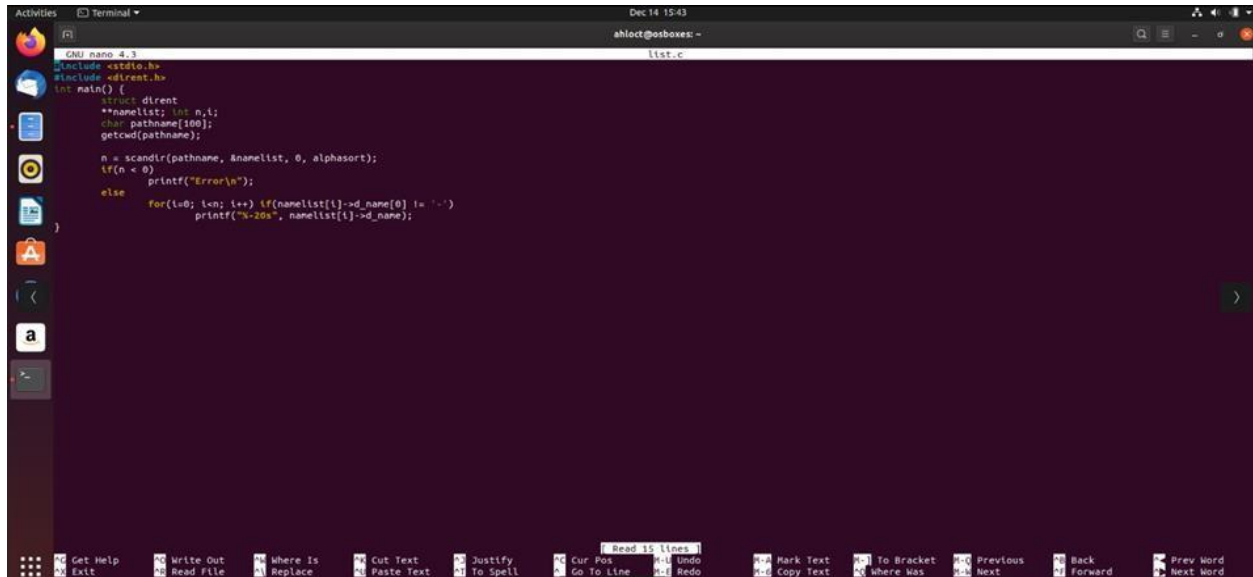


NAMA : Raihan Mazarul H

NIM : L200180162

KELAS : D

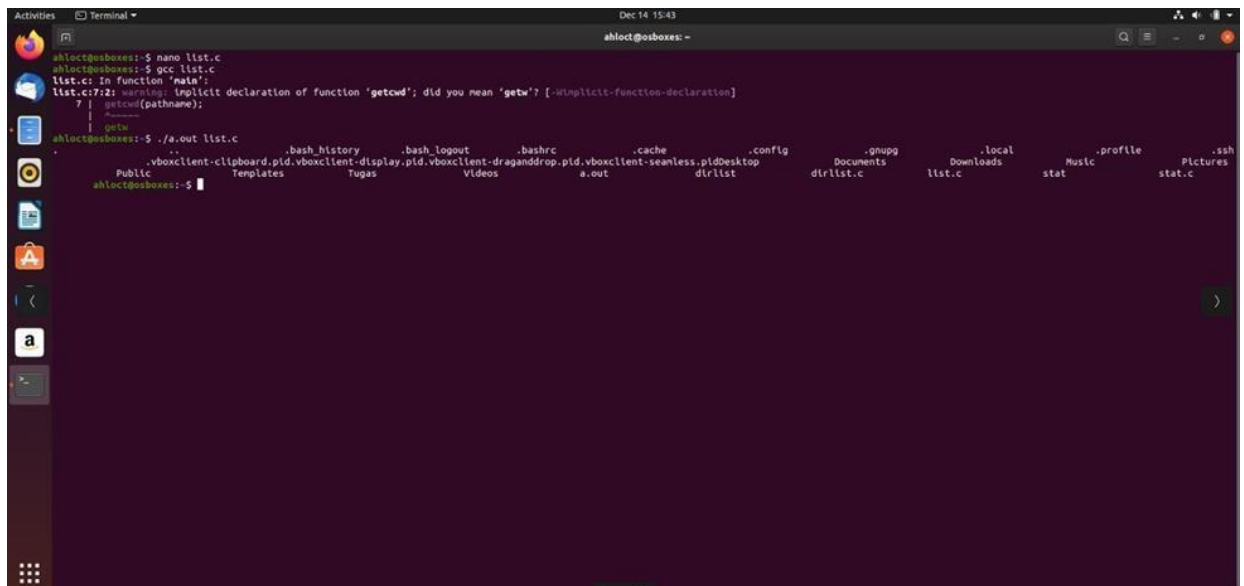
1. List



```
GNU nano 4.3 list.c
#include <stdio.h>
#include <dirent.h>

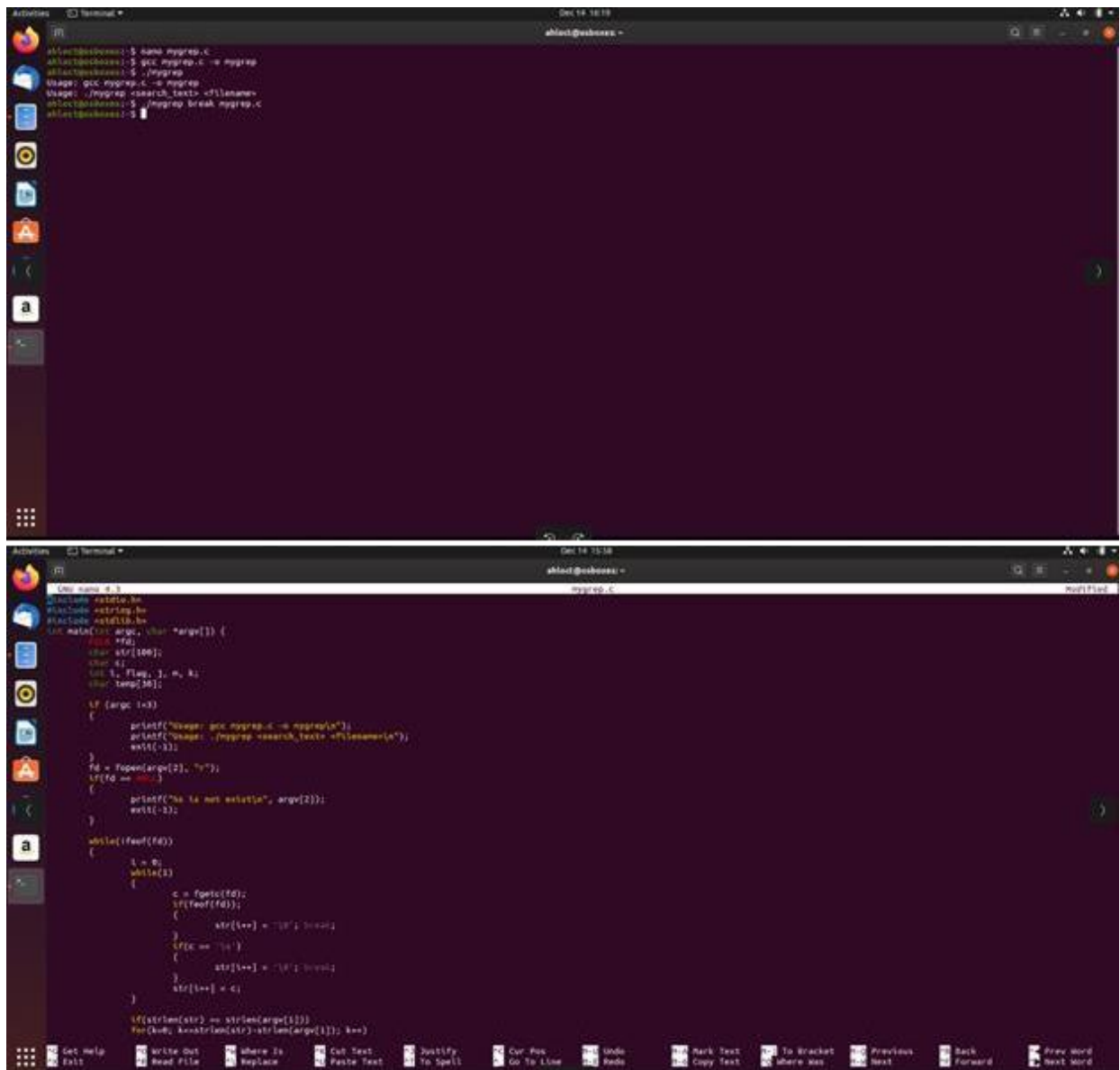
int main() {
    struct dirent
    **namelist; int n,l;
    char pathname[100];
    getcwd(pathname);

    n = scandir(pathname, Anamelist, 0, alphasort);
    if(n < 0)
        printf("Error\n");
    else
        for(l=0; l<n; l++) if(namelist[l]->d_name[0] != '.')
            printf("%-20s", namelist[l]->d_name);
}
```



```
ahloct@osboxes:~$ gcc list.c
list.c:7:21: warning: implicit declaration of function 'getcwd'; did you mean 'getw'? [-Wimplicit-function-declaration]
7 |     getcwd(pathname);
  |     ^~~~~~
  |     getw
ahloct@osboxes:~$ ./a.out list.c
..
.bash_history      .bash_logout      .bashrc           .cache            .config           .gnupg            .local            .profile          .ssh
.vboxclient-clipboard.pid.vboxclient-display.pid.vboxclient-draganddrop.pid.vboxclient-seamless.pid.Desktop
Public            Templates         Tugas             Videos            a.out             dirlist           dirlist.c         list.c            stat
Pictures          stat.c
```

2. Mygrep



```
ahlect@ubuntu:~$ nano mygrep.c
ahlect@ubuntu:~$ gcc mygrep.c -o mygrep
ahlect@ubuntu:~$ ./mygrep
Usage: gcc mygrep.c -o mygrep
Usage: ./mygrep <search_text> <filename>
ahlect@ubuntu:~$ ./mygrep break mygrep.c
ahlect@ubuntu:~$
```

```
ahlect@ubuntu:~$ nano mygrep.c
mygrep.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    char *str;
    char str[100];
    char *c;
    int i, flag, j, m, k;
    char temp[20];

    if (argc != 3)
    {
        printf("Usage: gcc mygrep.c -o mygrep\n");
        printf("Usage: ./mygrep <search_text> <filename>\n");
        exit(-1);
    }

    fd = fopen(argv[2], "r");
    if (fd == NULL)
    {
        printf("No file exists\n", argv[2]);
        exit(-1);
    }

    while (!feof(fd))
    {
        i = 0;
        while (1)
        {
            c = fgetc(fd);
            if (feof(fd))
            {
                str[i++] = '\0'; break;
            }
            if (c == '\n')
            {
                str[i++] = '\0'; break;
            }
            str[i++] = c;
        }

        if (strcmp(str) == strcmp(argv[1]))
        {
            printf("%s\n", str);
        }
    }
}
```

Activities Terminal Dec 14 15:58
ahloct@osboxes: ~
mygrep.c Modified

```
GNU nano 4.3 mygrep.c
printf("Usage: ./mygrep <search_text> <filenames>\n");
exit(-1);

fd = fopen(argv[2], "r");
if(fd == NULL)
{
    printf("File is not exist\n", argv[2]);
    exit(-1);
}

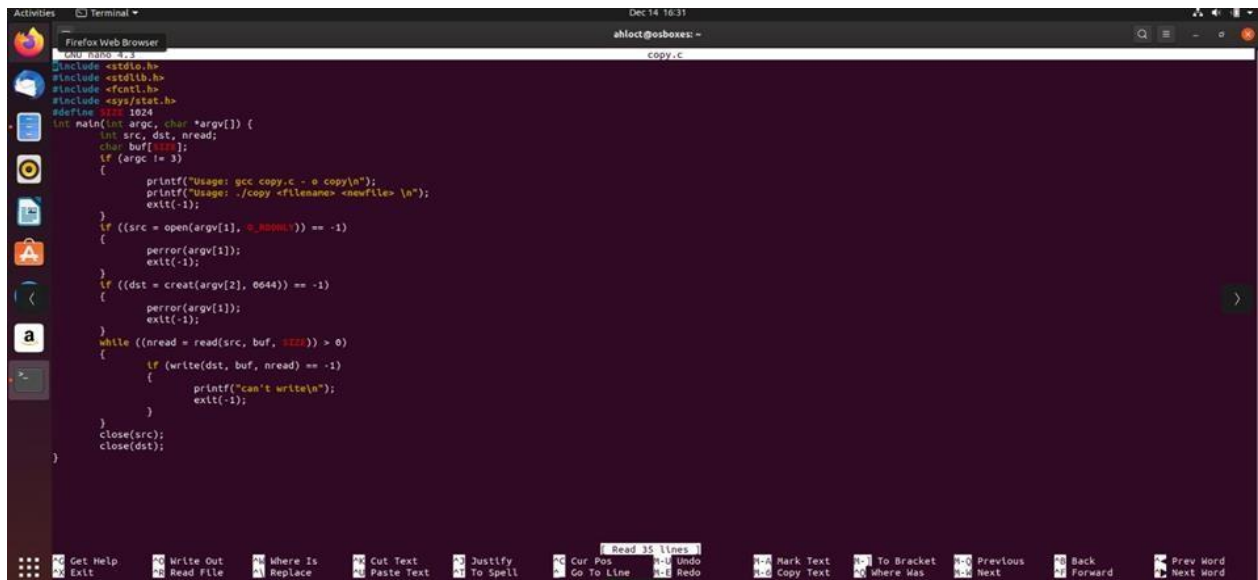
while(!feof(fd))
{
    t = 0;
    while(1)
    {
        c = fgetc(fd);
        if(feof(fd))
        {
            str[t++] = '\0'; break;
        }
        if(c == '\n')
        {
            str[t++] = '\0'; break;
        }
        str[t++] = c;
    }

    if(strlen(str) == strlen(argv[1]))
    for(k=0; k<strlen(str)-strlen(argv[1]); k++)
    {
        for(m=0; m<strlen(argv[1]);m++)
        temp[m] = str[k+m];
        if(strcmp(temp, argv[1]) == 0)
        {
            printf("%s\n", str);
            break;
        }
    }
}

1
```

Get Help Exit Write Out Read File Where Is Replace Cut Text Paste Text Justify To Spell Cur Pos Go To Line Undo Redo Mark Text Copy Text To Bracket Where Was Previous Next Back Forward Prev Word Next Word

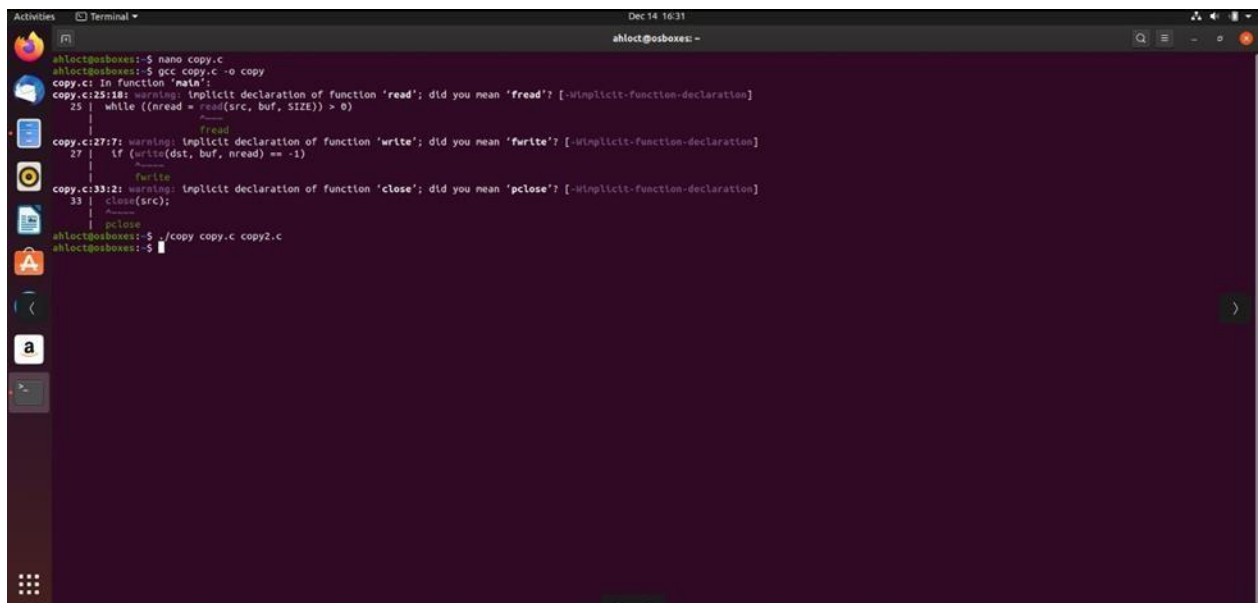
3. Copy



The screenshot shows a terminal window with the file 'copy.c' open in a text editor. The code is a C program that implements a simple file copy utility. It includes standard headers like `<stdio.h>`, `<stdlib.h>`, `<fcntl.h>`, and `<sys/stat.h>`. It defines a buffer size of 1024 bytes. The `main` function takes two command-line arguments: the source file and the destination file. It checks for the correct number of arguments and prints usage information if necessary. It then opens the source file in read mode and the destination file in write mode. A `while` loop reads data from the source file in chunks of 1024 bytes and writes it to the destination file. Error handling is provided for file opening and writing failures. The program closes both files before returning.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#define SIZE 1024

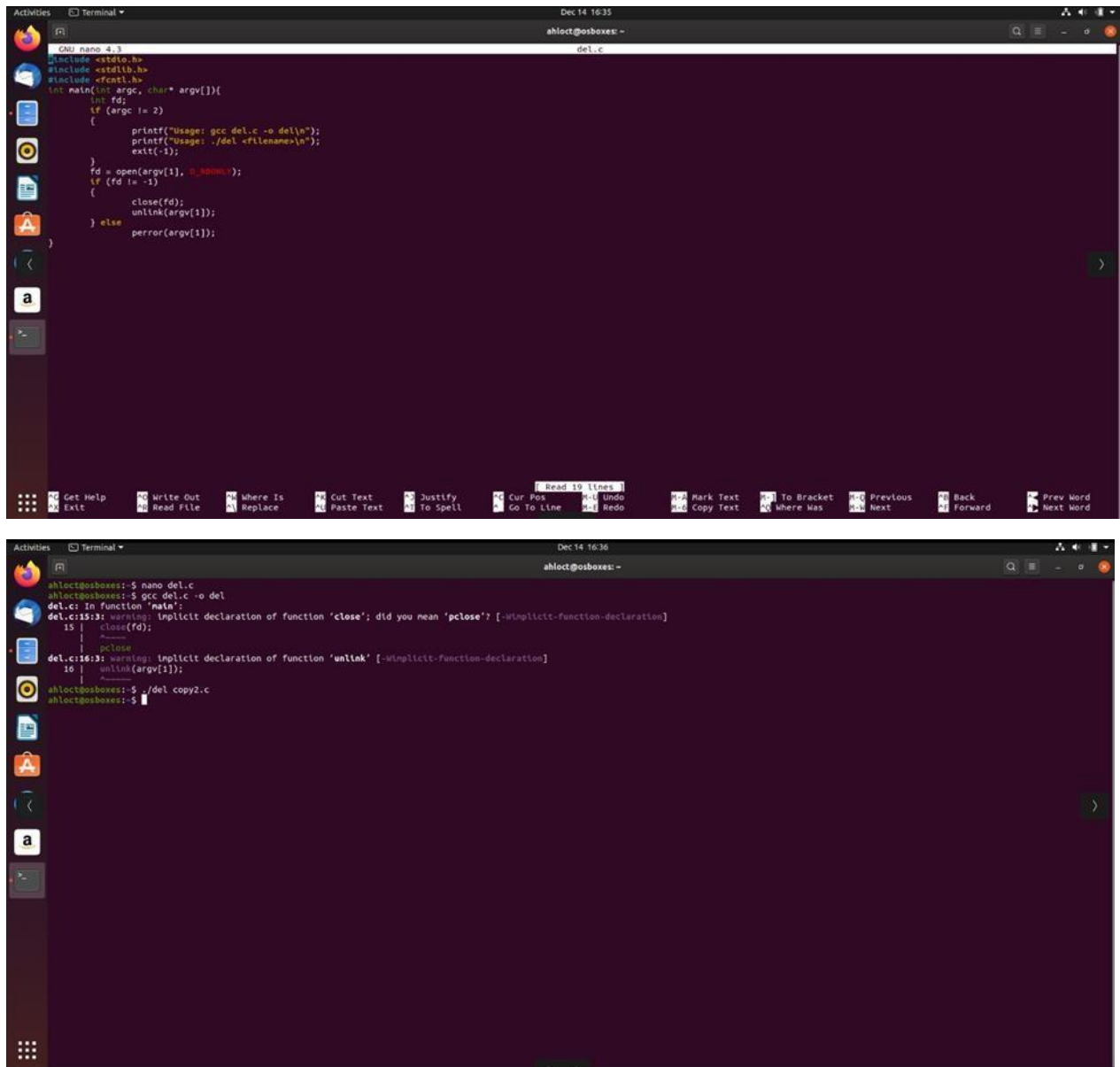
int main(int argc, char *argv[]) {
    int src, dst, nread;
    char buf[SIZE];
    if (argc != 3) {
        printf("Usage: gcc copy.c -o copy\n");
        printf("Usage: ./copy <filename> <newfile> \n");
        exit(-1);
    }
    if ((src = open(argv[1], O_RDONLY)) == -1) {
        perror(argv[1]);
        exit(-1);
    }
    if ((dst = creat(argv[2], 0644)) == -1) {
        perror(argv[2]);
        exit(-1);
    }
    while ((nread = read(src, buf, SIZE)) > 0) {
        if (write(dst, buf, nread) == -1) {
            printf("can't write\n");
            exit(-1);
        }
    }
    close(src);
    close(dst);
}
```



The screenshot shows the terminal window after the program has been compiled and executed. The user runs `gcc copy.c -o copy` to compile the program. The compiler outputs several warnings about implicit function declarations for `read`, `write`, and `close`. The user then runs `./copy copy.c copy2.c` to execute the program, which successfully copies the source file into a new file named `copy2.c`.

```
ahloct@osboxes:~$ gcc copy.c -o copy
copy.c: In function 'main':
copy.c:25:18: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
25 | while ((nread = read(src, buf, SIZE)) > 0)
    |                  ^~~~~
    |                  fread
copy.c:27:7: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
27 | if (write(dst, buf, nread) == -1)
    |     ^~~~~
    |     fwrite
copy.c:33:12: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
33 | close(src);
    |     ^~~~~
    |     pclose
ahloct@osboxes:~$ ./copy copy.c copy2.c
ahloct@osboxes:~$
```

4. Del



The first screenshot shows the nano text editor editing a file named `del.c`. The code is as follows:

```
GNU nano 4.3 del.c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main(int argc, char* argv[]){
    int fd;
    if (argc != 2)
    {
        printf("Usage: gcc del.c -o del\n");
        printf("Usage: ./del <filename>\n");
        exit(-1);
    }
    fd = open(argv[1], O_WRONLY);
    if (fd != -1)
    {
        close(fd);
        unlink(argv[1]);
    } else
        perror(argv[1]);
}
```

The second screenshot shows the terminal output after running the program. The user has compiled the program with `gcc del.c -o del` and executed it with `./del copy2.c`. The output shows two warnings: an implicit declaration of `close` and an implicit declaration of `unlink`.

```
ahloct@osboxes: ~$ nano del.c
ahloct@osboxes: ~$ gcc del.c -o del
del.c: In function 'main':
del.c:15:13: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
15 |     close(fd);
    |     ^~~~~
del.c:16:13: warning: implicit declaration of function 'unlink' [-Wimplicit-function-declaration]
16 |     unlink(argv[1]);
    |
ahloct@osboxes: ~$ ./del copy2.c
ahloct@osboxes: ~$
```