

PRAKTIKUM SO

TUGAS 3

Nama : Nashirudin Baqiy

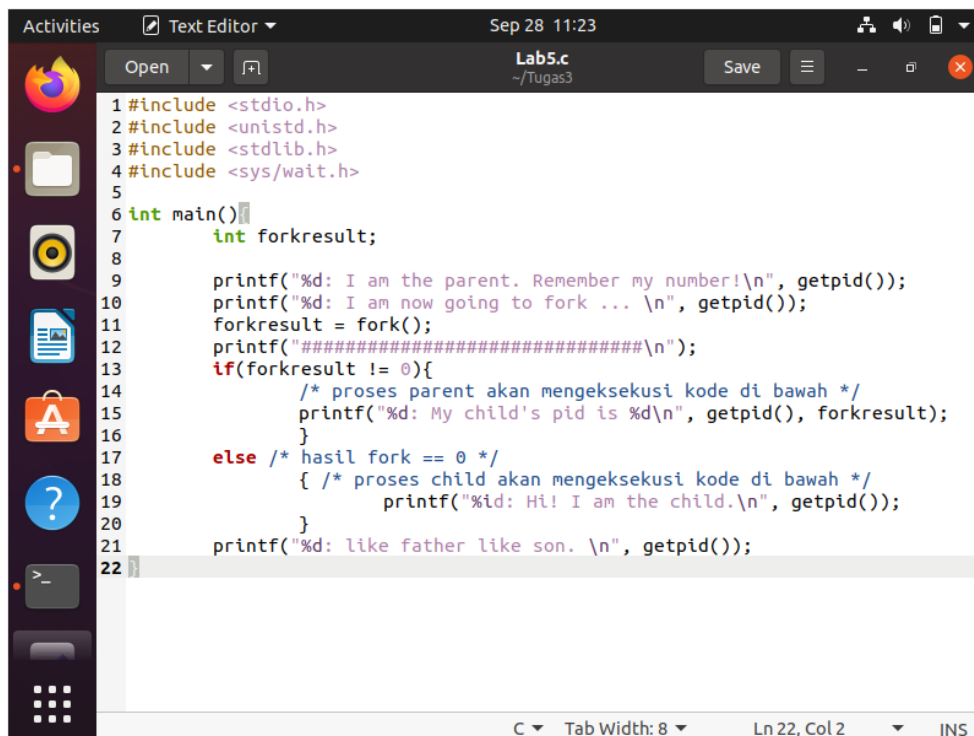
NIM : 24060119130045

Kelas : A1

Percobaan 5: Pembuatan dan Eksekusi Proses

Sleep

a. Lab5.c



```
Activities Text Editor Sep 28 11:23
Lab5.c
~/Tugas3
Save

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <sys/wait.h>
5
6 int main()
7 {
8     int forkresult;
9
10    printf("%d: I am the parent. Remember my number!\n", getpid());
11    printf("%d: I am now going to fork ... \n", getpid());
12    forkresult = fork();
13    printf("#####\n");
14    if(forkresult != 0){
15        /* proses parent akan mengeksekusi kode di bawah */
16        printf("%d: My child's pid is %d\n", getpid(), forkresult);
17    }
18    else /* hasil fork == 0 */
19    { /* proses child akan mengeksekusi kode di bawah */
20        printf("%d: Hi! I am the child.\n", getpid());
21    }
22    printf("%d: like father like son. \n", getpid());
23 }
```

C Tab Width: 8 Ln 22, Col 2 INS

Ouput:

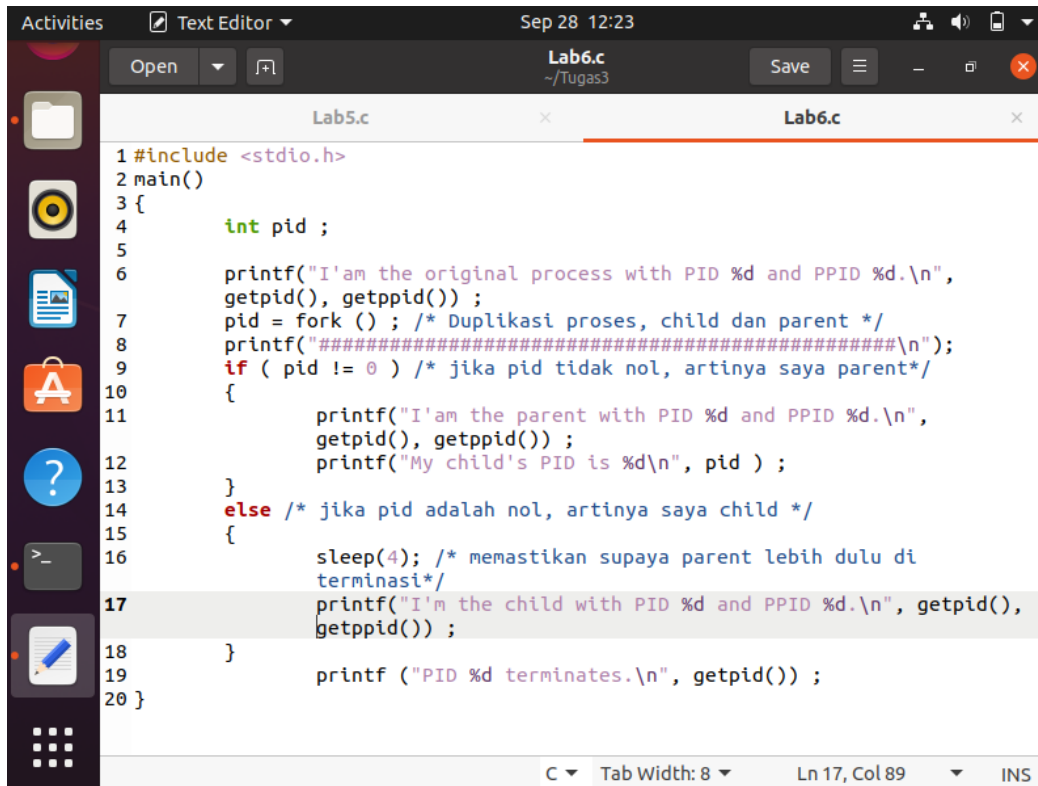
```
nashirudin@deen-VirtualBox:~/Tugas3$ ./Lab5
21481: I am the parent. Remember my number!
21481: I am now going to fork ...
#####
21481: My child's pid is 21482
21481: like father like son.
nashirudin@deen-VirtualBox:~/Tugas3$ #####
21482d: Hi! I am the child.
21482: like father like son.
```

Kesimpulan:

Memanggil unistd.h yang berisi prototype fork dan sys/wait.h yang mengandung fungsi wait. Int forkresult digunakan sebagai simpanan fungsi fork(). Fungsi getpid() yaitu mendapatkan proses id lalu dijalankan fork untuk membuat child (proses baru). Fungsi fork() mengembalikan nilai 0 dalam proses child dan mengembalikan proses id child dalam proses parent. Pertama masuk if karna dalam proses parent, lalu else karna dalam proses child menghasilkan child dari child's parent.

Proses Orphan

b. Lab6.c



```
1 #include <stdio.h>
2 main()
3 {
4     int pid ;
5
6     printf("I'am the original process with PID %d and PPID %d.\n",
7         getpid(), getppid());
8     pid = fork () ; /* Duplikasi proses, child dan parent */
9     printf("#####\n");
10    if ( pid != 0 ) /* jika pid tidak nol, artinya saya parent*/
11    {
12        printf("I'am the parent with PID %d and PPID %d.\n",
13            getpid(), getppid());
14        printf("My child's PID is %d\n", pid ) ;
15    }
16    else /* jika pid adalah nol, artinya saya child */
17    {
18        sleep(4); /* memastikan supaya parent lebih dulu di
19        terminasi*/
20        printf("I'm the child with PID %d and PPID %d.\n", getpid(),
21            getppid());
22    }
23    printf ("PID %d terminates.\n", getpid()) ;
24 }
```

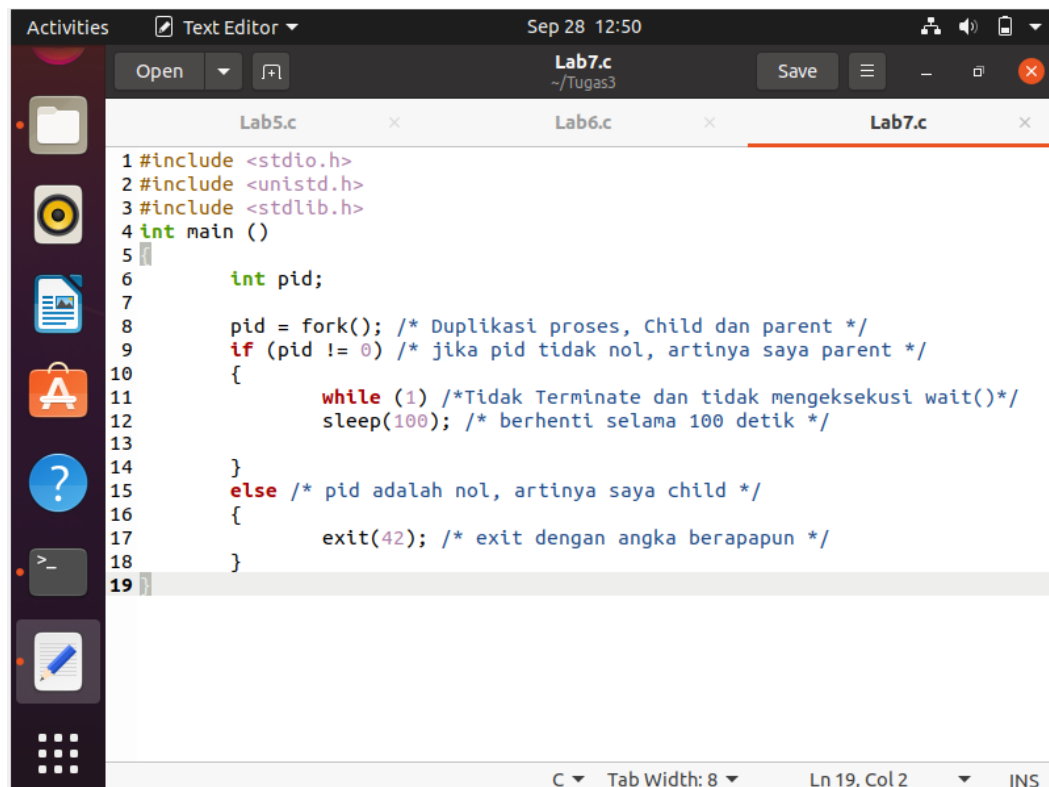
Output:

```
nashirudin@deen-VirtualBox:~/Tugas3$ ./Lab6
>_ I'am the original process with PID 22171 and PPID 2192.
#####
I'am the parent with PID 22171 and PPID 2192.
My child's PID is 22172
PID 22171 terminates.
nashirudin@deen-VirtualBox:~/Tugas3$ #####
#####
I'm the child with PID 22172 and PPID 1144.
PID 22172 terminates.
```

Kesimpulan:

Int pid yaitu untuk simpanan fork(). Pertama getpid() menampilkan proses id lalu getppid() menampilkan parent dari proses id. Dijalankan fork() lalu didapatkan nilai fork adalah pid tidak nol yaitu parent. Masuk if diprint pid dari child, parent diterminasi. Lalu dijalankan lagi masuk else karena pid 0 yaitu child, sleep(4) untuk memastikan parent diterminasi lebih dulu. Didapatkan proses id dari child diadopsi oleh proses id 1144.

c. Lab7.c



```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 int main ()
5 {
6     int pid;
7
8     pid = fork(); /* Duplikasi proses, Child dan parent */
9     if (pid != 0) /* jika pid tidak nol, artinya saya parent */
10    {
11        while (1) /*Tidak Terminate dan tidak mengeksekusi wait()*/
12            sleep(100); /* berhenti selama 100 detik */
13    }
14    else /* pid adalah nol, artinya saya child */
15    {
16        exit(42); /* exit dengan angka berapapun */
17    }
18 }
19 }
```

Output:

```
nashirudin@deen-VirtualBox:~/Tugas3$ ./Lab7 &
[1] 22930
nashirudin@deen-VirtualBox:~/Tugas3$ ps
  PID TTY          TIME CMD
 22915 pts/0    00:00:00 bash
 22930 pts/0    00:00:00 Lab7
 22931 pts/0    00:00:00 Lab7 <defunct>
 22932 pts/0    00:00:00 ps
nashirudin@deen-VirtualBox:~/Tugas3$ kill 22930
[1]+  Terminated                  ./Lab7
nashirudin@deen-VirtualBox:~/Tugas3$ ps
  PID TTY          TIME CMD
 22915 pts/0    00:00:00 bash
 22933 pts/0    00:00:00 ps
nashirudin@deen-VirtualBox:~/Tugas3$
```

Kesimpulan:

Pada Lab7.c ini mencoba membuat proses zombie, yaitu proses yang tidak dapat meninggalkan system sampai parent-nya menerima kode pengembalian. Proses parent pada pid pertama dihidupkan dan tidak pernah mengeksekusi wait() sehingga proses child akan menjadi zombie. PID 22931 yang ditandai <defunct> merupakan anak zombie. Untuk menghilangkannya kita perlu memberi kode pengembalian kepada parent, yaitu kill 22930 (terminasi parent). Setelah dicek dengan ps, zombie sudah hilang.