**LAPORAN PRAKTIKUM**

**SISTEM OPERASI**

**MODUL 8**



**DISUSUN OLEH:**

|  |  |
| --- | --- |
| **NIM** | **L200220277** |
| **NAMA** | **MHD. FARHAN LUBIS** |
| **KELAS** | **F** |

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS KOMUNIKASI DAN INFORMATIKA**

**UNIVERSITAS MUHAMMADIYAH SURAKARTA**

**2023**

# DAFTAR ISI

[**DAFTAR ISI 2**](#_Toc152017267)

[**LANGKAH KERJA 5**](#_Toc152017268)

[**1. Membuat sebuah ‘child process’ (proses baru) dengan menggunakan system call ‘fork’. 5**](#_Toc152017269)

[**a. Deklarasi sebuah variabel x yang akan diakses bersama antara child proses dan parent proses. 6**](#_Toc152017270)

[**b. Membuat sebuah child proses menggunakan system call fork. 6**](#_Toc152017271)

[**c. Jika return value bernilai -1, tampilkan teks ‘Pembuatan proses GAGAL’, dilanjutkan dengan keluar program dengan perintah system call ‘exit’. 6**](#_Toc152017272)

[**d. Jika return value sama dengan 0 (NOL), Tampilkan teks ‘Child Process’, tampilkan ID proses dari child proses menggunakan perintah system call ‘getpid’, tampilkan nilai x, dan tampilkan ID proses parent dengan perintah system call ‘getppid’. 6**](#_Toc152017273)

[**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. 6**](#_Toc152017274)

[**f. Stop 6**](#_Toc152017275)

[**Kode program: 6**](#_Toc152017276)

[**Terminal ketika program dicompile & dijalankan: 7**](#_Toc152017277)

[**2. Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunkaan perintah system call ‘wait’. 7**](#_Toc152017278)

[**a. Membuat sebuah child proses menggunakan sytem call ‘fork’. 8**](#_Toc152017279)

[**b. Jika return value bernilai -1, selanjutnya tampilkan teks ‘pembuatan proses gagal’, dan keluar program dengan menggunakan perintah system call ‘exit’. 8**](#_Toc152017280)

[**c. 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 sd 10, terakhir tampilkan teks ‘Parent end’. 8**](#_Toc152017281)

[**d. Jika return value bernilai 0 (NOL), tampilkan teks ‘Child start’, tampilkan nomor ganjil mulai dari 0 s/d 10, selanjutnya tampilkan teks ‘child ends’ 8**](#_Toc152017282)

[**e. Stop 8**](#_Toc152017283)

[**Kode program: 8**](#_Toc152017284)

[**Terminal ketika program dicompile & dijalankan: 10**](#_Toc152017285)

[**3. Loading program yang dapat dieksekusi dalam sebuah ‘child’ proses menggunakan perintah system call ‘exec’. Membuat program dengan algoritma sebagai berikut: 10**](#_Toc152017286)

[**a. Jika terdapat 3 argumen dalam command-line berhenti (stop). 11**](#_Toc152017287)

[**b. Membuat child proses dengan perintah system call ‘fork’ 11**](#_Toc152017288)

[**c. Jika return value adalah -1, selanjuntya tampilkan teks ‘Pembuatan proses Gagal’, dan keluar program dengan perintah system call exit. 11**](#_Toc152017289)

[**d. 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. 11**](#_Toc152017290)

[**e. 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 egatif, tampilkan error yang terjadi dan stop. Hentikan child- proses. 11**](#_Toc152017291)

[**f. Stop 11**](#_Toc152017292)

[**Kode program: 11**](#_Toc152017293)

[**Terminal ketika program dicompile & dijalankan: 13**](#_Toc152017294)

[**4. Menampilkan status file menggunakan perintah system call ‘stat’ 13**](#_Toc152017295)

[**a. Gunakan ‘nama file’ yang diberikan melalui argumen dalam perintah command- line. 14**](#_Toc152017296)

[**b. Jika ‘nama-file’ tidak ada maka stop disini (keluar program) 14**](#_Toc152017297)

[**c. Panggil system call ‘stat’ pada ‘nama-file’ tersebut yang akan mengembalikan sebuah struktur 14**](#_Toc152017298)

[**d. Tampilkan informasi mengenai st\_uid, st\_blksize, st\_block, st\_size, st\_nlink, etc. 14**](#_Toc152017299)

[**e. Ubah waktu dalam st\_time, st\_mtime dengan menggunakan fungsi ctime. 14**](#_Toc152017300)

[**f. Bandingkan st\_mode dengan konstanta mode seperti S\_IRUSR, S\_IWGRP, S\_IXOTH dan tampilkan informasi mengenai ‘file-permissions’. 14**](#_Toc152017301)

[**g. Stop 14**](#_Toc152017302)

[**Kode program: 14**](#_Toc152017303)

[**Terminal ketika program dicompile & dijalankan: 17**](#_Toc152017304)

[**5. Menampilkan isi direktori menggunakan perintah system call ‘readdir’ 18**](#_Toc152017305)

[**a. Gunakan ‘nama-direktori’ yang diberikan sebagai argumen pada command-line. 19**](#_Toc152017306)

[**b. Jika direktori tidak ditemukan stop, keluar program 19**](#_Toc152017307)

[**c. Buka direktori menggunakan perintah system call ‘opendir’ yang akan menghasilkan sebuah struktur. 19**](#_Toc152017308)

[**d. Baca direktori menggunakan perintah system call ‘readdir’ yang juga akan menghasilkan struktur data. 19**](#_Toc152017309)

[**e. Tampilkan d\_name (nama direkrori) 19**](#_Toc152017310)

[**f. Akhiri pembacaa direktori dengan perintah system call ‘closedir’. 19**](#_Toc152017311)

[**g. Stop 19**](#_Toc152017312)

[**Kode program: 19**](#_Toc152017313)

[**Terminal ketika program dicompile & dijalankan: 20**](#_Toc152017314)

# 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:

<https://www.tutorialspoint.com/compile_c_online.php>

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

Membuat program dengan algoritma sebagai berikut: (contoh program diberikan pada bagian berikutnya).

### Deklarasi sebuah variabel x yang akan diakses bersama antara child proses dan parent proses.

### Membuat sebuah child proses menggunakan system call fork.

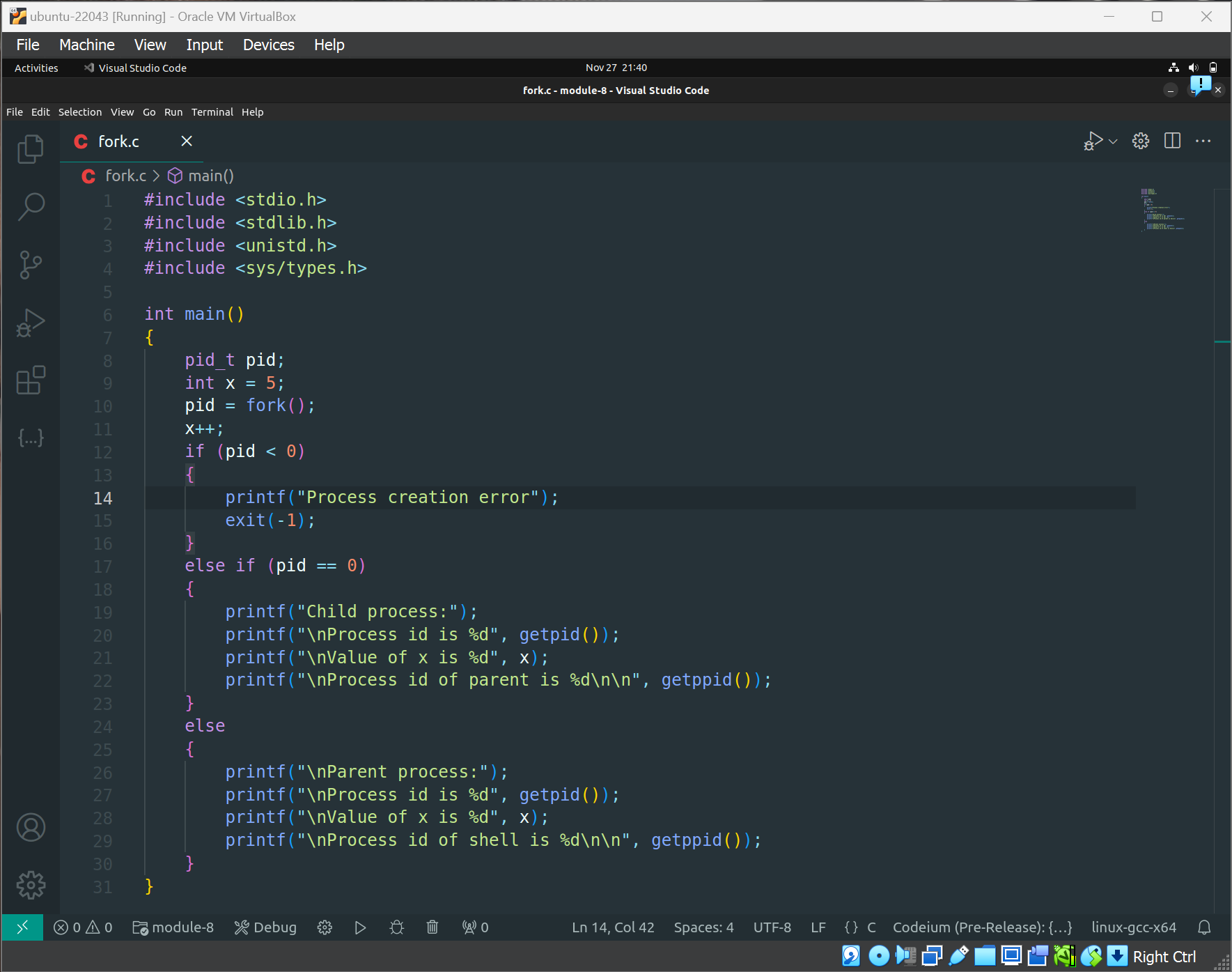
### Jika return value bernilai -1, tampilkan teks ‘Pembuatan proses GAGAL’, dilanjutkan dengan keluar program dengan perintah system call ‘exit’.

### Jika return value sama dengan 0 (NOL), Tampilkan teks ‘Child Process’, tampilkan ID proses dari child proses menggunakan perintah system call ‘getpid’, tampilkan nilai x, dan tampilkan ID proses parent dengan perintah system call ‘getppid’.

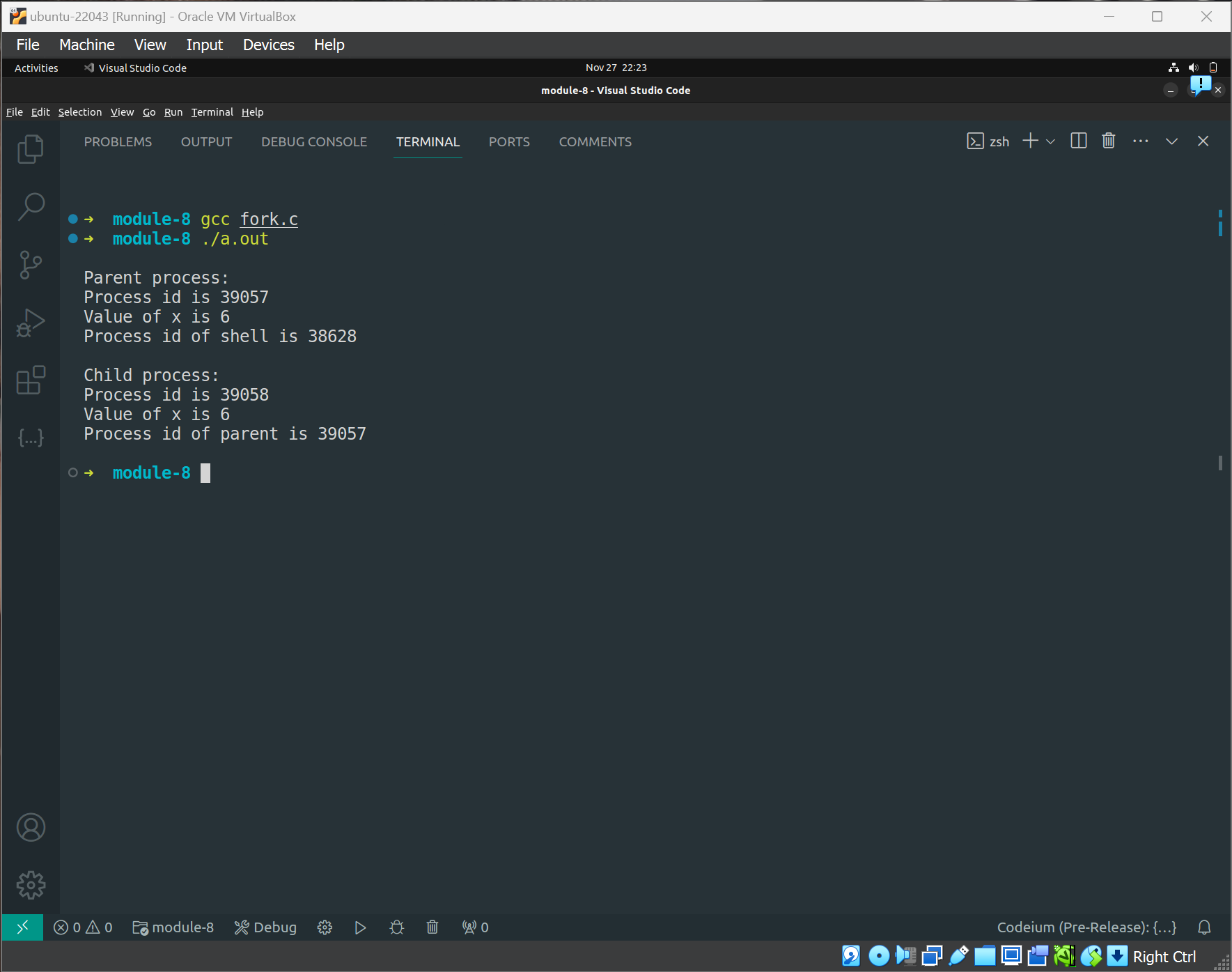
### Untuk nilai return value yang lainnya, tampilkan teks ‘Parent process’, tampilkan ID dari parent proses menggunakan perintah system call getpid, tampilkan nilai x, dan tampilkan ID dari proses shell menggunakan perintah system call getppid.

### Stop

### Kode program:



### Terminal ketika program dicompile & dijalankan:



## Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunkaan perintah system call ‘wait’.

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

### Membuat sebuah child proses menggunakan sytem call ‘fork’.

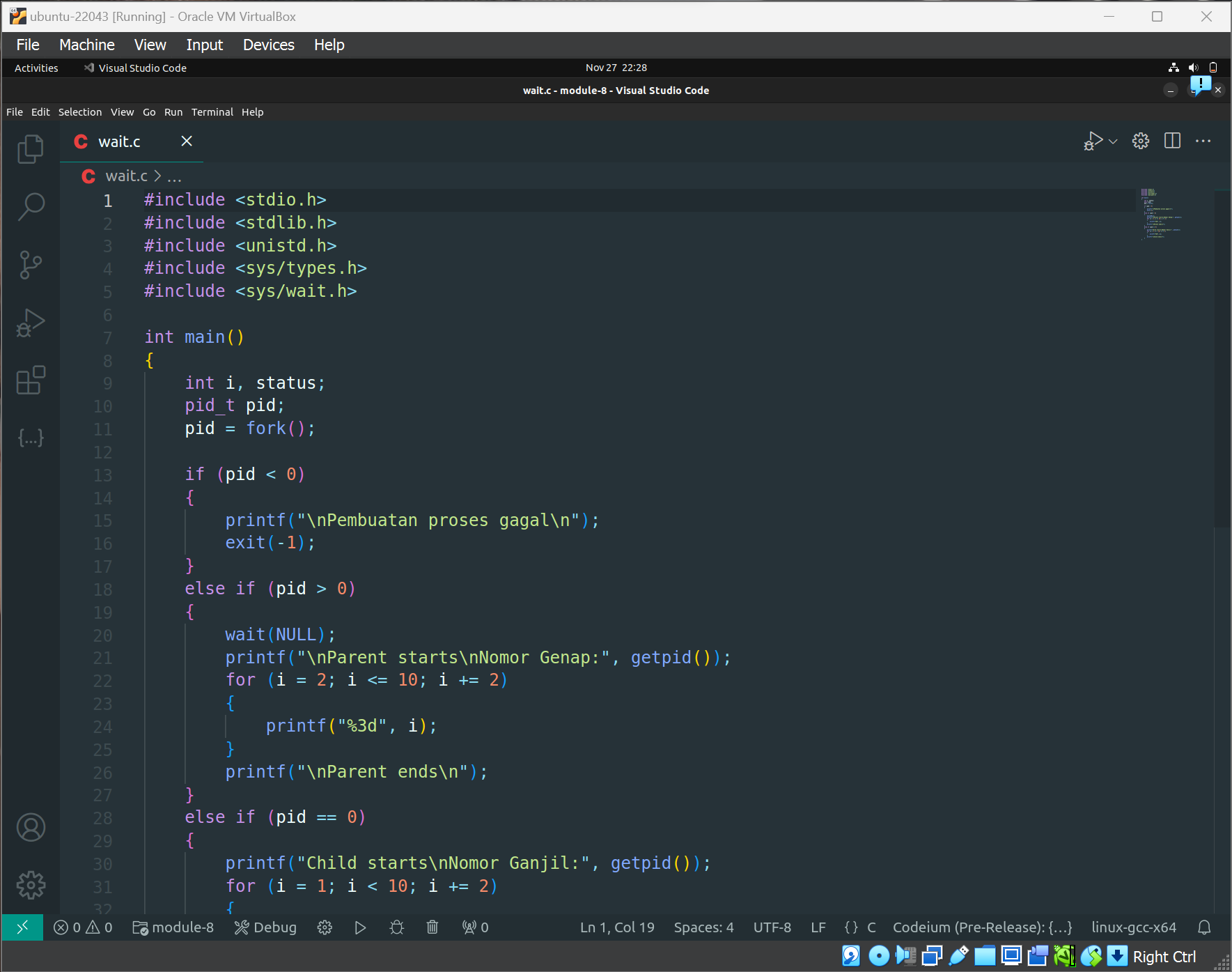
### Jika return value bernilai -1, selanjutnya tampilkan teks ‘pembuatan proses gagal’, dan keluar program dengan menggunakan perintah system call ‘exit’.

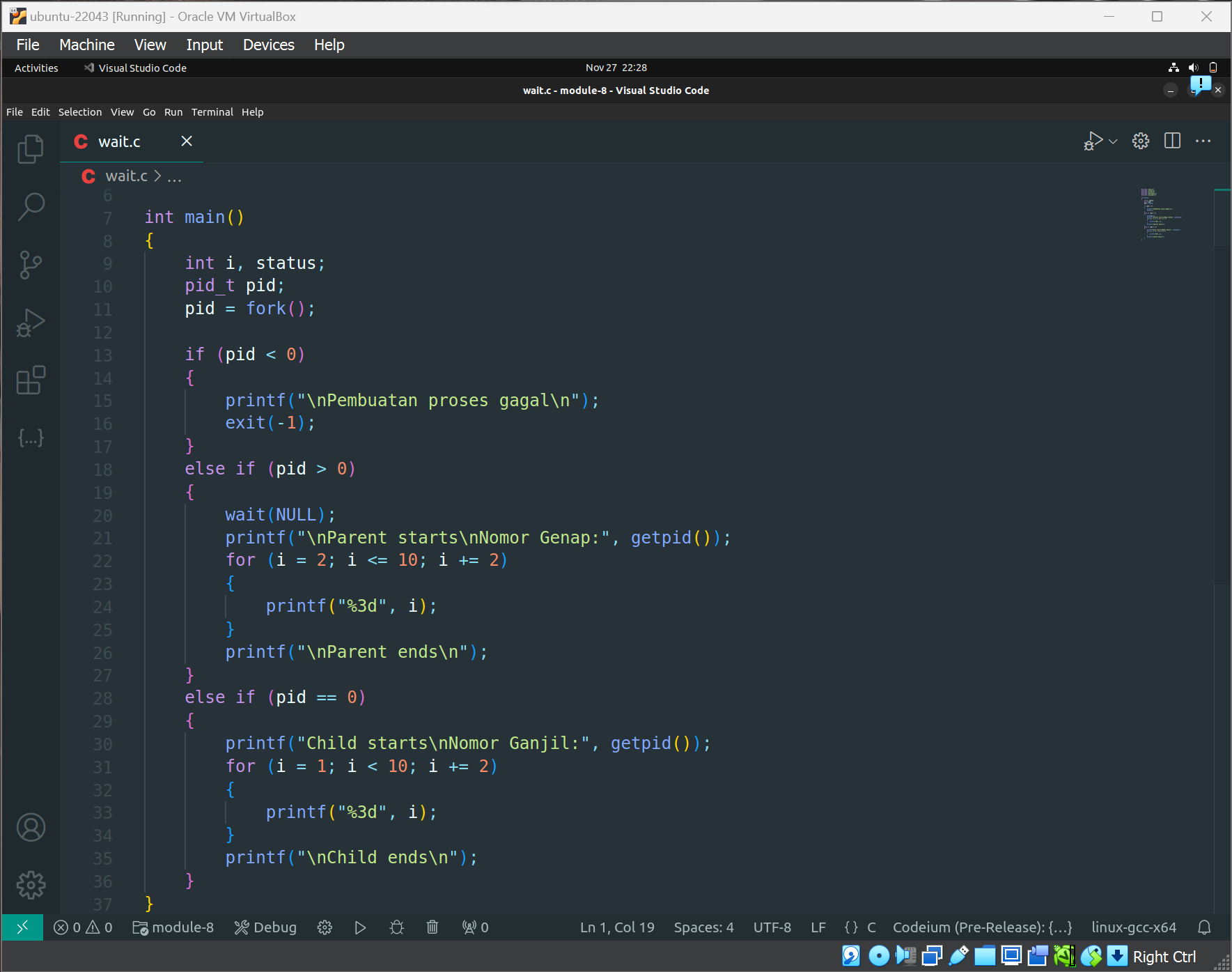
### Jika return value berupa angka positif ( > 0), ‘pause’ hentikan sementara ‘parent’ proses tunggu sampai child proses berakhir dengan menggunakan perintah system call ‘wait’. Tampilkan teks ‘Parent starts’, selanjutnya tampilkan nomor genap mulai dari 0 s/d 10, terakhir tampilkan teks ‘Parent end’.

### Jika return value bernilai 0 (NOL), tampilkan teks ‘Child start’, tampilkan nomor ganjil mulai dari 0 s/d 10, selanjutnya tampilkan teks ‘child ends’

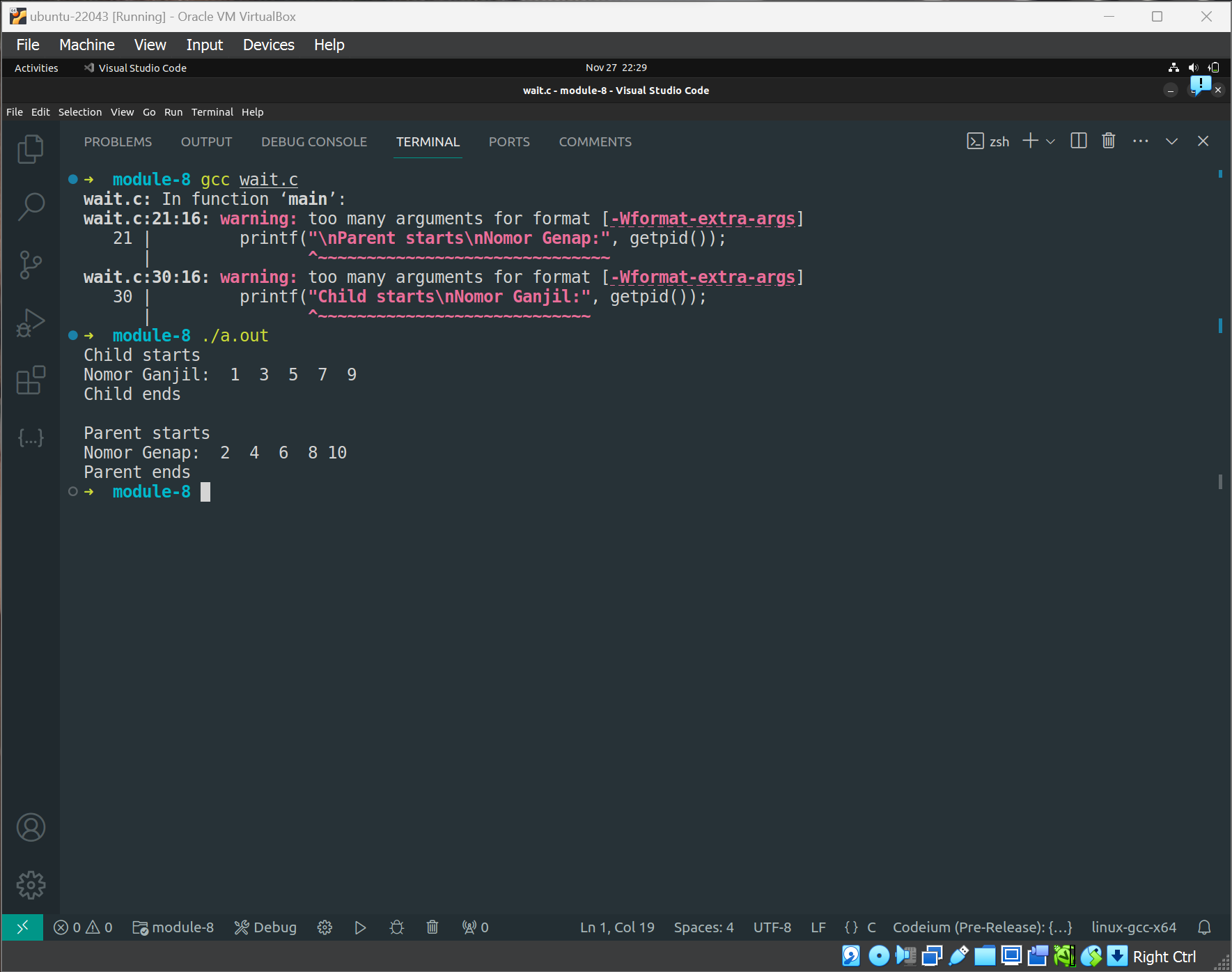
### Stop

### Kode program:





### Terminal ketika program dicompile & dijalankan:



## 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’

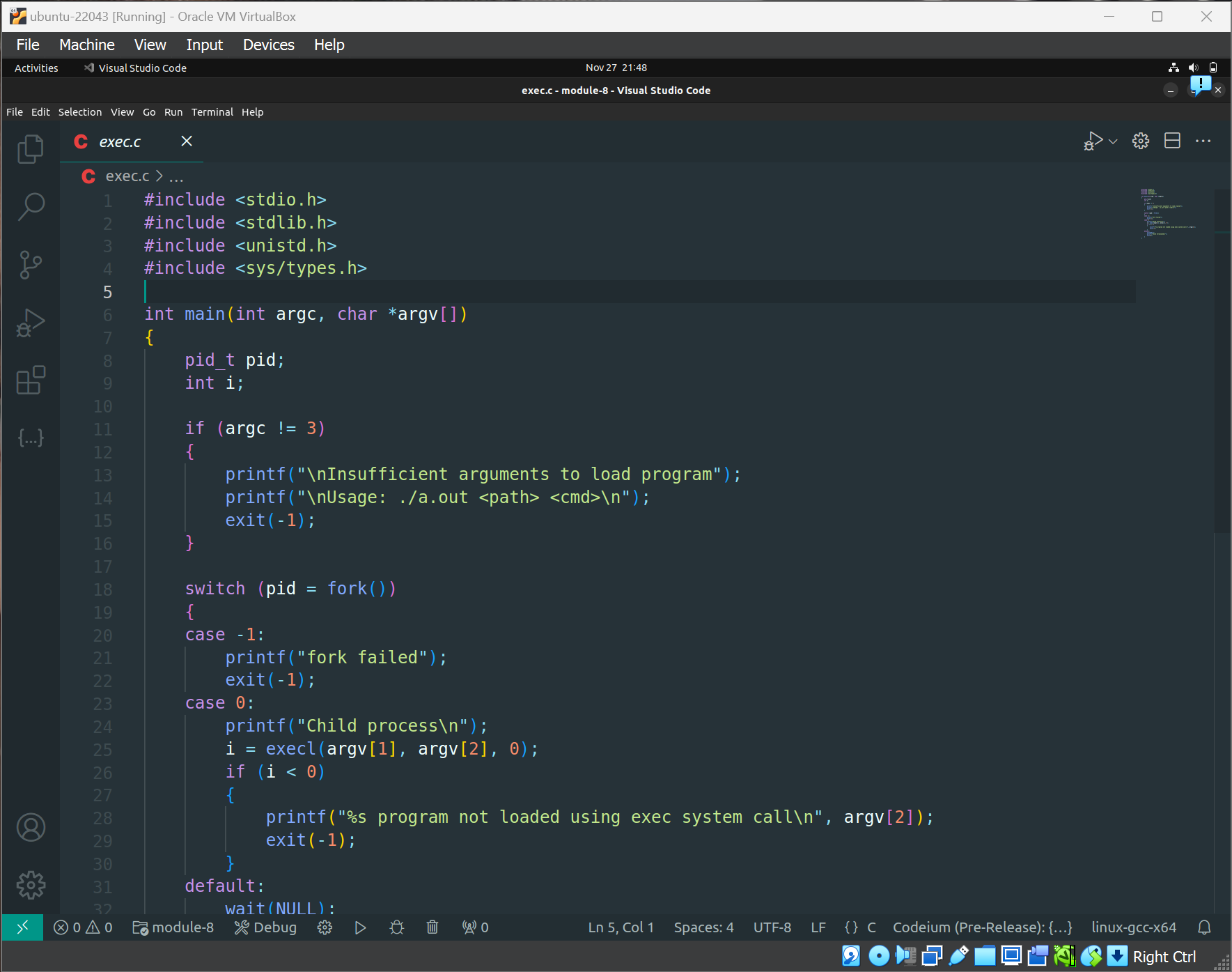
### Jika return value adalah -1, selanjuntya tampilkan teks ‘Pembuatan proses Gagal’, dan keluar program dengan perintah system call exit.

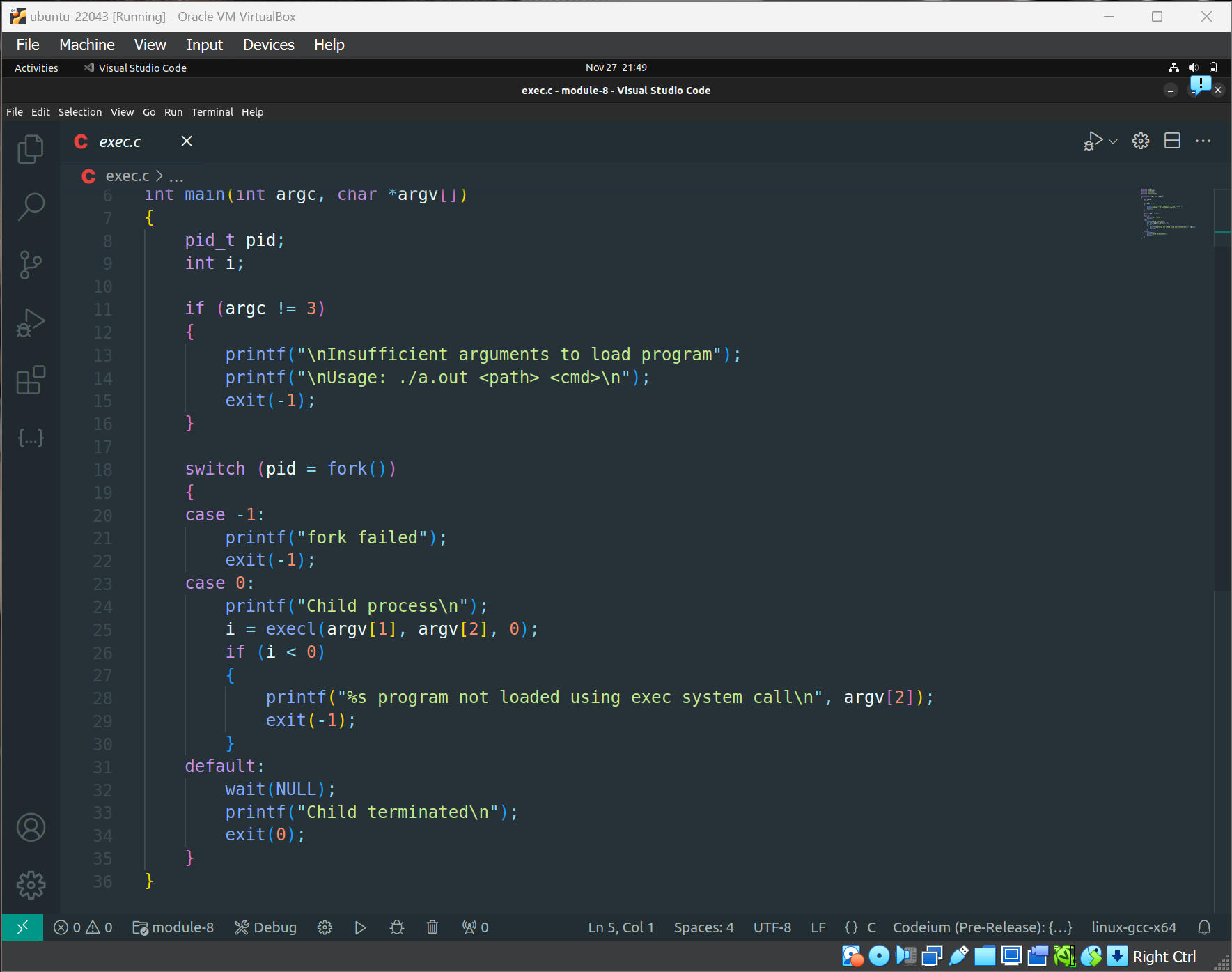
### Jika return value >0 (positif), selanjutnya hentikan parent-proses sementara hingga child-proses berakhir dengan menggunakan perintah system call wait. Tampilkan teks ‘Child berakhir’, dan hentikan parent-proses.

### 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.

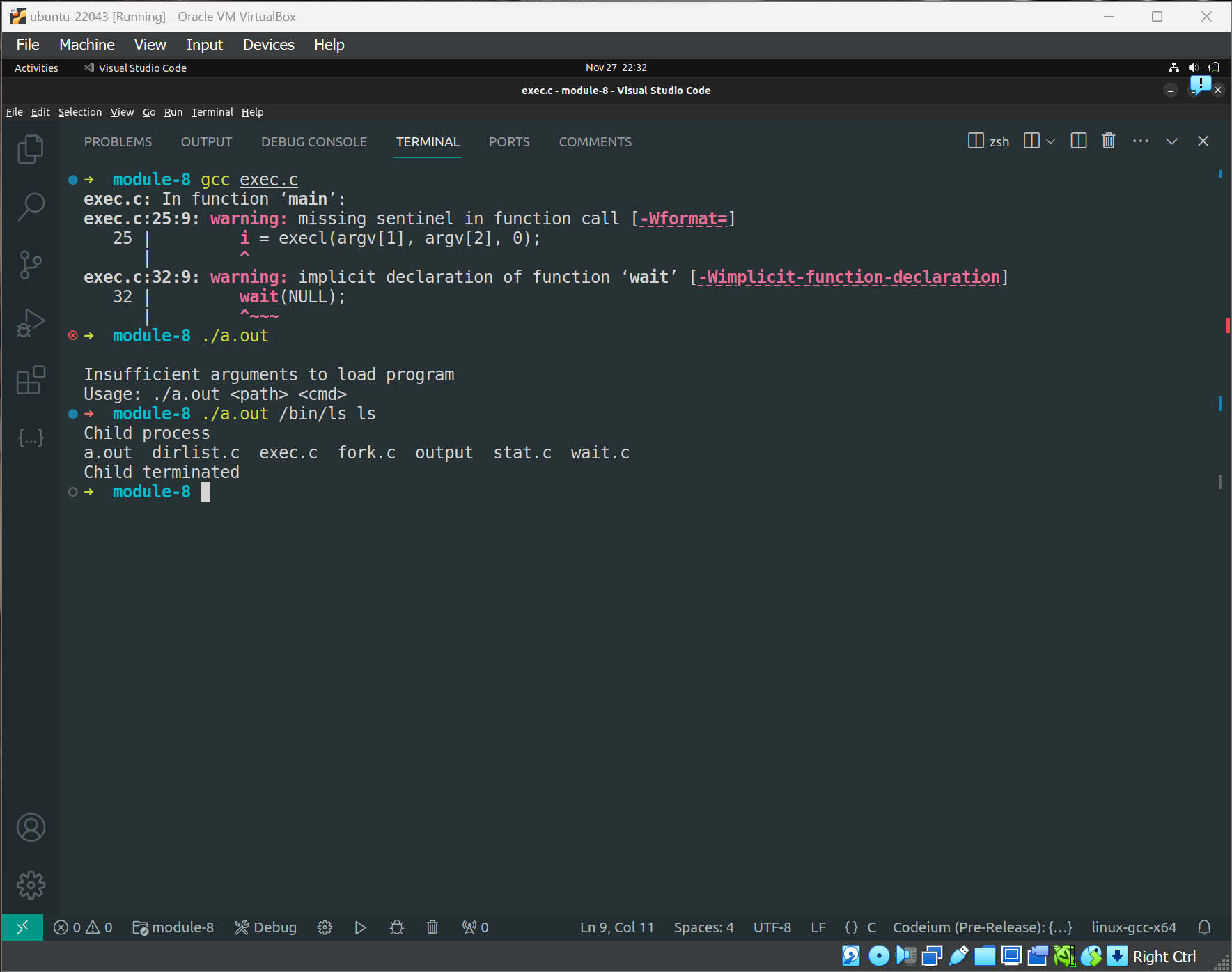
### Stop

### Kode program:





### Terminal ketika program dicompile & dijalankan:



## Menampilkan status file menggunakan perintah system call ‘stat’

Membuat program dengan algoritma sebagai berikut (contoh code ada di bagian berikutnya):

### Gunakan ‘nama file’ yang diberikan melalui argumen dalam perintah command- line.

### Jika ‘nama-file’ tidak ada maka stop disini (keluar program)

### Panggil system call ‘stat’ pada ‘nama-file’ tersebut yang akan mengembalikan sebuah struktur

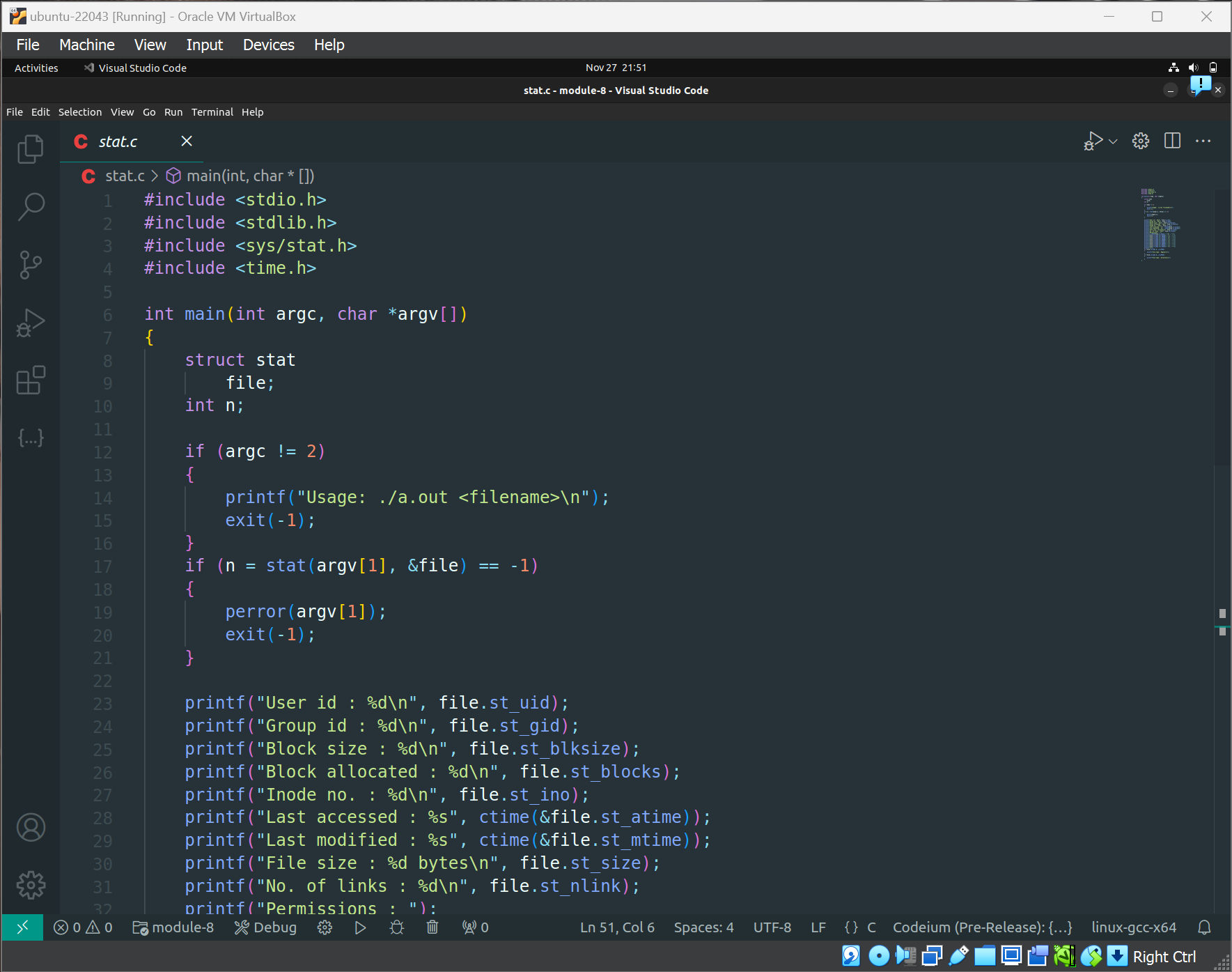
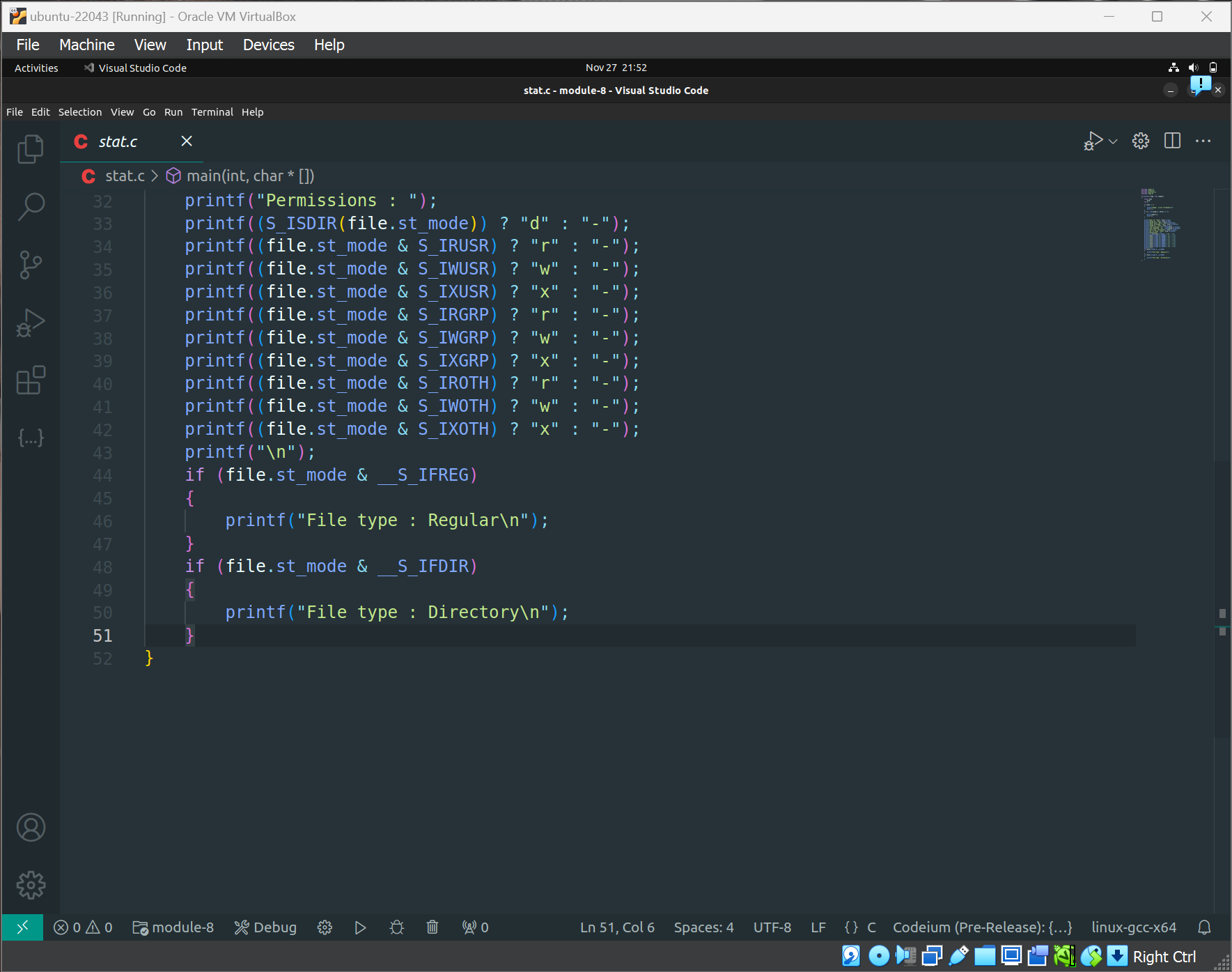
### Tampilkan informasi mengenai st\_uid, st\_blksize, st\_block, st\_size, st\_nlink, etc.

### Ubah waktu dalam st\_time, st\_mtime dengan menggunakan fungsi ctime.

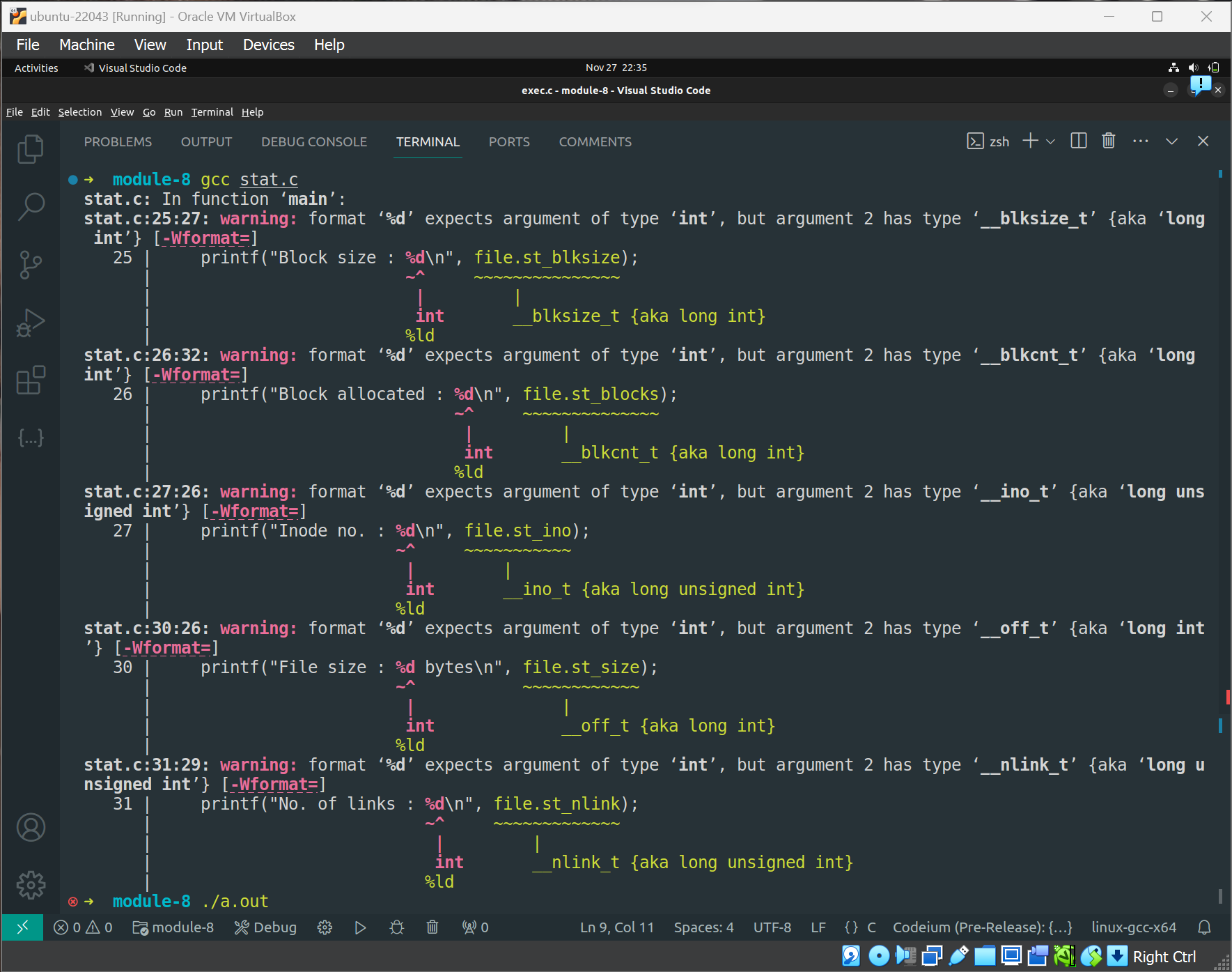
### Bandingkan st\_mode dengan konstanta mode seperti S\_IRUSR, S\_IWGRP, S\_IXOTH dan tampilkan informasi mengenai ‘file-permissions’.

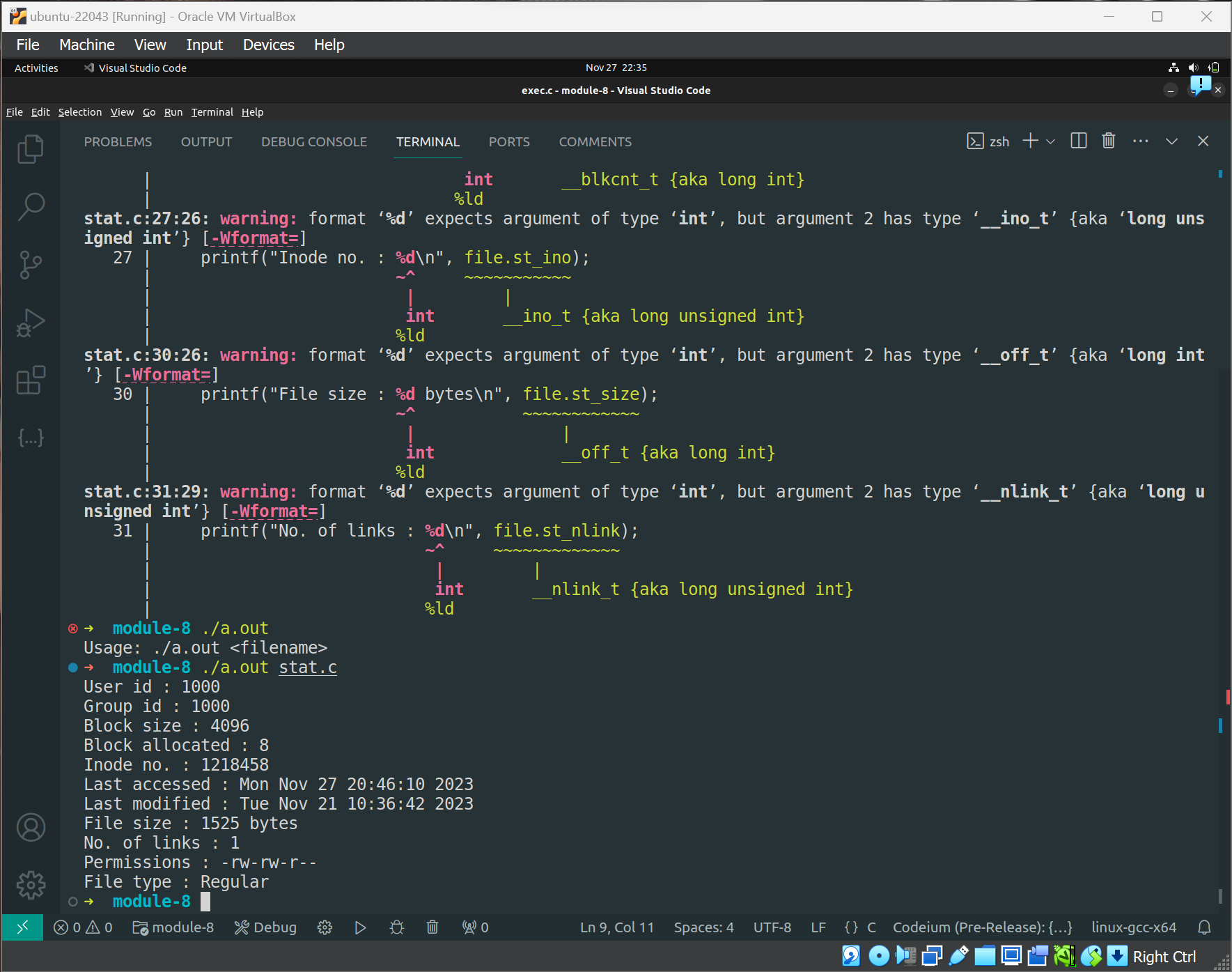
### Stop

### Kode program:

### Terminal ketika program dicompile & dijalankan:





## 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.

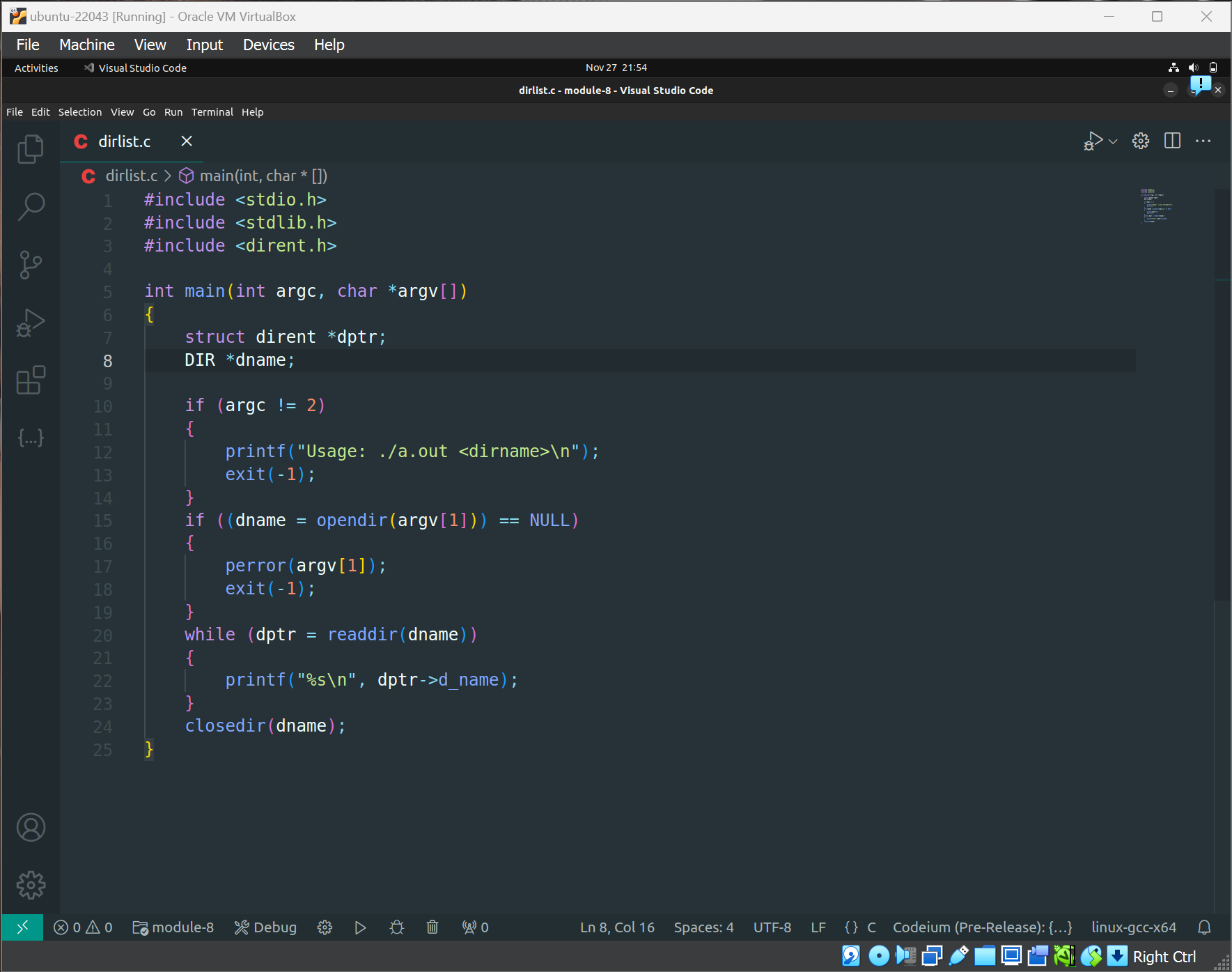
### Baca direktori menggunakan perintah system call ‘readdir’ yang juga akan menghasilkan struktur data.

### Tampilkan d\_name (nama direkrori)

### Akhiri pembacaa direktori dengan perintah system call ‘closedir’.

### Stop

### Kode program:



### Terminal ketika program dicompile & dijalankan:

