Institut Teknologi Del

MODUL PRAKTIKUM #1
1031202/1041202 - Sistem Operasi

MANAJEMEN PROSES diLINUX

Minggu	**:**	3
Setoran	**:**	Jawaban disetor ke ecourse untuk soal bagian A. Teori (nomor 1 sd. 4) B. Pemrograman (nomor 1, 2, 3, 4, 5, 7, 8, 10) Kode Program diunggah ke ecourse untuk soal bagian C. Tugas Pemrograman (child_process_ssort.c dan parent_process.c)
Batas Waktu Setoran	**:**	**H+3 pukul 17.00 WIB**
Tujuan	**:**	Mahasiswa mampu menulis program untuk pengelolaan proses pada Linux yang mencakup pembuatan proses dan terminasi proses.

Petunjuk Praktikum:

1. Tugas ini dikerjakan secara individu.
2. Mencontoh pekerjaan dari orang lain akan dianggap plagiarisme dan anda akan ditindak sesuai dengan sanksi akademik yang berlaku di IT Del atau sesuai dengan kebijakan saya dengan memberikan nilai 0.
3. Jawaban diketik menggunakan word processor kemudian dikonversi ke file berekstensi .pdf
4. Penamaan file HARUS sesuai dengan format No Kelompok_Tugas-X_NamaTugas.pdf (contoh: 01_Tugas-2_Struktur_Sistem_Operasi.pdf).
5. Keterlambatan menyerahkan laporan tidak ditolerir dengan alasan apapun. Oleh karena itu, laporan harus dikumpul tepat waktu.
6. Gunakan Sistem Operasi Linux boleh menggunakan Distro apapun namun disarankan untuk mempermudah praktikum gunakan Ubuntu.

Referensi:

1. A. Silberschatz, P.B. Galvin, and G. Gagne, Operating System Concepts, 9th edition, Chapter 1 and 2, John Wiley & Sons, Inc., 2013.
2. M. Neil, S. Richard, Beginning Linux Programming, 4th edition, Wiley, 2008.

A. Teori

1. Pertanyaan berikut terkait dengan konsep-konsep dasar Proses
    a. Definisikan program.
    b. Definisikan proses.
    c. Definisikan *Zombie Process*
    d. Definisikan *Orphan Process*
    e. Jelaskanlah *Process Control Block* (PCB)

2. Tulislah program berikut.

```c
/*fork_sederhana*/

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

3. Dari kode program di atas jelaskan apa yang dimaksud dengan:
    a. pid_t
    b. fork()
    c. execlp()
    d. wait()

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

B. Pemrograman

1. Tuliskan kode program berikut.

```c
/*cetakpid.c*/

#include <unistd.h>
#include <stdio.h>

int main (){

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

}
```

Eksekusi program di atas dengan cara sebagai berikut:

```
eka@eka-VirtualBox:~/Documents/Prak_OS_Process$ gcc -o cetakpid cetakpid.c
```

a. Jelaskan perbedaan antara **getpid()** dengan **getppid()**.
b. Jelaskan mengapa setiap kali program di atas dieksekusi, maka akan menampilkan process ID yang berbeda. Jelaskan mengapa?

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

```c
/*system.c*/
#include <stdlib.h>
#include <stdio.h>

int main (){

    printf("Running ps with system \n");
    system("ps -ax | more");
    printf("Done\n");

    return 0;

}
```

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

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

```c
/*execlp.c*/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int main (){

    printf("Running ps with execlp \n");
    execlp("ps","ps","-axl",NULL);
    printf("Done\n");

    exit (0);
```

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

4. Tuliskan kode program berikut.

```c
/*fork_3.c*/

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

    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;
}
```

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?

5. Tuliskan kode program berikut.

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

int main (){
    pid_t pid;
    char *message;
    int n;

    printf("Fork program starting\n");
    pid = fork();

    switch(pid){
        case -1:
        perror("Fork failed");
        exit(0);

        case 0:
        message = "This is the child";
        n = 5;
        break;

        default:
        message = "This is the parent";
        n = 3;
        break;
    }

    for(; n>0; n--){
        puts(message);
        sleep(1);
    }

    exit (0);
```

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?

6. Sesuai dengan siklus pembentukan proses anak dengan menggunakan fork() hingga proses anak diterminasi seperti yang ditunjukkan pada slide presentasi halaman 3.21, maka proses induk harus menunggu seluruh proses anak selesai dengan memanggil *system call* **wait()**. Dengan memodifikasi program nomor 5, tuliskan program di bawah (halaman selanjutnya). Pada program tersebut anda akan menerapkan *system call* **wait()** dengan menggunakan library **sys/wait.h**.

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

int main (){
    pid_t pid;
    char *message;
    int n;
    int exit_code;

    printf("Fork program starting\n");
    pid = fork();

    switch(pid){
        case -1:
        perror("Fork failed");
        exit(0);

        case 0:
        message = "This is the child";
        n = 5;
        exit_code = 37;
        break;

        default:
        message = "This is the parent";
        n = 3;
        exit_code = 0;
        break;
    }

    for(; n>0; n--){
        puts(message);
        sleep(1);
    }

    if(pid!=0){
        int stat_val;
        pid_t child_pid;

        child_pid = wait(&stat_val);

        printf("Child has finished with PID = %d\n", child_pid);
        if(WIFEXITED(stat_val)){
            printf("Child exited with code %d\n", WEXITSTATUS(stat_val)
        }else{
            printf("Child terminated abnormally\n");
        }
    }

    exit(exit_code);
```

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?

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

9. Tuliskan kode program berikut.

```c
/*zombie.c*/

#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;
}
```

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?

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.

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.