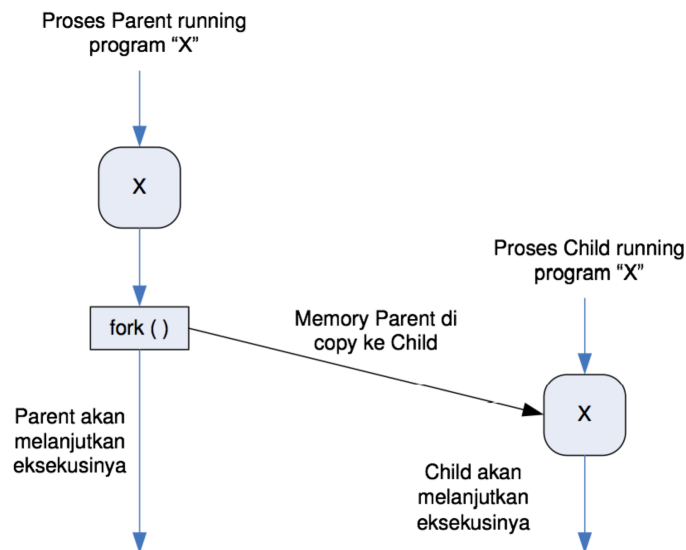


PROSES DAN PEMBUATAN PROSES

Proses adalah sebuah program yang dalam keadaan sedang dieksekusi. Saat sebuah program yang disimpan di dalam disk dieksekusi maka program tersebut akan running. Program (bersifat pasif) tersebut akan berubah menjadi proses (bersifat aktif).

Saat proses terbentuk maka dia akan memiliki **alamat sendiri** dan secara otomatis akan terdapat **satu buah thread**. Saat program dieksekusi, mungkin saja terdapat banyak proses yang terbentuk dimana masing-masing proses tersebut **memiliki alamat dan nomor identitas sendiri (PID) yang unik** (yang dimaksud dengan unik adalah tidak ada proses yang memiliki PID yang sama saat dieksekusi) dan setiap proses berkerja secara independen (tidak tergantung pada proses-proses lainnya).

Organisasi proses berbentuk seperti pohon proses. Setiap proses memiliki sebuah proses parent dan jika proses tersebut membuat proses baru maka proses yang baru tersebut dinamakan proses child. **Sebuah proses child yang terbentuk dengan system call fork() hampir identik dengan proses parentnya (yang identik antara lain: variable, code, dan file descriptors) tetapi memiliki PID (Process ID) yang berbeda**. Setelah proses baru (child) berhasil dibuat eksekusi dilanjutkan secara normal di masing-masing proses pada baris setelah pemanggilan system call fork(). Gambar dibawah ini menunjukkan bagaimana system call fork() bekerja.



Untuk pembuatan proses baru di sistem, GNU C menyediakan system call `fork()` yang disertakan file header `unistd.h`.

Dengan tipe data sbb : pid_t fork (void);

System call fork() mengembalikan suatu besaran bertipe pid_t yang dapat bernilai :

- ☐ Angka -1 menandakan pembuatan proses baru (child) gagal
- ☐ Angka 0 menandakan proses child
- ☐ Angka > 1 menandakan proses parent

Percobaan :

1. Dasar Programming Fork()

Buat program berikut ini menggunakan bahasa c untuk mengetahui dasar fork di linux.

```
#include <stdio.h>
#include <unistd.h> /* berisi pustaka fork( ) */
int main(void) {
    printf("Teknik Informatika\n"); /* string Teknik Informatika ditampilkan oleh parent */
    fork( ); /* buat proses child */
    printf("PENS\n"); /* string PENS ditampilkan oleh parent dan child */
    sleep(8);
}
```

Lakukan kompilasi menggunakan perintah sbb :

`$gcc nama_program.c -o nama_hasil`

Jalankan prrogram dengan menggunakan perintah sbb dan amati hasilnya:

`$/nama_hasil`

Sambil program masih jalan buka terminal baru dan jalankan perintah `ps tree -p`, amati hasilnya.

Amati program dan hasilnya serta Buat algoritma program diatas.

2. Buat program dibawah ini :

```
#include <stdio.h>
#include <unistd.h> /* pustaka untuk system call fork() */
#include <stdlib.h> /* pustaka untuk system call exit() */
int main() {
    pid_t sir;
    sir = fork();
    if(sir == -1){
        printf("Pembuatan Child gagal\n");
        exit(EXIT_FAILURE); }
    else if(sir == 0){
        printf("Child: Ini Proses Child\n");
        printf("Child: PID saya: %d, PID Parent saya: %d\n",getpid(),getppid());
        printf("Child: Selesai.....\n");
    } else {
        printf("Parent: Ini Proses Parent\n");
        printf("Parent: PID saya: %d, PID Parent saya: %d, PID Child saya: %d\n",getpid(),getppid(),sir);
        printf("Parent: Selesai.....\n");
    }
    sleep(25);
    return(0);
}
```

Jalankan dan amati hasilnya. Buatlah algoritama yang menunjukkan alur program diatas dan analisa hasilnya urutan output program kenapa bisa seperti itu ?

3. Proses parent dan chield, Buatlah program berikut :

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h> /* wait() */

int main(){
    pid_t pid;
    int rv;
    pid=fork();
    switch (pid){
        case -1:
            perror("fork");
            exit(EXIT_FAILURE);
        case 0:
            printf("CHILD: ini proses child! \n");
            sleep(5);
            printf("CHILD : Selesai....! \n");
            exit(EXIT_SUCCESS);
        default :
            printf("PARENT: INi Proses parent! \n");
            printf("PARENT: Sekarang menunggu child selesai.... \n");
            wait(&rv);
            printf("PARENT: Selesai....! \n");
    }
    sleep(10);
}
```

Jalankan dan amati hasilnya. Buatlah algoritama yang menunjukkan alur program diatas dan analisa hasilnya urutan output program kenapa bisa seperti itu ?

Lihatlah menggunakan pstree -p lihat hasil PID proses, jelaskan kenapa hasilnya seperti tersebut diatas

4. Selesaikan potongan program ini menjadi program dan bagaimana pstree hasil program ini serta jelaskan analisa anda.

```

void fork2 ()
{
    printf("L0\n");
    fork();
    printf("L1\n");
    fork();
    printf("Bye\n");
}

```

5. Selesaikan potongan program ini menjadi program dan bagaimana pstree hasil program ini serta jelaskan analisa anda.

```

void fork2 ()
{
    printf("L0\n");
    fork();
    printf("L1\n");
    fork();
    printf("Bye\n");
}

```

6. Selesaikan potongan program ini menjadi program dan bagaimana pstree hasil program ini serta jelaskan analisa anda.

```

void fork4()
{
    printf("L0\n");
    if (fork() != 0) {
        printf("L1\n");
        if (fork() != 0) {
            printf("L2\n");
            fork();
        }
    }
    printf("Bye\n");
}

```

7. Selesaikan potongan program ini menjadi program dan bagaimana pstree hasil program ini serta jelaskan analisa anda.

```

void fork5()
{
    printf("L0\n");
    if (fork() == 0) {
        printf("L1\n");
        if (fork() == 0) {
            printf("L2\n");
            fork();
        }
    }
    printf("Bye\n");
}

```