W1650209 Assignment No : 5

1. Is the “open” system call in UNIX absolutely essential? What would be the consequences of not having it?

Ans:

1) Most of the file operations mentioned involve searching the directory for the entry associated with the named file.
2) To avoid this constant searching, many systems require that an open () system call be made before a file is first used actively.
3) Some systems implicitly open a file when the first reference to it is made.
4) Most systems, however, require that the programmer open a file explicitly with the open () system call before that file can be used.
5) The open () system call typically returns a pointer to the entry in the open-file table.
6) This pointer, not the actual file name, is used in all I/O operations, avoiding any further searching, and simplifying the system-call interface.
7) The operating system keeps a small table, called the open-file table, containing information about all open files.
8) When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being actively used, it is *closed* by the process, and the operating system removes its entry from the open-file table.
9) The file is automatically closed when the job or program that opened the file terminates.
10) If no system call is made then for every file operation, it will be necessary to specify the name of referencing file.

2. In some systems it is possible to map part of a file into memory. What restrictions must such systems impose? How is this partial mapping implemented?

Ans:

1) The part of file can be mapped into the memory.
2) The mapped portion of the file must start at a page boundary and be an integral number of pages in length.
3) The mapped page use itself as a backing store. for a unmapped memory or a file uses a scratch file or partition as backing store.

3. Name one advantage of hard links over symbolic links and one advantage of symbolic links over hard links.

Ans:

1) A hard link shares the same file system structures, while a soft-link is a pathname redirect.
2) Hardlinks must be on the same filesystem, softlinks can be on cross filesystems.
3) Hardlinked files stay linked even if you move either of them. Softlinked files break if you move the original, and sometimes when you move the link .
4) Hardlinked files are treated similarly, while the original is special in softlinks, and deleting the original deletes the data. The data can be restored as long as hardlinks are preserved.
5) Softlinks can require special support from filesystem walking tools.
6) No additional space is required for the hardlinks, whereas softlinks needs space to store name of file pointed to it.
7) Hardlinks can point only files related to it's partition, whereas symbolic links can point to any file on other machine, even on internet.

4. One way to use contiguous allocation of disk and not suffer from holes is to compact the disk every time a file is removed. Since all files are contiguous, copying a file requires a seek and rotational delay to read the file, followed by the transfer at full speed. Writing the file back requires the same work. Assuming a seek time of 5 msec, a rotational delay of 4msec, a transfer rate of 8MB/sec and an average file size of 8KB, how long does it take to read a file into main memory then write it back to the disk at a new location? Using these numbers, how long would it take to compact half of a 16GB disk?

Ans:

Seek Time = 5m sec

Rotational delay = 4msec

Transfer rate = 80 MB/sec

File Size= 8KB.

Reading into main memory take :

$= 8 * 2^{10} = 2^{13}$ Bytes.

Transfer Rate = 80MB/sec

$= 80* 2^{20} = 2^{23}*10$ Bytes/sec

Transfer Required = average file size/ transfer rate

$= 2^{13}/2^{23}*10 = ½^{10}*10 = 0.0000977= 0/97$

Transfer = seek + rotation

Hence the total time to seek, rotate and transfer is 9.977msec, writing back take another 9.977msec.

Thus, copying average takes 9.77+9.77 = 19.954 msec, To compact half of 16GB disk would involve copying 8GB of storage.

8GB can store = $2^3*2^{10}*2^{10}*2^{10} = 2^{33}$ Bytes.

Number of files stored in 8GB= $2^{33}/2^{13} = 2^{20}$

At 19.954 msec per file, this takes 1024*1024*19.953= 20,923 seconds, which is 5.8 hours.

Clearly, compacting the disk every file removal is not great idea.

5. The beginning of the free space bitmap looks like this after the disk partition is first formatted: 1000 0000 0000 0000 (the first block is used by the root directory). The system always searches for free blocks starting at the lowest numbered block, so after writing a file A which uses 6 blocks, the bitmap looks like this: 1111 1110 0000 0000. Show the bitmap after each of the following additional actions:

   (a) File B is written, using 5 blocks.
   (b) File A is deleted.
   (c) File C is written, using 8 blocks.
   (d) File B is deleted.

Ans:

(a) 1111 1111 1111 0000

(b) 1000 0001 1111 0000

(c) 1111 1111 1111 1100

(d) 1111 1110 0000 1100

6. Explain how hard links and soft links differ with respective to i-node allocations.

Ans:

1) A hard link is an additional name of the original file which refers inode to access the target file. In contrast, soft link is distinct to the original file and is an alias to original file but do not uses inode.

2) In Linux, the command used for the creation of hard link is “ln”. As against, the command used for a soft link is “ln-s“.

3) Hard link has same inode number dissimilar to soft link, where target file and its soft link have distinct inode number.

4) Hard links are restricted to its own partitions, but soft links can cover different file systems.

5) The performance of hard link is better than soft link in some cases.

6) Relative path and absolute path both are allowed in soft links. On the contrary, the relative path is not allowed in a hard link.

7. Explain the tradeoffs between precise and imprecise interrupts on a superscalar machine.
   For the precise interrupt the main benefit is simplicity of code in os as machine is well defined.

Ans:

1) An benefit of precise interrupts is simplicity of code in the operating system since the machine state is well defined.
2) For imprecise interrupt, OS must figure what instructions have been partially executed and upto what pint, it will impact.
3) precise interrupt adds to the complications of chip design, which may result in slower in CPU.

8. Explain how an OS can facilitate installation of a new device without any need for recompiling the OS.

Ans:

1) Drivers are used to facilitate installation of new device.
2) Drivers are generally written by manufacture of the device.
3) To access the device has to be part of the OS kernel.
4) The OS classify drivers into blocks disks and character disks.