

**LAPORAN PRAKTIKUM SISTEM OPERASI**

**MODUL 8 : SYSTEM CALL**



**DISUSUN OLEH :**

**NAMA : BIMA TRIADMAJA**  
**NIM : L200210137**  
**KELAS : C**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS KOMUNIKASI DAN INFORMATIKA**  
**UNIVERSITAS MUHAMMADIYAH SURAKARTA**  
**TAHUN 2022/2023**

## Langkah Kerja

Gunakan aplikasi ‘nano’ atau ‘vi’ atau teks editor yang lain untuk mengedit kode program berikut, Selanjutnya untuk melakukan kompilasi dapat dilakukan dengan perintah berikut:

```
$gcc 'nama_file.c'
```

Jika tidak ada kesalahan maka akan dihasilkan sebuah program bernama ‘a.out’, dan untuk menjalankan program tersebut dapat dilakukan dengan cara berikut:

```
$ ./a.out
```

Jika pada PC anda tidak tersedia compiler ‘gcc’ dapat digunakan fasilitas online compiler yang disediakan oleh link berikut:

[http://www.tutorialspoint.com/compile\\_c\\_online.php](http://www.tutorialspoint.com/compile_c_online.php)

## PRAKTIKUM 1 (fork.c)

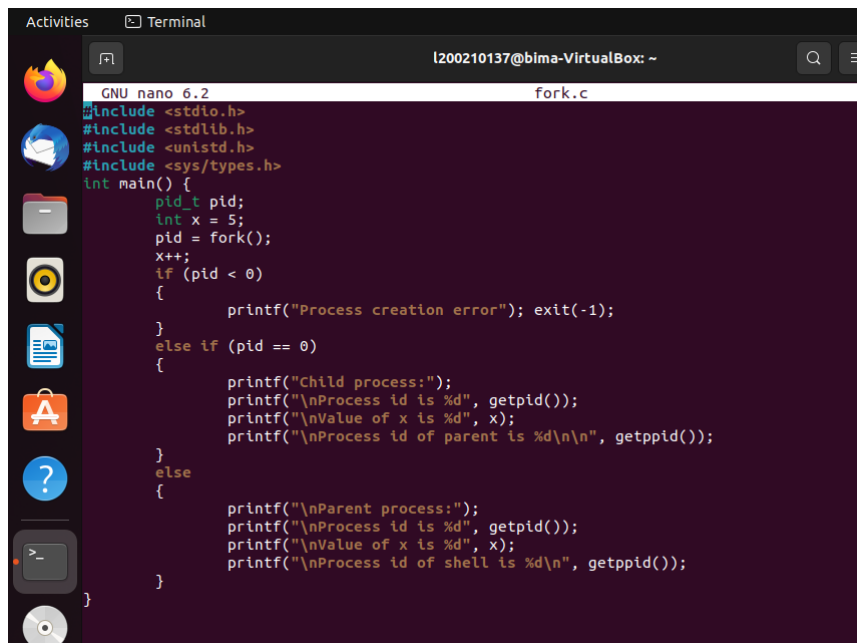
1. Membuat sebuah ‘child process’ (proses baru) dengan menggunakan system call ‘fork’.

Mambuat program dengan algoritma sebagai berikut :

(contoh program diberikan pada bagian berikutnya).

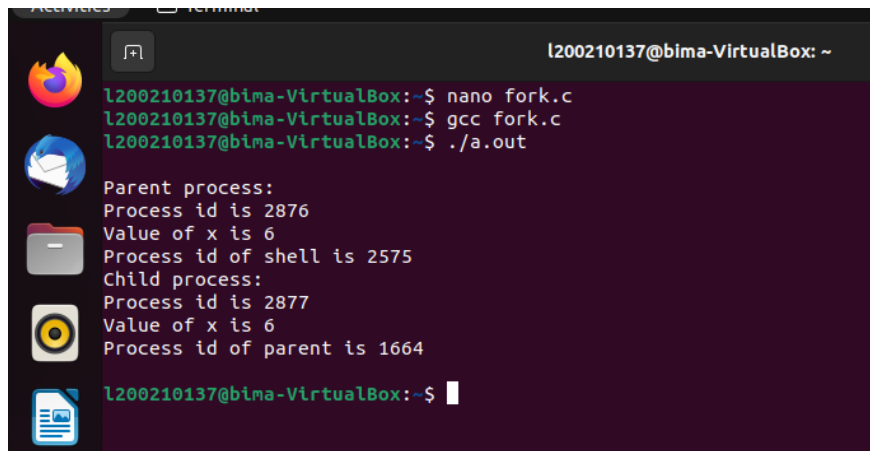
- a. Deklarasi sebuah variabel x yang akan diakses bersama antara child proses dan parent proses.
- b. Membuat sebuah child proses menggunakan system call fork.
- c. Jika return value bernilai -1, tampilkan teks ‘Pembuatan proses GAGAL’ , dilanjutkan dengan keluar program dengan perintah system call ‘exit’ .
- d. Jika return value sama dengan 0 (NOL), Tampilkan teks ‘Child Process’ , tampilkanID proses dari child proses menggunakan perintah system call ‘getpid’ , tampilkan nilai x, dan tampilkan ID proses parent dengan perintah system call ‘getppid’ .
- e. Untuk nilai return value yang lainnya, tampilkan teks ‘Parent process’ , tampilkan ID dari parent proses menggunakan perintah system call getpid, tampilkan nilai x, dan tampilkan ID dari proses shell menggunakan perintah system call getppid.
- f. Stop

➤ Kode Program :



```
GNU nano 6.2 fork.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
int main() {
    pid_t pid;
    int x = 5;
    pid = fork();
    x++;
    if (pid < 0)
    {
        printf("Process creation error"); exit(-1);
    }
    else if (pid == 0)
    {
        printf("Child process:");
        printf("\nProcess id is %d", getpid());
        printf("\nValue of x is %d", x);
        printf("\nProcess id of parent is %d\n\n", getppid());
    }
    else
    {
        printf("\nParent process:");
        printf("\nProcess id is %d", getpid());
        printf("\nValue of x is %d", x);
        printf("\nProcess id of shell is %d\n", getppid());
    }
}
```

➤ Outputnya :



```
l200210137@bima-VirtualBox: ~  
l200210137@bima-VirtualBox:~$ nano fork.c  
l200210137@bima-VirtualBox:~$ gcc fork.c  
l200210137@bima-VirtualBox:~$ ./a.out  
Parent process:  
Process id is 2876  
Value of x is 6  
Process id of shell is 2575  
Child process:  
Process id is 2877  
Value of x is 6  
Process id of parent is 1664  
l200210137@bima-VirtualBox:~$
```

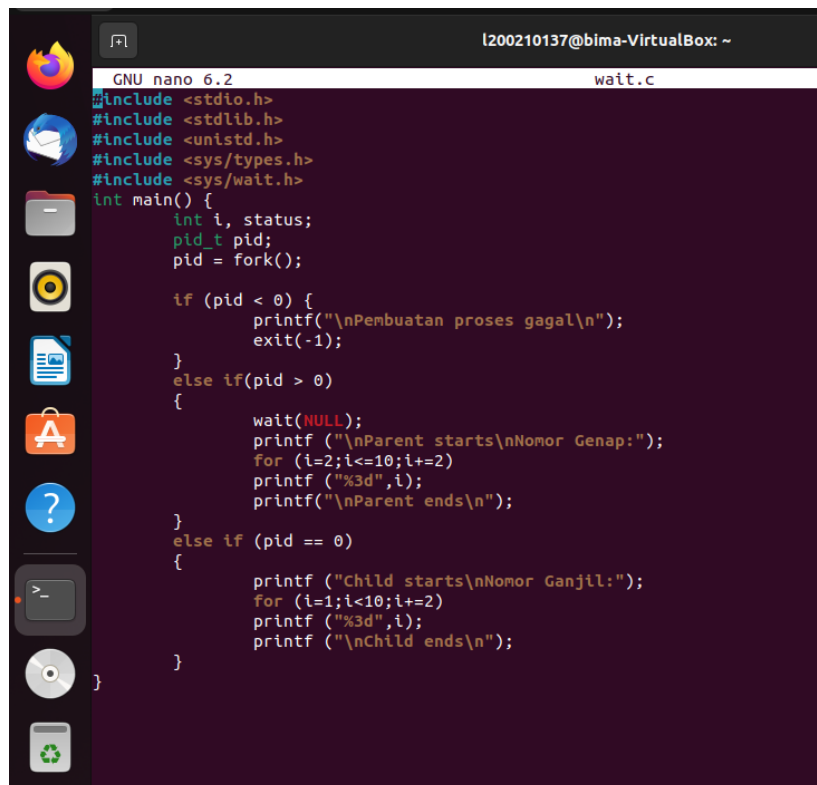
## PRAKTIKUM 2 (wait.c)

2. Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunakan perintah system call 'wait'.

Membuat program dengan algoritma sebagai berikut, contoh program diberikan pada bagian berikutnya..

- Membuat sebuah child proses menggunakan sytem call 'fork'.
- Jika return value bernilai -1, selanjutnya tampilkan teks 'pembuatan proses gagal', dan keluar program dengan menggunakan perintah system call 'exit'.
- Jika return value berupa angka positif ( $> 0$ ), 'pause' hentikan sementara 'parent' proses tunggu sampai child proses berakhir dengan menggunakan perintah system call 'wait'. Tampilkan teks 'Parent starts', selanjutnya tampilkan nomor genap mulai dari 0 s/d 10, terakhir tampilkan teks 'Parent end'.
- Jika return value bernilai 0 (NOL), tampilkan teks 'Child start', tampilkan nomor ganjil mulai dari 0 s/d 10, selanjutnya tampilkan teks 'child ends'.
- Stop

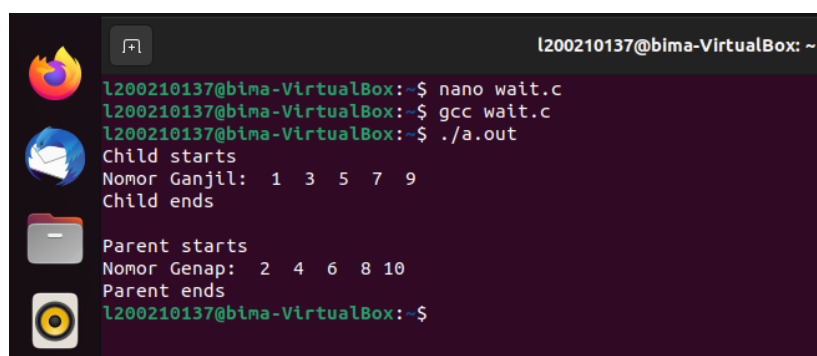
➤ Kode Program :



```
GNU nano 6.2 wait.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    int i, status;
    pid_t pid;
    pid = fork();

    if (pid < 0) {
        printf("\nPembuatan proses gagal\n");
        exit(-1);
    }
    else if (pid > 0)
    {
        wait(NULL);
        printf ("\nParent starts\nNomor Genap:");
        for (i=2;i<=10;i+=2)
            printf ("%3d",i);
        printf ("\nParent ends\n");
    }
    else if (pid == 0)
    {
        printf ("Child starts\nNomor Ganjil:");
        for (i=1;i<10;i+=2)
            printf ("%3d",i);
        printf ("\nChild ends\n");
    }
}
```

➤ Outputnya :



```
l200210137@bima-VirtualBox: ~
l200210137@bima-VirtualBox:~$ nano wait.c
l200210137@bima-VirtualBox:~$ gcc wait.c
l200210137@bima-VirtualBox:~$ ./a.out
Child starts
Nomor Ganjil:  1  3  5  7  9
Child ends

Parent starts
Nomor Genap:  2  4  6  8 10
Parent ends
l200210137@bima-VirtualBox:~$
```

### PRAKTIKUM 3 (exec.c)

3. Loading program yang dapat dieksekusi dalam sebuah 'child' proses menggunakan perintah system call 'exec'. Membuat program dengan algoritma sebagai berikut :  
(contoh program diberikan pada bagian berikutnya).
- Jika terdapat 3 argumen dalam command-line berhenti (stop).
  - Membuat child proses dengan perintah system call 'fork'
  - Jika return value adalah -1, selanjutnya tampilkan teks 'Pembuatan proses Gagal', dan keluar program dengan perintah system call exit.
  - Jika return value >0 (positif), selanjutnya hentikan parent-proses sementara hingga child-proses berakhir dengan menggunakan perintah system call wait. Tampilkan teks 'Child berakhir', dan hentikan parent-proses.
  - Jika return value sama dengan 0 (NOL), selanjutnya tampilkan teks 'Child starts', load program dari lokasi yang diberikan dalam 'path' ke dalam child-proses, menggunakan perintah system call 'exec'. Jika return value dari perintah 'exec' adalah bilangan negatif, tampilkan error yang terjadi dan stop. Hentikan child-proses.
  - Stop

Contoh cara mengkompilasi dan menjalankan program

```
$ gcc exec.c
$ ./a.out /bin/ls ls
```

➤ Kode Program :



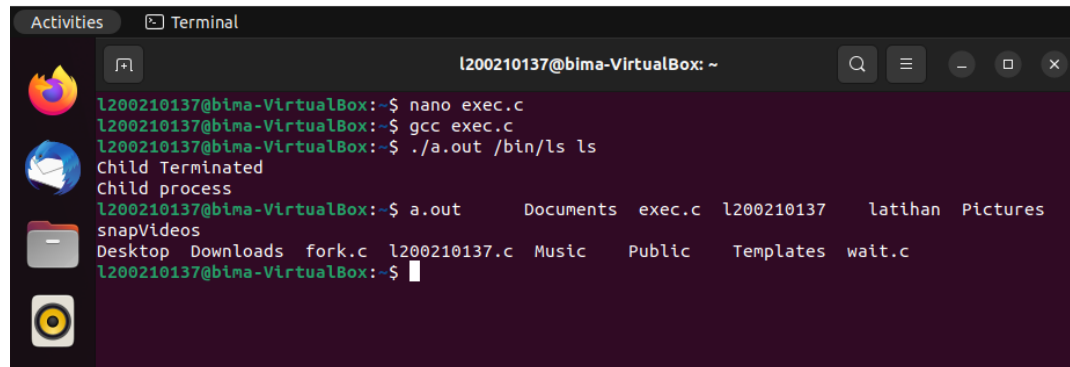
```
GNU nano 6.2 exec.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char*argv[]){

    pid_t pid;
    int i;

    if (argc != 3)
    {
        printf("\nInsufficient arguments to load program");
        printf("\nUsage: ./a.out <path> <cmd>\n");
        exit(-1);
    }

    switch(pid = fork())
    {
        case -1:
            printf("Fork failed");
            exit(-1);
        case 0:
            printf("Child process\n");
            i = execl(argv[1], argv[2], (void*)0);
            if (i < 0)
            {
                printf("%s program not loaded using exec system call\n", argv[2]);
                exit(-1);
            }
        default:
            printf("Child Terminated\n");
            exit(0);
    }
}
```

➤ Outputnya :

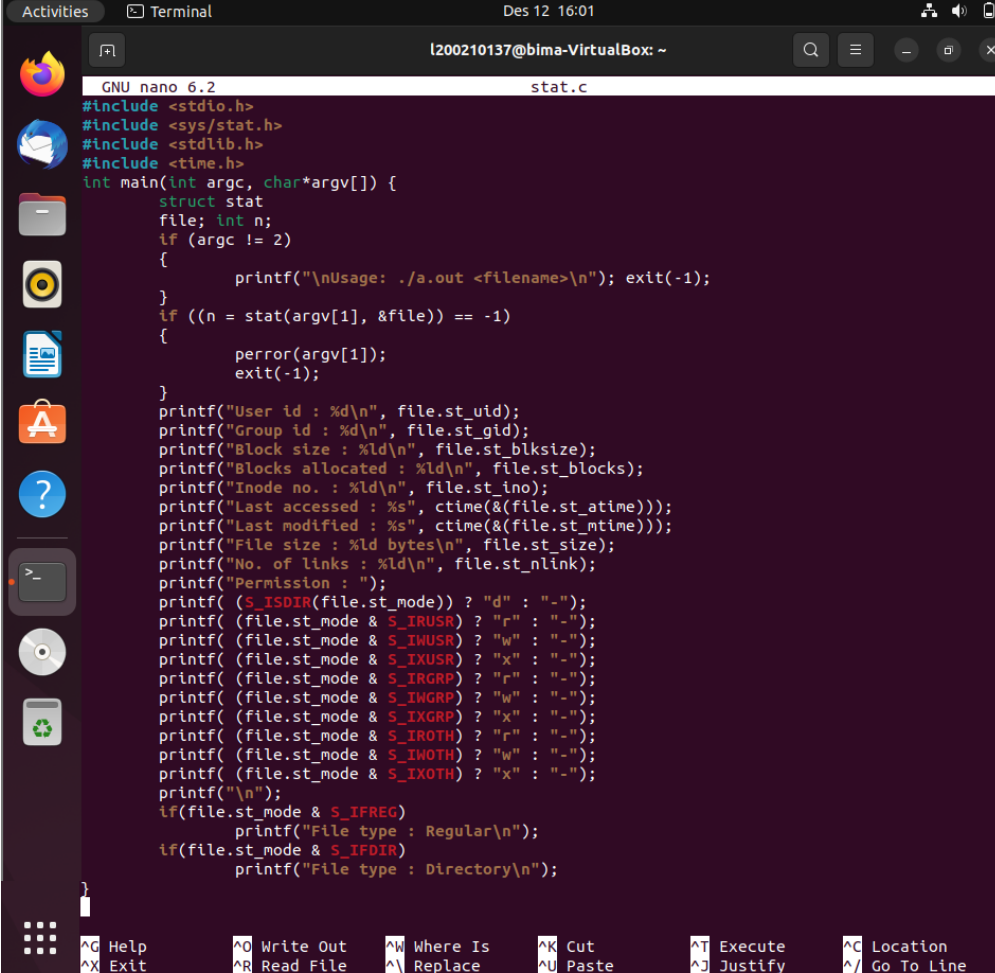


```
Activities Terminal
l200210137@bima-VirtualBox: ~
l200210137@bima-VirtualBox:~$ nano exec.c
l200210137@bima-VirtualBox:~$ gcc exec.c
l200210137@bima-VirtualBox:~$ ./a.out /bin/ls ls
Child Terminated
Child process
l200210137@bima-VirtualBox:~$ a.out Documents exec.c l200210137 latihan Pictures
snapVideos
Desktop Downloads fork.c l200210137.c Music Public Templates wait.c
l200210137@bima-VirtualBox:~$
```

## PRAKTIKUM 4 (stat.c)

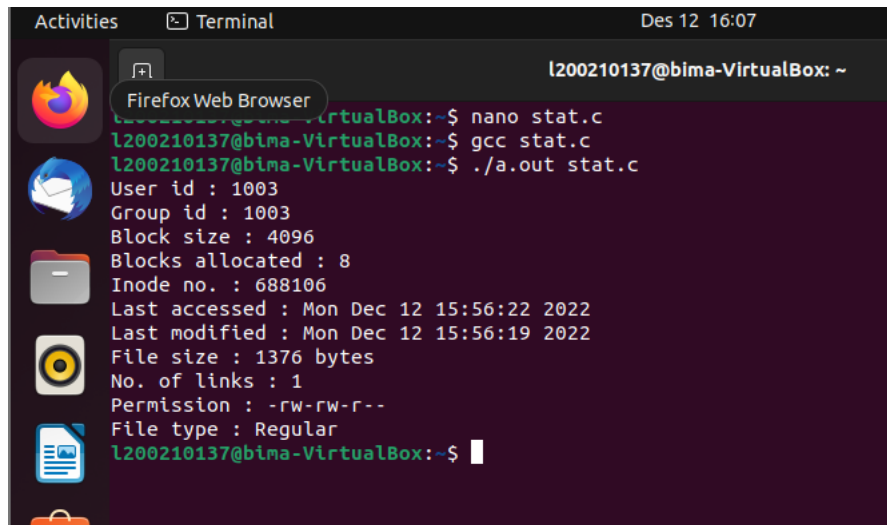
4. Menampilkan status file menggunakan perintah system call 'stat' Membuat program dengan algoritma sebagai berikut (contoh code ada di bagian berikutnya):
  - a. Gunakan 'nama file' yang diberikan melalui argument dalam perintah command-line.
  - b. Jika 'nama-file' tidak ada maka stop disini (keluar program)
  - c. Panggil system call 'stat' pada 'nama-file' tersebut yang akan mengembalikan sebuah struktur.
  - d. Tampilkan informasi mengenai st\_uid, st\_blksize, st\_block, st\_size, st\_nlink, etc.
  - e. Ubah waktu dalam st\_time, st\_mtime dengan menggunakan fungsi ctime.
  - f. Bandingkan st\_mode dengan konstanta mode seperti S\_IRUSR, S\_IWGRP, S\_IXOTH dan tampilkan informasi mengenai 'file-permissions' .
  - g. Stop

➤ Kode Program :



```
GNU nano 6.2 stat.c
#include <stdio.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char*argv[]) {
    struct stat
    file; int n;
    if (argc != 2)
    {
        printf("\nUsage: ./a.out <filename>\n"); exit(-1);
    }
    if ((n = stat(argv[1], &file)) == -1)
    {
        perror(argv[1]);
        exit(-1);
    }
    printf("User id : %d\n", file.st_uid);
    printf("Group id : %d\n", file.st_gid);
    printf("Block size : %ld\n", file.st_blksize);
    printf("Blocks allocated : %ld\n", file.st_blocks);
    printf("Inode no. : %ld\n", file.st_ino);
    printf("Last accessed : %s", ctime(&(file.st_atime)));
    printf("Last modified : %s", ctime(&(file.st_mtime)));
    printf("File size : %ld bytes\n", file.st_size);
    printf("No. of links : %ld\n", file.st_nlink);
    printf("Permission : ");
    printf( (S_ISDIR(file.st_mode)) ? "d" : "-");
    printf( (file.st_mode & S_IRUSR) ? "r" : "-");
    printf( (file.st_mode & S_IWUSR) ? "w" : "-");
    printf( (file.st_mode & S_IXUSR) ? "x" : "-");
    printf( (file.st_mode & S_IRGRP) ? "r" : "-");
    printf( (file.st_mode & S_IWGRP) ? "w" : "-");
    printf( (file.st_mode & S_IXGRP) ? "x" : "-");
    printf( (file.st_mode & S_IROTH) ? "r" : "-");
    printf( (file.st_mode & S_IWOTH) ? "w" : "-");
    printf( (file.st_mode & S_IXOTH) ? "x" : "-");
    printf("\n");
    if(file.st_mode & S_IFREG)
        printf("File type : Regular\n");
    if(file.st_mode & S_IFDIR)
        printf("File type : Directory\n");
}
```

➤ Outputnya :



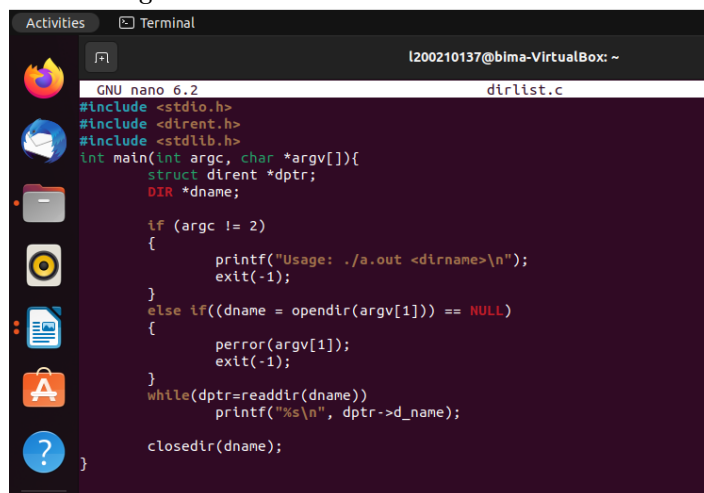
```
Activities Terminal Des 12 16:07
l200210137@bima-VirtualBox: ~
l200210137@bima-VirtualBox:~$ nano stat.c
l200210137@bima-VirtualBox:~$ gcc stat.c
l200210137@bima-VirtualBox:~$ ./a.out stat.c
User id : 1003
Group id : 1003
Block size : 4096
Blocks allocated : 8
Inode no. : 688106
Last accessed : Mon Dec 12 15:56:22 2022
Last modified : Mon Dec 12 15:56:19 2022
File size : 1376 bytes
No. of links : 1
Permission : -rw-rw-r--
File type : Regular
l200210137@bima-VirtualBox:~$
```



## PRAKTIKUM 5 (dirlist.c)

5. Menampilkan isi direktori menggunakan perintah system call 'readdir'  
Membuat program dengan algoritma sebagai berikut (contoh code ada di bagian berikutnya) :
- Gunakan 'nama-direktori' yang diberikan sebagai argumen pada command-line.
  - Jika direktori tidak ditemukan stop, keluar program
  - Buka direktori menggunakan perintah system call 'opendir' yang akan menghasilkan sebuah struktur.
  - Baca direktori menggunakan perintah system call 'readdir' yang juga akan menghasilkan struktur data.
  - Tampilkan d\_name (nama direktori)
  - Akhiri pembacaan direktori dengan perintah system call 'closedir' .
  - Stop

➤ Kode Program :

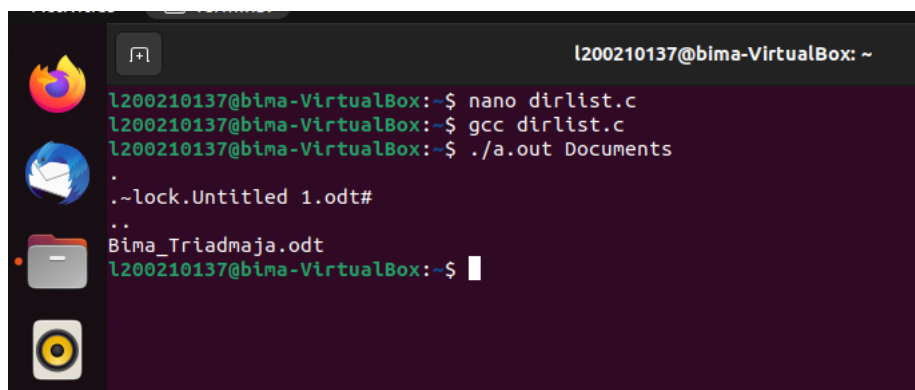


```
GNU nano 6.2 dirlist.c
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
    struct dirent *dptr;
    DIR *dname;

    if (argc != 2)
    {
        printf("Usage: ./a.out <dirname>\n");
        exit(-1);
    }
    else if((dname = opendir(argv[1])) == NULL)
    {
        perror(argv[1]);
        exit(-1);
    }
    while(dptr=readdir(dname))
        printf("%s\n", dptr->d_name);

    closedir(dname);
}
```

➤ Outputnya :



```
l200210137@bima-VirtualBox: ~
l200210137@bima-VirtualBox:~$ nano dirlist.c
l200210137@bima-VirtualBox:~$ gcc dirlist.c
l200210137@bima-VirtualBox:~$ ./a.out Documents
.
..lock.Untitled 1.odt#
..
Bima_Triadmaja.odt
l200210137@bima-VirtualBox:~$
```