

PRATIUM PEMROGRAMAN BERBASIS OBJEK

PRATIUM 10



Disusun Oleh :

Robby Febrian Saputro

L200210250

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2021/2022**

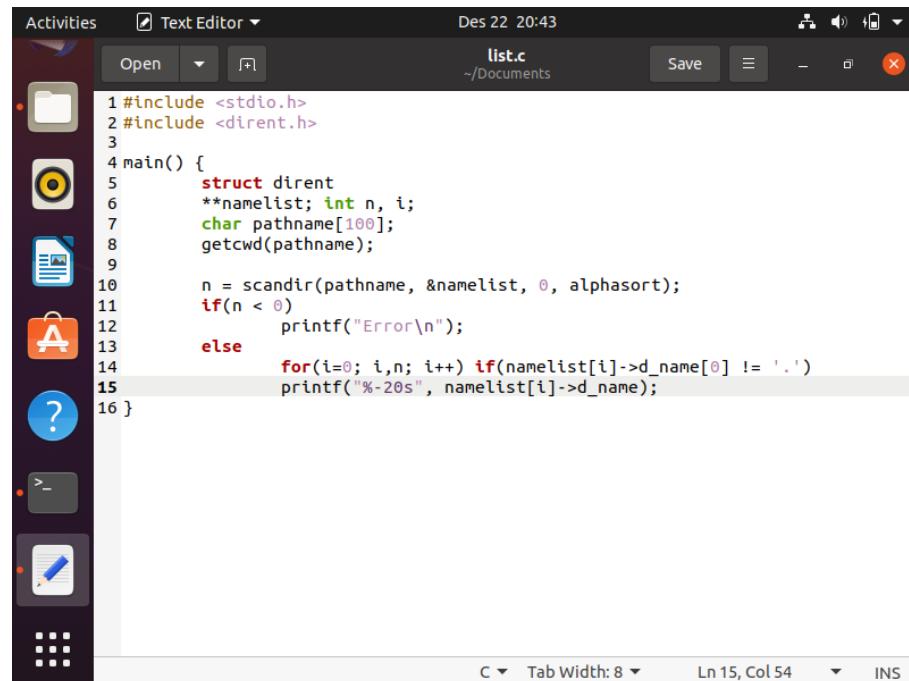
LANGKAH KERJA

1. Program untuk mensimulasi perintah “ls”

Membuat kode program dengan algorithm sebagai berikut :

1. Menyimpan ‘path’ dari direktori kerja saat ini menggunakan perintah system call ‘getcwd’
2. Membaca isi direktori dari path di atas menggunakan perintah system call ‘scandir’ dan mengurutkan hasil pembacaannya dan menyimpannya dalam sebuah variabel array.
3. Menampilkan nama direktori (dname) dan nama file didalamnya jika file atau direktori tersebut tidak memiliki properti ‘HIDE’.
4. Stop

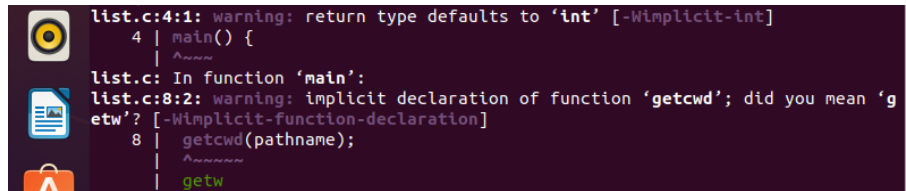
• KODE PROGRAM



```
Activities Text Editor Des 22 20:43 list.c ~/Documents Save
1 #include <stdio.h>
2 #include <dirent.h>
3
4 main() {
5     struct dirent
6     **namelist; int n, i;
7     char pathname[100];
8     getcwd(pathname);
9
10    n = scandir(pathname, &namelist, 0, alphasort);
11    if(n < 0)
12        printf("Error\n");
13    else
14        for(i=0; i,n; i++) if(namelist[i]->d_name[0] != '.')
15            printf("%-20s", namelist[i]->d_name);
16 }
```

C Tab Width: 8 Ln 15, Col 54 INS

- **OUTPUT**



```
list.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
4 | main() {
  |
list.c: In function 'main':
list.c:8:2: warning: implicit declaration of function 'getcwd'; did you mean 'getw'? [-Wimplicit-function-declaration]
8 |   getcwd(pathname);
  |   ^~~~~~
  |   getw
```

2. Program untuk mensimulasi perintah 'grep'.

Membuat kode program dengan algorithm sebagai berikut:

1. Gunakan nama file yang diberikan dalam argumen command-line
2. Buka file dalam mode 'read-only' menggunakan perintah system call 'open'
3. Jika file tidak ada, keluar program, stop
4. Misal panjang string yang dicari adalah n.
5. Baca file perbaris sampai akhir file (END-OF-FILE), untuk setiap baris lakukan hal-hal berikut: (a) Periksa untuk mencari string dalam baris tersebut dengan dalam range 1-n, 2-n+1, dan seterusnya, (b) Jika string ditemukan tampilan baris tersebut di layar.
6. Tutup file menggunakan perintah 'close'.
7. Stop

- **KODE PROGRAM**



The screenshot shows a Linux desktop with a sidebar on the left containing various application icons. The main window is a text editor titled "Text Editor" with a file named "mygrep.c" open at the path "~/Documents". The code is a C program that implements a simple grep functionality. It includes headers for stdio, string, and stdlib. The main function takes command-line arguments and checks if they are correct. It then opens a file and reads it character by character, comparing substrings with the search text provided in the arguments. If a match is found, it prints the matching line.

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 main(int argc, char*argv[]){
5     FILE *fd;
6     char str[100];
7     char c;
8     int i, flag, j, m, k;
9     char temp[30];
10
11     if(argc != 3)
12     {
13         printf("Usage: gcc mygrep.c -o mygrep\n");
14         printf("Usage: ./mygrep <search_text> <filename>\n");
15         exit(-1);
16     }
17
18     fd = fopen(argv[2], "r");
19     if(fd == NULL)
20     {
21         printf("%s is not exist\n", argv[2]);
22         exit(-1);
23     }
24     while(!feof(fd))
25     {
26         i = 0;
27         while(1)
28         {
29             c = fgetc(fd);
30             if(feof(fd))
31             {
32                 str[i++] = '\0'; break;
33             }
34             if(c == '\n')
35             {
36                 str[i++] = c;
37             }
38
39             if(strlen(str) >= strlen(argv[1]))
40             for(k=0; k<=strlen(str)-strlen(argv[1]); k++)
41             {
42                 for(m=0; m<strlen(argv[1]); m++)
43                     temp[m] = str[k+m];
44                 temp[m] = '\0';
45                 if(strcmp(temp,argv[1]) == 0)
46                 {
47                     printf("%s\n", str);
48                     break;
49                 }
50             }
51         }
52     }
```

- **OUTPUT**

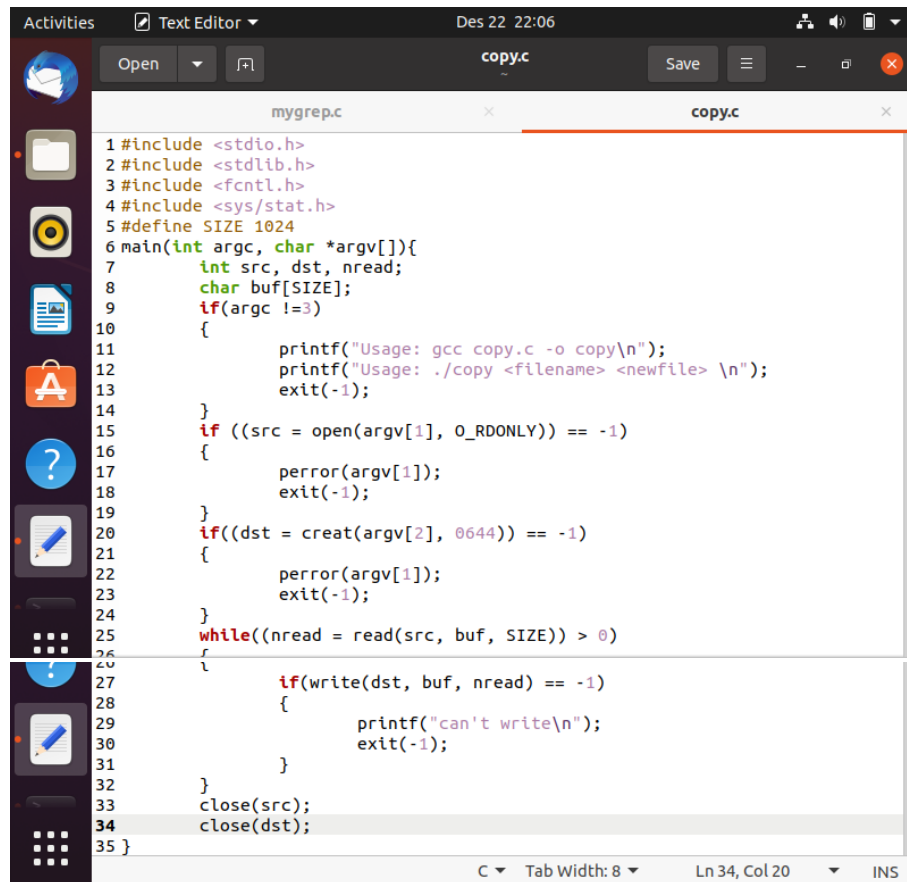
```
mygrep.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
4 | main(int argc, char*argv[]){
  | ^
mygrep.c: In function 'main':
mygrep.c:29:17: error: expected ';' before ')' token
29 |     c = fgetc(fd));
    |                   ^
mygrep.c:29:17: error: expected statement before ')' token
mygrep.c:45:28: warning: passing argument 2 of 'strcmp' makes pointer from integer without a cast [-Wint-conversion]
45 |     if(strcmp(temp,argv[1] == 0)
    |                   ~~~~~^~~~~~
    |                   |
    |                   int
In file included from mygrep.c:2:
/usr/include/string.h:137:50: note: expected 'const char *' but argument is of type 'int'
137 | extern int strcmp (const char *__s1, const char *__s2)
    |                                  ~~~~~^~~~~~
mygrep.c:45:33: error: expected ')' before '{' token
45 |     if(strcmp(temp,argv[1] == 0)
    |                               ^
46 |     {
    |     ~
mygrep.c:50:4: error: expected expression before '}' token
50 | }
    | ^
```

3. Program untuk mensimulasi perintah 'cp'.

Membuat kode program dengan algorithm sebagai berikut:

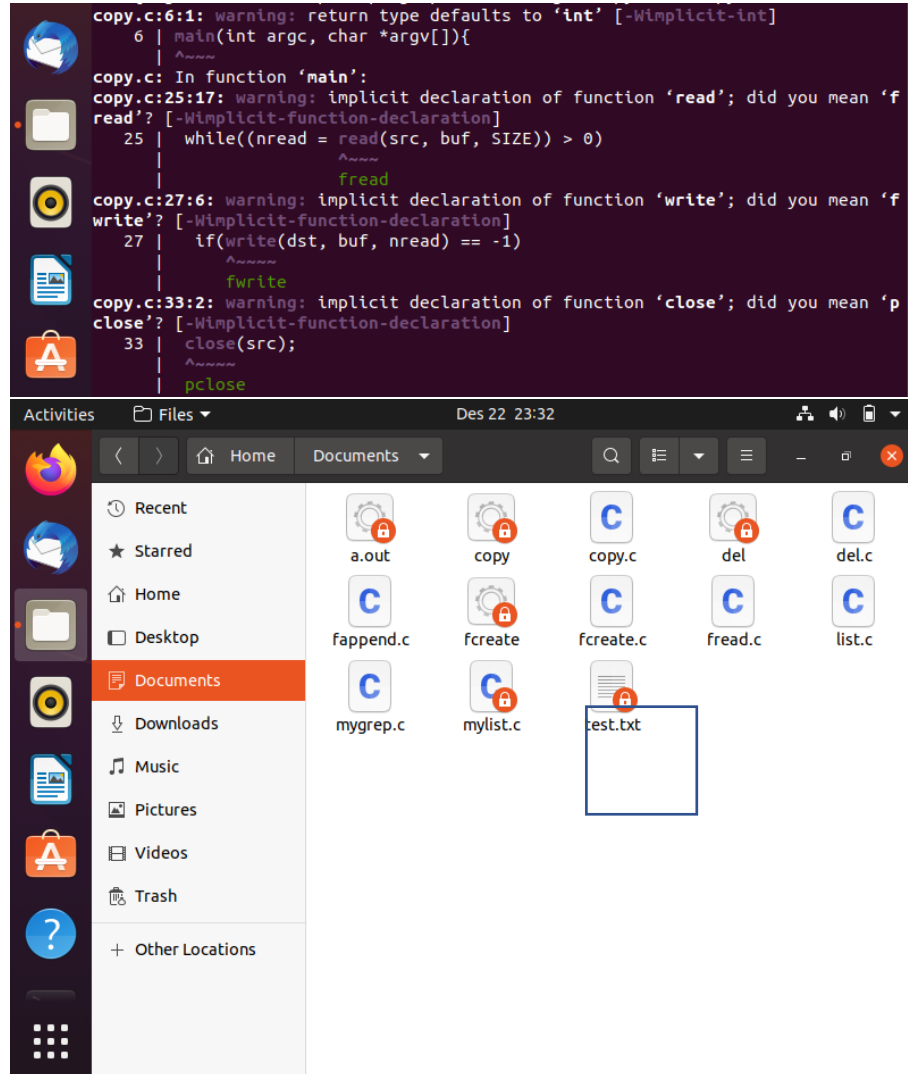
1. Gunakan nama file untuk sumber dan tujuan dari argumen yang diberikan dalam command line.
2. Deklarasi sebuah buffer berukuran 1 KB
3. Buka file sumber dalam mode 'read-only' menggunakan fungsi 'open'
4. Jika file sumber tidak ditemukan, stop keluar dari program
5. Membuat file baru sebagai file target dengan menggunakan perintah 'creat'.
6. Jika proses pembuatan file gagal, stop keluar dari program.
7. Proses penyalinan (copy) file dilakukan dengan cara berikut: (a) Membaca 1KB data dari file sumber dan menyimpan hasilnya dalam buffer menggunakan perintah 'read'. (b) Menuliskan isi buffer dalam file target menggunakan perintah 'write'. (c) Jika bertemu dengan kode 'END-OF-FILE' lanjut ke nomor 8, yang lain kembali ke perintah (a)
8. Tutup file sumber dan target menggunakan perintah 'close'.
9. Stop

• KODE PROGRAM



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <sys/stat.h>
5 #define SIZE 1024
6 main(int argc, char *argv[]){
7     int src, dst, nread;
8     char buf[SIZE];
9     if(argc !=3)
10     {
11         printf("Usage: gcc copy.c -o copy\n");
12         printf("Usage: ./copy <filename> <newfile> \n");
13         exit(-1);
14     }
15     if ((src = open(argv[1], O_RDONLY)) == -1)
16     {
17         perror(argv[1]);
18         exit(-1);
19     }
20     if((dst = creat(argv[2], 0644)) == -1)
21     {
22         perror(argv[1]);
23         exit(-1);
24     }
25     while((nread = read(src, buf, SIZE)) > 0)
26     {
27         if(write(dst, buf, nread) == -1)
28         {
29             printf("can't write\n");
30             exit(-1);
31         }
32     }
33     close(src);
34     close(dst);
35 }
```

- **OUTPUT (MENGCOPY FILE LIST.C DENGAN NAMA BARU MYLIST.C)**

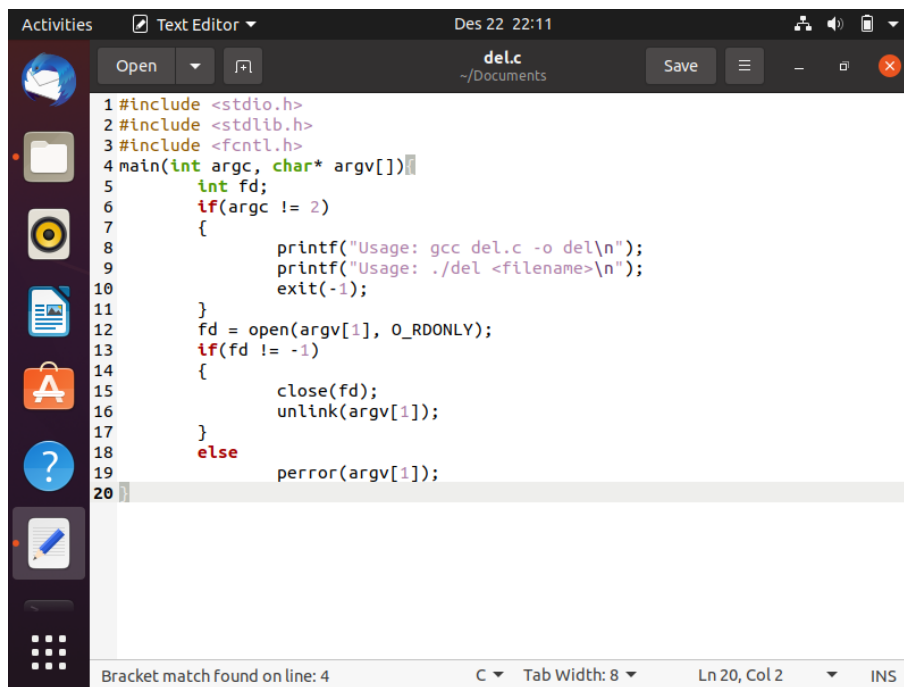


4. Program untuk mensimulasi perintah 'rm'.

Membuat kode program dengan algorithm sebagai berikut:

1. Gunakan nama file yang diberikan dalam argument command line
2. Buka file dalam mode 'read-only' menggunakan perintah 'read'
3. Jika file tidak ditemukan, stop keluar program
4. Tutup file menggunakan perintah 'close'
5. Menghapus file menggunakan perintah 'unlink'
6. Stop

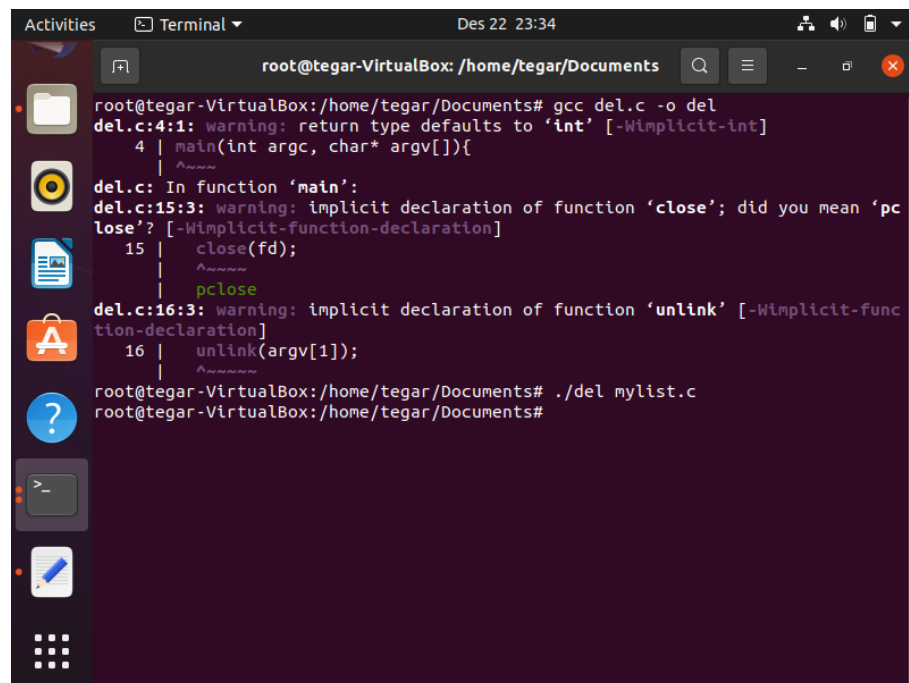
• KODE PROGRAM



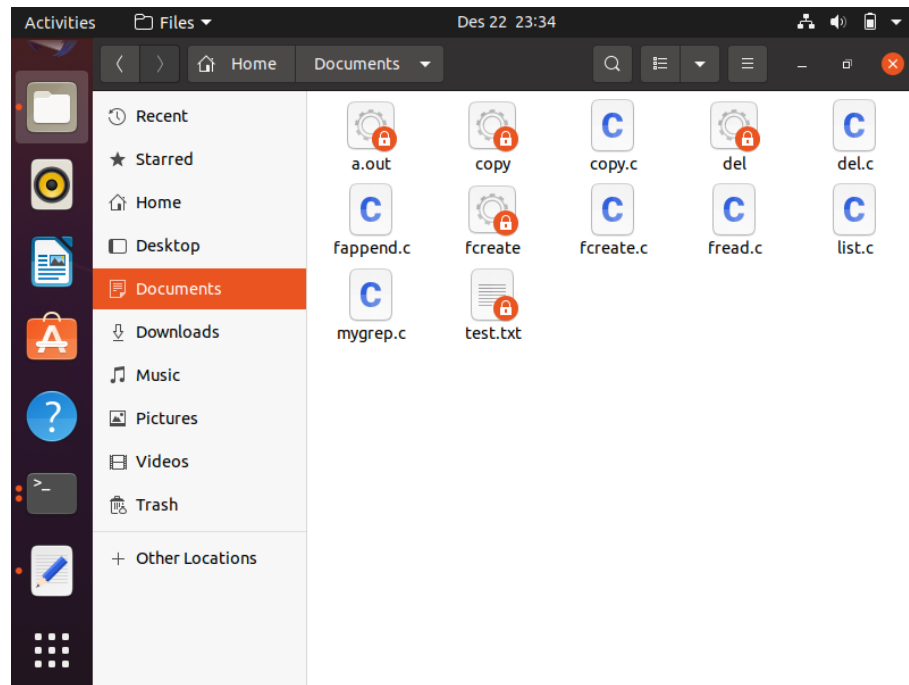
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 main(int argc, char* argv[])
5 {
6     if(argc != 2)
7     {
8         printf("Usage: gcc del.c -o del\n");
9         printf("Usage: ./del <filename>\n");
10        exit(-1);
11    }
12    fd = open(argv[1], O_RDONLY);
13    if(fd != -1)
14    {
15        close(fd);
16        unlink(argv[1]);
17    }
18    else
19        perror(argv[1]);
20 }
```

Bracket match found on line: 4 C Tab Width: 8 Ln 20, Col 2 INS

- **OUTPUT (MENGHAPUS FILE MYLIST.C)**



```
root@tegar-VirtualBox: /home/tegar/Documents
root@tegar-VirtualBox:/home/tegar/Documents# gcc del.c -o del
del.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
4 | main(int argc, char* argv[])
  | ~~~~~^
del.c: In function 'main':
del.c:15:3: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
15 |     close(fd);
    |     ^~~~~~
del.c:16:3: warning: implicit declaration of function 'unlink' [-Wimplicit-function-declaration]
16 |     unlink(argv[1]);
    |     ^~~~~~
root@tegar-VirtualBox:/home/tegar/Documents# ./del mylist.c
root@tegar-VirtualBox:/home/tegar/Documents#
```

5. HASIL

Telah dibuat dan diperiksa empat program simulasi dengan menggunakan perintah dalam system-call, terdiri atas perintah 'ls', 'grep', 'cp', dan 'rm'.

6. Kesimpulan

Perintah-perintah dalam system call, dapat digunakan untuk mensimulasi sebagian besar perintah dalam sistem operasi linux. Namun demikian mensimulasi sebuah perintah dengan semua opsi yang terdapat didalamnya merupakan pekerjaan yang melelahkan.