3) fork()

```
int main(void)
 {
     int pid;
     print("Before fork \n");
     pid = fork();
      if (pid>0) {
             Sleep(3);
            Print("Parent --PID %od PPID %od, CHILD PID: %od \n",
                           getpid(), getppid(), pid);
        }
    else if(pid ==0)
        printf("Child -- PID: %od PPID: %od \n", getpid(), getppId());
        else {
            printf("fork error \n");
            exit(1);
           }
       printf("Both processes continue from hereon");
       exit(0);
   }
```

1)
```
int main(int argc, char *agr[])
 {
     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);
    }
    return 0;
}
```

2)
```
int main (int argc, char *argv[])
{
  struct stat statbuf
  if (lstat (argv[1], &statbuf) == -1) {
          printf ("Couldn't stat file");
          exit(0);
      }
   printf ("file: %s \n", argv[1]);
   printf (" inode number: %d \n", statbuf, st_ino);
   printf (" GID = %d", statbuf st_gid);
   printf ("UID = %d", statbuf.st_uid);
   printf (" type & permission: %o \n", statbuf. st_mode);
   printf (" No. of links: %d \n ", statbuf. st_nlink);
   printf ("size in bytes: %d \n", statbuf.st_stat);
   printf ("Block allocated : %d \n", statbuf. st_blocks );
   exit(0);
}
```