Advanced Unix Programming
Lab 2

Purva Tendulkar : 111403049

Q1. Using APIs, write the equivalent program for the following shell script.

```
mkdir junk
for i in 1 2 3 4 5 do
echo hello >junk/$i
done
ls –l junk
chmod –r junk
ls –l
chmod +r junk
ls –l junk
chmod –x junk
cd junk
chmod +x junk
cd junk
```

Write appropriate comments in the program to observe the execution output.

Code :

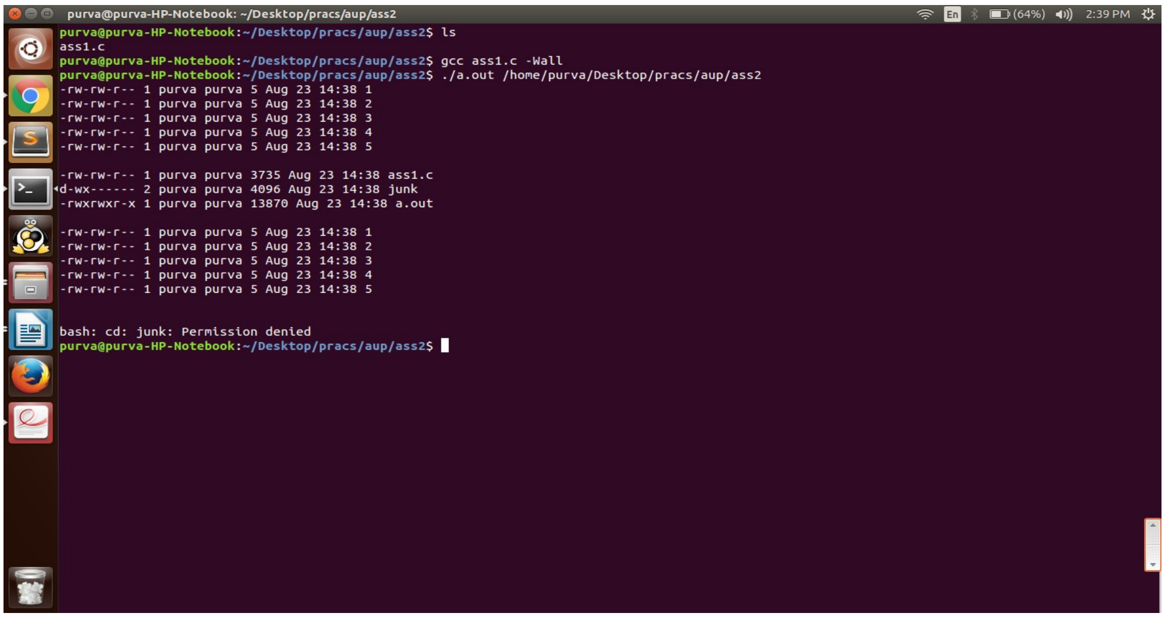
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include <time.h>
#include <pwd.h>
#include <grp.h>

void lsminusl(char *temp, char *filename) {
        int sz;
        char date[20], timet[20];
        struct stat fileStat;
        struct passwd *pw;
        struct group *gr;

        if(stat(temp, &fileStat) < 0)
        return;

        /* File permissions */
   printf( (S_ISDIR(fileStat.st_mode)) ? "d" : "-");
   printf( (fileStat.st_mode & S_IRUSR) ? "r" : "-");
   printf( (fileStat.st_mode & S_IWUSR) ? "w" : "-");
   printf( (fileStat.st_mode & S_IXUSR) ? "x" : "-");
   printf( (fileStat.st_mode & S_IRGRP) ? "r" : "-");
   printf( (fileStat.st_mode & S_IWGRP) ? "w" : "-");
```

```
    printf( (fileStat.st_mode & S_IXGRP) ? "x" : "-");
    printf( (fileStat.st_mode & S_IROTH) ? "r" : "-");
    printf( (fileStat.st_mode & S_IWOTH) ? "w" : "-");
    printf( (fileStat.st_mode & S_IXOTH) ? "x" : "-");
    printf(" ");

    /* Number of links/directories */
        printf("%d ", (int)fileStat.st_nlink);

        /* user */
        pw = getpwuid(fileStat.st_uid);
        printf("%s ", pw->pw_name);

        /* group */
        gr = getgrgid(fileStat.st_gid);
        printf("%s ", gr->gr_name);

        /* size in bytes */
        sz = fileStat.st_size;
        printf("%d ", sz);

        /* date */
        strftime(date, 20, "%b %d", localtime(&(fileStat.st_ctime)));
        printf("%s ", date);

        /* time */
        strftime(timet, 20, "%H:%M", localtime(&(fileStat.st_ctime)));
        printf("%s ", timet);

        /* name of file */
        printf("%s\n", filename);
}

int main(int argc, char* argv[]) {
        struct stat st = {0}, fileStat;
        int statchmod, i;
        FILE *fp;
        char filename[2], str[] = "Hello", directory[100], temp[100];
        DIR *dir;
        struct dirent *ent;

        if (argc != 2) {
                printf("Invalid numner of arguments\n");
                return -1;
        }

        strcpy(directory, argv[1]);
        strcat(directory, "/junk");

        /* mkdir junk */
        if (stat(directory, &st) == -1)
        mkdir(directory, 0700);
```

```
	/* for loop */
	for (i = 1; i <= 5; i++) {
		sprintf(filename,"%d", i);
		strcpy(temp, directory);
		strcat(temp, "/");
		strcat(temp, filename);
		fp = fopen(temp, "w+");
		fprintf(fp, "%s", str);
		fclose(fp);
	}

	/* ls -l junk */
	for (i = 1; i <= 5; i++) {
		sprintf(filename,"%d", i);
		strcpy(temp, directory);
		strcat(temp, "/");
		strcat(temp, filename);
		lsminusl(temp, filename);
	}
	printf("\n");

	stat(directory, &st);
	statchmod = st.st_mode & (S_IRWXU | S_IRWXG | S_IRWXO);

/* chmod -r junk : reset first bit */
statchmod = statchmod & 0377;
chmod(directory, statchmod);

	/* ls -l */
	if ((dir = opendir (argv[1])) != NULL) {
		while ((ent = readdir (dir)) != NULL) {
			if ((strcmp(ent->d_name, ".") == 0) || (strcmp(ent->d_name, "..") == 0))
				continue;
			strcpy(temp, argv[1]);
			strcat(temp, "/");
			strcat(temp, ent->d_name);
			lsminusl(temp, ent->d_name);
		}
		closedir (dir);
	}
	else {
	  /* could not open directory */
	  perror ("");
	  return EXIT_FAILURE;
	}
	printf("\n");

	/* chmod +r junk : set first bit */
statchmod = statchmod | 0400;
chmod(directory, statchmod);
```

```
        /* ls -l junk */
        for (i = 1; i <= 5; i++) {

                sprintf(filename,"%d", i);
                strcpy(temp, directory);
                strcat(temp, "/");
                strcat(temp, filename);

                lsminusl(temp, filename);
    }
    printf("\n");

    /* chmod -x junk */
    statchmod = statchmod & 0677;
    chmod(directory, statchmod);

    /* cd junk */
        i = chdir(directory);
        if (i == -1)
                printf("\nbash: cd: junk: Permission denied\n");

    /* chmod +x junk */
    statchmod = statchmod | 0100;
    chmod(directory, statchmod);

        /* cd junk */
        i = chdir(directory);
        if (i == -1) {
                printf("\nbash: cd: junk: Permission denied\n");
                return -1;
        }

        /* changed working directory to junk */

        return 0;
}
```

Input & Output Screenshot :



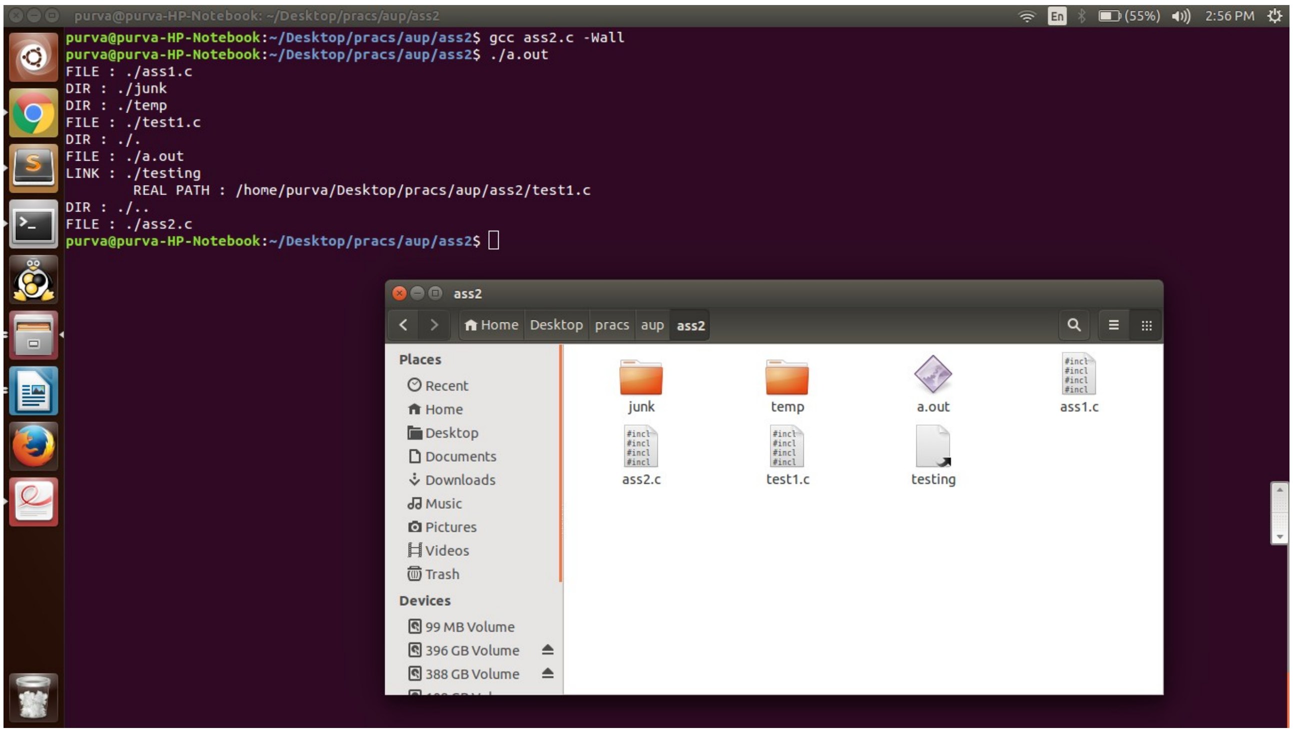
```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ls
ass1.c
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ gcc ass1.c -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ./a.out /home/purva/Desktop/pracs/aup/ass2
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 1
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 2
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 3
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 4
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 5

-rw-rw-r-- 1 purva purva 3735 Aug 23 14:38 ass1.c
d-wx------ 2 purva purva 4096 Aug 23 14:38 junk
-rwxrwxr-x 1 purva purva 13870 Aug 23 14:38 a.out

-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 1
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 2
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 3
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 4
-rw-rw-r-- 1 purva purva 5 Aug 23 14:38 5


bash: cd: junk: Permission denied
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ 
```

Q2. A function realpath() resolves all symbolic links in path and returns the ultimate target. Write a program to list the ultimate target of the only filenames that are symbolic links in a directory. The program takes one optional argument, which is the name of a directory to be searched for the links. When no argument is specified, the search is conducted in the current working directory. Display appropriate error messages.

Code :

```
#include <stdio.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <limits.h>
#include <stdlib.h>

int main (int argc, char * argv[]) {
        struct dirent *dp;
        DIR *dfd;
        char *dir;
        char filename_qfd[100];

        if (argc == 2) {
                dir = argv[1] ;
        }
        else
                dir = ".";

        if ((dfd = opendir(dir)) == NULL) {
                fprintf(stderr, "Can't open the directory%s\n", dir);
                return 0;
        }

        while ((dp = readdir(dfd)) != NULL) {
                struct stat stbuf ;
                sprintf( filename_qfd , "%s/%s",dir,dp->d_name) ;
                if( lstat(filename_qfd,&stbuf ) == -1 ) {
                        printf("Unable to stat file: %s\n",filename_qfd) ;
                        continue ;
                }
                if ( ( stbuf.st_mode & S_IFMT ) == S_IFDIR ) {
                        printf("DIR : %s\n",filename_qfd) ;
                }
                else {
                        if ( ( stbuf.st_mode & S_IFMT ) == S_IFLNK ) {
                                printf("LINK : %s\n",filename_qfd);
                                char rp[PATH_MAX+1];
                                realpath(filename_qfd,rp);
                                printf("\tREAL PATH : %s\n",rp);
                        }
                        else
                                printf("FILE : %s\n",filename_qfd) ;
                }
        }

        return 0;
}
```

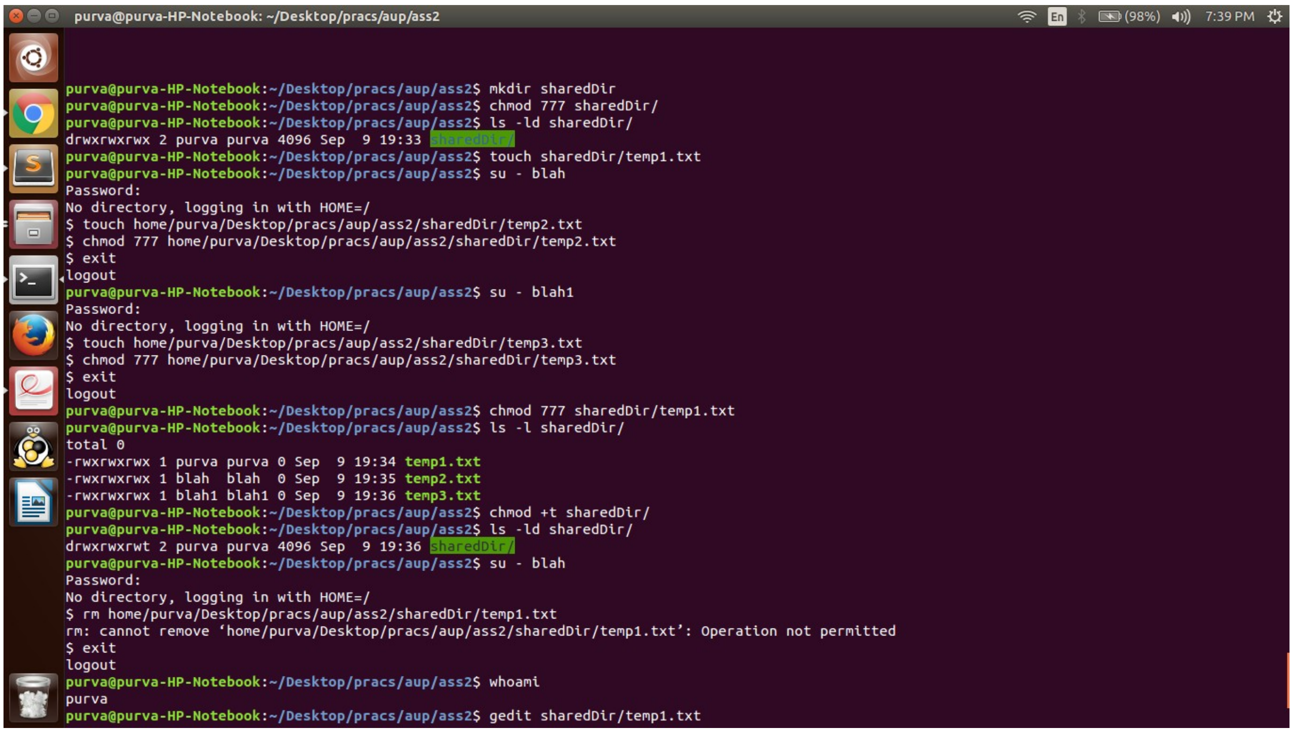
Input & Output Screenshot :



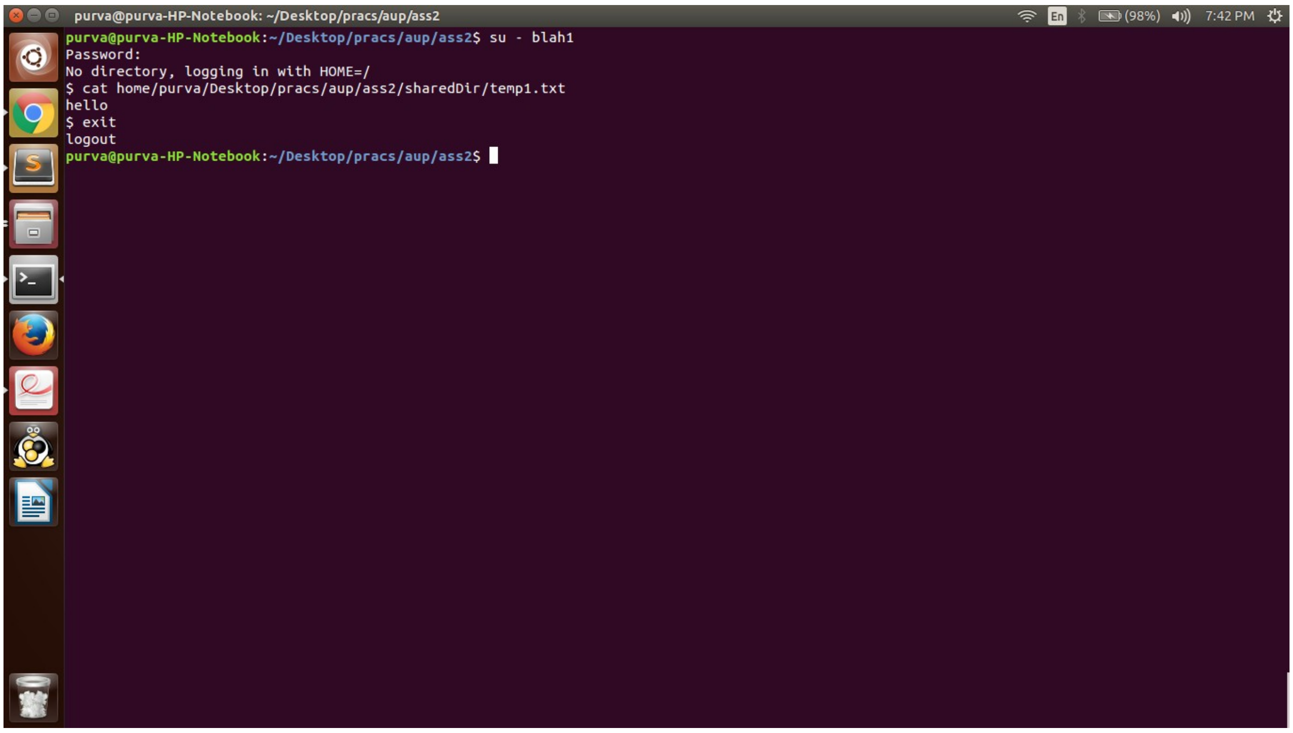
```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ gcc ass2.c -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ./a.out
FILE : ./ass1.c
DIR : ./junk
DIR : ./temp
FILE : ./test1.c
DIR : ./.
FILE : ./a.out
LINK : ./testing
        REAL PATH : /home/purva/Desktop/pracs/aup/ass2/test1.c
DIR : ./..
FILE : ./ass2.c
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$
```

Q3. Create a shared directory for usage with a purpose that any user (not super user) can create new files in this directory, but only the owner can delete his own files and everyone else can read all files?

Input and Output Screenshots :



```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ mkdir sharedDir
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ chmod 777 sharedDir/
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ls -ld sharedDir/
drwxrwxrwx 2 purva purva 4096 Sep  9 19:33 sharedDir/
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ touch sharedDir/temp1.txt
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ su - blah
Password:
No directory, logging in with HOME=/
$ touch home/purva/Desktop/pracs/aup/ass2/sharedDir/temp2.txt
$ chmod 777 home/purva/Desktop/pracs/aup/ass2/sharedDir/temp2.txt
$ exit
logout
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ su - blah1
Password:
No directory, logging in with HOME=/
$ touch home/purva/Desktop/pracs/aup/ass2/sharedDir/temp3.txt
$ chmod 777 home/purva/Desktop/pracs/aup/ass2/sharedDir/temp3.txt
$ exit
logout
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ chmod 777 sharedDir/temp1.txt
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ls -l sharedDir/
total 0
-rwxrwxrwx 1 purva purva 0 Sep  9 19:34 temp1.txt
-rwxrwxrwx 1 blah  blah  0 Sep  9 19:35 temp2.txt
-rwxrwxrwx 1 blah1 blah1 0 Sep  9 19:36 temp3.txt
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ chmod +t sharedDir/
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ ls -ld sharedDir/
drwxrwxrwt 2 purva purva 4096 Sep  9 19:36 sharedDir/
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ su - blah
Password:
No directory, logging in with HOME=/
$ rm home/purva/Desktop/pracs/aup/ass2/sharedDir/temp1.txt
rm: cannot remove ‘home/purva/Desktop/pracs/aup/ass2/sharedDir/temp1.txt’: Operation not permitted
$ exit
logout
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ whoami
purva
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ gedit sharedDir/temp1.txt
```



```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$ su - blah1
Password:
No directory, logging in with HOME=/
$ cat home/purva/Desktop/pracs/aup/ass2/sharedDir/temp1.txt
hello
$ exit
logout
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass2$
```

Explanation :

1. Create a directory sharedDir and provide all the users read-write-execute access to it.
2. A directory named 'sharedDir' is created and read-write-execute access to this directory is given to all the users through chmod command.
3. Create multiple files in this directory (with different users) such that all users have read-write-execute access to them.
4. Now the files temp2.txt and temp3.txt are created by different users but have read-write-execute access on for all the users. This means that the user 'blah' can delete or rename the file created by user 'purva' or 'blah2'.
5. In order to avoid this, sticky bit can be set on the directory sharedDir.
6. Turn ON the sticky bit on the directory by using +t flag of chmod command.
7. As can be observed, a permission bit ‘t’ is introduced in the permission bits of the directory.
8. Now, if the user 'blah' tries to remove the file 'temp1.txt, here is what happens :
9. rm: cannot remove ‘home/purva/Desktop/pracs/aup/ass2/sharedDir/temp1.txt’: Operation not permitted
10. So we see that the operation was not permitted.
11. However, the different users continue to have read access to the files as demonstrated.
12. The contents of temp1.txt now include "hello" and this is printed appropriately when read by user 'blah1'.

Thus the desired shared directory sharedDir has been created which satisfies all criteria mentioned in the question.