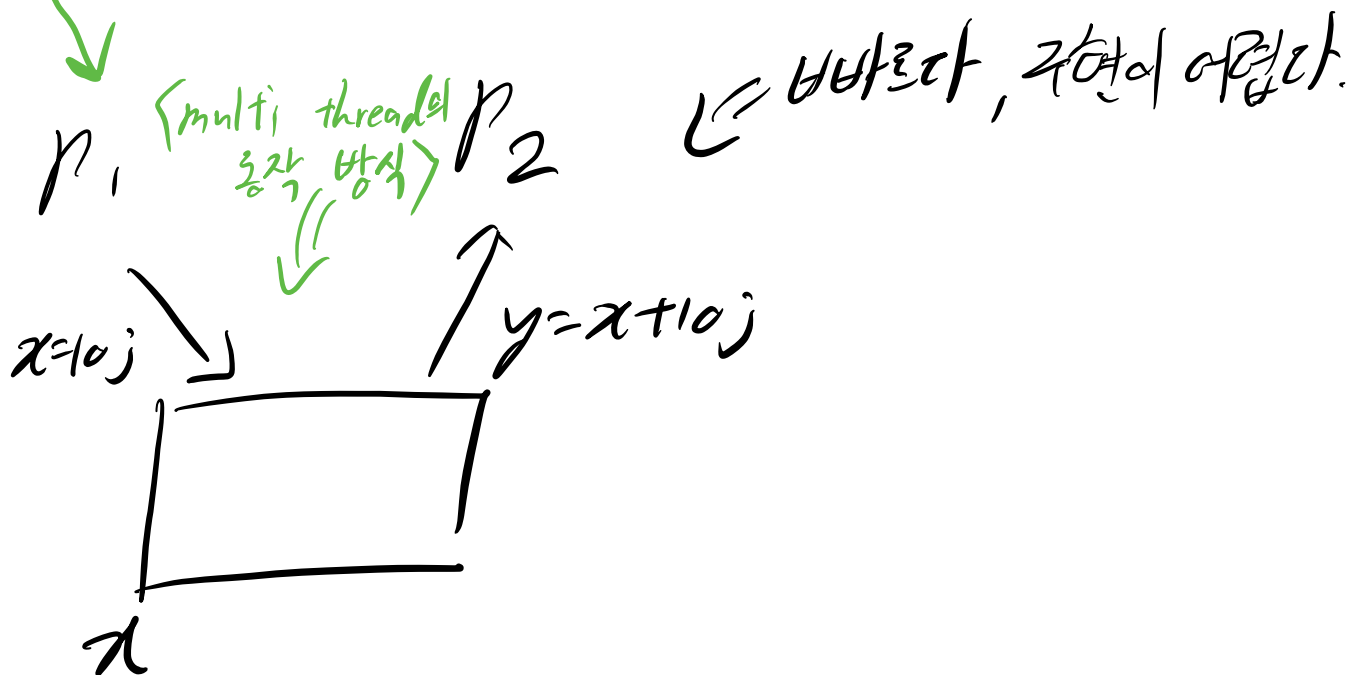
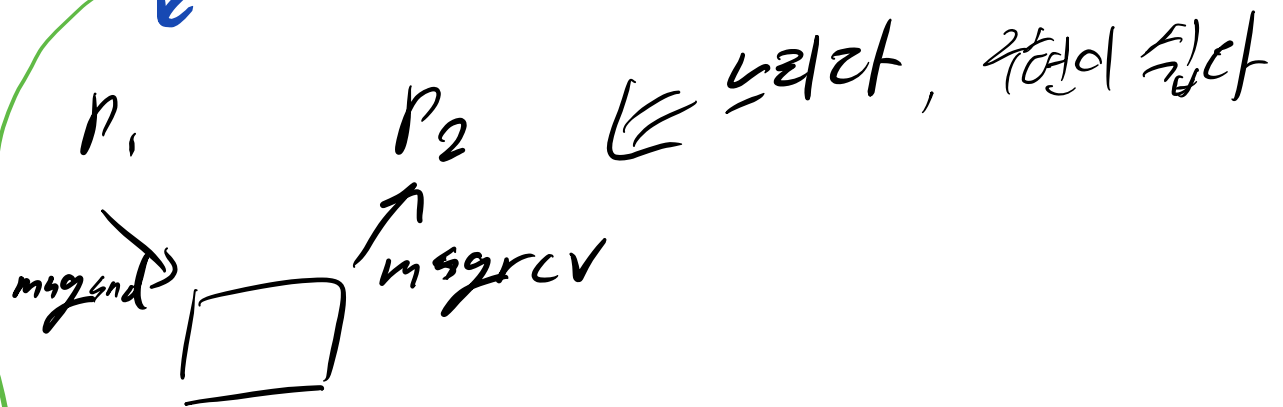


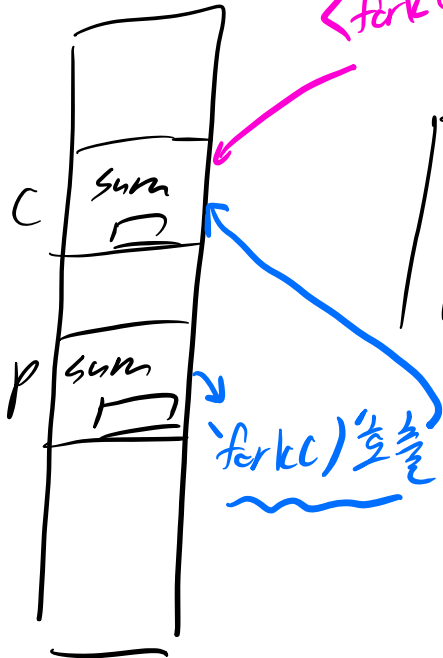
IPC (Inter Process Communication)

MP : Message Passing

SM : Shared Memory



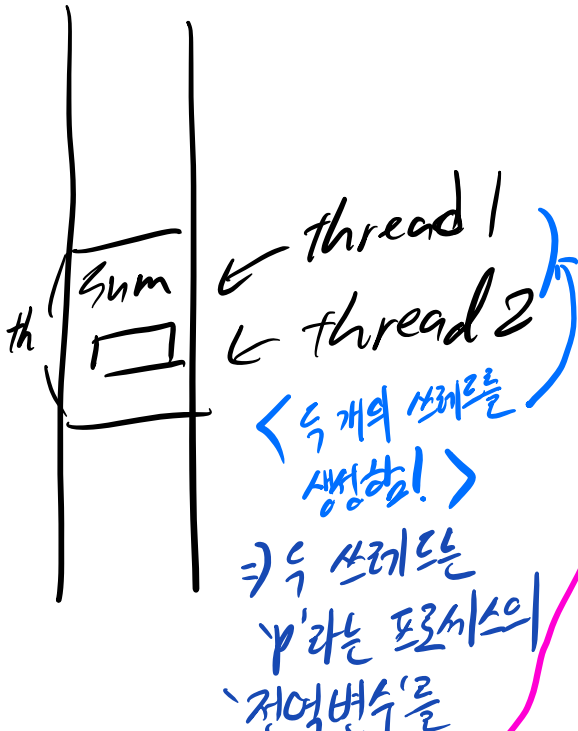
<ex1.c>



<fork() 호출을 통해 parent process와 독립적인 child process의 body가 메모리 상에 생성됨>

ch: 400,000
pa: 400,000

<th.c>



'thread 1' runs up to <sum = 20mil;
(=th1)

↓
stop ('thread 1'의 cpu 점유시간이 끝남.)

↓
'thread 2' runs up to <sum = 50mil>
(=th2)

↓
stop ('thread 2'의 cpu 점유시간이 끝남.)

↓
'thread 1' finishes. <sum = 20mil>

↓

공유한다!

'thread 2' finishes. $\langle \text{sum} = 80 \text{ mil} \rangle$

th 1 : 70 mil
th 2 : 80 mil

↑ \langle 하지만, 실제로 이런 결과가 출력되지 않는다.

$\langle \text{th.c} \rangle$ - th 1' runs up to $\langle \text{sum} = 20 \text{ mil} \rangle$



stops at (A) this case = 20 mil



'th 2' runs up to $\langle \text{sum} = 50 \text{ mil} \rangle$

↓
stops at (c)

↓
'th1' finishes. <sum = 40mil>

↓
'th2' finishes. <sum = 50mil>

↑ 이런 결과가 나온다! 왜 그런 것인가?

<th.c>

int sum = 0;

```
void *foo1(...) {
```

```
  for (i=0; i<20000; i=i+1) {
```

```
    for (j=0; j<20000; j=j+1) {
```

MOV eax, sum	(A)	}	sum = sum + 1 ;
INC eax	(B)		
MOV sum, eax	(C)		

MOV A, B :
B에 있는 것은
A에 저장하라

```
void *foo2(...) {
```

```
  for (i=0; i<20000; i=i+1) {
```

```
    for (j=0; j<20000; j=j+1) {
```

MOV eax, sum	(A)	}	sum = sum + 1 ;
INC eax	(B)		
MOV sum, eax	(C)		

INC A :
↑ A를 증가시켜라

(increase)

X 위 현상을 해결하기 위한 방법.

· semaphore (세마포어)를 통해, 'mutual exclusion'을
실시한다!

· mutual exclusion : 공유변수(전역변수)에 한 개의 쓰레드만
접근할 수 있도록 만드는 것.

⇒ 즉, 두 쓰레드가 동시에 특정 공유변수에
접근할 수 없다. 한 쓰레드가 특정 공유
변수에 대한 접근이 끝나면, 나머지
쓰레드가 해당 변수에 접근할 수
있다.