

Which company has had the greatest effect on the field of Computer Science and Consumer Computing: Apple Incorporated or Bell Laboratories?

Contents

Abstract.....	1
Introduction.....	2
Discussion.....	3
Chapter 1 – Bell Labs.....	3
Introduction to Bell Labs.....	3
UNIX.....	3
The ED Text editor.....	5
C Programming Language.....	6
The BC Calculator.....	8
Chapter 2- Apple Inc.....	9
Apple and the mouse.....	9
Mac OS.....	10
The Lightning Connector.....	11
M-Series Silicon Chips.....	11
File Systems.....	12
Conclusion.....	13
Project Evaluation.....	13
Literature Review.....	14
The Bell System Technical Journal.....	14
My own experiences with Bell Labs software.....	15
The C Programming Language (Book).....	15
Apple’s Own Documentation.....	16
Bibliography.....	16

Abstract

Apple Inc., as a company, is a household name. However, another organisation, Bell Laboratories, could be said to have had just as large, if not larger impact on the world of Computer Science and Consumer Computing. This Extended Project discusses several of the developments made at both companies, such as the UNIX operating system, the C Programming Language, the BC calculator, and the ED Text editor for Bell Labs, and the Popularisation of the Computer Mouse, the MacOS

operating system, the lightning connector and the M1 and M2 silicon chips for Apple Inc.

While this project has not succeeded in creating a full account of the hundreds of developments made by both companies, it aims to collate several of the most important ones for the reader's consideration. This project concludes that, while both companies have had a great deal of impact on their respective fields, the evidence that this project presents points towards Bell Laboratories being the slightly more influential company. The intended audience for this project is anyone with any interest in the use of computers, or anyone interested in knowing more about the history and the inner workings of the devices they use on a daily basis. It is not necessarily intended for subject experts, and I have tried where possible to use language and terminology that would be somewhat accessible to someone without any degree of expertise in computer science.

Introduction

This extended project is about which company has had the greatest effect on the field of Computer Science and the field of consumer computing: Apple Incorporated or Bell Laboratories. Apple Inc. is a significant multinational technology company with a market cap of 2.95 trillion USD as of November 2023. They manufacture large quantities of consumer-focused hardware including tablets, smartphones, and desktop computers, and develop software focused on a seamless user experience (UX/UI), such as the OSX operating system (OS) and application software such as Terminal (A terminal emulator designed for software developers and system administrators). Bell Labs is a research and scientific development company owned by Nokia. Previously known as AT&T Bell Laboratories, they acted as the research and development (R&D) division for AT&T (American Telephone and Telegraph company), presiding over the early stages of the field of Computer Science, and working at the forefront of electronic communications and technology.

Computer Science is a scientific field concerned with the inner workings of electronic computers. This includes topics such as operating systems (the software that runs on a computer to allow other programs to run, notable examples being Microsoft's Windows or Google's Android), programming (writing instructions for a computer), networking (connecting computers together to share hardware and information in the most efficient manner possible), and formal logic (a formalisation of the logic that humans apply to situations on a day-to-day basis).

In this extended project, "consumer computing" refers to the computers sold to the average buyer, and the experience that those computers give to those users. In this case, this also includes the use of computers by businesses, governments, academic organisations and other non-computer science focused organisations.

I have chosen this topic for several reasons: firstly, I am working towards an A-level in Computer science, and secondly because in my own work, I have used a great deal of the technology developed by Bell Labs. In my opinion, it has proved to be extremely useful, in many cases becoming the de-facto industry standard for that particular task. I have also used some (albeit less) Apple technology in my work as a musician: the recording studio that I have used in the past uses almost exclusively Apple products: both hardware and software. Furthermore, I am deeply interested in the field of computer science, and plan on continuing with it into university, perhaps pursuing a qualification in either Computer Science, or a related field.

In my research for this extended project, I have consulted numerous sources. First, I looked for scientific papers (mainly using Google Scholar), that discuss different developments that the two companies have made. This includes developments such as the C programming language for Bell Labs, and the Swift programming language for Apple Inc. I then looked online for subject experts in the concerned fields, and found the channel "Computerphile" on YouTube. This channel has hundreds of videos made by experts in the fields of computer science, such as Professor David

Brailsford, Dr Steve Bagley and Dr Michael Pound of the University of Nottingham. In these videos, they extensively mention both the developments made at Bell Labs and Apple Inc., and they have been a vital source of information and opinion for this extended project.

Furthermore, I have done my own experimentation with some of the more modern, Bell-Labs inspired tools, including the daily use of the GNU/Linux Operating system for over a year. I have also read books written by those at Bell Labs, such as “The C Programming Language” by Ritchie, Dennis and Kerninghan. Additionally, I have also looked at the popularity and design of a selection of the most widely-used Apple products, such as the iPhone, iPad and iMac. Similarly, I have looked at the popularity of some of the developments made at Bell Labs, such UNIX-based operating systems and the C Programming Language. I have also consulted the Bell System Technical Journal, a primary source published at Bell Labs, especially the edition from July-August 1978, which talks in depth about the UNIX operating system, the C programming language, and the “UNIX philosophy”. I have also looked at some of the technical documentation for Apple products, including the literature that they have available for consumers and developers online.

Chapter one will discuss the developments made by Bell Labs, and Chapter two will discuss the developments made by Apple Inc. Then, in the conclusion of this project, I am going to directly compare and contrast the two companies, and talk about the influence that the two companies have had historically. I will also talk about how the developments of both of the companies are used today, and draw a conclusion as to which one has been more influential. Of course, it is impossible to tell that with any degree of objectivity, but I can discuss links between the two companies (for example, the fact that some developments made at Apple may have been impossible without the work done at Bell Labs).

Discussion

Chapter 1 – Bell Labs

Introduction to Bell Labs

Bell Laboratories first came about as a group in 1924, when around 4000 engineers and scientists joined forces, in order to fully dedicate themselves to the research and development of communication systems (Nokia Bell Labs. (Date unknown) History <https://www.bell-labs.com/about/history/> Date accessed 12/12/2023.). Over the years, they have been responsible for myriad developments in the world of communications, telephony and later computer science, including the C programming language and the UNIX operating system. This Extended project will focus on their Computer Science related developments, and how they relate to the field of consumer computing.

UNIX

UNIX is a multi-user, time sharing operating system that was developed at Bell Laboratories. An operating system (OS) is the set of software products that jointly controls the system resources and the processes using these resources on a computer system (Oxford Dictionary of Computer Science Seventh edition (2016) page 383), such as Windows (an operating system for desktop computers made by Microsoft), or Android (an operating system for mobile devices made by Google.) In the case of an operating system, multi-user means that more than one person can use it at any one time. Windows, for example, is not a multi-user OS, because while it can have multiple user accounts (more than one person can use the computer over a given period), only one person can use the computer at any one time (barring esoteric methods such as having a computer run 2 copies of Windows independently of each other, in which case each copy is still only supporting one user.)

These multi-user operating systems were of particular importance in the late 1970s, when computer

resources were rare and expensive, and one physical computer would often have to serve an entire building or company. In this case, time-sharing means something very similar to multi-user, in that it is intended to be used by more than one person at any given time, allowing it to “share” its time between multiple people who all want to use its resources. At the time, multi-user operating systems were the norm because it was economically unviable for a company to buy a computer for each employee. Furthermore, even if money was no object, at the time computers took up orders of magnitude more room than they do today, so having one per user would be inconvenient as well as prohibitively expensive.

Sadly, computer users of today cannot emulate the fully authentic UNIX operating system, because it is a piece of copyrighted material and would no longer work on the computer architectures of today without significant edits. However, many operating systems today are “UNIX-Based”, and therefore share many of their underlying characteristics with the original UNIX. The most obvious example of a UNIX-based operating system is GNU/Linux, and by both using this, and restricting ourselves to a small subset of the programs available today (as well as tweaking some minor settings), we can get a command-line interface (CLI) that functions in a very similar way to the UNIX of 1978.¹

```
root@EPQ:~/EPQ# pwd
/root/EPQ
root@EPQ:~/EPQ# ls
file1 file2 file3
root@EPQ:~/EPQ# touch file4
root@EPQ:~/EPQ# ls
file1 file2 file3 file4
root@EPQ:~/EPQ# cd ..
root@EPQ:~# ls
EPQ
root@EPQ:~# cd EPQ
root@EPQ:~/EPQ# ls
file1 file2 file3 file4
root@EPQ:~/EPQ# rm file4
root@EPQ:~/EPQ# ls
file1 file2 file3
root@EPQ:~/EPQ# mkdir newdir
root@EPQ:~/EPQ# mv file3 newdir
root@EPQ:~/EPQ# ls
file1 file2 newdir
root@EPQ:~/EPQ# cd newdir
root@EPQ:~/EPQ/newdir# ls
file3
root@EPQ:~/EPQ/newdir# █
```

As you can see, the concept of “files” within a UNIX-Based OS is a very important one. In fact, the `/proc/` folder at the root of the filesystem is made up of several “virtual files”, which when read (for example with the “`cat`” command, are generated on-the-fly by the OS, showing information about the current state of the computer.

¹This can be reproduced much more easily by using a Virtual Machine rather than installing each operating system that you wish to test.

<pre> root@EPQ:~/EPQ# cat /proc/meminfo MemTotal: 2014512 kB MemFree: 262632 kB MemAvailable: 950488 kB </pre>	<pre> root@EPQ:~/EPQ# cat /proc/meminfo MemTotal: 2014512 kB MemFree: 267220 kB MemAvailable: 955060 kB </pre>
---	---

Furthermore, physical devices (sometimes known as “block devices”) are also listed as files, in the /dev/ directory. For example, here, /dev/sda1 corresponds to the first partition of the first-attached block device.

```

root@EPQ:/# cd dev
root@EPQ:/dev# ls
autofs          hwrng          ppp            tty            tty27          tty46          tty8           vcsa2
block           initctl       psaux         tty0           tty28          tty47          tty9           vcsa3
bsg             input         ptmx          tty1           tty29          tty48          ttyS0          vcsa4
btrfs-control  kmsg          pts           tty10          tty3           tty49          ttyS1          vcsa5
bus             log           random        tty11          tty30          tty5           ttyS2          vcsa6
cdrom           loop0         rfkill        tty12          tty31          tty50          ttyS3          vcsu
char            loop1         rtc           tty13          tty32          tty51          uhid           vcsu1
console         loop2         rtc0          tty14          tty33          tty52          uinput        vcsu2
core           loop3         sda           tty15          tty34          tty53          urandom        vcsu3
cpu            loop4         sda1          tty16          tty35          tty54          userfaultfd    vcsu4
cpu_dma_latency loop5         sda2          tty17          tty36          tty55          vboxguest      vcsu5
cuse           loop6         sda5          tty18          tty37          tty56          vboxuser       vcsu6
disk           loop7         sg0           tty19          tty38          tty57          vcs            vfio
dri            loop-control  sg1           tty2           tty39          tty58          vcs1           vga_arbiter
fb0            mapper        shm           tty20          tty4           tty59          vcs2           vhci
fd             mem           snapshot      tty21          tty40          tty6           vcs3           vhost-net
full           mqueue       snd           tty22          tty41          tty60          vcs4           vhost-vsock
fuse           net           sr0           tty23          tty42          tty61          vcs5           zero
hidraw0        null          stderr        tty24          tty43          tty62          vcs6
hpet           nvram         stdin         tty25          tty44          tty63          vcsa
hugepages      port          stdout        tty26          tty45          tty7           vcsa1

```

The “SD” used to mean SCSI (small computer systems interface) device, but now means any “block device”, such as a USB stick or floppy disk. This is a very simple example, and within the UNIX ecosystem there are many examples of files being used in rather unusual ways, such as the “/dev/zero” file, which is just a binary file containing a stream of binary zeros. Practically, this can be used to “zero a disk”, a process in which all the space on a magnetic disk is overwritten with “0”, ideally multiple times, to prevent data forensics experts from being able to read the data that the disk previously contained. It could also be done much faster with a physical stand-alone demagnetizer; however, this would be costly, inconvenient and would almost certainly significantly shorten the lifespan of the magnetic disk.

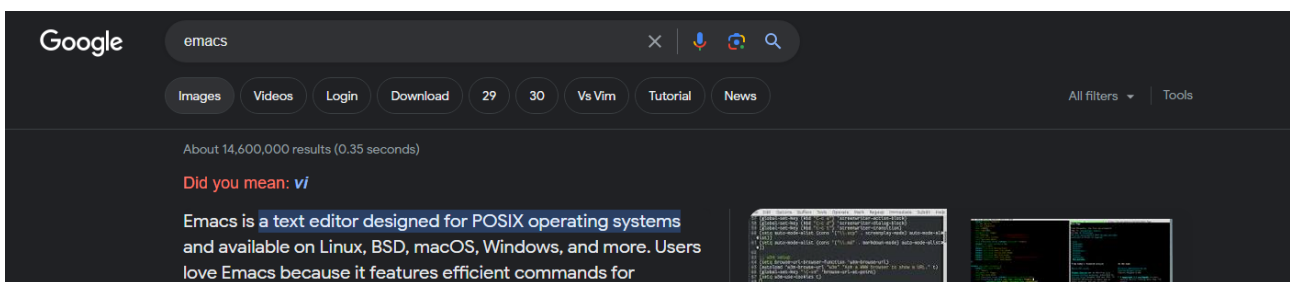
The ED Text editor

In a time before the advent of computer monitors, it was in the interests of software designers to make their software take up as little (vertical) screen space as possible. This is because UNIX (and therefore the “ED” editor) was designed to be used from a teletype, a machine which would produce computer output on a roll of paper, much like a typewriter), and would allow you to make inputs to the computer with a physical, typewriter-like keyboard. Therefore, line-based text editors, which only showed the user a line of the file at once, were the obvious way to go. To computer users today, ED might seem cumbersome and difficult to use, however with the alternative being

slowly and painstakingly re-drawing the entire file each time it “fell off” the top of the paper, it was the only real option.

The ED editor was later succeeded by “Vi”, standing for “visual”. It was called this because it displays a portion of the content of the file on screen, making it significantly easier to use. It also has a cursor- you can “be” at different points in the file. Vi is a modal editor: the user jumps in and out of several different modes while editing. There is “insert mode”, in which the keys you press on the keyboard simply get written to the file (or “buffer”, technically they aren’t written yet), “command mode” which allows you to use ED-like commands for complex, “batch-style” tasks. The most significant mode, however, is “normal mode”, in which each key on the keyboard is mapped to a specific shortcut. For example, the “h”, “j”, “k”, and “l” keys move the cursor left, down, up and right respectively: these are by far the most used commands in any editor (moving between different parts of the file), and they have stood the test of time, still being used in some editors today. These key-bindings keep the keys on the “home row”, allegedly being more ergonomic, and reducing the need for modifier keys: holding down something like “control” for long periods of time.

The rivalry between vi and EMACS (another popular editor at the time), or the “editor wars”, became something of a cultural phenomenon, with even the Google search results for “vi” correcting the user to “Did you mean “EMACS”, and vice versa.



After “vi”, came “vim”, designed to compliment the vi experience. It had a “vi compatibility mode”, making it act identically to vi, and it was “portable”, meaning that it worked on other, non-UNIX operating systems. Furthermore, it added in several other features, such as unlimited undo (the ability to roll back an infinite number of actions), syntax highlighting (certain key words being highlighted, which is especially useful for programmers), and “time travel”: the ability to revert a file to its state an arbitrary number of minutes ago.

Today, many common text editors have plug-ins or extensions that enable the vi-style key binds. Visual Studio Code, Microsoft’s IDE (Integrated development environment) has the “vscodevim” extension(<https://marketplace.visualstudio.com/items?itemName=vscodevim.vim>), for “Vim emulation in Visual Studio Code”, Sublime Text has “Vintage Mode” (<https://www.sublimetext.com/docs/vintage.html>) which does the same thing, and EMACS even has “Evil mode”- “an extensible vi layer for Emacs” (<https://www.emacswiki.org/emacs/Evil>).

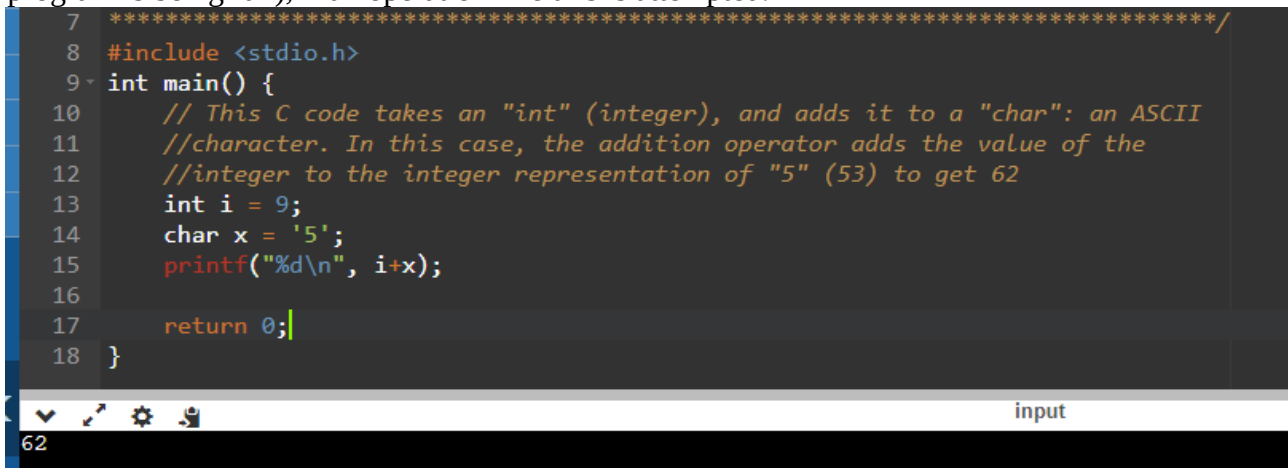
C Programming Language

A programming language is a way of expressing instructions to a computer in a format that humans can understand. Programming languages are generally either “interpreted” or “compiled”. An interpreted language, such as Python, runs with an interpreter, which needs to be installed on the computer it is ran on, and executes each code statement line-by-line. All that is required to run an interpreted language is the source code, and the source code in most cases must be freely available and modifiable in order for the program to be used. A compiled language, such as C, is “compiled” using a “compiler” such as GCC (Gnu Compiler Collection) into something called a binary. That binary can then be run on any instance of the system it has been compiled for, with no regard for

what software it has installed. Software like Microsoft Word and Adobe Photoshop are compiled: this can be seen by the fact that when you install them, only a ".exe" (an executable, binary file) is installed onto the computer.

The C programming language is perhaps one of the most famous programming languages of all time. It is taught in many university "introduction to programming modules", and it is considered by many to be a necessity to learn if a developer is ever to be truly comfortable with their system. C is a compiled language, which essentially means that you write the program (In C), and then use a compiler (such as the GNU C Compiler- GCC which is an executable usually ran from the command line) to translate it into machine code that the computer can understand and run. The compiler optimizes the program in certain ways, such as trying to use branch-less programming which is faster for the Processor to run. C is a statically typed programming language, which means that each variable has a set type when the program is compiled. For example, "4" would be an integer (a whole number), and "4.323" would be a float (a number with a "floating point": one with a decimal component). This helps to deal with issues such as the question of what happens if the word "55" is added to the number "55": is it "5555", because the second variable has been added to the end, or is it "110", because two numbers are being added together. The C language deals with this by having a set type for each variable, and throwing an error (at compile time, not when the program is being run), if an operation like this is attempted.

```
7 *****/
8 #include <stdio.h>
9 int main() {
10     // This C code takes an "int" (integer), and adds it to a "char": an ASCII
11     //character. In this case, the addition operator adds the value of the
12     //integer to the integer representation of "5" (53) to get 62
13     int i = 9;
14     char x = '5';
15     printf("%d\n", i+x);
16
17     return 0;|
18 }
```



C is, in some ways, a very simple language, with only 32 keywords (words that are treated specially by the compiler, such as "if" to check if a statement is true). In contrast to this, a more modern language such as C++, which has 95. Each keyword is another thing that a programmer needs to remember and understand to be completely competent with the language, and therefore in some ways, C could be considered easier to learn. On the other hand, C has potential to be extremely complex. For example, C allows direct access and manipulation of the computer's memory, and things like arrays and strings can often get extremely complicated. If a process (written in C) tries to access a piece of memory that it should not have access to, then this can be a significant issue: not just in the sense that it can cause the program to crash, but also in the sense that it can be a security risk. If a program can read a piece of memory "belonging" to another program, then theoretically it could be used to read (and edit) sensitive information "belonging" to that program. These so-called "memory leaks" can also cause the user's computer's memory to become full, causing it to resort to using swap space, which is extremely inefficient. This can cause a program to either work slowly, or not work at all, which is bad for both the user and the developer.

Some languages such as Python, do not allow this direct memory access, and this stops developers from "shooting themselves in the foot" by writing "unsafe" programs. However, there are some situations when direct memory access is appropriate, and ends up being the most efficient way to complete a task, which makes C a very powerful language. Furthermore, C is good for learning how a computer works, because it allows those learning to program to make those mistakes and gain a

deeper understanding of how the computer handles memory, and data structures. An array is a very simple example of this: a group of pieces of data all of the same size that are stored in a "contiguous" space in memory (all next to each other). This means that to read a given item in an array, you just need to take the address of the first item in the array, and add on the size of the items in the array multiplied by one less than the index of the item you want to access. This seems complicated, and indeed it is, but in usual use, this level of depth is abstracted away from the user, and they only need to worry about the fact that the indexes in arrays start at 0 rather than 1. Doing this sort of "pointer arithmetic" helps learners to better understand these data structures, which is a key part of the entire field of Programming and Computer Science.

```
8 #include <stdio.h>
9 int main() {
10     int myarray[5] = {1,2,3,4,5}; //Declares an array of 5 items
11     printf("%d\n", myarray[0]); //outputs the first item of the array (item 0)
12     printf("%d\n", myarray[4]); //Outputs the last item of the array (item 4)
13     //Now, we try to access items outside of the array
14     printf("Printing items outside of the array\n");
15     printf("%d\n", myarray[7]);
16     printf("%d\n", myarray[8]);
17     printf("%d\n", myarray[9]);
18     printf("As you can see, this prints numbers unrelated to the array");
19 }
input
1
5
Printing items outside of the array
-663818528
1
0
As you can see, this prints numbers unrelated to the array
...Program finished with exit code 0
Press ENTER to exit console.
```

Many very influential projects have been written in the C language. For example, the Linux kernel (the core of the operating system) is almost entirely written in C (at the time of writing). So too, are the Nginx and Apache web servers, which power several of the biggest websites in the world. Furthermore, the GNU Compiler Collection is written in C, and while this is slightly defining C in terms of itself, GCC is in use for several different languages including Fortran and C++. All of the Microsoft office programs (MS Word, PowerPoint, Excel) are written in C, as well as LLVM: a compiler framework in use for some of the most popular languages and it is considered by many to be the go-to language for writing high-performance programs.

The BC Calculator

According to the POSIX Standard: "The BC utility shall implement an arbitrary precision calculator. It shall take input from any files given, then read from the standard input. If the standard input and standard output to bc are attached to a terminal, the invocation of BC shall be considered to be interactive, causing behavioural constraints described in the following sections." For those using a computer with no GUI (Graphical User Interface), bc is a convenient, standardized way to use the processor to carry out basic calculations. Apart from the obvious, "2+2=4" questions, bc is also suited to more complex tasks. The "C-like" nature of bc allows the user to do things like define custom functions.

```
define f (x) {
    if (x <= 1) return (1);
    return (f(x-1) * x);
}
```

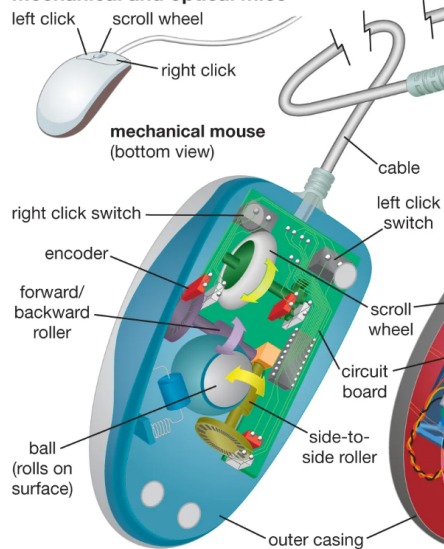

This is a recursive function, and with a few limitations, anything that you can do in C, you can do in BC. Furthermore, for those who are familiar with the syntax of C (which many are), it provides an accessible user interface, and a clear way to enter expressions and output their results.

Chapter 2- Apple Inc.

Apple inc. are a software and hardware company based in Cupertino, California. Established in 1976, (Steven Levy, Britannica (2024) <https://www.britannica.com/topic/Apple-Inc> (Date Accessed: 17/02/2024)) today they produce products such as the iPhone, iMac, and iPad. They have a market cap of \$3.02 Trillion as of the time of writing, making them the biggest company in the world by market cap. Like Bell Labs, they have also been responsible for several developments, particularly in the consumer computing market and the fields of user experience. This Extended project will focus on their Computer Science related developments, and how they relate to the field of consumer computing.

Apple and the mouse

Mechanical and optical mice



(Image: mechanical and optical mice Encyclopedia Britannica

<https://www.britannica.com/technology/mouse-computer-device#/media/1/395079/73815>)

The computer mouse, as we know it today, started off life as a mechanical device, utilising a rubber ball within a hemispherical housing. The modern mouse uses a much more modern optical sensor, and rather than relying on a physical ball moving around within the device, it uses a light sensor which detects reflections off of the surface by shining a laser at them.

(Britannica, T. Editors of Encyclopaedia. "mouse." Encyclopedia Britannica, October 30, 2023.

<https://www.britannica.com/technology/mouse-computer-device>. Date Accessed:17/02/2024)

Mice are often used in conjunction with a graphical user interface: the user is given a pointer that they are free to move around the screen, which they can then use to select and manipulate graphical objects (such as clicking on the start menu in the bottom left in Windows 10). This contributes to the “desktop metaphor” that many user interfaces try to implement: this is evident in the way that interfaces such as the one used in Windows 10 use a folder literally named “desktop” to indicate things that are available with little to no navigation.

Now, Apple inc. did not invent the computer mouse: that honour goes to SRI international: (US Patent Office, Patent No. 3541541 (1970):

<https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/3541541>) However, the Apple Mackintosh 128 was the first commercial device to popularize the use of the mouse with a graphical user interface (BBC News (2024), <https://www.bbc.com/future/article/20240123-the-apple-macintosh-was-first-released-40-years-ago-these-people-are-still-using-the->

[agingcomputers#:~:text=On%202024%20January%201984%2C%20the,Macintosh%20computer%20he%20ever%20bought](#) Date Accessed:17/02/2024). This kickstarted the widespread use of the mouse in consumer computing, and this in turn paved the way for mouse-focused software. This includes software such as Adobe Photoshop, which makes liberal use of the mouse in order to move different tools around the screen.

Mac OS

Like Bell Labs, Apple also developed their own operating system. At the time of writing, there is an apple operating system available for desktop computers, mobile phones, and tablets. It is much more prevalent than UNIX (and UNIX-based systems) in the consumer space, and when coupled with iOS (the operating system for the iPhone), it becomes apparent that 24.3% of the world's consumer computers are being run on apple software. The latest version of MacOS- Sonoma- has a fully featured desktop environment, and a range of useful features such as new screensavers², desktop widgets, and interaction with other apple devices such as the iPhone (Apple Inc., Official Apple Documentation, <https://www.apple.com/uk/macOS/sonoma/> Date Accessed:17/02/2024).

Built in to macOS are several examples of apple software. "Terminal" is a terminal emulator that aims to give a UNIX-like experience, tailored towards software developers and power users.(Apple Inc., Official Apple Documentation, <https://support.apple.com/en-gb/guide/terminal/welcome/mac> Date Accessed:17/02/2024). Terminal gives an easy way for users to run and design scripts to make their Apple device perform predetermined routines. It also allows users to connect to remote servers using technology such as SSH (Secure Shell) with a graphical interface (rather than the user typing in a specific shell command to invoke the SSH client), and this feature is useful for many software developers. This is because a common workflow in many tech companies is to have a "Development server" which stores the files that need to be worked on, and has enough computing power to build and run examples of the software. The developers then remotely access this server, and run commands on it. One major advantage of this is that developers can work completely from home, rather than needing to use specific hardware that might be elsewhere. Furthermore, the convenience of remote access in "Terminal" is helpful for website administrators, because it can allow them to make on-the-fly changes and optimizations to a web server.

The fact that remote access is built into the terminal itself, rather than being a separate piece of software (such as OpenSSH, a popular UNIX SSH client) (<https://www.openssh.com/>) is evidence of Apple's different design philosophy. Apple opts for one monolithic piece of software, rather than many small programs all designed to do exactly one thing. There are reasons that a user might want to use a separate SSH client, such as to be able to customize the encryption algorithms that it uses, or because the commands are familiar to them (Arch Linux Wiki (Date Unknown),https://wiki.archlinux.org/title/OpenSSH#Tips_and_tricks Date Accessed:17/02/2024). This design also means that "Terminal" is able to be less stripped-down than a UNIX-based terminal: if a UNIX terminal needed to be optimized for space, then a user could simply remove any unneeded software using a package manager (such as APT or Pacman), and continue using the terminal with no adverse effects. Because "Terminal" has remote server access built in, there is no way to be able to use "Terminal" without taking up the space needed for all of the remote access overhead. In reality, this is a spurious example: a zipped version of all of OpenSSH only runs to about 500Kb, which is not enough to make much of an impact on a modern storage drive, however it does speak to the design of the software, and how customizable it is possible to make it.

When compared to the UNIX terminal experience, there is little doubt that while MacOS provides a

² These are not screensavers in the strictest sense: they are designed more for aesthetic purposes rather than to "save" a monitor from burn in. Traditional screensavers are not necessary on the now common LCD monitors because the mechanism for burn-in to happen is not available, however on OLED monitors common on several Apple devices, it can be an issue, and a screensaver can be needed.

more streamlined and cohesive experience, it is significantly less customizable, and provides less ways for the user to optimize what software they run.

The Lightning Connector

Up until 2024, all iPhone, and the majority of iPad produced use a power connector called the Lightning connector (BBC News (2024)<https://www.bbc.co.uk/news/technology-58665809> Date Accessed: 17/02/2024). The lightning connector is a digital connector standard, that can transfer both power and data over the same connection. It can be inserted either way up, and the plug is made up of one solid, smooth piece of metal with several flat metal contacts on either side. Compared to micro USB, the lightning connector is more convenient to use, and more durable due to the fact that micro USB relies on metal pins that bend down when inserted into a device.

The lightning connector is able to provide more power than micro USB (12W rather than 9W), (Anker, 2023 <https://www.anker.com/blogs/cables/what-is-a-lightning-cable-ultimate-cable-guide>) which enables faster charging and allows more power-hungry peripherals to use it. It also provides data transfer speeds of 480Mbps³ (Apple Inc. (2024)<https://support.apple.com/en-us/109044>), which is matched by the maximum speed of USB 2.0 (the standard which micro USB is a part of) (Make Use Of (2023) <https://www.makeuseof.com/usb-c-data-transfer-speeds-how-fast-can-it-go/>). This means that, on USB 2.0 ports, which are still common on many computers⁴, if a connector that enabled higher data transfer speeds was used, then it would not provide any advantage, since the USB 2.0 port would act as a bottleneck, capping the speed at 480 Mbps.

In 2021, the EU (European Union) ruled that, in order to reduce E-waste (Electronic waste), all mobile phones and small electronic devices must move to USB-C: a new standard from the USB family. USB-C can also be inserted either way up, but provides technical specifications that vastly surpass both Lightning and Micro USB. USB-C can, in theory, reach transfer speeds of up to 80 Gbps (10 Gb/s), however in reality this is unlikely, because it relies on the cable being made to USB4 Gen4 standards, and on both devices having support for the latest standard built in. However, the use of USB-C does give manufacturers room to progress, and as the EU said, allows users to use their existing charging hardware when they buy a new device.

M-Series Silicon Chips

Nearly every computer contains a CPU (Central Processing Unit). Often referred to as the “Brains” of the computer, the CPU carries out the instructions laid out by computer programs, and handles tasks such as running the operating system and some programs. In most consumer desktop computers (rather than laptops), the CPU is a component like any other: it is inserted by the manufacturer into a socket on the motherboard, and can be swapped out for a different CPU, provided that it will fit. CPUs work by using transistors, which generate heat. Therefore, it is important for computer manufacturers to provide ways of adequately cooling the CPUs, to prevent “thermal throttling”. This can be done in several ways, but the most popular is by far air cooling: a heat sink (a large block of thermally-conductive metal, usually with fins to help dissipate the heat) will be bolted onto the IHS (Integrated Heat Spreader) of the CPU, and fans will be used to blow cool air through the heat sink to help carry off heat. Most desktop computers are designed from the ground up with this in mind, with some sort of a plan worked out for “airflow” through the case. This is the main reason that computers have fans at all: because the CPUs require constant active cooling.⁵

³ Mbps is megabits per second, so this is in effect 60 Mb/s (Megabytes per second).

⁴ Technically, if they are not coloured blue, then they are only rated for USB 2.0, although this standard is not always followed.

⁵ Some other components (such as PSUs, GPUs, and sometimes memory and storage) can require active cooling too, however usually only PSUs (Power Supply Units) contain fans in consumer computers, and the fans are built into

In 2020, Apple announced the release of the M1 Silicon chip. Created specifically for the Mac, it combines multiple components, which would ordinarily be inserted onto the motherboard individually, into one chip: the CPU, the GPU, and the Memory. The fact that these three crucial components were all now in the same place reduced data transfer speeds between the components: the instructions from the memory could get to where they needed to be to be processed quicker, so more could get done in less time. (Apple Inc., Apple Documentation (2020)<https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/> Date Accessed: 17/02/2024) The integrated chip was also easier to cool, because all the components that needed cooling were in one place, and could be actively cooled by one heat sink. This helps save money on manufacturing, because the dense, thermally conductive metals used to build heat sinks are expensive and difficult to shape. It could, in theory, also make the computers more reliable, because they need fewer moving parts (fans) which could fail. The M1 chip has been succeeded by the M2 Silicon, designed to be more of the same, with many of the same advantages.

Both the M1 and M2 chips use the ARM instruction set (a list of instructions that the CPU can carry out), rather than the more traditional x86_64 set, used by the majority of modern Intel and AMD desktop CPUs. The ARM instruction set is considered a type of RISC (Reduced Instruction Set Computing), because it has significantly less instructions than x86_64. This makes it more simple to write, however some would argue that it is more difficult to optimize, because it gives experienced developers less options. Furthermore, because it requires less transistors, the ARM architecture is usually more power-efficient, because it requires less transistors to enable all of its instructions. ARM Assembly (the language used to communicate with ARM chips at a low level), while simpler to write than x86 assembly, is much less standardized, and this can make code written for ARM chips less portable (less able to be run on a different computer than it was designed for.)

The ARM instruction set is usually used by mobile devices, and hobbyist devices such as the Raspberry Pi. At the time of writing, there is no official version of Windows for the ARM architecture, nor are there versions of many programs that will run on a non-x86 device. While this does mean that modern Mac computers are less versatile: a user cannot easily boot a non-apple Operating system on them and start using different programs, it does encourage developers to start building their programs for this new architecture. This is good, because it provides competition with the two major CPU manufacturers: Intel and AMD, and because it forces innovation onto program designers if they wish to deploy to Apple devices.

File Systems

File systems are standards that govern the way that files are stored on a device. They specify things like what metadata is stored about each file (such as the precision and method of storage of the “last accessed” date), and the partitioning scheme used on the drive (usually either GPT or MBR). FAT32, a popular file system for removable media, has certain limitations, such as the fact that it can only support files up to 4Gb. (Microsoft Corporation, Microsoft FAT Specification (2005), Page 27) This means that it is unable to store most movies at a 4k resolution without resorting to splitting the file in two⁶. There are many other, more modern file systems such as NTFS from Microsoft, and Apple use their own as well.

APFS, the Apple file system, is a proprietary file system used by Apple. It is only natively supported on Mac OS, and is the default file system on all Mac computers. There are four main versions of APFS: (Apple Inc. (Date Unknown) <https://support.apple.com/en-gb/guide/disk-utility/dsku19ed921c/mac> Date Accessed: 17/02/2024)

the unit.

⁶It is possible to use software to have a file technically be split into multiple pieces, but act as one file. However, since the only advantage of using a file system such as FAT is compatibility, this is rarely the best course of action.

- Regular APFS
- Encrypted APFS, which is simply an encrypted APFS volume
- Case-Sensitive APFS, where file and folder names are case-sensitive: “FILE” would be a different file from “File”, which would be different from “file”.
- Case-Sensitive, Encrypted APFS, which is simply Case-Sensitive APFS, but encrypted.

The Apple File System does provide some advantages over some others, especially FAT as mentioned earlier, however its proprietary nature and Apple’s refusal to use an already existing file system, as well as insisting that all drives with MacOS running on them must be formatted with APFS mean that overall, APFS makes computers more difficult for consumers to use.

Conclusion

In conclusion, I would say that, while both companies have had a great deal of impact upon the world of consumer computing and computer science, Bell Labs have had a slightly larger impact than Apple Inc. This is because many of their developments are widely used in all computers today, not just those made by a specific company, and because many of their developments have paved the way for Apple to make some of theirs. Furthermore, the impact of Apple Inc. as a company on the world of computer science has been lessened by the fact that many of their products are prohibitively expensive, making their developments less easily accessible for consumers.

While Apple Inc. is certainly more of a household name, many of the modern-day infrastructure that we rely on today would not exist without the use of UNIX-based operating systems and the C programming language. While the developments of Apple have certainly been influential, many of them are superficial and have alternatives that can be used. Furthermore, many of the hardware and software products that are currently produced by Apple Inc. could be said by some to be actively slowing the development of new technology. One such example of this is Apple’s reluctance to forgo the use of the lightning connector on their devices until being forced to by the EU (European Union).

Bell Laboratories are, by comparison, nowhere near as famous. However, much of the modern computing world is built on their developments. The C programming language is used to create a great deal of the software used in our society’s most important applications, and UNIX-based operating systems are the standard for nearly all enterprise-grade hardware and scientific computing to this day. The key bindings from the ED text editor endure even now, in editors such as Vim, or in the numerous extensions for nearly every other piece of text-editing software.

Both companies have made many more developments that I have not had time to write about. This is because it would be beyond the scope of an essay of this length to go into details of every development made by these two massively influential companies. Bell Labs pioneered the first lasers and transistors (among other things), and Apple have produced a whole host of professional-grade software such as Logic Pro and Final Cut. This project aims to present a few of the most influential developments made by each company, and if the full extent of each company’s contribution to Computer Science had been evaluated, then the conclusion reached at the end of this project may have been different.

Project Evaluation

Completing this project has taught me a lot about conducting research, critical analysis of sources, and writing a formal paper. It has been very interesting to look into the different ways that two companies design products, and how the use of computers has changed over time, from the time of Bell Labs to modern day. I have also learned a lot more about the general field of Computer Science, which is sure to help provide a good basis for my learning in A-level Computer Science. I have learned a lot of time management and project planning skills throughout the creation of this

Extended Project, and I have no doubt that the skills that I have learned here will come in useful to me during the rest of my time at Sixth Form, during my time at university, and during my future career. Furthermore, I have learned a lot that I wouldn't usually touch on with the A-levels that I have chosen: for example writing a paper, and citing sources which are not usually addressed in more science and theory-focused qualifications.

While making this project, I have been especially pleased with the quality of information that I was able to present on the UNIX operating system. It was brilliant to be able to create my own examples to show the reader, and I think that it helped to illustrate a potential workflow of a user of that operating system rather than just describing it. Making those examples also taught me a lot about using Virtual Machines, and using UNIX-based operating systems in general, as well as the skills of being able to construct an authentic, period-accurate software environment in order to best emulate the computers of the time.

If I were to redo this project with the benefit of the experience gleaned this time round, I would either narrow my focus further, or choose a different format to present the information. I feel that this would allow me to present a wider range of developments of either company, as well as allowing me to investigate each development in more detail. This would allow me evaluate the precise impact that each development has had, and create and show more examples of that development in use. I would also try to conduct more primary research, perhaps using a survey of my peers to find out the prevalence of the two companies' developments in the context of my school, or reaching out to experts in both subjects, and those who use the software on a daily basis, in order to find out more about the usefulness of each development.

Literature Review

The Bell System Technical Journal

Throughout this project, a large amount of the information concerning the UNIX operating system and the C programming language has been from the Bell System Technical Journal, particularly the edition from July-August 1978. The Bell system technical journal is the internal journal for Bell Labs, meaning that it has a high degree of reliability: it is an accurate presentation of the knowledge available to them at the time. In terms of bias, while it is made by those with a vested interest in Bell Labs products, it was never intended to be shown to the wider public, and therefore it is void of any overt bias against other products. The aim of the creation of this journal was not to convince or persuade anyone of anything, rather to inform in the most efficient manner possible. It aims to be objective, and does not fail to mention issues or shortcomings with any of the products that it mentions.

The Bell System Technical Journal may be less reliable than something published closer to the present day, because of the fact that it was not fact checked by anyone outside of Bell Labs. This means that there may be issues with some of the material within it that some of the Bell Labs employees were unable to see, but would have been caught had it been more available for review.

It is influenced by those at Bell Labs at the time, which makes it a very useful primary source, and those who wrote it have a high degree of expertise in the subject (often they were the ones who developed the product, giving them a deep understanding of how it functioned). The Bell System Technical Journal has been a valuable piece of information throughout this project, and has been concise and factual enough to use several times, both in research and in referencing.

My own experiences with Bell Labs software

Throughout this project, I have had the opportunity to try out and use a great deal of software. I

have developed software using the C programming language, and I have used a UNIX-based operating system daily for several months. This has enabled me to see the user experience of using Bell Labs products, and how they work on a deep level. There are many variants of UNIX-based operating systems out there, and I deliberately chose a “Do It Yourself” distribution, so that I could learn more about the inner workings of my system. Things I have done using this system include creating a backup system (to copy important files to a different location) using shell scripting, and running virtual machines (an operating system inside an operating system) using KVM/QEMU. This has allowed me to gain a deeper understanding of how computers as a whole, and particularly UNIX-based systems run on a low level, and showed me why Bell Labs products can be so useful.

During my time using Bell Labs software, the built-in documentation, called manual pages, has been extremely helpful to me. Often referencing the POSIX standard, these manual pages feature in-depth guides on how to use a whole host of Bell Labs products. Furthermore, there is a large amount of free and open-source information concerning Bell Labs software, information which a user almost always needs to consult at some point in order to understand and solve an issue. Websites such as the Arch Wiki (a Wikipedia-style repository of information centred around the Arch Linux distribution), and the Linux Kernel mailing list (an email list designed to help kernel users and developers discover and solve issues with the Linux kernel, as well as implementing new features.)

I understand that my experiences with Bell Labs software will not necessarily reflect the experiences that the general public would have if they were to use it. My use cases for computers in general is quite unusual, and a lot of the software that I have used during the course of this project has had an extremely steep learning curve. Many users would find it annoying, and perhaps almost unusable, to have, for example, an operating system that needed to be manually updated from the command line. Furthermore, I have no idea what any of the software that I have tested would work like for someone with a different use case, such as video editing or audio production. Therefore, my personal experiences with Bell Labs software are biased towards the software, perhaps showing it in a more positive light than it deserves.

[The C Programming Language \(Book\)](#)

The C Programming Language, by Brian W. Kernighan and Dennis M. Ritchie has been an invaluable resource in the research for this project. In the preface, it states that “C is not a big language, and it is not well served by a big book.” This methodology means that the book is concise enough to read for a project like this, while still being informative enough to get all the information that I need. The C programming Language (the book) was written, in part, by Dennis Ritchie, one of the creators of the actual C programming language. This makes it extremely reliable, because no one else knows the ins and outs of a product like that quite like its creator. The book is also able to go into detail about the thought process behind a lot of the features in the language, such as the fact that arrays start at zero, or the fact that structures were intended to emulate things like “payroll records.”

“The C programming language” is obviously biased towards the language it refers to, thanks in large part to the fact that one of the main authors played a part in its development. Furthermore, because it was written in 1988, it fails to address many of the questions that a more modern audience may have. It completely fails to mention modern networking at all, and spends all of chapter 8 talking about the UNIX system. There is no mention of how to develop a C program for Windows, MacOS or any mobile platform, and while this is of course not the fault of the author, it still makes the book less useful in the modern day.

In a way, The C Programming language can be seen as a primary source, due to the fact that it was written by those who were actually there at the time. The book was written in order to inform the

public, and not to persuade anyone of anything, and this makes it more reliable and less biased than it would otherwise be. It is influenced heavily by the philosophy of those working at Bell Labs at the time, and has been an extremely valuable resource while working on this project. While I have not sourced it directly, it acted as a valuable jumping-off-point for my own experimentation, and gave me a great deal of ideas of features to show in this project.

Apple's Own Documentation

The main source that I have used to find out about Apple products, especially the technical details needed for comparisons with other software, has been Apple's own documentation. The obvious issue with using this as a source is bias: anything made by the Apple company itself is going to be heavily biased towards Apple product. While I highly doubt that anyone at Apple would introduce factual inaccuracies deliberately, there is no question that issues with Apple software and hardware may be overlooked, or ignored in the interests of making the company look better in the public eye.

The Apple documentation, in my experience, has been disjointed, hard to navigate, and lacking in structure. There is a great deal of promotional information, using Apple-created buzzwords such as "Retina Display", and a shockingly low amount of actual, end-user focused documentation. Furthermore, there is a clear lack of service documentation- documentation designed for those repairing an Apple product- due to Apple's negative attitude towards third-party, unapproved repair shops. This is a case of Apple making it deliberately harder to use their products, and it casts a shadow of doubt upon their entire documentation process. It is hard to believe that the people behind the documentation are writing it solely for the purpose of helping consumers, when they refuse to give out repair guides to anyone except their own shops.

The Apple documentation has been somewhat useful in verifying information about Apple products, especially hardware. However, it's reliability is questionable, and it has a very clear bias. Therefore, while I have consulted it many times during the creation of this project, it has only been somewhat useful, and I have only referenced it when it was clearly correct, such as for numerical statistics and product announcements.

Bibliography

- Nokia Bell Labs Website: History page (Nokia Bell Labs. (Date unknown) History <https://www.bell-labs.com/about/history/> Date accessed 12/12/2023)
- Oxford Dictionary of Computer Science Seventh Edition (2016) page 383
- Britannica (2024), Steven Levy: <https://www.britannica.com/topic/Apple-Inc> Date Accessed: 17/02/2024
- Image of mechanical and optical mice, Britannica (2024) (<https://www.britannica.com/technology/mouse-computer-device#/media/1/395079/73815> Date Accessed: 17/02/2024)
- Britannica (2024), (<https://www.britannica.com/technology/mouse-computer-device>. Date Accessed:17/02/2024)
- (US Patent Office, Patent No. 3541541 (1970): <https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/3541541>, Date Accessed:17/02/2024)
- (BBC News (2024) (<https://www.bbc.com/future/article/20240123-the-apple-macintosh-was-first-released-40-years-ago-these-people-are-still-using-the-aging-computers#:~:text=On%20January%201984%2C%20the,Macintosh%20computer%20he%20ever%20bought> Date Accessed:17/02/2024, <https://www.bbc.co.uk/news/technology-58665809> Date Accessed: 17/02/2024)
- (Apple Inc., Official Apple Documentation (<https://www.apple.com/uk/macOS/sonoma/> Date

Accessed:17/02/2024, <https://support.apple.com/en-gb/guide/terminal/welcome/mac> Date
Accessed:17/02/2024, <https://support.apple.com/en-us/109044>, Date Accessed: 20/02/2024,
<https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/> Date Accessed:
17/02/2024, <https://support.apple.com/en-gb/guide/disk-utility/dsku19ed921c/mac> Date
Accessed: 17/02/2024)

- Arch Linux Wiki (https://wiki.archlinux.org/title/OpenSSH#Tips_and_tricks Date
Accessed:17/02/2024).
- Anker (2023) (<https://www.anker.com/blogs/cables/what-is-a-lightning-cable-ultimate-cable-guide> Date Accessed: 19/02/2024)
- (Make Use Of (2023))<https://www.makeuseof.com/usb-c-data-transfer-speeds-how-fast-can-it-go/>
- Make Use Of (2023) (<https://www.makeuseof.com/usb-c-data-transfer-speeds-how-fast-can-it-go/> Date Accessed: 05/02/2024)
- Microsoft Corporation, Microsoft FAT Specification (2005), Page 27