Fork()

- Eksekusi fork antara parent dan child *nondeterministic (tidak ada hubungan 1 dengan yang lain)* dan terjadi secara *concurrently(Bersama-sama)*

- parent &child identik tapi terpisah

```c
void fork1 () {
    int x = 1;

    if (fork() == 0) {
        printf("Child has x = %d\n", ++x);
    } else {
        printf("Parent has x = %d\n", --x);
    }
}
```

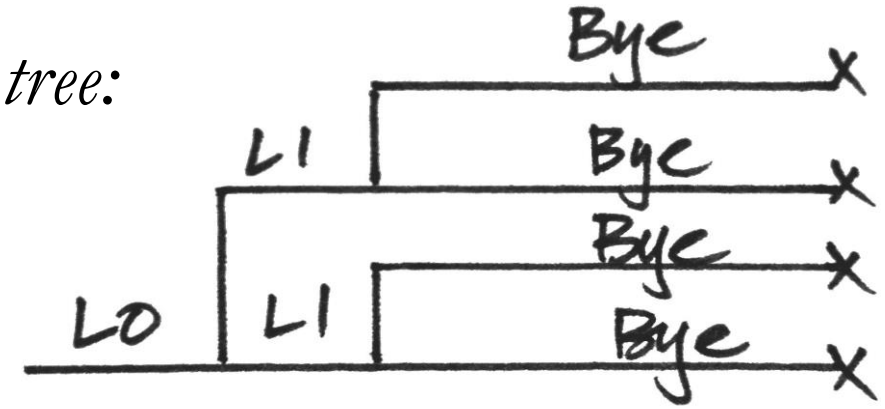
```
Parent has x = 0
Child has x = 2
```

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

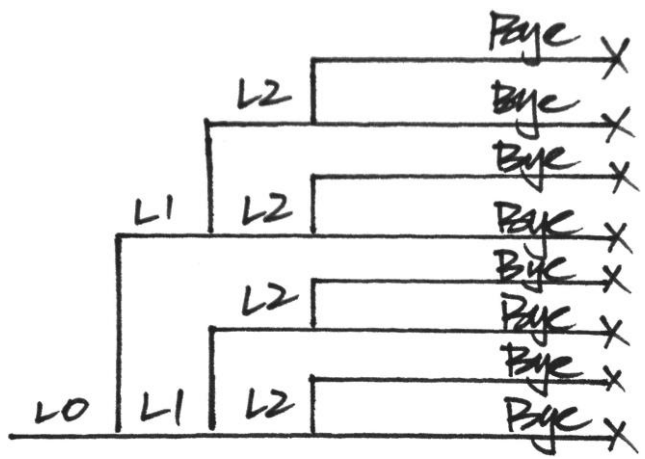
```
L0
L1
L1
Bye
Bye
Bye
Bye
```

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

process tree:



```c
void fork3() {
    printf("L0\n");
    fork();
    printf("L1\n");
    fork();
    printf("L2\n");
    fork();
    printf("Bye\n");
}
```



Tugas : Amati Hasilnya dan Buat peta tree processnya

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

Tugas : Amati Hasilnya dan Buat peta tree processnya

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