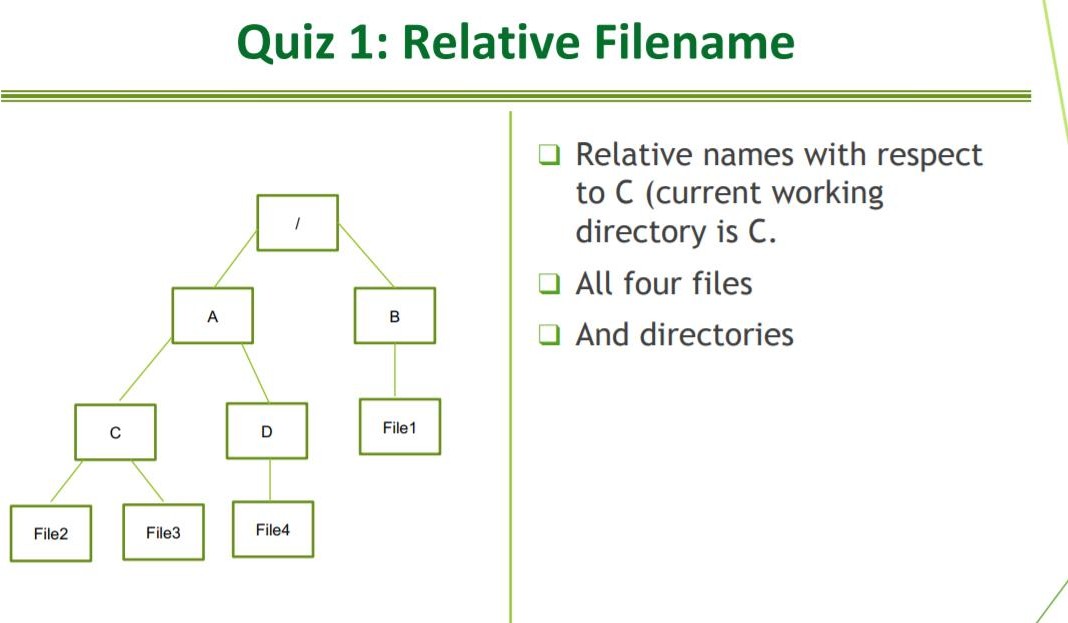
**ASSIGNMENT – 8**

Name – Ashutosh Soni

Id – 2018ucp1505

**1.**

1. **All four files**

../A/B/file1 file2

file3

../A/D/file4

1. **All four directories**

. (for C itself)

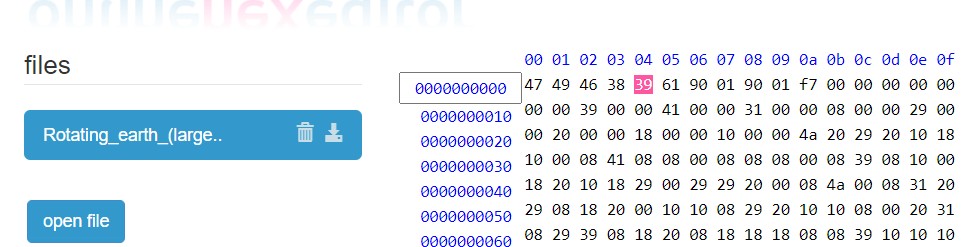
../A

../A/B

../A/D

1. **What are magic numbers of most popular file types? Check against** [**https://en.wikipedia.org/wiki/List\_of\_file\_signatures**](https://en.wikipedia.org/wiki/List_of_file_signatures)

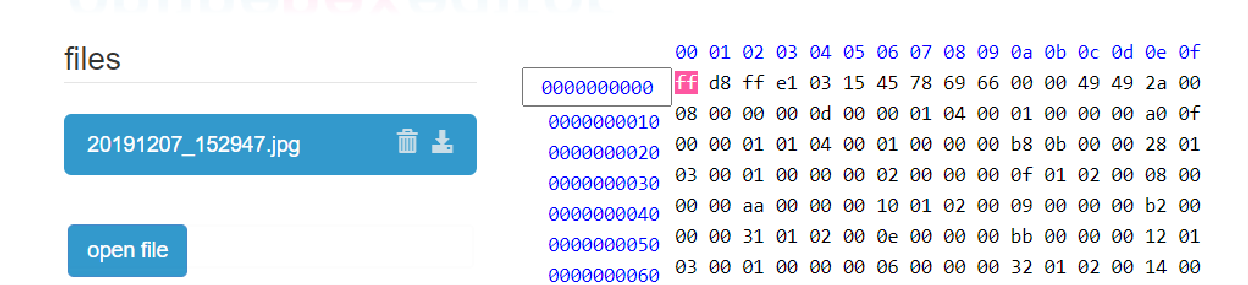
gif : 47 49 46 38 39 61 / 47 49 46 38 37 61

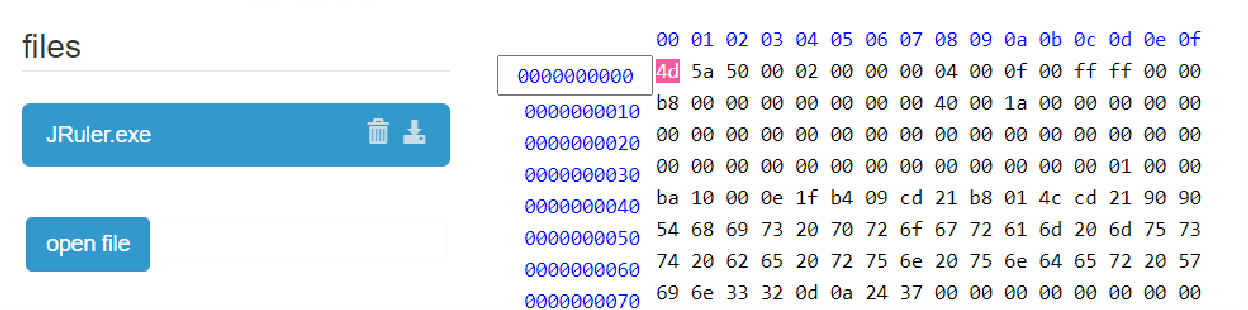


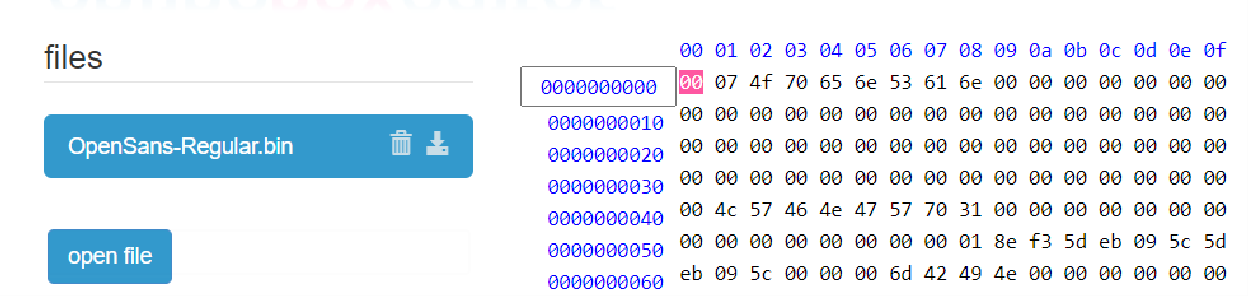
jpg/jpeg :FF D8 FF DB

FF D8 FF E0 00 10 4A 46 49 46 00 01

FF D8 FF EE

FF D8 FF E1 ?? ?? 45 78 69 66 00 00

exe :4D 5A

bin : 53 50 30 31

1. **Identify what fields related to a file are stored in a *directory entry*. Write a small program to read and list a directory entry related to each file.**

Not everything about the file, most of the metadata about the file is stored within the file inode, not the directory entry. The directory entry is just a struct of inode and filename - just enough information to translate from a filename to an inode and get to the actual file. Each *directory entry* contains a filename along with a structure of information describing the attributes of the file. The attributes of a file are such things as the type of file (regular file, directory), the size of the file, the owner of the file, permissions for the file (whether other users may access this file), and when the file was last modified.

## What are linux commands related to identifying/modifying (where applicable)

* **Name of the file :-**

:- ls (list all the files in a directory)

## Type of the file

:- file filename

## Size of the file

:- ls -lh

## Owner of the file

:- ls -l /path/to/file

## Date and time when a file was created

:- stat filename

## Date and time when a file was last accessed/modified

:- stat filename

## Permissions of the file

:- ls –l file\_name

## What shall be chmod command for setting following permissions

* the owner has read and execute access; group has read access only; others have no access.

## chmod u=r-x,g=r fileName

* owner and group have all permissions; other have read and execute permissions only.

## chmod u=rwx,g=rwx,o=r-x fileName

* Current permissions are: owner has read, write and execute access; group has read and execute access only; others can only read. How to drop write permission for the owner also. **chmod u=rwx,g=r-x,o=r—fileName**

## Q.(6). Compute maximum file size that can be stored in a linux system where 15 direct, 1 single indirect, 1 double indirect and 1 triple indirect pointers are stored per i-node. Block size = 512 bytes and disk has 32-bit address.

**Ans :**

Block Size = 512 bytes

32-bit disk address i.e. pointer = 4 bytes Number of pointers per block = 512/4 =128

Number of direct pointers = 15

Number of data block per direct pointer = 1

Number of data block per single indirect pointer = 128 = 2^7 Number of data block per double indirect pointer = 128\*128 = 2^14

Number of data block per triple indirect pointer = 128\*128\*128 = 2^21 Size of file = (15+2^7+2^14+2^21)\*512 bytes

= 3.75KB+32KB+4MB+0.5GB

## How is stat command used to gain information on i-node of a file?

stat is a command-line utility that displays detailed information about given files or file systems.

The syntax for the stat command is as follows:

## stat [OPTION]... FILE...

stat displays the following file information:

* **File** - The name of the file.
* **Size** - The size of the file in bytes.
* **Blocks** - The number of allocated blocks the file takes.
* **IO Block** - The size in bytes of every block.
* **File type** - (ex. regular file, directory, symbolic link.)
* **Device** - Device number in hex and decimal.
* **Inode** - Inode number.
* **Links** - Number of hard links.
* **Access** - File permissions in the numeric and symbolic methods.
* **Uid** - User ID and name of the owner.
* **Gid** - Group ID and name of the owner.
* **Context** - The SELinux security context.
* **Access** - The last time the file was accessed.
* **Modify** - The last time the file’s content was modified.
* **Change** - The last time the file’s attribute or content was changed.
* **Birth** - File creation time (not supported in Linux).

## Explanatory notes on

* 1. **Network File System**

The Network File System (NFS) is a client/server application that lets a computer user view and optionally store and update files on a remote computer as though they were on the user's own computer. The NFS protocol is one of several distributed file system standards for network-attached storage (NAS).

NFS allows the user or system administrator to mount (designate as accessible) all or a portion of a file system on a server. The portion of the file system that is mounted can be accessed by clients with whatever privileges are assigned to each file (read-only or read-

write). NFS uses Remote Procedure Calls (RPCs) to route requests between clients and servers.

## ELF (executable linkable format) file format

ELF is widely used for executable files, relocatable object files, shared libraries, and core dumps. an ELF file consists of two sections – an ELF header, and file data. The file data section can consist of a program header table describing zero or more segments, a section header table describing zero or more sections, that is followed by data referred to by entries from the program header table, and the section header table. Each segment contains information that is necessary for run-time execution of the file, while sections contain important data for linking and relocation

## PE (portable executable) file format

Portable Executable (PE) file format is a file format for executable / dll files introduced in Windows NT. It's based on COFF (Common Object File Format) specification.

To remain compatible with previous versions of the MS-DOS and Windows, the PE file format retains the old MZ header from MS-DOS.

The PE file format is organized as a linear stream of data. It begins with an MS-DOS header, a real-mode program stub, and a PE file signature. Immediately following is a PE file header and optional header. Beyond that, all the section headers appear, followed by all of the section bodies. Closing out the file are a few other regions of miscellaneous information, including relocation information, symbol table information, line number information, and string table data. All of this is more easily absorbed by looking at it graphically.

## Compute cylinder skew

* 1. **Disk speed= 6,000 rpm, 200 sectors per track, seek time = 1000 μ sec** Time to move over 200 sectors = time for one revolution = 60/6000 = 10000 μ sec Time to move over 1 sectors = 10000/200 = 50 μ sec

Sectors moved in 1000 μ sec = 1000/50 = 20 Cylinder skew = 20 sectors

* 1. **Disk speed= 7,500 rpm, 240 sectors per track, seek time = 800 μ sec** Time to move over 240 sectors = time for one revolution = 60/7500 = 8000 μ sec Time to move over 1 sectors = 8000/240 = 33.33 μ sec

Sectors moved in 800 μ sec = 800/33.33 = 24 Cylinder skew = 24 sectors

* 1. **Disk speed= 8,000 rpm, 320 sectors per track, seek time = 600 μ sec** Time to move over 320 sectors = time for one revolution = 60/8000 = 7500 μ sec Time to move over 1 sectors = 7500/320 = 23.4375 μ sec

Sectors moved in 600 μ sec = 600/23.4375 = 25 Cylinder skew = 25 sectors

## Write notes on

* 1. **Swap space management.**

Swap-Swap management is another low-level task of the operating system. Disk space is used as an extension of main memory by the virtual memory. As we know the fact that disk access is much slower than memory access, In the swap-space management we are using disk space, so it will significantly decreases system performance. Basically, in all our systems we require the best throughput, so the goal of this swap-space implementation is to provide the virtual memory the best throughput. In these article, we are going to discuss how swap space is used, where swap space is located on disk, and how swap space is managed.

## RAID

RAID, or “Redundant Arrays of Independent Disks” is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both.

Key evaluation points for a RAID System

Reliability: How many disk faults can the system tolerate?

Availability: What fraction of the total session time is a system in uptime mode, i.e. how available is the system for actual use?

Performance: How good is the response time? How high is the throughput (rate of processing work)? Note that performance contains a lot of parameters and not just the two. Capacity: Given a set of N disks each with B blocks, how much useful capacity is available to the user?

RAID is very transparent to the underlying system. This means, to the host system, it appears as a single big disk presenting itself as a linear array of blocks. This allows older technologies to be replaced by RAID without making too many changes in the existing code