Assignment 8

11.9 Consider a file system in which a file can be deleted and its disk space reclaimedwhile links to that file still exist.What problemsmay occur if a newfile is created in the same storage area orwith the same absolute path name? How can these problems be avoided?

11.12 Provide examples of applications that typically access files according to the following methods:

- Sequential
- Random

11.17 Some systems provide file sharing by maintaining a single copy of a file. Other systems maintain several copies, one for each of the users sharing the file. Discuss the relative merits of each approach.

12.13 Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the free-space management scheme in order to support this feature?

12.16 Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

12.17 Fragmentation on a storage device can be eliminated by recompaction of the information. Typical disk devices do not have relocation or base registers (such as those used when memory is to be compacted), so how can we relocate files? Give three reasons why recompacting and relocation of files are often avoided.

7 Run the following program and observe the output. The objective is to understand how the *open* files are shared by the child and the parent process.

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <fcntl.h>

int main( )

{
```

```
int fp;
char buff[20];
int pid;

pid = fork( );
if (pid==0)
 {
 fp = open("sample", O_RDONLY);
 printf("child begins %d\n" , getpid( ));
 read(fp,buff,10);
 buff[10] = '\0';
 printf("child read: ");
 puts(buff);
 printf("\nchild exits");
 }
else
 {
 fp = open("sample", O_RDONLY);
 printf("parent begins %d\n" , getpid( ));
 read(fp,buff,10);
 buff[10] = '\0';
 printf("parent read: ");
 puts(buff);
 printf("parent exits\n");
 }
}
```