILE)
  scm=$(substitute_variables "$scheme_template" DISK HOSTNAME USERNAME ROOT_UUID TIMEZONE KEYMAP)

  mkdir /mnt/etc
  cp ../logos/Kudu_grub_image.svg /mnt/etc/Kudu_grub_image.svg

  echo "$scm" > /mnt/etc/config.scm
  guix pull
  guix package -u
  hash guix
  guix pull
  guix package -u
  hash guix
  guix system init /mnt/etc/config.scm /mnt

}
function setup_system() {
  USERNAME=$1

  mkdir -p /mnt/home/$USERNAME/
  git clone https://github.com/JanJoar/Kudu-Emacs.git /mnt/home/$USERNAME/.emacs.d -b devel
}
function setup_iso() {
  mkdir -p /mnt/root
  cp ./* /mnt/root
  git clone https://github.com/JanJoar/Kudu-Emacs.git /mnt/root/Kudu-Emacs -b devel
  dir="/root/Kudu-Emacs/installer"
  echo "emacs -nw -q -l $dir/installer.el --eval \"(Kudu-installer)\" --chdir $dir" > /mnt/root/.bashrc
}

make_disk_iso $disk
guixInit      \
  $disk     \
  $hostname   \
  $username   \
  "guix_iso.scm"    \
  $timezone   \
  $keymap
setup_iso
```

I.I.V.I.XI. install_quick.sh

Code: /installer/install_quick.sh

```
#!/bin/sh

echo choose option
# bash ./install_test.sh --hostname kud --username tobi --disk /dev/sda --timezone Europe/Stockholm --keymap sv
# bash ./install.sh --hostname kud --username tobi --disk /dev/sda --timezone Europe/Stockholm --keymap sv
#
# bash ./install_test.sh --hostname kud --username tobi --disk /dev/sda --timezone Europe/Stockholm --keymap sv
# bash ./install_iso.sh --hostname kud --username tobi --disk /dev/sda --timezone Europe/Stockholm --keymap sv

```

I.I.V.I.XII. install.sh

Code: /installer/install.sh

```
#!/bin/bash

while [[ "$#" -gt 0 ]]; do
  case $1 in
    --hostname)
      hostname="$2"
      shift
      ;;
    --username)
      username="$2"
      shift
      ;;
    --disk)
      disk="$2"
      shift
      ;;
    --timezone)
      timezone="$2"
      shift
      ;;
    --keymap)
      keymap="$2"
      shift
      ;;
    *)
      echo "Unknown option: $1"
      exit 1
      ;;
  esac
  shift
done

function get_parts() {
  disk=$1
  part=$(lsblk -o NAME,TYPE -n -p -l | awk -v disk="$disk" '$2=="part"' | grep $disk)
  echo "$part"
}
function substitute_variables() {
  local str="$1"
  shift
  for var; do
  str="${str//\$$var/${!var}}"
  done
  echo "$str"
}
function scm_file() {
  iso=$1
  if [ "$iso" = true ]; then
    echo "guix_iso.scm"
    return
  fi
  echo "guix_config.scm"
}

function make_disk() {
  disk=$1
  sfdisk -f $disk < part.sfdisk
  parted -s $disk resizepart 3 100%
  part=$(get_parts $disk)
  BOOT_PART=$(echo "$part" | awk 'NR==1{print $1}')
  SWAP_PART=$(echo "$part" | awk 'NR==2{print $1}')
  ROOT_PART=$(echo "$part" | awk 'NR==3{print $1}')

  mkfs.fat -F32 $BOOT_PART
  mkfs.ext4 -F $ROOT_PART
  mkswap $SWAP_PART

  swapon $SWAP_PART
  mount $ROOT_PART /mnt
  herd start cow-store /mnt
}
function get_part_uuid() {
  part=$1
  blkid -s UUID -o value $part
}
function guixInit() {
  DISK=$1
  HOSTNAME=$2
  USERNAME=$3
  SCM_FILE=$4
  TIMEZONE=$5
  KEYMAP=$6



  part=$(get_parts $disk)
  swap_part=$(echo "$part" | awk 'NR==2{print $1}')
  root_part=$(echo "$part" | awk 'NR==3{print $1}')

  SWAP_UUID=$(get_part_uuid $swap_part)
  ROOT_UUID=$(get_part_uuid $root_part)

  scheme_template=$(cat $SCM_FILE)
 scm=$(substitute_variables "$scheme_template" DISK HOSTNAME USERNAME SWAP_UUID ROOT_UUID TIMEZONE KEYMAP)

  mkdir /mnt/etc
  cp ../Logos/Kudu_grub_image.svg /mnt/etc/Kudu_grub_image.svg

  echo "$scm" > /mnt/etc/config.scm
  guix pull
  guix package -u
  hash guix
  guix pull
  guix package -u
  hash guix
  guix system init /mnt/etc/config.scm /mnt

}
function setup_system() {
  USERNAME=$1

  mkdir -p /mnt/home/$USERNAME/
  git clone https://github.com/JanJoar/Kudu-Emacs.git /mnt/home/$USERNAME/.emacs.d -b devel
}
make_disk $disk
guixInit      \
  $disk     \
  $hostname   \
  $username   \
  $(scm_file $iso)  \
  $timezone   \
  $keymap
setup_system $username

```

I.I.V.I.XIII. install_test.sh

Code: /installer/install_test.sh

```bash
#!/bin/bash

while [[ "$#" -gt 0 ]]; do
  case $1 in
    --hostname)
      hostname="$2"
      shift
      ;;
    --username)
      username="$2"
      shift
      ;;
    --disk)
      disk="$2"
      shift
      ;;
    --create-iso)
      iso=true
      shift
      ;;
    --timezone)
      timezone="$2"
      shift
      ;;
    --keymap)
      keymap="$2"
      shift
      ;;
    *)
```

```
      echo "Unknown option: $1"
      exit 1
      ;;
  esac
  shift
done

function substitute_variables() {
  local str="$1"
  shift
  for var; do
  str="${str//\$$var/${!var}}"
  done
  echo "$str"
}
function scm_file() {
  iso=$1
  if [ "$iso" = true ]; then
    echo "guix_iso.scm"
    return
  fi
  echo "guix_config.scm"
}

DISK=$disk
USERNAME=$username
HOSTNAME=$hostname
SCM_FILE=$(scm_file $iso)
TIMEZONE=$timezone
KEYMAP=$keymap
SWAP_UUID="swaps uuid"
ROOT_UUID="roots uuid"
scheme_template=$(cat $SCM_FILE)
scm=$(substitute_variables "$scheme_template" DISK HOSTNAME USERNAME SWAP_UUID ROOT_UUID TIMEZONE KEYMAP)
echo "$scm"
echo "Hostname: $hostname"
echo "Username: $username"
echo "Partition: $disk"

```

I.I.V.I.XIV. keymaps

Code: /installer/keymaps

```
af
al
altwin
am
apl
ara
at
au
az
ba
bd
be
bg
bqn
br
brai
bt
bw
by
ca
capslock
cd
ch
cm
cn
compose
ctrl
cz
```

```
de
digital_vndr
dk
dz
ee
eg
empty
epo
es
et
eu
eurosign
fi
fo
fr
fujitsu_vndr
gb
ge
gh
gn
gr
group
hp_vndr
hr
hu
id
ie
il
in
inet
iq
ir
is
it
jolla_vndr
jp
ke
keypad
kg
kh
kpdl
kr
kz
la
latam
latin
level2
level3
level5
lk
lt
lv
ma
macintosh_vndr
mao
md
me
mk
ml
mm
mn
mt
mv
my
nbsp
nec_vndr
ng
nl
no
nokia_vndr
np
olpc
parens
pc
```

```
ph
pk
pl
pt
ro
rs
ru
rupeesign
se
sgi_vndr
sharp_vndr
shift
si
sk
sn
sony_vndr
srvr_ctrl
sun_vndr
sy
terminate
tg
th
tj
tm
tr
trans
tw
typo
tz
ua
us
uz
vn
xfree68_vndr
za

```

I.I.V.I.XV. logo.ascii_art

Code: /installer/logo.ascii_art

```
                  .==               ::.
                .#@+            :-+%@@@%*-.
               -@@-           +*.    :+#@@@@##**+-
              -@%.          -#:          .:--:.
             =@*          .*=               *    .#%%+
            =@=          +*.               +=     :%@@:
           +@:         =+. .-*=         . :%       -@@.
          *%.  :++=---=. .  *#     .----. %.       +@-   .:
         *%   -++++-       +*   .:::     +=       =%: ..:@*       .--.
        ##  :+****+=:     +@=:--.       -%       ==     *+    .:--:
       ##         +@@.   .**=.         :@.     ::      ##  :-:
     .%*          *@:                 .@-  :::.       #@@+-.
    .%*          -@:                  %@+=-
   .%*          :%.                  :-.
  .%*          -@.              .:-+#@@@#=      .
 .%*          =@:          .....     :+%@@@%*+=:
.%+          =@@.   .:::::.
::           %@@%*+=:.

```

I.I.V.I.XVI. part_iso.sfdisk

Code: /installer/part_iso.sfdisk

```

label: gpt
label-id: 03273926-5F0F-468D-A19F-C2E0DC71B283

```

```
start=          2048, size= 4096, type=21686148-6449-6E6F-744E-656564454649,  uuid=A45601B8-CF20-4EAF-A097-07D9F62B413C, bootable
start=         6144, size= 1G, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4,  uuid=A8D6CE0E-31AC-4C73-855C-EF7F1329930A

```

I.I.V.I.XVII. part.sfdisk

Code: /installer/part.sfdisk

```
label: gpt
label-id: 03273926-5F0F-468D-A19F-C2E0DC71B283

start=          2048, size= 4096, type=21686148-6449-6E6F-744E-656564454649,  uuid=A45601B8-CF20-4EAF-A097-07D9F62B413C, bootable
start=                 6144,   size=   2097152,      type=0657FD6D-A4AB-43C4-84E5-0933C84B4F4F, uuid=4C0F761A-9246-457E-8340-8506C16701C9
start=      2103296, size= 1G, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4,  uuid=A8D6CE0E-31AC-4C73-855C-EF7F1329930A

```

I.I.V.I.XVIII. template.scm

Code: /installer/template.scm

```

(use-modules (gnu))
(use-service-modules networking ssh)
(use-package-modules screen ssh)

(operating-system
  (host-name "{{hostname}}")
  (timezone "{{timezone}}")
  (locale "{{locale}}")

  ;; Boot in "legacy" BIOS mode, assuming /dev/sdX is the
  ;; target hard disk, and "my-root" is the label of the target
  ;; root file system.
  (bootloader (bootloader-configuration
                (bootloader grub-bootloader)
                (targets '("{{disk_bootloader}}"))))
  ;; It's fitting to support the equally bare bones ‘-nographic’
  ;; QEMU option, which also nicely sidesteps forcing QWERTY.
  (kernel-arguments (list "console=ttyS0,115200"))
  (file-systems (cons (file-system
                        (device (file-system-label "kudu-root"))
                        (mount-point "/")
                        (type "ext4"))
                      %base-file-systems))

  ;; This is where user accounts are specified.  The "root"
  ;; account is implicit, and is initially created with the
  ;; empty password.
  (users
    (list
      {% for user in users %}
        (user-account
          (name "{user.name}")
          (comment "{user.comment}")
          (group "{user.group}")
          (supplementary-groups '("wheel" "audio" "video"))
        )
      {% endfor %}
    %base-user-accounts
  ))

  ;; Globally-installed packages.
  (packages (list
        screen
        emacs
        emacs-exwm
        wmctl
        brightnessctl
        git
        icecat
        openttd
        %base-packages
  ))

  ;; Add services to the baseline: a DHCP client and
  ;; an SSH server.
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type
                                   (openssh-configuration
                                    (openssh openssh-sans-x)
                                    (port-number 2222))))
                    %base-services)))

```

I.I.V.I.XIX. timezones

Code: /installer/timezones

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
```

```
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Ciudad_Juarez
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Nelson
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
```

```
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Nuuk
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Punta_Arenas
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
```

```
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Atyrau
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Barnaul
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Chita
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Famagusta
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
```

```
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qostanay
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Tomsk
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yangon
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
```

```
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST
EST5EDT
Egypt
Eire
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5
Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
```

```
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/GMT0
Etc/Greenwich
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Astrakhan
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Kirov
Europe/Kyiv
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Saratov
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Ulyanovsk
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
Factory
```

```
GB
GB-Eire
GMT
GMT+0
GMT-0
GMT0
Greenwich
HST
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kanton
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
```

```
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROC
ROK
Singapore
Turkey
UCT
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Samoa
UTC
Universal
W-SU
WET
Zulu
```

I.I.V.I.XX. test_get_parts.sh

Code: /installer/test_get_parts.sh

```bash
#!/bin/bash


function test_part() {

  disk=$1
  part=$(lsblk -o NAME,TYPE -n -p -l | awk -v disk="$disk" '$2=="part" && index($1, disk)==1 {print $1}')
  echo "$part"
}

```

I.I.V.II. quick-init.el

Code: /quick-init.el

```elisp
  (setq corfu-auto t
        visible-bell t
        vertico-mode t
        vertico-count 10
        show-paren-mode t
        show-paren-delay 0
        xterm-mouse-mode t
        load-prefer-newer t
        global-corfu-mode t
        pixel-scroll-mode t
        electric-pair-mode t
        corfu-prescient-mode t
        prescient-persist-mode t
        vertico-prescient-mode t
        prescient-history-length 5
        global-hide-mode-line-mode t
        pixel-scroll-precision-mode t
        prescient-sort-full-matches-first t
        native-comp-async-report-warnings-errors nil)

  (defalias 'yes-or-no-p 'y-or-n-p)
  (add-hook 'prog-mode-hook #'rainbow-delimiters-mode)
  (unless (display-graphic-p)
        (corfu-terminal-mode +1))

  (add-to-list 'completion-at-point-functions #'cape-dabbrev)
  (add-to-list 'completion-at-point-functions #'cape-file)
  (add-to-list 'completion-at-point-functions #'cape-elisp-block)
  (add-to-list 'completion-at-point-functions #'cape-history)
  (add-to-list 'completion-at-point-functions #'cape-keyword)

  (vertico-indexed-mode)
  (vertico-mouse-mode)
  (add-hook 'vertico-mode-hook #'marginalia-mode)
  (completion-styles '(orderless basic prescient))
     (completion-category-overrides '((file (styles basic partial-completion))))

  (defun sudo ()
    "Opens the current buffer at point with root privelages using TRAMP"
    (interactive)
    (let ((position (point)))
      (find-alternate-file (concat "/sudo::"
                                   (buffer-file-name (current-buffer))))
      (goto-char position)))

  (defun ! (n)
    "An emacs function to calculate the factorial of n using the calc library"
    (string-to-number (calc-eval (format "%s!" n))))

  (defun nPr (n k)
    "A function for calculating the number of permutations in combinatorics"
    (/
     (! n)
     (! (- n k))))

  (defun nCr (n k)
    "A function for calculating the number of combinations in combinatorics"
    (/
     (! n)
     (* (! k) (! (- n k)))))

  (defalias 'binomial 'nCr)

```

I.I.V.III. snippets

I.I.V.III.I. org-mode

I.I.V.III.I.I. cases

Code: /snippets/org-mode/cases

```
# -*- mode: snippet -*-
# name: LaTeX case
# key: cases
# --
\begin{cases}
${1:}
\end{cases}
```

I.I.V.III.I.II. cases

Code: /snippets/org-mode/cases~

```
# -*- mode: snippet -*-
# name: LaTeX case
# key: cases
# --
\begin{cases}
{${1:}}
\end{cases}
```

I.I.V.III.I.III. display_math

Code: /snippets/org-mode/display_math

```
# -*- mode: snippet -*-
# name: Display math environment
# key: math
# --
\[
${0:}
\]
```

I.I.V.III.I.IV. fraction

Code: /snippets/org-mode/fraction

```
# -*- mode: snippet -*-
# name: fraction
# key: fr
# --
\frac{${1:}}{${2:}}
```

I.I.V.III.I.V. fraction_dollar

Code: /snippets/org-mode/fraction_dollar

```
# -*- mode: snippet -*-
# name: fraction
# key: $fr
# --
$\frac{${1:}}{${2:}}

```

I.I.V.III.I.VI. fraction_dollar_2

Code: /snippets/org-mode/fraction_dollar_2

```
# -*- mode: snippet -*-
# name: fraction
# key: $fr$
# --
$\frac{${1:}}{${2:}}$
```

I.I.V.III.I.VII. f(x)

Code: /snippets/org-mode/f(x)

```
# -*- mode: snippet -*-
# name: f(x)
# contributor: JanJoar
# key: f
# --
f(x)
```

I.I.V.III.I.VIII. g(x)

Code: /snippets/org-mode/g(x)

```
# -*- mode: snippet -*-
# name: g(x)
# contributor: JanJoar
# key: g
# --
g(x)
```

I.I.V.III.I.IX. infinity

Code: /snippets/org-mode/infinity

```
# -*- mode: snippet -*-
# name: Infinty
# key: inf
# --
\infty
```

I.I.V.III.I.X. integral

Code: /snippets/org-mode/integral

```
# -*- mode: snippet -*-
# name: Integral
# key: int
# --
\[ \int_{${1:}}^{${2:}} ${3:} \,\mathrm{d}x \]
```

I.I.V.III.I.XI. integral_dollar

Code: /snippets/org-mode/integral_dollar

```
# -*- mode: snippet -*-
# name: Integral_dollar
# key: $int
# --
$\int_{${1:}}^{${2:}}\,\mathrm{d}x

```

I.I.V.III.I.XII. integral_dollar_2

Code: /snippets/org-mode/integral_dollar_2

```
# -*- mode: snippet -*-
# name: Integral_double_dollar
# key: $int$
# --
$\int_{${1:}}^{${2:}}\,\mathrm{d}x$
```

I.I.V.III.I.XIII. L

Code: /snippets/org-mode/L

```
# -*- mode: snippet -*-
# name: Calc L
# contributor: jdhao <jdhao@hotmail.com>
# key: L
# --
\mathcal{L}
```

I.I.V.III.I.XIV. lhd

Code: /snippets/org-mode/

```
# -*- mode: snippet -*-
# name: lhd
# key: lhd
# --
\lhd
```

I.I.V.III.I.XV. limit

Code: /snippets/org-mode/limit

```
# -*- mode: snippet -*-
# name: limit
# key: lim
# --
\lim_{{${1:}\to\ ${2:}}}

```

I.I.V.III.I.XVI. limit_dollar

Code: /snippets/org-mode/limit_dollar

```
# -*- mode: snippet -*-
# name: limit
# key: $lim
# --
$\lim_{{${1:}\to\ ${2:}}}

```

I.I.V.III.I.XVII. limit_dollar_2

Code: /snippets/org-mode/limit_dollar_2

```
# -*- mode: snippet -*-
# name: limit_dollar_double
# key: $lim$
# --
$\lim_{{${1:}\to\ ${2:}}}$
```

I.I.V.III.I.XVIII. mathbb

Code: /snippets/org-mode/mathbb

```
# -*- mode: snippet -*-
# name: set
# key: set
# --
\mathbb{${1:}}
```

I.I.V.III.I.XIX. rhd

Code: /snippets/org-mode/rhd

```
# -*- mode: snippet -*-
# name: rhd
# key: rhd
# --
\rhd
```

I.I.V.III.I.XX. sim

Code: /snippets/org-mode/sim

```
# -*- mode: snippet -*-
# name: sim
# key: ~
# --
\sim
```

I.I.V.III.I.XXI. sube

Code: /snippets/org-mode/sube

```
# -*- mode: snippet -*-
# name: sube
# key: sube
# --
\subseteq
```

I.I.V.III.I.XXII. subset

Code: /snippets/org-mode/subset

```
# -*- mode: snippet -*-
# name: sub
# key: sub
# --
\subset
```

I.I.V.III.I.XXIII. sum

Code: /snippets/org-mode/sum

```
# -*- mode: snippet -*-
# name: sum
# key: su
# --
\sum_{${1:}}^{${2:}}
```

I.I.V.III.I.XXIV. sum_dollar

Code: /snippets/org-mode/sum_dollar

```
# -*- mode: snippet -*-
# name: sum_dollar
# key: $su
# --
$\sum_{${1:}}^{${2:}}
```

I.I.V.III.I.XXV. org-mode

Code: /snippets/org-mode/sum_dollar_2

```
# -*- mode: snippet -*-
# name: sum_dollar_double
# key: $su$
# --
$\sum_{${1:}}^{${2:}}$
```

I.I.V.III.I.XXVI. sup

Code: /snippets/org-mode/

```
# -*- mode: snippet -*-
# name: sup
# key: sup
# --
\supset
```

I.I.V.III.I.XXVII. org-mode

Code: /snippets/org-mode/

```
# -*- mode: snippet -*-
# name: supeseteq
# key: supe
# --
\supe
```