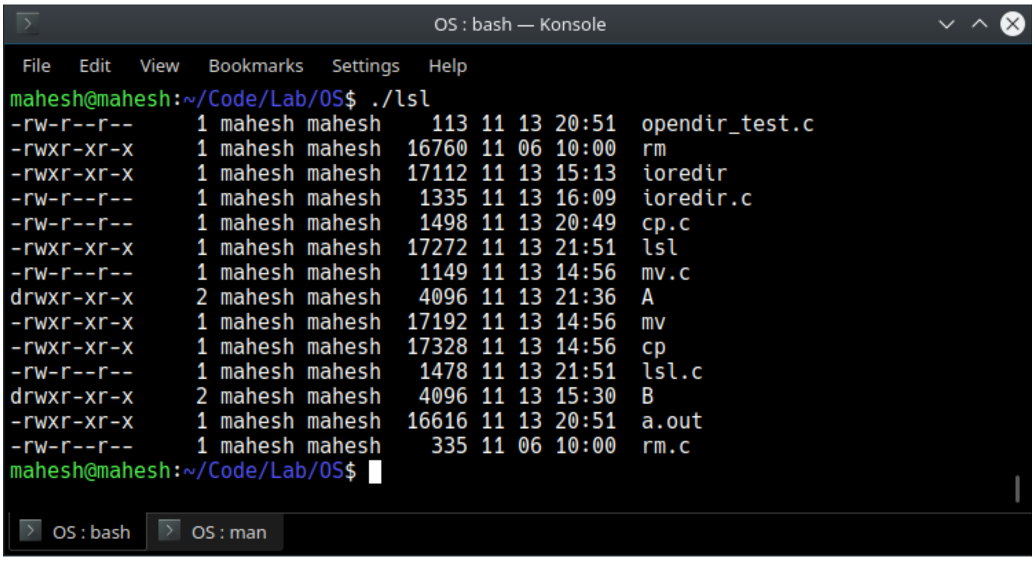
Program 1.
Implement ls -l, cp, mv and rm

1. ls -l

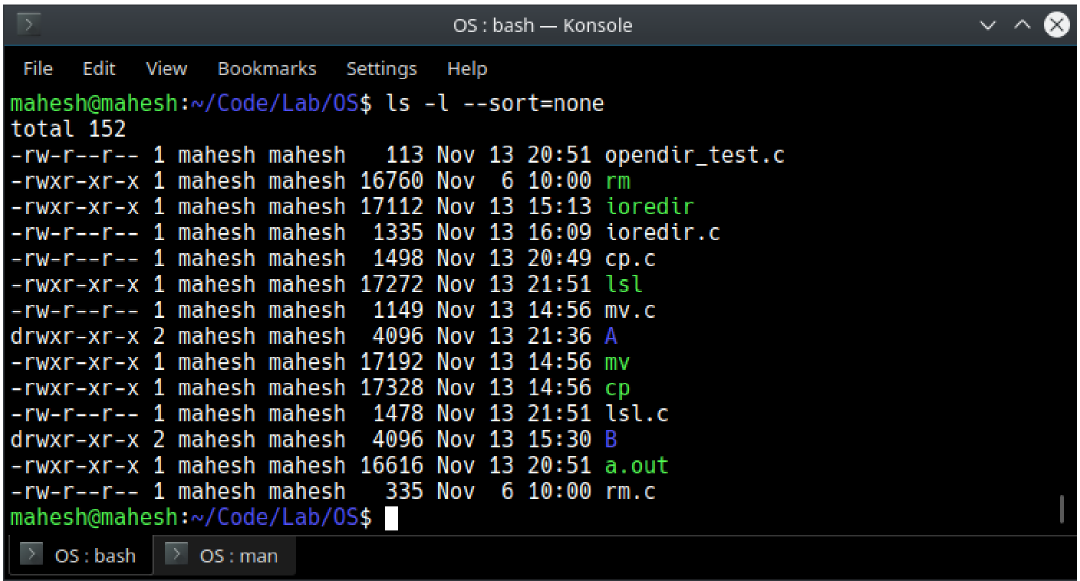
```
#include<stdio.h>
#include<stdlib.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<dirent.h>
#include<string.h>
#include<pwd.h>
#include<grp.h>
#include<time.h>
#include<unistd.h>

void lsl(const char *path) {
     if (chdir(path) != 0) {
          perror("Cannot change dir");
          return;
     }
     DIR *dir = opendir(".");
     struct dirent *entry = NULL;
     char time[50], p[10] = {'r','w','x','r','w','x','r','w','x','\
0'};
     int i, j;

     while((entry = readdir(dir)) != NULL) {
          const char *name = entry->d_name;
          if(name[0] == '.') {
               continue;
               // don't display hidden files
               // including "." and ".."
          }

          struct stat s;
          stat(entry->d_name, &s);
          printf(S_ISDIR(s.st_mode)? "d"
               : S_ISREG(s.st_mode)? "-"
               : S_ISCHR(s.st_mode)? "c"
               : S_ISBLK(s.st_mode)? "b"
               : S_ISLNK(s.st_mode)? "l"
               : S_ISFIFO(s.st_mode)? "f"
               : S_ISSOCK(s.st_mode)? "s"
               : "?"
          );
          for(i = 0, j = (1<<8); i < 9; i++, j>>=1) {
               if (s.st_mode & j) {
                    printf("%c", p[i]);
               } else {
                    printf("%c", '-');
               }
          }
          printf("%6ld ", (long)s.st_nlink);
          printf("%6s ", getpwuid(s.st_uid)->pw_name);
          printf("%6s ", getgrgid(s.st_gid)->gr_name);
          printf("%6ld ", (long)s.st_size);
          strftime(time, sizeof(time), "%m %d %H:%M",
localtime(&(s.st_mtime)));
          printf("%s  ", time);
          printf("%s\n", entry->d_name);
     }
}

int main(int argc, char **argv) {
     switch(argc) {
     case 1: lsl(".");
          break;
     case 2: lsl(argv[1]);
          break;
     default: printf("Too many arguments\n");
          return 1;
     }
}
```

Output:



```
mahesh@mahesh:~/Code/Lab/OS$ ./lsl
-rw-r--r--     1 mahesh mahesh    113 11 13 20:51  opendir_test.c
-rwxr-xr-x     1 mahesh mahesh  16760 11 06 10:00  rm
-rwxr-xr-x     1 mahesh mahesh  17112 11 13 15:13  ioredir
-rw-r--r--     1 mahesh mahesh   1335 11 13 16:09  ioredir.c
-rw-r--r--     1 mahesh mahesh   1498 11 13 20:49  cp.c
-rwxr-xr-x     1 mahesh mahesh  17272 11 13 21:51  lsl
-rw-r--r--     1 mahesh mahesh   1149 11 13 14:56  mv.c
drwxr-xr-x     2 mahesh mahesh   4096 11 13 21:36  A
-rwxr-xr-x     1 mahesh mahesh  17192 11 13 14:56  mv
-rwxr-xr-x     1 mahesh mahesh  17328 11 13 14:56  cp
-rw-r--r--     1 mahesh mahesh   1478 11 13 21:51  lsl.c
drwxr-xr-x     2 mahesh mahesh   4096 11 13 15:30  B
-rwxr-xr-x     1 mahesh mahesh  16616 11 13 20:51  a.out
-rw-r--r--     1 mahesh mahesh    335 11 06 10:00  rm.c
mahesh@mahesh:~/Code/Lab/OS$
```

for comparison, below is output of system ls command



```
mahesh@mahesh:~/Code/Lab/OS$ ls -l --sort=none
total 152
-rw-r--r-- 1 mahesh mahesh   113 Nov 13 20:51 opendir_test.c
-rwxr-xr-x 1 mahesh mahesh 16760 Nov  6 10:00 rm
-rwxr-xr-x 1 mahesh mahesh 17112 Nov 13 15:13 ioredir
-rw-r--r-- 1 mahesh mahesh  1335 Nov 13 16:09 ioredir.c
-rw-r--r-- 1 mahesh mahesh  1498 Nov 13 20:49 cp.c
-rwxr-xr-x 1 mahesh mahesh 17272 Nov 13 21:51 lsl
-rw-r--r-- 1 mahesh mahesh  1149 Nov 13 14:56 mv.c
drwxr-xr-x 2 mahesh mahesh  4096 Nov 13 21:36 A
-rwxr-xr-x 1 mahesh mahesh 17192 Nov 13 14:56 mv
-rwxr-xr-x 1 mahesh mahesh 17328 Nov 13 14:56 cp
-rw-r--r-- 1 mahesh mahesh  1478 Nov 13 21:51 lsl.c
drwxr-xr-x 2 mahesh mahesh  4096 Nov 13 15:30 B
-rwxr-xr-x 1 mahesh mahesh 16616 Nov 13 20:51 a.out
-rw-r--r-- 1 mahesh mahesh   335 Nov  6 10:00 rm.c
mahesh@mahesh:~/Code/Lab/OS$
```

2. cp

```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>

// needed if directories are accepted as destinations
char *append_path(const char *a, const char *b) {
     ssize_t len_a = strlen(a), len_b = strlen(b);
     ssize_t sz = (a[len_a-1] == '/') ? len_a+len_b+1 :
len_a+len_b+2;
     char *target = malloc((len_a+len_b+1)*sizeof(char));
     memcpy(target, a, len_a);

     if (a[len_a-1] != '/') {
          target[len_a] = '/';
          len_a++;
     }

     memcpy(target+len_a, b, len_b + 1);
     return target;
}

// name of file given by complete path
const char *file_name(const char *path) {
     char *s;
     if (s = strrchr(path, '/')) {
          return s+1;
     }
     return path;
}


void copyfile(char *dest, char *src) {
     struct stat s;
     int statret = stat(dest, &s);
     if (statret != -1 && S_ISDIR(s.st_mode)) {
          dest = append_path(dest, file_name(src));
     }
     // open dest and src
     int dest_fd, r, w;
     char buf[256];
     int src_fd = open(src, O_RDONLY);
     if (src_fd < 0) {
          perror("Cannot open source file");
          exit(EXIT_FAILURE);
     }

     dest_fd = open(dest, O_WRONLY | O_TRUNC | O_CREAT, S_IRUSR |
S_IWUSR);
```

```
    if (dest_fd < 0) {
        perror("Cannot open destination file");
        exit(EXIT_FAILURE);
    }

    while((r=read(src_fd,buf,256)) > 0) {
        write(dest_fd, buf, r);
    }

    if (r != 0) {
        perror("Read failed");
    }
    if (S_ISDIR(s.st_mode)) free(dest);
    return;
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Provide exactly 2 arguments");
        exit(EXIT_FAILURE);
    }

    copyfile(argv[2], argv[1]);
    return 0;
}
```

3. **mv**

```
#include<fcntl.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

#include<sys/stat.h>
#include<sys/types.h>

#include<string.h>

// If destination argument is a directory, we need this to work
same as unix mv command

char *append_path(const char *a, const char *b) {
    ssize_t len_a = strlen(a), len_b = strlen(b);
    ssize_t sz = (a[len_a-1] == '/') ? len_a+len_b+1 :
len_a+len_b+2;
    char *target = malloc((len_a+len_b+1)*sizeof(char));
    memcpy(target, a, len_a);

    if (a[len_a-1] != '/') {
        target[len_a] = '/';
        len_a++;
    }

    memcpy(target+len_a, b, len_b + 1);
```

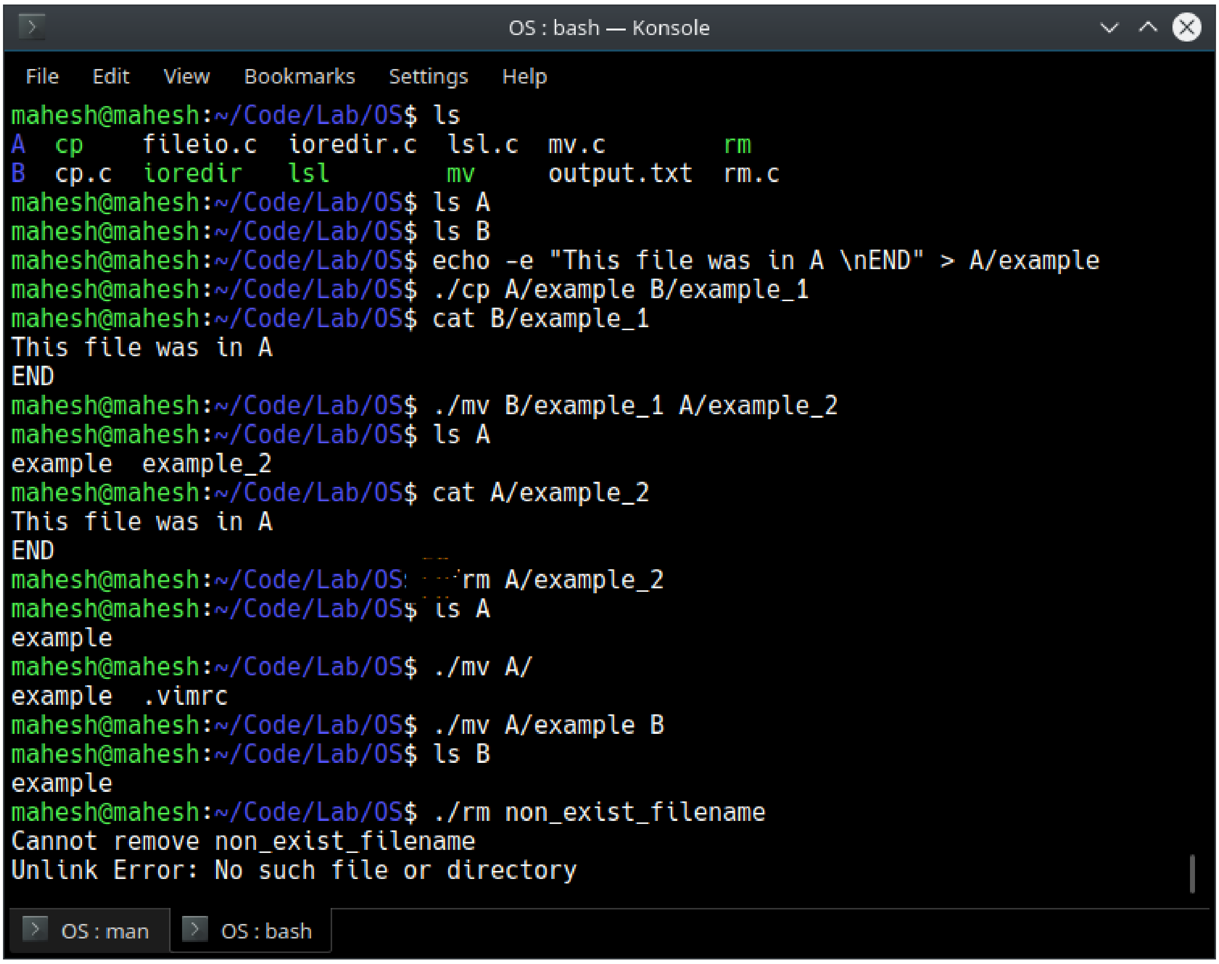
```
     return target;
}

// name of file given by complete path
const char *file_name(const char *path) {
     char *s;
     if (s = strrchr(path, '/')) {
          return s+1;
     }
     return path;
}

int main(int argc, char *argv[]) {
     char *dest = argv[2];
     int statret;
     if(argc < 3) {
          printf("Need 2 arguments\n");
          exit(EXIT_FAILURE);
     }

     struct stat s;
     statret = stat(argv[2], &s);

     if(statret != -1 && S_ISDIR(s.st_mode)) {
          dest = append_path(argv[2], file_name(argv[1]));
     }

     if(rename(argv[1], dest) == -1) {
          perror("Move failed");
     }

     if (S_ISDIR(s.st_mode)) free(dest);
     return 0;
}
```

4. rm

```
#include<stdio.h>
#include<unistd.h>

int main(int argc, char *argv[]) {
     int i;
     if (argc < 2) {
          printf("Need at least 1 argument");
          return 1;
     }

     for(i=1;i<argc;i++) {
          int status = unlink(argv[i]);
          if (status == -1) {
               printf("Cannot remove %s\n", argv[i]);
               perror("Unlink Error");
               printf("\n");
          }
```

```
    }
    return 0;
}
```

Output for cp, mv and rm



```
mahesh@mahesh:~/Code/Lab/OS$ ls
A  cp    fileio.c  ioredir.c  lsl.c  mv.c        rm
B  cp.c  ioredir   lsl        mv     output.txt  rm.c
mahesh@mahesh:~/Code/Lab/OS$ ls A
mahesh@mahesh:~/Code/Lab/OS$ ls B
mahesh@mahesh:~/Code/Lab/OS$ echo -e "This file was in A \nEND" > A/example
mahesh@mahesh:~/Code/Lab/OS$ ./cp A/example B/example_1
mahesh@mahesh:~/Code/Lab/OS$ cat B/example_1
This file was in A
END
mahesh@mahesh:~/Code/Lab/OS$ ./mv B/example_1 A/example_2
mahesh@mahesh:~/Code/Lab/OS$ ls A
example  example_2
mahesh@mahesh:~/Code/Lab/OS$ cat A/example_2
This file was in A
END
mahesh@mahesh:~/Code/Lab/OS$ ./rm A/example_2
mahesh@mahesh:~/Code/Lab/OS$ ls A
example
mahesh@mahesh:~/Code/Lab/OS$ ./mv A/
example  .vimrc
mahesh@mahesh:~/Code/Lab/OS$ ./mv A/example B
mahesh@mahesh:~/Code/Lab/OS$ ls B
example
mahesh@mahesh:~/Code/Lab/OS$ ./rm non_exist_filename
Cannot remove non_exist_filename
Unlink Error: No such file or directory
```