Batch-3
17/12/2020

USP LAB

Chirag Swamy
1BM19CS402

1)
```
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <time.h>
void PrintFileProperties (struct stat stats)
{
  struct tm dt;
  printf("\n File Permissions : \t");
  if (S_ISDIR (stats.st_mode))
      printf("d");
  else
      printf("-");
  if (stats.st_mode & S_IRUSR)
      printf("r");
  else
      printf("-");
  if (stats.st_mode & S_IWUSR)
      printf("w");
  else
      printf("-");
  if (stats.st_mode & S_IXUSR)
      printf("x");
  else
      printf("-");
  printf("\t");
  if (stats.st_mode & S_IRGRP)
      printf("r");
  else
      printf("-");
  if (stats.st_mode & S_IWGRP)
      printf("w");
  else printf("-");
```

Chirag Swamy
1BM19CS402

```
if (Stats.st_mode & S_IXGRP)
        printf("x");
else    printf("-");
printf("\t");
printf("\n File Size : \t %ld", Stats.st_size);
printf("\n No of links : \t %ld", Stats.St_nlink);
printf("\n File inode Number : \t % ld", Stats.st_ino);
dt = *(gmtime (&Stats. st.ctime));
printf("\n File Createda on : \t %d -%d -%d \t %d : %d : %d",
        dt.tm_mday, dt.tm_mon, dt.tm_year + 1900, dt.tm_hour, dt.tm_min,
        dt.tm_sec);
dt = * (gmtime (& Stats . st.mtime));
printf ("\n File last modified on : \t %d - %d .- %d \t %d:%d:%d",
        dt.tm_mday, dt.tm_mon, dt.tm_year +1900, dt.tm_hour, dt.tm_min,
        dt.tm_sec);
}
int main ()
{
  Char path [1000];
  struct Stat Stats;
  printf ("Enter file path : ");
  scanf ("%s", path);
  if (stat (path, & stats) == 0)
  {
     printFileProperties (Stats);
  }
  else
  {
   printf (" Unable to get file properties ");
   printf (" \n Check whether % s file exists ", path);
  }
}
```

Chiang Swamy
1BM19CS402

2]
```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/stat.h>
int main (int argc, char *argv[])
{
  char buf;
  int size, fd;
  fd = open (argv[1], O_RDONLY);
  size = lseek (fd, -1, SEEK_END);
  while (size-- >=0)
  {
    read (fd, &buf, 1);
    write (STDOUT_FILENO, &buf, 1)
    lseek (fd, -2, SEEK_CUR);
  }
  printf ("\n");
  return 0;
}
```

3]
```
#include stdio.h>
# include <unistd.h>
int main ()
{
  int j;
  j = fork ();
  if (j == 0)
  {
    printf (
```

3]

```
#include <stdio.h>
#include <unistd.h>
int main()
{
  int j= fork();
  if (j==0)
  {
    Printf ("This is child process , child process id : %d \n", getpid());
    Printf ("The Child's parent process id is : %d \n ", getppid());
  }
  else
  {
    sleep(3);
    Printf("This is parent process , Parent process id : %d \n", getpid());
    Printf (" Parent's parent process id is : %d \n ", getppid());
    Printf (" Return value of fork to parent is child's PID : %d \n",
                                                              j);
  }
return 0;
}
```