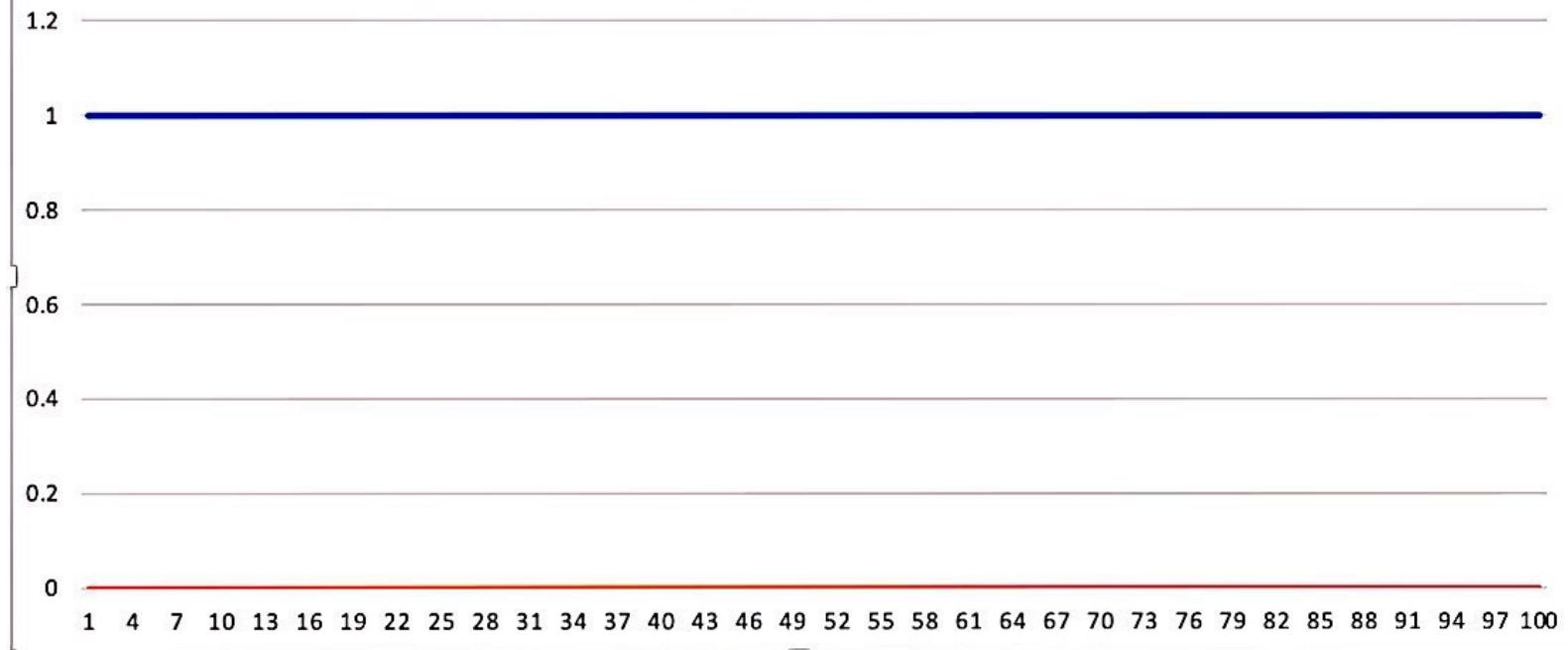


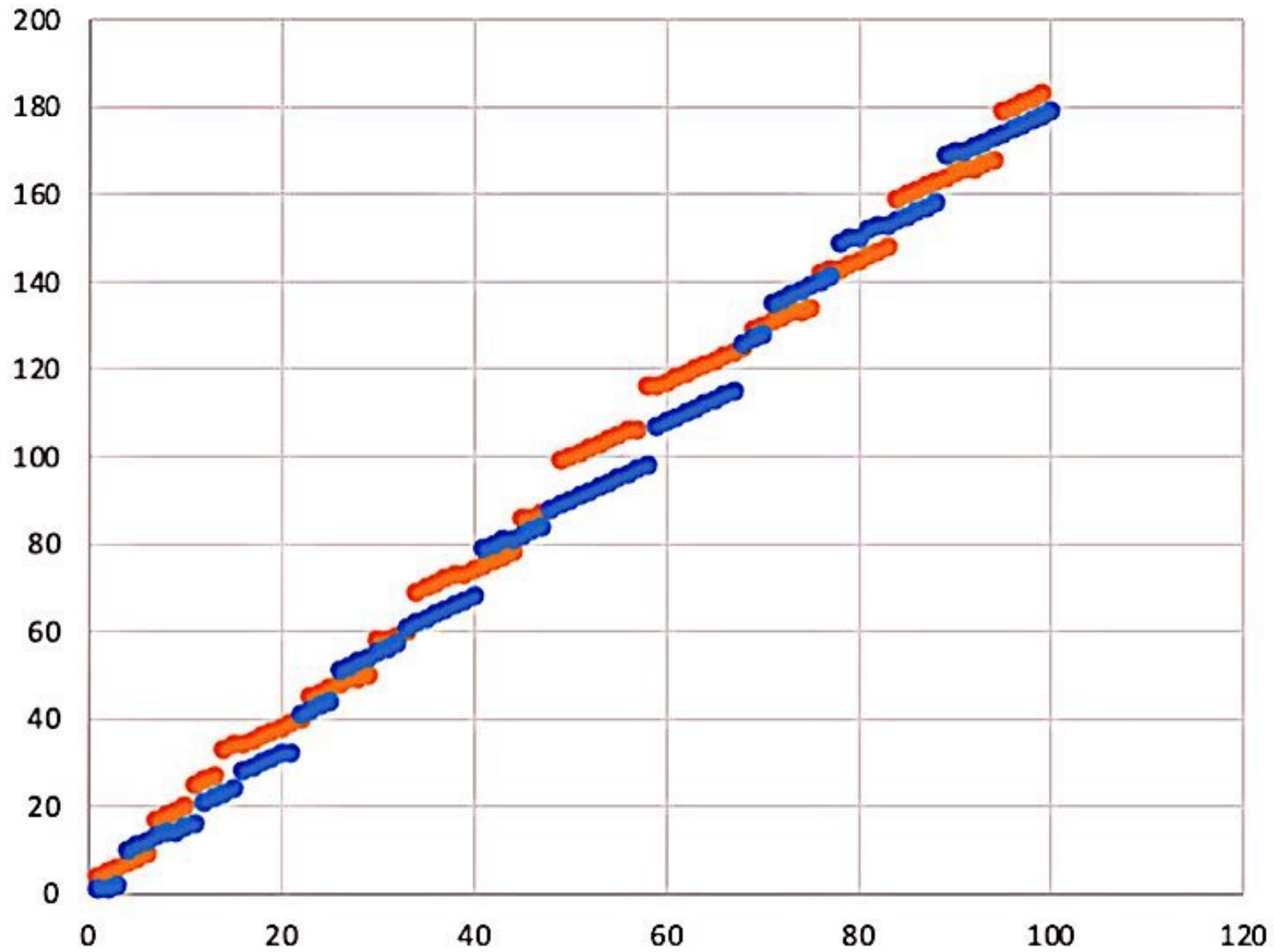
2018213354 지이.

1. 1) 슬랜 4-2. , 프로세스 2개 원 때.

실험 4-2 실행되고있는 프로세스



실험 4-2



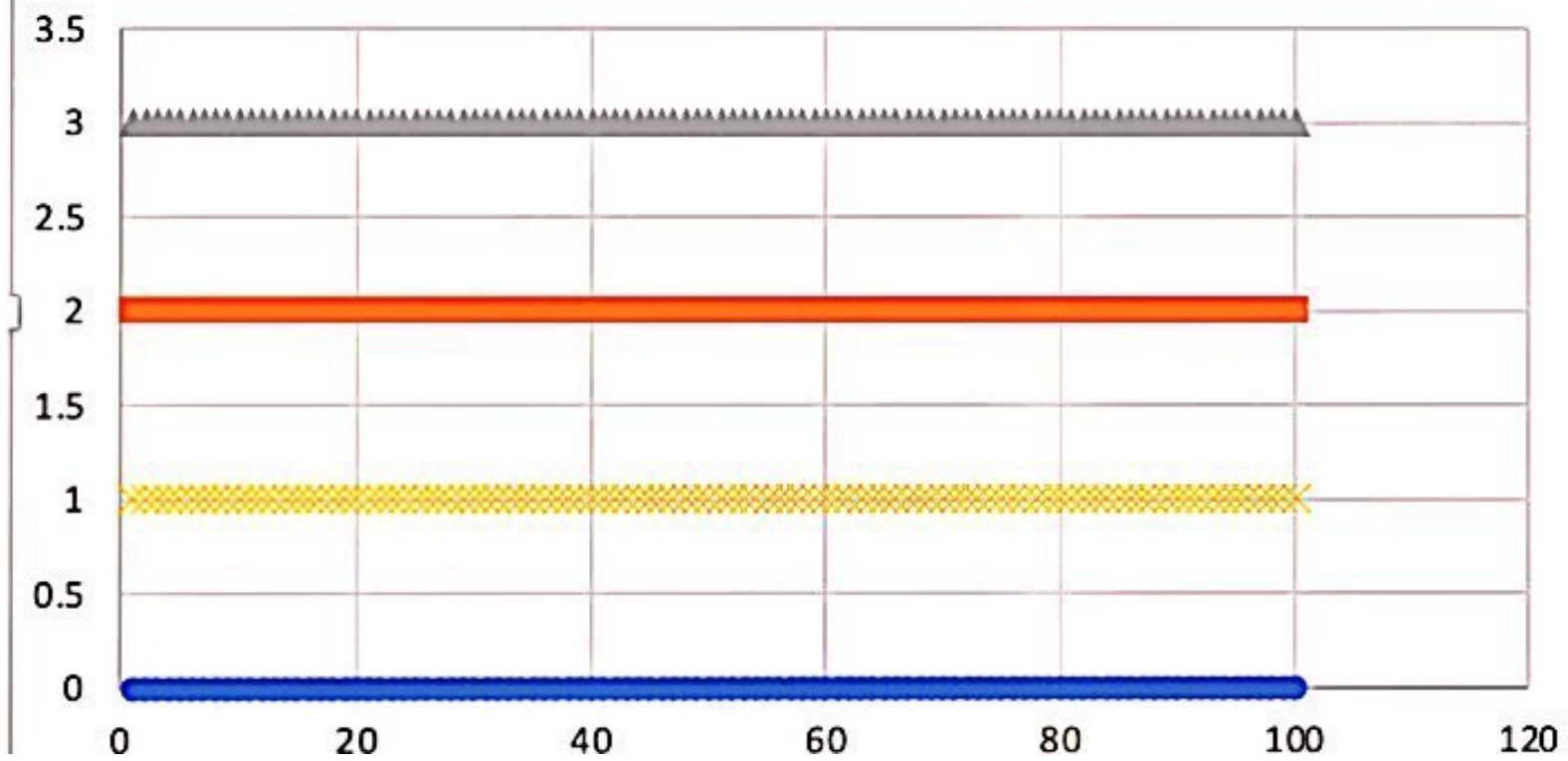
그럼에서는 잘 안보이지만 2개의 process가 서로 인라인이어서 논리 CPU사용한다.
(중시비 사용XX).

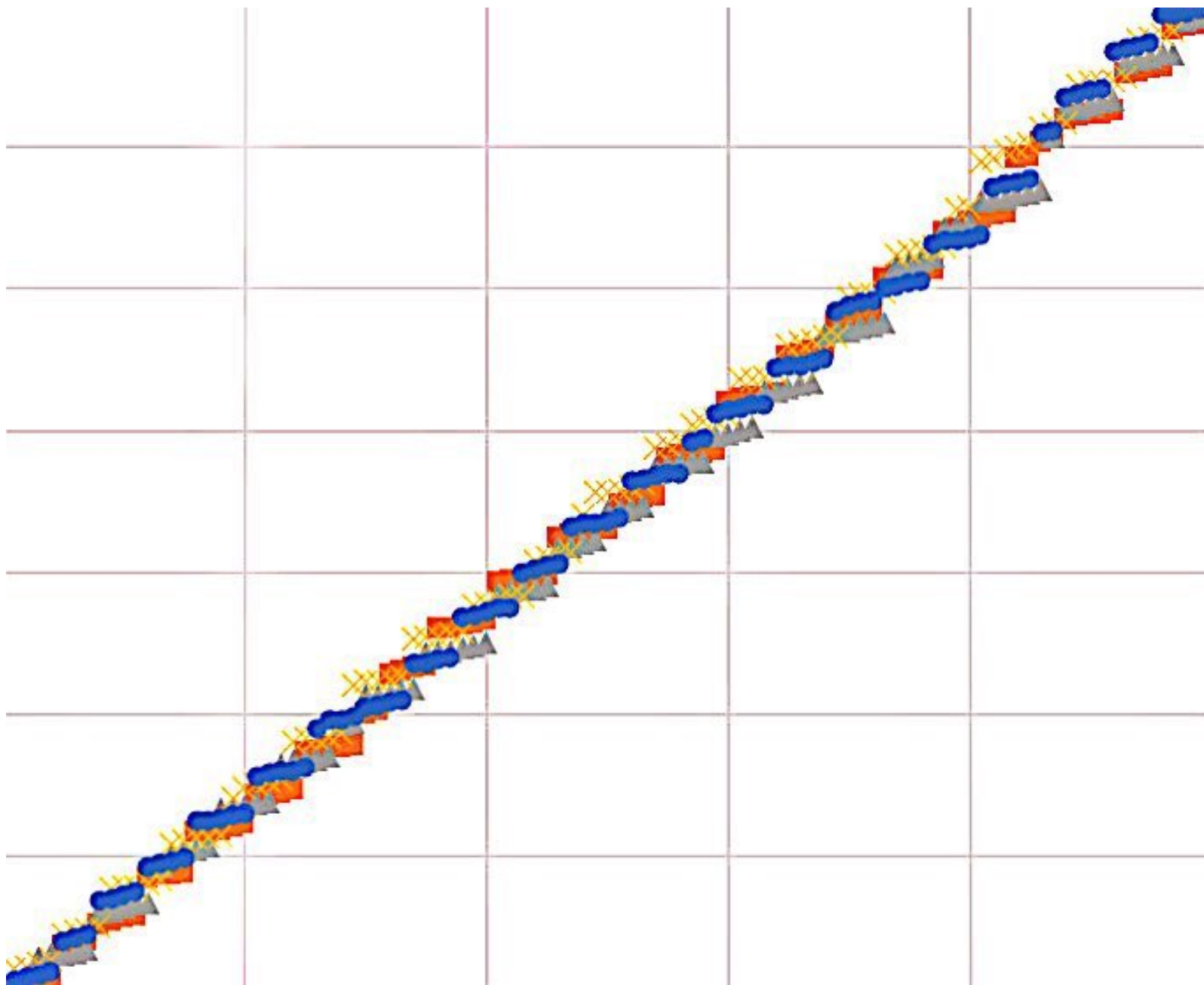
각 process는 논리CPU를 사용하고 있을 때만 처리가 진행됨은 안 수 있다.
단위 시간별 진행도는 process가 1개일 때만 비교해서 약 절반정도.

performance 4-1 내로 그이상이 된다.
처리연속까지

(2) 식 4-3, 프로세스 수에 의해서.

실험 4-3





CamScanner로 스캔하기

교비 항목이 비슷한 특성을 보임. 특히 시간당 진행률 process가 왜 원래보다
약 4배 정도 이었는지.

∴ 동시에 여러 process 실행을 허용하는 멀티태스킹을 지원하는 process는 소개했??
여러 process가 동시 순차적으로 프로그래밍 동작한다.
첫번째 process ~ 마지막 process까지 순차로 풀면 다시 첫번째 process 동작
(RR 방식)

2. throughput & latency 측정하기.

- throughput = 1초에 시간당 처리된 데이터 양을 측정. (process / 1초) (단위: process/1초)
- latency => 처리가 시작 ~ 종료까지 걸린 시간을 측정. (처리종료시간 - 처리시작시간)

(1) process 2개일 때

$$\text{throughput} \Rightarrow 2 \text{ process} / 10.2 [\text{s}]$$

$$= 10 \text{ process} / 1 [\text{s}].$$

$$\text{latency} = \sqrt[n]{\text{process}} \text{ process } 1 \text{ latency}$$

$$= 200 [\text{ms}].$$

(2) process 4개일 때

$$\text{throughput} \Rightarrow 4 \text{ process} / 0.4 [\text{s}]$$

$$= 10 \text{ process} / 1 [\text{s}]$$

$$\text{latency} \Rightarrow \text{process } 0 \sim \text{process } 3 \text{ latency.}$$

$$= 400 [\text{ms}].$$

3. multiprocessor.

```
judith@judith-VirtualBox:~$ grep -c processor /proc/cpuinfo  
1  
judith@judith-VirtualBox:~$
```

1. 확인해보니 이 시스템에 논리 CPU 개수는 4개이다.

Sched 프로그램은 여러 개의 논리 CPU가 인식되는 시스템에서만

Test 할 수 있는 것 같다. 따라서 이 부분은 다른 교재를 참고하여

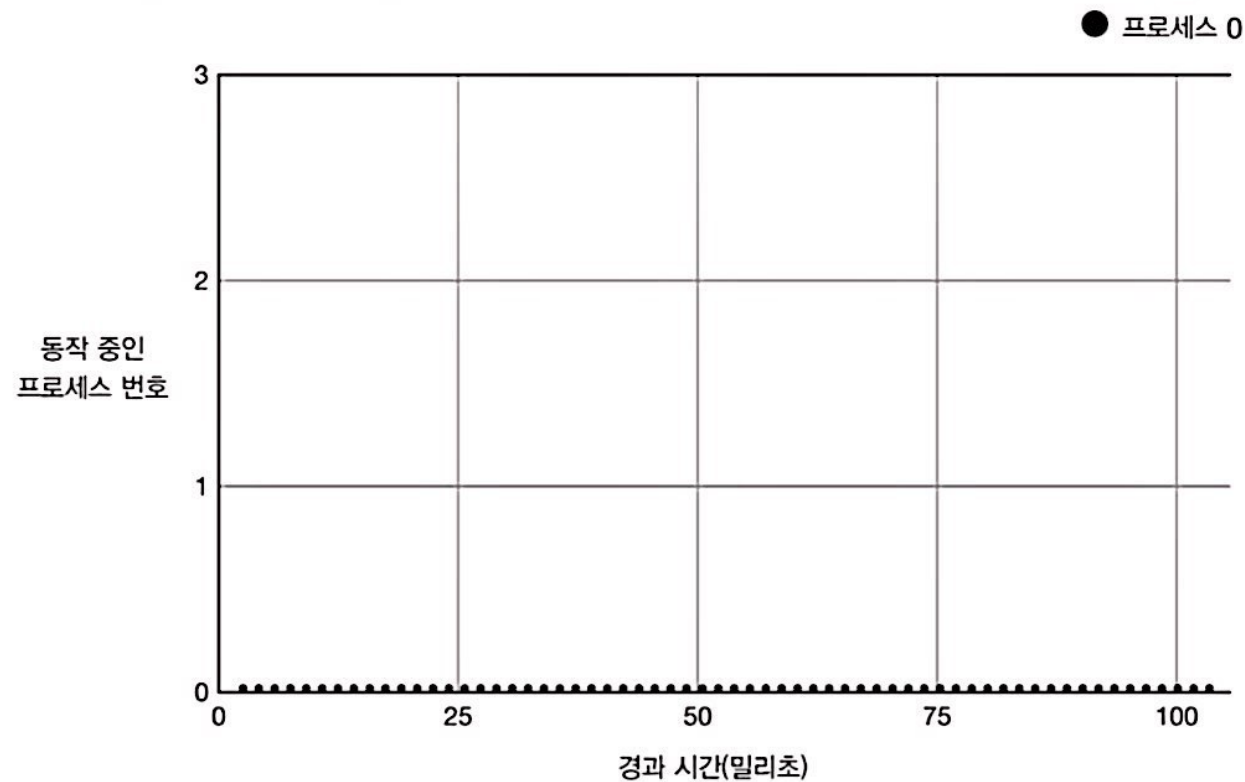
더 공부했다. (그 교재에서 내준 결과를 토대로 분석해보았다.)

프로세스 1개 : 실험 4-D

프로세스가 1개인 경우에는 [그림 4-25]와 같이 동작합니다.

그림 4-25 논리 CPU에 동작 중인 프로세스

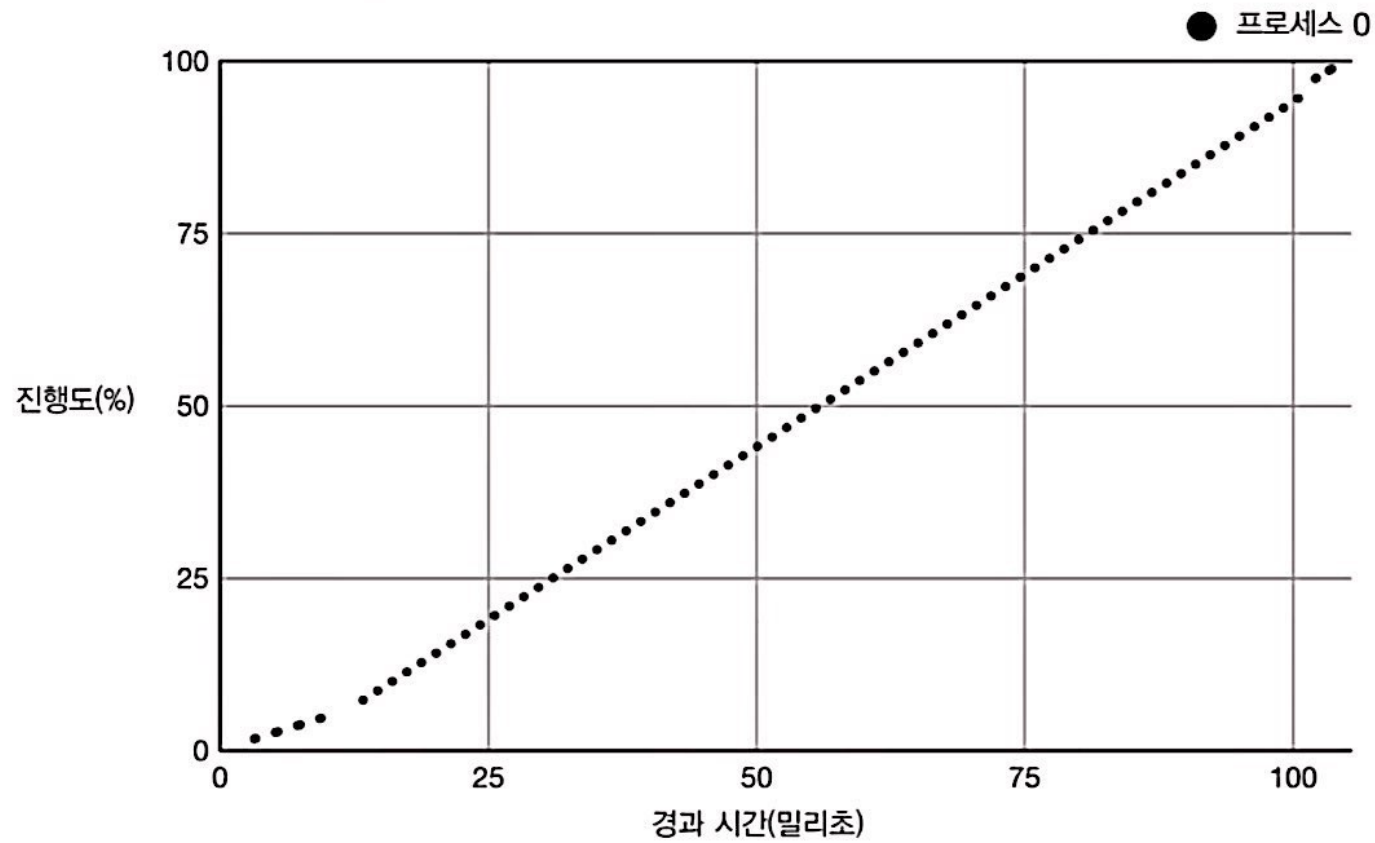
(실험 4-D, 그래프 ①)



프로세서 0이 안쪽아든 논리 CPU에서 계속 동작한다.
(다른 쪽의 논리 CPU는 idle 상태)

그림 4-26 프로세스 0의 진행도

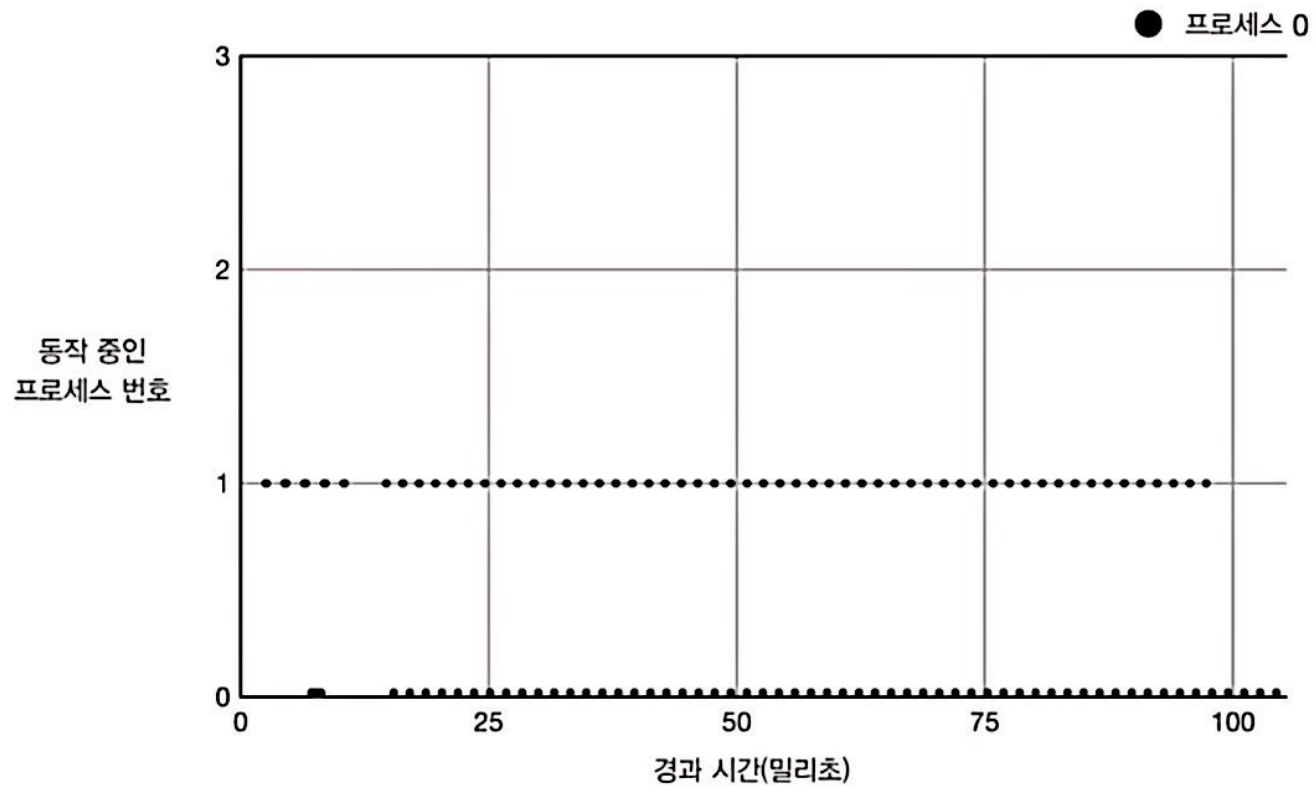
(실험 4-D, 그래프 ②)



process의 진행도이다. 논리 CPU가 1개일 때만 같다.
process 2개 일 때는.

그림 4-27 논리 CPU에 동작 중인 프로세스

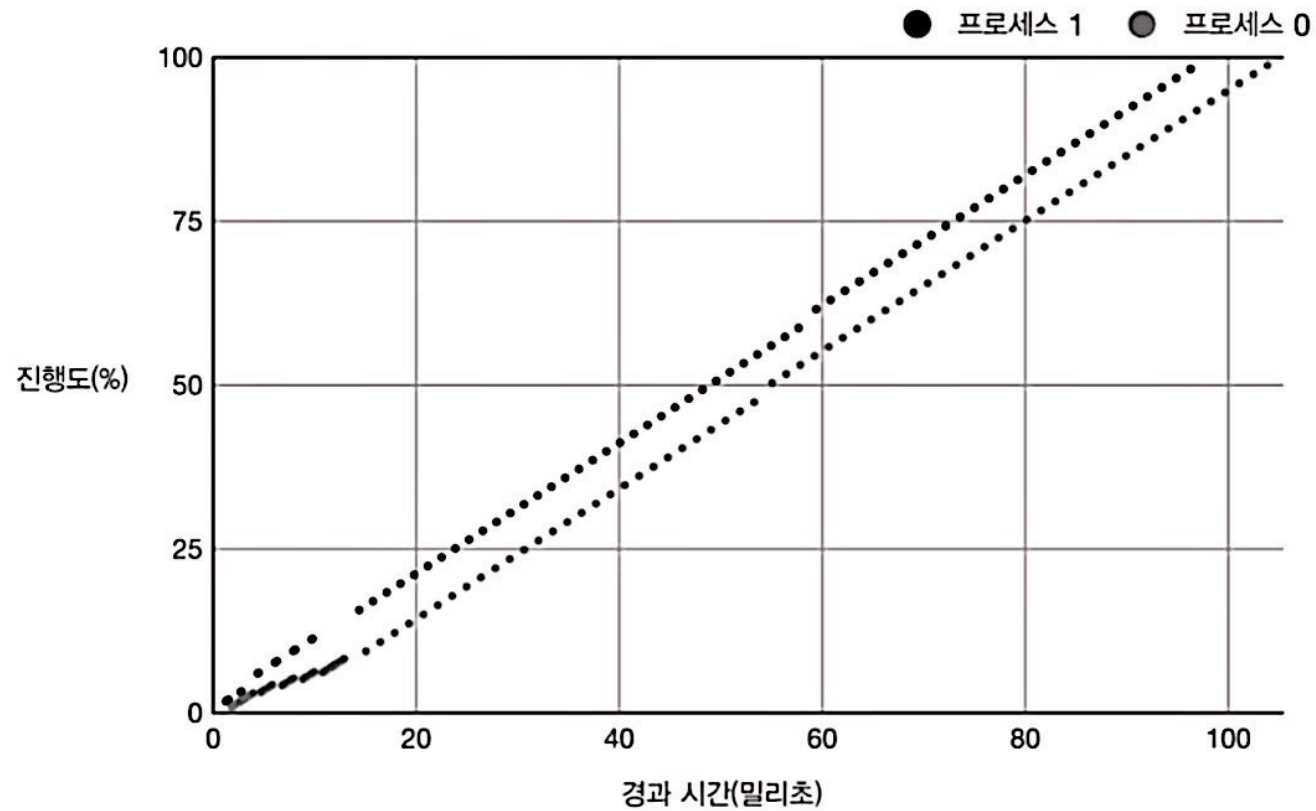
(실험 4-E, 그래프 ①)



이렇게 동작한다.
각각의 다스미 process 0 ~ process 10이 동시에 동작한다.
idle 상태의 다스미가 있으면 변신 리소스를 최대한 사용하고 있다.

그림 4-28 프로세스 0과 프로세스 1의 진행도

(실험 4-E, 그래프 ②)

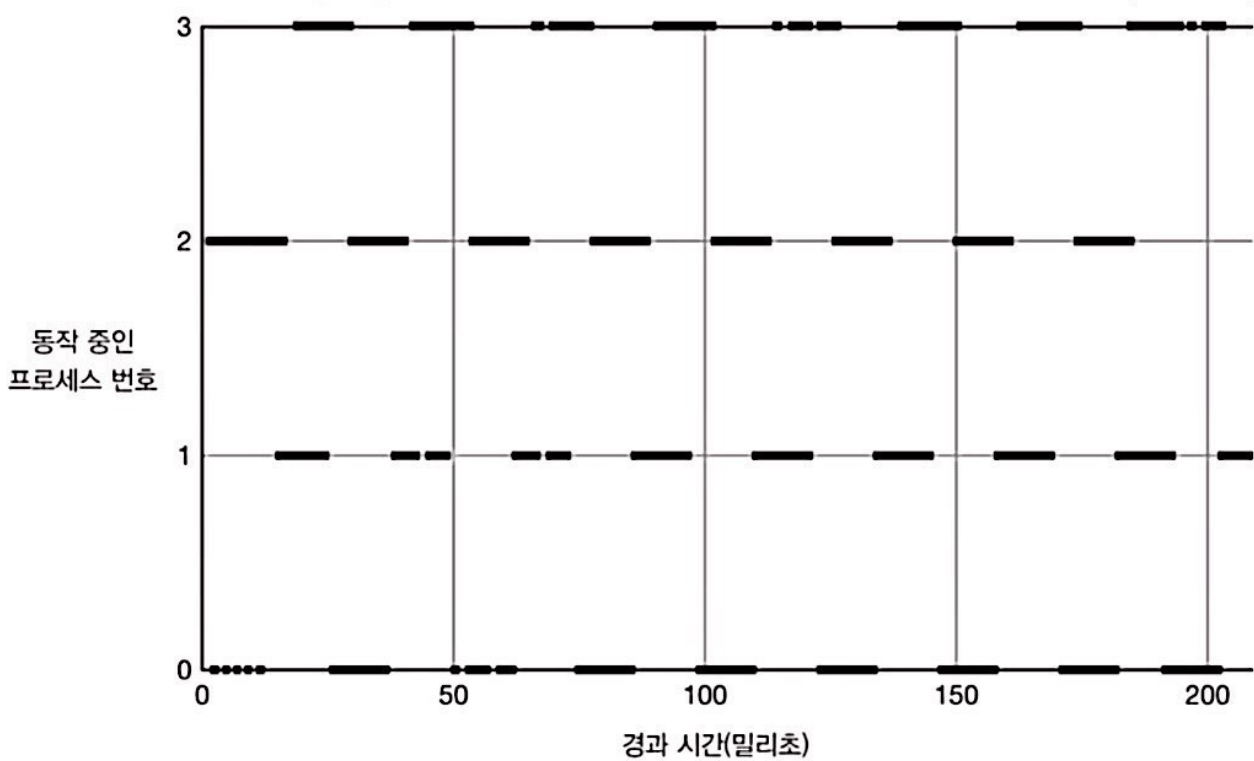


각 process 진행중이다. 각 process가 논리 CPU를 독점해서 동작한다.
그레이트론 논리 CPU가 1개인 경우 이외에 쉼터의 시리게 처리 안된다.

process가 4개인 경우는

그림 4-29 논리 CPU에 동작 중인 프로세스

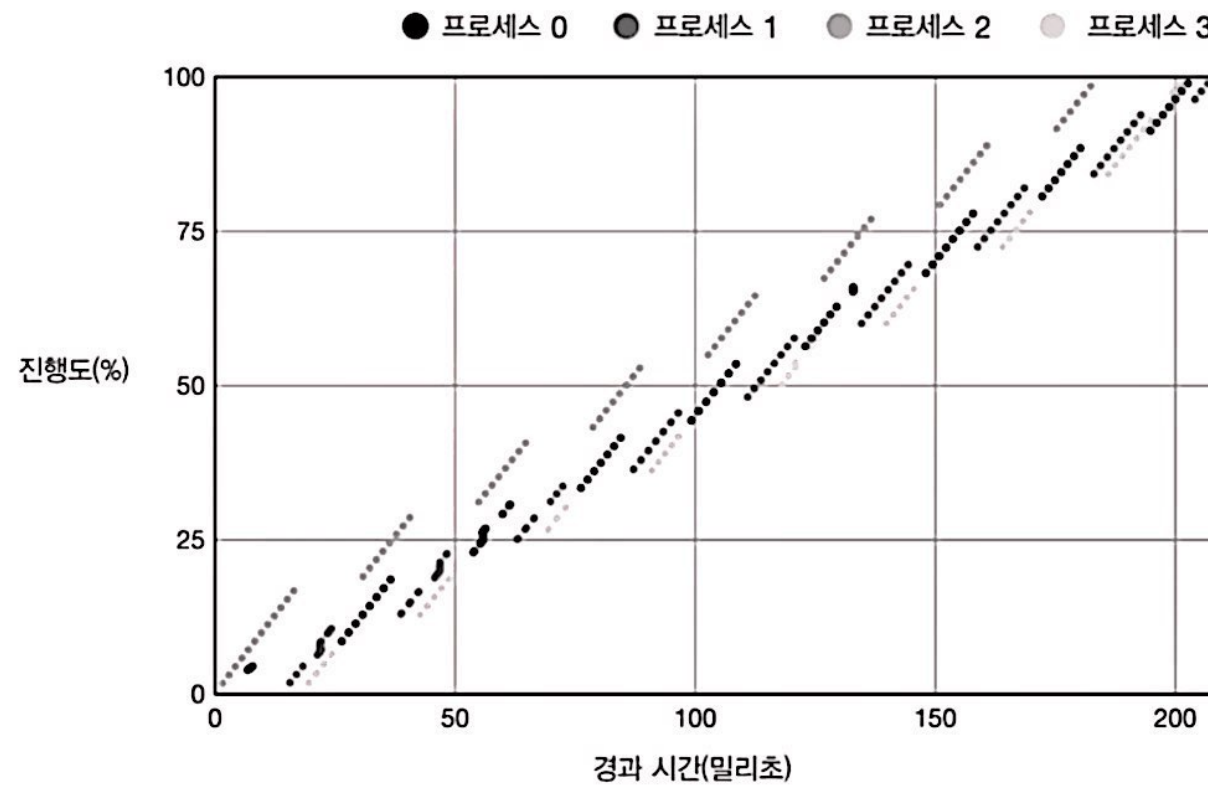
(실험 4-F, 그래프 ①)



2개의 동위 입자에 각각 2개의 process를 서로 붙여가며 동작하였다.

그림 4-30 프로세스 0~프로세스 3의 진행도

(실험 4-F, 그래프 ②)



process 실행도를 보면 각 process가 버퍼 통에 실행되고 있는 것을 알 수 있다.
최종적으로 각 process가 논리 CPU를 독점 가능했던 경우가 이하 소용시은 X2 관련.

각 슬롯에서 throughput, latency

→ 슬롯 4-D

$$\begin{aligned}\text{throughput} &\Rightarrow 1 \text{ process} / 100 [\text{ms}] \\ &= 1 \text{ process} / 0.1 [\text{cs}] \\ &= 10 \text{ process} / 1 [\text{cs}]\end{aligned}$$

$$\text{latency} \Rightarrow 100 [\text{ms}]$$

슬롯 4-E

$$\begin{aligned}\text{throughput} &\Rightarrow 2 \text{ process} / 100 [\text{ms}] \\ &= 2 \text{ process} / 0.1 [\text{cs}] \\ &= 20 \text{ process} / 1 [\text{cs}]\end{aligned}$$

$$\text{latency} \Rightarrow 100 [\text{ms}]$$

슬롯 4-F

$$\begin{aligned}\text{throughput} &\Rightarrow 4 \text{ process} / 200 [\text{ms}] \\ &= 4 \text{ process} / 0.2 [\text{cs}] \\ &= 20 \text{ process} / 1 [\text{cs}]\end{aligned}$$

$$\text{latency} \Rightarrow 200 [\text{ms}]$$

```
judith@judith-VirtualBox:~$ time taskset -c 0 ./sched 1 10000 10000
estimating the workload which takes just one milli-second...
end estimation
0          9463      100

real      0m10.954s
user      0m10.457s
sys       0m0.063s
```

```
$ time taskset -c 0 ./sched 2 10000 10000
```

```
1 19716 100
```

```
0 19732 100
```

```
real 0m21.487s
```

```
user 0m21.480s
```

```
sys 0m0.000s
```

```
$
```

처리시간과 사용시간이 거의 같다.

처리 시간을 빼면 경과 시간, 사용시간 둘다 거의 0이 된다.

(cpu 활용도가 높지 않다).

```
$ time taskset -c 0,1 ./sched 1 10000 10000  
0 9813 100
```

```
real 0m11.569s  
user 0m11.564s  
sys 0m0.000s
```

문리 답이 소제인 경우거나 잘못된 data가 있을.
C 프로그램으로 왼쪽 답은 계속 입력 받는다.

```
$ time taskset -c 0,1 ./sched 2 10000 10000
```

```
1 10326 100
```

```
0 10350 100
```

```
real 0m12.103s
```

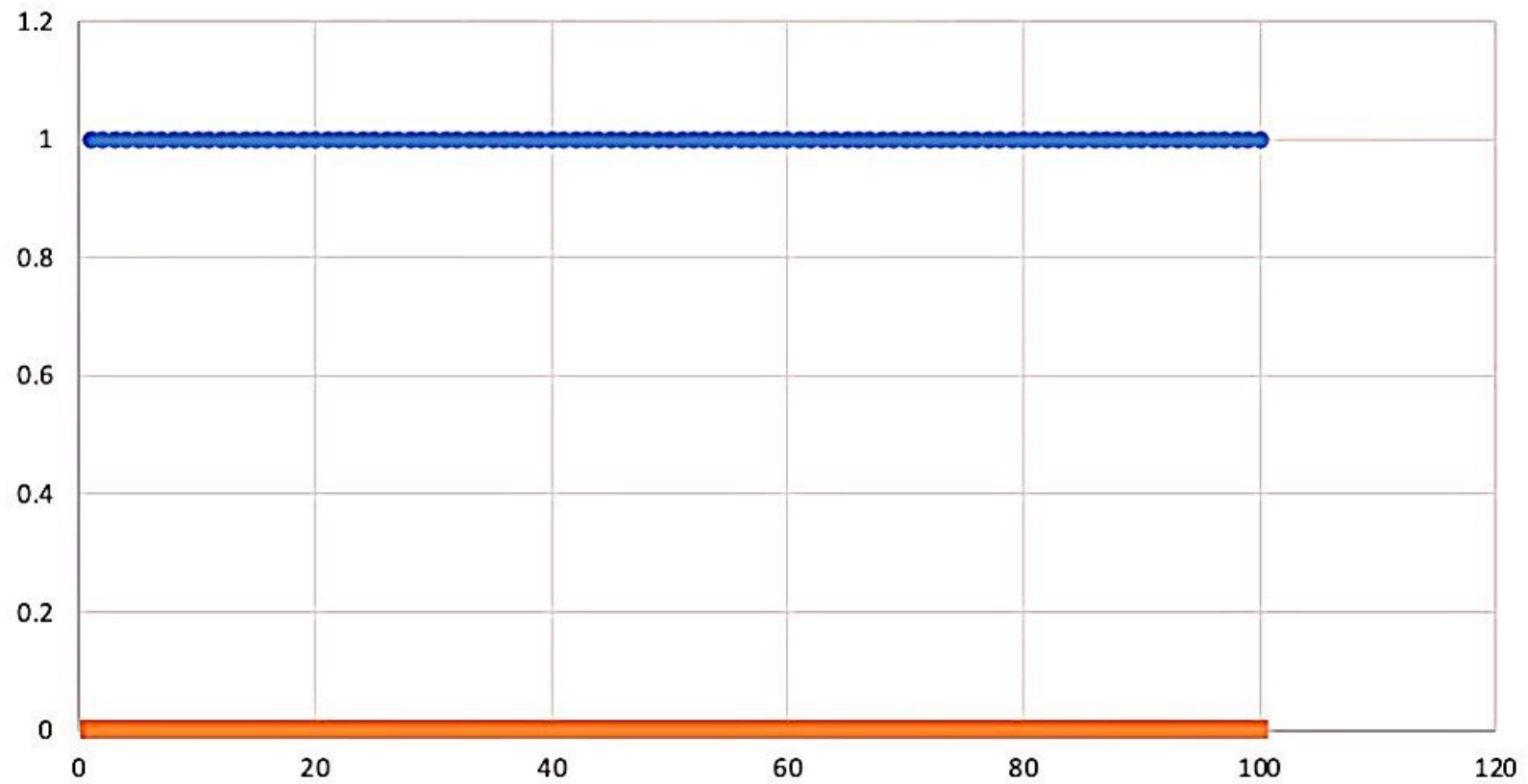
```
user 0m22.424s
```

```
sys 0m0.000s
```

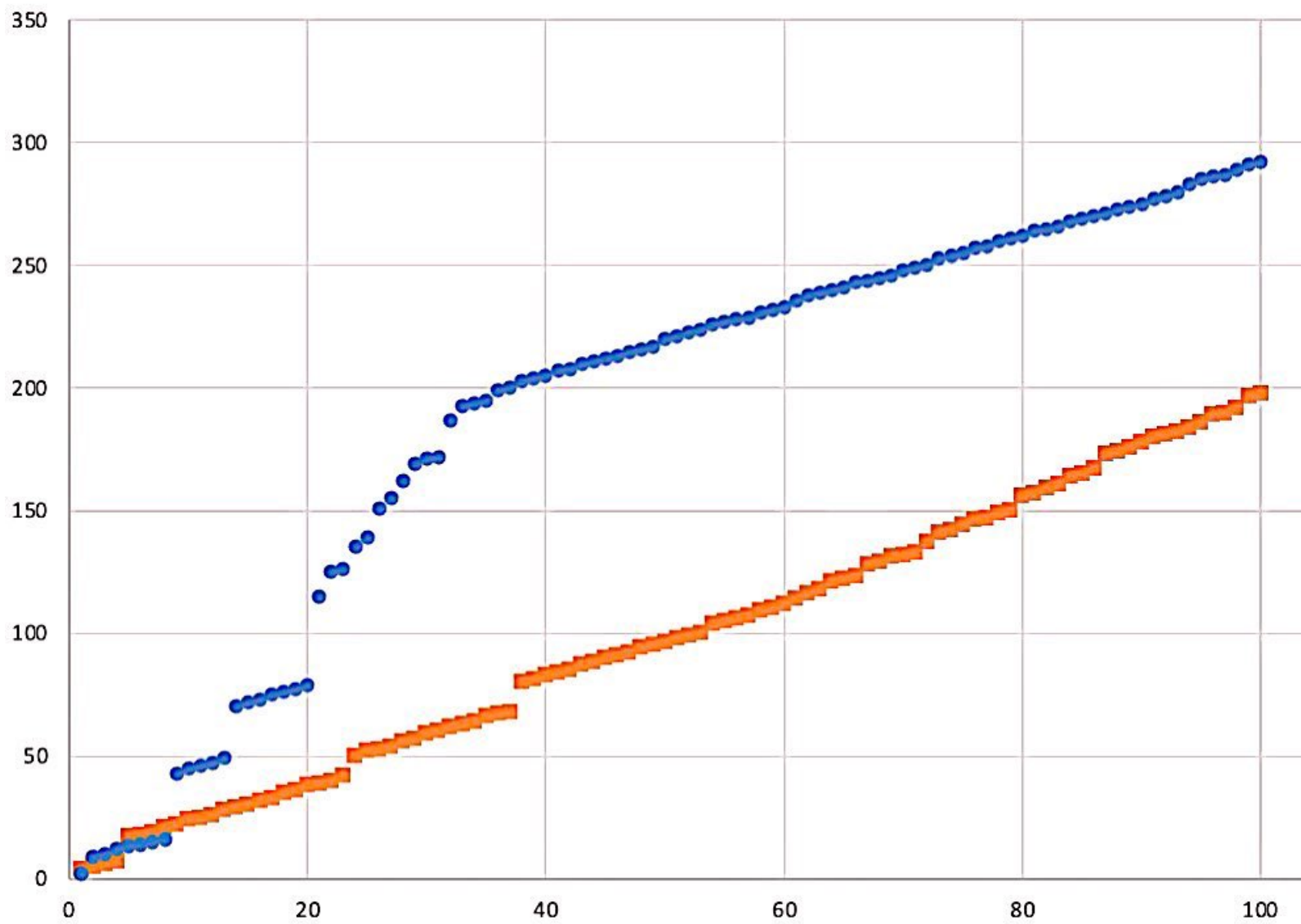

경라 시간은 $cpu=1$, $process=1$ 일때 n 이숫하지만 사용시간은 $2n$ 이.
C 따르면 cpu 가 $2n$ 이 $process$ 실행 가능한).

(3). nice ().

sched_nice process



너무 ~~출동~~해서 타는 줄 안나갈까 걱정은 우선위가 높은 process 이
process 1 이 비교해서 더 많은 cpu 시간 얻음.



process이 ~~무선적~~ 데이터를 먼저 종료했는지
C process은 그 다음에 종료됨?