lseek system call

- The lseek System call has been used to read some specific characters or data from a file or to write some content at the specific location of any file.
- This means you can read or write from in between the content of a file.
- lseek() system call helps us to manage the position of the pointer within a file.
- This "lseek" system call requires two header files, e.g., "sys/types.h" and "unistd.h".

Syntax:

```
        off_t lseek(int fd, off_t offset, int whence);
```

- The first parameter "fd" is the file descriptor of the file, which you can get using open() system call.
- The second parameter "offset" specifies how much you want the pointer to move .
- The third parameter "whence" is the reference point of the movement i.e., beginning of file(SEEK_SET), current position(SEEK_CUR) of pointer or end of file(SEEK_END).

Examples:

- lseek(fd,5,SEEK_SET) – this moves the pointer 5 positions ahead starting from the beginning of the file
- lseek(fd,10,SEEK_CUR) – this moves the pointer 10 positions ahead from the current position in the file
- lseek(fd,-15,SEEK_CUR) – this moves the pointer 15 positions back from the current position in the file
- lseek(fd,-5,SEEK_END) -> this moves the pointer 5 positions back from the end of the file.

Returns

On success, lseek() returns the position of the pointer within the file as measured in bytes from the beginning of the file. But, on failure, it returns -1

Listing Directory in UNIX

- The **ls** is the list command in Linux.
- It will show the full list or content of your directory.

Syntax:

```
        ls
```

ls command options

ls options	Description
ls -a	In Linux, hidden files start with . (dot) symbol and they are not visible in the regular directory. The (ls -a) command will enlist the whole list of

	the current directory including the hidden files.
ls –l	It will show the list in long list format.
ls –r	It is used to print the list in reverse order.
ls –R	It will display the content of the sub-directories also.
ls –lt	It will sort the list by displaying recently modified filed at top.
ls ~	It gives the contents of home directory

Help command

help command which as its name says help you to learn about any built-in command.

Syntax:

$help [-dms] [name of the command]

help command options	Description	Example
-d	It only gives short description.	$help –d help help- Display information about built-in command
-s	It gives syntax of the command	$help -s help help: help [-dms] [name of the command]
-m	It displays information of command in an organised format.(i.e. name of command description ,syntax (synopsis), options etc.)	$help –m help NAME help - Display information about builtin commands. SYNOPSIS help [-dms] [pattern ...] DESCRIPTION Display information about builtin commands.

Programming Method

1) **Debugging**

- Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities.
- Debugging checks, detects and corrects errors (or "bugs") to allow proper program operation, according to set specifications.
- Debugging is a part of testing.
- Gdb is debugger for C. It uses command line interface.
- Gdb can step through your source code line-by-line or even instruction by instruction. You may also watch the value of any variable at run-time. In addition, it also helps to identify the place and the reason making the program crash.
- All program to be debugged in gdb must be compiled by gcc with the option "-g" turning on. If we want to debug the program "garbage", we can simply start gdb by:

```
$ gdb ./garbage
```

gdb will give you a prompt like (gdb) and from that prompt you can run your program.

- To start running and debugging the program, we can simply type the "run" command after the (gdb) prompt as below:

```
(gdb) run
```

- The process of debugging begins as soon as the **code** of the **software** is written.

2) **Compiling**

- When you compile your program, the compiler produces a file containing binary code which is directly readable by the machine.
- This file is called an *executable* file, because it can be executed by the machine.
- In Unix, for compiling C code is gcc compiler is used.
- To prepare your program for debugging with gdb, you must compile program with –g flag.
- The basic way of compiling garbage.c into an executable file called "garbage123" is:

```
$ gcc –g -o  garbage123  garbage.c
```

If the program is compiled without errors, you can execute the program by typing "./garbage123" .

```
$ ./garbage123
```

Differentiate between debugger and compiler

Compiler	Debugger
Compiler converts the source code into its equivalent machine code.	Debugger helps to identify the errors in the program and fix them correctly.

It is a software	It is a computer program
Compiler converts the code at once .	The debugger allows you to run your code step by step and it can halt when it crashes.
It takes less time as compared to debugger.	Takes more time to fix errors.
Complied programs might have some eroors/bugs.	Debugged programs cannot have any errors.
It generates Intermediate Language code.	It checks Intermediate Language Code line by line.
Examples: Languages like C, C++ have compliers.	Examples: GNU debugger like GDB, Microsoft Visual Studio debugger