THE UNIVERSITY OF SYDNEY

# INFO1910 S1 2023 Week 0 Tutorial

**Linux Setup**

## Installing Linux

Linux is a family of free[1] and open source operating systems. You **will** require a Linux operating system for this course; WSL will **not** be considered sufficient.

Here we will cover a general overview of different ways to install Linux, or get access to a Linux environment on your own computer. Depending on what you chose to do here, this can take anywhere from a few minutes to several days worth of work.

## Picking a Distribution

There are several hundred Linux distributions (or distros) out there, each with varying degrees of popularity. In general there is a trade-off between ease of use and flexibility between these distributions which range from 'working out of the box' to 'build from scratch yourself'. A few of the more common ones are listed here in order of their ease of use.

If you're new I'd suggest Mint with Cinnamon (it's something like a Windows 7 GUI with a Linux backend), Elementary OS (if you like the look and feel of a Mac) or Zorin (if you like the look and feel of Windows). All of these distros are derived from Ubuntu, and can share packages with it; when looking for help online the Ubuntu help will also apply to you.

- **Mint** A distribution heavily based on and compatible with Ubuntu, this is a rather popular modern distro that is suitable for both those new to Linux and longer term users. Currently the most popular Linux Distribution around. Cinnamon UI is very windows 7-ish.
- **Elementary OS** Looks like a Mac and is based on Ubuntu.
- **Zorin** Looks like Windows 8 and 10 and is based on Ubuntu.
- **Ubuntu** Now one of the older distributions and is based on Debian. It's somewhat bloated these days and is only the third most popular distribution, fine for running natively, but might chug if used in a VM. Would suggest getting the 'server version' if you're using a VM.

[1]Free as in both for free (gratis), and free as in free to modify or use with no restriction (libre). You are completely free to copy the entire base of the Linux source code and make your own operating system based on it, so long as you attribute the original authors.

- **Debian** A rock solid distribution with well tested packages. Can be considered the spiritual development distribution for Ubuntu. Also the second most popular distribution.
- **MX Linux** A very new distro, but seems to be gaining popularity. Based on debian, tries to balance the stability of debian with a more modern UI. A merger between antiX and MEPIS.
- **Fedora** The free version of Red Hat, well supported and maintained. Red Hat sees quite a lot of use in larger organisations as it is arguably the best Linux distro for supporting large numbers of different groups of users.
- **openSUSE** A rather old distribution, but still one of the better supported ones, it's quite friendly to both new users and to people who have been using Linux for a while.
- **Manjaro** An arch based distro with an easier installation. Comes with pre-installed window managers, audio and video software, (unlike arch).
- **Arch** A very flexible distribution that comes with nearly nothing but the bare bones out of the box. You will need to install your own window manager and other essential utilities (unless you like working in a raw tty environment). Very barebones but is supported by the most in depth Linux wiki around. Arch is famous for being something of a testing distro and receives package updates sometimes on the order of years before other distributions. This also means that there will be the occasional bug.

If you have some niche requirements, or just want to see some other distros here is (another) list of ones that might suit

- **Ubuntu MATE** Excellent compatibility for old hardware (in case you want to drop Linux onto an otherwise defunct computer).
- **openBSD** Incredibly secure operating system, once went a decade without needing a security patch. Also incredibly hard to actually do anything on.
- **Gentoo** Effectively 'just the source code' you need to compile the operating system yourself from the command line during installation. One of the best supported and most flexible Linux distributions, installing this for the first time may take a few days to a few weeks.
- **Linux From Scratch** Not actually a Linux distribution, more a set of tools to let you build your own Linux distribution.
- **CentOS** Good for servers, free, and compatible with Red Hat Enterprise Linux (RHEL), especially as Red Hat isn't itself free.
- **Tails** Edward Snowden's operating system of choice. A debian based distro that routes all internet traffic via Tor.
- **Asahi** A distro based on Arch that is designed to run on Apple's M1 chips (and outperforms macOS)
- **Alpine** a tiny distro designed to run purely in RAM, great for use in docker images.

As said before, there are hundreds more of these and if you're looking for something specific it's probably there.

# Window Managers and Desktop Environments

You may have seen above some discussion about ‘window managers’. This is basically what your desktop environment will look like. Some distros will give you different options when you go to get the iso so we should cover this briefly now. In general window managers take a "stacking" or a "tiling" approach. Stacking allows users to place windows on top of each other, tiling doesn’t but is more resource efficient. Unless otherwise stated all the window managers and environments take the "stacking" approach.

- **GNOME** One of the most common window managers, highly contentious post gnome-2, would probably recommend skipping it.
- **KDE** a reasonably flexible, but somewhat high performance window manager. Supports a range of visual effects.
- **MATE** Really just Gnome-2, there’s a reason that this has been forked and made its own desktop environment
- **Unity** Ubuntu’s new default, there’s a reason that there are a range of different linux distros that are "Ubuntu but not with unity"
- **XFCE** Debian’s default desktop environment, also comes with MX Linux, mid range on performance but with a large number of features.
- **Cinnamon** Mint’s WM, very useable. Can be extended with Cinnamon Spices for more features. Quite intuitive and Windows-y.
- **Pantheon** Elementary’s WM, looks like Mac works like a Mac, less resource intensive than a Mac.
- **i3** or the more aesthetic **i3-gaps** is a virtually mouseless tiling window manager. Very fast, very low resource footprint, can noticeably increase battery life in laptops.
- **dwm** something of a hybrid between the "stacking" and "tiling" approaches. Slightly more resource intensive than a tiling window manager, but far less restrictive.
- **xmonad** Lightweight tiling approach, heavily editable.
- **None** It’s an option to simply not have a window manager; you get to run everything through a tty session. This may be a good idea if you’re using a virtual machine as it’s very lightweight.

You can install virtually any of these, or other window managers on whatever flavour of Linux you see fit if you want to test them out. But most will come with a couple of different defaults for you to pick between. Each of these (and there are many more), are themselves customisable and come with a range of options.

# Different ways to Install

Once you've picked your distro and what window manager you want to use, go and grab the relevant ISO file from the site. You may need either the UEFI installer for recent computers, or BIOS installer for older computers, most modern installers cover both, but be sure to check.

For this course, the recommended method of installation is either straight onto your computer (wiping the existing OS), or dual booting, which will split the computer between multiple operating systems and allow you to select between them at launch. For INFO1910 using macOS or WSL is **not** recommended. Using a VM should be considered a stopgap option. If your particular computational setup is delicate for whatever reason, you might also consider using a small single board computer for this course (for example bring and sshing to a Raspberry Pi).

- Installation onto 'bare metal' - The hard drive is reformatted and the computer will only boot to Linux.
- Dual Boot - The hard drive is partitioned such that when you boot it can select between a Linux partition, or another operating system that has been installed. Harder than the bare metal install but more flexible. Linux will typically require at least 20GB of hard drive space. If your computer has a small hard drive you may want to consider disabling the SWAP partition.
- Virtual Machine - Linux is virtualised under the host operating system. The computer boots to your existing OS, and then virtualisation software launches an emulated Linux. All calls from the VM are routed back through the host operating system making this slower than a native installation.
- USB - Linux is installed on a USB (at least 16 GB is a good idea), the USB is then used as the boot drive on another computer. When the computer is booted without the USB it will run the original OS.

## Straight onto the Computer

If you have a computer lying around that you're not using (or are tired of the constant windows updates pre-allocating 25% of your bandwidth to itself at all times and spying on you) you might consider simply installing Linux on the machine directly. For this you will need a USB that will get wiped (so don't leave anything important on there). Also your computer's hard drive will be wiped, so again, clear anything you need off there first.

Depending on the distro this may range from a few hundred megabytes to a couple of gigabytes. Once you have your ISO you are going to want to map it to a USB, either the **dd** command or a utility such as rufus (for windows) or etcher (for Mac) can be used here to make your USB bootable.

A general word of warning, the following instructions worked when tested, but across different machines or different versions things may have changed, if something is not working look it up, and if you still can't manage it feel free to ask questions on Ed.

**Windows**
**These instructions have been confirmed to be working up to Windows 10, there may be some minor changes in Windows 11.** Next you're going to need to boot your computer from the USB.

This can be quite a chore on more modern windows and Mac computers. Pre-windows 8 simply restart your machine and press whatever button the vendor specifies to see your BIOS/UEFI menu (generally F2, F8, F10 or del, but look it up if you're unsure).

For windows 8 and 10 you need to Settings, Update and Security, Recovery, Advanced Startup, Restart Now, Troubleshoot, Advanced Options, UEFI Firmware settings, Restart[2]. Your computer should reboot and you should be shown your UEFI firmware settings. If you're not you should try disabling fastboot too. (Disabling fastboot is in general a good idea as you will be able to use a single key to enter the UEFI in the future rather than the whole menu approach).

For BIOS this is simple, change the boot order to boot from your USB device first. For UEFI you need to disable secure boot[3], wipe the signed bootloader table and then boot from your USB device. Your computer should restart and you should now see a GRUB or similar menu with options to boot, install and other options such as debug or recovery modes.

**macOS- Intel x86**
**Asahi Linux for the M1 chipset is still in alpha. By all accounts it works and may have better performance than macOS on the same hardware, but there may be bugs. If this applies to you consider dual booting or running a VM.** If you're a Mac user who skipped the last few paragraphs then first we need to disable 'secure boot'[4], reboot your computer and hold command and R while doing so, this should open the Utilities window. From the menu, find utilities, startup security utility and enter your username and password. Select "no security" to disable secure boot, then restart your device.

Power it up again, this time while holding the option key and select EFI boot from the menu that appears. When you get to your grub menu, instead of selecting any of the options press 'e' and you should see a few lines of text. You're going to need to edit the grub boot entry. One of the lines should start with "file=/", append "nomodeset" to the very end of that line after "quiet splash" make sure there is a space. Then press F10 to boot.

**macOS - M1**
Currently the only option for M1 is Asahi Linux. You can find details on how to install this here.

**Installation**
From here it's relatively straightforward to simply follow the installation prompts. If you've picked a distro such as Arch that does not have a graphical installation environment, then you are on your own from here on out.

## Dual Booting

Sometimes you want to be able to boot your computer into either Linux or something else (such as a different flavour of Linux). If you want to have the option to boot into either your current OS or into follow the initial instructions to boot from a USB and get to the start of the installation. When you start installing Linux you should instead partition your current drive in two and only install Linux to the new partition. You will typically want at the very least 20GB for this partition, though you may

[2] At this point you might get the impression that whoever designed these menus explicitly did not want you getting here.

[3] Microsoft decided to try to lock computers to only boot Windows by joining an 'independant' UEFI design panel and suggesting this as a security feature

[4] Apple also decided to try to lock their devices so that they can only boot macOS

find this amount of space a bit restrictive, 32GB or more is suggested.

With Mac you should also install rEFInd to manage booting.

When you reboot after installing you should see a grub menu with both Linux and your previous operating system listed. In the past there have been some compatibility issues here, though Ubuntu is reported to work with nearly everything. On a Mac you should instead find both Linux and Mac listed in your rEFInd menu.

Given you are playing around with partitions on your device, it is **strongly** recommended that you back up any files on the device before trying to set your computer up for dual booting.

## Virtual Machines

If your computer has less than 6GB of RAM then you probably don't want to do this, as Windows and macOS generally eat 2-4GB idle, and you then are looking for a second OS that needs another 2GB of RAM. If you *really* want to do this on a lower end machine, look into running a server version of your distro such as the server version of Ubuntu. This will mean that you have no GUI, but performance should be much better. Given that you should be using the terminal for just about everything you need, the lack of a GUI should not be a hinderance.

You're going to want either virtualbox (free) or vmware (not-free) and the ISO. Create a new virtual machine, you will need enough virtual hard disc space for your operating system; 16-32GB is more than enough for this, you can go lower with some of the lighter-weight distributions. You will need to also set RAM, typically you will need to allocate enough RAM that guest OS can function normally while still leaving enough for the normal functions of the host OS.

Boot your virtual machine and follow the installation as normal.

## Non-Persistent USB

You can use your bootable USB to boot and try the OS out rather than actually installing it. Then simply save your files to some location on your regular hard drive. The downsides of this method is that it significantly decreases the lifespan of the USB, that you may not be able to install packages, and that it will be IO bound by your USB drive and consequently be slow (though probably still faster than a VM).

## Persistent Live USB

You can also create and install a persistent usb version that you can boot your computer from. The entire operating system, and your user files will be sorted on the USB and treated as the hard drive when you boot from it.

Downside is that this very quickly burns through the life of the USB due to the large number of reads and writes. It's strongly suggested to not have a swap partition if doing this. This also means that if you run out of RAM your computer will simply crash rather than slow down. Also reading and writing is IO bound on the USB, so will be slower than a real installation.

For this you should check your distros exact installation instructions for a persistent boot. This method

can be extended to high speed portable SSDs, sometimes achieving near inbuilt disc speeds while still being portable. However the hardware for this does tend to be expensive and IO bounds can still be a limitation.

## KVM

This presumes that you are already running Linux, but you can also set up a Kernel Virtual Machine with hardware passthrough and simply hand control of the hardware to your virtual machine. This eliminates the downside of VMs being slow, while adding the new complication that you cannot control the hardware while the VM is. This requires two monitors and two graphics cards (the keyboard and mouse can be shared via synergy, which is open source and has its own git repo).

This can also run other operating systems, such as Windows from within Linux without too much of a slow-down.

# Text Editors

If you do not have a preferred text editor yet, feel free to consider the following.

- **Sublimetext or Limetext:** Cross platform text editor, has its own internal package manager and can be configured into a full IDE for most languages, Limetext claims to be an API compatible FOSS equivalent, but may have significant issues running on Windows. Also developed in Sydney, so there's that.
- **Atom:** Developed by github (who are now owned by Microsoft), also has an internal package manager and also can be configured into a full IDE for most languages. FOSS. Uses the electron framework (basically it's a bunch of javascript and runs in its own browser), so it can consume quite a bit of RAM. There might be concerns for the long term support of Atom once Microsoft realises that they're developing two different competing text editors, both using the electron framework (see the next entry). Atom is also pre-installed on the lab machines, so having a passing familiarity might be useful.
- **Vscode:** Developed by Windows (obvious owned by Microsoft), has its own internal package manager, can be configured into a full IDE for most languages. Mostly FOSS despite being developed by windows (some bits in some installations have odd black boxes built in). Also uses electron and so can be heavy on RAM.
- **Vi/Vim, Emacs:** Text editors with no mouse support, it is incredibly useful to know how to use at least one of these. The primary reason these editors have no mouse support is that they nearly predate the mouse. The arguments about the merits of Vi and Emacs pre-date the internet and have their own wikipedia page that may be considered suggested reading. In short emacs is considered more featured (having its own web browser, e-mail client and games and is often joked as being an operating system), while vim is more lightweight and is more ubiquitous. If learning vim, there is a dedicated `vimtutor` command that provides an in-vim, vim tutorial.
- **Nano:** Code for 'I have not learnt how to use vim or emacs', ok as a stopgap measure while you learn one of the above, but otherwise should not be something you use in the long term.
- **IDEs:** Integrated development environments, fancy text editors with GUIs and run buttons such as IDLE, Codeblocks, Eclipse and others. Generally when learning to program and how to work with a computer these tools tend to create an over-reliance on inbuilt features and are particularly detrimental for learning to use the command line. Useful to learn later down the track, explicitly prohibited for use in this course.