Ls , cp

ls–l command

#include<stdio.h>

#include<dirent.h>

#include<sys/stat.h>

#include<pwd.h>

#include<grp.h>

#include<time.h>

int main()

{

DIR \*d;

struct dirent \*de;

struct stat buf;

int i,j;

char P[10]="rwxrwxrwx",AP[10]=" ";

struct passwd \*p;

struct group \*g;

struct tm \*t;

char time[26];

d=opendir(".");

readdir(d);

readdir(d);

while( (de=readdir(d))!=NULL)

{

stat(de->d\_name,&buf);

// File Type

if(S\_ISDIR(buf.st\_mode))

printf("d");

else if(S\_ISREG(buf.st\_mode))

printf("-");

else if(S\_ISCHR(buf.st\_mode))

printf("c");

else if(S\_ISBLK(buf.st\_mode))

printf("b");

else if(S\_ISLNK(buf.st\_mode))

printf("l");

else if(S\_ISFIFO(buf.st\_mode))

printf("p");

else if(S\_ISSOCK(buf.st\_mode))

printf("s");

//File Permissions P-Full Permissions AP-Actual Permissions

for(i=0,j=(1<<8);i<9;i++,j>>=1)

AP[i]= (buf.st\_mode & j ) ? P[i] : '-' ;

printf("%s",AP);

//No. of Hard Links

printf("%5d",buf.st\_nlink);

//User Name

p=getpwuid(buf.st\_uid);

printf(" %.8s",p->pw\_name);

//Group Name

g=getgrgid(buf.st\_gid);

printf(" %-8.8s",g->gr\_name);

//File Size

printf(" %8d",buf.st\_size);

//Date and Time of modification

t=localtime(&buf.st\_mtime);

strftime(time,sizeof(time),"%b %d %H:%M",t);

printf(" %s",time);

//File Name

printf(" %s\n",de->d\_name);

}

}

cp command

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <errno.h>

#include <sys/types.h>

#include <unistd.h>

#define BUF\_SIZE 8192

int main(int argc, char\* argv[]) {

int input\_fd, output\_fd;

descriptors \*/

ssize\_t ret\_in, ret\_out;

read() and write() \*/

char buffer[BUF\_SIZE];

/\* Input and output file

/\* Number of bytes returned by

/\* Character buffer \*/

/\* Are src and dest file name arguments missing \*/

if(argc != 3){

printf ("Usage: cp file1 file2");

return 1;

}

/\* Create input file descriptor \*/

input\_fd = open (argv [1], O\_RDONLY);

if (input\_fd ==-1) {

perror ("open");

return 2;

}

/\* Create output file descriptor \*/

output\_fd = open(argv[2], O\_WRONLY | O\_CREAT, 0644);

if(output\_fd ==-1){

perror("open");

return 3;

}

/\* Copy process \*/

while((ret\_in = read (input\_fd, &buffer, BUF\_SIZE)) > 0){

ret\_out = write (output\_fd, &buffer, (ssize\_t)

ret\_in);

if(ret\_out != ret\_in){

/\* Write error \*/

perror("write");

return 4;

}

}

/\* Close file descriptors \*/

close (input\_fd);

close (output\_fd);

return (EXIT\_SUCCESS);

}