**Write briefly on the UNIX operating system especially the Linux flavor**

**UNIX and Linux**

UNIX is an operating system (an operating system is a software that acts like an interface between the computer's hardware and other system software). It was created in the 1960s by Ken Thompson, Dennis Ritchie and others, the UNIX operating system was first written in assembly but later rewritten in c, the UNIX operating system is composed of three main parts, the kernel, the shell and the programs.

**Kernel**

The kernel is the lowest level of the operating system, even sometimes referred to as the hub of the operating system, this the part of the operating system that allocates time and memory to programs, manages files, input/output devices and other critical functions required of the operating system.

**Shell**

This is an environment that acts like an interface between the user and the kernel, whenever the user wants to make a system call to the kernel to perform some operation this is done through the shell. Upon opening the shell, the shell prompts you to enter commands through the % character. Shells are customizable and a user can run multiple shells on the same machine at the same time but all the requests made by all the multiple shells is carried out by the same kernel.

**Programs**

This consists of all the system applications that the user uses.

*UNIX offers additional benefits and properties like;*

* Portability: Ever since the UNIX operating system was written in c this made it possible for the software to be copied and run on multiple machines without recompiling.
* Multi-tasking: The Unix operating system can handle the running of multiple processes simultaneously (a process is an executing program).
* Multi-user functionality: it allows for multiple users to log into the operating system.
* It also stores files as plain text as opposed to a binary format used on other operating system.

**The directory structure**

UNIX files are arranges in a tree like structure (more like inverted tree) which starts from the root (i.e. /) and extends to various sub-directories. This file system makes it easier to organize, search and manage files in UNIX.

**Linux**

Linux just like UNIX is an open source operating system created by Linus Torvalds in 1991. It was built using the UNIX philosophy but shares not of its source code. Linux is light weight, flexible scalable and secure which makes it popular across many devices from desktops, to servers and even super computers but its learning curve is steeper than other operating systems like windows or mac os. Because of its open-source nature there are many distributions (called distros) of Linux, some examples are red hat, Ubuntu, fedora, and even android. These distributions are customized to solve a specific type of problem or for a certain kind of people.

**Software functional requirements**

Before I start defining software functional requirements, I want to define what ***requirements*** is. Using the dictionary, requirements is defined as something that is needed or wanted, something that is compulsory, a necessary condition.

Hence gaining from that definition, we can define that software functional requirements is a documentation that states the necessary conditions a software application must meet. This is usually written by the developer, a business analyst or someone or even a group of people that is on the development team before the actual development of the software application. While writing a functional requirement the requirements should be short, clear, concise and straight to the point so as not to confuse anyone on the developing team.

**Advantages of software functional requirements**

1. the most obvious advantage is that it provides a clear goal of what the software is meant to do.

2. It allows you to spot errors in the design beforehand which makes it easy and cheap to fix.

3. It allows to verify if the software is doing what it’s meant to be doing

4. provides a cost estimate for the customers, likewise it allows developers to estimate the time they'll need to complete the application

5. Probably the most important is that it allows everyone on the engineering team to be carried along and allows them to understand the big picture as well as the things needed to be done.

**Disadvantages of software functional requirements**

After seeing the advantages, the main disadvantage of not writing a good software functional requirement or not writing any one at all!!! Should be obvious. That is leads to a poorly defined objective, which makes developers to either over engineer or under engineer the software leading to excess costs and unnecessary delays. Also since the objectives of the applications is not clear, it also stagnates the productivity of the development team as a whole since there won't be clear communication (i.e. there would be confusion) between them.

Hence having discussed all these, if you want to develop a website, your functional requirements might look like this;

* The website must send an email to whomever logged in
* The website must be able to integrate with the database

If the website uses a ML model at the background, you can add that

* The Machine learning model must be able to extract all the necessary information from the website
* The Machine learning predictions must be displayed on the website.

**Why is UNIX often preferred?**

1. **Open-source**: Because UNIX is open-source and free, this makes it a great choice as an os, even to common users especially since the advents to distributions like Ubuntu which provides a friendly GUI to interact with.

2. **Security**: UNIX and Unix-like operating systems are far more secure even than common operating systems like windows.

3. **Control and customization**: Unlike some other os, UNIX gives more control to the users, even while maintaining a good security.

4. **Multi-tasking**: UNIX can run multiple-processes at the same time.

5. Because of things like multi-tasking and security it provides, you can see it as a server operating system.

**Why is UNIX being referred to as a scientist os?**

The reason why Unix is referred to as a scientist operating system is because almost all super computers in the world used for scientific research is has a variant of Unix as its operating system, also many early adopters of the Unix operating systems where schools and scientists. It’s also popular because it’s better in handling large amounts of data than other operating system, which is crucial being a scientist because they work with really large amounts of data.

**What type of programming language is C?**

C is a *statically typed procedural language*. It’s statically typed because while defining variables and functions you have to specify the data types of these variables and the return types for these functions. And it’s considered a procedural language because it doesn't support the use of classes and object-oriented programming rather it depends on functions, statements and instructions performed in a specific order.

But while one can't do object oriented programming using C, one can still make structs which enable you to specify attributes by grouping different data types together under one place.

**Give a detailed structure of a complete C programming language**

The structure of the c programming c language is divided into six main parts, this makes the code easy to read, understand and allows it to be in a consistent format. The six main parts of the c programming language are;

1. Documentation

2. Link

3. Declaration

4. Global declaration

5. Main function

6. Subprograms

**Documentation**

This is the first part of the code, written as a comment, this usually contains information about what the program is doing, who created it, when it was created and when it was last edited, the version of the compiler being used among so many other things. This provides information to anyone who would later go on to read the code in the future to be able to know what the main goal of the program is without having to read every line of the code, it also provides a way of referencing the person who wrote the code when they can't understand something. An example of a documentation might be like this;

**Link**

This part following the documentation of the code, defines all header files you'll be using in your program, header files are files that contain functions you'll be using in your code, these header files can be ones that programmer writes or ones that comes with your computer. The most common header files you'll see are the stdio.h and stdlib.h files.

**Definition**

This is where you define your macros and constants that you'll be using in your program, these macros can might parameters or not. But now constant variables and functions are preferred over to macros. These macros are always written in capital letters (convention) and defined using the # symbol.

**Global variables**

This is where you'll declare all your global variables, static global variables (i.e. variables visible only to the source files) and function declarations

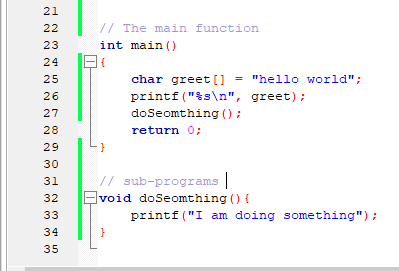
**Main function**

This should is present in every c program, as this is where c would start executing the code written.

**Sub programs**

This includes all user defined functions that the programmer writes for the code to work.





**How can I create a c file on the os?**

* **Step 1**: Install a gcc compiler using the following code

sudo apt install gcc

* **Step 2**: Write a c program in a text editor then save the file with a .c extension.
* **Step 3**: Compile the program, for example if the c program is named sampleProgram

$ gcc sampleProgram.c -o sampleProgram

* **Step 4**: Run the program

$ ./sampleProgram

After running the program, you should see the output of your c program in the console (if you printed any information)