Unit XXIX Assignment II

By Nathan Windisch

PIII - Plan an installation and an upgrade

Introduction & Planning

The software that we are going to be installing will be *Everything*. *Everything* is an ultrafast item indexer that searches through any and all folders that you choose. It is, in my opinion, far superior to the Windows search, which is very slow even on an SSD. A non-administrative version of *Everything* on all systems within UTC Reading, allowing users to access their files faster and easier.

Testing

The one issue with *Everything* is that is very powerful. Permissions will need to be set up correctly so that no damage is made to the system, such as editing registry files or even other people's files. Lots of testing will need to be done before the final patch will be rolled out to all systems, including seeing if certain files can be edited, and if the program works with different environments with certain programs and features enabled and disabled.

Delivery

Everything shall be delivered to the user's systems over the course of several weeks, starting with the IT team, then the Senior Leadership Team, then the heads of department, then the teachers and finally the students. This process means that if there are any issues that are encountered during the delivery period, less people will be affected due to the fact that it is an incremental amount of people over a long time, meaning that there is less damage an issue is found within the first few groups of people that it is rolled out to.

Shipping

Everything will be shipped out a few days after it is installed. This is to ensure that all the systems that have the software delivered, work at 100% efficiency and at an optimum speed for a workplace environment. While it is easy to think of Delivery and Shipping as the same thing, it is important to realize that they are different things. Delivery comes first and is used to install the software onto the system, and Shipping comes second and is used to distribute the software amongst the users.

Storage

The storage systems that will be used will be the internal Hard Disk Drives, or HDDs, within the user's computers. *Everything* is quite a small program, as it only organizes and displays files. This means that small amounts of data will be created, such as the user's preferences and files that they want to view on default. This means that the IT team at UTC Reading will need to check the HDDs, and ensure that a few megabytes of storage are available, at least. This is because the program will either not be able to install properly, or will not be able to work correctly after installation due to a lack of storage. A fresh Windows 8 install with about 80,000s files will take approximately 6MB of

RAM and less than 3MB of Disk Space. For reference, 1,000,000 files will use about 50MB of RAM and 15MB of Disk Space.

Software Specifications

The software specifications that will be required is Windows XP, Windows Vista, Windows 7, Windows 8 or Windows 10. No other operating systems will be able to run *Everything*. *Everything* also needs to be run as an Administrator or the *Everything Service*, in order to access NTFS indexing. Also, the Central Processing Unit will need to be fast enough to perform the calculations, but most CPUs have the power to do that now. Along with the CPU, there will need to be enough Random Access Memory to perform the searches but, again, most computers have enough RAM to do that. The final thing that is needed is the correct type of architecture to run the program. Luckily, *Everything* has both x86 and x64 installs, meaning that it will work with all versions of Windows that I have previously mentioned.

Communication

Communication will be required when rolling out this update, as all participants who are involved will need to be aware of the changes that are occurring. This means that emails will need to be sent out periodically in order to keep everyone up to speed. Also, the Senior Leadership Team will need to be given a biweekly update on the progress that the IT team has been making, including a list of all users and machines that *Everything* has been installed to. Also VoidTools, the developers of the product, will need to be contacted if there are any issues or bugs so that they can both give advice and fix the bugs for future versions of the software.

Logs

Logs will also need to be stored as a way of viewing what went wrong if there are any issues, and also as a method of storing data about which machines and users have installed *Everything*, meaning that it can be used as a tracking device for which users have got the program installed.

Security

Security is vital to UTC Reading and this means that *Everything* will need to be locked down to prevent access to files that users cannot normally access. As a result of this, the logs for the program will need to be carefully monitored and any malicious activity using *Everything* will need to be reported to the Senior Leadership Team. Also, the program's source code will need to be scanned to ensure that there are no viruses or malicious code within it. Luckily, *Everything* is open source, meaning that the code will be easy to view and will require no decompiling.

Confidentiality

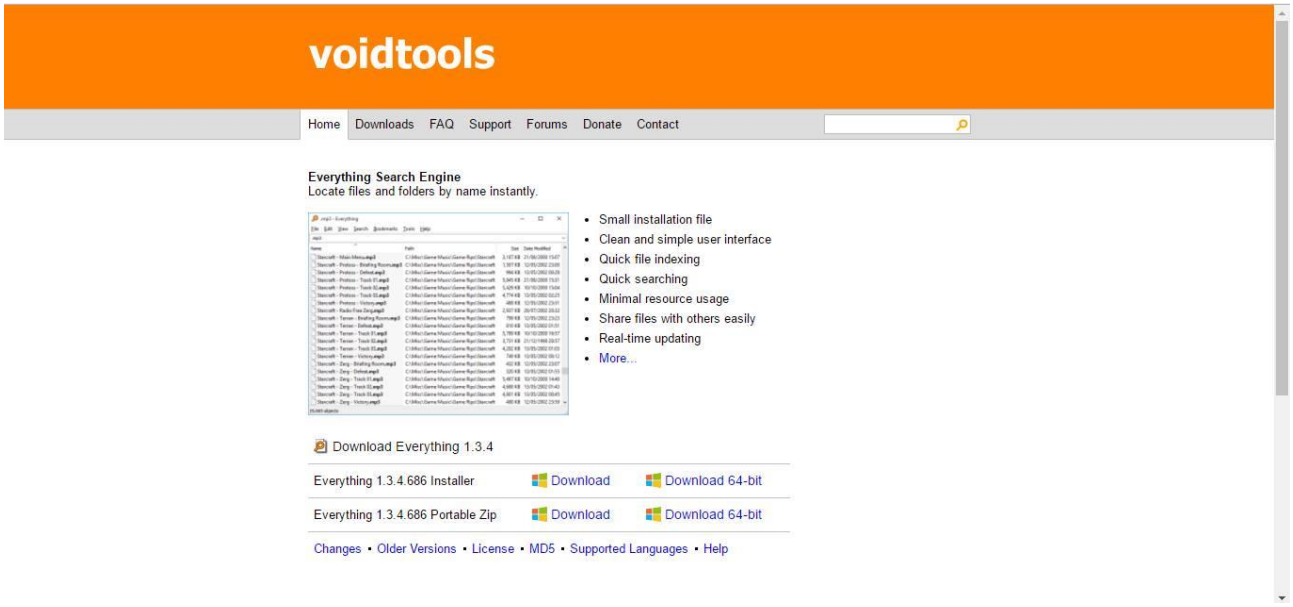
While logs need to be saved to ensure that the software is being used correctly, certain privacy rules must be followed. For instance, login details should only be used by those authorized to use them. Also, system administrators and users should only use *Everything* for accessing their own data. Every user's data is their own, and should only be accessed when certain conditions are met, such as the data being used in a criminal investigation. Also, any remote connections to the network must be sent over a secure connection using SSH or SSL.

Contractual Requirements

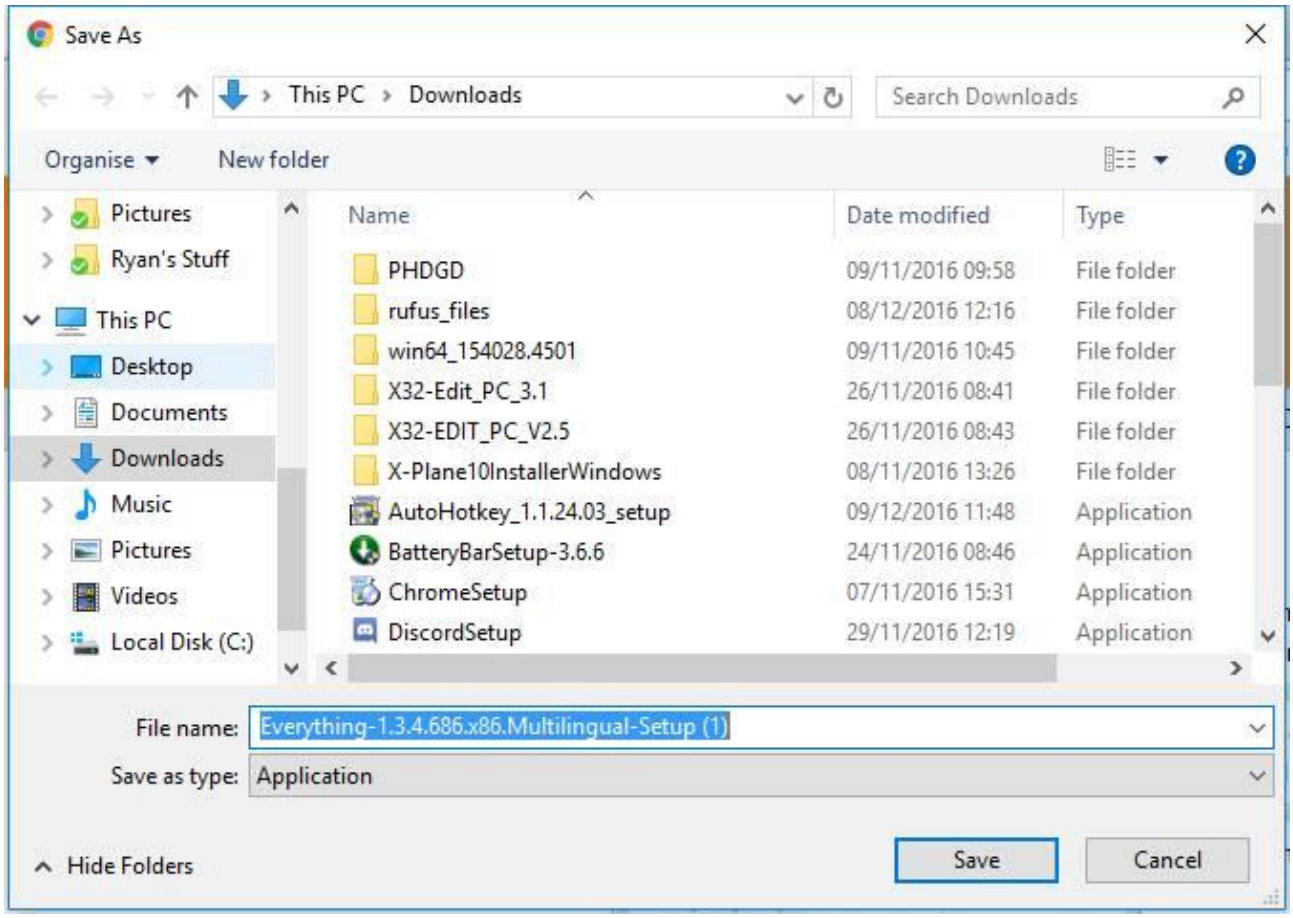
Contractual requirements are important to installing any software and *Everything* is no exception. While *Everything* is open source, any changes made to the code for either optimization or security purposes cannot be sold to a third party. All code edits must show that it was based off the *Everything* software, and cannot be sold for profit without the consent of the original author. Also, any Terms of Service contracts that the users of the UTC Reading network sign may need to be updated in order to align with the usage of the new software. For instance, the agreement must state that access of other user's data is prohibited.

PIV - Record and Complete a Software Installation

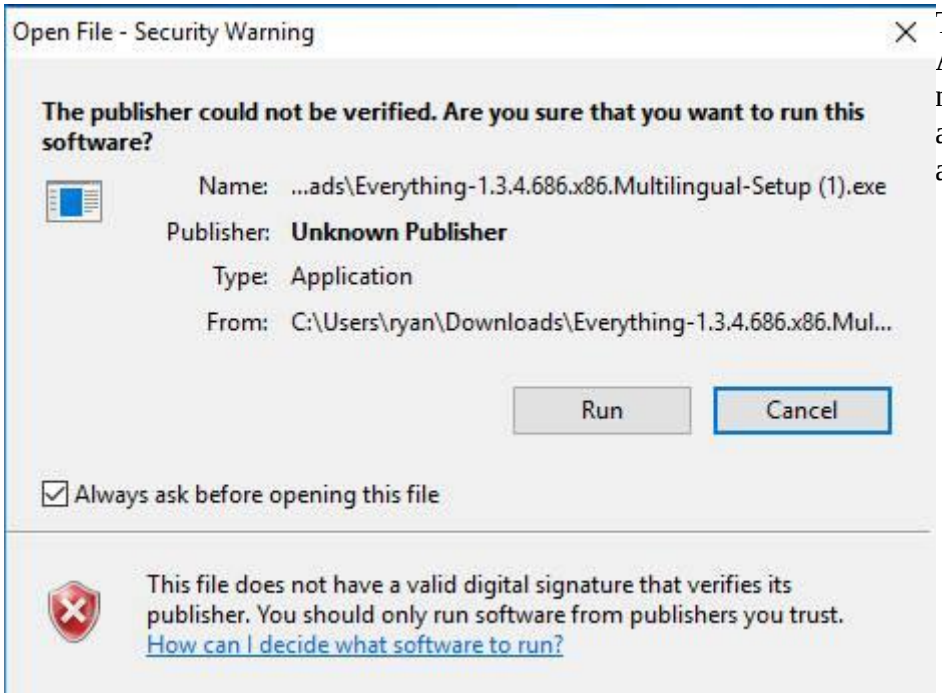
The software was installed over a few weeks on a variety of different machines. The following report is based on my installation on our test machine running Windows 10 (x64). *Everything* was a rather quick install, taking about 3 minutes to install.



The following image is of the website, http://voidtools.com. This is the official website of *Everything* and contains downloads, development versions, FAQs, documentation and source code.

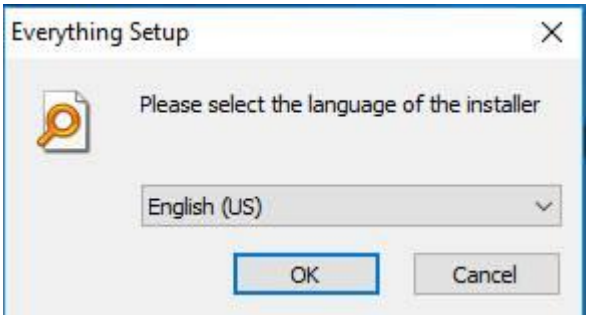


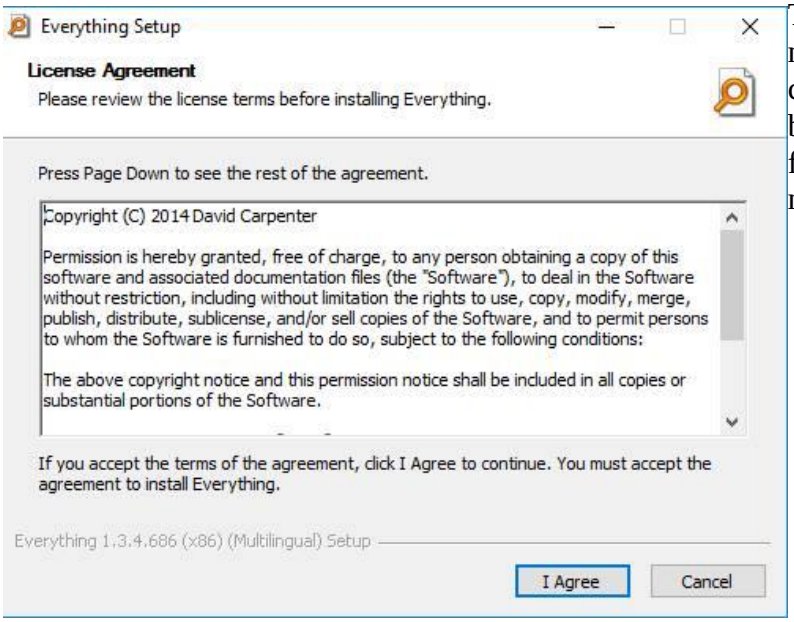
This is an image of me saving the .exe file. This is the program that is required to run in order to install *Everything*.



The installation required Administrative rights, meaning that I had to log in as an Administrator account.

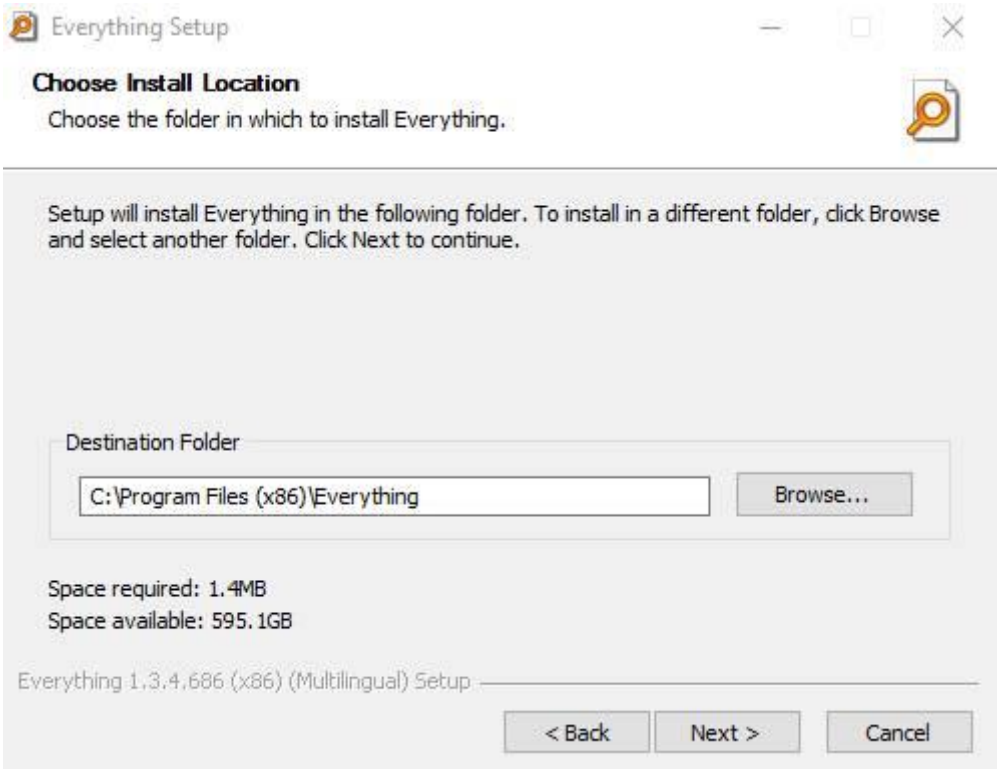
In this step I had to choose the language that the software was going to use. Given that I am fluent in English, as are all the students and staff at UTC Reading, I decided to choose English as the language used.

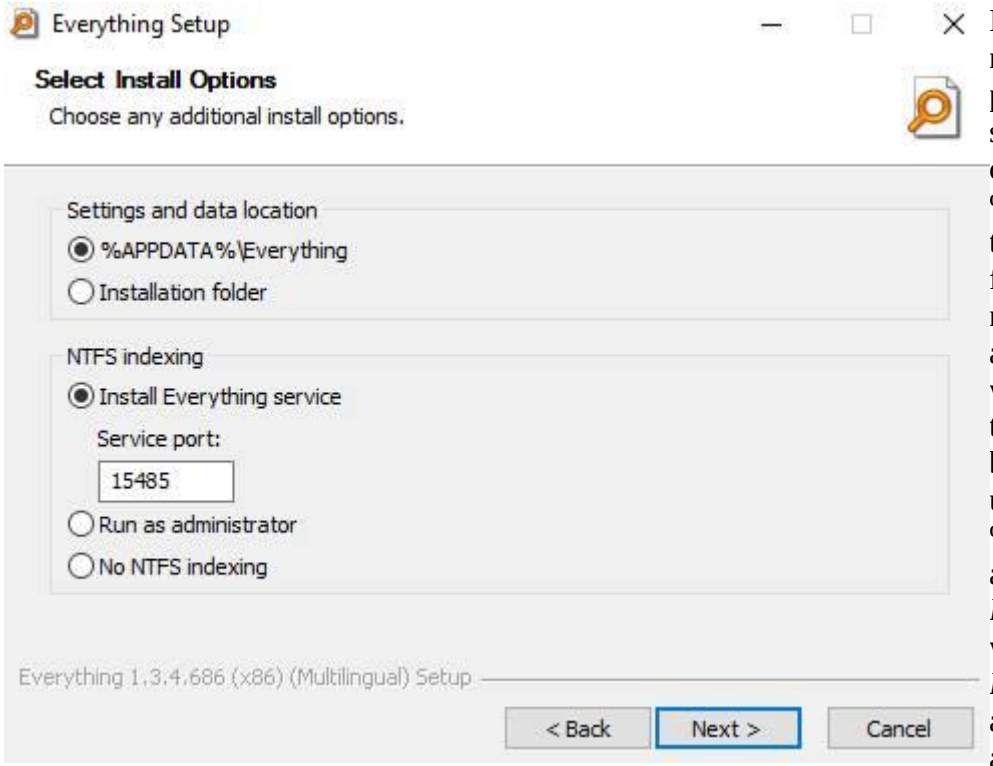




This is the license agreement that must be accepted before the software can be installed. If this license is broken the UTC Reading can be liable for a law suit, costing both time and money.

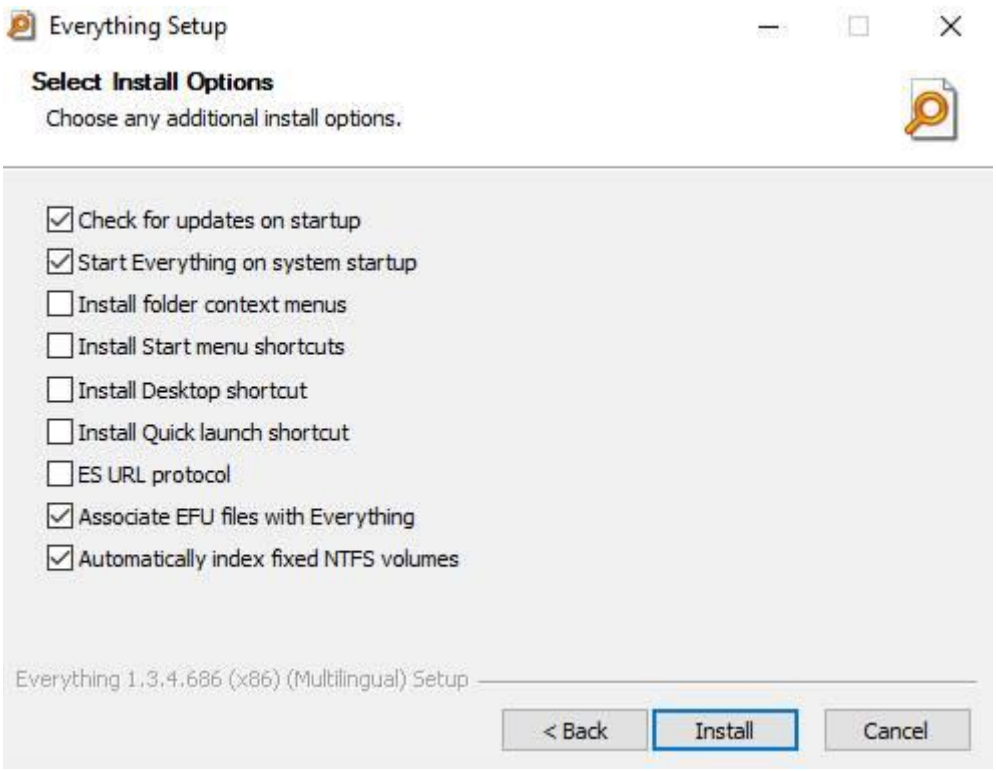
In this step I had to choose what location I would save the program's files to. I chose the default location as it was the recommended place to store it. It is important to note that I had to pick a space that had enough storage space but it only takes up 1.4MB and I had 595.1GB of space free, which was plenty.



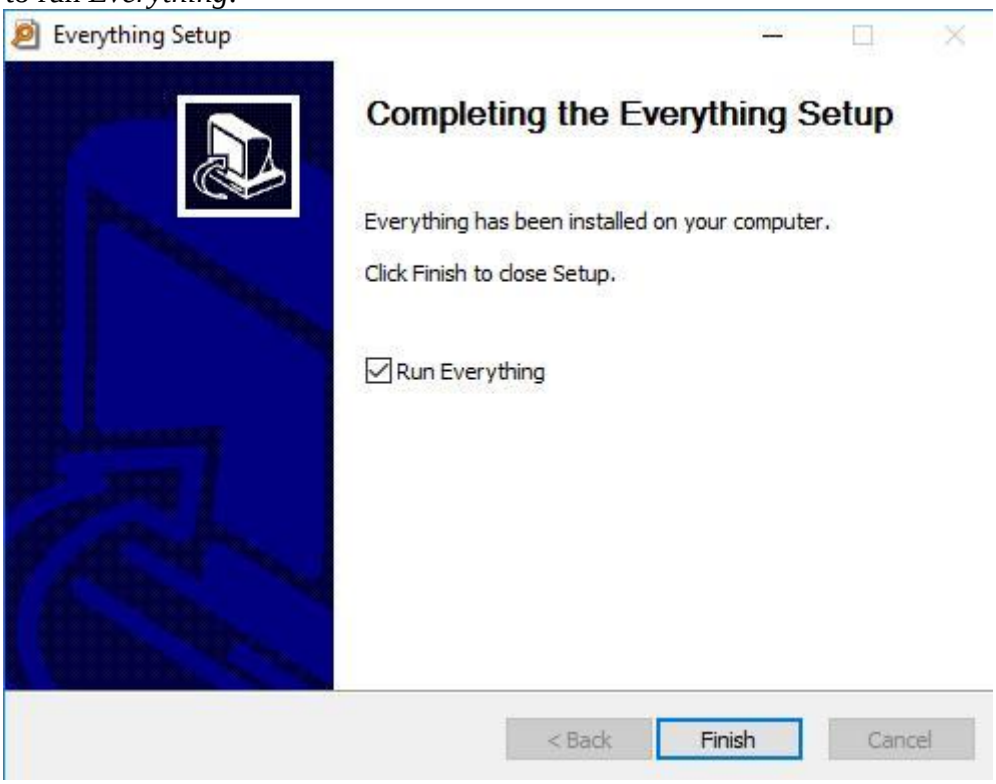


In this image, and the next one, I chose the place to save the settings and data. I chose to save it under %appdata%, rather than the installation folder as it was the recommended location and it meant that users wouldn't be able to edit the configuration to break the program, as users cannot access the %appdata% folder. I also chose to install *Everything Service*, which is a way to run *Everything* without administrator access and still use NTFS indexing.

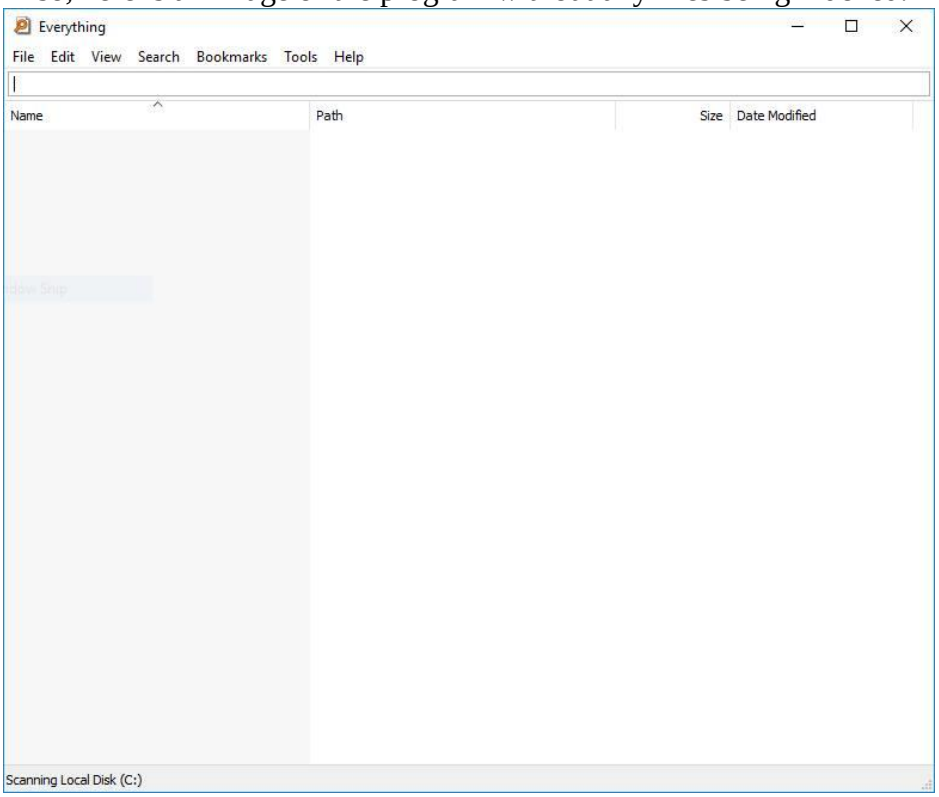
In the image to the left, I chose to check for updates on start, meaning that the program is always updated. I also chose to run *Everything* on startup, meaning that it would always be run. I decided to not install shortcuts and context menus as I felt that I would take up extra space and would be unused, making them pointless. I decided to associate EFU files with *Everything* as it should speed up the program, which is the same reason as to why I used automatic index fixing.



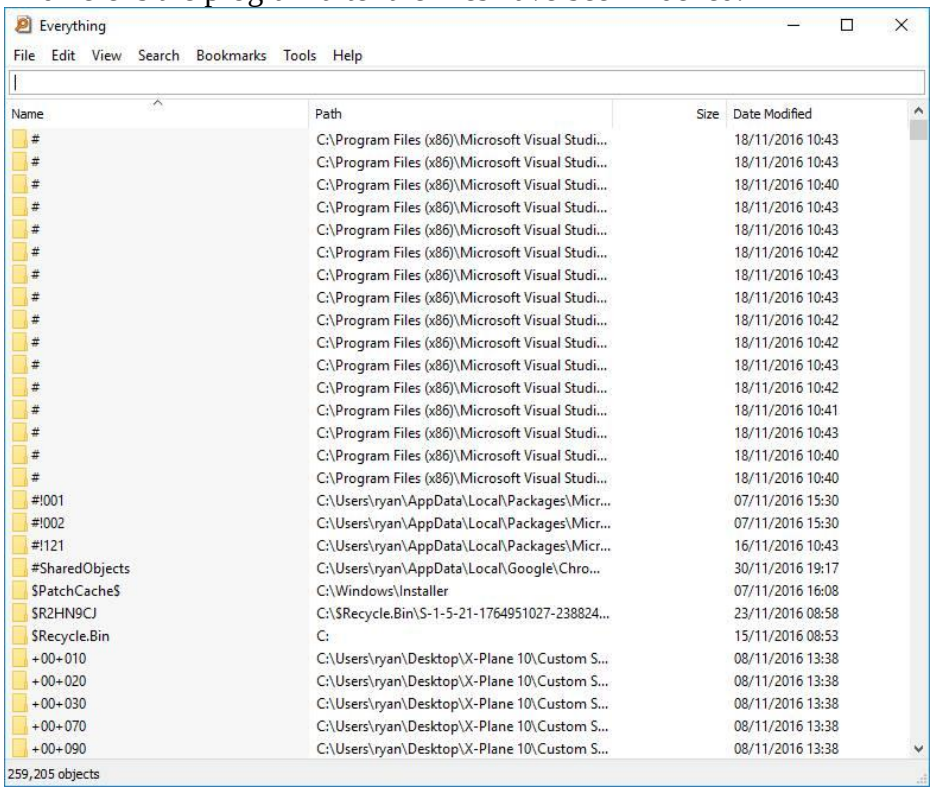
I was finally finished with the program installation. Here is the final step which gave me the option to run *Everything*.



Also, here is an image of the program without any files being indexed.

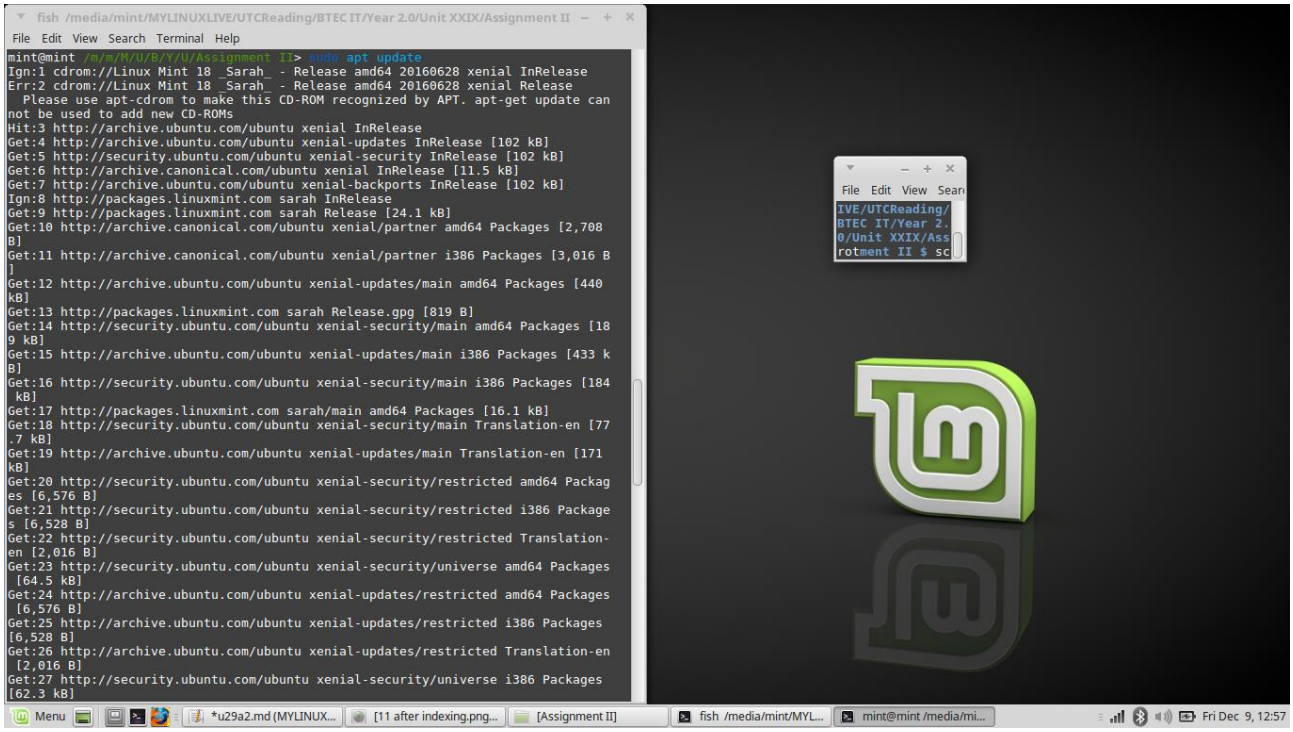


And here is the program after the files have been indexed.

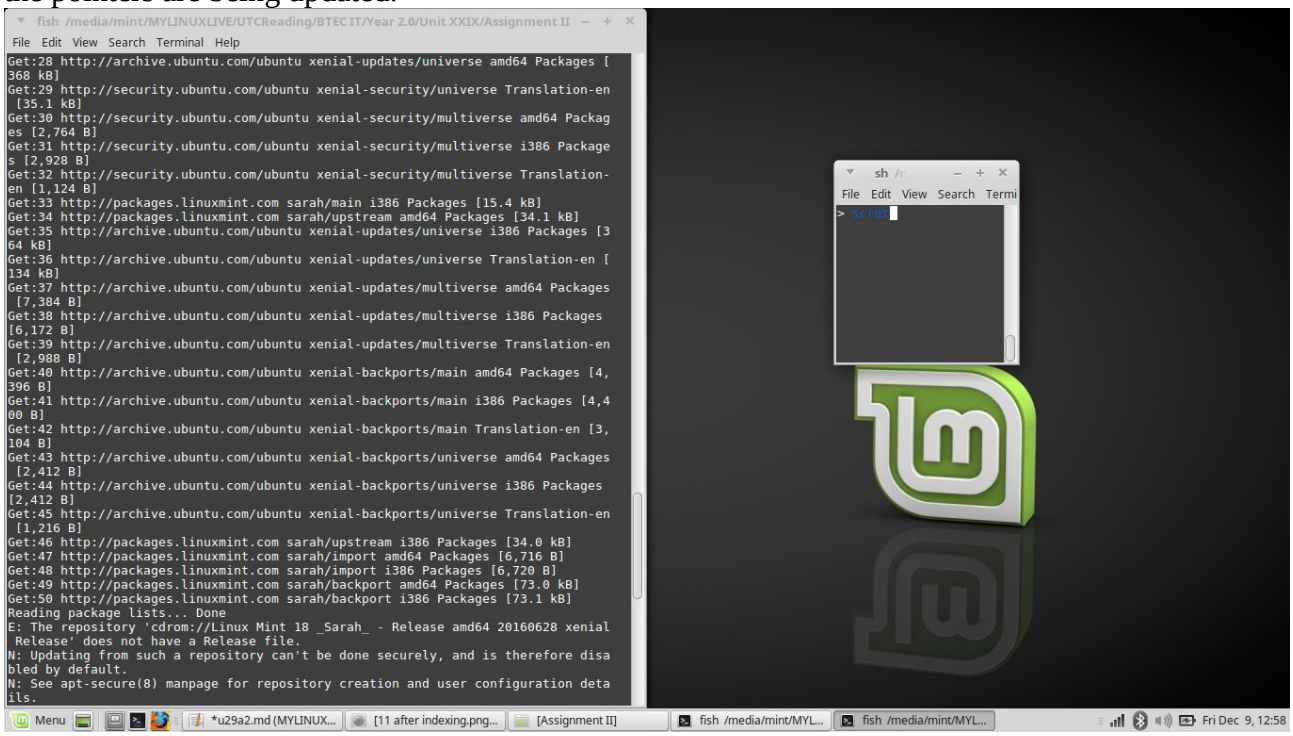


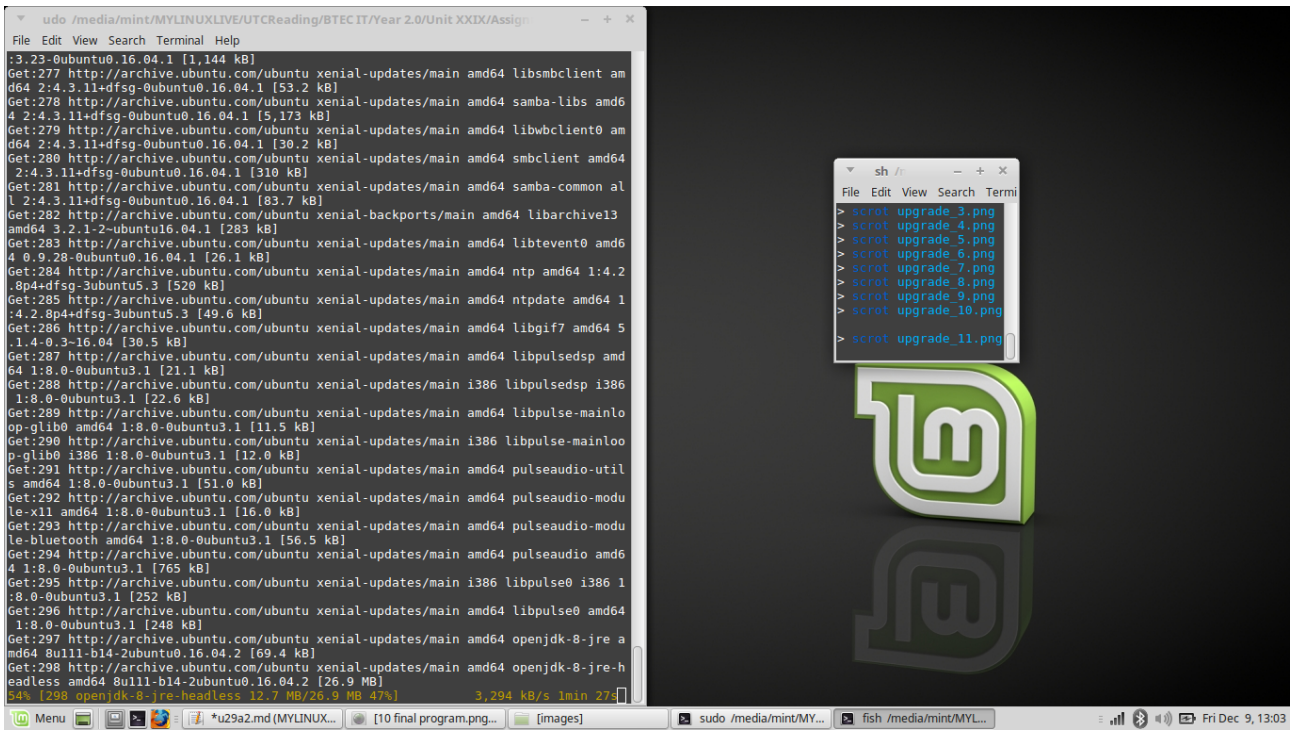
PV - Record and Complete a Software Upgrade

The version of *Everything* that I installed was the latest version, meaning that I could not upgrade anything. Because of this, I decided to update my Linux Mint distribution on my secondary Hard Disk Drive.



At first, I ran the command `sudo apt update`. This command pings all the file servers for packages that are installed on the machine, including the operating system. It then updates the list of packages that are installed on the machine, and performs changes to the list accordingly. This means that the machine knows where to go to download the packages when the upgrade command is sent. In short, the pointers are being updated.





After this I ran the `sudo apt upgrade` command. This command then goes to all the pointers that I previously mentioned and downloads the files. After that, the system upgrades all the packages that require updates, by extracting the files.

MIII - Design and Implement Procedure to Preserve Data Integrity

There are a few things required in the IT world, and one of which is data integrity. Data integrity is the "accuracy and consistency of data within a data structure". This means that all data sent and received within a network needs to be secure. This can be confirmed by using things called checksums. Checksums use mathematics to generate two numbers that are intrinsically linked, the first being a larger number that is bound to the file, and the second is a file that is sent along with it. If the numbers do not match after a mathematical equation is performed on them, such as multiplication, division or modulo, then the file that was sent has been compromised. Compromising can either be in the form of the data not being sent properly, or by the data being manipulated during transfer, either by a system process or an outside force.

Another point of data integrity is the security of the data once it is on the system. A way that this can be achieved is by performing backups. One type of backup is a full backup, where ALL of the data specified is backed up to the external servers. This type of backup may only be performed once every few months.

Another type of data backup is differential, where only the data that has been changed since the last backup occurs is stored elsewhere. This type of backup should be done once every few weeks or so. An example of this is as follows:

- A full backup is performed on Day #1.
- A differential backup is performed on Days #2 and #3. All of these backups will contain the data that has changed from Day #1, meaning that Day #2 will only contain Day #2's work, where as Day #3 will have Day #2 and Day #3's work.

This means that differential backups can take up lots of space very quickly, which is why it shouldn't be done all the time.

The final type of backup that I shall be covering is incremental, where only the data that has changed since the last incremental backup occurs is stored elsewhere. This type of backup should be performed daily or every other day. An example of this is as follows:

- A full backup is performed on Day #1.
- An incremental backup is performed on Days #2 and #3. All of these backups will contain the data that has changed from Day #x-1, meaning that Day #2 will only contain Day #2's work, and Day #3 will have only have Day #3's work.

Another type of data integrity is restore points. Restore points are a way that Microsoft systems revert to a previous state, including system files, installed applications, registry settings and system settings, from a different point in time. These files are stored using NTFS compression, meaning that they are very compact and can only be used if necessary.

My Linux distribution is running on a live system, meaning that no changes are saved. This means if I mess up an installation all I need to do is restart my machine and all the system files shall be reset, along with any other files and packages I have saved on the system.

MIV - Design a Procedure to Back Out of Software Upgrades

Back out procedures are highly important when installing new software. They are to be used if the software installed creates errors that prevent the system from running at an optimum standard. The back out procedure that UTC Reading should follow should be:

- Create a system restore point at the start of the upgrade.
- Perform the upgrade.
- Run tests on the system.
- If required, restore from the system restore point.

The system restore point should be made to take up as little data as possible, to ensure that the Hard Disk Drive can be used for actual work. Another feature of this backup procedure could be that the system restore points are saved to an off site location, for even greater security to prevent from physical threats such as a fire. Another thing that can be done is Hard Drive Cloning, which is where the data on a Hard Disk Drive or Solid State Drive is saved to another location for greater storage. Also, the software that has just been installed could be uninstalled from the system, which may fix the issue. Registry backups can be taken to ensure that the finer configurations of the system are saved for later use. Also, data integrity must be considered when performing backups. All the data in the backups must be identical to the ones taken from the drives before the upgrade was performed.

DI - Justify a Particular Installation up Upgrade

The upgrade that we are performing on the system are necessary as they allow for both faster access to files, along with updated software. The cost of not having these upgrades occurring may be more than the cost of performing them, due to the fact that lots of time can be lost searching for files. Also, both of the upgrades are free so, forgetting about backup costs and the possible price of failure due to incompatibility errors, there will be a definite profit incurred when upgrading the system and installing *Everything*. Hardware requirements should already be fine, due to the fact that the two changes that we have made to the system are very lightweight and not very resource intensive. Security threats should not be a concern from *Everything*, as it is open source program meaning that there should be no viruses as all the code is clearly visible. As far as updating the Linux system, only packages that you trust should be installed, meaning that the upgrade should not add any viruses that were not there before. As long as the files have been backed up, data and system integrity should be sound and any corruption should not affect the system too greatly.