

PRAKTIKUM SISTEM OPERASI
MODUL 10 : SIMULASI COMMAND



Disusun oleh:

AFIFAH GH AISANI IMANA

L200190198

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2019/2020

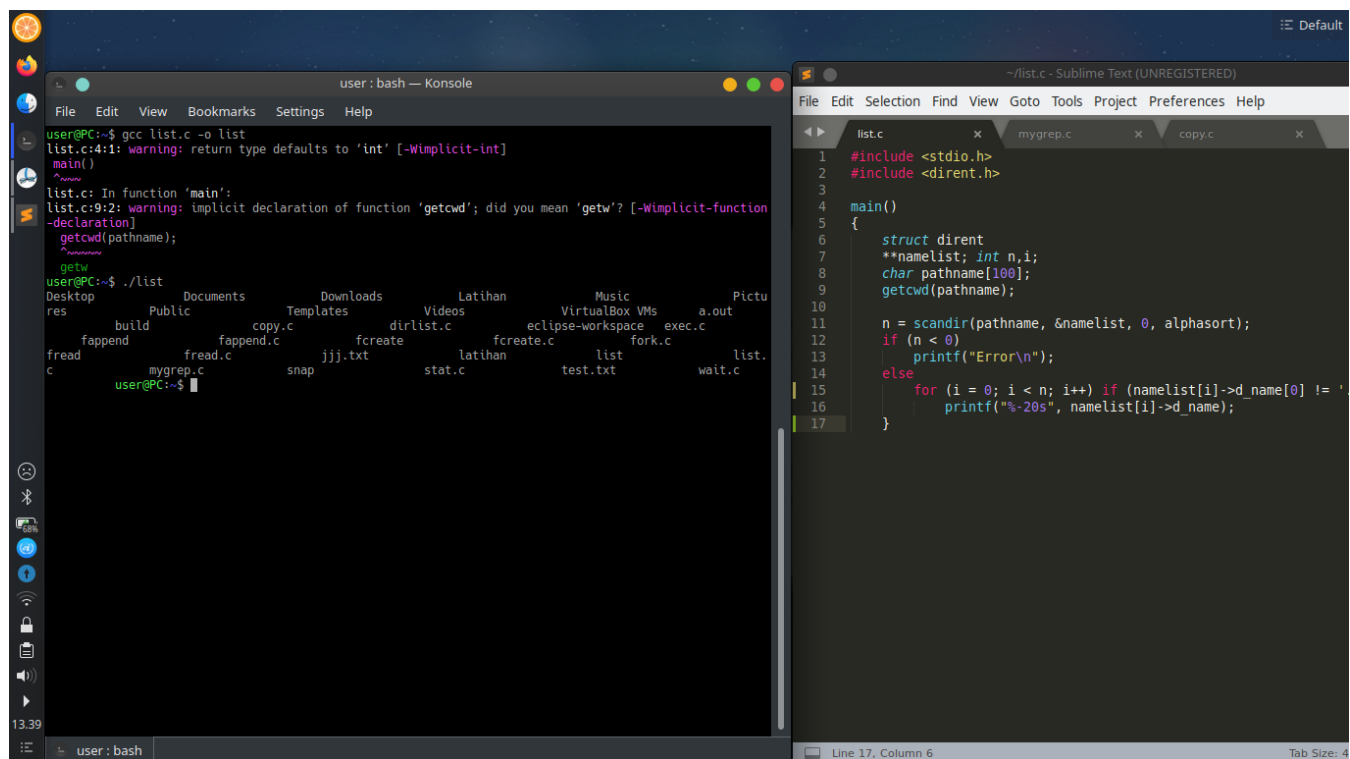
Langkah Kerja.

1. Program untuk mensimulasikan perintah “ls”

Membuat kode program dengan algorithm sebagai berikut :

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

Kode Program list.c dan tampilan saat di run pada terminal.



The image shows a terminal window on the left and a code editor on the right. The terminal window displays the compilation and execution of a C program named 'list.c'. The code editor shows the source code of 'list.c'.

```
user@PC:~$ gcc list.c -o list
list.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~~~
list.c: In function 'main':
list.c:9:2: warning: implicit declaration of function 'getcwd'; did you mean 'getw'? [-Wimplicit-function-declaration]
  getcwd(pathname);
  ^~~~~~
user@PC:~$ ./list
Desktop    Documents  Downloads  Latihan    Music      Pictu
res        Public    Templates  Videos    VirtualBox VMs  a.out
build      fappend   copy.c     dirlist.c  eclipse-workspace  exec.c
fread      fappend.c fcreate.c  fcreate.c  fork.c        list.
c          mygrep.c  snap       stat.c     test.txt      wait.c
user@PC:~$
```

```
1  #include <stdio.h>
2  #include <dirent.h>
3
4  main()
5  {
6      struct dirent
7      **namelist; int n,i;
8      char pathname[100];
9      getcwd(pathname);
10
11      n = scandir(pathname, &namelist, 0, alphasort);
12      if (n < 0)
13          printf("Error\n");
14      else
15          for (i = 0; i < n; i++) if (namelist[i]->d_name[0] != '.')
16              printf("%-20s", namelist[i]->d_name);
17  }
```

2. Program untuk mensimulasikan perintah “grep”

Membuat kode program dengan algorithm sebagai berikut :

- Gunakan nama file yang diberikan dalam argumen command-line
- Buka file dalam mode ‘read-only’ menggunakan perintah system call ‘open’
- Jika file tidak ada, keluar program, stop
- Misal panjang string yang dicari adalah n.
- 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.
- Tutup file menggunakan perintah ‘close’.
- Stop

Kode Program mygrep.c dan tampilan saat di run pada terminal.

The image shows a terminal window on the left and a code editor window on the right. The terminal window displays the output of the compilation and execution of the mygrep.c program. The code editor window shows the source code of mygrep.c, which implements a simple grep-like functionality. The code includes headers for stdio, string, and stdlib, and defines a main function that takes command-line arguments. It checks for the correct number of arguments, opens the file specified in the second argument, and then reads the file line by line, searching for the string specified in the first argument. If a match is found, the line is printed to the screen.

```
user: bash — Konsole
root@PC:/home/user# gcc mygrep.c -o mygrep
mygrep.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(int argc, char const *argv[])
^~~~~~
root@PC:/home/user# ./mygrep Nama test.txt
Nama = Afifah Ghaisani Inana, NIM = L200190198,
root@PC:/home/user#
```

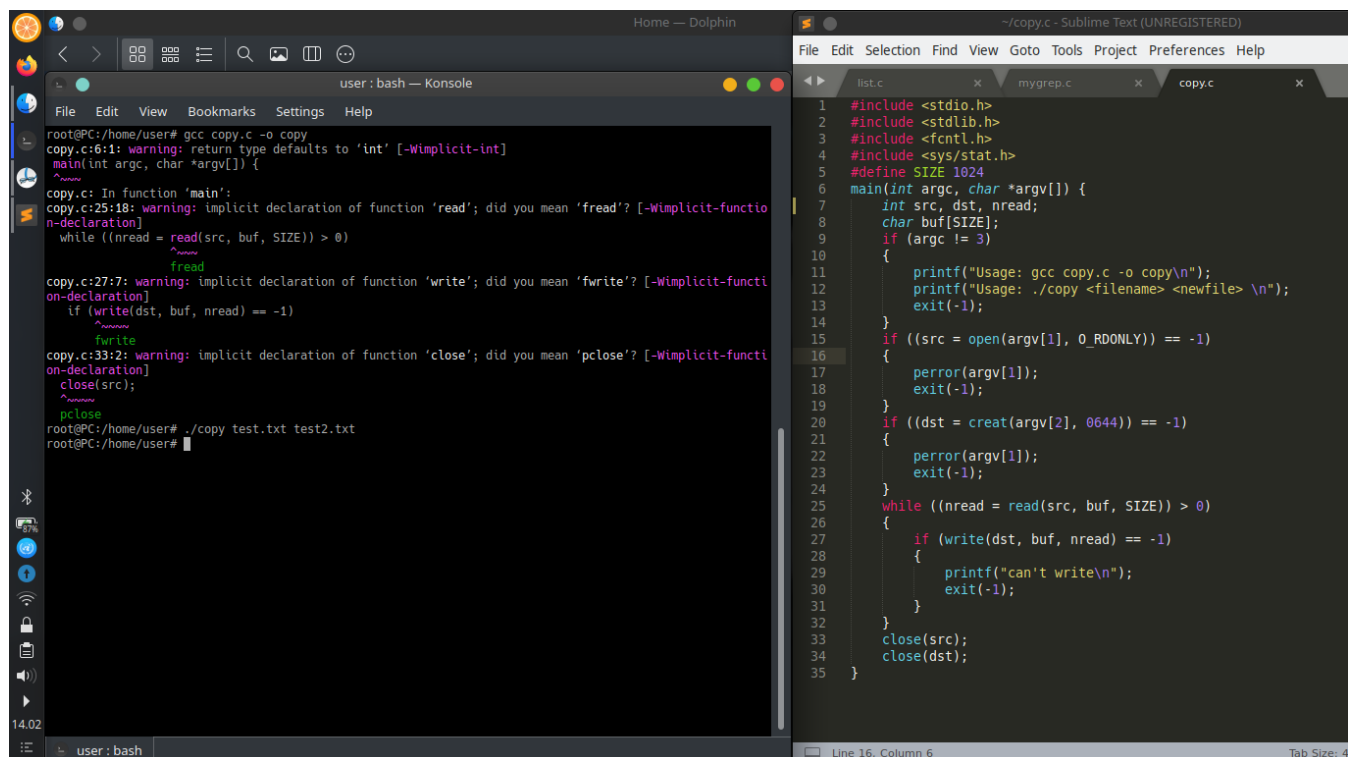
```
~/mygrep.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
list.c mygrep.c copy.c
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 main(int argc, char const *argv[])
5 {
6     FILE *fd;
7     char str[100];
8     char c;
9     int i, flag, j, m, k;
10    char temp[30];
11
12    if (argc != 3)
13    {
14        printf("Usage: gcc mygrep.c -o mygrep\n");
15        printf("Usage: ./mygrep <search_text> <filename>\n");
16        exit(-1);
17    }
18
19    fd = fopen(argv[2], "r");
20    if (fd == NULL)
21    {
22        printf("%s is not exist\n", argv[2]);
23        exit(-1);
24    }
25
26    while(!feof(fd))
27    {
28        i=0;
29        while(1)
30        {
31            c = fgetc(fd);
32            if (feof(fd))
33            {
34                str[i++] = '\0'; break;
35            }
36            if (c == '\n')
37            {
38                str[i++] = '\0'; break;
39            }
40
41        }
42
43        if (strlen(str) >= strlen(argv[1]))
44            for (k=0; k <= strlen(str)-strlen(argv[1]); k++)
45            {
46                for(m=0; m<strlen(argv[1]); m++)
47                    temp[m] = str[k+m];
48                temp[m] = '\0';
49                if(strcmp(temp, argv[1]) == 0)
50                {
51                    printf("%s\n", str);
52                    break;
53                }
54            }
55        }
56    }
```

3. Program untuk mensimulasikan perintah “cp”

Membuat kode program dengan algorithm sebagai berikut :

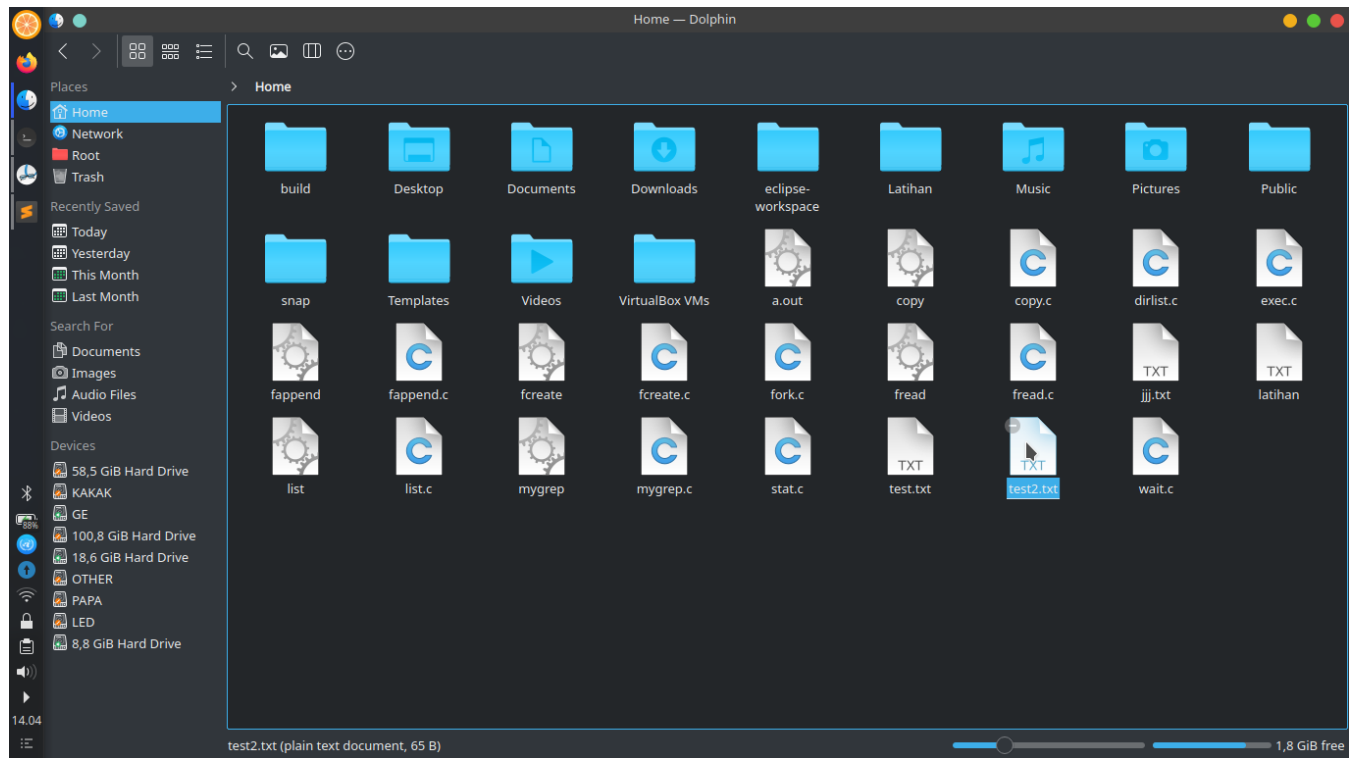
- Gunakan nama file untuk sumber dan tujuan dari argumen yang diberikan dalam command line.
- Deklarasi sebuah buffer berukuran 1 KB
- Buka file sumber dalam mode ‘read-only’ menggunakan fungsi ‘open’
- Jika file sumber tidak ditemukan, stop keluar dari program
- Membuat file baru sebagai file target dengan menggunakan perintah ‘creat’.
- Jika proses pembuatan file gagal, stop keluar dari program.
- 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) Tutup file sumber dan target menggunakan perintah ‘close’.
- Stop

Kode Program copy.c dan tampilan saat di run pada terminal.



```
user: bash — Konsole
root@PC:/home/user# gcc copy.c -o copy
copy.c:6:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(int argc, char *argv[]) {
^~~~~
copy.c: In function 'main':
copy.c:25:18: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
while ((nread = read(src, buf, SIZE)) > 0)
                 ^~~~~
copy.c:27:7: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
if (write(dst, buf, nread) == -1)
    ^~~~~
copy.c:33:2: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
close(src);
^~~~~
pclose
root@PC:/home/user# ./copy test.txt test2.txt
root@PC:/home/user#
```

```
list.c x mygrep.c x copy.c x
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 }
```

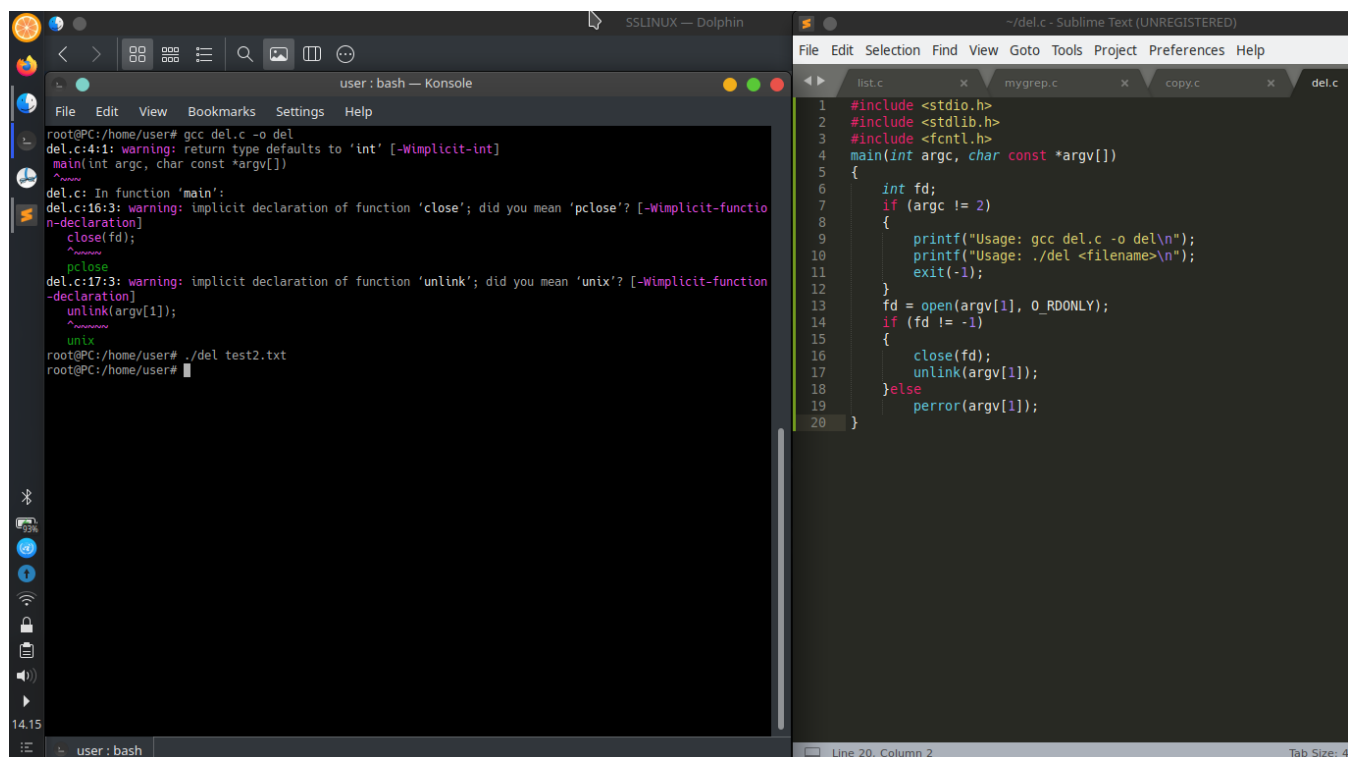


4. Program untuk mensimulasikan perintah “rm”

Membuat kode program dengan algorithm sebagai berikut :

- Gunakan nama file yang diberikan dalam argumen command line
- Buka file dalam mode ‘read-only’ menggunakan perintah ‘read’
- Jika file tidak ditemukan, stop keluar program
- Tutup file menggunakan perintah ‘close’
- Menghapus file menggunakan perintah ‘unlink’
- Stop

Kode Program del.c dan tampilan saat di run pada terminal.



The image shows a dual-pane window. The left pane is a terminal window titled 'user: bash — Konsole' with a menu bar (File, Edit, View, Bookmarks, Settings, Help). It shows the compilation of 'del.c' using 'gcc del.c -o del'. The output includes several warnings: 'warning: return type defaults to 'int' [-Wimplicit-int]', 'warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]', 'warning: implicit declaration of function 'unlink'; did you mean 'unix'? [-Wimplicit-function-declaration]', and 'warning: implicit declaration of function 'unix' [-Wimplicit-function-declaration]'. The user then runs './del test2.txt' and the prompt returns. The right pane is a code editor titled '~/.del.c - Sublime Text (UNREGISTERED)' with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help). It shows the source code for 'del.c' with line numbers 1 through 20. The code includes headers for <stdio.h>, <stdlib.h>, and <fcntl.h>. The main function takes argc and argv, checks for correct usage (argc == 2), opens the file in read-only mode (O_RDONLY), prints usage if it fails, closes the file, and then unlinks the file. It also includes error handling with perror.

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

