Nama : Nashirudin Baqiy

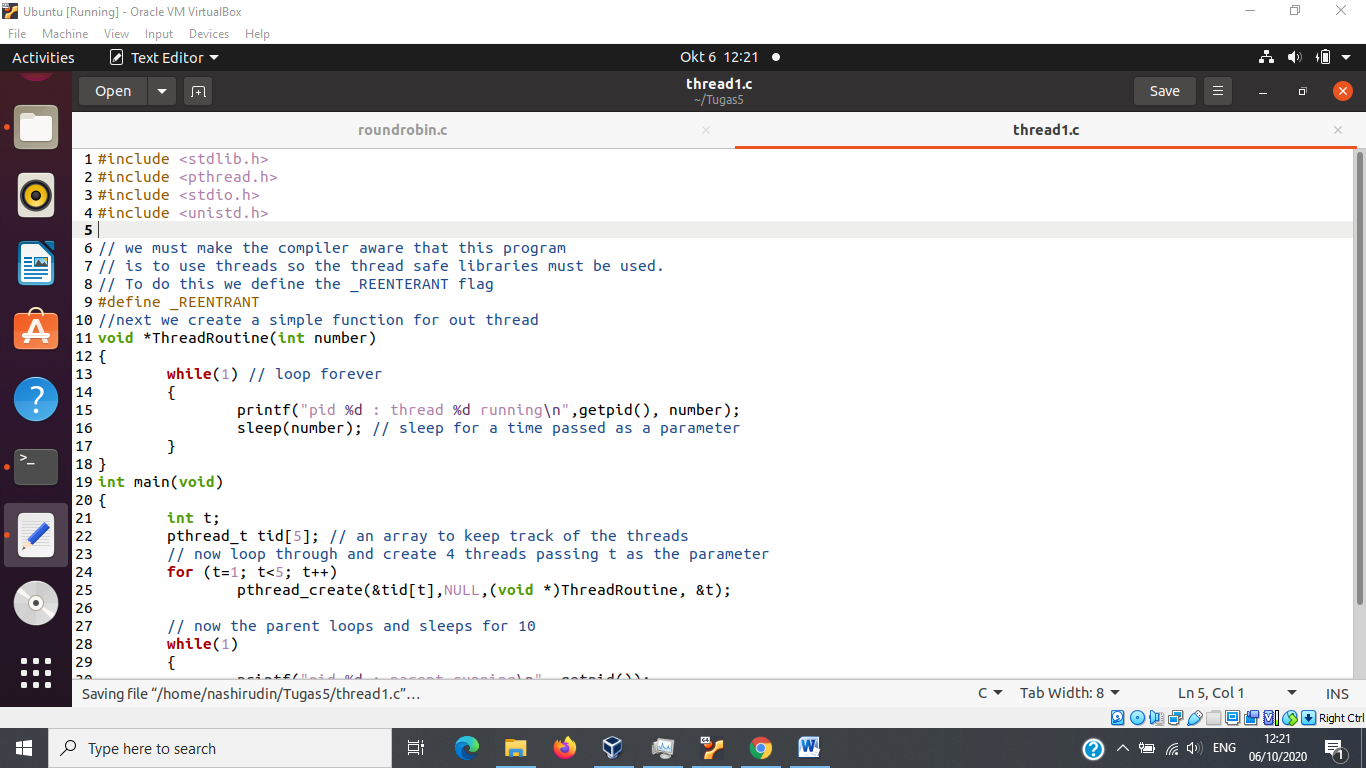
NIM : 24060119130045

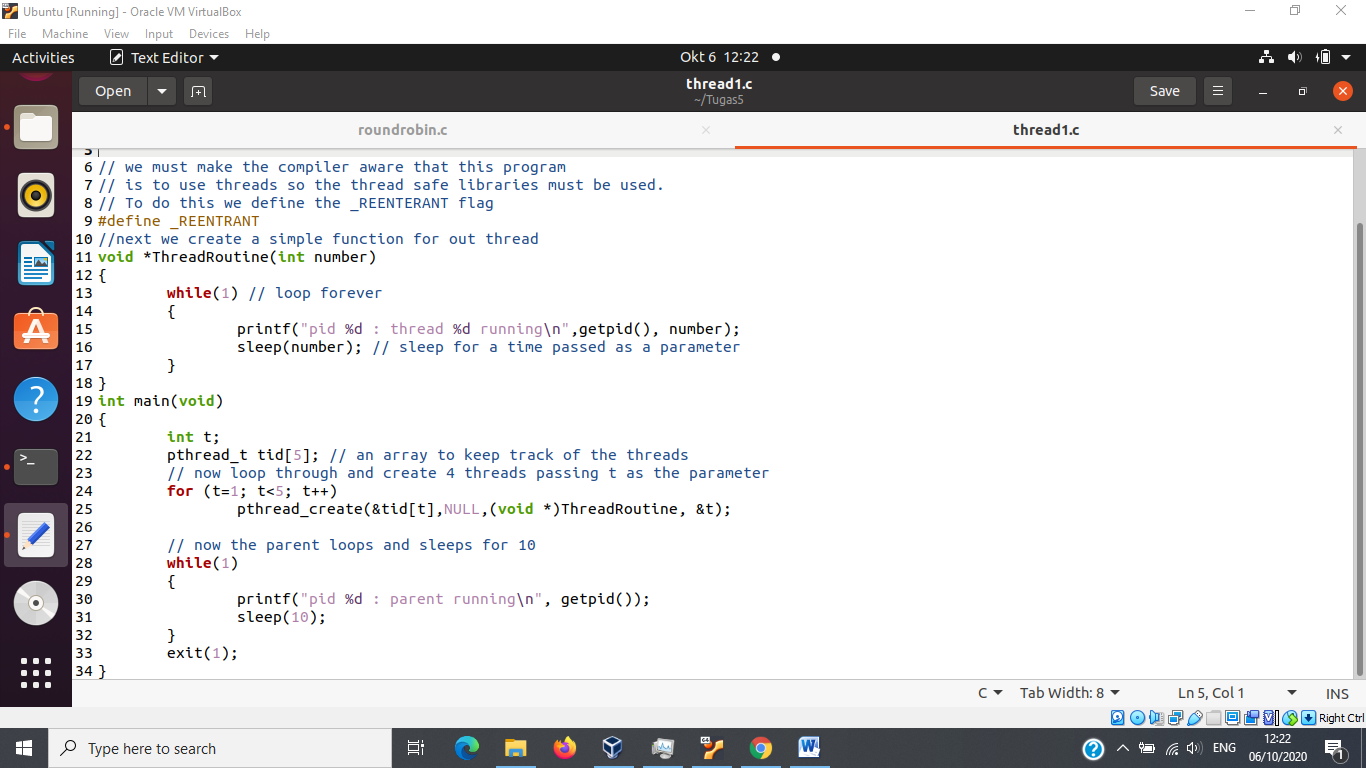
Kelas : A1

**Tugas 5**

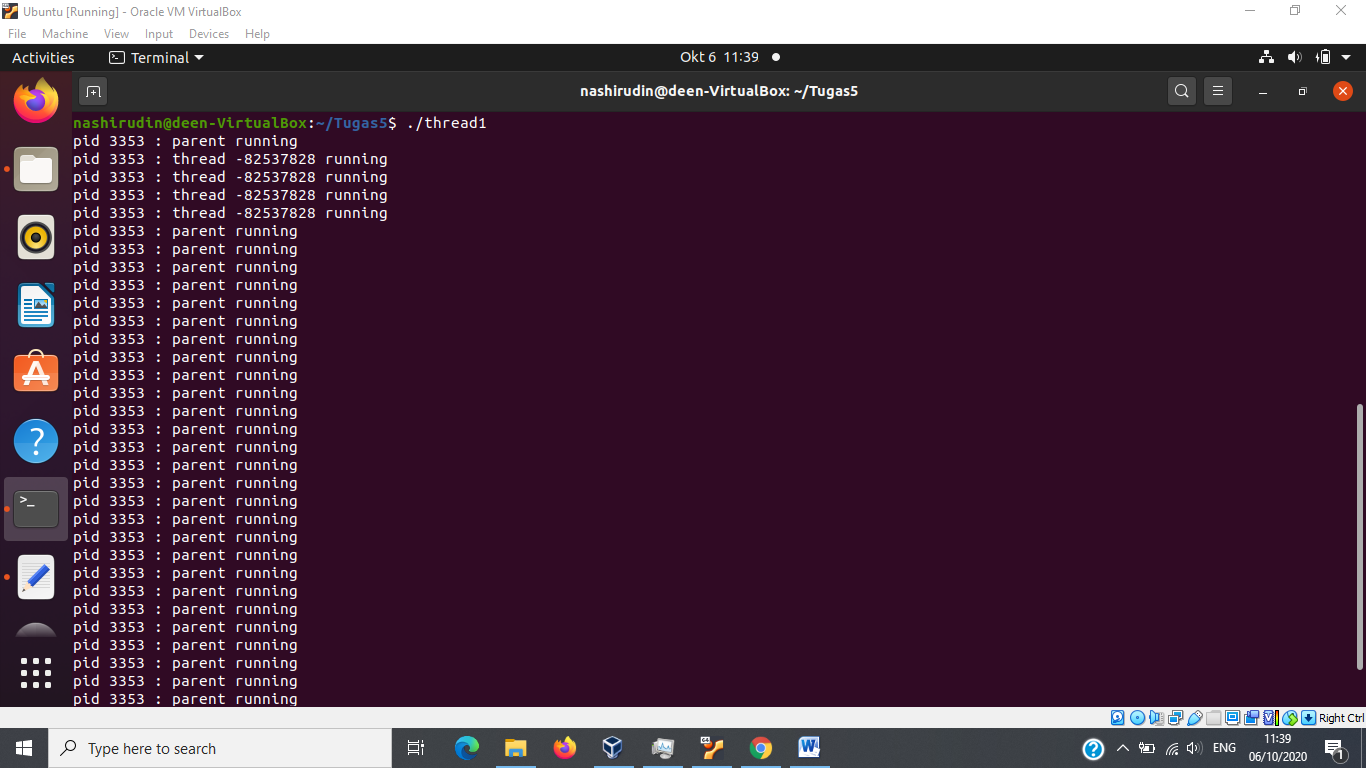
**Praktikum Sistem Operasi**

1. 5a Pembuatan Thread
   1. thread1.c





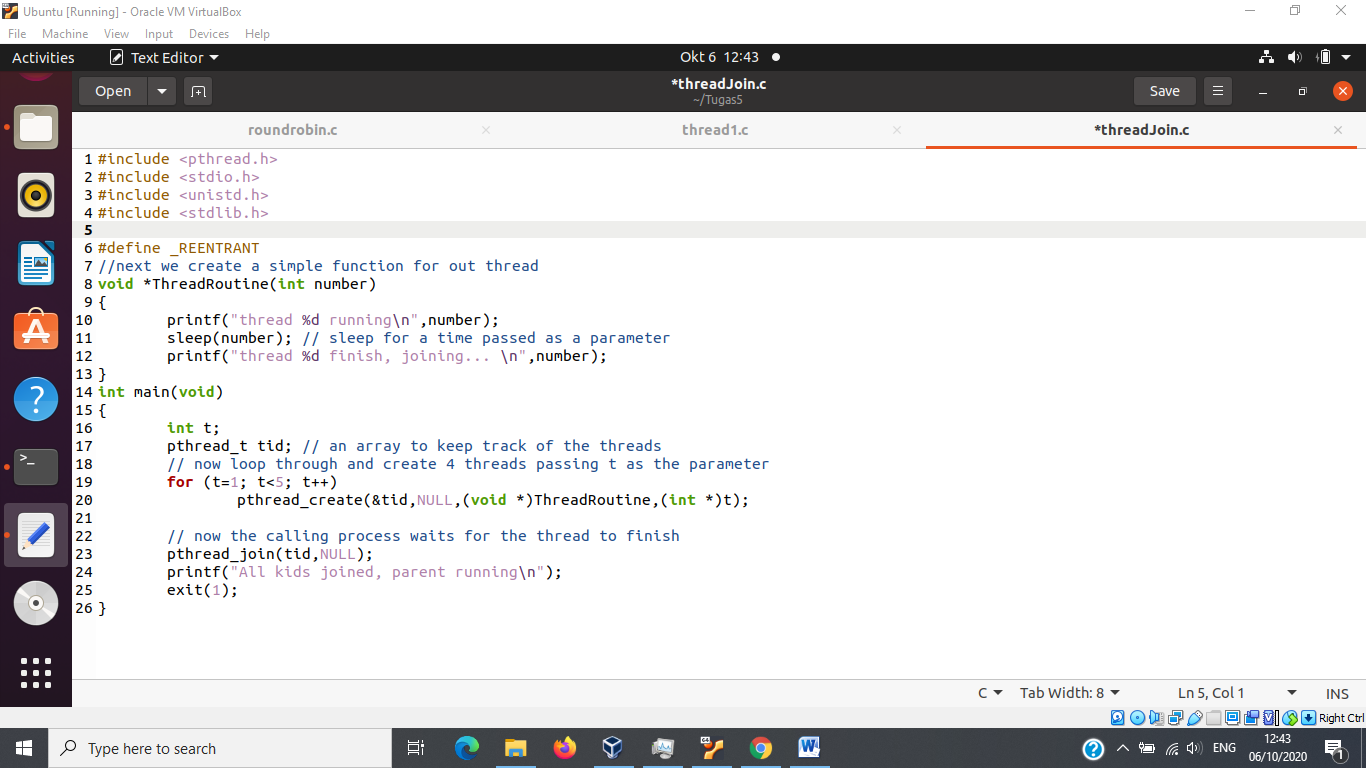
Output:



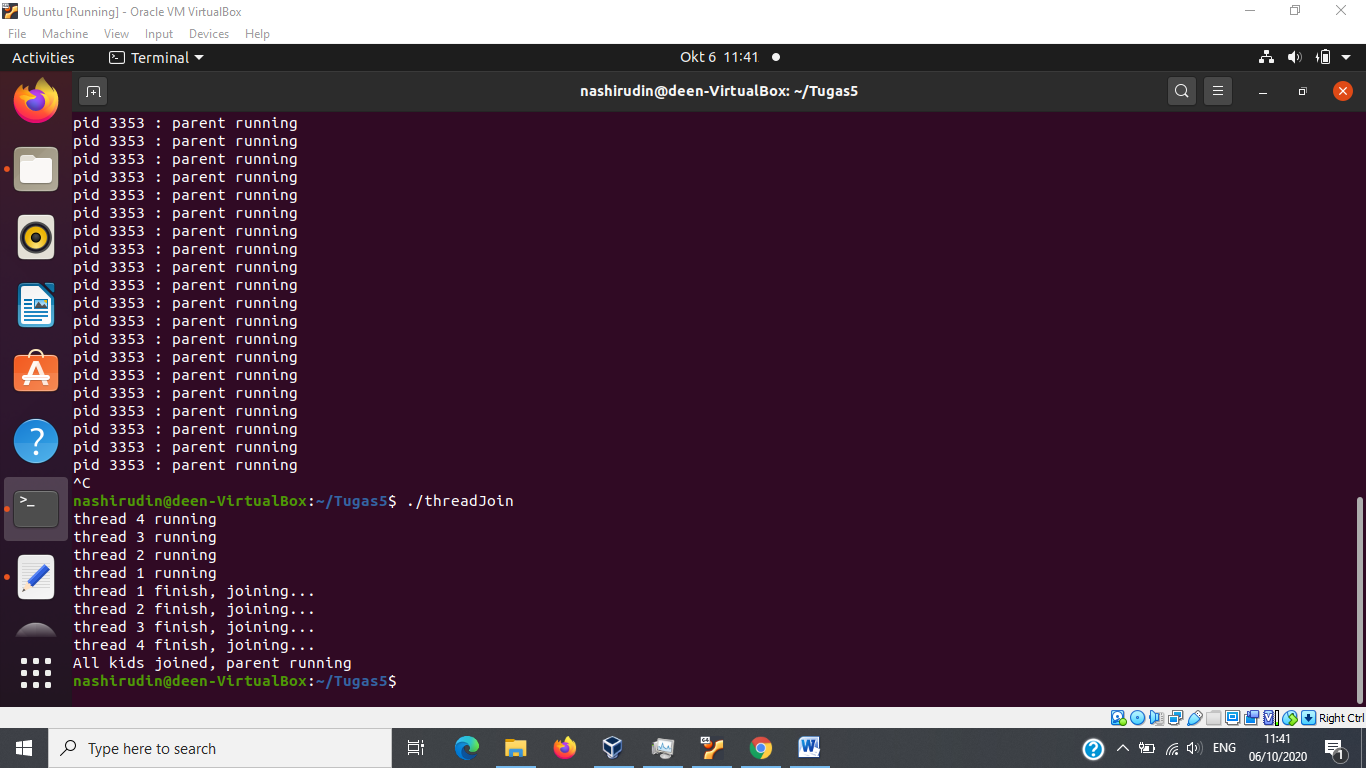
Pada program thread1.c define \_REENTRANT untuk memberitahu compiler bahwa codenya ada thread. Thread tid didefinisikan pthread\_t. Dibuat variabel array yang menyimpan track of the threads. Parent dijalankan terlebih dahulu karena traversalnya menunggu return parentnya lalu sleep(10). Kemudian dijalankan traversal pthread\_create membuat thread dan menjalankan ThreadRoutine sampai 4 threads. ThreadRoutine adalah isi thread.

Terakhir, hanya ada looping terus menerus tiap 10 detik terhadap parent. Pengaruh nilai sleep pada program thread1.c yaitu membiarkan parent running dahulu agar dapat menjalankan threads dan konstan sleep 10 detik terhadap looping parent.

* 1. threadJoin.c



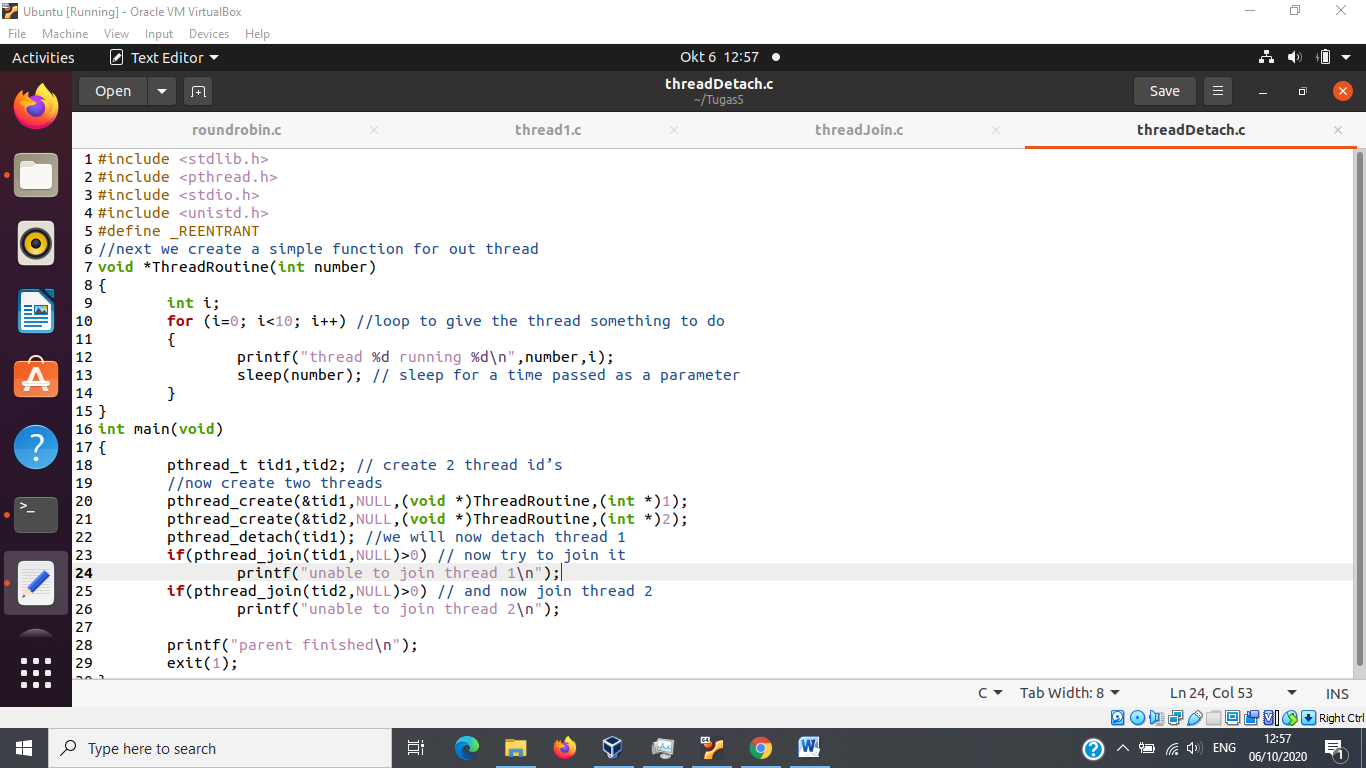
Output:



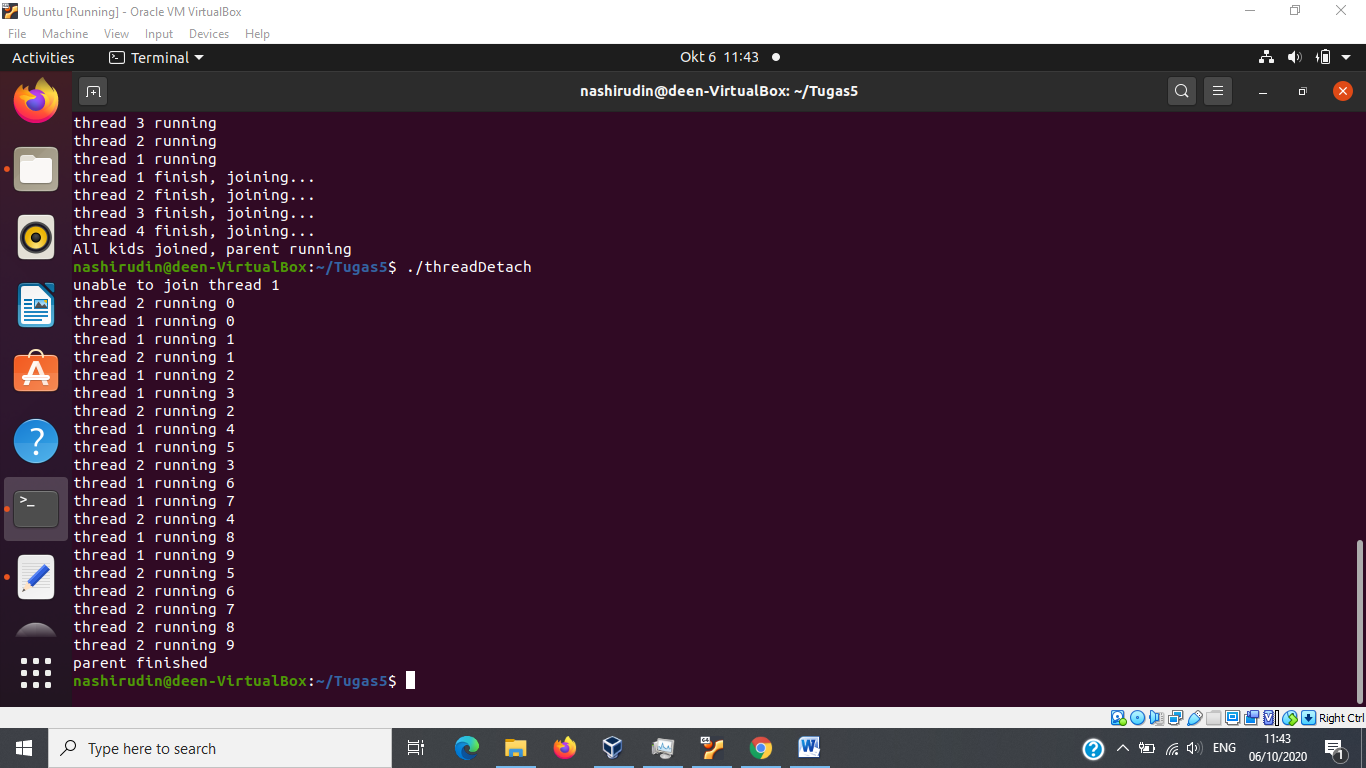
Pada program threadJoin.c define \_REENTRANT untuk memberitahu compiler bahwa codenya ada thread. Thread tid didefinisikan pthread\_t. Melakukan traversal pthread\_create membuat thread (child) dan menjalankan ThreadRoutine sebanyak 4 kali. ThreadRoutine adalah isi thread. ThreadRoutine mengandung penjalanan thread, thread manapun menunggu dengan sleep(number) agar semua thread dirun sebelum thread finishing. Keempat thread membagi resource yang sama dalam satu proses.

Terakhir, dijalankan pthread\_join. Tujuan penggunaan pthread\_join pada program threadJoin.c ini adalah memanggil tid (parent) bergabung dengan thread-thread lain penjadwalan terminasinya. Maka tid (parent) akan berjalan selama thread lain berjalan dan terminasi bila thread lain terminated. Hal ini agar parent menunggu childnya selesai.

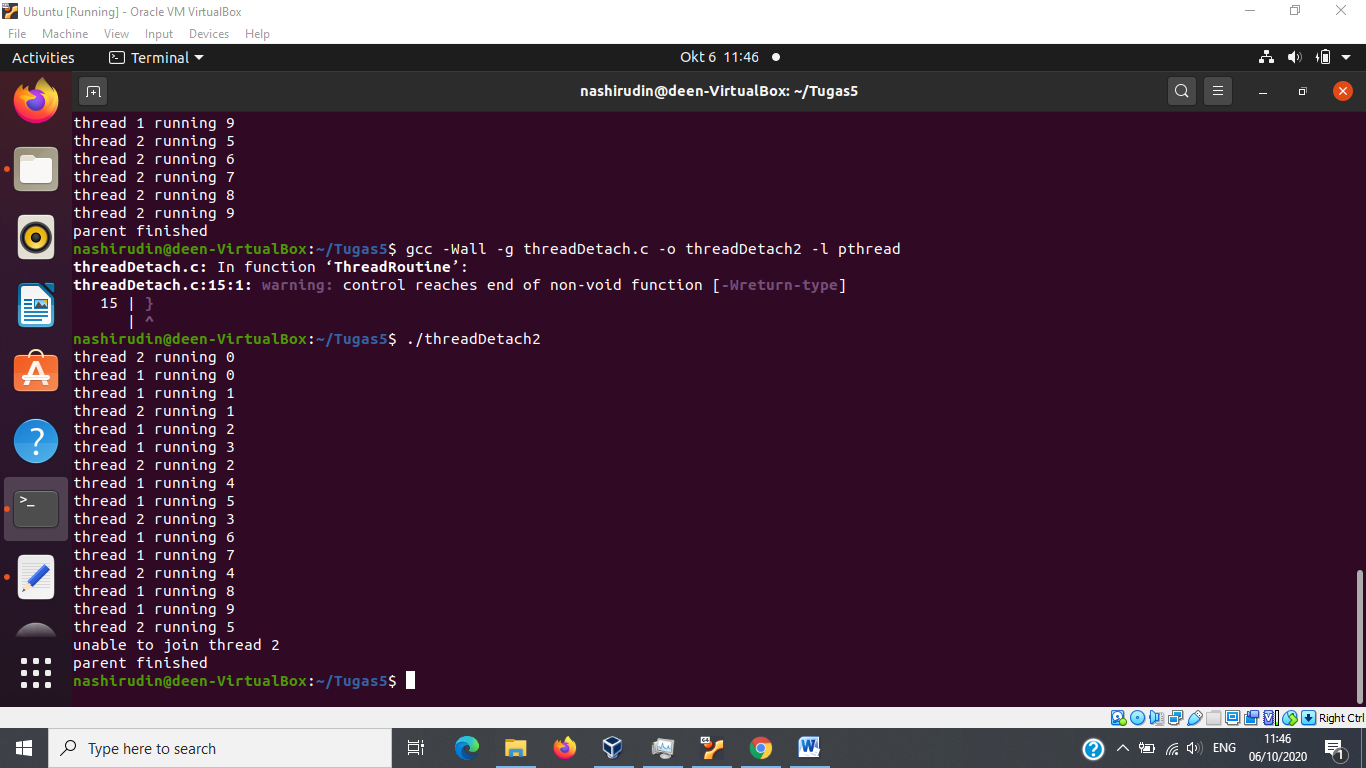
* 1. threadDetach.c



Output pthread\_detach(tid1):



Output pthread\_detach(tid2):



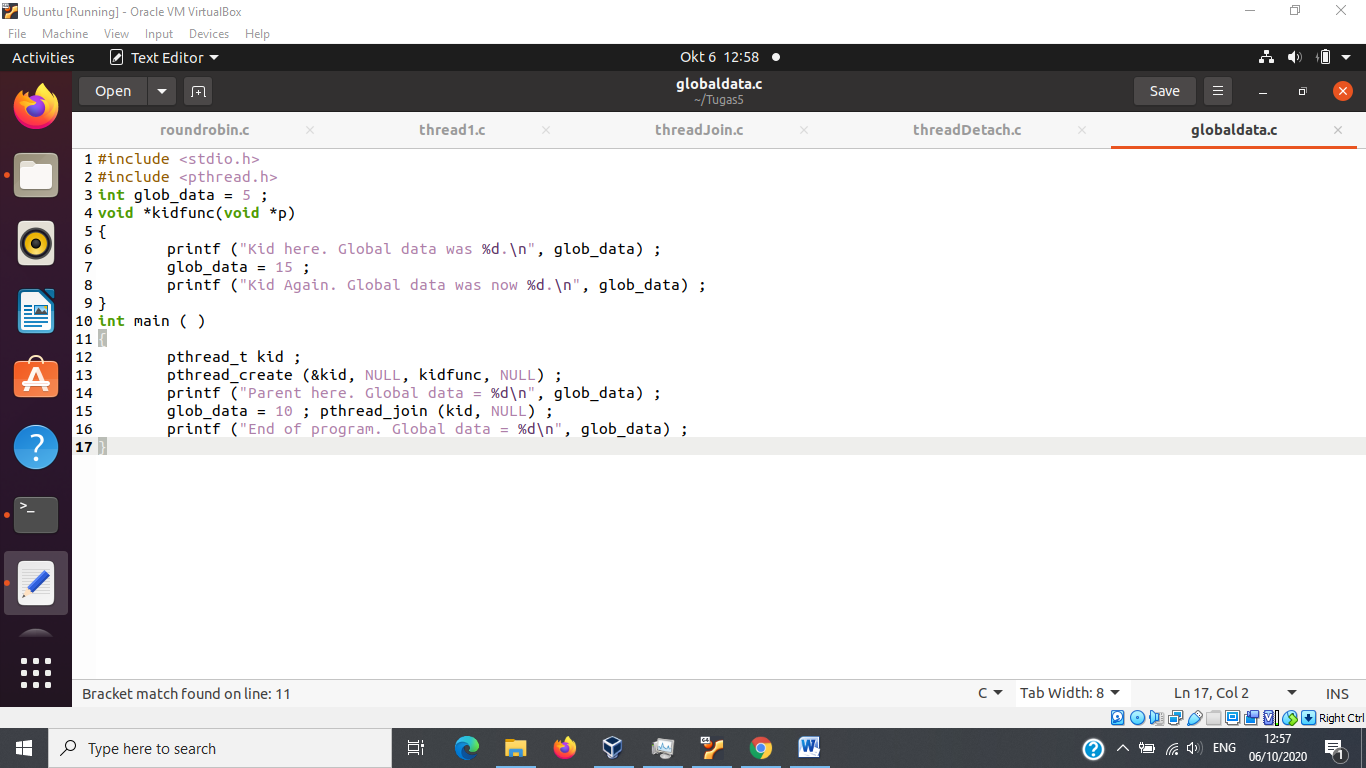
Pada program threadDetach.c define \_REENTRANT untuk memberitahu compiler bahwa codenya ada thread. Thread tid1 tid2 didefinisikan pthread\_t. dibuat 2 threads dengan pthread\_create lalu menjalankan ThreadRoutine yang mengandung traversal running thread sebanyak 10 kali yang index awalnya 0. ThreadRoutine adalah isi thread. Setelah pthread\_create, dijalankan pthread\_detach untuk melepas thread. Di sini kita membandingkan pthread\_detach tid1 dan tid2.

Ketika detach tid1, tid1 dilepas koneksi joinablenya kemudian mengembalikan nilai error saat join di awal. Ketika detach tid2, tid2 dilepas koneksi joinablenya kemudian mengembalikan nilai error join saat tid1 selesai. Error join saat detach tid1 di awal karena tid1 berhasil dilepas duluan sebelum join. Join saat detach tid2 tidak error karena join telah terjadi dan tidak bisa diganggu oleh detach sehingga baru mengembalikan error join ketika thread lain (tid1) terminated.

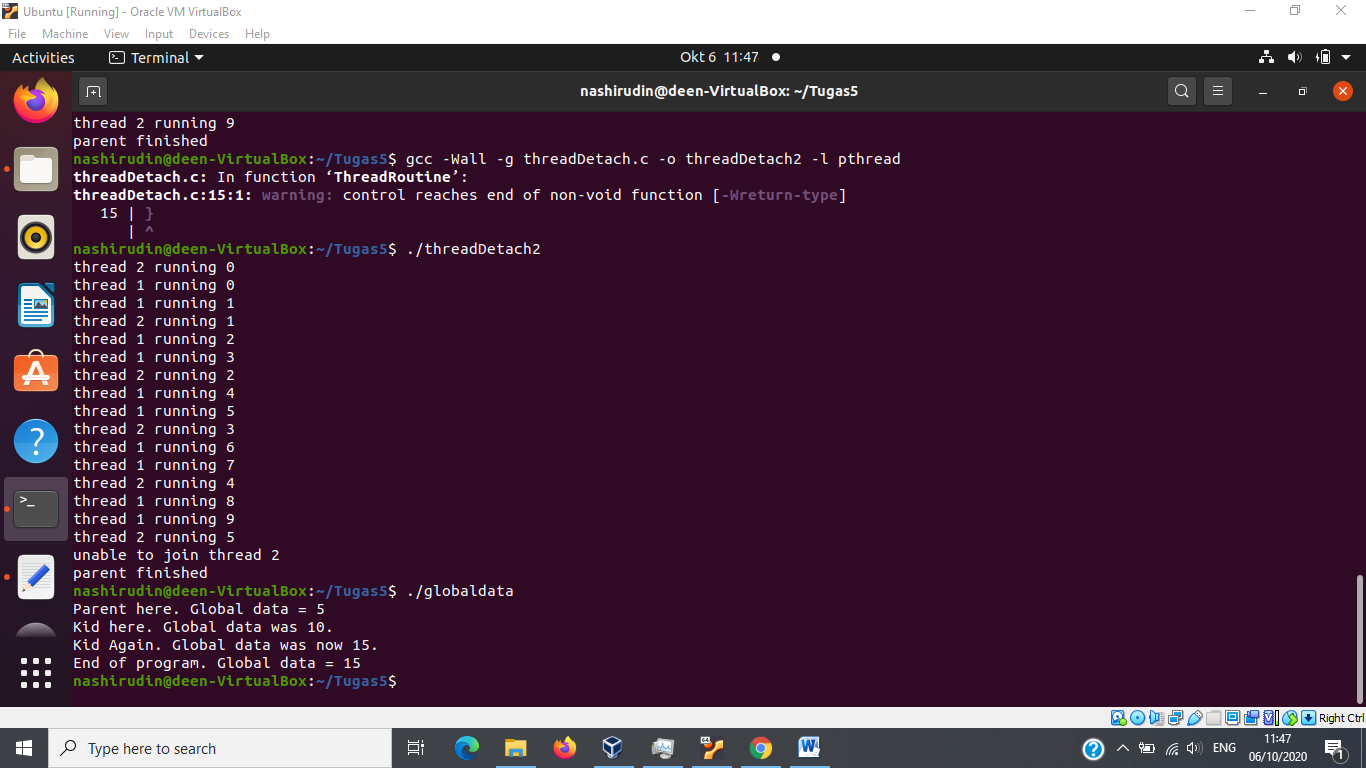
Fungsi pthread\_detach pada program threadDetach.c yaitu menandai thread bila terminated maka resource yang dipakai dikembalikan ke sistem secara otomatis.

Perbedaan dari detach tid1 dan tid2 adalah, detach tid1 resource tetap berjalan disistem meski tid1 terminated sehingga tid2 tetap berjalan sedangkan detach tid2 berhenti ketika tid1 terminated padahal membutuhkannya.

* 1. globaldata.c



Output:



Penjelasan globaldata.c, definisi kid sebagai pthread\_t. Kemudian pthread\_create membuat thread kid namun kidfunc menunggu parent jalan dahulu. Parent mempunyai 5 global data. Kemudian global data ditambah 5 dari kid pertama menjadi 10. Thread kid join ke parent. Barulah dijalankan kidfunc karena parent menunggu kid dahulu. Membuat kid lagi bertambah 5 global data menjadi 15.

Pada percobaan globaldata.c suatu variabel **dapat** diakses oleh semua thread. Buktinya yaitu parent, kid lalu kidnya lagi semua terhitung sebagai global data.

1. 5b Penjadwalan Proses

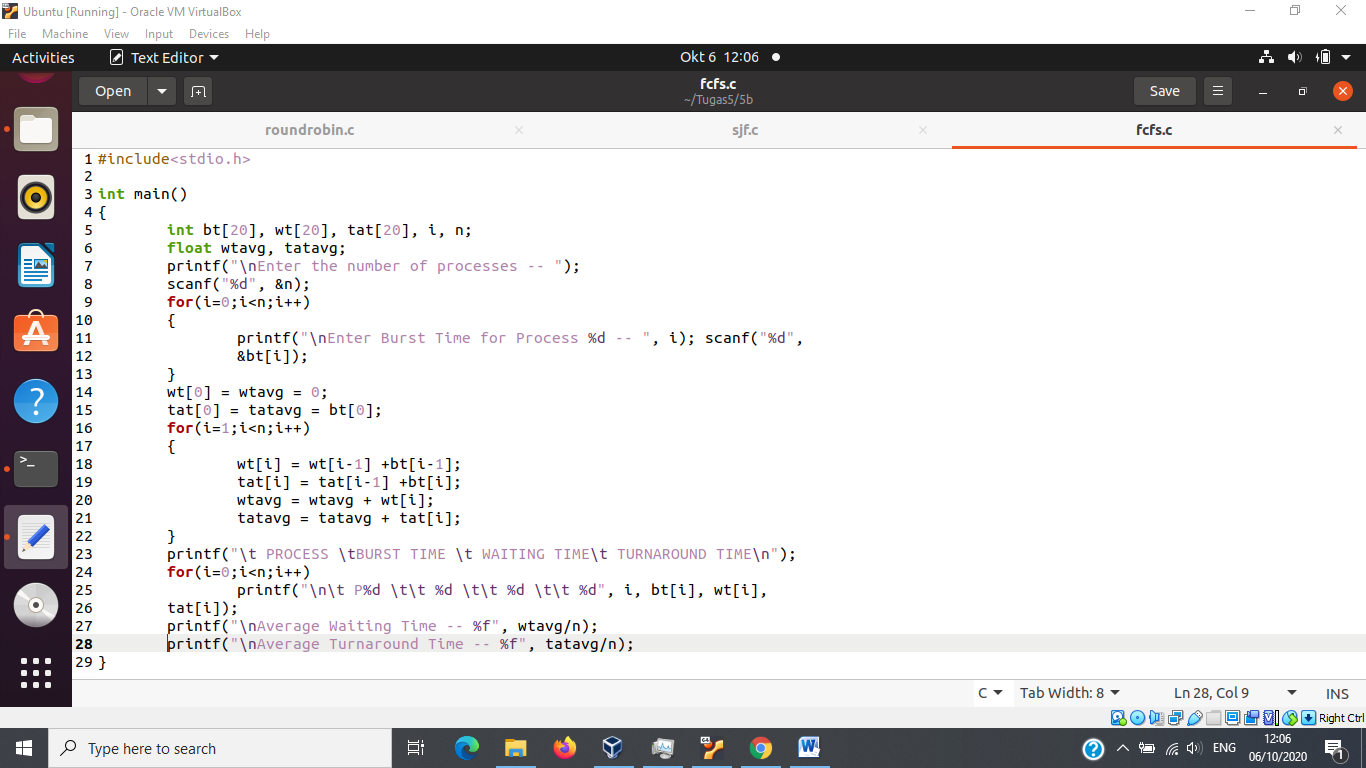
First come first serve=datang lebih dulu maka lebih dulu dilayani, sjf=shortest job first, pekerjaan paling kecil selesaikan lebih dulu

Roundrobin mendukung multitasking.

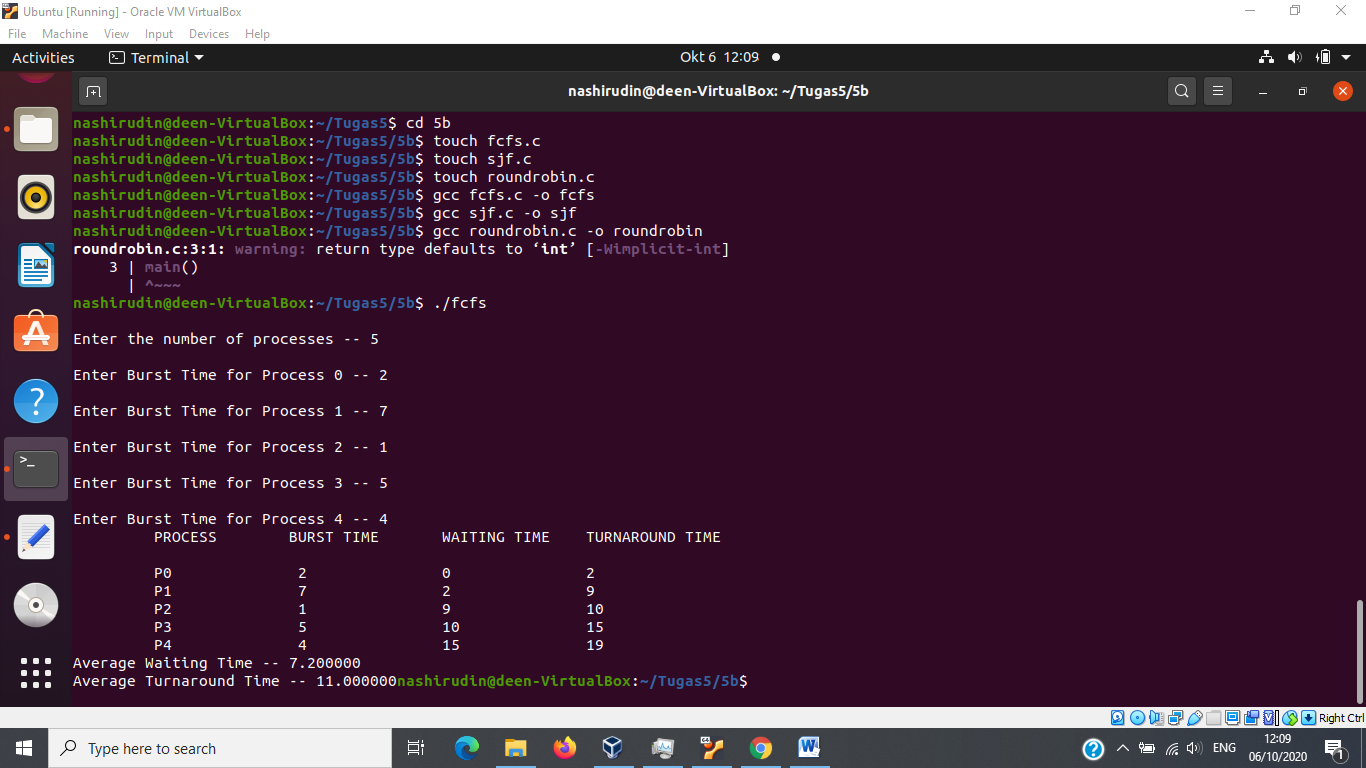
1. Bandingkan implementasi program FCFS dengan SJF, mana yang lebih ringkas? Beri penjelasannya.

**FCFS**

fcfs.c

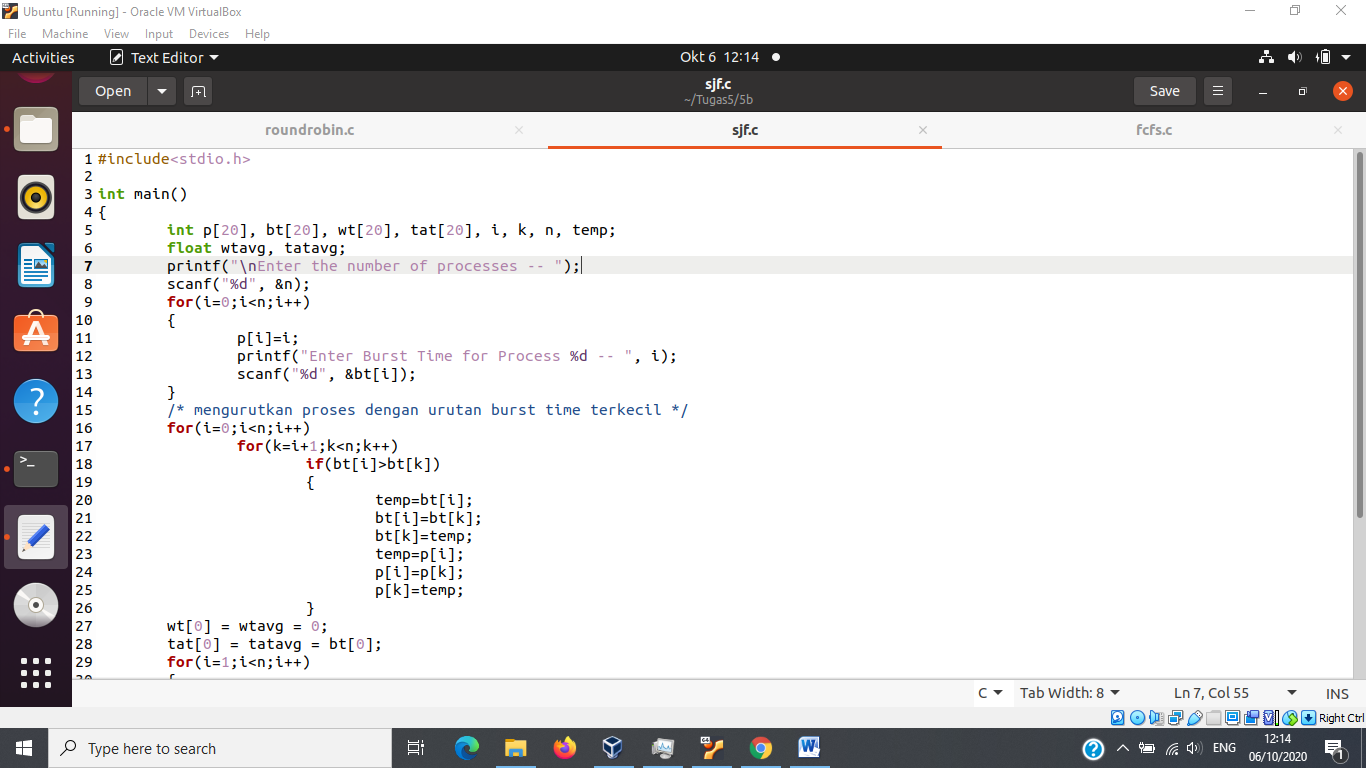


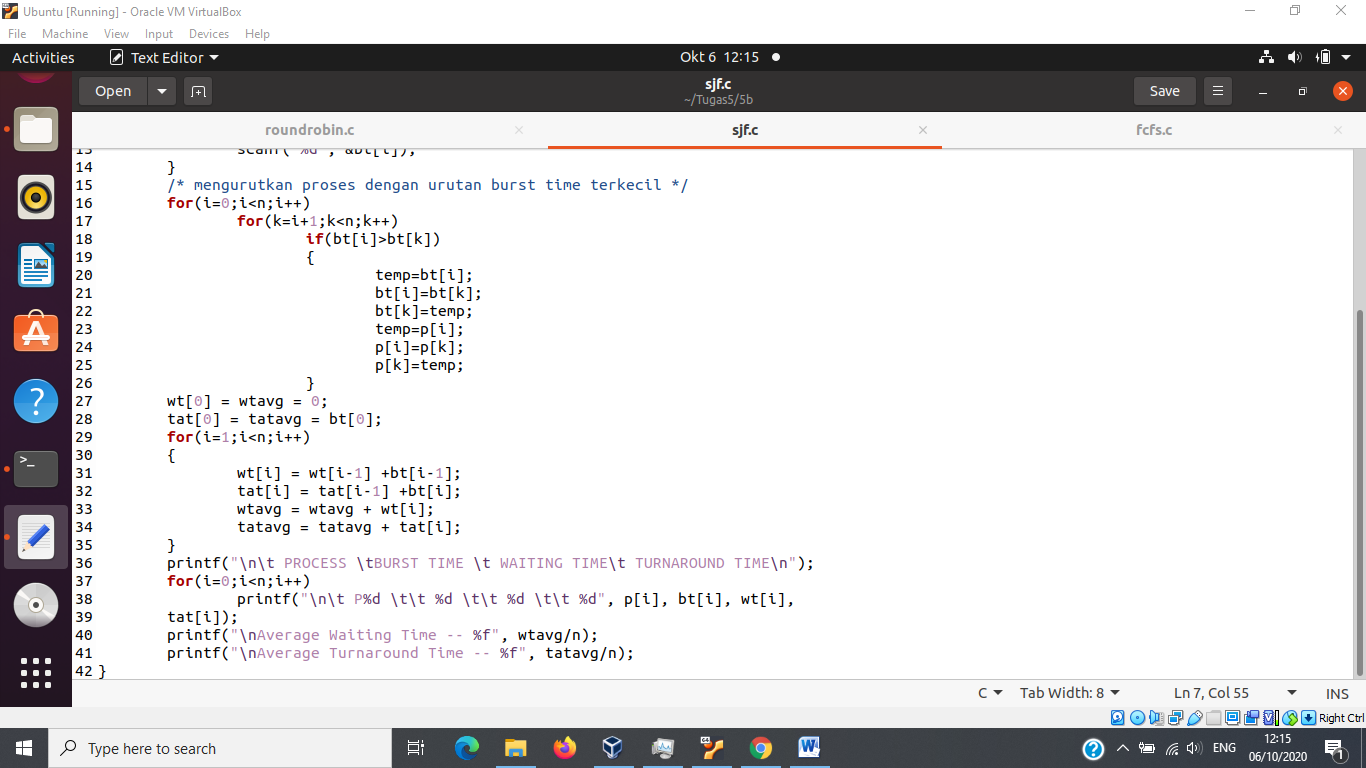
Output:



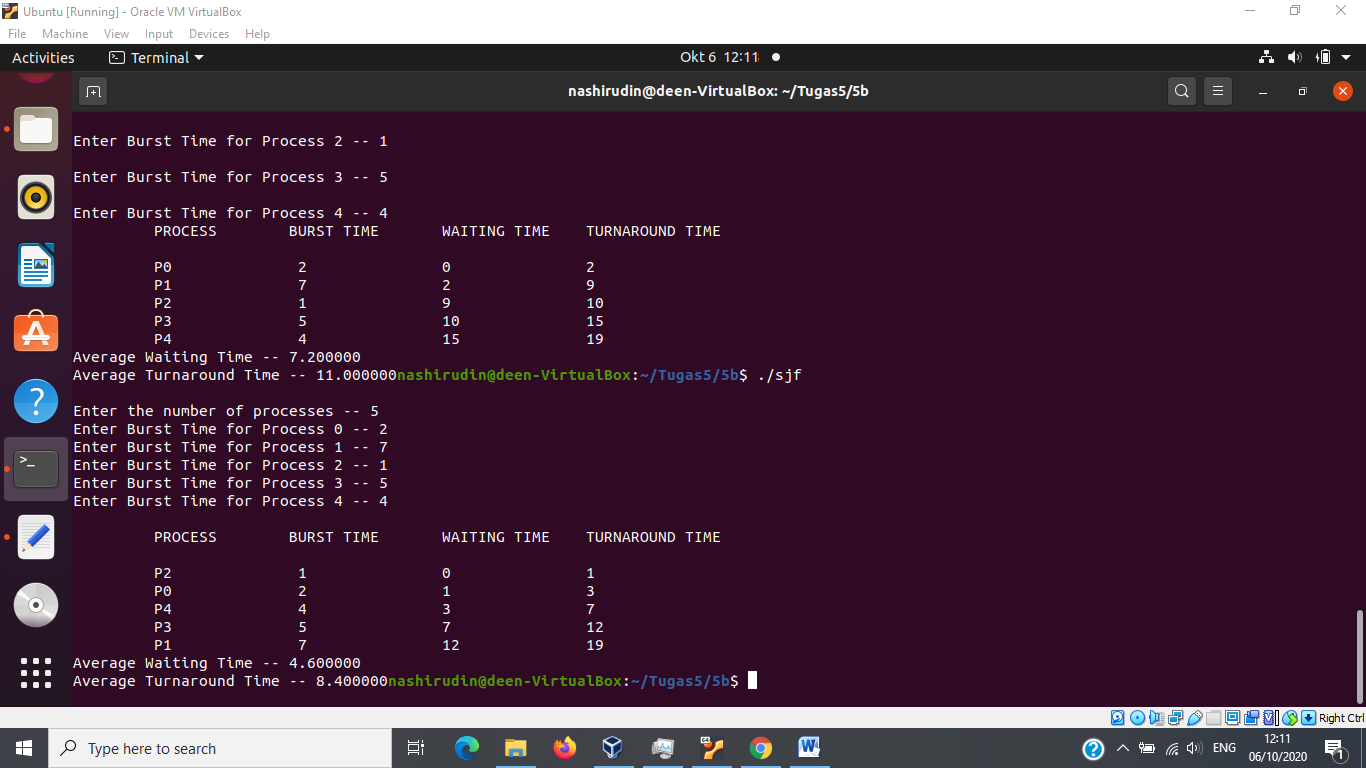
**SJF**

sjf.c





Output:

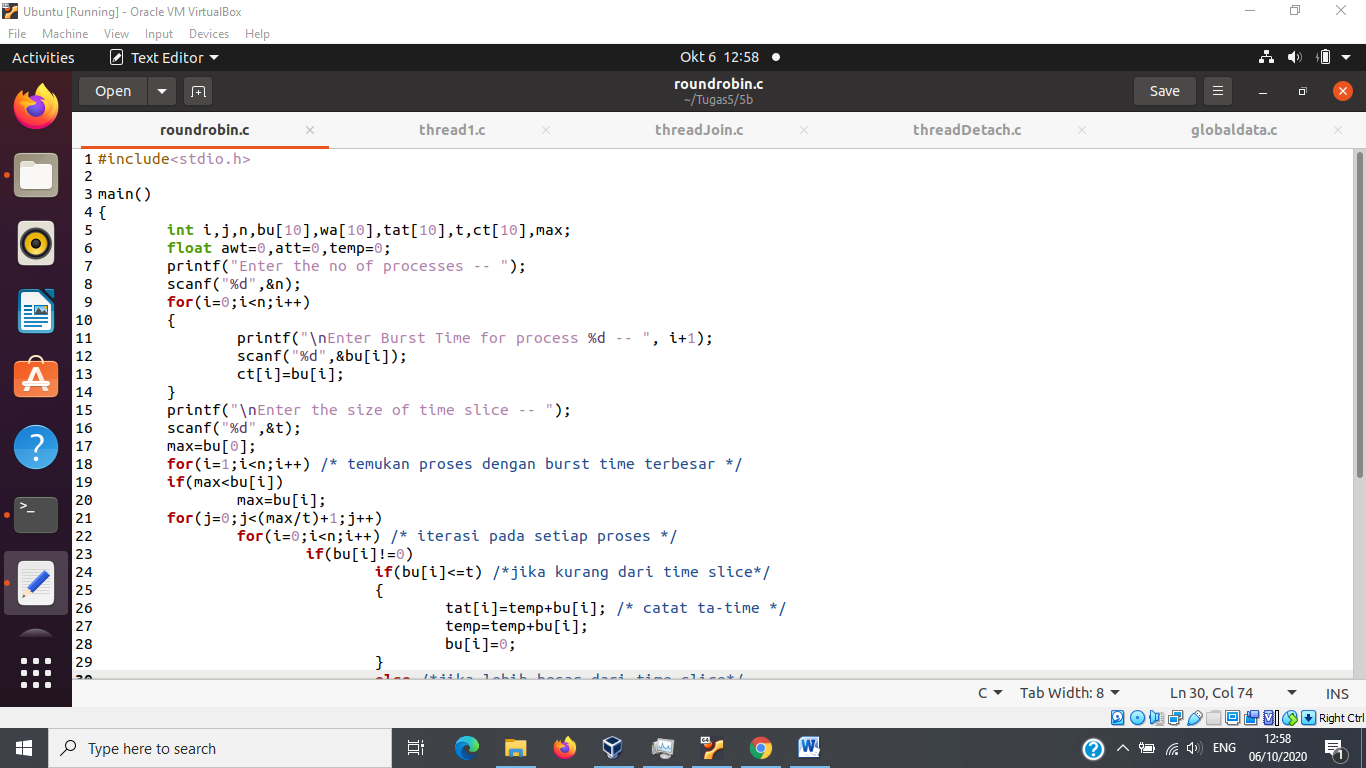


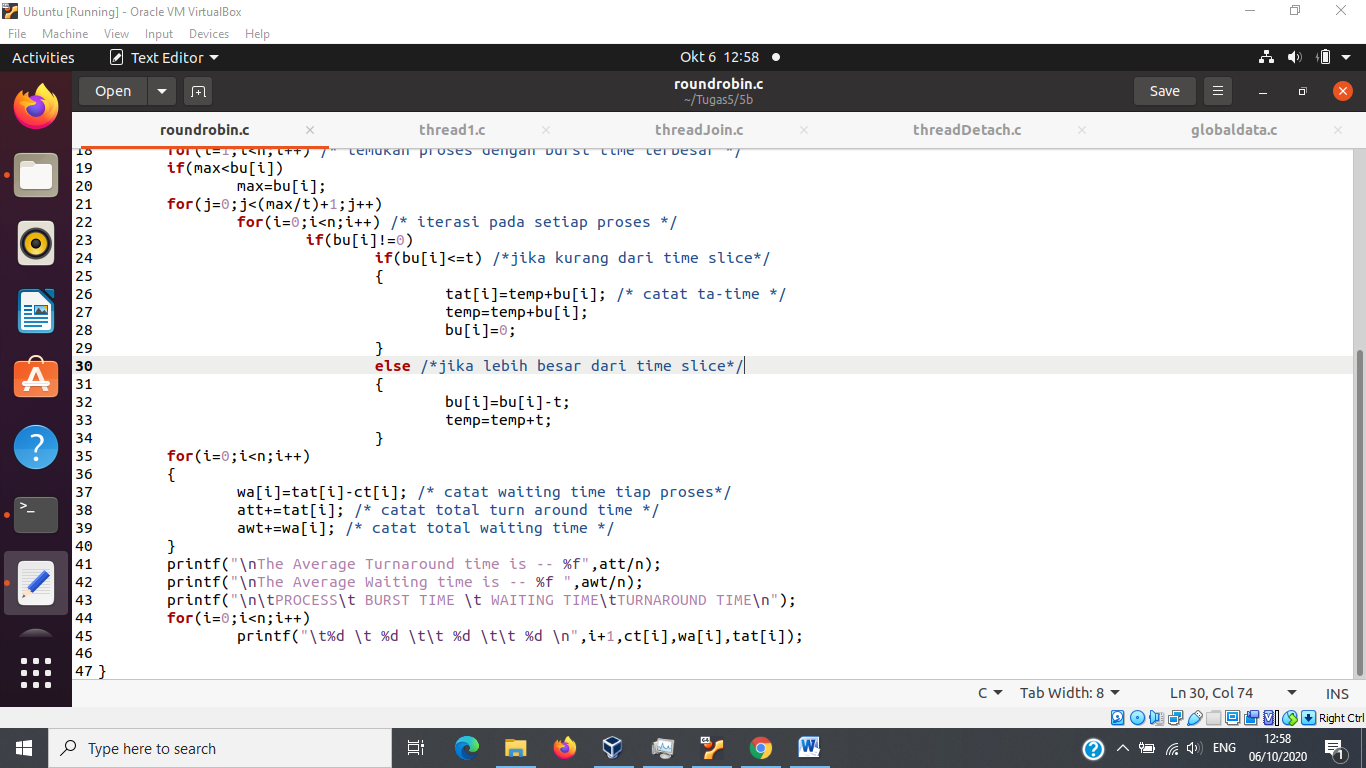
Dapat dilihat perbandingannya, average waiting time lebih rendah SJF dan average turnaround time lebih rendah sjf. Dapat disimpulkan SJF lebih ringkas karena SJF mengefisienkan pengambilan process dari yang terpendek lebih dahulu. Awal burst time pendek dan beban burst time tertinggi di akhir.

1. Dengan jumlah proses dan nilai burst time yang sama pada percobaan FCFS, SJF, dan RR, bandingkan nilai average waiting-time dan average turn-around time, algoritma mana yang terbaik?

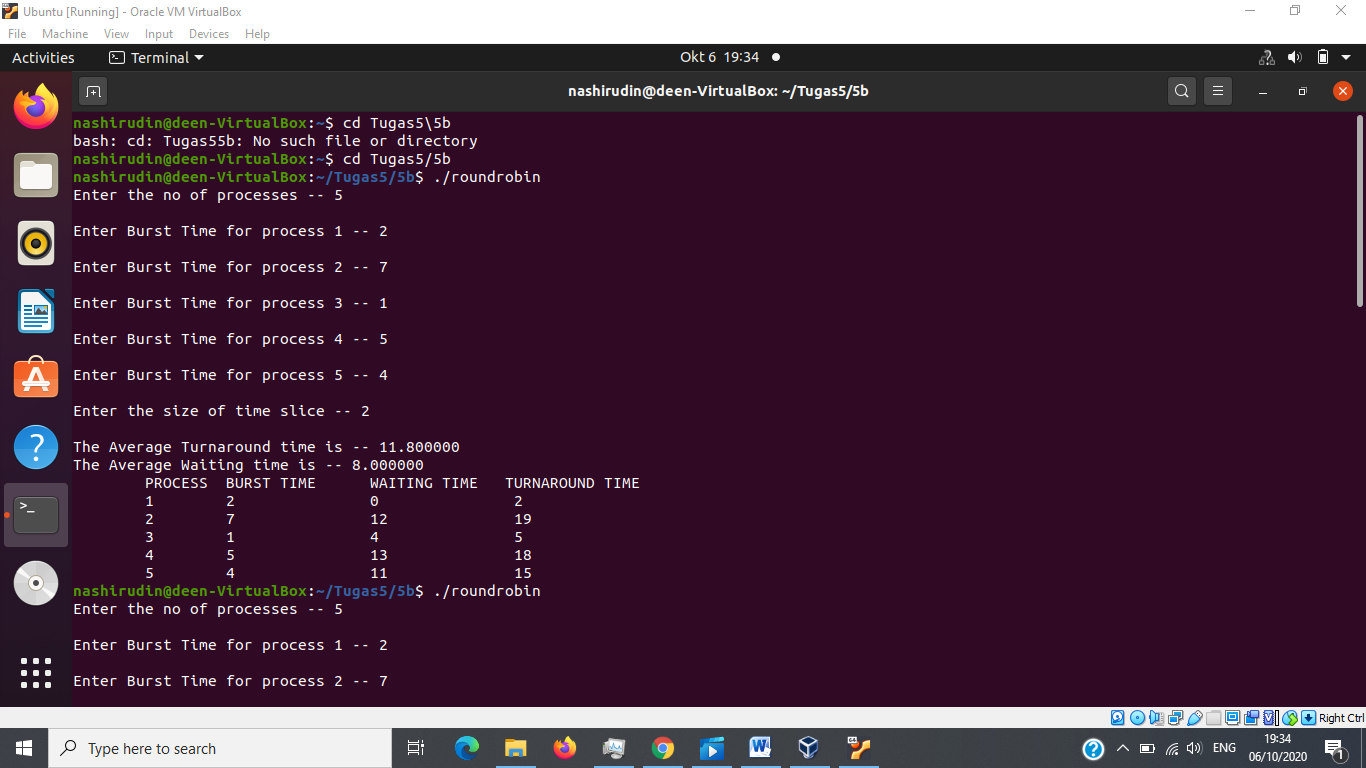
**RR**

roundrobin.c

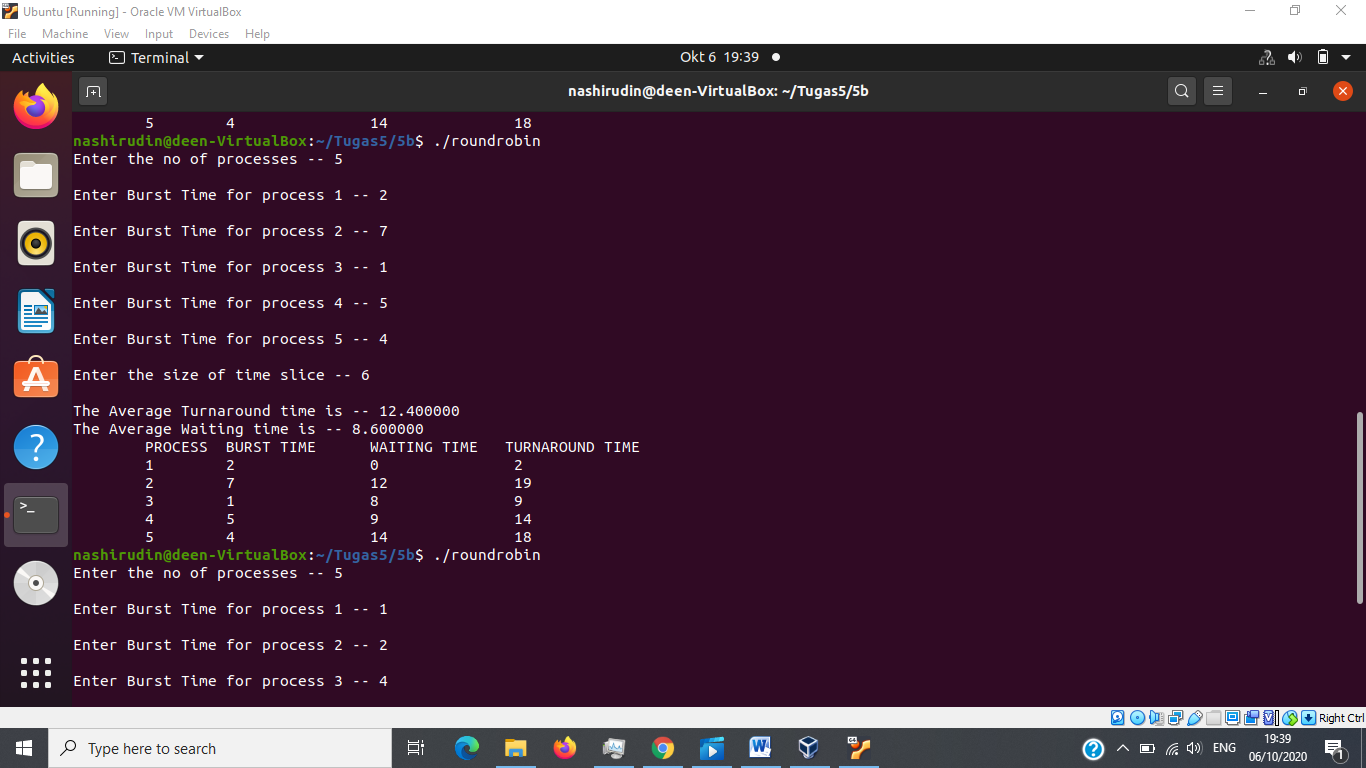




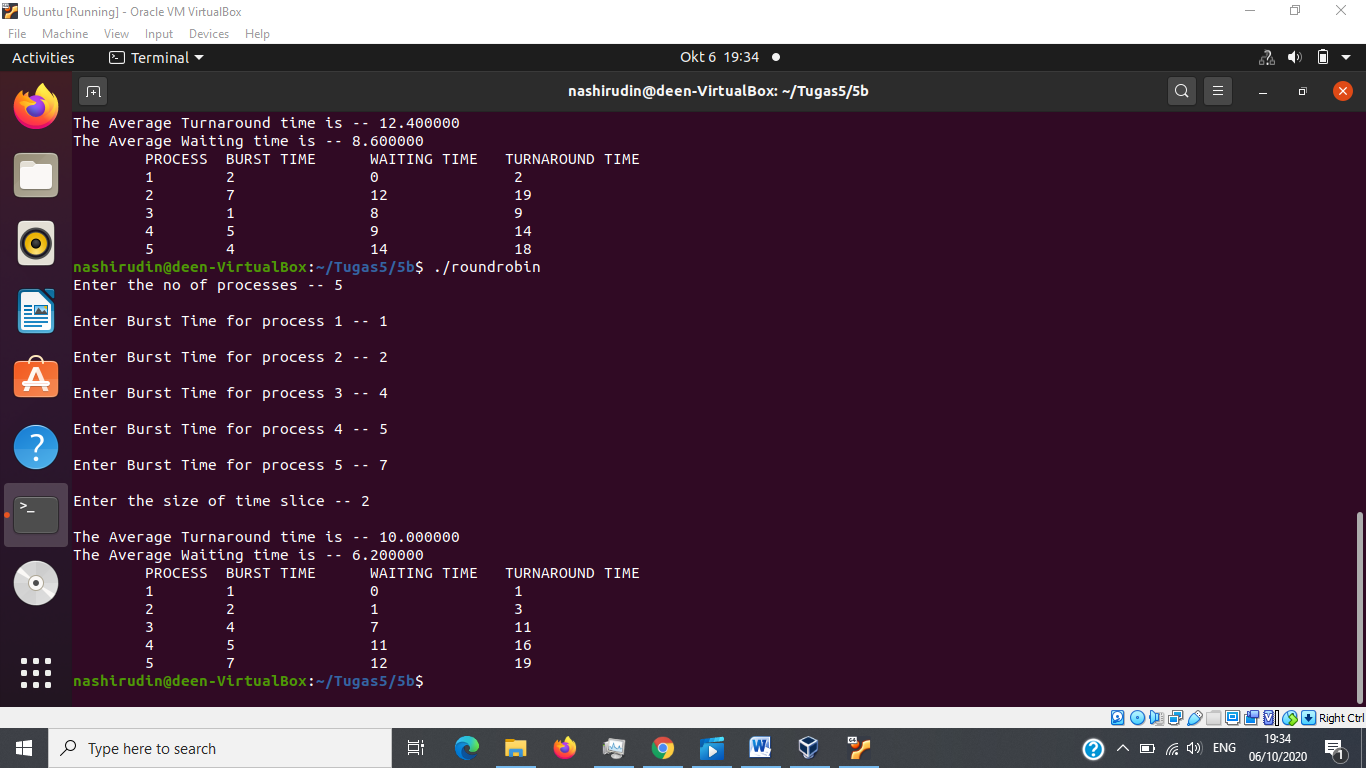
Output roundrobin (tidak urut dari terkecil) time slice 2:



Output roundrobin time slice 6:

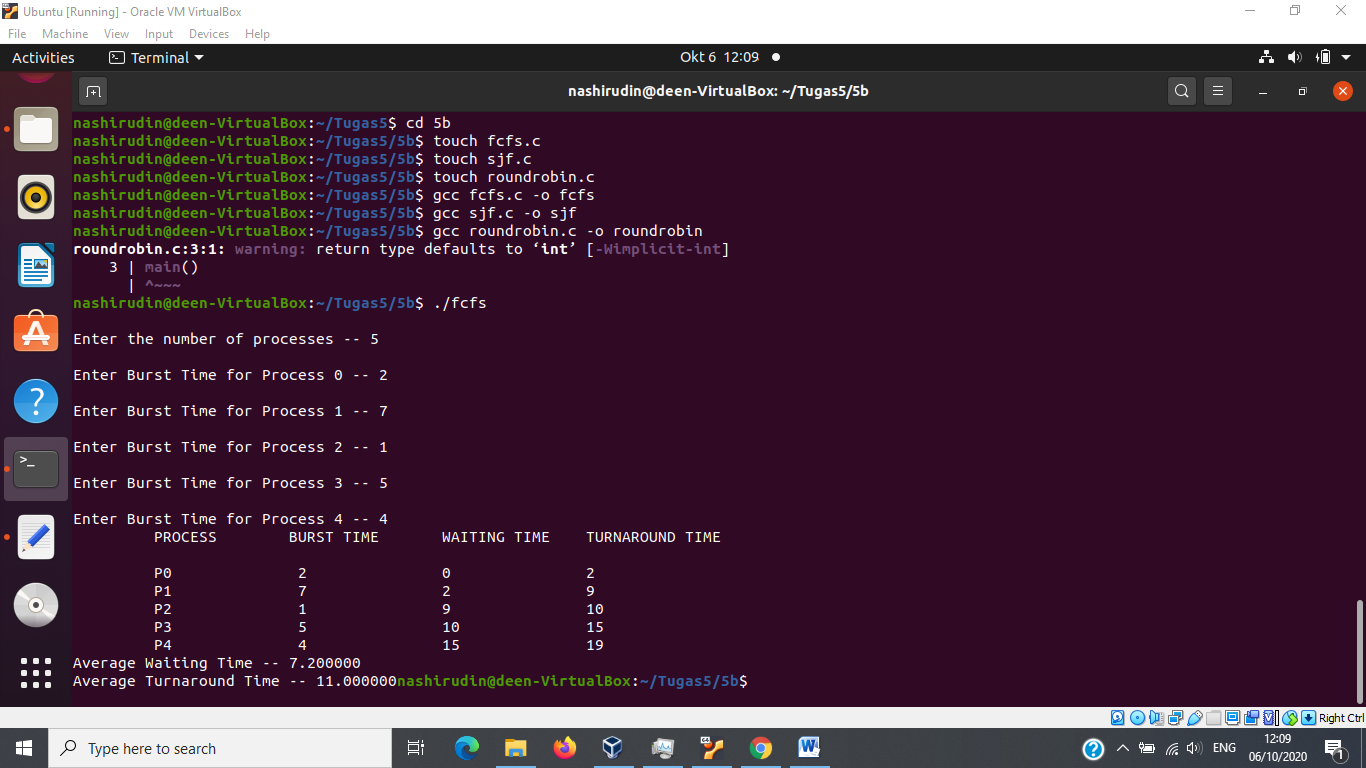


Output roundrobin (urut dari terkecil) time slice 2:

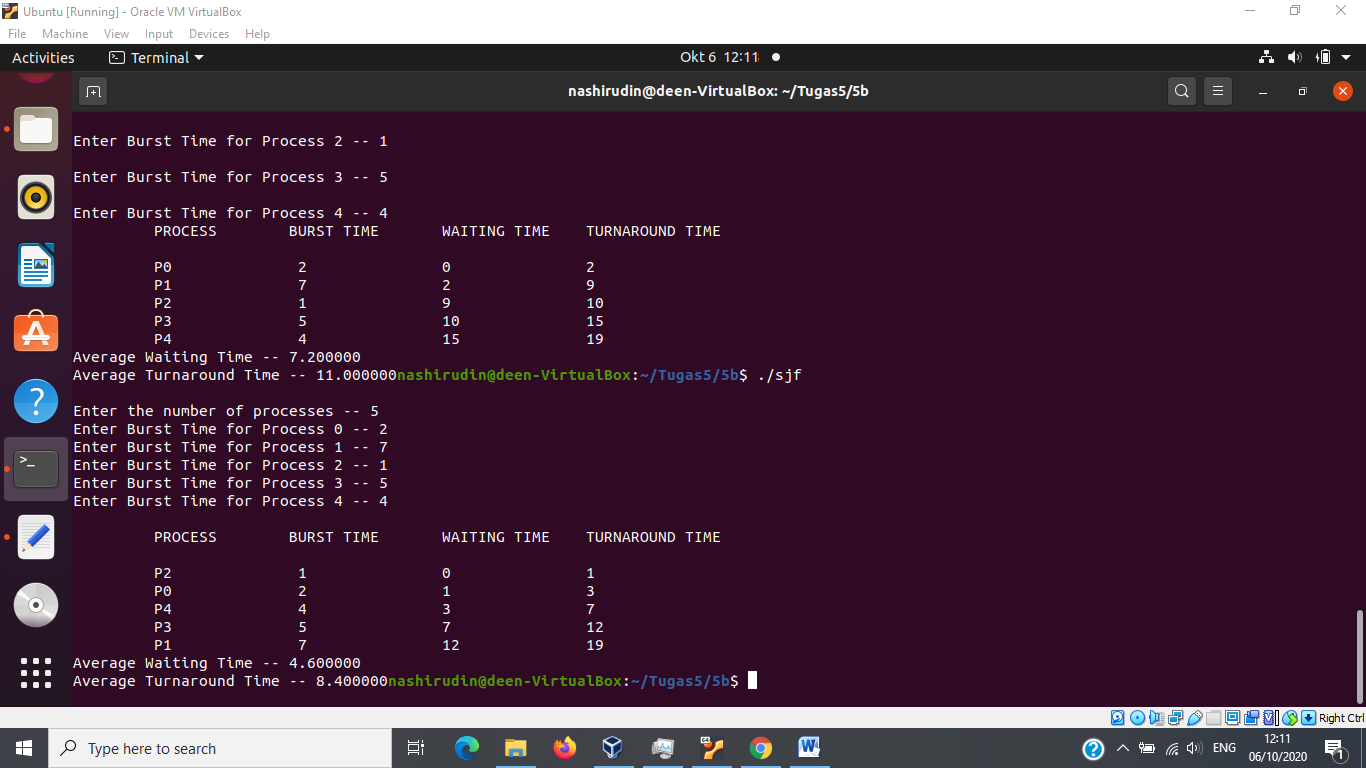


RoundRobin akan semakin rendah average turnaround dan waiting timenya apabila menggunakan konsep SJF yaitu dari terpendek. Time slice juga berpengaruh, time slice lebih baik melihat burst time tiap proses agar time slice memiliki waiting time sesedikit mungkin tiap proses tiap round.

Output FCFS:



Output SJF:



Berdasarkan average waiting time dan average turnaround time menjadikan SJF sebagai algoritma yang terbaik. Urut dari terkecil sangat mempengaruhi waiting time, sehingga juga mendapatkan turnaround time yang minimum. Dapat disimpulkan bahwa konsep algoritma SJF yang urut dari terkecil pemenang terbaik.

1. Berdasarkan hasil pengamatan terhadap ketiga algoritma penjadwalan, apakah algoritma RR memiliki kelebihan dibandingkan FCFS dan SJF? Beri penjelasannya (silahkan merujuk buku referensi)

Kelebihan dari algoritma RR (RoundRobin) dibandingkan FCFS dan SJF adalah RR dapat multitasking untuk mempersingkat waiting time apabila burst time antar proses memiliki kesenjangan yang rendah dan memiliki time slice serendah mungkin dari rata-rata burst time yang dapat menyelesaikan banyak proses dalam ronde sesedikit mungkin.