

Laporan Praktikum

Sistem Operasi

Manajemen Proses



Nama : Josep Phyto Napitupulu

NIM : 11421039

**Program Studi: DIV TEKNOLOGI REKAYASA
PERANGKAT LUNAK**

**INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI**

A. Teori

1. Pertanyaan berikut terkait dengan konsep-konsep dasar Proses

a. Definisikan program.

Jawab:

Program Adalah suatu rancangan struktur, desain, kode skema, maupun dengan bentuk yang lain yang disusun sesuai alur Algoritma dengan tujuan mempermudah suatu permasalahan.

b. Definisikan proses.

Jawab:

Proses adalah rangkaian tindakan, perbuatan, atau pengolahan yang mengubah masukan menjadi keluaran.

c. Definisikan Zombie Process

Jawab:

Zombie process adalah sebuah proses pada sistem operasi yang telah menyelesaikan eksekusinya tetapi terdapat pada entry process. Pada proses ini sudah berhenti dieksekusi oleh CPU. Ketika proses ini berhenti dieksekusi proses ini akan mengirimkan signal SIGCHLD ke parent proses.

d. Definisikan Orphan Process

Jawab:

Orphan Process adalah sebuah proses yang ada dalam komputer dimana parent process telah selesai atau berhenti bekerja namun proses anak sendiri tetap berjalan.

e. Jelaskanlah Process Control Block (PCB)

- Keadaan proses: Keadaan mungkin, new ,ready ,running, waiting, halted, dan juga banyak lagi.
- Program counter: Counter mengindikasikan address dari perintah selanjutnya untuk dijalankan untuk ditambah code information pada kondisi apapun. Besertaan dengan program counter, keadaan/ status informasi harus disimpan ketika gangguan terjadi, untuk memungkinkan proses tersebut berjalan/bekerja dengan benar setelahnya.
- Informasi manajemen memori: Informasi ini dapat termasuk suatu informasi sebagai nilai dari dasardan batas register. tabel page/ halaman, atau tabel segmen tergantung pada sistem memori yangdigunakan oleh sistem operasi (ch 9).
- Informasi pencatatan: Informasi ini termasuk jumlah dari CPU dan waktu riil yang digunakan bataswaktu, jumlah akun, jumlah job atau proses, dan banyak lagi.
- Informasi status I/O: Informasi termasuk daftar dari perangkat I/O yang di gunakan pada proses ini,suatu daftar open file dan banyak lagi.

- PCB hanya berfungsi sebagai tempat menyimpan/gudang untuk informasi apapun yang dapat bervariasi dari proses ke proses.
- CPU register: Register bervariasi dalam jumlah dan jenis, tergantung pada rancangan komputer. Register tersebut termasuk accumulator, index register, stack pointer, general-purpose register.

2. Kode program.

```
#include <stdio.h>

int main(){
    pid_t pid;

    pid = fork();

    if (pid < 0){
        fprintf(stderr, "Fork Failed");
        return 1;
    } else if (pid == 0){
        execlp("/bin/ls", "ls", NULL);
    } else {
        wait(NULL);
        printf("Child complete\n");
    }
    return 0;
}
```

[Wrote 20 lines]

```
[root@localhost ~]# cc fork.c -o fork
[root@localhost ~]# ./fork
anaconda-ks.cfg  eth1  foo  foo.c  fork  fork.c  hello word  program1  program1
Child complete
[root@localhost ~]# .
```

3. Dari kode program di atas jelaskan apa yang dimaksud dengan:

a. pid_t

Jawab:

Merupakan tipe data pada Bahasa C yang digunakan pada Linux serta memiliki kepanjangan process identification dan digunakan untuk mempresentasikan id dari process. Tipe data ini dapat digunakan dengan meng-include kamus/header `sys/types.h`.

b. fork()

Jawab:

Merupakan suatu system call yang digunakan untuk membuat sebuah proses baru, yang disebut child process, yang dimana berjalan secara bersamaan dengan proses yang memanggil `fork()` (parent process).

c. `execlp()`

Jawab:

Merupakan sebuah system call untuk menjalankan command pada terminal dengan command yang akan dijalankan terdapat pada parameter yang dikirimkan melalui function.

d. `wait()`

Jawab:

Merupakan sebuah system call untuk menunda proses pemanggilan parent process hingga satu dari children process-nya keluar sebuah pertanda diterima. Setelah children process berhenti dieksekusi, parent process melanjutkan pengeksesksiannya setelah menunggu instruksi system call `wait()`.

4. Beri tanda bagian program yang merupakan proses induk (parent process) dan proses anak (child process).

```
#include <stdio.h>

int main(){
    pid_t pid;

    pid = fork();

    if(pid<0){
        fprintf(stderr, "Fork Failed");
        return 1;
    }else if(pid == 0){
        execl("/bin/ls", "ls", NULL);
    }else{
        wait(NULL);
        printf("Child complete\n");
    }
    return 0;
}
```

[Wrote 20 lines]

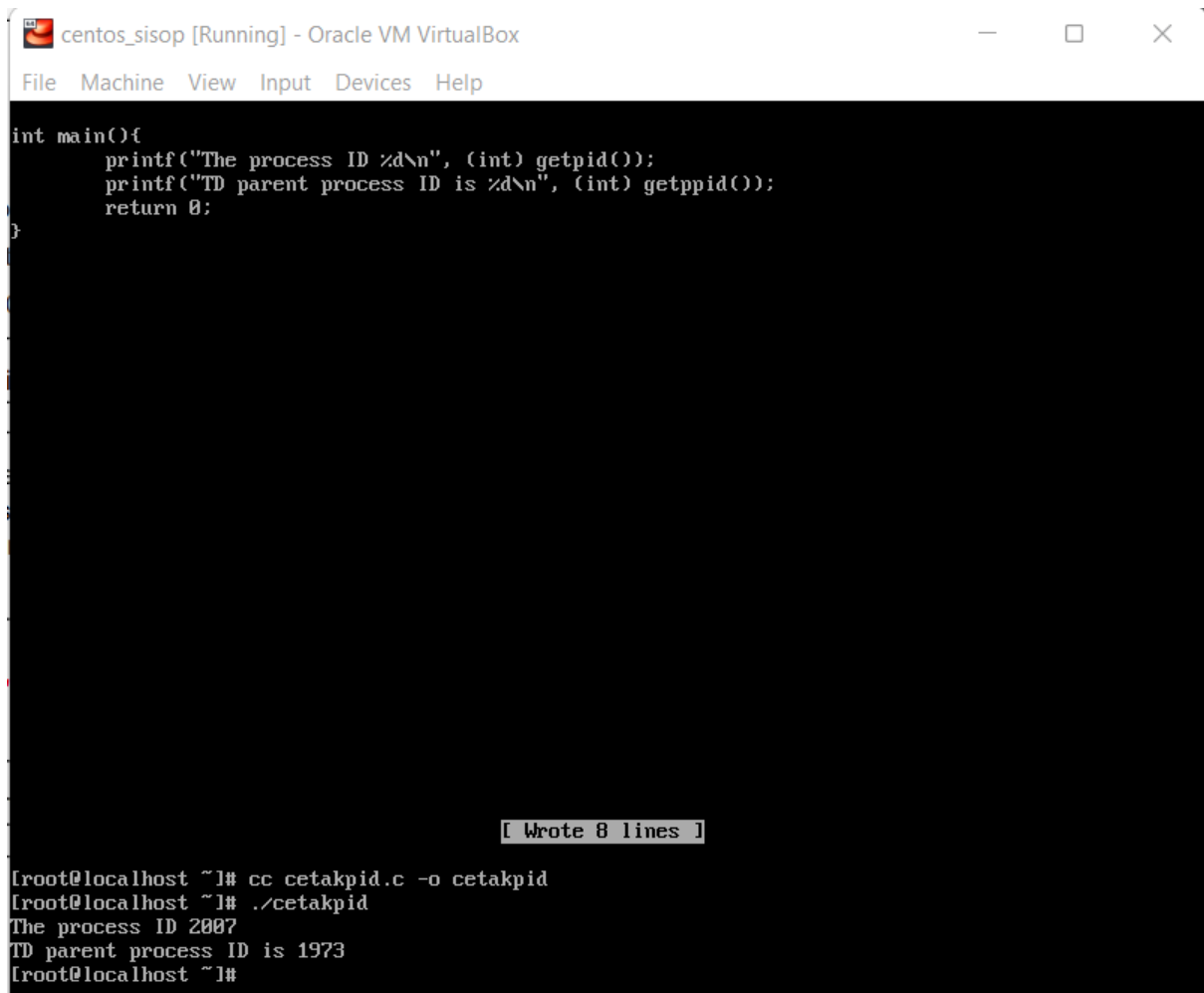
```
root@localhost ~# cc fork.c -o fork
root@localhost ~# ./fork
anaconda-ks.cfg cth 1 foo foo.c fork fork.c hello word program1 program 1
Child complete
root@localhost ~# .
```

kotak merah merupakan child process

kotak biru merupakan parent process

B. Pemograman

1. Kode program



```
centos_sisop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

int main(){
    printf("The process ID %d\n", (int) getpid());
    printf("TD parent process ID is %d\n", (int) getppid());
    return 0;
}

[ Wrote 8 lines ]

[root@localhost ~]# cc cetakpid.c -o cetakpid
[root@localhost ~]# ./cetakpid
The process ID 2007
TD parent process ID is 1973
[root@localhost ~]#
```

- a. Jelaskan perbedaan antara `getpid()` dengan `getppid()`.

Jawab:

`getpid()` adalah function yang mengembalikan ID proses yang sedang berjalan.

`getppid()` adalah function yang mengembalikan ID proses dari parent process yang melakukan pemanggilan proses atau yang sedang berjalan prosesnya.

- b. Jelaskan mengapa setiap kali program di atas dieksekusi, maka akan menampilkan process ID yang berbeda. Jelaskan mengapa?

Jawab:

ketika dijalankan secara terus menerus dan berturut-turut process ID akan bertambah satu, yang dapat diartikan bahwa setiap proses baru (masing masing instans baru dengan nama program yang sama) memiliki sebuah process ID yang baru. Ketika process ID tersebut mencapai jumlah maksimumnya maka process ID akan dimulai dari angka satu.

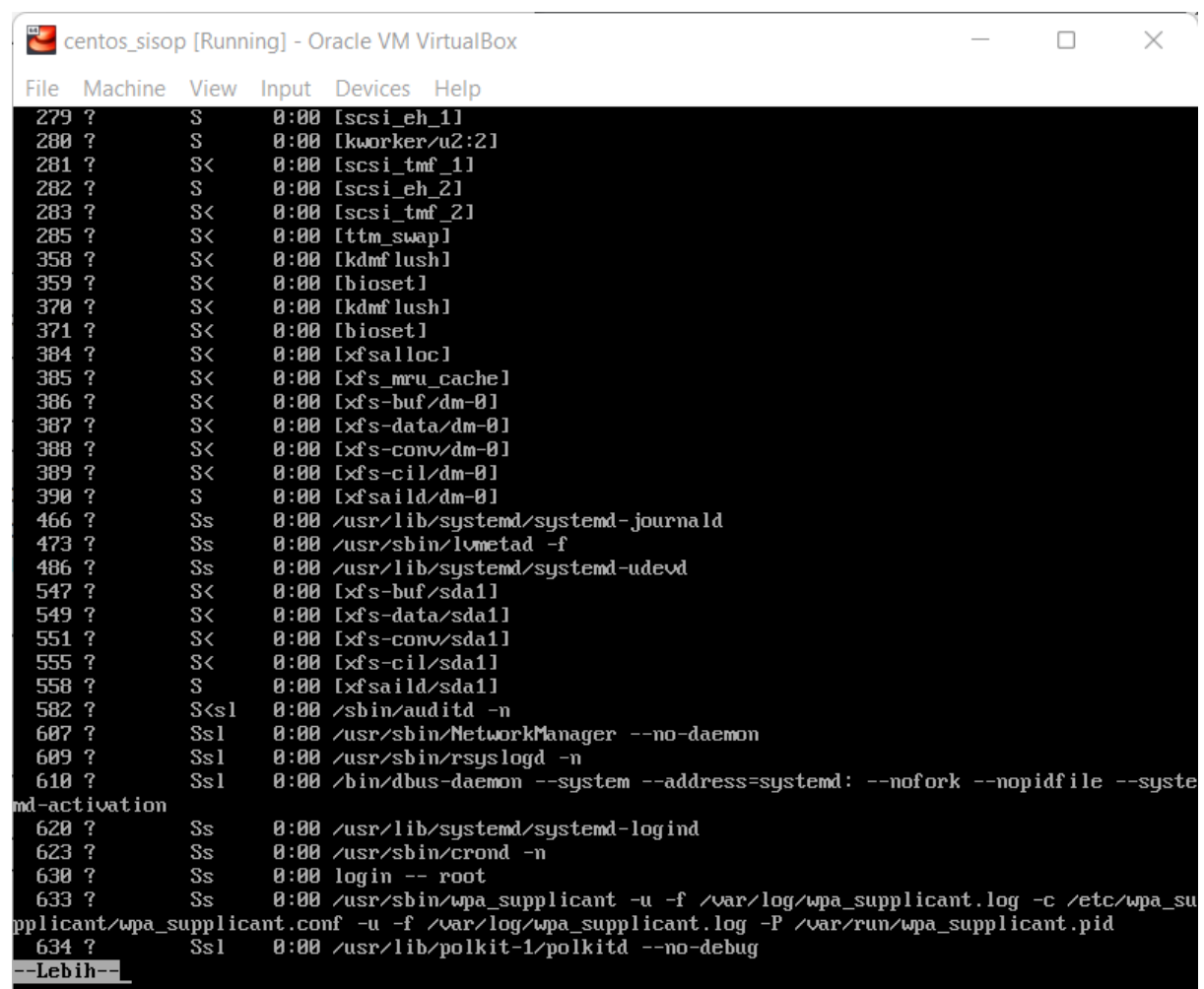
2. Pada kode program berikut, proses baru akan dibentuk dengan menggunakan fungsi `system()`.

Eksekusilah program di atas kemudian capture hasilnya. Tunjukkanlah proses mana yang menjalankan proses `ps -axl | more` dengan menandai ID proses induk-nya.

Jawab:

```
[root@localhost ~]# gcc system.c -o system
```

```
[root@localhost ~]# ./system
```



```
centos_sisop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
279 ? S 0:00 [scsi_ah_1]
280 ? S 0:00 [kworker/u2:2]
281 ? S< 0:00 [scsi_tm_1]
282 ? S 0:00 [scsi_ah_2]
283 ? S< 0:00 [scsi_tm_2]
285 ? S< 0:00 [ttm_swap]
358 ? S< 0:00 [kdmflush]
359 ? S< 0:00 [bioset]
370 ? S< 0:00 [kdmflush]
371 ? S< 0:00 [bioset]
384 ? S< 0:00 [xfsalloc]
385 ? S< 0:00 [xfs_mru_cache]
386 ? S< 0:00 [xfs-buf/dm-0]
387 ? S< 0:00 [xfs-data/dm-0]
388 ? S< 0:00 [xfs-conw/dm-0]
389 ? S< 0:00 [xfs-cil/dm-0]
390 ? S 0:00 [xfsaild/dm-0]
466 ? Ss 0:00 /usr/lib/systemd/systemd-journald
473 ? Ss 0:00 /usr/sbin/lvmtd -f
486 ? Ss 0:00 /usr/lib/systemd/systemd-udev
547 ? S< 0:00 [xfs-buf/sd1]
549 ? S< 0:00 [xfs-data/sd1]
551 ? S< 0:00 [xfs-conw/sd1]
555 ? S< 0:00 [xfs-cil/sd1]
558 ? S 0:00 [xfsaild/sd1]
582 ? S<sl 0:00 /sbin/auditd -n
607 ? Ssl 0:00 /usr/sbin/NetworkManager --no-daemon
609 ? Ssl 0:00 /usr/sbin/rsyslogd -n
610 ? Ssl 0:00 /bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
620 ? Ss 0:00 /usr/lib/systemd/systemd-logind
623 ? Ss 0:00 /usr/sbin/crond -n
630 ? Ss 0:00 login -- root
633 ? Ss 0:00 /usr/sbin/wpa_supplicant -u -f /var/log/wpa_supplicant.log -c /etc/wpa_supplicant/wpa_supplicant.conf -u -f /var/log/wpa_supplicant.log -P /var/run/wpa_supplicant.pid
634 ? Ssl 0:00 /usr/lib/polkit-1/polkitd --no-debug
--Lebih--
```

Dapat dilihat bahwa kolom bagian paling kiri berisi informasi PID (Process ID). Untuk proses induk yang menjalankan proses `ps -axl | more` sebagai berikut. Bisa kita lihat dimana parent process adalah process dengan PID 2376, yang ditandai dengan `./system` yang merupakan perintah yang sebelumnya digunakan untuk mencompile program.

3. Pada kode program berikut, proses baru akan dibentuk dengan menggunakan fungsi `exec()`.

```

centos_sisop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 0 385 2 0 -20 0 0 rescue S< ? 0:00 [xfs_mru_cache]
1 0 386 2 0 -20 0 0 rescue S< ? 0:00 [xfs-buf/dm-0]
1 0 387 2 0 -20 0 0 rescue S< ? 0:00 [xfs-data/dm-0]
1 0 388 2 0 -20 0 0 rescue S< ? 0:00 [xfs-conv/dm-0]
1 0 389 2 0 -20 0 0 rescue S< ? 0:00 [xfs-cil/dm-0]
1 0 390 2 20 0 0 0 xfsail S ? 0:00 [xfsail/dm-0]
4 0 466 1 20 0 35032 2272 ep_pol Ss ? 0:00 /usr/lib/systemd/systemd-journald
4 0 473 1 20 0 195068 4684 poll_s Ss ? 0:00 /usr/sbin/lsmd -f
4 0 486 1 20 0 43440 2264 ep_pol Ss ? 0:00 /usr/lib/systemd/systemd-udev
1 0 547 2 0 -20 0 0 rescue S< ? 0:00 [xfs-buf/sda1]
1 0 549 2 0 -20 0 0 rescue S< ? 0:00 [xfs-data/sda1]
1 0 551 2 0 -20 0 0 rescue S< ? 0:00 [xfs-conv/sda1]
1 0 555 2 0 -20 0 0 rescue S< ? 0:00 [xfs-cil/sda1]
1 0 558 2 20 0 0 0 xfsail S ? 0:00 [xfsail/sda1]
4 0 582 1 16 -4 51244 1612 ep_pol S<sl ? 0:00 /sbin/auditd -n
4 0 607 1 20 0 511004 8104 poll_s Ssl ? 0:00 /usr/sbin/NetworkManager --no-d
4 0 609 1 20 0 283508 2940 poll_s Ssl ? 0:00 /usr/sbin/rsyslogd -n
4 81 610 1 20 0 35008 1860 ep_pol Ssl ? 0:00 /bin/dbus-daemon --system --add
4 0 620 1 20 0 26452 1788 ep_pol Ss ? 0:00 /usr/lib/systemd/systemd-logind
4 0 623 1 20 0 126496 1684 hrtime Ss ? 0:00 /usr/sbin/cron -n
4 0 630 1 20 0 90268 2416 wait Ss ? 0:00 login -- root
4 0 633 1 20 0 53112 2676 poll_s Ss ? 0:00 /usr/sbin/wpa_supplicant -u -f
4 997 634 1 20 0 522760 12840 poll_s Ssl ? 0:00 /usr/lib/polkit-1/polkitd --no-
4 0 641 607 20 0 110568 15812 poll_s S ? 0:00 /sbin/dhclient -d -q -sf /usr/l
4 0 831 1 20 0 82608 3604 poll_s Ss ? 0:00 /usr/sbin/sshd -D
4 0 832 1 20 0 553232 16352 poll_s Ssl ? 0:00 /usr/bin/python -Es /usr/sbin/t
5 0 1333 1 20 0 91188 2112 ep_pol Ss ? 0:00 /usr/libexec/postfix/master -w
4 89 1351 1333 20 0 91292 3920 ep_pol S ? 0:00 pickup -l -t unix -u
4 89 1352 1333 20 0 91360 3948 ep_pol S ? 0:00 qmgr -l -t unix -u
4 0 1973 630 20 0 115544 2052 wait Ss tty1 0:00 -bash
1 0 1998 2 0 -20 0 0 worker S< ? 0:00 [kworker/0:0H]
1 0 2001 2 0 -20 0 0 worker S< ? 0:00 [kworker/0:1H]
1 0 2010 2 20 0 0 0 worker S ? 0:00 [kworker/0:0]
1 0 2021 2 20 0 0 0 worker S ? 0:00 [kworker/0:2]
1 0 2022 2 0 -20 0 0 worker S< ? 0:00 [kworker/0:2H]
0 0 2029 1973 20 0 137540 1296 - R+ tty1 0:00 ps -axl
[root@localhost ~]# _

```

Jalankan kode program pada nomor 3. Amati hasilnya dan bandingkan hasilnya dengan program pada nomor 2. Temukan perbedaannya dan jelaskan mengapa?

Jawab:

Dari hasil compile process ID yang menampilkan seluruh daftar ditandai dengan "ps-ax" adalah PID 2427.

Terdapat hal hal yang berbeda ketika daftar proses dengan menggunakan file `exec.c` dengan file `system.c`. Perbedaan pertama adalah informasi yang ditampilkan pada file `system.c` yang menggunakan function `system()` menampilkan PID, TTY, STAT, TIME, COMMAND, sedangkan pada file `exec.c` tidak tampak karena langsung menunjukkan akhir dari eksekusi. Perbedaan selanjutnya yaitu pada saat file `system.c` di compile file tersebut tidak langsung menunjukkan hasil keseluruhan compile melainkan kita harus menekan tombol more untuk melihat compile selanjutnya hingga akhir, sedangkan file `exec.c` ketika di compile make akan langsung mengcompile seluruh/menampilkan secara keseluruhan. Perbedaan lainnya yaitu pada file `system.c` kita dapat melihat command `./system` yang dijalankan beserta dengan

argument pada system yang kita berikan children process dari ./system tersebut dimana kita dapat melihat proses dengan command `sh -c ps -ax | more`, kemudian `ps -ax`, dan `more` memiliki process IDnya tersendiri. Sedangkan pada file `execlp.c` kita hanya melihat satu proses saja yang menandakan munculnya daftar proses ini. Proses tersebut adalah PID 2427 ditandai dengan command `ps -ax` yang menampilkan keseluruhan daftar secara langsung.

4. Tuliskan kode program berikut

```
printf("Main Process ID (PID) = %d Parent Process ID (PPID) = %d\n", getpid(), getppid());

pid = fork();

if(pid == 0){
    printf("This is the child process\n");
    printf("PID = %d\n", pid);
    printf("Child's PID %d parent PID %d\n", getpid(), getppid());
}else{
    printf("This is the parent process\n");
    printf("PID = %d\n", pid);
    printf("Parent's PID %d parent PID %d\n", getpid(), getppid());
}

return 0;
}
```

[Wrote 22 lines]

```
[root@localhost ~]# cc fork_3.c -o fork_3
[root@localhost ~]# ./fork_3
Main Process ID (PID) = 2059 Parent Process ID (PPID) = 1973
This is the parent process
PID = 2060
Parent's PID 2059 parent PID 1973
[root@localhost ~]# This is the child process
PID = 0
Child's PID 2060 parent PID 1
./fork_3
```

Eksekusi kode program pada nomor 4, amati hasilnya, kemudian jelaskan hasil dari program tersebut. Proses manakah yang dijalankan pertama kali, apakah proses induk atau proses anak? Mengapa?

Jawab:

Dari hasil compiler yang pertama kali dieksekusi oleh kode program adalah parent process, karena child process akan menduplikasi ruang memori dari parent process.

5. Kode program

```
        n = 3;
        break;
    }
    for(; n>0; n--){
        puts(message);
        sleep(1);
    }
    exit(0);
}
```

[Wrote 34 lines]

```
[root@localhost ~]# cc child_parent.c -o child_parent
[root@localhost ~]# ./child_parent
Fork program starting
This is the parent
This is the child
This is the parent
This is the child
This is the parent
This is the child
This is the child
[root@localhost ~]# This is the child
```

Program di atas akan menjalankan dua proses yaitu proses induk dan proses anak. Proses anak akan dijalankan sebanyak 5 kali, dan proses induk akan dijalankan sebanyak 3 kali. Jalankan program di atas dan amati hasil yang terjadi?

Jawab:

Parent process yang terlebih dahulu muncul dan diikuti dengan child process. Terdapat loop/perulangan yang dibuat untuk melakukan decremental untuk variable n yang di assign ke masing masing parent process dan child process. Ketika dijalankan terlihat parent process ditampilkan ke layar sebanyak 3 kali sedangkan child process sebanyak 5 kali. Namun untuk child process ketika ditampilkan untuk ke 5 kali, tulisan “This is the child” muncul pada bagian untuk user mengetik command pada terminal.

6. Kode Program

```
        printf("Child exited with code %d\n", WEXITSTATUS(stat_val));
    }else{
        printf("Child terminated abnormally\n");
    }
}
exit(exit_code);
}
```

[Wrote 53 lines]

```
[root@localhost ~]# cc child_parent_2.c -o child_parent_2
[root@localhost ~]# ./child_parent_2
Fork program starting
This is the parent
This is the child
This is the parent
This is the child
This is the parent
This is the child
This is the child
This is the child
Child has finished with PID =2160
Child exited with code 37
[root@localhost ~]#
```

7. Jelaskan apa efek dari menggunakan fungsi wait() dari program di atas? Bandingkan hasil dari program pada nomor 5 dengan hasil program nomor 6, apa yang dapat anda simpulkan dari kedua program tersebut?

Jawab:

sebagaimana yang telah dijelaskan pada jawaban no 5, hasil decrement terakhir dari variable n untuk mengetikkan command pada terminal. Namun pada nomor 6 function wait() yang diberikan membuat parent process untuk menunggu child process selesai dieksekusi untuk kemudian parent process melakukan terminasi sehingga dapat dilihat exit code = 0.

8. Jelaskan mengapa parent process harus memanggil system call wait() dan apa yang terjadi apabila system call wait() tidak dipanggil?

Jawab:

system call wait() dipanggil untuk membuat parent process menunggu child process menyelesaikan eksekusi processnya hingga selesai. Stelah child process berhenti atau diterminasi maka giliran parent process berikutnya untuk berhenti beroperasi, jika system call wait() tidak dipanggil maka dapat terjadi yang namanya orphan process, parent pocess berhenti tanpa menunggu child process untuk berhenti terlebih dahulu atau dengan

kata lain dimana suatu child process masih sedang berjalan atau dieksekusi namun tidak lagi memiliki suatu parent process yang manaunginya.

9. Kode Program

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(int arg, char *argv[]){
    pid_t pid;

    pid = fork();

    if (pid > 0){
        sleep(60);
    }else{
        exit(0);
    }
    return 0;
}
```

[Wrote 16 lines]

```
[root@localhost ~]# cc zombie.c -o zombie
[root@localhost ~]# ./zombie
```

10. Eksekusi program di atas. Observasi hasil dari eksekusi program, Anda dapat menggunakan perintah `ps -al` pada terminal yang lain untuk melihat proses yang sedang berjalan. Apakah hasil dari perintah `ps -al`, jelaskan mengapa terjadi hal demikian?

Jawab:

```
File Machine View Input Devices Help
int main(int arg, char *argv[]){
    pid_t pid;

    pid = fork();

    if (pid > 0){
        sleep(60);
    }else{
        exit(0);
    }
    return 0;
}

[ Wrote 16 lines ]

[root@localhost ~]# cc zombie.c -o zombie
[root@localhost ~]# ./zombie
^C
[root@localhost ~]# ps -al
  S  UID    PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
  R      0  2000   1974  0  80   0 - 34377 -        tty1      00:00:00 ps
[root@localhost ~]#
```

melalui hasil compile terlihat bahwa yang dieksekusi pada terminal lainnya yang menjalankan zombie mengalami hal yang disebut zombie process sebagaimana dimaksudkan pada kode di soal no 9 dengan suatu pernyataan `if > pid = 0` maka parent process akan melakukan sleep selama 60 s walaupun sebenarnya pengekseskuan dari parent process sudah selesai dilakukan dan process masih terlihat ketika dipanggil `ps -al`

C. Tugas Pemrograman

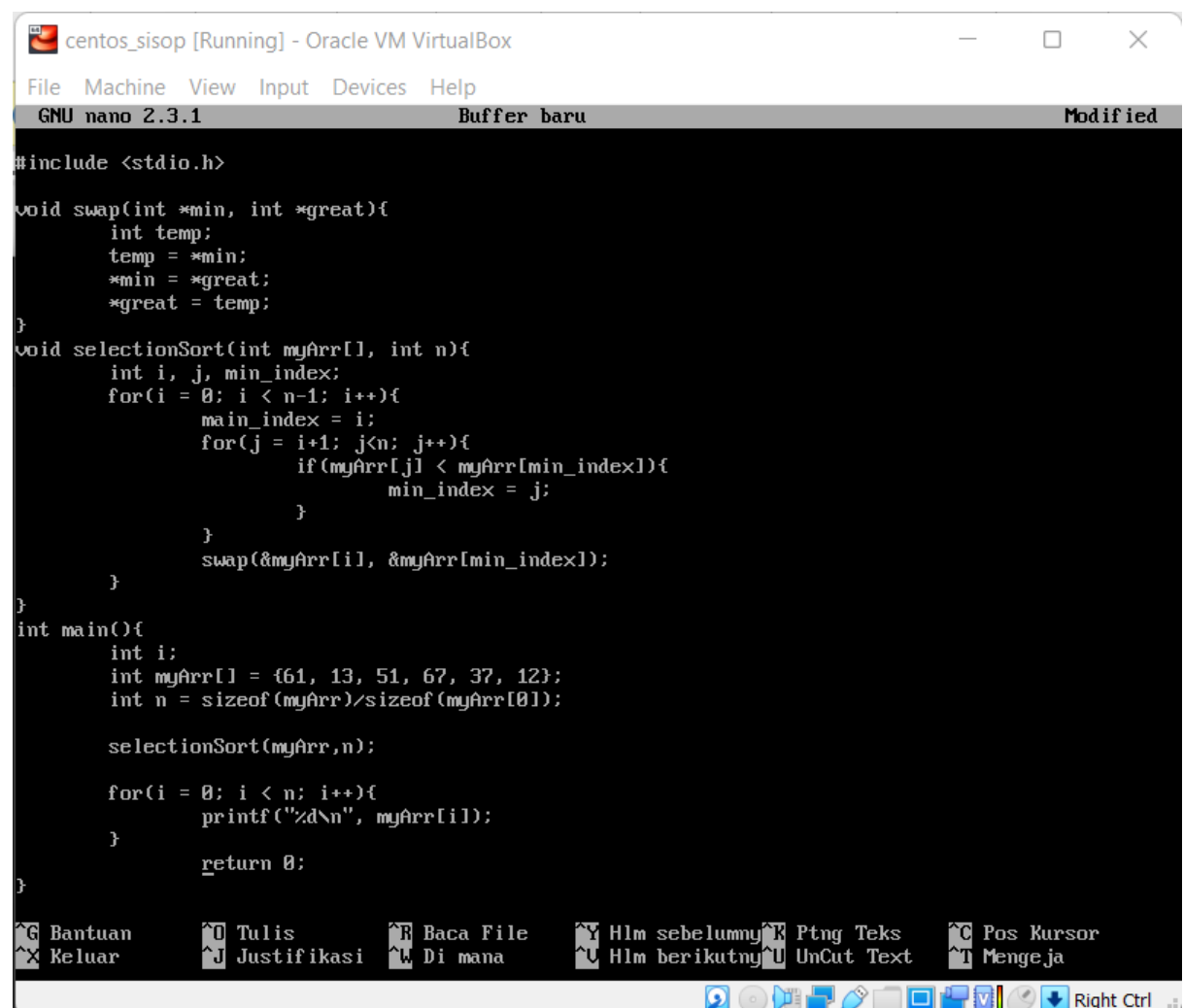
1. Buatlah sebuah program untuk mengurutkan data menggunakan algoritma Selection Sort. Namakan program sebagai `child_process_ssort.c`. Kompilasi program untuk menghasilkan berkas tereksekusi (executable file): `child_process_ssort.exe`. Proses ini akan anda gunakan sebagai proses anak.

Jawab:

1. Buatlah sebuah program untuk mengurutkan data menggunakan algoritma Selection Sort. Namakan program sebagai `child_process_ssort.c`. Kompilasi program untuk menghasilkan berkas tereksekusi (executable file): `child_process_ssort.exe`. Proses ini akan anda gunakan sebagai proses anak.

Jawab:

Input



```
#include <stdio.h>

void swap(int *min, int *great){
    int temp;
    temp = *min;
    *min = *great;
    *great = temp;
}

void selectionSort(int myArr[], int n){
    int i, j, min_index;
    for(i = 0; i < n-1; i++){
        min_index = i;
        for(j = i+1; j<n; j++){
            if(myArr[j] < myArr[min_index]){
                min_index = j;
            }
        }
        swap(&myArr[i], &myArr[min_index]);
    }
}

int main(){
    int i;
    int myArr[] = {61, 13, 51, 67, 37, 12};
    int n = sizeof(myArr)/sizeof(myArr[0]);

    selectionSort(myArr,n);

    for(i = 0; i < n; i++){
        printf("%d\n", myArr[i]);
    }

    return 0;
}
```

Output

```
[root@localhost ~]# gcc child_process_ssort.c -o out
[root@localhost ~]# ./out
12
13
37
51
61
67
[root@localhost ~]#
```

2. Buatlah proses induk yang bertujuan untuk membuat dan mengeksekusi proses anak pada no. 1 di atas. Beri nama program anda sebagai parent_process.c. Kompilasi dan jalankan program.

Jawab:

```
#include <sys/wait.h>

int main(){
    pid_t pid;
    pid = fork();

    if(pid < 0){
        fprintf(stderr, "Fork failed\n");
        return 1;
    }else if(pid == 0){
        printf("Child is being executed\n");
        execlp("./child_process_ssort", "child_process_ssort", NULL);
    }else{
        wait(NULL);
        printf("Child execution complete\n");
    }
    printf("End of parent program\n");
    return 0;
}
```

[Wrote 23 lines]

```
[root@localhost ~]# cc parent_process.c -o out
[root@localhost ~]# ./out
Child is being executed
End of parent program
Child execution complete
End of parent program
[root@localhost ~]# _
```